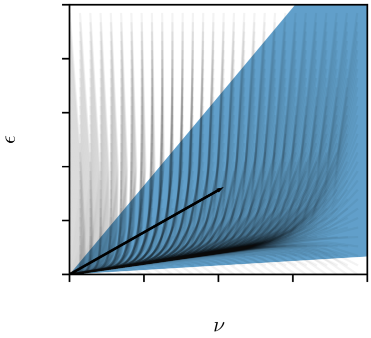
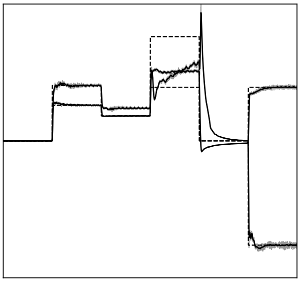
Albert-Ludwigs-Universität Freiburg
Technische Fakultät

Inexact methods for nonlinear model predictive control

Stability, applications, and software

Andrea Zanelli





University of Freiburg, Faculty of Engineering
Department of Microsystems Engineering
Systems Control and Optimization Laboratory

March 8, 2021

Inexact methods for nonlinear model predictive control

Stability, applications, and software

Andrea ZANELLI

Dean: Prof. Dr. Rolf Backofen

Examination committee:

Supervisor: Prof. Dr. Moritz Diehl
Second referee: Prof. Dr. James Rawlings
Observer: Prof. Dr. Lars Pastewka
Chair of committee: Prof. Dr. Joschka Boedecker

Dissertation zur Erlangung des Doktorgrades der Technischen Fakultät der Albert-Ludwigs-Universität Freiburg im Breisgau

March 8, 2021

© 2021 University of Freiburg – Faculty of Engineering
Self-published, Andrea Zanelli, Georges-Köhler-Allee 102, DE-79110 Freiburg i. Br. (Germany)

Alle Rechte vorbehalten. Alle Inhalte dieses Werkes, insbesondere Texte, Fotografien und Grafiken, sind urheberrechtlich geschützt. Das Urheberrecht liegt, soweit nicht ausdrücklich anders gekennzeichnet, bei der Albert-Ludwigs-Universität Freiburg.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from Albert-Ludwigs-Universität Freiburg.

Acknowledgement

This thesis is the outcome of a long and exciting journey which would have not been possible without the supervision, support and encouragement of a long list of people.

First and foremost I would like to thank Moritz Diehl for giving me the opportunity to pursue a PhD under his supervision in Freiburg. In these years, he taught me not only about numerical optimization and model predictive control, but also about mathematical thinking in general, scientific writing and beyond. His genius and contagious excitement for research have shaped the way I look at problems and I am truly honored to have had the chance to work in his lab.

I am extremely grateful to Quoc Tran-Dinh for introducing me to generalized equations and co-supervising with Moritz the work that led to one of the publications I am proud of the most. A considerable part of this thesis owes to Quoc's sharp comments, availability to discuss at any time, and encouragement. Similarly, Christoph Hackl has constantly supported and encouraged the project on reluctance synchronous machines and hosted me several times at the Technical University of Munich and the at the Munich University of Applied Sciences to carry out experiments. Working with him, Julian Kullick and Hisham Eldeeb has been incredibly exciting.

Taking a step back in time, I would like to thank Manfred Morari and Alexander Domahidi for sparking in me the fascination for model predictive control and embedded optimization during a class at ETH. That eventually led me to pursue an internship at ABB, under the supervision of Joachim Ferreau, Helfried Peyrl and Alessandro Zanarini and a master thesis at embotech under the supervision of Alexander Domahidi, Juan Jerez and Manfred Morari. Joachim first, and Alex and Juan later, played a fundamental role in the decision to pursue a PhD in numerical optimization.

I am thankful to Rien Quirynen for his initial mentorship when I joined SYSCOP and for always being available to discuss even my most bizarre ideas. Rien's ability to connect the dots and to reason about a broad range of topics has been a great source of inspiration. On top of this, he gets credit for having been, until he left the lab, the driving force behind social life outside the office. Even if I managed to digest only a fraction of his knowledge, Gianluca Frison has taught me lots of exciting concepts in high-performance computing and low-level programming. He also gets credit for creating `BLASFEO` and `HPIPM`, which constitute the backbone of `acados` (in general *giaf*'s hard-core programming skills deserve a special mention). Moreover, Gianluca was always there when I needed non strictly technical advice too. Per Rutquist, with whom I had the pleasure to share an office, should be mentioned for being the most versatile conversation partner. It's probably not entirely an exaggeration to say that Per can discuss anything ranging from optimal control to advanced programming and again from quantum physics to machine learning.

Although not strictly related to my PhD, working with Matilde Gargiani was a truly exciting and enriching experience. She taught me about numerical optimization for deep learning and large-scale optimization in general. Her love for the values of good science and determination to honor them will always be a source of inspiration.

I would like to thank Tommaso Sartor, Per Rutquist and Gianluca Frison for believing in `prometeo` since the beginning (even when I was uncertain myself that it would be a feasible project) and for all the insightful discussion about it. A big thank you also goes to the `acados` team which I had the pleasure to be part of. Working with Gianluca Frison, Dimitris Kouzoupis, Robin Verschueren and Jonathan Frey gave me the chance to become a much better programmer and to learn about the joy (and the struggles) of being part of a team of software developers.

I want to express my gratitude to Greg Horn for hosting me for two months at Kitty Hawk in Mountain View, California. Sitting next to his legendary setup with six screens and running flight tests was a rather unique experience. Although he left right before I joined the group, Mario Zanon has always encouraged me and gave me advice at various key stages of my PhD.

I would like to thank all the people that I have encountered at SYSCOP, for contributing to create a great atmosphere at the lab and a thriving scientific environment. These are, in sparse order: Rachel Leuthold, Dang Doan, Per Rutquist, Jörg Fischer, Gianluca Frison, Jonas Könemann, Giovanni Licitra, Dimitris Kouzoupis, Robin Verschueren, Rien Quirynen, Jonathan Frey, Florian Messerer, Katrin Baumgärtner, Fabian Girrbach, Jochem De Schutter, Tobias Schöls, Mikhail Katliar, Armin Nurkanovic, Adrian Bürger, Jonas Schlagenhauf, Jia-Jie Zhu, Yuning Jiang, Tommaso Sartor and Jesus Lago Garcia. I want to

further express my gratitude to our secretaries Christine Paasch, Gaby Kieninger and Kerstin Pfeiffer for their kindness, patience and help with administrative matters. Likewise, I am grateful to all the people I have encountered through AWESCO who shared with me the first three years of this journey.

A big thank you goes to all my friends around the world who helped me not to forget that there is much more to life than work. Finally, and most importantly, I want to say a special thank you to my girlfriend Matilde for the endless love, support and patience in these years. At the same time I am infinitely grateful to my parents Maurizio and Mariagrazia for teaching me to be curious and follow my passions as well as to my sister Serena, for always supporting me and always having been there when I most needed her.

Abstract

This thesis is about numerical methods for nonlinear model predictive control (NMPC) with a focus on system theoretic guarantees. Although considerable progress has been made since its early application in the late 1970s in the process industry, NMPC still requires a computational effort that is prohibitive for many applications due to fast dynamics or the low computational power available. For this reason, despite NMPC being nowadays the state-of-the-art control strategy in many applications, its applicability to a broader range of systems, still hinges on the development of efficient methods for numerical optimization.

In particular, we propose inexact methods that can speed up the computations associated with the solution of the underlying nonconvex programs. Although drawing from rather diverse areas, from an abstract point of view, such methods exploit a common idea. In fact, in many cases, carefully chosen perturbations to exact solutions and formulations do not jeopardize stability properties and can be leveraged to alleviate the computational burden of NMPC.

The algorithmic contributions of the thesis revolve around three main ideas. First, we analyze and extend a class of inexact methods, the so-called *real-time* methods. These strategies take inspiration from the popular real-time iteration (RTI) and rely on a limited number of iterations of the optimizer in order to compute an approximate solution to the underlying nonconvex programs. By warmstarting the iterates it is possible to “track” a parametric solution as the state of the system evolves. We prove asymptotic stability of the system-optimizer dynamics for a class of RTI-like methods that need not be based on standard sequential quadratic programming. This class of methods encompasses different strategies proposed in the thesis.

Second, we propose and analyze numerical methods that exploit inexact first- and second-order derivatives in order to efficiently compute suboptimal, but feasible solutions to nonconvex programs. These methods, which have strong ties with the celebrated second-order corrections in numerical optimization, can be used to reduce the computational cost associated with derivative computation and linear algebra. We show that the feedback policy resulting from such methods is locally stabilizing and enjoys favorable asymptotic suboptimality bounds.

The third class that we analyze is the one of *progressive tightening* methods. In this case, the main idea lies in exploiting stage-varying constraints and costs that give rise to increasingly conservative predictions the farther we look into the future. We prove asymptotic stability of progressive tightening NMPC and propose a numerical method that can exploit a specific barrier-based tightening in order to reduce the computational burden.

Finally, real-time variants of the second and third class of methods are proposed and analyzed from a system theoretic point of view and numerically validated.

In addition to the proposed algorithms and system theoretic results, we experimentally validate the classical RTI strategy as implemented in the software package `acados` on the task of controlling a reluctance synchronous machine. We present the experimental results of a current controller based on indirect NMPC running with a sampling time of 250 µs.

Lastly, motivated by the high degree of interaction between software components involved in the implementation of complex numerical strategies, we propose a high-level software framework for embedded high-performance computing. This framework, named `prometeo`, provides a Python-to-C transpiler and a static analysis tool that allows one to combine the performance of low-level languages (such as C) with the user-friendliness of high-level ones.

Kurze Zusammenfassung

In dieser Arbeit geht es um numerische Methoden für nichtlineare modellbasierte prädiktive Regelung (NMPC) mit Schwerpunkt auf systemtheoretischen Stabilitätsgarantien. Obwohl seit ihrer frühen Anwendung in der Prozessindustrie Ende der 1970er Jahre erhebliche Fortschritte erzielt wurden, erfordert NMPC immer noch einen Rechenaufwand, der für viele Anwendungen aufgrund der schnellen Dynamik oder der geringen verfügbaren Rechenleistung Prohibitiv ist. Obwohl NMPC heute in vielen Anwendungen die modernste Regelungsstrategie ist, hängt ihre Anwendbarkeit auf ein breiteres Spektrum von Systemen immer noch von der Entwicklung effizienter Methoden zur numerischen Optimierung ab.

Insbesondere schlagen wir ungenaue Methoden vor, welche die mit der Lösung der zugrundeliegenden nichtkonvexen Probleme verbundenen Berechnungen beschleunigen können. Obwohl die Ansätze aus recht unterschiedlichen Bereichen stammen, nutzen die Methoden aus abstrakter Sicht eine gemeinsame Idee. Tatsächlich gefährden in vielen Fällen sorgfältig ausgewählte Störeinflüsse auf exakte Lösungen und Formulierungen die Stabilitätseigenschaften nicht und können dazu genutzt werden, die rechnerische Belastung durch die prädiktive Steuerung nichtlinearer Modelle zu verringern.

Die algorithmischen Beiträge der Arbeit drehen sich um drei Hauptideen. Erstens analysieren und erweitern wir eine Klasse von ungenauen Methoden, die so genannten *Echtzeit*-Methoden. Diese Strategien lassen sich von der populären Echtzeit-Iteration (RTI) inspirieren und stützen sich auf eine begrenzte Anzahl von Iterationen des Optimierers, um eine approximative Lösung für die zugrundeliegenden nichtkonvexen Optimierungsprobleme zu berechnen. Durch günstige Initialisierungen (Warmstart) der Iterationen ist es möglich, eine parametrische Lösung zu “verfolgen”, während sich der Zustand des Systems entwickelt. Wir beweisen die asymptotische Stabilität der System-Optimierer-Dynamik für eine Klasse von RTI-ähnlichen Methoden, die nicht auf standardmäßiger sequentieller quadratischer Programmierung basieren müssen. Diese Klasse von Methoden umfasst verschiedene in dieser Dissertation vorgeschlagene Ansätze.

Zweitens präsentieren und analysieren wir numerische Methoden, die ungenaue Ableitungen erster und zweiter Ordnung nutzen, um suboptimale, aber zulässige Lösungen für nicht-konvexe Probleme effizient zu berechnen. Diese Methoden, die eng mit den klassischen Korrekturen zweiter Ordnung in der numerischen Optimierung verbunden sind, können zur Reduzierung der Rechenkomplexität für die Berechnung von Ableitungen und linearer Algebra eingesetzt werden. Wir zeigen, dass das Rückkopplungsgesetz, das sich aus solchen Methoden ergibt, lokal stabilisierend ist und sich günstiger asymptotischer Suboptimalitätsgrenzen erfreut.

Die dritte Klasse, die wir analysieren, ist die der *progressive tightening*-Methoden. In diesem Fall liegt die Hauptidee in der Ausnutzung stufenweise variierender Beschränkungen und Kosten, die zu immer konservativeren Vorhersagen führen, je weiter wir in die Zukunft blicken. Wir weisen die asymptotische Stabilität der progressive tightening NMPC nach und schlagen eine numerische Methode vor, die eine spezifische barrierenbasierte Verschärfung ausnutzen kann, um den Rechenaufwand zu verringern.

Schließlich werden Echtzeit-Varianten der zweiten und dritten Klasse von Methoden vorgeschlagen und aus systemtheoretischer Sicht analysiert und numerisch validiert.

Zusätzlich zu den vorgeschlagenen Algorithmen und systemtheoretischen Ergebnissen validieren wir experimentell die klassische RTI-Strategie, wie sie im Softwarepaket `acados` zur Aufgabe der Steuerung einer Reluktanzsynchronmaschine implementiert ist. Wir stellen die experimentellen Ergebnisse eines Stromreglers vor, der auf einem indirekten NMPC basiert und mit einer Abtastzeit von 250 µs läuft.

Abschließend präsentieren wir das High-Level-Software-Framework `prometeo` für eingebettete hocheffiziente Berechnungen. Diese Tool ist motiviert durch den hohen Grad Interaktion zwischen Softwarekomponenten, die an der Umsetzung komplexer numerischer Strategien beteiligt sind.

List of abbreviations

NMPC	nonlinear model predictive control
MPC	model predictive control
QP	quadratic program
NLP	nonlinear program
SQP	sequential quadratic programming
ADMM	alternating direction method of multipliers
RTI	real-time iteration
OCP	optimal control problem
ODE	ordinary differential equation
IVP	initial value problem
DAE	differential algebraic equation
LICQ	linear independence constraint qualification
SOSC	second-order sufficient conditions
FONC	first-order necessary conditions
KKT	Karush-Kuhn-Tucker
CPU	central processing unit
AST	abstract syntax tree
BLAS	basic linear algebra subroutines
PMSM	permanent magnet synchronous machine
RSM	reluctance synchronous machine

Notation

Throughout the thesis we will use the following notation.

$\mathbb{R}$	set of real numbers
$:=$	definition
$>, \geq, <, \leq$	for vectors $v \in \mathbb{R}^n$, inequalities are intended to be component-wise
$\mathbb{S}_+$	set of symmetric positive semidefinite matrices
$\mathbb{S}_{++}$	set of symmetric positive definite matrices
$\mathbb{R}^n_+$	set of vectors $\in \mathbb{R}^n$ with non-negative elements
$\mathbb{R}^{n\times m}_+$	set of matrices $\in \mathbb{R}^{n\times m}$ with non-negative elements
$\mathbb{R}^n_{++}$	set of vectors $\in \mathbb{R}^n$ with strictly positive elements
$\mathbb{R}^{n\times m}_{++}$	set of of matrices $\in \mathbb{R}^{n\times m}$ with strictly positive elements
$\mathbf{1}_n$	vector of ones in $\mathbb{R}^n$
$\mathbb{I}_n$	$n \times n$ identity matrix
(x, y)	(with $v \in \mathbb{R}^n$, $w \in \mathbb{R}^m$) stacked vector, i.e., $(v, w) := \begin{bmatrix} v \\ w \end{bmatrix}$
$\lVert v \rVert$	Euclidean norm of $v \in \mathbb{R}^n$ (or $\lVert v \rVert_2$ for improved readability)
$\lVert v \rVert_1$	one norm of $v \in \mathbb{R}^n$
$\lVert v \rVert_W$	(with $W \in \mathbb{S}_+$) weighted semi-norm of $v \in \mathbb{R}^n$, i.e., $\lVert v \rVert_W := \sqrt{v^\top W v}$
$\mathcal{L}_2$	space of square-integrable functions

$\Vert f\Vert_{\mathcal{L}_2}$	$\mathcal{L}_2$ norm of a function $f : [a,\, b] \to \mathbb{R}^n$ with $f \in \mathcal{L}_2$, i.e., $\Vert f\Vert := \left(\int_a^b \Vert f(x)\Vert^2 \mathrm{d}x\right)^{\frac{1}{2}}$
$\mathcal{B}(v, r)$	Euclidean ball of radius r centered at $v \in \mathbb{R}^n$, i.e., $\mathcal{B}(v, r) := \{w \in \mathbb{R}^n \,:\, \Vert w - v\Vert \leq r\}$
$\mathcal{B}_W(v, r)$	weighted norm ball of radius r centered at $v \in \mathbb{R}^n$, i.e., $\mathcal{B}_W(v, r) := \{w \in \mathbb{R}^n \,:\, \Vert w - v\Vert_W \leq r\}$
$\nabla_v f(u)$	gradient of the function $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $v \in \mathbb{R}^n$ at $u \in \mathbb{R}^n$, i.e., $\nabla_v f(u) := \frac{\partial f}{\partial v}(u)^\top$ $(\in \mathbb{R}^{n \times m})$
$\sigma_{\min}(Q)$	smallest eigenvalue of $Q \in \mathbb{S}_+$.
$\sigma_{\max}(Q)$	largest eigenvalue of $Q \in \mathbb{S}_+$.
$\rho(A)$	spectral radius of $A \in \mathbb{R}^{n \times n}$
$\mathcal{I}_\Omega(v)$	indicator function of the set $\Omega \subseteq \mathbb{R}^n$ at $v \in \mathbb{R}^n$, i.e., $\mathcal{I}_\Omega(v) := \begin{cases} 0, & \text{if} \quad v \in \Omega \\ \infty, & \text{otherwise} \end{cases}$
$\mathcal{N}_\Omega(v)$	normal cone to the set $\Omega \subseteq \mathbb{R}^n$ at $v \in \mathbb{R}^n$, i.e., $\mathcal{N}_\Omega(v) := \begin{cases} \{u \in \mathbb{R}^n : u^\top (v - w) \geq 0,\, \forall w \in \Omega\}, & \text{if } v \in \Omega, \\ \emptyset \ \text{otherwise.} \end{cases}$

Contents

Abstract v

Kurze Zusammenfassung vii

List of abbreviations ix

Notation xi

Contents xiii

List of Figures xv

List of Tables xxi

1 Introduction 1
1.1 Contributions and outline of the thesis 2

2 Background on numerical optimization and predictive control 7
2.1 Numerical optimization 7
2.2 Numerical optimal control and model predictive control 24

3 Asymptotic stability of the system-optimizer dynamics in NMPC 39
3.1 Contraction estimates for real-time methods 41
3.2 Asymptotic stability of the system-optimizer dynamics 57
3.3 Chapter summary and outlook 81

4 Zero-order methods for NMPC with stability guarantees 83
4.1 Zero-order SQP for NMPC 84
4.2 Asymptotic stability of zero-order NMPC 95
4.3 Implementation details and benchmarking 106
4.4 Chapter summary and outlook 114

5 Progressive tightening methods for NMPC with stability guarantees 117
5.1 Asymptotic stability of progressive tightening model predictive control . . . 118
5.2 A partial tightening real-time method . . . 122
5.3 Progressive tightening NMPC for attitude control of a quadcopter 141
5.4 Chapter summary and outlook . . . 152

6 Continuous control set NMPC of reluctance synchronous machines 155
6.1 Nonlinear model predictive control for electrical drives . . . 155
6.2 Simulation and experimental results . . . 171
6.3 Chapter summary and outlook . . . 175

7 `prometeo`: a domain specific language for embedded high-performance computing 177
7.1 Introducing `prometeo` . . . 177
7.2 Usage and performance . . . 190
7.3 Chapter summary and outlook . . . 199

8 Conclusions and outlook 203

A Appendix 209
A.1 Proof of Proposition 3.2.16 . . . 209
A.2 Proof of Theorem 3.2.25 . . . 211
A.3 Lyapunov function for the system-optimizer dynamics in error form . . . 212
A.4 Asymptotic stability of the system-optimizer dynamics for equality constrained NMPC . . . 214
A.5 A globalization heuristic for the feasible SQP strategy of Section 4.1 . . . 221
A.6 CS-NMPC for RSMs: additional results . . . 223

Curriculum vitae 239

Bibliography 239

List of publications 241

List of Figures

2.1 Example of an unconstrained optimization problem with a global strict nonisolated solution [Bertsekas, 1999, Example 1.1.11]. Although $\bar{y} = 0$ is the unique global minimizer, there are sequences of local minima that converge to $\bar{y}$ such that there does not exist a neighborhood of 0 inside which $\bar{y}$ is the only local solution. 9
2.2 Illustration of a single-valued localization. 14
2.3 Sparsity pattern of the Hessian of the Lagrangian for an NLP obtained with direct single shooting: in this case no sparsity can be exploited. 28
2.4 Sparsity pattern of the Hessian of the Lagrangian for an NLP obtained with direct multiple shooting: the matrix has a specific sparsity pattern that can be leveraged by tailored structure-exploiting solvers. 30

3.1 Coupled system-optimizer dynamics $\xi_+ = \Phi(T;\xi)$: when a limited number of iterations of the optimization algorithm are carried out in order to obtain an approximate solution that is then used to control the system, the system's and the optimizer's dynamics interact with one another. 40
3.2 Illustration of Assumption 3.1.2 and Lemma 3.1.5. Our setting does not require that $\bar{Z}$ is single-valued, but rather that there exists a single-valued and Lipschitz continuous localization $\bar{z}$ of the solution map. For sufficiently small changes in the parameter, any z within a ball of radius r_z centered at $\bar{z}(x)$ leads to a warmstart that is at most $\hat{r}_z$ "distant" from $\bar{z}(x_+)$ such that contraction (3.7) can be applied. 44

3.3 Comparison of closed-loop trajectories obtained with exact (thick, grey) and real-time (thin, black) feedback policies. A single iteration of RT-ADMM (real-time ADMM) is carried out per sampling time. Although a suboptimal solution is used to control the system, the closed-loop trajectories are very similar. 54
3.4 Numerical error incurred during the closed-loop simulation. The plot shows the Euclidean norm of the difference between the approximate primal solution w and the exact one $\bar{w}(x)$ during a part of the closed-loop simulation under consideration. Although the numerical error experiences peaks every time that the reference is changed, contraction is preserved and the suboptimality decreases with subsequent iterations until the next reference step. 55
3.5 Trajectories of the auxiliary dynamics (3.81) for different initial conditions ($\kappa = 0.4$, $\bar{a} = 0.5$, $\hat{\gamma} = 0.2$, $\hat{\mu} = 0.1$) - $T = 1.0$ (top) and $T = 0.4$ (bottom). The black vector describes the direction defined by $\hat{w}$ as in Theorem 3.2.27, while the shaded area defines the cone that contains all the vectors that would satisfy (3.89), i.e., all the vectors $\hat{w}$ that define a valid Lyapunov function $V_l(w) = \hat{w}^\top w$. 71
3.6 Illustrative example adapted from [Chen and Allgöwer, 1998] - closed-loop state trajectories obtained using the approximate feedback policy computed with a single iteration of a Gauss-Newton real-time algorithm (solid) and contour lines of $\frac{\mathrm{d}V(x(t))}{\mathrm{d}t}$ (dashed). 77
3.7 Illustrative example adapted from [Chen and Allgöwer, 1998] - although the numerical error does not necessarily decrease monotonically over time, the Lyapunov function for the combined system-optimizer dynamics $V_{\mathrm{so}}(\xi)$ does decrease over time. . . 78

4.1 Illustrative example: comparison of convergence rate of feasible SQP with iterated second-order corrections and Gauss-Newton on the nonlinear least squares problem (4.31) - Euclidean norm of the primal-dual steps. Convergence rate computed according to Theorem 4.1.7: $\kappa = 0.412$. "Empirical" convergence rate computed with linear regression: $\tilde{\kappa} = 0.408$. Gauss-Newton asymptotic convergence rate: 0.842. 93
4.2 Illustrative example: iterates in the primal space obtained with the proposed zero-order sequential programming algorithm and with Gauss-Newton. The iterates obtained with zero-order sequential programming are always feasible with respect to the nonlinear equality constraint. 94

4.3 ΔV for optimal value function (left) and $\tilde{V}(x)$ (right). Decreasing cost $\Delta V < 0$ in green, non-decreasing cost $\Delta V \geq 0$ in red, maximum number of iterations reached or infeasible SQP step in blue. The optimal value function is guaranteed to decrease by construction. For the inexact strategy, the cost can be non-decreasing due to the approximation introduced. However, $\Delta V < 0$ holds in a non-negligible region around the origin. 102
4.4 Relative suboptimality $\epsilon_{\%} = \frac{\tilde{V}(x)-V(x)}{V(x)} \cdot 100$. The inexact strategy gives rise to largely suboptimal policies in certain regions of the state space, however, as shown in Figure 4.3 stability is guaranteed in a non-negligible region of the state space. 103
4.5 Suboptimality as a function of the directional coordinate x_d. $\tilde{V}(x) - V(x)$ is compared with $\frac{1}{2}x^\top Qx$. Due to Lemma 4.2.6, the suboptimality is dominated by the quadratic stage cost. 103
4.6 Nonlinear hanging chain benchmark for $n_m = 5$ masses [Kouzoupis et al., 2018]. The dashed sketch describes the equilibrium at which the fixed quantities used by Algorithm 3 and 4 are computed. 110
4.7 Open-loop trajectories obtained with the original (solid), zero-order (dotted) and linear-quadratic (dashed) formulations. An additional constraint is added to the problem used in [Wirsching et al., 2006] that requires the position of the actuated mass to be within a ball. Although the zero-order strategy is clearly suboptimal, the obtained trajectories satisfy the nonlinear constraint (unlike with the linear-quadratic formulation). . . . 113
4.8 Asymptotic suboptimality of zero-order and linear-quadratic formulations in open-loop as a function of the deviation of x from the steady-state. The zero-order strategy provides a superior approximation of the optimal cost. 114

5.1 Simplified illustration of the progressive tightening Assumption 5.1.3 based on the cost-constraint epigraph inclusion. 119
5.2 Backward and forward Riccati recursion. The variables associated with stages M to N (in green) can be efficiently eliminated due to the (relaxed) linearized complementarity conditions. 131
5.3 Converged open-loop trajectories for $\tau = 0.1$. 138
5.4 Converged open-loop trajectories for $\tau = 1.0$. 139
5.5 Closed-loop trajectories obtained with the proposed partial tightening real-time iteration strategy for different number of iterations k per sampling time and $\tau = 10.0$. 140

5.6 Inverted pendulum swing-up: comparison of RTI strategies. Partial tightening RTI strategy (bold solid red) with $N = 100$, $M = 20$ and $\tau = 1$, standard formulation with $N = 100$ (dashed blue) and with $N = 50$ (dashed yellow). The partial tightening formulation stabilizes the system keeping the original constraints only in the initial 20 stages. The same does not hold for a standard formulation with a shortened horizon of $N = 50$ nodes. 141
5.7 Attitude tracking simulation results comparing different control strategies - closed-loop state trajectories: PD in solid yellow, LQR in dashed red and converged-NMPC in solid blue. The NMPC controller achieves a more accurate tracking of the reference attitude. 146
5.8 Attitude tracking simulation results comparing different control strategies - closed-loop input trajectories. PD in solid yellow, LQR in dashed red and converged-NMPC in solid blue. 148
5.9 Human-sized quadrotor equipped with a low-power Xilinx Zynq SoC with a dual-core ARM Cortex-A9 running at 800 MHz: snapshot from the experiment video (`https://www.youtube.com/watch?v=-dsezQa7nzk&feature=youtu.be`). 151
5.10 Experimental results - attitude in Euler angles. 152
5.11 Experimental results - actuators. 152
5.12 Experimental results - actuators (zoom in). 153

6.1 Nonlinear flux linkage of a real RSM obtained from FEM data $\hat{\Psi}_\mathrm{s}^\mathrm{d}$ (solid surface) and fitted grey box model $\Psi_\mathrm{s}^\mathrm{d}$ (dotted) - d-component. The worst-case relative error amounts to less than 10%. 157
6.2 Nonlinear flux linkage of a real RSM obtained from FEM data $\hat{\Psi}_\mathrm{s}^\mathrm{d}$ (solid surface) and fitted grey box model $\Psi_\mathrm{s}^\mathrm{d}$ (dotted) - q-component. The worst-case relative error amounts to less than 10%. 157
6.3 Voltage hexagon associated with the two-level VSI. 161
6.4 Laboratory setup including `dSPACE` real-time system, voltage-source inverters connected back-to-back, RSM, PMSM and torque sensor. 162
6.5 Control diagram: the MTPA LUTs provide the reference flux $\bar{\psi}$ and voltage $\bar{u}$ associated with a given reference torque $\bar{m}$. The NMPC controller computes the optimal control action based on the current state and disturbance estimate provided by an EKF. 164

6.6 Current steps at 157 rad s^{-1} (simulation): results obtained using the CS-NMPC (left) and the gain-scheduled PI controller (right). The voltage spherical constraints are directly included into the control formulation using the SCQP strategy proposed in [Verschueren et al., 2016]. Additionally, a “safety” polytopic constraint is included which, due to its linearity, is always satisfied exactly. 165

6.7 Current steps at 157 rad s^{-1} (simulation): results obtained using the CS-NMPC (left) and gain-scheduled PI controller (right). The CS-NMPC controller outperforms the PI controller, especially when the input constraints become active (e.g., between $t = 0.75$ s and $t = 1.00$ s). At the same time, a faster transient can be achieved even when the constraints become active only for a short time. 166

6.8 Current steps at 157 rad s^{-1} (experiment): two-norm of measured voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference. Notice that the input commanded by the PI controller remains saturated during the entire step. On the contrary, the CS-NMPC controller, after an initial saturation, steers the current to the (feasible) reference values. 167

6.9 Current steps at 157 rad s^{-1} (experiment): results obtained using the proposed CS-NMPC controller (left) and gain-scheduled PI controller (right). The CS-NMPC controller outperforms the PI controller, especially when the input constraints become active (e.g., between $t = 0.75$ s and $t = 1.00$ s). At the same time, as it can be seen especially between $t = 1.25$ s and $t = 1.50$ s, a faster transient can be achieved, even when the constraints are active only for a short time. 169

6.10 Current steps at 157 rad s^{-1} (experiment): two-norm of measured voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference. Notice that the input commanded by the PI controller remains saturated during the entire step. On the contrary, the CS-NMPC controller, after an initial saturation, steers the current to the (feasible) reference values. 170

6.11 Current steps at $157\,\mathrm{rad\,s^{-1}}$: overall control loop turnaround time (in black) obtained with the CS-NMPC controller using `acados` with `qpOASES` (available computation time of $250\,\mu\mathrm{s}$ in red). About 90% of the computation time is due the CS-NMPC controller (together with the EKF). 173

7.1 `prometeo`'s transpilation and worst case heap analysis. 181
7.2 AST associated with Listing 7.1 183
7.3 AST associated with Listing 7.3: typing information is added to the AST. 186
7.4 Call graph with memory-weighted edges. The shortest path from `main` to any node with the most negative cost determines the worst-case heap usage of the program. Only non negative cycles are allowed. 190
7.5 Computation time associated with a Riccati factorization for increasing matrix sizes. `prometeo`'s performance is almost identical to the one achieved with hand-coded C code. Although not embeddable, `Numpy` and `Julia` are included in the benchmark for reference. 200

A.1 Eigenvalues as a function of the sampling time T for problem (A.34). For sufficiently short sampling times, the auxiliary system in Definition A.4.6 is asymptotically stable. 219
A.2 Modulus of the eigenvalues associated with the system-optimizer dynamics and their trajectories in the complex plane. The plot reveals some conservatism of the analysis based on the auxiliary dynamics. In fact, the largest T for which the system-optimizer dynamics are stable is $T \approx 0.6$ as opposed to $T \approx 0.15$ obtained from Figure A.1. 220
A.3 Current steps at $165\,\mathrm{rad\,s^{-1}}$ (simulation): closed-loop trajectories obtained using CS-NMPC (left) and gain-scheduled PI controller (right). 224
A.4 Current steps at $165\,\mathrm{rad\,s^{-1}}$ (simulation): two-norm of voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference. 225
A.5 Current steps at $165\,\mathrm{rad\,s^{-1}}$ (experiment): closed-loop current trajectories obtained using the two controllers under analysis. Due to the strong effect of input saturation observed in simulation, for this value of the reference speed it was not possible to run the experiment with the PI controller. 226

List of Tables

3.1 Closed-loop relative suboptimality and average computation times in microseconds obtained with RT-ADMM-i, where i denotes the number of iterations per sampling time. Notice that, in order to compute a solution to the subproblems with a tolerance of 10^{-4}, about 30 ADMM iterations are needed, leading to an average computation time of 17 µs. 56

4.1 Closed-loop worst-case computation time, in milliseconds, for the standard (RTI) and zero-order (0-RTI) real-time iteration strategies for different prediction horizons N, numbers of masses n_m and numbers of stages for the collocation integrators n_s. In all simulations the system is steered to the steady state. Using the 0-RTI strategy, a maximum increase of less than 0.1% in the closed-loop cost is incurred with respect to the standard RTI. . 112

5.1 Pendulum example: worst-case computation times for the swing-up closed-loop scenario (1000 sampling steps) in milliseconds and closed-loop cost with $N = 100$, $\tau = 1$ and decreasing values of M (standard RTI with $N = 100$ in the first column). 142
5.2 Quadrotor model - values of the parameter used for the simulation results in Section 5.3.2. Notice that these values are fictitious. They have been used for simulation purpose and do not correspond to the parameters of the physical system. 144
5.3 Maximum and average CPU time and relative closed-loop suboptimality with respect to converged NMPC of the RTI and partial tightening RTI (pt-RTI) strategies. Using the pt-RTI strategy a speedup of about a factor 5 can be achieved with a moderate increase in suboptimality. 146

6.1 Parameters of physical setup (used in simulation too). 174

7.1 CPU time for the Fibonacci benchmark: `prometeo` is more than one order of magnitude faster than `Nuitka` and Python and about three times faster than PyPy. 200

Chapter 1

Introduction

Although several other definitions could be used, automatic control, from ancient Greek *autòmatos*, namely “acting of one’s own will”, can be described as the application of control theory for regulation of processes without direct human intervention. As such, according to Heraclitus and Democritus, it fits, as well as any other form of technology, the idea that it must deal with the creation of something, which philosophers would call *techne*, that imitates nature. The first control systems (the *techne* of automatic control) that we know of date back to 270 B.C, when Ctesibius in Ptolemaic Egypt described a floating regulator for a water clock. Since then, our ability to “develop devices that control other devices” hence helping us to delegate increasingly complex tasks to machines has greatly improved. From Ctesibius’s water clocks, through the early proportional-integral-derivative controller for ship steering by Minorsky, to controllers in today’s applications ranging from autonomous driving to aerospace, from power grid operation to applications in biomedics, we have improved our capacity to create devices that imitate, and sometimes surpass, humans’ ability to take decisions in order to achieve a certain goal (although many today call this artificial intelligence, rather than automatic control).

Model predictive control (MPC), which is the central topic of this thesis, is a particular form of automatic control that relies on the rather fundamental idea that, whenever possible, one should better use “a” model of the underlying process to be controlled. In fact such a model can be used to predict to a certain accuracy the system’s behavior for different actions, or policies, and pick the policy that better achieves whichever goal we are trying to attain. This process of optimization over predictions can be, with a moderate amount of bias, seen as an ubiquitous element in humans’ decision making process: most decisions that we make are based on a more or less accurate model of what they will lead to in the future.

More specifically, and more technically, in MPC, decisions are made by solving a series of (potentially) nonconvex programs, typically one at every *sampling instant*. The solutions to those programs provide state and input trajectories of the system to be controlled that minimize a certain loss function, while satisfying constraints that define regions of the input and state space the system must operate in. Although such a solution of mathematical programs can be in certain special cases carried out in closed form, this is in general a computationally demanding task. For this reason, MPC found its first applications in the late 1970s in the field of process control where the underlying time constants of the systems to be controlled give rise to sampling times sufficiently long to carry out the necessary computations. Since then, considerable technical advances have been made in the fields of computing, which, together with the development of increasingly efficient algorithms, have made MPC a viable control strategy in applications with much shorter sampling times: a control strategy that was initially applicable to systems with sampling times in the range of hours, is gradually becoming applicable in the milli- and microsecond timescale. Despite the great deal of progress made, MPC still attracts much research interest due to the many open challenges in meeting shorter and shorter sampling times and the many open system theoretic questions.

Among others, a way of addressing the challenge of reducing the computational footprint of MPC is the one of relying on inexact computations. In particular, in this thesis the term inexact will be used to refer to both the employment of computationally cheaper *inexact solutions* and the use of *inexact formulations*. As much as it sounds intuitive that inexact solutions and computations can be computationally cheaper than exact ones, it immediately becomes apparent that control strategies based on them might not enjoy the same properties, e.g., optimality, feasibility and stabilizing properties. This interplay between inexactness and satisfaction of desirable properties typically enjoyed by exact solutions to MPC formulations is the core topic of this thesis.

1.1 Contributions and outline of the thesis

In this thesis we investigate methods that rely on inexact solutions to nonconvex programs associated with MPC formulations or, similarly, on inexact MPC formulations. The thesis' structure as well as the contributions associated with each chapter are outlined below.

Chapter 2 - Background on numerical optimization and predictive control

This chapter introduces contents in the fields of numerical optimization, optimal control and system theory that lie the foundations for most of the contributions in the thesis. First, the theory of constrained optimization is reviewed and the fundamental results on sensitivity analysis, which will play an important role in much of the rest of the thesis, are reported. Second, the principal numerical approaches to solving optimal control problems are briefly discussed. Finally, standard stability results for model predictive control are reported.

Chapter 3 - Asymptotic stability of system-optimizer dynamics in NMPC

This chapter deals with the stability analysis of the system-optimizer dynamics in the context of NMPC. In particular, when inexact solutions to the underlying noncovenx programs are obtained through early termination, a nontrivial interaction between the system and the optimizer must be taken into account. The results presented in this chapter provide novel asymptotic stability results for the system-optimizer dynamics in a general setting. It is shown that, under mild assumptions, asymptotic stability can be recovered if the feedback policy is applied to the system with a sufficiently short sampling time and, in the most general setting, a Lyapunov function for the combined dynamics is constructed. The contents of the chapter are part of the publications [Zanelli et al., 2019a, Zanelli et al., 2021b, Zanelli et al., 2020].

Chapter 4 - Efficient zero-order methods for NMPC with stability guarantees

In this chapter, we regard a class of numerical methods, which we call *zero-order* methods, that make use of sequential quadratic programming (SQP)-type iterations that employ inexact first- and second-order information. The main idea - which originates in [Bock et al., 2007] and has strong connections with early work on second-order corrections in [Fletcher, 1982] and projection methods in [Sargent and Murtagh, 1973] - lies in solving a series of quadratic programs in which only the constraint *evaluations* are updated across iterations. With this simple, and yet powerful, algorithmic ingredient, it is possible to recover a feasible, but suboptimal solution. We specialize this concept into different numerical strategies for NMPC and study the error bound associated with the inexact solutions as well as stability properties of the feedback policy therefore obtained. A stability analysis of the closed-loop obtained with the inexact solutions is provided. Moreover, by applying the results of Chapter 3, it is shown how the real-time variant of zero-order NMPC leads to asymptotically stable system-optimizer dynamics. Finally, the algorithmic elements of zero-order NMPC are used to develop a feasible SQP strategy that exploits a Schur complement active-set strategy. The contents of the chapter are part of the publications [Zanelli et al., 2016, Zanelli et al., 2019a] as well as of the publication in preparation *"Zanelli, A., and Diehl, M. - A feasible sequential quadratic programming strategy with iterative second-order corrections"*.

Chapter 5 - Progressive tightening methods for NMPC with stability guarantees

We introduce a class of NMPC formulations, that we call *progressive tightening NMPC*. In particular, we deal with formulations with stage-varying costs and constraints that satisfy a tightening property and prove their stabilizing properties under standard assumptions. Loosely speaking, the underlying idea exploited in progressive tightening NMPC is that the optimal value function is a valid Lyapunov function for problems with stage-varying cost and constraints if increasingly pessimistic predictions are made. This amounts to the same point in the state-input space being associated with increasingly higher costs as the stage index increases. Since, the costs and constraints in progressive tightening NMPC are not time-varying, but rather stage-varying, classical results on NMPC with time-varying costs and constraints cannot be trivially applied.

In the second part of the chapter, a specific instance of progressive tightening NMPC, that we call *partial tightening NMPC* is proposed. In this approach, the prediction horizon is split into two sections. While in the initial section the original constraints are preserved, in the terminal section, the constraints are tightened using a barrier formulation. In this way, the numerical difficulties associated with the treatment of the nonsmooth complementarity manifold of the underlying mathematical programs can be partially mitigated. Due to the partially smoothened complementarity manifold, at each optimizer iteration, the variables associated with the terminal section can be efficiently eliminated. In this way, the computational burden of each iteration can be substantially reduced. Using the results of Chapter 3, it is shown that the real-time variant of partial tightening NMPC gives rise to asymptotically stable system-optimizer dynamics. Finally, the proposed strategy is experimentally validated on a quadcopter. The contents of this chapter are associated with the publications [Zanelli et al., 2017b, Zanelli et al., 2018].

Chapter 6 - Continuous control set nonlinear model predictive control of reluctance synchronous machines

This chapter presents an application of the real-time iteration, an algorithm belonging to the class of inexact strategies analyzed in Chapter 3, to the problem of controlling an electrical drive. A solver for nonconvex problems based on the software toolbox `acados` is deployed on an embedded control platform in order to control the currents (and indirectly the torque) of a reluctance synchronous machine. The approach chosen, that relies on an external modulator for the actual assignment of switching sequences, namely continuous set (or indirect) NMPC (CS-NMPC), is often regarded as computationally out of reach and it is largely unexplored. The simulation and experimental results discussed in this chapter show that a real-time implementation of CS-NMPC is both capable of achieving sampling times below 250 μs and largely outperforming state-of-the-art field oriented control strategies. The results presented in this chapter have been published in [Zanelli et al., 2021a].

Chapter 7 - `prometeo`: a domain specific language for embedded high-performance computing

A key aspect to the successful application of advanced control strategies, and NMPC in particular, to challenging problems is the availability of reliable and user-friendly software for numerical computing. In this chapter, we introduce `prometeo`, a domain specific language and Python-to-C transpiler whose aim is to propose a paradigm for the development of embedded high-performance software. Due to its transpiler, `prometeo` allows one to write high-performance code for embedded applications in a high-level language (a restricted version of the Python language) and to translate them into self-contained C code that can be easily deployed onto embedded platforms. Moreover, a particular structure on the programs is enforced such that, through static analysis, the worst-case memory usage can be determined. In this way, `prometeo`'s memory management system greatly simplifies dealing with otherwise error-prone and tedious memory book-keeping that is typically required in programs heavily relying on small scale computations such as NMPC. The contents of this chapter are based on the publication in preparation *"Zanelli, A., Sartor, T., Rutquist, P., Frison, G., Diehl, M. prometeo: a domain specific language and Python-to-C transpiler for embedded high-performance computing"*.

Chapter 8 - Conclusions and outlook

In this chapter, the contributions of the thesis are summarized and possible future research directions are proposed.

Chapter 2

Background on numerical optimization and predictive control

In this chapter, we review key concepts present in the literature of numerical optimization and predictive control that will be used throughout the thesis. Most of the results presented are well established ones and can be found in numerical optimization textbooks such as [Nocedal and Wright, 2006]. For what concerns results in the field of strongly regular generalized equations we will mostly refer to the seminal paper [Robinson, 1980] and the more recent book [Dontchev and Rockafellar, 2009]. Similarly, for what concerns the concepts on numerical optimal control and stability of model predictive control, we will refer to [Diehl and Gros, 2018] and [Rawlings et al., 2017], respectively.

2.1 Numerical optimization

Let us review some fundamental concepts in constrained optimization and numerical algorithms for the solution of nonconvex programs.

2.1.1 Theory of constrained optimization

Regard the following optimization problem:

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) = 0, \\ & h(y) \geq 0, \end{aligned} \tag{2.1}$$

where $y \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^q$ are twice continuously differentiable functions. We will refer to f as the *objective* function of (2.1), while we will call g and h its *equality* and *inequality* constraints, respectively.

Definition 2.1.1 (Feasible set)**.** *We define as feasible set of* (2.1) *the set of points in* $\mathbb{R}^n$ *that satisfy all the constraints:*

$$\Omega := \{y \in \mathbb{R}^n \,|\, g(y) = 0,\, h(y) \geq 0\}. \tag{2.2}$$

In the following, we define different types of solutions to (2.1).

Definition 2.1.2 (Local solution)**.** *A vector $\bar{y}$ is a local solution to* (2.1) *if it is a feasible point, i.e., $\bar{y} \in \Omega$, and there exists a neighborhood $\mathcal{M}$ of $\bar{y}$ such that $f(y) \geq f(\bar{y})$ for all $y \in \mathcal{M} \cap \Omega$.*

Definition 2.1.3 (Strict local solution)**.** *A vector $\bar{y}$ is a strict local solution (or strong local solution) to* (2.1) *if $\bar{y} \in \Omega$ and there exists a neighborhood $\mathcal{M}$ of $\bar{y}$ such that $f(y) > f(\bar{y})$ for any $y \in \mathcal{M} \cap \Omega \setminus \{\bar{y}\}$.*

Definition 2.1.4 (Isolated local solution)**.** *A vector $\bar{y}$ is an isolated local solution to* (2.1) *if $\bar{y} \in \Omega$ and there exists a neighborhood $\mathcal{M}$ of $\bar{y}$ such that $\bar{y}$ is the only local solution in $\mathcal{M} \cap \Omega$.*

Note that, although isolated local solutions are strict local solutions, strict local solutions are not necessarily isolated. The following example from [Bertsekas, 1999], and depicted in Figure 2.1, describes this fact.

Example 2.1.5. *Regard the following optimization problem:*

$$\min_{y} \quad f(y) = \begin{cases} y^2 \left(\sqrt{2} - \sin\left(\frac{4\pi}{3} - \sqrt{3}\log(y^2)\right)\right), & \textit{if } y \neq 0 \\ 0, & \textit{if } y = 0. \end{cases} \tag{2.3}$$

It is possible to show that $\bar{y} = 0$ is the unique global solution to (2.3)*, while, for any $k > 0$, $y_k = e^{\frac{(1-8k)\pi}{8\sqrt{3}}}$ is a local solution, such that there is a sequence $\{y_k\}$*

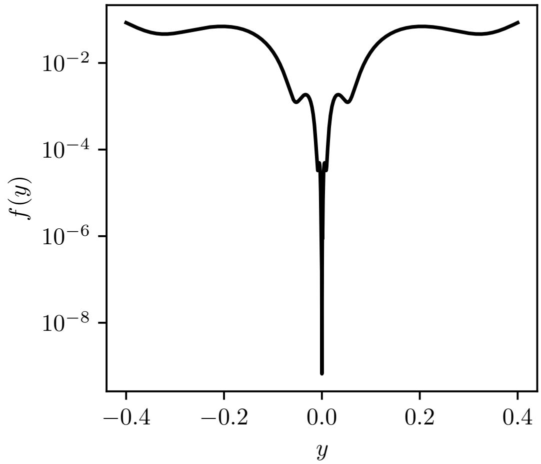


Figure 2.1: Example of an unconstrained optimization problem with a global strict nonisolated solution [Bertsekas, 1999, Example 1.1.11]. Although $\bar{y} = 0$ is the unique global minimizer, there are sequences of local minima that converge to $\bar{y}$ such that there does not exist a neighborhood of 0 inside which $\bar{y}$ is the only local solution.

of local minima that converges to $\bar{y} = 0$*. Due to the definition of a convergent series (topological version) we cannot define a neighborhood of* 0 *such that* $\bar{y}$ *is the only solution in it.*

First-order necessary conditions of optimality

In order to be able to define the so-called first- and second-order optimality conditions, we introduce the following definitions.

Definition 2.1.6 (Active set)**.** *The active set* $\mathcal{A}(y)$ *at a feasible point* y *is the set of all the indices associated with the active inequality constraints, i.e.,*

$$\mathcal{A}(y) := \{i \in \{1, \dots, q\} \,|\, h_i(y) = 0\}. \tag{2.4}$$

Notice that, unlike in the definition of active set in [Nocedal and Wright, 2006], $\mathcal{A}(y)$ *does not include the indices associated with the equality constraints.*

Definition 2.1.7 (Tangent vector)**.** *We call* d *a tangent vector to* Ω *at* y *if there exists a feasible sequence* $\{z_k\}$ *such that* $\lim_{k\to\infty}\{z_k\} = y$ *and a sequence* *of positive scalars* $\{t_k\}$ *such that*

$$\lim_{k\to\infty} \frac{z_k - y}{t_k} = d. \tag{2.5}$$

Definition 2.1.8 (Tangent cone)**.** *We call tangent cone to* Ω *at* $\bar{y}$ *the set of all tangent vectors to* Ω *at* $\bar{y}$ *and we denote such set by* $\mathcal{T}_\Omega(\bar{y})$*.*

Definition 2.1.9 (Linearized feasible cone)**.** *Given a feasible point* y*, the linearized feasible cone* $\mathcal{F}(y)$ *is defined as follows:*

$$\mathcal{F}(y) = \{d \,|\, d^\top \nabla g(y) = 0, d^\top \nabla h_i(y) \geq 0, \textit{ for all } i \in \mathcal{A}(y)\}. \tag{2.6}$$

Notice that the specification of the tangent cone is purely geometric and that $\mathcal{T}(y)$ can differ from $\mathcal{F}(y)$ whose specification is instead algebraic. Moreover, the two sets are not necessarily equivalent (see, e.g., [Nocedal and Wright, 2006, Example 12.5]). In order to make sure that the linearized feasible set, which has a much more practical specification than the tangent cone, (locally) captures the geometry of the feasible set, various types of *constraint qualifications* can be used. In this thesis, we will mostly rely on the following type of constraint qualification.

Definition 2.1.10 (Linear independence constraint qualification (LICQ))**.** *We say that linear independence constraint qualification holds at a feasible point* y*, if the vectors in the set* $\{\nabla g_i(y),$ *for* $i = 1, \ldots, m,$ $\nabla h_i(y),$ *for* $i \in \mathcal{A}(y)\}$ *are linearly independent.*

The condition involved in Definition 2.1.10 clearly fails to hold, e.g., at any point where two equivalent constraints (or two constraints whose linearization is geometrically equivalent) become active. Weaker constraint qualifications exist that hold even when LICQ does not. The interested reader is referred to [Nocedal and Wright, 2006] for further details. With the above definitions in place, we are ready to state the following first-order conditions for optimality of a point $\bar{y}$.

Theorem 2.1.11 (First-order necessary conditions (FONC))**.** *Let* $\bar{y}$ *be a local solution of* (2.1)*. Assume further that* f*,* g *and* h *are continuously differentiable and that LICQ holds at* $\bar{y}$*. Then there exist Lagrange multipliers* $\bar{\lambda} \in \mathbb{R}^m$ *and*

$\bar{\mu} \in \mathbb{R}^q$ *such that the following hold:*

$$\begin{aligned} \nabla_y f(\bar{y}) - \nabla_y g(\bar{y})\,\bar{\lambda} - \nabla_y h(\bar{y})\,\bar{\mu} &= 0, \\ g(\bar{y}) &= 0, \\ h(\bar{y}) &\geq 0, \\ \bar{\mu} &\geq 0, \\ \bar{\mu}_i h_i(\bar{y}) &= 0, \text{ for } i = 1, \dots, q. \end{aligned} \tag{2.7}$$

Remark 2.1.12. *We report a sketch of a proof of Theorem 2.1.11 in the following. A full proof can be found in [Nocedal and Wright, 2006].*

In order to show that, under the assumptions of the theorem, any minimizer $\bar{y}$ satisfies conditions (2.7)*, it is necessary to show that, i) if LICQ holds, then $\mathcal{F}(\bar{y}) = \mathcal{T}_\Omega(\bar{y})$, and ii) using Farkas' Lemma, show that either*

$$\nabla_y f(\bar{y}) = \nabla_y g(\bar{y})\,\bar{\lambda} + \sum_{i \in \mathcal{A}(\bar{y})} \nabla_y h_i(\bar{y})\,\bar{\mu}_i \tag{2.8}$$

with $\bar{\mu}_i \geq 0$ for $i \in \mathcal{A}(\bar{y})$, or there is a feasible direction $d \in \mathcal{F}(\bar{y})$ such that $d^\top \nabla_y f(\bar{y}) < 0$.

Conditions (2.7) are often referred to as *Karush-Kuhn-Tucker conditions* or *KKT conditions.*

Definition 2.1.13 (KKT point)**.** *We call a point $(\bar{y}, \bar{\lambda}, \bar{\mu})$ that satisfies conditions* (2.7) *and LICQ a KKT point.*

Definition 2.1.14 (Strongly and weakly active constraints)**.** *Given a KKT point $(\bar{y}, \bar{\lambda}, \bar{\mu})$, we say that a constraint h_i with $i \in \mathcal{A}(\bar{y})$ is strongly active if $\bar{\mu}_i > 0$. Otherwise, we say that it is weakly active.*

Definition 2.1.15 (Strict complementarity (SC))**.** *A point $(\bar{y}, \bar{\lambda}, \bar{\mu})$ where $\bar{y}$ is a local solution of* (2.1) *is said to satisfy strict complementarity if all active constraints are strongly active, i.e., $\bar{\mu}_i > 0$ for all $i \in \mathcal{A}(\bar{y})$.*

Second-order conditions of optimality

Theorem 2.1.11 shows that, under suitable differentiability assumptions, any local solution $\bar{y}$ to (2.1) is associated with properly defined Lagrange multipliers such that $(\bar{y}, \bar{\lambda}, \bar{\mu})$ is a KKT point. These conditions are merely necessary since first-order information cannot be used to determine the local behavior of the objective function for the directions $d \in \mathcal{F}(\bar{y})$ for which $d^\top \nabla_y f(\bar{y}) = 0$ (using the argument in the proof of Theorem 2.1.11 one can only conclude that, if conditions 2.7 are satisfied, then $d^\top \nabla_y f(\bar{y}) \geq 0$ holds for any feasible direction). For this reason, in order to develop sufficient conditions of optimality, it is necessary to characterize the behavior of the objective function in these "undecided" directions. The following definition defines the set of such directions.

Definition 2.1.16 (Critical cone). *Given a KKT point $(\bar{y}, \bar{\lambda}, \bar{\mu})$, we call critical cone the set*

$$\mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu}) := \{d \in \mathcal{F}(\bar{y}) \;:\; \nabla_y h_i(\bar{y})^\top d = 0, \, \forall i \in \mathcal{A}(\bar{y}) \, \textit{with} \, \bar{\mu}_i > 0\} \tag{2.9}$$

or, equivalently

$$d \in \mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu}) \iff \begin{cases} \nabla_y g(\bar{y})^\top d = 0, \\ \nabla_y h_i(\bar{y})^\top d = 0, \; \textit{for all } i \in \mathcal{A}(\bar{y}) \; \textit{with } \bar{\mu}_i > 0, \\ \nabla_y h_i(\bar{y})^\top d \geq 0, \; \textit{for all } i \in \mathcal{A}(\bar{y}) \; \textit{with } \bar{\mu}_i = 0. \end{cases} \tag{2.10}$$

The directions inside the critical cone are the ones for which first-order information is not enough to guarantee that a point is a local minimizer. In fact, since $\bar{\mu}_i = 0$ for all inactive constraints, the following holds:

$$\begin{aligned} d \in \mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu}) \implies \quad & \bar{\mu}_i \nabla_y h_i(\bar{y})^\top d = 0, \text{ for } i = 1, \dots, q, \\ & \bar{\lambda}_i \nabla_y g_i(\bar{y})^\top d = 0, \text{ for } i = 1, \dots, m. \end{aligned} \tag{2.11}$$

Using conditions (2.7), we can conclude that

$$d \in \mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu}) \implies d^\top \nabla_y f(\bar{y}) = \sum_{i=1}^{m} \bar{\lambda}_i \, d^\top \nabla_y g_i(\bar{y}) + \sum_{i=1}^{q} \bar{\mu}_i \, d^\top \nabla_y h_i(\bar{y}) = 0, \tag{2.12}$$

which shows that the critical cone contains feasible directions for which we cannot assess whether the objective increases or decreases from first-order information only. The following theorems provide conditions that exploit second-order derivatives in order to overcome this obstacle.

Definition 2.1.17 (Lagrangian function). *We call*

$$\mathcal{L}(y, \lambda, \mu) := f(y) - \lambda^\top g(y) - \mu^\top h(y) \tag{2.13}$$

the Lagrangian function of (2.1)*.*

Theorem 2.1.18 (Second-order necessary conditions (SONC)). *Let $\bar{y}$ be a local solution to* (2.1) *and let $(\bar{y}, \bar{\lambda}, \bar{\mu})$ be a KKT point. Assume that LICQ holds and that f, g and h are twice continuously differentiable. Then, for all $d \in \mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu})$, the following holds:*

$$d^\top \nabla_{yy}^2 \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\mu}) d \geq 0. \tag{2.14}$$

Theorem 2.1.19 (Second-order sufficient conditions (SOSC))**.** *Let* $(\bar{y}, \bar{\lambda}, \bar{\mu})$ *be a KKT point. Assume that* f*,* g *and* h *are twice continuously differentiable and that, for all* $d \in \mathcal{C}(\bar{y}, \bar{\lambda}, \bar{\mu})$*, with* $d \neq 0$*, the following holds:*

$$d^\top \nabla^2_{yy} \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\mu}) d > 0. \tag{2.15}$$

Then $\bar{y}$ *is a local minimizer for* (2.1)*.*

Perturbation analysis and stability

Throughout the thesis we will often use concepts from perturbation analysis of parametric nonlinear programs to determine local continuity properties and compute sensitivities of parametric solutions. In the following, we recall some of the fundamental concepts in the field of perturbation analysis for nonlinear programming. First, regard the simplified setting in which problem (2.1) does not have inequality constraints:

$$P_0(\varepsilon): \quad \begin{aligned} \min_{y} \quad & f(y, \varepsilon) \\ \text{s.t.} \quad & g(y, \varepsilon) = 0, \end{aligned} \tag{2.16}$$

where we have introduced the dependency of both the objective and constraints on the parameter $\varepsilon \in \mathbb{R}^r$ and where we denote by $\bar{y}(\varepsilon) : \mathbb{R}^r \to \mathbb{R}^n$ the (potentially set-valued) solution map. In this setting, under suitable regularity assumptions, it is possible to determine well-definedness and continuity of the solution map $\bar{y}(\varepsilon)$ in a neighborhood of a solution $\bar{y}(0)$ associated with $\varepsilon = 0$ using the implicit function (or Dini's) theorem. For the sake of completeness, we report Dini's theorem (as stated by Dontchev and Rockafellar in [Dontchev and Rockafellar, 2009]) in the following. To this end, it becomes useful to introduce the concepts of graphical localization and single-valued localization of a map.

Definition 2.1.20 (Graphical localization)**.** *Let* $\Psi : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ *be a set-valued mapping and let* $(\bar{u}, \bar{v}) \in \textbf{gph } \Psi$*. A graphical localization of* Ψ *at* $\bar{u}$ *for* $\bar{v}$ *is a (potentially set-valued) mapping* $\tilde{\Psi}$ *such that*

$$\textbf{gph } \tilde{\Psi} = (U \times V) \cap \textbf{gph } \Psi \tag{2.17}$$

for some neighborhoods U *of* $\bar{u}$ *and* V *of* $\bar{v}$ *so that*

$$\tilde{\Psi}(u) = \begin{cases} \Psi(u) \cap V, & \text{when}\, u \in U, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.18}$$

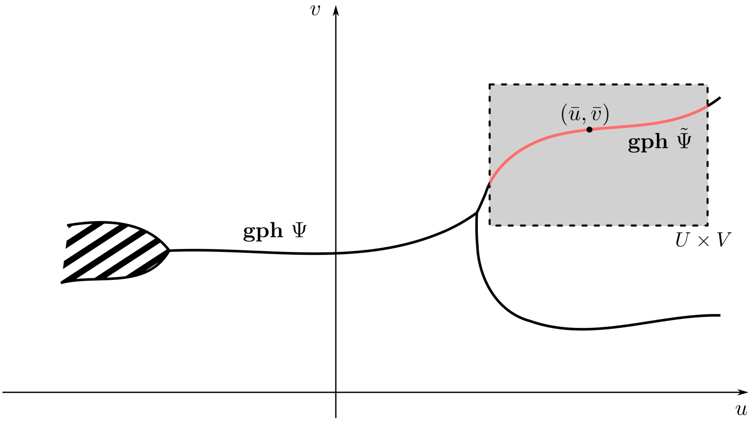


Figure 2.2: Illustration of a single-valued localization.

The inverse of $\tilde{\Psi}$ is defined by

$$\tilde{\Psi}^{-1}(v) = \begin{cases} \Psi^{-1}(v) \cap U, & \text{when}\, v \in V, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.19}$$

and is thus a graphical localization of the set-valued mapping Ψ^{-1} at $\bar{v}$ for $\bar{u}$.

In particular, we will be interested in situations in which solution maps admit graphical localizations that are single-valued.

Definition 2.1.21 (Single-valued localizations)**.** *By a single-valued localization of Ψ at $\bar{u}$ for $\bar{v}$ will be meant a graphical localization that is a function, its domain not necessarily being a neighborhood of $\bar{u}$. In case its domain is indeed a neighborhood of $\bar{u}$ we will say that it is a single-valued localization around $\bar{u}$ for $\bar{v}$.*

Figure 2.2 describes the concept of single-valued localization.

Theorem 2.1.22 (Implicit function or Dini's theorem)**.** *Regard the nonlinear root-finding problem*

$$\phi(z, \varepsilon) = 0, \tag{2.20}$$

where $\phi : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is a continuously differentiable function in a neighborhood of $(\hat{z}, 0)$ and such that $\phi(\hat{z}, 0) = 0$. Let the partial Jacobian of ϕ with respect to z at $(\hat{z}, 0)$, namely $\nabla_z \phi(\hat{z}, 0)^\top$, be nonsingular. Then the solution mapping

$$S : \varepsilon \to \{z \in \mathbb{R}^n \,|\, \phi(z, \varepsilon) = 0\}, \quad \textit{for}\ \varepsilon \in \mathbb{R}^r, \tag{2.21}$$

has a single-valued localization s around $\varepsilon = 0$ for $\hat{z}$ which is continuously differentiable in a neighborhood $\mathcal{M}$ of $\varepsilon = 0$ with Jacobian satisfying

$$\nabla_\varepsilon s(\varepsilon)^\top = -\nabla_z \phi(s(\varepsilon), \varepsilon)^{-\top} \nabla_\varepsilon \phi(s(\varepsilon), \varepsilon)^\top, \quad \text{for all } \varepsilon \in \mathcal{M}. \tag{2.22}$$

Applying Dini's theorem to the first-order optimality conditions of (2.16), together with some additional assumptions, we can prove the following classical result on perturbation analysis for equality constrained nonlinear programming.

Theorem 2.1.23. *Let $\bar{y}$ be a solution to $P_0(0)$ at which LICQ and SOSC hold with associated Lagrange multiplier $\bar{\lambda}$. Then there exist neighborhoods $\mathcal{M}_z$ of $\bar{z} := (\bar{y}, \bar{\lambda})$ and $\mathcal{M}_\varepsilon$ of $\varepsilon = 0$ such that the solution map $\bar{z}(\varepsilon) := (\bar{y}(\varepsilon), \bar{\lambda}(\varepsilon))$ is single-valued in $\mathcal{M}_z$ and Lipschitz continuous over $\mathcal{M}_\varepsilon$.*

Proof. Let $z := (y, \lambda)$. The result is a direct consequence of the IFT since LICQ and SOSC imply that the Jacobian of the Lagrangian $\nabla_z \mathcal{L}(z, 0)$ is nonsingular (see, e.g. [Nocedal and Wright, 2006, Lemma 16.1]). ☐

This result on parametric sensitivities of equality constrained nonlinear programs can be extended to the inequality constrained setting

$$P_1(\varepsilon): \qquad \begin{aligned} \min_y \quad & f(y, \varepsilon) \\ \text{s.t.} \quad & g(y, \varepsilon) = 0, \\ & h(y, \varepsilon) \geq 0, \end{aligned} \tag{2.23}$$

with the additional assumption that the inequality constraints locally behave as equality constraints under small perturbations of the nonlinear program.

Theorem 2.1.24 (Theorem 2.4.4, [Fiacco, 1983])**.** *Let $(\bar{y}, \bar{\lambda}, \bar{\mu})$ be a local solution to $P_1(0)$ and assume that the following hold:*

1. *The functions f, g and h are $\mathcal{C}^2$ in both y and ε.*
2. *SOSC holds at $(\bar{y}, \bar{\lambda}, \bar{\mu})$.*
3. *LICQ holds at $\bar{y}$.*
4. *SC holds, i.e., $\bar{\mu}_i > 0$, for all $i \in \mathcal{A}(\bar{y})$.*

Then the following hold:

1. *The point $\bar{y}$ is a local isolated minimizer for $P_1(0)$ and $\bar{\lambda}$ and $\bar{\mu}$ are the unique Lagrange multipliers associated with $\bar{y}$.*

2. *For ε near* 0*, there exists a unique $\mathcal{C}^1$ function $\bar{z}(\varepsilon) = (\bar{y}(\varepsilon), \bar{\lambda}(\varepsilon), \bar{\mu}(\varepsilon))$ with $\bar{z}(0) = (\bar{y}, \bar{\lambda}, \bar{\mu})$, such that $\bar{y}(\varepsilon)$ is an isolated local minimizer for $P_1(\varepsilon)$ with associated unique Lagrange multipliers $\bar{\lambda}(\varepsilon)$ and $\bar{\mu}(\varepsilon)$.*

3. *For ε near 0, the gradients of the binding constraints are linearly independent and SC holds for $\bar{y}(\varepsilon)$ and $\bar{\mu}(\varepsilon)$.*

Finally, under a slightly more restrictive assumption on curvature properties, the following theorem drops the assumption of strict complementarity allowing for weakly active constraints at the unperturbed solution. In particular, we will rely on the following stronger version of the second-order sufficient conditions of optimality.

Definition 2.1.25 (Strong second-order sufficient conditions (SSOSC))**.** *Let $(\bar{y}, \bar{\lambda}, \bar{\mu})$ be a KKT point for $P_1(0)$ and let $\mathcal{C}_s(\bar{y}, \bar{\lambda}, \bar{\mu})$ be defined as follows:*

$$d \in \mathcal{C}_s(\bar{y}, \bar{\lambda}, \bar{\mu}) \iff \begin{cases} \nabla_y g(\bar{y})^\top d = 0, \\ \nabla_y h_i(\bar{y})^\top d = 0, \ \textit{for all } i \in \mathcal{A}(\bar{y}) \textit{ with } \bar{\mu}_i > 0. \end{cases} \tag{2.24}$$

We say that strong second-order sufficient conditions hold at $(\bar{y}, \bar{\lambda}, \bar{\mu})$ if, for all $d \in \mathcal{C}_s(\bar{y}, \bar{\lambda}, \bar{\mu})$, with $d \neq 0$, the following holds:

$$d^\top \nabla^2_{yy} \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\mu}) d > 0. \tag{2.25}$$

Theorem 2.1.26 (Theorem 2.4.5, [Fiacco, 1983])**.** *Let $(\bar{y}, \bar{\lambda}, \bar{\mu})$ be a KKT point for $P_1(0)$ and assume that the following hold:*

1. *The functions f, g and h are $\mathcal{C}^2$ in both and y and ε in a neighborhood of $(\bar{y}, 0)$.*
2. *SSOSC holds at $(\bar{y}, \bar{\lambda}, \bar{\mu})$.*
3. *LICQ holds at $\bar{y}$.*

Then the following hold:

1. *The point $\bar{y}$ is a local isolated minimizer for $P_1(0)$ and $\bar{\lambda}$ and $\bar{\mu}$ are the unique Lagrange multipliers associated with $\bar{y}$.*
2. *For ε near* 0*, there exists a unique $\mathcal{C}^1$ function $\bar{z}(\varepsilon) = (\bar{y}(\varepsilon), \bar{\lambda}(\varepsilon), \bar{\mu}(\varepsilon))$ with $\bar{z}(0) = (\bar{y}, \bar{\lambda}, \bar{\mu})$ satisfying SSOSC for $P_1(\varepsilon)$ such that $\bar{y}(\varepsilon)$ is an isolated local minimizer for $P_1(\varepsilon)$ with associated unique Lagrange multipliers $\bar{\lambda}(\varepsilon)$ and $\bar{\mu}(\varepsilon)$.*

3. *For ε near 0, the gradients of the binding constraints are linearly independent for $\bar{y}(\varepsilon)$.*

4. *There exist $0 < \alpha, \beta, \gamma < \infty$ and $\delta > 0$ such that, for any ε with $\|\varepsilon\| < \delta$, the following hold:*
$$\|\bar{y}(\varepsilon) - \bar{y}\| \leq \alpha\|\varepsilon\|, \tag{2.26}$$
$$\|\bar{\lambda}(\varepsilon) - \bar{\lambda}\| \leq \beta\|\varepsilon\| \tag{2.27}$$
and
$$\|\bar{\mu}(\varepsilon) - \bar{\mu}\| \leq \gamma\|\varepsilon\|. \tag{2.28}$$

5. *The function $\bar{f}(\varepsilon) := f(\bar{y}(\varepsilon), \varepsilon)$ is continuously differentiable with respect to ε near $\varepsilon = 0$.*

6. *The directional derivative of $\bar{z}(\varepsilon)$ exists in any direction near $\varepsilon = 0$.*

Theorem 2.1.26 shows that under the assumption that SSOSC holds, we can guarantee that the localization of the parametric solution map $\bar{z}(\varepsilon)$ is locally Lipschitz. As we will see in the next section, SSOSC is a sufficient condition for a more general property called *strong regularity.*

Example 2.1.27 (Nondifferentiable Lipschitz continuous optimal value)**.** *Regard the following parametric linear program:*

$$\begin{aligned} \min_{x,t \in \mathbb{R}} \quad & t \\ s.t. \quad & x = \varepsilon, \\ & t - x \geq 0, \\ & x + t \geq 0. \end{aligned} \tag{2.29}$$

We can easily verify that the parametric solution to (2.29) *is*

$$\bar{x}(\varepsilon) = \varepsilon, \quad \bar{t}(\varepsilon) = |\varepsilon|, \tag{2.30}$$

which is globally Lipschitz in ε with Lipschitz constant 1*. However, the parametric optimal cost $\bar{f}(\varepsilon) = \bar{t}(\varepsilon)$ is not differentiable at $\varepsilon = 0$. In fact the assumption that LICQ holds made in Theorem 2.1.26 is violated at $\bar{y}(0) = (\bar{x}(0), \bar{t}(0)) = (0, 0)$ since the Jacobian of the binding constraints reads:*

$$J = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \tag{2.31}$$

which is obviously rank deficient. At the same time, we can easily verify that the dual solution for $\varepsilon = 0$ is indeed not a singleton either by inspection or using the so called strict Mangasarian Fromovitz constraint qualification.

2.1.2 Strongly regular generalized equations

The well established results reported in Section 2.1.1 provide extremely useful tools to analyze the local behavior of the solutions to parametric nonlinear programs under small perturbations. In this section, we introduce the mathematical framework of generalized equations, which can be used to derive a generalization of Theorem 2.1.26. The type of results that can be obtained with such a framework are more general because *i)* they apply to a broader class of problems - namely generalized equations - rather than parametric programs only *ii)* the assumptions made to obtain them are somewhat less stringent than the one of Theorem 2.1.26: although, for parametric nonlinear programs, we can obtain results similar to the ones in Theorem 2.1.26, it is not necessary to assume SSOSC and LICQ, but rather certain Lipschitz continuity properties of the solution to a properly defined parametric linearized problem.

According to the definition in the seminal paper [Robinson, 1980], a generalized equation is an inclusion of the form

$$0 \in F(z) + \mathcal{N}_K(z), \tag{2.32}$$

where F is a function from a subset Ξ of a normed linear space Z to its topological dual Z', K is a nonempty closed and convex set and $\mathcal{N}_K(z)$ denotes the normal cone to K at z:

$$\mathcal{N}_K(z) := \begin{cases} \{v \in Z' : v^\top (z-u) \geq 0, \, \forall u \in K\}, & \text{if } z \in K, \\ \emptyset \text{ otherwise.} \end{cases}$$

In particular, in the context of nonconvex programming, generalized equations can be used to represent first-order necessary optimality conditions. In fact, with reference to (2.1), we can define

$$F(z) := \begin{bmatrix} \nabla_y f(y) + \nabla_y g(y)\lambda + \nabla_y h(y)\mu \\ -g(y) \\ -h(y) \end{bmatrix}, \tag{2.33}$$

where $z := (y, \lambda, \mu)$, and $K := \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q_+$, such that (2.32), represents the first-order optimality conditions of (2.1).

Remark 2.1.28. *A different and somewhat advantageous reformulation is possible in cases where* (2.1) *takes the form*

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) = 0, \\ & y \in \Omega. \end{aligned} \tag{2.34}$$

where $\Omega \in \mathbb{R}^n$ *is a closed convex set. Following the notation used in [Tran-Dinh et al., 2012], it is possible to write the first-order necessary optimality conditions associated with* (2.34) *as follows:*

$$\begin{aligned} 0 &\in \nabla f(y) + \nabla g(y)\lambda + \mathcal{N}_\Omega(y), \\ 0 &= g(y). \end{aligned} \tag{2.35}$$

In this way, defining $z := (y, \lambda)$, $K = \Omega \times \mathbb{R}^m$ *and*

$$F(z) := \begin{pmatrix} \nabla f(y) + \nabla g(y)\lambda \\ g(y) \end{pmatrix}, \tag{2.36}$$

equations (2.35) *can be reworked into* (2.32)*.*

The main question addressed in [Robinson, 1980], and the one which is of our interest in this section, is to assess whether (2.32) and problems “close” to it have locally unique solutions and whether such solutions have good continuity properties with respect to perturbations introduced into (2.32). In particular, assuming that F is differentiable in a neighborhood of a solution z_0, it is possible to impose a condition on the linearized generalized equation

$$0 \in F(z_0) + \nabla_z F(z_0)^\top (z - z_0) + \mathcal{N}_K(z), \tag{2.37}$$

such that (2.32) satisfies the desired properties. Such condition, named by Robinson *strong regularity*, can be seen as a generalization of the nonsingularity condition used in the implicit function theorem (it is indeed easy to verify that it reduces to that condition if $K = Z$). Although the results in [Robinson, 1980] are more general, in this thesis we are interested in the simplified setting in which $Z = Z' = \mathbb{R}^{n_z}$, so we will restrict our attention to it.

Definition 2.1.29 (Strong regularity)**.** *Let* Ξ *be an open subset of* $\mathbb{R}^{n_z}$ *containing a point* z_0*. Let* K *be a closed convex set in* $\mathbb{R}^{n_z}$ *and let* $F : \Xi \to \mathbb{R}^{n_z}$ *be differentiable at* z_0*. Suppose that the generalized equation*

$$0 \in F(z) + \mathcal{N}_K(z) \tag{2.38}$$

has a solution at z_0 *and, for a given* $z \in \mathbb{R}^{n_z}$*, define*

$$Tz := F(z_0) + \nabla_z F(z_0)^\top (z - z_0) + \mathcal{N}_K(z). \tag{2.39}$$

We say that (2.38) *is strongly regular at* z_0 *with associated Lipschitz constant* σ*, if there exist neighborhoods* U *of the origin in* $\mathbb{R}^{n_z}$ *and* V *of* z_0 *such that the restriction to* U *of* $T^{-1} \cap V$ *is a single-valued function from* U *to* V *which is Lipschitz continuous on* U *with modulus* σ*.*

Remark 2.1.30. *Definition 2.1.29 is the one given in [Robinson, 1980] and we report it here in its original form and using the same language. It is however useful to be aware of the fact that other popular textbooks and publications on the topic may use a slightly different language. In particular, Bonnans et al. [Bonnans and Shapiro, 1998] would say that a solution is strongly stable rather than that the generalized equation is strongly regular at that solution. Moreover, the definition of strong regularity given in [Bonnans and Shapiro, 1998] is somewhat more intuitive and we report it below for completeness.*

Definition 2.1.31 (Strong regularity [Bonnans and Shapiro, 1998])**.** *Let* z_0 *be a solution to* (2.38)*. We say that* z_0 *is a strongly stable solution to* (2.38)*, if there exist neighborhoods* $\mathcal{B}(0, \bar{r}_\delta)$ *and* $\mathcal{B}(0, \bar{r}_z)$ *such that the linearized generalized equation*

$$\delta \in F(z_0) + \nabla_z F(z_0)^\top \Delta z + \mathcal{N}_K(z_0 + \Delta z), \tag{2.40}$$

with unknown Δz *has a unique solution in* $\mathcal{B}(0, \bar{r}_z)$ *and its solution is Lipschitz continuous in* $\mathcal{B}(0, \bar{r}_\delta)$ *with Lipschitz constant* σ*:*

$$\|\Delta \bar{z}(\delta') - \Delta \bar{z}(\delta)\| \leq \sigma \|\delta' - \delta\|, \quad \forall \delta, \delta' \in \mathcal{B}(0, \bar{r}_\delta).$$

We can interpret the concept of strong regularity in terms of perturbed optimization by looking at the case in which (2.40) represents the optimality conditions associated with a perturbed optimization problem and where $\delta = (\delta_1, \delta_2)$ is a perturbation parameter. In fact, (2.40) can be used to represent the first-order optimality conditions of the following problem:

$$\begin{aligned} \min_{y} \quad & (\nabla f(\bar{y}) - \delta_1)^\top y + \frac{1}{2}(y - \bar{y})^\top \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda})(y - \bar{y}) \\ \text{s.t.} \quad & g(\bar{y}) + \nabla_y g(\bar{y})^\top (y - \bar{y}) = \delta_2, \\ & y \in \Omega, \end{aligned} \tag{2.41}$$

for some convex set Ω, which, for $\delta = 0$, is in turn a local approximation at $(\bar{y}, \bar{\lambda})$ of the problem

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) = 0, \\ & y \in \Omega. \end{aligned} \tag{2.42}$$

Hence, problem (2.42) is strongly regular at $\bar{z}$ if and only if, for any $\delta \in \mathcal{B}(0, \bar{r}_\delta)$, (2.41) has a unique KKT point $\bar{z}(\delta)$ in $\mathcal{B}(\bar{z}, \bar{r}_z)$ and $\bar{z}(\delta)$ is Lipschitz continuous in δ over $\mathcal{B}(0, \bar{r}_\delta)$.

The following theorem provides a fundamental result, a type of implicit function theorem for strongly regular generalized equations.

Theorem 2.1.32 (Theorem 2.1, [Robinson, 1980])**.** *Let K, Ξ and z_0 be defined as in 2.1.29. Let $\varepsilon \in \mathbb{R}^r$ be a perturbation parameter and let $F : \Xi \times \mathbb{R}^r \to \mathbb{R}^{n_z}$. Assume that F is differentiable with respect to z, that $\nabla_z F$ and F are continuous at $(z_0, 0)$ and that z_0 solves*

$$0 \in F(z, 0) + \mathcal{N}_K(z). \tag{2.43}$$

If (2.43) *is strongly regular at z_0, with associated Lipschitz constant σ, then for any $\tilde{\sigma} > 0$, there exist neighborhoods $\mathcal{M}_{\tilde{\sigma}}$ of 0 and $\mathcal{Z}_{\tilde{\sigma}}$ of z_0 and a single-valued function $\bar{z} : \mathcal{M}_{\tilde{\sigma}} \to \mathcal{Z}_{\tilde{\sigma}}$, such that, for any $\varepsilon \in \mathcal{M}_{\tilde{\sigma}}$, $\bar{z}(\varepsilon)$ is the unique solution in $\mathcal{Z}_{\tilde{\sigma}}$ of the inclusion*

$$0 \in F(z, \varepsilon) + \mathcal{N}_K(z). \tag{2.44}$$

Further, for each ε_1 and ε_2 in $\mathcal{M}_{\tilde{\sigma}}$, one has

$$\|\bar{z}(\varepsilon_1) - \bar{z}(\varepsilon_2)\| \le (\sigma + \tilde{\sigma})\|F(\bar{z}(\varepsilon_1), \varepsilon_1) - F(\bar{z}(\varepsilon_2), \varepsilon_2)\|. \tag{2.45}$$

Corollary 2.1.33 (Corollary 2.2, [Robinson, 1980])**.** *Assume the notation and hypothesis of Theorem 2.1.32. Assume further that there exists a positive constant L such that, for each $\varepsilon_1, \varepsilon_2 \in \mathcal{M}_{\tilde{\sigma}}$ and each $z \in \mathcal{Z}_{\tilde{\sigma}}$, the following holds:*

$$\|F(z, \varepsilon_1) - F(z, \varepsilon_2)\| \le L\|\varepsilon_1 - \varepsilon_2\|. \tag{2.46}$$

Then, $\bar{z}(\cdot)$ is Lipschitz continuous on $\mathcal{M}_{\tilde{\sigma}}$, i.e.,

$$\|\bar{z}(\varepsilon_1) - \bar{z}(\varepsilon_2)\| \le L(\sigma + \tilde{\sigma})\|\varepsilon_1 - \varepsilon_2\|. \tag{2.47}$$

Finally, the following theorem [Bonnans and Shapiro, 1998, Theorem 5.1] shows that it is possible to approximate the solution $\bar{z}(\varepsilon)$ by $z_0 + z_1(\varepsilon)$, which we will call a *pseudoexpansion* [Bonnans and Shapiro, 1998], where z_0 denotes the unperturbed solution $\bar{z}(0)$ and $z_1(\varepsilon)$ represents the solution to the generalized equation (with unknown ζ)

$$0 \in F(z_0, 0) + \nabla_z F(z_0, 0)^\top \zeta + \nabla_\varepsilon F(z_0, 0)^\top \varepsilon + \mathcal{N}_K(z_0 + \zeta). \tag{2.48}$$

Theorem 2.1.34. *Let K, Ξ and z_0 be defined as in 2.1.29. Let $\varepsilon \in \mathbb{R}^r$ be a perturbation parameter and let $F : \Xi \times \mathbb{R}^r \to \mathbb{R}^{n_z}$. Assume that F is continuously differentiable with respect to z and ε at $(z_0, 0)$ and that z_0 solves*

$$0 \in F(z, 0) + \mathcal{N}_K(z). \tag{2.49}$$

Then, for all ε in a neighborhood of 0, the mappings $\bar{z}(\varepsilon)$ and $z_1(\varepsilon)$ are well defined in the vicinity of z_0 and in a neighborhood of the origin, respectively. In addition, $\bar{z}(\varepsilon)$ is Lipschitz continuous,

$$z_1(\varepsilon) = O\left(\|\varepsilon\|\right) \tag{2.50}$$

and the following holds:

$$\bar{z}(\varepsilon) = z_0 + z_1(\varepsilon) + o\left(\|\varepsilon\|\right). \tag{2.51}$$

2.1.3 Local convergence of Newton-type methods

With the help of relatively standard assumptions, a contraction for generalized Newton-type methods can be obtained by using the assumption of strong regularity. Although several variants of this result can be found in the literature, in the following, we present a simple derivation for the sake of completeness.

We are interested in generalized Newton-type iterates generated by solving the following linearized inclusion:

$$0 \in F(z) + \tilde{F}'(z)(\zeta - z) + \mathcal{N}_K(\zeta), \tag{2.52}$$

with unknown ζ, where $\tilde{F}'(z)$ is an approximation of the exact Jacobian $F'(z)$ computed at the linearization point z. In particular, if $K = \mathbb{R}^{n_g} \times \mathbb{R}^n$, then (2.52) reduces to a Newton-type scheme. Moreover, if $\tilde{F}'(z) = F'(z)$, we obtain the exact generalized Newton method.

Assumption 2.1.35. *Let $\bar{z}$ be a solution to* (2.38)*. Assume that there exists a constant $r_z > 0$, such that*

1. *there exists a positive constant $\tilde{\kappa}$ satisfying $\sigma\tilde{\kappa} < \frac{1}{2}$ such that, for any z in $\mathcal{B}(\bar{z}, r_z)$, the following holds:*

$$\left\|F'(\bar{z}) - \tilde{F}'(z)\right\| \leq \tilde{\kappa}. \tag{2.53}$$

2. *there exists a constant $0 < \tilde{\omega} < \infty$ such that, for any z in $\mathcal{B}(\bar{z}, r_z)$, the following holds:*

$$\|F'(z) - F'(\bar{z})\| \leq \tilde{\omega}\,\|z - \bar{z}\|. \tag{2.54}$$

Notice that the assumptions above are analogous to the ones in [Tran-Dinh et al., 2012] and similar to the standard “kappa” and “omega” conditions for Newton-type methods used for example in [Diehl, 2016].

Lemma 2.1.36. *Let Assumption 2.1.35 hold and assume that* (2.38) *is strongly regular at $\bar{z}$ and let z_+ be a solution to* (2.32)*. Then, if $z \in \mathcal{B}(\bar{z}, r_z)$, the following contraction estimate holds:*

$$\|z_+ - \bar{z}\| \leq \kappa\|z - \bar{z}\| + \frac{\omega}{2}\|z - \bar{z}\|^2, \tag{2.55}$$

where $\kappa = \frac{\sigma\tilde{\kappa}}{(1-\sigma\tilde{\kappa})}$ and $\omega = \frac{\sigma\tilde{\omega}}{1-\sigma\tilde{\kappa}}$.

Proof. Using Assumption 2.1.35, we derive the contraction estimate as follows. The generalized equation (2.52) can be written as

$$\bar{\delta} \in F(\bar{z}) + F'(\bar{z})(z - \bar{z}) + \mathcal{N}_K(z), \tag{2.56}$$

where

$$\begin{aligned} \bar{\delta} :=& \; - F(z) + F(\bar{z}) - \tilde{F}'(z)(z - \bar{z}) + F'(\bar{z})(z - \bar{z}) \\ =& \; - F(z) + F(\bar{z}) - (\tilde{F}'(z) - F'(\bar{z}))(z - \bar{z}) + \tilde{F}'(z)(z - \bar{z}) \\ =& \; - F(z) + F(\bar{z}) - (\tilde{F}'(z) - F'(\bar{z}))(z - \bar{z}) \\ & + (\tilde{F}'(z) - F'(\bar{z}))(z - \bar{z}) + F'(\bar{z})(z - \bar{z}), \end{aligned} \tag{2.57}$$

which is the perturbed linearized generalized equation involved in Definition 2.1.31. Then, using strong regularity at $\bar{z}$, we have that

$$\|z_+ - \bar{z}\| \leq \sigma \left\|\bar{\delta}\right\|, \tag{2.58}$$

for sufficiently small perturbations $\bar{\delta}$. In particular, after defining $\Delta z_+ := \|z_+ - \bar{z}\|$, we can write the following:

$$\begin{aligned} \Delta z_+ \leq \sigma \| & F(z) - F(\bar{z}) - F'(\bar{z})(z - \bar{z}) \\ & + (\tilde{F}'(z) - F'(\bar{z}))(z_+ - \bar{z}) - (\tilde{F}'(z) - F'(\bar{z}))(z - \bar{z})\| \end{aligned}$$

and, using Assumption 2.1.35, we can easily see that the following holds:

$$\Delta z_+ \leq \sigma\|F(z) - F(\bar{z}) - F'(\bar{z})(z - \bar{z})\| + \sigma\tilde{\kappa}\Delta z_+ + \sigma\tilde{\kappa}\|z - \bar{z}\|. \tag{2.59}$$

Using again Assumption 2.1.35 and the mean value theorem, we obtain

$$\begin{aligned} \Delta z_+ \leq \frac{1}{(1 - \sigma\tilde{\kappa})} \Bigg(& \sigma\tilde{\kappa}\|z - \bar{z}\| \\ & + \sigma\| \int_0^1 (F'(\bar{z} + t(z - \bar{z})) - F'(\bar{z}))(z - \bar{z})\mathrm{dt}\| \Bigg) \end{aligned}$$

and, defining $\kappa := \frac{\sigma\tilde{\kappa}}{1 - \sigma\tilde{\kappa}}$ and $\omega := \frac{\tilde{\omega}\sigma}{1 - \sigma\tilde{\kappa}}$, we obtain

$$\Delta z_+ \leq \kappa\|(z - \bar{z})\| + \frac{\omega}{2}\|z - \bar{z}\|^2, \tag{2.60}$$

which concludes the proof. □

2.2 Numerical optimal control and model predictive control

In this section we introduce key concepts on optimal control and model predictive control that, together with the contents of Section 2.1 constitute the theoretical and algorithmic basis most of this thesis relies on. In particular, we will focus both on system theoretic and numerical aspects, i.e., *i)* on stability properties of feedback controllers based on optimal control problems and *ii)* on numerical methods that can be used to solve such problems, and their properties.

2.2.1 Dynamical systems

We are interested in controlling dynamical systems whose dynamics can be described by ordinary differential equations or differential algebraic equations whose definitions we give below.

Definition 2.2.1 (Ordinary differential equation (ODE)). *Let* $t \in \mathbb{R}$ *represent time,* $x(t) \in \mathbb{R}^{n_x}$ *represent the states of the system and* $u(t) \in \mathbb{R}^{n_u}$ *represent the control inputs to the system. An ordinary differential equation (ODE) is an equation of the form*

$$0 = \phi(x(t), \dot{x}(t), u(t)), \tag{2.61}$$

where we assume that $\frac{\partial \phi}{\partial \dot{x}}$ *is invertible for all arguments in an appropriate domain. In the case in which an ODE takes the form*

$$\dot{x}(t) = \phi_e(x(t), u(t)), \tag{2.62}$$

we call it an explicit ODE.

Definition 2.2.2 (Differential algebraic equation (DAE)). *A differential algebraic equation (DAE) is an equation of the form*

$$\phi(\dot{x}(t), x(t), w(t), u(t)) = 0, \tag{2.63}$$

where, we have made the distinction between differential variables $x(t)$ *and algebraic variables* $w(t) \in \mathbb{R}^{n_w}$ *and where* $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$*. If the Jacobian matrix* $\frac{\partial \phi}{\partial (\dot{x}, w)}$ *is invertible, we say that the DAE is of index 1.*

An important special case is the one of semi-explicit DAEs.

Definition 2.2.3 (Semi-explicit DAE). *We call semi-explicit DAEs, equations of the form*

$$\begin{aligned} \dot{x}(t) &= \phi_e(x(t), w(t), u(t)), \\ 0 &= g(x(t), w(t), u(t)). \end{aligned} \tag{2.64}$$

where the function $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_w}$ *defines the algebraic constraints of the DAE. In this case, if the Jacobian matrix* $\frac{\partial g}{\partial w}$ *is invertible, the DAE is of index 1.*

When simulating the dynamics of a system, we are interested in the trajectories $x(t)$ and $w(t)$ of the differential and algebraic variables, respectively, as a function of time and with $t_0 \leq t \leq t_1$, for a given input trajectory $u(t)$ and initial condition $x(t_0) = x_0$.

Definition 2.2.4 (Initial value problem (IVP))**.** *We call initial value problem (IVP) an ODE (or a DAE) together with an appropriate number of initial conditions, e.g.,*

$$\begin{aligned} \dot{x}(t) &= \phi_e(x(t), u(t)), \text{ for all } t_0 \leq t \leq t_1, \\ x(t_0) &= x_0. \end{aligned} \tag{2.65}$$

In order for the solution to an IVP to be of any use in the numerical solution of optimal control problems, we assume throughout the thesis that the conditions of the following fundamental theorem are satisfied.

Theorem 2.2.5 (Picard-Lindelöf theorem)**.** *Regard the initial value problem in Definition 2.2.4 and a given input* $u(t)$*. If* ϕ_e *is uniformly Lipschitz continuous in* x *and continuous in* t*, then there exists an* $\varepsilon > 0$ *such that the initial value problem has a unique solution over the interval* $[t_0 - \varepsilon, t_0 + \varepsilon]$*.*

2.2.2 Optimal control

The main questions addressed in the present thesis, although addressed from different angles, revolve around the challenge of numerically solving optimal control problems in an efficient fashion. The problems that will be regarded take the following general form:

$$\begin{aligned} \min_{s(\cdot),u(\cdot),w(\cdot)} \quad & \int_0^T L(s(\tau), u(\tau), w(\tau))\mathrm{d}\tau + m(s(T), w(T)) \\ \text{s.t.} \quad & s(0) - x = 0, \\ & \dot{s}(t) - \phi(s(t), u(t), w(t)) = 0, \quad t \in [0, T], \\ & g(s(t), u(t), w(t)) = 0, \quad t \in [0, T], \\ & h(s(t), u(t), w(t)) \leq 0, \quad t \in [0, T], \\ & r(s(T), w(T)) \leq 0. \end{aligned} \tag{2.66}$$

Here, the states are denoted by $s(t) \in \mathbb{R}^{n_x}$ - this notation will make it easier to distinguish between optimization variables associated with the state trajectories and the actual state of the system which is described by the parameter x (especially when dealing with the stability results for the combined system-optimizer dynamics in Chapter 3). At the same time, we have introduced the path and terminal constraint functions $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_h}$ and $r : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_r}$. The so-called *Lagrange* and *Mayer* cost terms are defined by $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}$ and $m : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}$. Finally, the time $T \in (0, \infty]$ over which the cost integral is carried out is considered to be fixed.

Although some special instances of continuous-time optimal control problems can be solved in their original formulation either numerically or in closed form and potentially discretized afterwards, solving nonconvex programs in infinite dimensional spaces is in general computationally challenging. For this reason, it is common to first discretize problem (2.66) into a finite dimensional nonlinear program, such that state-of-the-art algorithms for nonlinear programming can be leveraged in order to efficiently solve the discretized problem (this is the so-called *"first discretize, then optimize"* approach).

The main focus of this thesis is on the so-called *direct optimal control methods* that make use of this idea, which will be described next. The two main alternative approaches, namely *indirect approaches* and *state-space approaches* based on the *Hamilton-Jacobi-Bellman equation* will not be discussed since outside of the scope and the interested reader is referred to, e.g., [Diehl and Gros, 2017].

Direct optimal control

In this section, we introduce the class of numerical methods that constitutes the basis for all the numerical algorithms for nonlinear model predictive control presented in this thesis, namely the class of *direct optimal control methods.* In this case, we regard the formulation with path constraints

$$\begin{aligned}
\min_{s(\cdot),u(\cdot)} \quad & \int_0^T L(s(\tau), u(\tau))\mathrm{d}\tau + m(s(T)) \\
\text{s.t.} \quad & s(0) - x = 0, \\
& \dot{s}(t) - \phi(s(t), u(t)) = 0, \quad t \in [0, T], \\
& h(s(t), u(t)) \leq 0, \quad t \in [0, T], \\
& r(s(T)) \leq 0.
\end{aligned} \tag{2.67}$$

Notice that, although we could in principle regard general formulations with, e.g., algebraic variables and constraints and free terminal time, we will restrict our attention to this formulation in order to ease the presentation of the key concepts.

The main idea behind direct methods is to transform the infinite dimensional problem (2.67) into a much simpler to solve finite dimensional nonlinear program whose solution provides an approximation to the solution of the original continuous-time formulation. This is achieved by employing a finite dimensional parametrization of the control trajectories $u(t)$ with parameter q. Among other options, this can be done by choosing a piece-wise constant or piece-wise polynomial parametrization of the control trajectories. For example, if a piece-wise constant parametrization over $N-1$ time intervals defined by the time grid $0 = t_0 < t_1 < \cdots < t_N$ is chosen, we obtain:

$$u(t,q) = q_i, \quad t \in [t_i, t_{i+1}), \quad \text{for } i = 0, \ldots, N-1, \tag{2.68}$$

where $q = (q_0, \ldots, q_{N-1}) \in \mathbb{R}^{N \cdot n_u}$ fully characterizes the input trajectories.

As we will see in the following, there are three main direct methods which all exploit this general concept: *direct single shooting*, *direct multiple shooting* and *direct collocation.*

Direct single shooting

In direct single shooting, the states $s(t)$, for $t \in [0,T]$ and for a given initial condition $s(0) = x$, are regarded as a function of the input $u(t,q)$ and the path constraints are evaluated at a finite number of points in time (usually on the same points as the ones specified by the time grid used to parametrize the control trajectories) such that the optimal control problem can be written as

$$\begin{aligned} \min_{q \in \mathbb{R}^{N \cdot n_u}} \quad & \int_0^T L(s(\tau,q), u(\tau,q))\mathrm{d}\tau + m(s(T,q)) \\ \text{s.t.} \quad & h(s(t_i,q), u(t_i,q)) \leq 0, \quad i = 0, \ldots, N-1, \\ & r(s(T,q)) \leq 0. \end{aligned} \tag{2.69}$$

On the one side, as shown in Figure 2.3, due to the elimination of the state trajectories, the resulting NLP (2.69) does not exhibit a particularly advantageous sparsity pattern and general purpose dense NLP solvers can be used to solve OCP parameterized with direct single shooting. On the other side, the number of variables involved in the NLPs can be significantly reduced

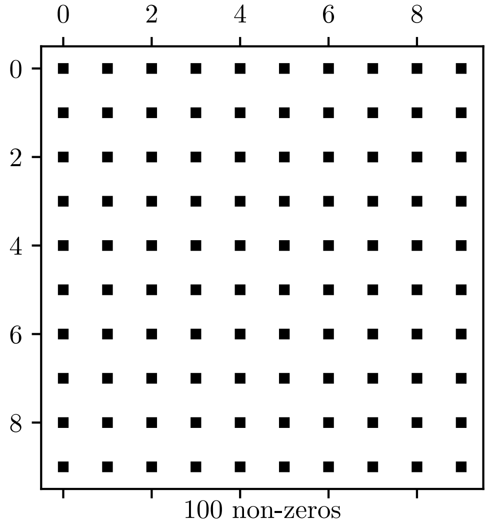


Figure 2.3: Sparsity pattern of the Hessian of the Lagrangian for an NLP obtained with direct single shooting: in this case no sparsity can be exploited.

through this elimination, which can in some cases result beneficial from a computational point of view.

However, one can in general expect that the benefits of such a reduction in problem dimensionality loose their impact on computation times as the number of grid points N increases. In fact, as we will see, other methods that do not eliminate the state trajectories, lead to NLPs with increased sparsity and a structured sparsity pattern that can be leveraged by either general purpose sparse solvers or tailored structure exploiting solvers, respectively. A second major drawback of the direct single shooting method lies in the fact that high sensitivity to initialization and numerical difficulties in general can be encountered when solving OCPs involving highly nonlinear and unstable systems. In fact the function $s(t, q)$ can become increasingly nonlinear in q for larger and larger values of t. As an informal explanation of this fact, if one was to discretize the underlying ODE with explicit Euler with step-size δ, the resulting numerical approximation of $s(t_i, q)$ with $i > 1$, for a given initial condition $s(0) = x$ and a given q would read

$$s(t_i, q) = x + \delta \cdot \phi(x, q) + \delta \cdot \phi(x + \delta \cdot \phi(x, q), q) + \dots, \tag{2.70}$$

which shows that compositions of ϕ with itself appear in the expression for $s(t_i, q)$. This simple observation can provide an intuitive, although informal, explanation of why the direct single shooting method can exhibit inferior numerical performance if compared to other direct methods, namely direct multiple shooting and direct collocation, that do not eliminate the state trajectories from the underlying NLP.

Direct multiple shooting

The numerical difficulties associated with direct single shooting can, to some extent, be circumvented by using the direct multiple shooting method which was first introduced in [Bock and Plitt, 1984]. The key idea in direct multiple shooting is linked to our observation that an integration of the system's dynamics over a long time can lead to highly nonlinear problems. In order to avoid integration from $t = 0$ to $t = T$ the state trajectories are "sliced" into shorter segments and continuity of the trajectory is enforced through *continuity conditions*. This is achieved by introducing additional intermediate variables s_i for $i = 0, \dots, N$ associated with the value of the state trajectories at the so-called shooting notes (which usually coincide with the points defined by the time grid used to parametrize the input trajectories). If, for example, a piece-wise parametrization of the input trajectories is chosen, we obtain a distinct initial value problem of the following form for each $i = 0, \dots, N-1$:

$$\begin{aligned} \dot{\psi}_i(t, s_i, q_i) &= \phi(\psi_i(t, s_i, q_i), q_i), \quad t \in [t_i, t_{i+1}] \\ \psi_i(t_i, s_i, q_i) &= s_i. \end{aligned} \tag{2.71}$$

Together with the numerically computed integrals

$$l_i(s_i, q_i) := \int_{t_i}^{t_{i+1}} L(\psi_i(\tau, s_i, q_i), q_i)\mathrm{d}\tau, \quad \text{for } i = 0, \dots, N-1 \tag{2.72}$$

and the continuity conditions $s_{i+1} = \psi_i(t_{i+1}, s_i, q_i)$ for $i = 0, \dots, N-1$, we obtain the following NLP:

$$\begin{aligned} \min_{\substack{s_0, \cdots, s_N \\ q_0, \cdots, q_{N-1}}} \quad & \sum_{i=0}^{N-1} l_i(s_i, q_i) + m(s_N) \\ \text{s.t.} \quad & s_0 - x = 0 \\ & s_{i+1} - \psi_i(t_{i+1}, s_i, q_i) = 0, \quad i = 0, \cdots, N-1 \\ & h(s_i, q_i) \leq 0, \quad i = 0, \cdots, N-1 \\ & r(s_N) \leq 0. \end{aligned} \tag{2.73}$$

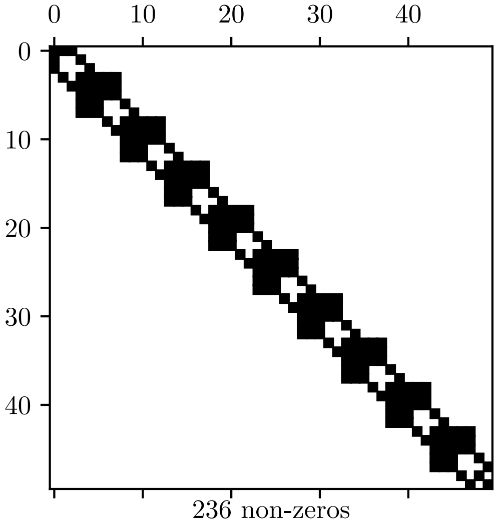


Figure 2.4: Sparsity pattern of the Hessian of the Lagrangian for an NLP obtained with direct multiple shooting: the matrix has a specific sparsity pattern that can be leveraged by tailored structure-exploiting solvers.

Although in comparison to (2.69) this NLP involves more optimization variables it exhibits the favorable sparsity pattern shown in Figure 2.4 that can be leveraged by both general purpose sparse solvers and tailored structure-exploiting solvers. At the same time, this "lifting" allows us to mitigate the nonlinearity associated with the propagation of the state trajectories and to circumvent the numerical issues associated with unstable dynamics. Apart from this qualitative observations, the direct multiple shooting method can be interpreted as a lifted Newton's method, which, as shown in [Albersmeyer and Diehl, 2010], can often achieve better local contraction with respect to the Newton's method applied to the (non-lifted) formulations obtained with direct single shooting.

Direct collocation

The direct collocation method brings the concept of lifting one step further by embedding the nonlinear equations associated with the chosen integration method into the NLP in order to obtain an even larger and sparser structured problem of the form

$$\begin{aligned}
\min_{\substack{s_0,\dots,s_N\\ q_0,\dots,q_{N-1}\\ v_0,\dots,v_{N-1}}} \quad & \sum_{i=0}^{N-1} l_i(s_i,q_i)+m(s_N)\\
\text{s.t.} \quad & s_0 - x = 0,\\
& \varphi(s_i,q_i,v_i) = 0, \quad i = 0,\dots,N-1,\\
& s_i + Cv_i - s_{i+1} = 0, \quad i = 0,\dots,N-1,\\
& h(s_i,u_i) \leq 0, \quad i = 0,\dots,N-1,\\
& r(s_N) \leq 0,
\end{aligned} \tag{2.74}$$

where $v \in \mathbb{R}^{n_v}$ are the variables associated with the integration method. The equation $\varphi(s_i,q_i,v_i) = 0$ represents the collocation equations

$$\varphi(s_i,q_i,v_i) := \begin{bmatrix} \phi(s_i + T_{\text{int}} \sum_{k=1}^{d} a_{1,k} v_i^k, q_i) \\ \vdots \\ \phi(s_i + T_{\text{int}} \sum_{k=1}^{d} a_{d,k} v_i^k, q_i) \end{bmatrix} - \begin{bmatrix} v_i^1 \\ \vdots \\ v_i^d \end{bmatrix} \tag{2.75}$$

associated with the shooting node i, where d denotes the number of collocation nodes and the scalars $a_{i,j}$ with $i, j = 1,\dots,d$ are the coefficients of the collocation method. The integration step size is represented by T_{int} and C in (2.74) is a constant matrix that depends on T_{int} and the collocation nodes.

Numerical methods that can exploit the specific structure arising from the application of the direct collocation method can be found, for example, in [Steinbach, 1996] and [Quirynen et al., 2015].

2.2.3 Model predictive control

Nonlinear model predictive control (NMPC) is an optimization-based control strategy that relies on the solution of optimal control problems in order to compute a feedback policy. Due to the considerable computational burden associated with the solution of such optimal control problems, NMPC has first found application in fields where the sampling times are generally long enough to carry out the required computations. In particular, since the 1970s, successful applications of NMPC have been reported in the process control industry [Rawlings et al., 2017].

In more recent years, due to the significant progress made in the development of efficient algorithms and software implementations and due to the increasing computational power available on embedded control units, NMPC has gradually become a viable strategy for applications with much shorter sampling times. Among others, we report on recent applications such as [Zanelli et al., 2021a], [Albin et al., 2017] and [Besselmann et al., 2015], where sampling times in the milli- and microsecond range are met.

Although numerical solution with direct methods is the main approach of interest within the context of this thesis (and arguably one of the most widespread approaches in general), in this section we will abstract away from the numerical and computational aspects and focus instead on system theoretic considerations. In particular, we are interested in addressing the non trivial question of under which assumptions the NMPC feedback law stabilizes the system to be controlled. Such policy will be regarded as the solution to a discrete-time optimal control problem of the following form:

$$
\begin{aligned}
\min_{\substack{s_0,\cdots,s_N \\ u_0,\cdots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l(s_i, u_i) + m(s_N) \\
\text{s.t.} \quad & s_0 - x = 0 \\
& s_{i+1} - \psi(s_i, u_i) = 0, \quad i = 0, \cdots, N-1 \\
& h(s_i, u_i) \leq 0, \quad i = 0, \cdots, N-1 \\
& r(s_N) \leq 0,
\end{aligned}
\tag{2.76}
$$

where, for simplicity, we have assumed that the cost term l, the dynamics ψ and the constraints h do not vary along the prediction horizon. We will denote by $\bar{y}(x)$ the solution map of (2.76) and by $\bar{u}(x) := M_{u,y}\bar{y}(x)$, for some matrix $M_{u,y}$ of appropriate dimensions, the associated feedback policy. The main idea behind NMPC lies in, at every sampling time, solving an instance of (2.76) with x equal to the current (measured or estimated) state of the system and applying the first “move” in the optimal input trajectory to the system, thus obtaining the following closed-loop dynamics:

$$
x_{\text{next}} = \psi(x, \bar{u}(x)). \tag{2.77}
$$

In the following section, before introducing some fundamental definitions and results on stability theory, we report a rather standard result on stability of tracking NMPC adapted from [Rawlings et al., 2017].

Remark 2.2.6. *Although stability results for more general NMPC settings with, e.g., time-varying costs are present in the literature (see, e.g.,* *[Rawlings et al., 2017]), we restrict our attention to this setting which is the most common and most interesting within the scope of this introductory discussion of the stability theory for NMPC. In Chapter 5, we will address a specific class of problems with stage-varying (as opposed to time-varying) costs and constraints, namely the class of progressively tightening formulations, for which it is possible to derive stability guarantees.*

2.2.4 Stability theory

In this section, we introduce some fundamental definitions and results on stability theory that will be used to address stability of closed-loop systems controlled with NMPC. Most of the contents of this section comply with the definitions in [Rawlings et al., 2017, Appendix B]. In particular, we are interested in assessing system theoretic properties of an autonomous system of the form

$$x_+ = \psi(x) \tag{2.78}$$

with state $x \in \mathbb{R}^{n_x}$ and dynamics $\psi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$. The first fundamental concept required to define the properties of interest is the concept of equilibrium.

Definition 2.2.7 (Equilibrium). *A point x^* is said to be an equilibrium point for $x_+ = \psi(x)$ if $\psi(x^*) = x^*$.*

Loosely speaking, we would like the trajectories of the autonomous system (2.78) to be well behaved in the sense that small perturbations to the initial condition do not lead to large variations in the future values of the state and, potentially, that, after a perturbation, the state trajectories converge back to the equilibrium under consideration.

In certain situations it might be useful to refer to stability and convergence to a set, which we will call a *positive invariant*, rather than to an equilibrium point.

Definition 2.2.8 (Positive invariant set). *A positive invariant set for the dynamics $x_+ = \psi(x)$ is a closed set $\mathcal{A}$ such that $x \in \mathcal{A} \implies \psi(x) \in \mathcal{A}$.*

The following definitions will be useful to introduce the concept of Lyapunov function, which is a key mathematical tool for the characterization of stability properties of dynamical systems.

Definition 2.2.9 ($\mathcal{K}$, $\mathcal{K}_\infty$, $\mathcal{KL}$ and $\mathcal{PD}$ functions). *We call $\alpha : \mathbb{R}_+ \to \mathbb{R}_+$ a $\mathcal{K}$ function if it is continuous, zero at zero, i.e., $\alpha(0) = 0$, and strictly increasing. We say that α belongs to class $\mathcal{K}_\infty$ if it is a $\mathcal{K}$ function and additionally $\lim_{s\to\infty} \alpha(s) = \infty$. We call $\beta : \mathbb{R}_+ \times \mathbb{I}_+ \to \mathbb{R}_+$ a $\mathcal{KL}$ function if it is continuous* *and if, for any* $t \geq 0$*,* $\beta(\cdot, t)$ *is a* $\mathcal{K}$ *function and, for any* $s \geq 0$*,* $\beta(s, \cdot)$ *is nonincreasing and satisfies* $\lim_{t\to\infty} \beta(s,t) = 0$*. Finally, we call* $\gamma : \mathbb{R} \to \mathbb{R}_+$ *a* $\mathcal{PD}$ *function if it is zero at zero, i.e,* $\gamma(0) = 0$*, and positive anywhere else.*

We are now ready to introduce the definitions of stability, attractivity and their variants. We will assume that ψ is locally bounded and that $\mathcal{A}$ is a closed and positive invariant set for $x_+ = \psi(x)$ for the rest of this section.

Definition 2.2.10 (Local stability)**.** *For a given initial condition* x*, let* $\phi(k; x) : \mathbb{I}_+ \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ *denote the value of the state of system* (2.78) *after* k *time instants, i.e.,*

$$\begin{aligned} \phi(0, x) &= x \\ \phi(1, x) &= \psi(x) \\ \phi(2, x) &= \psi(\psi(x)) \\ &\vdots \end{aligned} \tag{2.79}$$

The (closed and positive invariant) set $\mathcal{A}$ *is locally stable for* $x_+ = \psi(x)$ *if, for any* $\varepsilon > 0$*, there exists a* $\delta > 0$ *such that* $|x|_{\mathcal{A}} < \delta$ *implies that* $|\phi(k; x)|_{\mathcal{A}} < \varepsilon$ *for all* $k > 0$*.*

Remark 2.2.11. *Notice that the definition of local stability involves an* $\varepsilon - \delta$ *argument that closely resembles one used in the definition of continuity of a function. In fact, interestingly, we could say that local stability is equivalent to continuity of the map* $x \to \mathbf{x} := \{x, \phi(1, x), \phi(2, x), \dots\}$ *at the equilibrium under analysis.*

Definition 2.2.12 (Attractivity)**.** *The (closed, positive invariant) set* $\mathcal{A}$ *is said to be globally attractive for* $x_+ = \psi(x)$ *if, for any* $x \in \mathbb{R}^{n_x}$*,* $\lim_{k\to\infty} |\phi(k, x)|_{\mathcal{A}} = 0$*. Similarly, we say that* $\mathcal{A}$ *is locally attractive for* $x_+ = \psi(x)$ *if there exists a constant* $\eta > 0$ *such that* $|x|_{\mathcal{A}} < \eta$ *implies* $\lim_{k\to\infty} |\phi(k, x)|_{\mathcal{A}} = 0$*.*

Definition 2.2.13 (Asymptotic stability)**.** *The (closed, positive invariant) set* $\mathcal{A}$ *is said to be globally asymptotically stable for* $x_+ = \psi(x)$*, if it is locally stable and globally attractive. Similarly, we say that* $\mathcal{A}$ *is locally asymptotically stable if it is locally stable and locally attractive.*

In order to assess the stability properties of a dynamical system, rather than verifying that it satisfies the $\varepsilon - \delta$ property, we rely on Lyapunov theory. In particular, if a function can be found that satisfies certain “dissipation-like” properties, then we are able to prove asymptotic stability of the system under analysis.

Definition 2.2.14 (Lyapunov function)**.** *Let X be a positive invariant set and $\mathcal{A}$ a closed positive invariant set for $x_+ = \psi(x)$. Assume that ψ is locally bounded. We call $V : X \to \mathbb{R}_+$ a Lyapunov function for $x_+ = \psi(x)$ in X if there exist functions $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ and a continuous function $\alpha_3 \in \mathcal{PD}$ such that*

$$\alpha_1 \left(|x|_{\mathcal{A}}\right) \leq V(x) \leq \alpha_2(|x|_{\mathcal{A}}), \tag{2.80a}$$

$$V(\psi(x)) - V(x) \leq -\alpha_3(|x|_{\mathcal{A}}) \tag{2.80b}$$

hold for any $x \in X$.

The following fundamental theorem shows that we can use Lyapunov functions to certify asymptotic stability of a dynamical system.

Theorem 2.2.15 (Theorem B.13, [Rawlings et al., 2017])**.** *Let X be a positive invariant set and $\mathcal{A}$ a closed and positive invariant set for $x_+ = \psi(x)$ and assume that ψ is locally bounded. If V is a Lyapunov function in X for $x_+ = \psi(x)$ and $\mathcal{A}$, then $\mathcal{A}$ is globally asymptotically stable.*

2.2.5 Stability of tracking NMPC

In this section we summarize a standard result on stability of tracking NMPC which lies the basis for the stability proofs derived in Chapters 3, 4 and 5. In particular, the result exploits Theorem 2.2.15 in the sense that it shows that, under reasonable and rather general assumptions, the optimal cost associated with problem (2.76) is a Lyapunov function for the closed-loop system.

To this end, and with reference to (2.76), let $\mathbb{Z} := \{(s, u) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \,|\, h(s, u) \leq 0\}$, $\mathbb{U}(s) := \{u \in \mathbb{R}^{n_u} \,|\, (s, u) \in \mathbb{Z}\}$, $\mathbb{X} := \{s \in \mathbb{R}^{n_x} \,|\, \mathbb{U}(s) \neq \emptyset\}$ and $\mathbb{X}_f := \{s \in \mathbb{R}^{n_x} | \, r(s) \leq 0\}$. Moreover, let $\bar{\mathbb{X}} \subseteq \mathbb{X}$ denote the set containing all the x for which (2.76) has a solution. We require the following Assumptions to hold:

Assumption 2.2.16 (Continuity of system and cost)**.** *Assume that the origin is a steady state with $\psi(0, 0) = 0$, $l(0, 0) = 0$, and $m(0) = 0$. Moreover, assume that l and h, m, ψ, r and are continuous.*

Assumption 2.2.17 (Properties of constraint sets)**.** *The set $\mathbb{Z}$ is closed and the set $\mathbb{U}(s)$ is compact and uniformly bounded in $\mathbb{X}$. The set $\mathbb{X}_f \subseteq \mathbb{X}$ is compact and each set contains the origin.*

Assumption 2.2.18 (Basic stability assumption)**.** *The terminal cost m, the set $\mathbb{X}_f$ and the cost term l satisfy the following properties:*

1. *For all* $x \in \mathbb{X}_f$*, there exists a* u*, such that* $(x, u) \in \mathbb{Z}$*, satisfying*

$$\psi(x, u) \in \mathbb{X}_f,$$

$$m(\psi(x, u)) - m(x) \leq -l(x, u).$$

2. *There exists a* $\mathcal{K}_\infty$ *function* α_l *such that*

$$l(x, u) \geq \alpha_l(\|x\|), \quad \forall x \in \bar{\mathbb{X}},$$

for any u *such that* (x, u) *is in* $\mathbb{Z}$*.*

Assumption 2.2.19 (Weak controllability)**.** *There exists a* $\mathcal{K}_\infty$ *function* $\alpha_2(\cdot)$ *such that, for the optimal cost associated with* (2.76)

$$V(x) := \sum_{i=0}^{N-1} l(\bar{s}_i(x), \bar{u}_i(x)) + m(\bar{s}_N(x)), \tag{2.81}$$

the following holds:

$$V(x) \leq \alpha_2(\|x\|), \forall x \in \bar{\mathbb{X}}. \tag{2.82}$$

Under the requirement that Assumptions 2.2.16, 2.2.17, 2.2.18 and 2.2.19 hold, the main stability result can be derived. In particular, it can be shown that $V(x)$ is a Lyapunov function for the closed-loop system obtained by controlling the system with the optimal feedback law $M_{u,y}\bar{y}(x)$ in a receding horizon fashion [Rawlings et al., 2017]:

Theorem 2.2.20 (Asymptotic stability of NMPC)**.** *Suppose that Assumptions 2.2.16, 2.2.17, 2.2.18 and 2.2.19 are satisfied. Then the following hold:*

1. *There exist* $\mathcal{K}_\infty$ *functions* $\alpha_1(\cdot)$ *and* $\alpha_2(\cdot)$ *and a positive definite function* $\alpha_3(\cdot)$ *such that*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|), \tag{2.83a}$$

$$V(\psi(x, M_{u,y}\bar{y}(x))) - V(x) \leq -\alpha_3(\|x\|), \tag{2.83b}$$

for all $x \in \bar{\mathbb{X}}$*.*

2. *The origin* $x = 0$ *is an asymptotically stable equilibrium with attraction region* $\bar{\mathbb{X}}$ *for the closed-loop system.*

Proof. First, it is necessary to identify valid candidates for the functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ such that the inequality (2.83a) is satisfied. The fact that a $\mathcal{K}_\infty$ function $\alpha_2(\cdot)$ exists such that

$$V(x) \leq \alpha_2(\|x\|), \quad \forall x \in \bar{\mathbb{X}}$$

is a direct consequence of Assumption 2.2.19. Due to Assumption 2.2.18, we have

$$V(x) \geq l(x, \bar{u}(x)) \geq \alpha_l(\|x\|), \quad \forall x \in \bar{\mathbb{X}},$$

for any u such that (x, u) is in $\mathbb{Z}$, which implies that $\alpha_1(\cdot) = \alpha_l(\cdot)$ can be used.

Second, it is necessary to show that the decrease property in (2.83b) holds. To this end, consider the following feasible control and state sequences obtained by shifting the solution to (2.76) for a given x:

$$\tilde{u} = \{\bar{u}_1(x), \ldots, \bar{u}_{N-1}(x), \tilde{u}_N\}$$

and

$$\tilde{s} = \{\bar{s}_1(x), \ldots, \psi(\bar{s}_N(x), \tilde{u}_N)\},$$

where $\tilde{u}_N$ is any feasible control action that guarantees that $\psi(\bar{s}_N(x), \tilde{u}_N) \in \mathbb{X}_f$. Such a control action $\tilde{u}_N$ is guaranteed to exist by Assumption 2.2.18. We introduce the cost associated with the suboptimal sequences $\tilde{u}$ and $\tilde{s}$

$$\begin{aligned}\tilde{V}(\tilde{s}, \tilde{u}) :=& \sum_{i=1}^{N} l(\bar{s}_i(x), \bar{u}_i(x)) + m(\psi(\bar{s}_N(x), \tilde{u}_N)) \\ =& \, V(x) - l(x, \bar{u}_0(x)) + l(\bar{s}_N(x), \tilde{u}_N) \\ & - m(\bar{s}_N(x)) + m(\psi(\bar{s}_N(x), \tilde{u}_N))\end{aligned}$$

and, due to Assumption 2.2.18 together with the fact that $(\tilde{s}, \tilde{u})$ is a feasible, but suboptimal solution, we obtain that

$$V(\psi(x, \bar{u}_0(x))) \leq \tilde{V}(\tilde{s}, \tilde{u}) \leq V(x) - l(x, \bar{u}_0(x)). \tag{2.84}$$

This last inequality shows that (2.83b) holds with $\alpha_3(\cdot) = \alpha_l(\cdot)$.

Finally, in order to conclude the proof, Theorem 2.2.15 can be used. The set $\bar{\mathbb{X}}$ can be shown to be positive invariant for the closed-loop system following the arguments in the proof of [Rawlings et al., 2017, Proposition 2.10(b)]. Moreover, inequalities (2.80a) and (2.80b) are trivially satisfied through (2.83a) and (2.83b). This implies that Theorem 2.2.15 can be applied and the origin can be shown to be asymptotically stable in $\bar{\mathbb{X}}$. □

Chapter 3

Asymptotic stability of the system-optimizer dynamics in NMPC

NMPC applications with high sampling rates often rely on approximate feedback policies in order to meet the required computation times. Among other approaches, the real-time iteration (RTI) strategy proposed in [Diehl, 2002] exploits a single iteration of a sequential quadratic programming programming (SQP) algorithm in order to compute an approximate solution of the current instance of the nonlinear parametric optimization problem. By using this solution to warmstart the SQP algorithm at the next sampling time, it is possible to track an optimal solution and eventually converge to it, as the system is steered to a steady state [Diehl et al., 2005]. In this chapter, we analyze RTI-like methods for nonlinear model predictive control (NMPC) that rely on a limited number of iterations to compute an approximate solution to the underlying parametric nonlinear programs. Loosely speaking, the main challenge present in real-time approaches lies in the fact that the dynamics of the system and the ones of the optimizer interact with each other in a non-trivial way. A formal definition of the system-optimizer dynamics requires the introduction of several concepts that we delay to Section 3.2. However, we can picture a setting in which

$$x_+ = \psi(T; x, u) \tag{3.1}$$

describes the discrete-time dynamics of the system to be controlled, where T denotes the sampling time, x the state of the system, and u the input applied

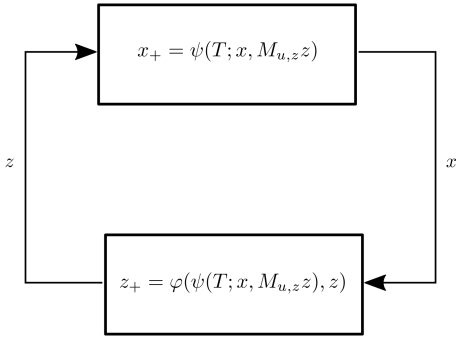

$x_+ = \psi(T; x, M_{u,z}z)$
z
x
$z_+ = \varphi(\psi(T; x, M_{u,z}z), z)$

Figure 3.1: Coupled system-optimizer dynamics $\xi_+ = \Phi(T;\xi)$: when a limited number of iterations of the optimization algorithm are carried out in order to obtain an approximate solution that is then used to control the system, the system's and the optimizer's dynamics interact with one another.

to it. Similarly,

$$z_+ = \varphi(x, z) \tag{3.2}$$

denotes the dynamics of the optimizer, where z and x describe the iterates and the parameter of the underlying (potentially) nonconvex programs, respectively. Whenever the global minimizer to the nonconvex programs is used to control the system, standard stability results hold for the closed-loop dynamics. However, when using a real-time method, a limited number of iterations described by (3.2) are carried out. For a given state x and an approximate primal-dual solution z, the system is controlled using the control $u = M_{u,z}z$, where $M_{u,z}$ is a constant matrix, and, after the sampling time T, its state is steered to $x_+ = \psi(T; x, M_{u,z}z)$. Analogously, the optimizer generates a new approximate solution $z_+ = \varphi(\psi(T; x, M_{u,z}z), z)$. We will refer to these coupled dynamics, with state $\xi = (x, z)$, as $\xi_+ = \Phi(T;\xi)$, which will be defined later in the chapter. The system-optimizer interaction is visualized in Figure 3.1. The main focus of the results presented in this chapter is the stability analysis of $\xi_+ = \Phi(T;\xi)$ for general real-time methods.

Related work

Attractivity proofs for the RTI strategy in slightly different settings, and under the assumption that inequalities are either absent or inactive in a neighborhood of the equilibrium, are presented in [Diehl et al., 2005] and [Diehl et al., 2007]. In the same spirit, similar algorithms that rely on a limited number of iterations are present in the literature and, in the following, we give an overview of such approaches. In [Graichen and Kugi, 2010] a general framework that covers methods with linear contraction in the objective function value is analyzed and an asymptotic stability proof is provided. The recent work of [Liao-McPherson et al., 2019] addresses a more general setting where an SQP algorithm is used. A proof of local input-to-state stability is provided based on the assumption that a sufficiently high number of iterations is carried out per sampling time. In the convex setting, the works in [Feller and Ebenbauer, 2017] and [Van Parys and Pipeleers, 2018] introduce stability results for relaxed barrier *anytime* methods and real-time projected gradient methods, respectively. Finally, the works in [Scokaert et al., 1999], [Pannocchia et al., 2011] and [Allan et al., 2017], analyze conditions under which suboptimal NMPC is stabilizing given that a feasible warmstart is available.

Outline

The chapter is structured as follows. Section 3.1 is based on [Zanelli et al., 2019b] and presents general contraction estimates for a broad class of real-time algorithms. Section 3.2 is based on [Zanelli et al., 2021b] and [Zanelli et al., 2020] and introduces extensions to the results in [Diehl et al., 2005, Diehl et al., 2007]. Namely, it introduces a proof of asymptotic stability for a general setting with two fundamental ingredients *i)* a Lyapunov function for the ideal closed-loop system *ii)* a solver whose iterates converge Q-linearly to the ideal policy. Under these assumptions, it is shown that, for sufficiently short sampling times, we can explicitly construct a Lyapunov function and hence prove asymptotic stability of the system-optimizer dynamics.

3.1 Contraction estimates for real-time methods

Before studying the interaction between the system and optimizer dynamics from a system theoretic point of view, we will focus our attention on the optimizer dynamics. In this section, we derive a general contraction estimate for a class of algorithms that can be used in the context of NMPC to compute approximate and computationally cheap feedback policies. The main idea lies in performing a small number or, in the limit, a single iteration of a Q-linearly convergent algorithm in order to trade control performance for computational speed. In this way, we aim at generalizing existing results based on specific settings. We derive a contraction estimate that provides a bound on the numerical error incurred when using such strategies under the assumption of Robinson's strong regularity of the underlying generalized equation. Moreover, we show that several algorithms fall in the class to which our results are applicable. Finally, we present a numerical example where, using one of the proposed real-time methods, a speedup of more than an order of magnitude can be obtained, at the cost of moderate closed-loop suboptimality.

3.1.1 Real-time methods and the real-time dilemma

The results derived in Section 3.1.2 are not specific to optimal control. However, a motivation for practical usefulness of the algorithms described in this section lies in the fact that in NMPC one might be interested in obtaining approximate solutions within shorter computation times, rather than accurate ones within longer computation times. This consideration is a consequence of the fact that the solution to the nonconvex program based on the latest available state estimate might become excessively outdated if we dedicate too much time to its computation as the true state of the system evolves in time. The RTI is a specific strategy that addresses this issue. The main concept used in the RTI is based on numerical continuation ideas (see, e.g., [Allgower and Georg, 1990]): under proper regularity assumptions, we can track the parametric optimal solution with a limited number of iterations of the optimizer.

Although a complete survey of approaches that exploit similar ideas is well beyond the scope of this section, in the following, we mention a few strategies that make use of similar concepts. The work in [Diehl, 2001, Diehl et al., 2002a, Diehl et al., 2003, Diehl et al., 2007] focuses on convergence and attractivity properties of Newton-type methods under the assumption of stable active set. In [Tran-Dinh et al., 2012] and [Zavala and Anitescu, 2010], based on the framework of strongly regular generalized equations, contraction results are obtained for path-following sequential convex programming and truncated Newton-type methods, respectively. Results in the same spirit are derived in [Graichen and Kugi, 2010] for linearly convergent and feasible iterates. Finally, for linear systems, recent results have been obtained using relaxed barrier formulations [Feller and Ebenbauer, 2016] and the projected gradient method [Van Parys and Pipeleers, 2018]. All of the above mentioned methods aim at tracking the parametric optimal solution with a limited number of iterations. We will refer to these methods as *real-time methods* in order to make an explicit connection at the semantic level with the well known RTI strategy. Although results on contraction properties for real-time algorithms have been derived for specific settings, we aim at providing general results that can be applied to a broad class of algorithms.

In particular, we regard methods that generate Q-linearly convergent iterates in a neighborhood of the solution. Under the assumption of strong regularity

[Robinson, 1980], we derive bounds for the numerical error attained when a single iteration of the abstract method is applied per sampling time in a real-time setting. The results obtained in this way do not depend on the assumption of a fixed active set across iterates and can be interpreted as extensions to the works in [Tran-Dinh et al., 2012] and [Zavala and Anitescu, 2010], where similar contractions have been obtained for sequential convex programming and truncated exact Newton algorithms, respectively. We emphasize that the proposed framework is independent of the numerical method used to solve the NLPs as we only require that it has at least local Q-linear convergence rate in a generic semi-norm. The abstraction from a specific numerical method both simplifies the results and provides more general estimates that can be applied to several algorithms.

3.1.2 Contraction estimates

In order to derive the contraction estimates, we will regard the following parametric optimization problem:

$$P(x): \quad \begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) + \hat{C}x = 0, \\ & y \in \Omega, \end{aligned} \tag{3.3}$$

where $y \in \mathbb{R}^n$, $\Omega \subseteq \mathbb{R}^n$ is a nonempty, closed and convex set, the functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n_g}$ are twice continuously differentiable functions, $x \in \mathbb{R}^{n_x}$ is a parameter and $\hat{C} \in \mathbb{R}^{n_g \times n_x}$.

Remark 3.1.1. *Notice that the optimal control problem in* (2.76) *can be easily reformulated into* (3.3) *by properly defining* f, g, $\hat{C}$ *and* Ω. *Although the results derived in this section do not rely on the specific formulation in* (2.76), *we will make considerations specific to NMPC in Section 3.1.4.*

Following the reformulation introduced in Section 2.1.2, we rework the first-order optimality conditions of (3.3) as the generalized equation

$$0 \in F(z) + Cx + \mathcal{N}_K(z), \tag{3.4}$$

where $C := [0 \;\; \hat{C}^\top]^\top$.

Let $\bar{Z}(x)$ be the set of KKT points satisfying (3.4) for a given x. In order to be able to refer to a well-behaved "branch" of the solution map, as illustrated in Figure 3.2, we will make the following assumption.

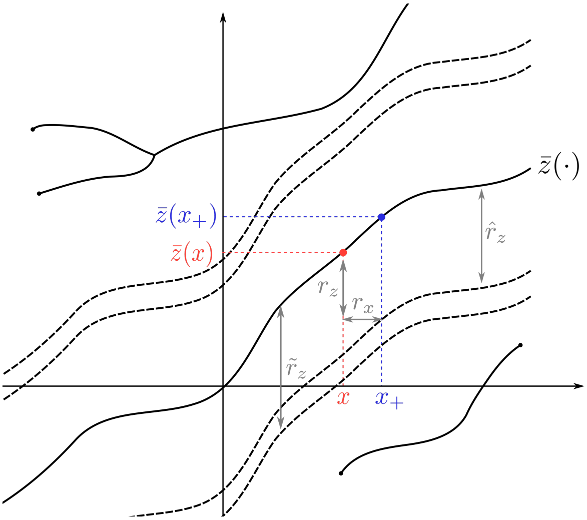


Figure 3.2: Illustration of Assumption 3.1.2 and Lemma 3.1.5. Our setting does not require that $\bar{Z}$ is single-valued, but rather that there exists a single-valued and Lipschitz continuous localization $\bar{z}$ of the solution map. For sufficiently small changes in the parameter, any z within a ball of radius r_z centered at $\bar{z}(x)$ leads to a warmstart that is at most $\hat{r}_z$ "distant" from $\bar{z}(x_+)$ such that contraction (3.7) can be applied.

Assumption 3.1.2. *There exist a single-valued localization $\bar{z} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ of the solution map of* (3.4) *and a non-empty set X such that, for all $x \in X$,* (3.4) *is strongly regular at $\bar{z}(x)$.*

Proposition 3.1.3 (Lemma 3.3, [Tran-Dinh et al., 2012])**.** *Let Assumption 3.1.2 hold. Then, there exist positive constants $\tilde{r}_x$ and $\tilde{r}_z$ such that, for any x in X and any $x' \in \mathcal{B}(x, \tilde{r}_x)$, $\bar{Z}(x') \cap \mathcal{B}(\bar{z}(x), \tilde{r}_z)$ contains a single point $\bar{z}(x')$. Moreover, there exists $\tilde{\gamma} > 0$ such that the following holds:*

$$\|\bar{z}(x') - \bar{z}(x)\| \leq \tilde{\gamma}\, \|x' - x\| \,. \tag{3.5}$$

Let P be a symmetric positive semidefinite matrix. We define a semi-norm:

$$\|z\|_P := \left(z^\top P z\right)^{\frac{1}{2}} \tag{3.6}$$

and denote balls defined using the above semi-norm by $\mathcal{B}_P(\cdot,\cdot)$. Moreover, we define the constant $\gamma := \sigma_{\max}(P)\,\tilde{\gamma}$.

Our goal is to provide a unified framework to analyze real-time methods for the parametric problem (3.3), which is independent of the numerical strategy used to solve the underlying NLP associated with each parameter value x. To this end, we require the use of methods to compute solutions of (3.4) at x that have at least local Q-linear convergence rate in the semi-norm $\|\cdot\|_P$ as stated in Assumption 3.1.4. Loosely speaking, the use of semi-norms allows for a treatment of algorithms which only provide contraction in a subspace of the space of the iterates.

Assumption 3.1.4. *There exists a radius $\hat{r}_z$ such that, for any $x \in X$, and any z in $\mathcal{B}_P(\bar{z}(x), \hat{r}_z)$, the optimization routine can produce z_+ such that*

$$\|z_+ - \bar{z}(x)\|_P \leq \kappa \|z - \bar{z}(x)\|_P + \frac{\omega}{2}\|z - \bar{z}(x)\|_P^2, \tag{3.7}$$

for some positive constants $0 \leq \kappa < 1$ and $0 \leq \omega < \infty$.

Lemma 3.1.5. *Let Assumptions 3.1.2 and 3.1.4 hold, then there exist strictly positive constants r_z and r_x, such that, for any $x \in X$, any z in $\mathcal{B}_P(\bar{z}(x), r_z)$, and any x_+ in $\mathcal{B}(x, r_x) \cap X$, the following holds:*

$$\begin{aligned}\|z_+ - \bar{z}(x_+)\|_P \leq{}& \kappa \|z - \bar{z}(x)\|_P + \frac{\omega}{2}\|z - \bar{z}(x)\|_P^2 \\ &+ \gamma\kappa \|x_+ - x\| + \frac{\omega\gamma^2}{2}\|x_+ - x\|^2 \\ &+ \omega\gamma \|z - \bar{z}(x)\|_P \|x_+ - x\|.\end{aligned} \tag{3.8}$$

Proof. First, we show that, for sufficiently small $r_x > 0$ and $r_z > 0$, for any $x \in X$ and any z in $\mathcal{B}_P(\bar{z}(x), r_z)$, we can guarantee that $\|z - \bar{z}(x_+)\|_P \leq \hat{r}_z$. This, together with the fact that $x_+ \in \mathcal{B}(x, r_x) \cap X$, and hence $x_+ \in X$, allows us to apply the contraction from Assumption 3.1.4 at $\bar{z}(x_+)$. Since

$$\begin{aligned}\|z - \bar{z}(x_+)\|_P &\leq \|z - \bar{z}(x)\|_P + \|\bar{z}(x_+) - \bar{z}(x)\|_P \\ &\leq \|z - \bar{z}(x)\|_P + \sigma_{\max}(P)\|\bar{z}(x_+) - \bar{z}(x)\| \\ &\leq \|z - \bar{z}(x)\|_P + \sigma_{\max}(P)\tilde{\gamma}\|x_+ - x\| \\ &\leq r_z + \gamma r_x,\end{aligned} \tag{3.9}$$

by choosing $r_z \leq \hat{r}_z - \gamma r_x$, with $r_x < \min\{\tilde{r}_x, \frac{\hat{r}_z}{\gamma}\}$, we can ensure that $z \in \mathcal{B}_P(\bar{z}(x_+), \hat{r}_z)$ for any $z \in \mathcal{B}_P(\bar{z}(x), r_z)$ and $x_+ \in \mathcal{B}(x, r_x)$. Then, by Assumption 3.1.4, for any z in $\mathcal{B}_P(\bar{z}(x_+), \hat{r}_z)$, we can write

$$\|z_+ - \bar{z}(x_+)\|_P \leq \kappa \|z - \bar{z}(x_+)\|_P + \frac{\omega}{2}\|z - \bar{z}(x_+)\|_P^2. \tag{3.10}$$

Using the triangle inequality we obtain

$$\begin{aligned}\left\|z-\bar{z}(x_+)\right\|_P &= \left\|z+\bar{z}(x)-\bar{z}(x)-\bar{z}(x_+)\right\|_P \\ &\leq \left\|z-\bar{z}(x)\right\|_P + \left\|\bar{z}(x)-\bar{z}(x_+)\right\|_P .\end{aligned}$$

Plugging this into (3.10), and using (3.5) and the fact that $\|\cdot\|_P \leq \sigma_{\max}(P)\|\cdot\|$, we obtain that

$$\begin{aligned}\left\|z_+-\bar{z}(x_+)\right\|_P \leq{}& \kappa\left\|z-\bar{z}(x)\right\|_P + \frac{\omega}{2}\left\|z-\bar{z}(x)\right\|_P^2 \\ &+ \gamma\kappa\left\|x_+-x\right\| + \frac{\omega\gamma^2}{2}\|x_+-x\|^2 \\ &+ \omega\gamma\left\|z-\bar{z}(x)\right\|_P\left\|x_+-x\right\|\end{aligned}$$

holds, for any x_+ in $\mathcal{B}(x,\tilde{r}_x)\cap X$ and any $z \in \mathcal{B}_P(\bar{z}(x_+),\hat{r}_z)$. It follows that the above contraction holds for any z in $\mathcal{B}_P(\bar{z}(x),r_z)$, and any x_+ in $\mathcal{B}(x,r_x)\cap X$, which concludes the proof. □

Lemma 3.1.5 shows that, for sufficiently small changes in the value of the parameter x, contraction of the iterates can be preserved. This property is of great help in the context of real-time methods and, as we will see in Section 3.1.4, in the context of NMPC in particular. In fact, thanks to Lemma 3.1.5, we can guarantee that the contraction of a method employed to approximately solve the nonlinear programs is not lost if we perform a single iteration per problem instance. Moreover, the following theorem shows that we can guarantee that certain bounds on the numerical error can be satisfied under the assumption that the parameter variation $\|x_+-x\|$ is sufficiently small.

Theorem 3.1.6. *Let Assumptions 3.1.2 and 3.1.4 hold. There exists a positive constant* $0 < r_x^s < r_x$ *(*cf. *Lemma 3.1.5), such that, for any* $x, x_+ \in X$*, if* $\|x_+-x\| \leq r_x^s$ *and* $\|z-\bar{z}(x)\|_P \leq r_z$*, then*

$$\|z_+-\bar{z}_+\|_P \leq r_z. \tag{3.11}$$

Proof. Since $\|z-\bar{z}(x)\|_P \leq r_z$ and $\|x_+-x\| \leq r_x^s$, the following holds:

$$\left\|z_+-\bar{z}(x_+)\right\|_P \leq \kappa r_z + \frac{\omega}{2}r_z^2 + \gamma\kappa r_x^s + \frac{\omega\gamma^2}{2}(r_x^s)^2 + \omega\gamma r_z r_x^s.$$

In order to ensure that $\left\|z_+-\bar{z}(x_+)\right\|_P \leq r_z$, it suffices to choose

$$r_x^s < \frac{1}{\omega\gamma^2}\Big(-\gamma(\kappa+\omega r_z) + \sqrt{\gamma^2(\kappa+\omega r_z)^2 - \omega\gamma^2(2(\kappa-1)+\omega r_z)}\Big)$$

if $\frac{\omega\gamma^2}{2} > 0$, and

$$r_x^s < \frac{r_z(2(1-\kappa) - \omega r_z)}{2\gamma(\kappa + \omega r_z)}, \tag{3.12}$$

if $\frac{\omega\gamma^2}{2} = 0$, which concludes the proof. □

3.1.3 Concrete real-time algorithms

In this section, concrete examples of real-time algorithms that satisfy Assumption 3.1.4 are given. Apart from the obvious choice of generalized Newton-type methods analyzed in Section 2.1.3, we list below two substantially different algorithms that are covered by the framework under analysis.

Alternating direction method of multipliers

Although the contraction estimates derived in Section 3.1.2 can be applied to general parametric nonconvex programs, we focus in the following on parametric convex programs that arise from (2.76) and can be formulated as follows:

$$\begin{aligned} \min_{v,w} \quad & \hat{f}(v) + \hat{g}(w) \\ \text{s.t.} \quad & Av + Bw + \hat{C}x = c, \end{aligned} \tag{3.13}$$

with variables $v \in \mathbb{R}^{n_v}$ and $w \in \mathbb{R}^{n_w}$. Here we have introduced the matrices $A \in \mathbb{R}^{n_c \times n_v}$ and $B \in \mathbb{R}^{n_c \times n_w}$ and the vector $c \in \mathbb{R}^{n_c}$. Problems of this form arise from (2.76) when considering linear dynamics, expressed as $Av + Bw + \hat{C}x = c$ and convex constraints and objectives that can be included in $\hat{f}(v) + \hat{g}(w)$ by means of indicator functions.

We regard the alternating direction method of multipliers (ADMM) and report a slightly adapted version of the result from [Nishihara et al., 2015] that leads to Q-linear convergence to a solution $\bar{z}(x) := (\bar{v}(x), \bar{w}(x), \bar{t}(x))$. The ADMM variant analyzed in [Nishihara et al., 2015] is described in Algorithm 1. In order to make use of the Q-linear convergence result in [Nishihara et al., 2015], we make the following assumption:

Assumption 3.1.7. *Assume that $\hat{f} : \mathbb{R}^{n_v} \to \mathbb{R}$ is strongly convex and Lipschitz, $\hat{g} : \mathbb{R}^{n_w} \to \mathbb{R} \cup \{+\infty\}$ is convex, A is invertible and B has full column rank. Moreover, assume that there exists a nonempty closed set X, such that, for any $x \in X$,* (3.13) *has a solution.*

In order to apply Theorem 3.1.6 to (3.13), we regard $z := (v, w, t)$, concatenating primal and dual variables, and we denote the corresponding semi-norm as

$$\|z\|_P^2 := \begin{bmatrix} Bw \\ t \end{bmatrix}^\top \tilde{P} \begin{bmatrix} Bw \\ t \end{bmatrix},$$

where $\tilde{P}$ is a symmetric positive-definite matrix. Under this setting, we can prove the following result.

Proposition 3.1.8 (Q-linear convergence)**.** *Let Assumption 3.1.7 and the Assumptions in [Nishihara et al., 2015, Theorem 6] hold and let $z := (w, v, t)$ denote the iterates generated by Algorithm 1. Then, there exists a symmetric positive semi-definite matrix P and a positive constant $0 \leq \tau < 1$ such that, for any z, the following holds:*

$$\|z_+ - \bar{z}(x)\|_P \leq \tau \|z - \bar{z}(x)\|_P. \tag{3.14}$$

Proof. Define $\mu := (Bw, t)$. Then, due to [Nishihara et al., 2015, Theorems 6-7], for any $\rho > 0$ and any $\alpha \in (0, 2)$, it holds that

$$\|\mu_+ - \bar{\mu}(x)\|_{\tilde{P}}^2 \leq \tau^2 \|\mu - \bar{\mu}(x)\|_{\tilde{P}}^2 \tag{3.15}$$

for some symmetric positive definite matrix $\tilde{P}$ and some $\tau < 1$ and $\bar{\mu}(x) = (B\bar{w}(x), \bar{t}(x))$, which implies

$$\|z_+ - \bar{z}(x)\|_P \leq \tau \|z - \bar{z}(x)\|_P, \tag{3.16}$$

where $P := \begin{bmatrix} 0 & 0 \\ 0 & \hat{B}^\top \tilde{P} \hat{B} \end{bmatrix}$ with $\hat{B} := \begin{bmatrix} B & 0 \\ 0 & \mathbb{I} \end{bmatrix}$. □

Proposition 3.1.9 (Q-linear convergence in Euclidean norm)**.** *Assume that $B = \mathbb{I}_{n_w}$ and $\alpha = 1$. Then, the iterate sequence $\{w^k, t^k\}$ generated by Algorithm 1 converges Q-linearly in the Euclidean norm.*

Proof. Since B is the identity matrix and $\alpha = 1$, the result follows directly from [Nishihara et al., 2015, Theorem 6]. □

Truncated sequential quadratic programming

In the following we outline a second simple Q-convergent strategy which relies on SQP with inexact solution of the QP subproblems. In particular, it is possible to show that carrying out a sufficiently high number of iterations of any Q-linearly

Algorithm 1 Alternating direction method of multipliers

```
input: functions $\hat{f}$ and $\hat{g}$, matrices $A$ and $B$, vector $c$, parameters $\rho$ and $\alpha$, and a parameter value $x \in X$.
output: approximate solution $(\tilde{v}, \tilde{w}, \tilde{t})$ at $x$
repeat:
    $v^{k+1} = \arg\min_{v} \hat{f}(v) + \frac{\rho}{2}\|Av + Bw^k + \hat{C}x - c + t^k\|^2$
    $w^{k+1} = \arg\min_{w} \hat{g}(w) + \frac{\rho}{2}\|\alpha Av^{k+1} - (1-\alpha)Bw^k + Bw + \alpha\hat{C}x - \alpha c + t^k\|^2$
    $t^{k+1} = t^k + \alpha Av^{k+1} - (1-\alpha)Bw^k + Bw^{k+1} + \alpha\hat{C}x - \alpha c$
until termination criterion met
```

Algorithm 2 Q-linearly convergent truncated SQP

```
input: initial primal-dual point $z^0$
output: approximate solution $z^0 \approx \bar{z}$
for $k = 0, \dots$ repeat:
    form subproblem (3.17) based on current iterate $z^k$
    set $\hat{z}^0 = z^k$
    solve (3.17) approximately with $j^\star$ iterations
    update linearization point $z^{k+1} \leftarrow \hat{z}^{j^\star}$
until termination criterion met
```

convergent method for convex quadratic programs in between linearizations the SQP iterates converge Q-linearly to a local solution. More specifically, at every iterate z, we regard the (potentially regularized) subproblem

$$0 \in F(z) + \tilde{F}'(z)(\zeta - z) + Cx + \mathcal{N}_K(\zeta) \tag{3.17}$$

with unknown ζ and denote its solution as $\bar{z}_{\mathrm{QP}}(z)$. We assume that we dispose of a method for solving convex QPs that generates iterates $\{\hat{z}^j\}$ with $j \in \mathbb{N}$, such that

$$\|\hat{z}^j - \bar{z}_{\mathrm{QP}}(z)\| \leq \zeta^j \|\hat{z}^0 - \bar{z}_{\mathrm{QP}}(z)\|. \tag{3.18}$$

Given a linearization point z, we construct the subproblem 3.17 and initialize the iterates of the QP solver as $\hat{z}^0 = z$. The subproblem is then approximately solved with $j^\star$ iterations and the linearization point is updated accordingly as $z_+ = \hat{z}^{j^\star}$.

The truncated SQP algorithm under consideration is summarized in Algorithm 2. This simple observation allows one not only to readily extend the applicability of virtually any algorithm for convex QPs to nonconvex programs. In fact it is also of particular interest in the context of real-time methods in the sense that it allows one to distribute the computational burden associated with the solution of the QP subproblems across iterations. Apart from the fact that this strategy can be used to obtain methods for nonconvex programs in an extremely simple way, carrying out inexact computations leads to suboptimality and, as we will see in the next lemma, worsens the contraction rate. This makes the approach potentially less interesting for "offline" optimization (or in any situation where we would like to solve the nonlinear programs to machine precision). However, in a real-time context, it gives access to algorithms for which contraction rate can be easily traded for cheaper per-iteration cost such that the approximate feedback law can be computed within the available time. Moreover, being the iterates Q-linearly convergent, the results from Lemma 3.1.5 and Theorem 3.1.6, together with the stability guarantees of the system-optimizer dynamics in NMPC that will be introduced in Section 3.2 can be easily applied.

Lemma 3.1.10. *For a fixed value of x, regard a solution $\bar{z}$ to* (3.4) *and assume that the contraction estimate*

$$\|\bar{z}_{\mathrm{QP}}(z) - \bar{z}\| \leq \kappa \|z - \bar{z}\| \tag{3.19}$$

holds for any $z \in \mathcal{B}(\bar{z}, r_z)$. Assume that, for any z and any $\hat{z}^0$, applying j inner iterations, the contraction

$$\|\hat{z}^j - \bar{z}_{\mathrm{QP}}(z)\| \leq \zeta^j \|\hat{z}^0 - \bar{z}_{\mathrm{QP}}(z)\|, \quad \text{for} \quad j = 0, 1, \ldots \tag{3.20}$$

holds. Moreover, assume that $j^\star > \left\lceil \log_\zeta \left(\frac{1-\kappa}{1+\kappa}\right) \right\rceil$, $\hat{z}^0 = z$ and $z_+ = z^{j^\star}$. Then, for any $z \in \mathcal{B}(\bar{z}, r_z)$, the following holds:

$$\|z_+ - \bar{z}\| \leq \tilde{\kappa} \|z - \bar{z}\|, \tag{3.21}$$

where $\tilde{\kappa} = (\zeta^j + \zeta^j \kappa + \kappa) < 1$.

Proof. The following holds:

$$\begin{aligned}
\|z_+ - \bar{z}\| \quad & \leq \|z_+ - \bar{z}_{\mathrm{QP}}(z)\| + \|\bar{z}_{\mathrm{QP}}(z) - \bar{z}\| \\
& = \|z^j - \bar{z}_{\mathrm{QP}}(z)\| + \|\bar{z}_{\mathrm{QP}}(z) - \bar{z}\| \\
& \leq \zeta^j \|\hat{z}^0 - \bar{z}_{\mathrm{QP}}(z)\| + \|\bar{z}_{\mathrm{QP}}(z) - \bar{z}\| \\
& \leq \zeta^j \|\hat{z}^0 - \bar{z}\| + \zeta^j \|\bar{z}_{\mathrm{QP}}(z) - \bar{z}\| + \kappa \|z - \bar{z}\| \\
& \leq \zeta^j \|\hat{z}^0 - \bar{z}\| + \zeta^j \kappa \|z - \bar{z}\| + \kappa \|z - \bar{z}\| \\
& = \zeta^j \|z - \bar{z}\| + \zeta^j \kappa \|z - \bar{z}\| + \kappa \|z - \bar{z}\| \\
& = (\zeta^j + \zeta^j \kappa + \kappa) \|z - \bar{z}\|
\end{aligned}$$

from which we can see that, for $\zeta^j < \frac{1-\kappa}{1+\kappa}$, contraction takes place. Choosing $j^\star > \left\lceil \log_\zeta \left(\frac{1-\kappa}{1+\kappa}\right) \right\rceil$ we recover the contraction stated by the lemma. □

Remark 3.1.11. *Notice that the result in Lemma 3.1.10 can be easily generalized to the setting where the subproblems associated with a Q-linearly convergent method are approximately solved with an algorithm whose iterates converge Q-linearly. These methods need not be SQP-like methods and, in fact, need not even be methods to solve nonconvex programs.*

3.1.4 Considerations specific to NMPC

The results derived under Assumptions 3.1.2 and 3.1.4, apply to any real-time algorithm used to solve a parametric optimization problem of the form (3.3). In this section, we derive consequences of Lemma 3.1.5 and Theorem 3.1.6 specific to the setting where problem (3.3) stems from an OCP of the form (2.76) to be solved online in order to control a system. Although the results are not entirely surprising and have a strong connection with the error bounds proposed, for example, in [Zavala and Anitescu, 2010] and [Diehl, 2001], they can be useful in making practical considerations when designing NMPC controllers based on general real-time algorithms.

First, in the spirit of the considerations made in [Diehl, 2001, Corollary 5.10], we can use Lemma 3.1.5 to compute bounds on the numerical error incurred if, due to a disturbance at steady state, the state of the system changes from x to x_+.

Corollary 3.1.12. *Assume that the discrete-time system $x_+ = \psi(x, u)$ has reached a steady-state x_{ss} associated with the steady-state input u_{ss}, i.e., $x_{\text{ss}} = \psi(x_{\text{ss}}, u_{\text{ss}})$. Moreover, assume that the iterates have converged to the optimal solution $\bar{z}(x_{\text{ss}})$. Then, if, after a disturbance, the state of the system is $x_+ \in \mathcal{B}(x_{\text{ss}}, r_x) \cap X$, after one iteration, the following inequality holds:*

$$\|z_+ - \bar{z}(x_+)\|_P \leq \gamma\kappa \|x_+ - x_{\text{ss}}\| + \frac{\omega\gamma^2}{2}\|x_+ - x_{\text{ss}}\|^2. \tag{3.22}$$

Proof. The inequality trivially follows from the result of Lemma 3.1.5 setting $\|z - \bar{z}(x_{\text{ss}})\|_P = 0$. □

Corollary 3.1.12 shows that the numerical error will be of second-order in the state perturbation, when the method has quadratic convergence. Additionally, we might be interested in analyzing the dependence of the derived contraction estimates on the underlying sampling time.

Definition 3.1.13. *Regard the following IVP:*

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{dt}}\psi(t;x,u) &= \phi(\psi(t;x,u),u), \\ \psi(0;x,u) &= x, \end{aligned} \tag{3.23}$$

where x denotes the initial conditions, u the constant input applied over the interval $[0,T]$ and ϕ defines the continuous-time dynamics. Here, with a slight abuse of notation, we have added an argument to ψ in order to explicitly state its dependency on the sampling time.

In the following, we will assume that x_+ is associated with the solution to 3.23, i.e.,

$$x_+ = \psi(T;x,u). \tag{3.24}$$

Assumption 3.1.14. *Assume that $0 \in X$ and, without loss of generality, let the origin $x = 0$, be a steady-state of the discrete time system x associated with the steady-state input $u = 0$:*

$$\psi(T;0,0) = 0. \tag{3.25}$$

Assume that the optimal solution to (2.76) at the origin yields $\bar{z}(0) = 0$. Let T denote the sampling time and let positive constants $0 < \eta < \infty$ and $0 < \theta < \infty$ exist, such that, for any x in X and any z in $\mathcal{B}(\bar{z}(x), r_z)$, the following inequality holds:

$$\|x_+ - x\| \leq T\eta \|x\| + T\theta \|z - \bar{z}(x)\| . \tag{3.26}$$

Remark 3.1.15. *Assumption 3.1.14 can be justified using the arguments reported in Appendix A.1.*

Combining Assumption 3.1.14 and the inequality from Theorem 3.1.6, we can state the following result, where we assume $\omega = 0$, for the sake of simplicity:

Corollary 3.1.16. *Let Assumptions 3.1.2, 3.1.4 and 3.1.14 be satisfied. Moreover, assume that $\omega = 0$ and $P = \mathbb{I}_{n_z}$. Then, for any z, such that $\|z - \bar{z}(x)\| \leq r_z$, and any $x, x_+ \in X$, with x additionally satisfying*

$$\|x\| \leq \hat{r}_x^s := \frac{r_x^s - T\theta r_z}{T\eta}, \tag{3.27}$$

the following holds:

$$\|z_+ - \bar{z}(x_+)\| \leq r_z. \tag{3.28}$$

Proof. Using Assumption 3.1.14, we obtain, that the condition

$$\|x\| \leq \hat{r}_x^s \tag{3.29}$$

implies that $\|x_+ - x\| \leq r_x^s$. Then, the result follows directly from Theorem 3.1.6. ☐

This shows that, for a sufficiently small initial numerical error $\|z - \bar{z}(x)\|$, we can guarantee that, as long as the state remains in a neighborhood of the origin, the numerical error will be bounded by r_z. Moreover, the size of this neighborhood increases as we shrink the sampling time T.

Corollary 3.1.17. *Let Assumptions 3.1.2, 3.1.4 and 3.1.14 hold and let* $P = \mathbb{I}_{n_z}$. *Then, for any* z *such that* $\|z - \bar{z}(x)\| = O\left(T\right)$, *and any* $x, x_+ \in X$ *such that* $\|x_+ - x\| \leq r_x^s$, *the following holds:*

$$\|z_+ - \bar{z}(x_+)\| = O\left(T\right). \tag{3.30}$$

Moreover, if $\kappa = 0$ *and* $\|z - \bar{z}(x)\| = O\left(T^2\right)$, *then* $\|z_+ - \bar{z}(x_+)\| = O\left(T^2\right)$.

Proof. The results follow directly from Lemma 3.1.5 by using Assumption 3.1.14 and the fact that $\|z - \bar{z}(x)\| = O\left(T\right)$ or $\|z - \bar{z}(x)\| = O\left(T^2\right)$, respectively. ☐

The results from Corollary 3.1.17 relate the magnitude of the numerical error to the sampling time T and are analogous to the ones in [Zavala and Anitescu, 2010]. We intentionally report them here to explicitly make a connection between the two frameworks and show that they hold for general real-time methods, too.

3.1.5 Illustrative example

In this section, we focus on an illustrative example involving a Q-linearly convergent method used as a real-time one. Since results related to Newton-type methods are relatively widespread in the literature, e.g., in [Diehl et al., 2007], [Tran-Dinh et al., 2012] and [Zavala and Anitescu, 2010], we intentionally focus on a different algorithm to highlight the generality of the results derived in Section 3.1.2.

We will use the real-time version of Algorithm 1, which we will refer to as RT-ADMM, to compute an approximate linear-quadratic MPC feedback policy for a permanent magnet synchronous machine (PMSM).

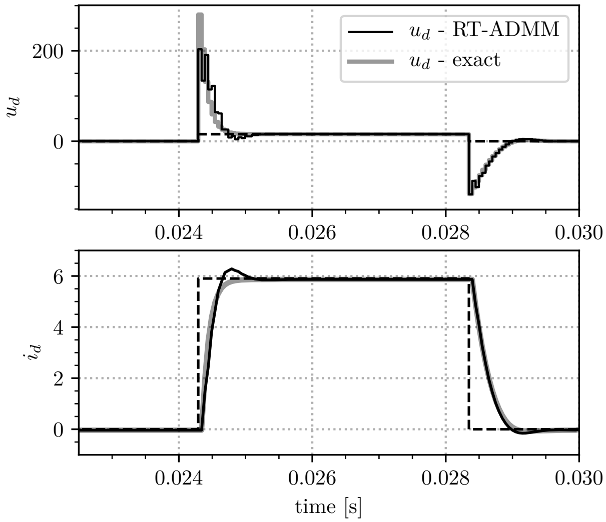


Figure 3.3: Comparison of closed-loop trajectories obtained with exact (thick, grey) and real-time (thin, black) feedback policies. A single iteration of RT-ADMM (real-time ADMM) is carried out per sampling time. Although a suboptimal solution is used to control the system, the closed-loop trajectories are very similar.

The dynamics of the system under consideration can be modelled using a linear ordinary differential equation as follows:

$$\begin{aligned} \dot{i}_d &= \frac{1}{L_d}\left(-R_s i_d + \omega L_q i_q + u_d\right) \\ \dot{i}_q &= \frac{1}{L_q}\left(-R_s i_q - \omega L_d i_d - \omega \psi_{\mathrm{pm}} + u_q\right), \end{aligned} \tag{3.31}$$

where i_d, i_q, u_d and u_q denote the currents and voltages, respectively, in the dq frame. The stator resistance is denoted by R_s, the inductances by L_d and L_q, the constant flux due to the permanent magnets by ψ_{pm} and, finally, ω represents the constant rotor speed. We formulate a tracking optimal control

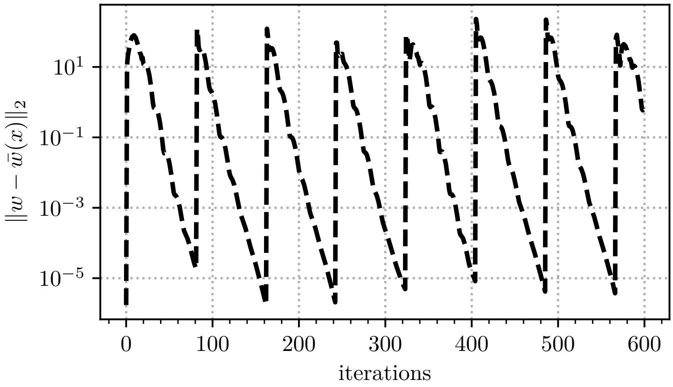


Figure 3.4: Numerical error incurred during the closed-loop simulation. The plot shows the Euclidean norm of the difference between the approximate primal solution w and the exact one $\bar{w}(x)$ during a part of the closed-loop simulation under consideration. Although the numerical error experiences peaks every time that the reference is changed, contraction is preserved and the suboptimality decreases with subsequent iterations until the next reference step.

problem as follows:

$$\begin{aligned}
\min_{\substack{s_0,\ldots,s_N \\ u_0,\ldots,u_{N-1}}} \quad & \frac{T_\mathrm{f}}{2N}\sum_{i=0}^{N-1}\begin{bmatrix} s_i \\ u_i \\ 1\end{bmatrix}^\top H \begin{bmatrix} s_i \\ u_i \\ 1\end{bmatrix} + \frac{1}{2}\begin{bmatrix} s_N \\ 1\end{bmatrix}^\top H_N \begin{bmatrix} s_N \\ 1\end{bmatrix} \\
\text{s.t.} \quad & s_0 - x = 0, \\
& s_{i+1} = A_\mathrm{d} s_i + B_\mathrm{d} u_i + c_\mathrm{d}, \quad i = 0,\ldots,N-1, \\
& \|u_i\|^2 \leq u_{\max}^2, \qquad\qquad\quad i = 0,\ldots,N-1,
\end{aligned} \tag{3.32}$$

where s and u denote the state and input of the system, respectively, A_d, B_d and c_d define the discretized linear dynamics for a constant speed $\bar{\omega}$ of the system and where the matrices

$$H = \begin{bmatrix} Q & 0 & q \\ 0 & R & r \\ q^\top & r^\top & 1 \end{bmatrix} \quad \text{and} \quad H_N = \begin{bmatrix} Q_N & q_N \\ q_N^\top & 1 \end{bmatrix} \tag{3.33}$$

have been used to define the stage, and terminal cost. Notice that, in this context, assuming ω to be constant is in general a reasonable assumption that

	RT-ADMM-1	RT-ADMM-2	RT-ADMM-3	RT-ADMM-5
rel subopt.	4.82%	3.07%	0.76%	0.16%
CPU time [µs]	0.813	1.387	1.961	3.109

Table 3.1: Closed-loop relative suboptimality and average computation times in microseconds obtained with RT-ADMM-i, where i denotes the number of iterations per sampling time. Notice that, in order to compute a solution to the subproblems with a tolerance of 10^{-4}, about 30 ADMM iterations are needed, leading to an average computation time of 17 µs.

is often used in practice due to the fact that electrical and mechanical dynamics of the system have very different time constants.

We eliminate the equality constraints in (3.32) and split the resulting problem by choosing $A = -B = \mathbb{I}_{Nn_u}$ and defining $\hat{g}$ in (3.13) to be the indicator function associated with the convex inequality constraints in (3.32). In this way, Step 1 of Algorithm 1 boils down to the solution of a linear system with precomputed Cholesky factors, and Step 2 involves the computationally inexpensive projection onto the convex set defined by the voltage constraints. The algorithm has been implemented in C using the high-performance linear algebra package `BLASFEO` [Frison et al., 2018]. All the experiments have been run on an Intel Core i7-7560U CPU.

We perform a closed-loop simulation where we want to track a time-varying reference and compare the control performance obtained with RT-ADMM and using the exact solution to problem (3.32). Figure 5.6 compares a section of the closed-loop trajectories obtained with RT-ADMM with a single iteration and with the exact solution, where we can see that, RT-ADMM leads to a limited degradation of the control performance. In particular, the closed-loop relative suboptimality with respect to the cost obtained with the exact MPC feedback policy, is shown in Table 4.1 for different numbers of ADMM iterations per sampling time. Finally, Figure 3.4 shows the deviation of the approximate solution computed by RT-ADMM from the exact one. Although large spikes can be seen whenever the reference is changed, the numerical error keeps contracting and the closed-loop trajectories do not seem to be heavily affected.

3.1.6 Conclusions

In this section, we presented contraction estimates for general real-time algorithms in the context of parametric optimization and NMPC in particular. Under the assumption of strong regularity of the underlying parametric optimization problems and Q-linear or Q-quadratic convergence of the optimization algorithm in some semi-norm, we provided generic numerical error bounds that can be applied to a broad class of methods. With the help of a numerical benchmark, we showed that the proposed approach can drastically reduce the computational burden associated with the implementation of MPC with moderate performance degradation.

In the rest of the chapter, we will exploit the contraction estimates derived in this section in order to study the stability of the system-optimizer dynamics in NMPC.

3.2 Asymptotic stability of the system-optimizer dynamics

The contraction estimate in Lemma 3.1.5 provides insights on the behavior of the numerical error, measured with an appropriate norm or semi-norm, as the parameter of the parametric nonconvex programs undergoes a general update. As such, the value of the parameter needs not be driven by dynamics of the system. In this and in the following section, we focus on the interaction arising between the optimizer and the system to be controlled. In fact in the most interesting setting the parameter x will take on the values determined by system's evolution under the approximate feedback $u = M_{u,z}z$.

3.2.1 System and optimizer dynamics

In order to study the interaction between the system to be controlled and the optimizer, we will first formally define their dynamics and describe the assumptions required for the stability analysis proposed.

System dynamics

The system under control obeys the following sampled-feedback closed-loop dynamics:

Definition 3.2.1 (System dynamics). *Let the following differential equation describe the dynamics of the system controlled using a constant input* u_0*:*

$$\begin{aligned} \frac{d\psi}{dt}(t;x_0,u_0) &= \phi(\psi(t;x_0,u_0),u_0), \\ \psi(0;x_0,u_0) &= x_0. \end{aligned} \tag{3.34}$$

Here $\psi : \mathbb{R}\times\mathbb{R}^{n_x}\times\mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ *describes the trajectories of the system,* x_0 *denotes the state of the system at a given sampling instant and* u_0 *the corresponding constant input. We will refer to the strictly positive parameter* $T > 0$ *as the sampling time associated with the corresponding discrete-time system*

$$x_{\text{next}} = \psi(T;x,u). \tag{3.35}$$

We will assume that a slightly tailored type of Lyapunov function is available for the closed-loop system controlled with a specific policy.

Assumption 3.2.2. *Let* $\bar{u} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$*, and let* $V : \mathbb{R}^{n_x} \to \mathbb{R}$ *be a continuous function. Let* $\bar{V}$ *be a strictly positive constant and define*

$$X_{\bar{V}} := \{x : V(x) \leq \bar{V}\}. \tag{3.36}$$

Assume that there exist positive constants a_1, a_2, a_3, T_0 *and* $q \in [1,\infty)$ *such that, for any* $x \in X_{\bar{V}}$ *and any* $T \leq T_0$*, the following hold:*

$$a_1\|x\|^q \leq V(x) \leq a_2\|x\|^q, \tag{3.37a}$$

$$V(\psi(T;x,\bar{u}(x))) - V(x) \leq -T\cdot a_3\|x\|^q. \tag{3.37b}$$

Notice that Assumption 3.2.2, for a fixed T boils down to the standard assumption for exponential asymptotic stability (see, e.g., Theorem 2.21 in [Rawlings et al., 2017]). Moreover, the dependency on T in (3.37b) can be justified, for example, by assuming that a continuous-time Lyapunov function V_{c} exists such that $\frac{\mathrm{d}}{\mathrm{d}t}V_{\mathrm{c}}(x(t)) \leq -\underline{a}\|x\|^2$, for some positive constant $\underline{a}$ and that V is a sufficiently good approximation of V_{c} in the following sense.

Remark 3.2.3. *Regard the simpler case in which the system under consideration is linear time-invariant, i.e.,* $\dot{x}(t) = \phi(x(t),u(t)) = A_{\mathrm{c}}x(t) + B_{\mathrm{c}}u(t)$*. Its discretized counterpart reads* $x_{\text{next}} = A_{\mathrm{d}}x + B_{\mathrm{d}}u$*, where*

$$A_{\mathrm{d}} := \exp(A_{\mathrm{c}}T_{\mathrm{d}}), \quad B_{\mathrm{d}} := \left(\int_0^{T_{\mathrm{d}}} \exp(A_{\mathrm{c}}\tau)\mathrm{d}\tau\right)B_{\mathrm{c}}.$$

When controlling a discrete-time linear time-invariant system with the linear feedback policy $u(x) = K_{\mathrm{d}}x$*, we know that, if* $x^\top P_{\mathrm{d}}x$ *is a Lyapunov function for* *the resulting closed-loop system* $x_{\mathrm{next}} = (A_{\mathrm{d}} + B_{\mathrm{d}} K_{\mathrm{d}})x$*, then it must satisfy the following discrete-time Lyapunov equation:*

$$(A_{\mathrm{d}} + B_{\mathrm{d}} K_{\mathrm{d}})^{\top} P_{\mathrm{d}} (A_{\mathrm{d}} + B_{\mathrm{d}} K_{\mathrm{d}}) - P_{\mathrm{d}} + Q_{\mathrm{d}} = 0, \tag{3.38}$$

for some positive-definite Q_{d}*. It is easy to show that, if the discretization time* T_{d} *is sufficiently small, then* $x^{\top} P_{\mathrm{d}} x$ *is a Lyapunov function for the continuous-time closed-loop system* $\dot{x}(t) = (A_{\mathrm{c}} + B_{\mathrm{c}} K_{\mathrm{d}})x(t)$*, where we use the discrete-time gain* K_{d}*. In particular, it suffices to show that a positive-definite matrix* Q_{c} *exists such that the following continuous-time Lyapunov equation is satisfied:*

$$(A_{\mathrm{c}} + B_{\mathrm{c}} K_{\mathrm{d}})^{\top} P_{\mathrm{d}} + P_{\mathrm{d}} (A_{\mathrm{c}} + B_{\mathrm{c}} K_{\mathrm{d}}) + Q_{\mathrm{c}} = 0. \tag{3.39}$$

To this end, we note that $A_{\mathrm{d}} = \mathbb{I} + T_{\mathrm{d}} A_{\mathrm{c}} + O(T_{\mathrm{d}}^2)$ *and* $B_{\mathrm{d}} = T_{\mathrm{d}} B_{\mathrm{c}} + O(T_{\mathrm{d}}^2)$*, such that we obtain*

$$\begin{aligned}&\left(\frac{A_{\mathrm{d}} - \mathbb{I}}{T_{\mathrm{d}}} + O(T_{\mathrm{d}}) + \left(\frac{B_{\mathrm{d}}}{T_{\mathrm{d}}} + O(T_{\mathrm{d}})\right) K_{\mathrm{d}}\right)^{\top} P_{\mathrm{d}} \\ &\qquad + P_{\mathrm{d}} \left(\frac{A_{\mathrm{d}} - \mathbb{I}}{T_{\mathrm{d}}} + O(T_{\mathrm{d}}) + \left(\frac{B_{\mathrm{d}}}{T_{\mathrm{d}}} + O(T_{\mathrm{d}})\right) K_{\mathrm{d}}\right) + Q_{\mathrm{c}} = 0\end{aligned}$$

and, multiplying by T_{d}*,*

$$(A_{\mathrm{d}} - \mathbb{I} + B_{\mathrm{d}} K_{\mathrm{d}})^{\top} P_{\mathrm{d}} + P_{\mathrm{d}} (A_{\mathrm{d}} - \mathbb{I} + B_{\mathrm{d}} K_{\mathrm{d}}) + E^{\top} E = -T_{\mathrm{d}} Q_{\mathrm{c}}, \tag{3.40}$$

where $E = O(T_{\mathrm{d}})$*. Let* $\tilde{A}_{\mathrm{d}} := A_{\mathrm{d}} + B_{\mathrm{d}} K_{\mathrm{d}}$*. Simplifying, we obtain*

$$\begin{aligned}&(A_{\mathrm{d}} - \mathbb{I} + B_{\mathrm{d}} K_{\mathrm{d}})^{\top} P_{\mathrm{d}} + P_{\mathrm{d}} (A_{\mathrm{d}} - \mathbb{I} + B_{\mathrm{d}} K_{\mathrm{d}}) + E^{\top} E \\ &\qquad = \tilde{A}_{\mathrm{d}}^{\top} P_{\mathrm{d}} + P_{\mathrm{d}} \tilde{A}_{\mathrm{d}} - 2P_{\mathrm{d}} + E^{\top} E \\ &\qquad \preceq \tilde{A}_{\mathrm{d}}^{\top} P_{\mathrm{d}} + P_{\mathrm{d}} \tilde{A}_{\mathrm{d}} - 2P_{\mathrm{d}} + E^{\top} E + (\tilde{A}_{\mathrm{d}} - \mathbb{I})^{\top} P_{\mathrm{d}} (\tilde{A}_{\mathrm{d}} - \mathbb{I}) \\ &\qquad = \tilde{A}_{\mathrm{d}}^{\top} P_{\mathrm{d}} \tilde{A}_{\mathrm{d}} - P_{\mathrm{d}} + E^{\top} E = -Q_{\mathrm{d}} + E^{\top} E,\end{aligned}$$

where we have exploited the fact that $(\tilde{A}_{\mathrm{d}} - \mathbb{I})^{\top} P_{\mathrm{d}} (\tilde{A}_{\mathrm{d}} - \mathbb{I}) \succeq 0$*. Due to the fact that* $E^{\top} E = O(T_{\mathrm{d}}^2)$ *and* $Q_{\mathrm{d}} \succ 0$*, we obtain that* $-T_{\mathrm{d}} Q_{\mathrm{c}} \preceq -Q_{\mathrm{d}} + O(T_{\mathrm{d}}^2)$*, and, for any sufficiently small discretization time* T_{d}*, there must exist a positive-definite* Q_{c} *such that the continuous-time Lyapunov equation* (3.39) *is satisfied. Finally, with similar arguments it is possible to show that, if* T_{d} *is small enough, for any sufficiently small sampling time* T*,* $x^{\top} P_{\mathrm{d}} x$ *is a valid Lyapunov function for the closed-loop system* $x_{\mathrm{next}} = (A_s + B_s K_{\mathrm{d}})x$*, where*

$$A_s := \exp(A_{\mathrm{c}} T),\ B_s := \left(\int_0^T \exp(A_{\mathrm{c}} \tau) \mathrm{d}\tau\right) B_{\mathrm{c}}.$$

Moreover, we make the following assumption which establishes additional regularity properties of the Lyapunov function.

Assumption 3.2.4. *Assume that $V^{\frac{1}{q}}$, where $q \in [1, \infty)$ is the constant introduced in Assumption 3.2.2, is Lipschitz continuous over $X_{\bar{V}}$, i.e., there exists a positive constant $\tilde{\mu}$ such that*

$$|V(x')^{\frac{1}{q}} - V(x)^{\frac{1}{q}}| \leq \tilde{\mu}\|x' - x\|, \tag{3.41}$$

$\forall x', x \in X_{\bar{V}}$.

Remark 3.2.5. *Notice that a sufficient condition for Assumption 3.2.4 to hold is that V is Lipschitz continuous over $X_{\bar{V}}$ and that $V^{\frac{1}{q}}$ is Lipschitz continuous at $x = 0$. These conditions are satisfied, for example, by Lyapunov functions which are twice continuously differentiable at the origin if $q = 2$ or simply Lipschitz continuous at the origin if $q = 1$.*

Remark 3.2.6. *If we take V to be the value function of the underlying MPC nonconvex program, results on the inherent robustness and stability of suboptimal MPC exist even for the case where V is discontinuous (see, e.g., [Allan et al., 2017]). However, in this context we deliberately choose to assume that stronger properties of the Lyapunov function hold, in order to analyze the system-optimizer interaction to a deeper level of detail.*

In particular, in the setting of [Scokaert et al., 1999], [Pannocchia et al., 2011], and [Allan et al., 2017], no regularity assumptions are required for the optimal solution and optimal value function, which are even allowed to be discontinuous. However, a decrease in the objective function is required in order for the optimizer's iterates to be accepted. This condition is in general difficult to satisfy given that commonly used numerical methods do not generate feasible iterates and, for this reason, it is not easy to enforce decrease in the objective function. Although robust stability could still be guaranteed by shifting the warmstart (cf. [Allan et al., 2017]), the improved iterates might be rejected unnecessarily. Moreover, the optimizer's dynamics are completely neglected. With respect to [Scokaert et al., 1999], [Pannocchia et al., 2011], and [Allan et al., 2017], we propose in this chapter an analysis that, although requires stronger assumptions on the properties of the optimal solution, incorporates knowledge on the optimizer's dynamics and does not require a decreasing cost.

The following proposition provides asymptotic stability of the closed-loop system obtained using the feedback policy $\bar{u}$.

Proposition 3.2.7 (Lyapunov stability)**.** *Let Assumption 3.2.2 hold. Then, for any $T \leq T_0$, the origin is an exponentially asymptotically stable equilibrium with region of attraction $X_{\bar{V}}$ for the closed-loop system $x_{\text{next}} = \psi(T; x, \bar{u}(x))$.*

Proof. Due to Assumption 3.2.2, the function V is a valid Lyapunov function for the closed-loop dynamics for any $T \leq T_0$. $\square$

The Lyapunov function defined in Assumption 3.2.2 guarantees that, if the ideal policy $\bar{u}$ is employed, the resulting closed-loop system is (exponentially) asymptotically stable. In the following, we define the dynamics of the optimizer used to numerically compute approximations of $\bar{u}(x)$ for a given state x.

Optimizer dynamics

We will assume that we dispose of a numerical method that defines what we will call the *optimizer* (or more generally a *solver*) that, for a given x, can compute a vector $\bar{z}(x)$ from which we can compute $\bar{u}(x)$ through a linear map.

Assumption 3.2.8. *Assume that there exists a function* $\bar{z} : X_{\bar{V}} \to \mathbb{R}^{n_z}$ *and a matrix* $M_{u,z}$ *such that, for any* $x \in X_{\bar{V}}$*, the following holds:*

$$\bar{u}(x) = M_{u,z}\bar{z}(x). \tag{3.42}$$

For simplicity of notation, we will assume further that $\|M_{u,z}\| = 1$.

Definition 3.2.9 (Optimizer dynamics)**.** *Let the following discrete-time system describe the dynamics of the optimizer*

$$z_+ = \varphi(\psi(T; x, M_{u,z}z), z), \tag{3.43}$$

where $\varphi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$ *and where* z *represents the state of the optimizer.*

Remark 3.2.10. *Notice that the optimizer dynamics* (3.43) *make use of the current approximate solution* z *and a forward-simulated state* $x_+ = \psi(T; x, M_{u,z}z)$*. This setting corresponds, for example, to the case where a real-time iteration is carried out by solving the linearized generalized equation (or, loosely speaking, the QP associated with it)*

$$0 \in F(z) + \nabla F(z)^\top (z_+ - z) + Cx_+ + \mathcal{N}_K(z_+), \tag{3.44}$$

with unknown z_+ *and linearization point* z *and where the parameter* $x_+ = \psi(T; x, M_{u,z}z)$ *is used. This amounts to assuming that either a perfect prediction* x_+ *of the system's state is available ahead of time, or that instantaneous feedback can be delivered to the system. In both cases, small perturbations introduced by either model mismatch or feedback delay could be introduced explicitly. This goes however beyond the scope of the present work (although inherent robustness of continuous Lyapunov functions would provide the necessary properties to guarantee stability under sufficiently small perturbations).*

In order to be able to leverage a contraction estimate similar to the one from Section 3.1, we will make the two following assumptions.

Assumption 3.2.11 (Lipschitz continuity)**.** *Assume that there exist positive constants* $\hat{r}_x$ *and* σ *such that, for any* $x \in X_{\bar{V}}$ *and any* $x' \in \mathcal{B}(x, \hat{r}_x)$*, the following holds*

$$\|\bar{z}(x') - \bar{z}(x)\| \leq \sigma \|x' - x\|. \tag{3.45}$$

Moreover, we assume that $\bar{z}(0) = 0$*.*

Assumption 3.2.12 (Contraction)**.** *There exists a radius* $\hat{r}_z > 0$ *and a positive constant* $\hat{\kappa} < 1$ *such that, for any* $x \in X_{\bar{V}}$ *and any* $z \in \mathcal{B}(\bar{z}(x), \hat{r}_z)$*, the optimizer dynamics produce* z_+ *such that*

$$\|z_+ - \bar{z}(x)\| \leq \hat{\kappa} \|z - \bar{z}(x)\|. \tag{3.46}$$

The following lemma, based on Lemma 3.1.5, provides a way of quantifying the perturbation to the nominal contraction (3.46) due to changes in the value of x across iterations of the optimizer.

Lemma 3.2.13. *Let Assumptions 3.2.11 and 3.2.12 hold. Then there exist strictly positive constants* r_z *and* r_x *such that, for any* x *in* $X_{\bar{V}}$*, any* z *in* $\mathcal{B}(\bar{z}, r_z)$*, and any* x' *in* $\mathcal{B}(x, r_x)$*, it holds that*

$$\|z_+ - \bar{z}(x')\| \leq \hat{\kappa} \|z - \bar{z}(x)\| + \sigma\hat{\kappa} \|x' - x\|. \tag{3.47}$$

Proof. The result is a special case of Lemma 3.1.5. □

Notice that the assumptions made do not require that V is the optimal value function of a discretized OCP nor of an optimization problem in general. We require instead that it is a Lyapunov function with some additional properties according to Assumptions 3.2.2 and 3.2.4. Similarly, $\bar{z}$ needs not be the primal-dual solution to an optimization problem. We require instead that it is associated with the policy $\bar{u}$, i.e., for any $x \in X_{\bar{V}}$, $\bar{u}(x) = M_{u,z}\bar{z}(x)$, that it is Lipschitz continuous and that the optimizer (or "solver" in general) can generate Q-linearly contracting iterates that converge to $\bar{z}(x)$. This setting is formalized by Assumptions 3.2.8, 3.2.11 and 3.2.12.

Remark 3.2.14. *To make a more concrete connection with a classical setting, in NMPC we can often assume that* V *is the optimal value function of a discretized version of an OCP of the form* (2.66)*. In this case, we can refer to the resulting parametric nonlinear program of the form* (2.23) *and to the generalized equation representing its first-order optimality conditions. Under proper regularity assumptions, we can then obtain that a single-valued localization*

$\bar{z}$ of the solution map of the generalized equation must exist. Under the same assumptions, we can usually prove Lipschitz continuity of $\bar{z}$ and Q-linear contraction of, for example, Newton-type iterations as shown in Section 2.1.3, hence satisfying Assumptions 3.2.11 and 3.2.12.

Similarly, using the standard argumentation introduced in Section 2.2.5, we obtain that the feedback policy associated with the global solution to the underlying nonlinear programs is stabilizing and the associated optimal value function is a Lyapunov function. In this way, Assumptions 3.2.2, 3.2.4 and 3.2.8 are satisfied. Note that, in this context, an aspect that remains somewhat unresolved is the fact that the standard argumentation for the stability analysis of NMPC requires that the global optimal solution is found by the optimizer (unless more sophisticated tools are brought into the picture, cf. [Allan et al., 2017]). Hence, in order to be able to identify V with the optimal value function it is necessary to assume that the single-valued localization $\bar{z}$ attains the global minimum for all $x \in X_{\bar{V}}$.

3.2.2 Combined system-optimizer dynamics

Proposition 3.2.7 and Lemma 3.2.13 provide key properties of the system and optimizer dynamics, respectively. In this section, we analyze the interaction between these two dynamical systems and how these properties are affected by such an interplay. To this end, let us define the following coupled system-optimizer dynamics.

Definition 3.2.15 (System-optimizer dynamics)**.** *Let the following discrete-time system describe the coupled system-optimizer dynamics:*

$$\begin{aligned} x_+ &= \psi(T; x, M_{u,z} z), \\ z_+ &= \varphi(\psi(T; x, M_{u,z} z), z) \end{aligned} \tag{3.48}$$

or, in compact form

$$\xi_+ = \Phi(T; \xi) \tag{3.49}$$

where $\xi := (x, z)$ and $\Phi : \mathbb{R} \times \mathbb{R}^{n_x + n_z} \rightarrow \mathbb{R}^{n_x + n_z}$.

Our ultimate goal is to prove that, for a sufficiently short sampling time T, the origin is a locally asymptotically stable equilibrium for (3.49).

Error contraction perturbation due to state evolution

In order to use the contraction from Lemma 3.2.13 in the context where $\|x' - x\|$ is determined by the evolution of the system to be controlled under the effect of the approximate policy, we will make a general assumption on the behavior of the closed-loop system in a neighborhood of the equilibrium and for a bounded value of the numerical error.

Assumption 3.2.16 (Lipschitz system dynamics)**.** *Assume that* $\phi(0,0) = 0$ *and that positive finite constants* $L_{\phi,x}$, $L_{\phi,u}$ *and* ρ *exist such that, for all* $x', x \in X_{\bar{V}+\rho} := \{x \in X : V(x) \leq \bar{V} + \rho\}$, *all* $u' = M_{u,z}z'$, $u = M_{u,z}z$, *with* $z', z \in \mathcal{B}(\bar{z}(x), r_z)$, *the following holds:*

$$\|\phi(x', u') - \phi(x, u)\| \leq L_{\phi,x}\|x' - x\| + L_{\phi,u}\|u' - u\|. \tag{3.50}$$

The following propositions establish bounds on the rate at which the state can change for given x and z.

Proposition 3.2.17. *Let Assumption 3.2.16 hold. Then, there exist positive finite constants* $L_{\psi,x}$, $L_{\psi,u}$ *and* $T_1 > 0$ *such that for all* $x \in X_{\bar{V}}$, *all* $z \in \mathcal{B}(\bar{z}(x), r_z)$ *and any* $T \leq T_1$, *the following holds:*

$$\|\psi(T; x, M_{u,z}z) - x\| \leq T \cdot (L_{\psi,x}\|x\| + L_{\psi,u}\|M_{u,z}z\|). \tag{3.51}$$

Moreover, for all $x \in X_{\bar{V}}$, *all* $u' = M_{u,z}z'$, $u = M_{u,z}z$, *such that* $z', z \in \mathcal{B}(\bar{z}(x), r_z)$ *and any* $T \leq T_1$, *the following holds:*

$$\|\psi(T; x, u') - \psi(T; x, u)\| \leq T \cdot L_{\psi,u}\|u' - u\|. \tag{3.52}$$

Proof. See Appendix A.1. □

Proposition 3.2.18. *Let Assumptions 3.2.11 and 3.2.16 hold. Then, there exists positive finite constants* η, θ *and* T_2, *such that for any* $x \in X_{\bar{V}}$, *any* $z \in \mathcal{B}(\bar{z}(x), r_z)$ *and any* $T \leq T_2$, *the following holds:*

$$\|\psi(T; x, u) - x\| \leq T \cdot (\eta\|x\| + \theta\|z - \bar{z}(x)\|). \tag{3.53}$$

Proof. Define $\eta := L_{\psi,x} + L_{\psi,u}\sigma$ and $\theta := L_{\psi,u}$. Due to Proposition 3.2.17 we have that

$$\|\psi(T; x, M_{u,z}z) - x\| \leq T \cdot (L_{\psi,x}\|x\| + L_{\psi,u}\|M_{u,z}z\|) \tag{3.54}$$

for any $T \leq T_1$ and, due to the assumption of strong regularity at the solution $\bar{z}(x)$, and the fact that $\bar{z}(0) = 0$, we can write

$$\|M_{u,z}z\| \leq \|\bar{z}(x)\| + \|z - \bar{z}\| \leq \sigma \|x\| + \|z - \bar{z}\|, \tag{3.55}$$

and the following holds:

$$\|\psi(T; x, u) - x\| \leq T(L_{\psi,x} + L_{\psi,u}\sigma) \|x\| + TL_{\psi,u} \|z - \bar{z}\|. \tag{3.56}$$

Finally, defining $T_2 := \min\{T_0, T_1\}$ completes the proof. □

Using the bound from Proposition 3.2.18 together with the contraction from Lemma 3.2.13 we obtain the following perturbed contraction.

Proposition 3.2.19. *Let Assumptions 3.2.2, 3.2.11, 3.2.12 and 3.2.16 hold. Moreover, define*

$$T_3' := \min\left\{\frac{r_x}{\eta r_{\bar{V}} + \theta r_z}, \frac{r_z(1-\hat{\kappa})}{\sigma\hat{\kappa}(\theta r_z + \eta r_{\bar{V}})}\right\}, \tag{3.57}$$

where $r_{\bar{V}} := \left(\frac{\bar{V}}{a_1}\right)^{\frac{1}{q}}$.

Then, for any x, z *such that* $x \in X_{\bar{V}}$ *and* $\|z - \bar{z}(x)\| \le r_z$ *and any* $T \le T_3 := \min\{T_3', T_2\}$*, the following holds:*

$$\|z_+ - \bar{z}(x_+)\| \le \kappa\|z - \bar{z}(x)\| + T\gamma\|x\|, \tag{3.58}$$

where

$$\kappa := \hat{\kappa}(1 + T_3\sigma\theta) < 1, \quad \gamma := \sigma\hat{\kappa}\eta. \tag{3.59}$$

Moreover, $\|z_+ - \bar{z}(x_+)\| \le r_z$*.*

Proof. Given that $\|z - \bar{z}(x)\| \le r_z$ and that, due to Assumption 3.2.16 and the fact that $T \le T_3 \le T_2$, we have $\|x_+ - x\| \le r_x$ for all $x \in X_{\bar{V}}$, we can apply the contraction from Theorem 3.2.13:

$$\|z_+ - \bar{z}(x_+)\| \le \hat{\kappa}\left\|z - \bar{z}(x)\right\| + \sigma\hat{\kappa}\left\|x_+ - x\right\|. \tag{3.60}$$

Applying the inequality from Proposition 3.2.18, we obtain

$$\|z_+ - \bar{z}(x_+)\| \le \kappa\|z - \bar{z}(x)\| + T\gamma\|x\|, \tag{3.61}$$

where

$$\kappa := \hat{\kappa}(1 + T_3\sigma\theta), \quad \gamma := \sigma\hat{\kappa}\eta. \tag{3.62}$$

Finally, due to Assumption the definition of T_3 in (3.57) we have that $\|z_+ - \bar{z}(x_+)\| \le r_z$ and $\kappa < 1$. ☐

Proposition 3.2.19 shows that under suitable assumptions and, in particular, for any $T \le T_3$, we can guarantee that the numerical error does not increase after one iteration of the optimizer.

In the following, we will make similar considerations for the behavior of $V(x)$ across iterations.

Lyapunov decrease perturbation due to numerical error

In the following, we analyze the impact of the fact that the approximate feedback policy $M_{u,z}z$ is used, instead of the exact one $M_{u,z}\bar{z}(x)$, on the nominal Lyapunov contraction. Throughout the rest of the section, we will make use of the following shorthand:

$$V_+(T;x,z) := V(\psi(T;x,M_{u,z}z)) \tag{3.63}$$

to denote the value taken by the optimal cost at the state reached applying the suboptimal control action $M_{u,z}z$ starting from x. Similarly, we introduce

$$E(x,z) := \|z - \bar{z}(x)\| \tag{3.64}$$

and

$$E_+(T;x,z) := \|\varphi(\psi(T;x,M_{u,z}z),z) - \bar{z}(\psi(T;x,M_{u,z}z))\| \tag{3.65}$$

to denote the numerical error attained at the "current" and next iteration of the optimizer, where the error is computed with respect to the exact solution associated with the "current" and next state of the system.

Proposition 3.2.20. *Let Assumptions 3.2.2, 3.2.8, 3.2.11, 3.2.12, and 3.2.16 hold. Then, there exists a finite positive constant* μ *such that, for any* $z \in \mathcal{B}(\bar{z}(x), r_z)$*, any* x *in* $X_{\bar{V}}$ *and any* $T \leq T_1$*, the following holds:*

$$V_+(T;x,z) \leq (1 - T\bar{a})V(x) + T\mu E(x,z), \tag{3.66}$$

where $\bar{a} := \frac{a_3}{a_2}$.

Proof. Assumption 3.2.2 implies that, for any $x \in X_{\bar{V}}$ and any $T \leq T_0$ the following holds:

$$\begin{aligned} V(\psi(T;x,M_{u,z}\bar{z}(x))) &\leq V(x) - Ta_3\|x\|^q \\ &\leq V(x) - T\frac{a_3}{a_2}V(x) \\ &= (1 - T\bar{a})V(x) \end{aligned} \tag{3.67}$$

Due to Assumption 3.2.4, for any $x', x \in X_{\bar{V}}$, we can write

$$\begin{aligned} |V(x') - V(x)| &\leq |(V(x')^{\frac{1}{q}})^q - (V(x)^{\frac{1}{q}})^q| \\ &= |(V(x')^{\frac{1}{q}} + V(x)^{\frac{1}{q}})(V(x')^{\frac{1}{q}} - V(x)^{\frac{1}{q}})| \\ &\leq 2\bar{V}^{\frac{1}{q}}\tilde{\mu}\|x' - x\| \end{aligned} \tag{3.68}$$

and defining $L_V := 2\bar{V}^{\frac{1}{q}}\tilde{\mu}$ we obtain

$$|V(x') - V(x)| \leq L_V \|x' - x\|. \tag{3.69}$$

Together with Assumption 3.2.16, this implies that we can write:

$$\begin{aligned}
V(\psi(T; x, M_{u,z}z)) - V(\psi(T; x, M_{u,z}\bar{z}(x)))& \\
&\leq |V(\psi(T; x, M_{u,z}z)) - V(\psi(T; x, M_{u,z}\bar{z}(x)))| \\
&\leq L_V \|\psi(T; x, M_{u,z}z) - \psi(T; x, M_{u,z}\bar{z}(x))\| \\
&\leq T L_{\psi,u} L_V \|z - \bar{z}(x)\|
\end{aligned}$$

which implies

$$V_+(T; x, z) \leq (1 - T\bar{a})V(x) + T\mu E(x, z), \tag{3.70}$$

where $\mu := L_V L_{\psi,u}$. □

Using Proposition 3.2.20 we can formulate Lemma 3.2.22 below which establishes positive invariance of the following set for the system-optimizer dynamics (3.48).

Definition 3.2.21 (Invariant set)**.** *Define the following set:*

$$\Sigma := \{(x, z) \in \mathbb{R}^{n_x + n_z} : V(x) \leq \bar{V}, \|z - \bar{z}(x)\| \leq \tilde{r}_z\},$$

where

$$\tilde{r}_z := \min\left\{r_z, \frac{\bar{a}\bar{V}}{\mu}\right\}. \tag{3.71}$$

Lemma 3.2.22 (Invariance of Σ)**.** *Let Assumptions 3.2.2, 3.2.4, 3.2.8, 3.2.11, 3.2.12, and 3.2.16 hold. Define*

$$T_4' := \frac{(1 - \kappa)\tilde{r}_z a_1^{\frac{1}{q}}}{\bar{V}^{\frac{1}{q}}\gamma}. \tag{3.72}$$

Then, for any $\xi \in \Sigma$ and any $T \leq T_4 := \min\{T_4', T_3\}$, it holds that $\xi_+ \in \Sigma$. Moreover, the following coupled inequalities hold:

$$\begin{aligned}
V_+(T; x, z) &\leq (1 - T\bar{a})V(x) + T\mu E(x, z), \\
E_+(T; x, z) &\leq T\hat{\gamma}V(x)^{\frac{1}{q}} + \kappa E(x, z),
\end{aligned} \tag{3.73}$$

where $\hat{\gamma} := \gamma / a_1^{\frac{1}{q}}$.

Proof. Given that $E(x,z) \le \tilde{r}_z \le r_z$ and $x \in X_{\bar{V}}$, we can apply the contraction from Proposition 3.2.20, such that

$$V_+(T;x,z) \le (1 - T\bar{a})V(x) + T\mu E(x,z), \tag{3.74}$$

holds. Moreover, due to the definition of $\tilde{r}_z$ in (3.71), we have that $V_+(T;x,z) \le \bar{V}$ since

$$\begin{aligned} V_+(T;x,z) &\le (1 - T\bar{a})V(x) + T\mu E(x,z) \\ &\le (1 - T\bar{a})\bar{V} + T\mu\tilde{r}_z \\ &\le (1 - T\bar{a})\bar{V} + T\mu\frac{\bar{a}\bar{V}}{\mu} \\ &\le \bar{V}. \end{aligned} \tag{3.75}$$

This implies that x_+ is in $X_{\bar{V}}$. Similarly, due to the fact that that $E(x,z) \le \tilde{r}_z \le r_z$ and $x \in X_{\bar{V}}$, we can apply the result from Proposition 3.2.18, which shows that

$$\|z_+ - \bar{z}(x_+)\| \le \kappa\|z - \bar{z}(x)\| + T\gamma\|x\| \tag{3.76}$$

and

$$\|z_+ - \bar{z}(x_+)\| \le r_z \tag{3.77}$$

must hold. Using Assumption 3.2.2 in Equation (3.76), we obtain

$$\|z_+ - \bar{z}(x_+)\| \le \kappa\|z - \bar{z}(x)\| + T\hat{\gamma}\left(V(x)\right)^{\frac{1}{q}}. \tag{3.78}$$

Moreover, due to (3.72), we have that $\|z_+ - \bar{z}(x_+)\| \le \tilde{r}_z$ for any $T \le T_4$. □

In principle, we could study the behavior of the coupled contraction estimate by looking at the “worst-case” dynamics associated with (3.73):

$$\begin{aligned} v_+ &= (1 - T\bar{a})v + T\mu e, \\ e_+ &= T\hat{\gamma}v^{\frac{1}{q}} + \kappa e, \end{aligned} \tag{3.79}$$

which, however, are not Lipschitz continuous at the origin $(v,e) = (0,0)$, for $q > 1$. Nonetheless, we can still reformulate (3.79) such that standard tools can be used to obtain important information about the behavior of the $V_+(T;x,z)$ and $E_+(T;x,z)$ under the combined contraction established in Lemma 3.2.22 and ultimately establish asymptotic stability of the system-optimizer dynamics.

3.2.3 Asymptotic stability of the system-optimizer dynamics

In the following, we derive the main asymptotic stability result, which relies on a reformulation of the worst-case dynamics (3.79).

Proposition 3.2.23. *Let Assumptions 3.2.2, 3.2.4, 3.2.8, 3.2.11, 3.2.12 and 3.2.16 hold. Moreover, let* $\hat{\mu} := L_{\phi,u} e^{T_1 L_{\phi,x}} \tilde{\mu}$. *Then, for any* $\xi \in \Sigma$ *and any* $T \leq T_4$, *we have* $\xi_+ \in \Sigma$ *and the following holds:*

$$\begin{aligned} V_+(T;x,z)^{\frac{1}{q}} &\leq (1-T\bar{a})^{\frac{1}{q}} V(x)^{\frac{1}{q}} + T\hat{\mu}E(x,z), \\ E_+(T;x,z) &\leq T\hat{\gamma}V(x)^{\frac{1}{q}} + \kappa E(x,z). \end{aligned} \tag{3.80}$$

Proof. The fact that $\xi_+ \in \Sigma$ is a direct consequence of Lemma 3.2.22. Moreover, due to Assumption 3.2.4, the following holds, for any $x \in X_{\bar{V}}$:

$$\begin{aligned} V(\psi(T;x,M_{u,z}z))^{\frac{1}{q}} \leq\; & V(\psi(T;x,M_{u,z}\bar{z}(x))^{\frac{1}{q}} \\ & + \tilde{\mu}\|\psi(T;x,M_{u,z}z) - \psi(T;x,M_{u,z}\bar{z}(x))\| \end{aligned}$$

and, using the nominal Lyapunov contraction and Proposition 3.2.18, we obtain that, for any $\xi \in \Sigma$, the following holds:

$$V(\psi(T;x,M_{u,z}z))^{\frac{1}{q}} \leq (1-T\bar{a})^{\frac{1}{q}} V(x)^{\frac{1}{q}} + T\hat{\mu}\|z-\bar{z}(x)\|,$$

where $\hat{\mu} := L_{\phi,u} e^{T_1 L_{\phi,x}} \tilde{\mu}$. □

Unlike (3.79), the worst-case dynamics associated with (3.80) are not only Lipschitz continuous, but can also be cast as a linear positive system. We define the following dynamical system based on (3.80).

Definition 3.2.24 (Auxiliary dynamics)**.** *We will refer to the linear time-invariant discrete-time dynamical system*

$$\begin{aligned} \nu_+ &= (1-T\bar{a})^{\frac{1}{q}}\nu + T\hat{\mu}\epsilon, \\ \epsilon_+ &= T\hat{\gamma}\nu + \kappa\epsilon, \end{aligned} \tag{3.81}$$

with states ν, $\epsilon \in \mathbb{R}$, *as auxiliary dynamics. Due to the definitions of* $\kappa, \hat{\mu}$ *and* $\hat{\gamma}$ *and Assumption 3.2.2,* (3.81) *is a positive system [Kaczorek, 2008].*

Remark 3.2.25. *Given the definition of the auxiliary dynamics in Definition 3.2.24, for any* $\xi \in \Sigma$, *if* $V(x)^{\frac{1}{q}} = \nu$ *and* $E(x,z) = \epsilon$, *then* $V_+(T;x,z)^{\frac{1}{q}} \leq \nu_+$ *and* $E_+(T;x,z) \leq \epsilon_+$. *For this reason, intuitively, we can study the stability of* *the auxiliary dynamics and infer stability properties of the combined system-optimizer dynamics. This concept will be later formalized with the explicit construction of a Lyapunov function for the system-optimizer dynamics in Theorem 3.2.28.*

We exploit properties of positive systems in order to construct an explicit linear Lyapunov function for the auxiliary dynamics which can be rewritten in the compact form

$$w_+ = A_a w, \tag{3.82}$$

where

$$A_a := \begin{bmatrix} (1 - T\bar{a})^{\frac{1}{q}} & T\hat{\mu} \\ T\hat{\gamma} & \kappa \end{bmatrix}, \tag{3.83}$$

and $w := (\nu, \epsilon)$. We will make use of the following result adapted from [Kaczorek, 2008].

Theorem 3.2.26 (Stability of positive systems). *A positive discrete-time linear system of the form*

$$w_+ = Aw, \tag{3.84}$$

where $A \in \mathbb{R}_+^{n \times n}$ *and* $w \in \mathbb{R}_+^n$ *is asymptotically stable if there exist a strictly positive vector* $\hat{w} \in \mathbb{R}_{++}^n$ *and a strictly positive constant* $\hat{d}$ *such that*

$$\max_{i=1,\dots,n} [\,(A^\top - \mathbb{I})\hat{w}\,]_i \leq -\,\hat{d}. \tag{3.85}$$

Moreover, the linear function $V_l(w) := \hat{w}^\top w$ *is a Lyapunov function for such system in* $\mathbb{R}_{\geq 0}^n$ *and the following holds:*

$$V_l(w_+) - V_l(w) \leq -\hat{d} \cdot \|w\|. \tag{3.86}$$

Proof. See Appendix A.2. □

Theorem 3.2.27. *The positive discrete-time linear system* (3.82) *is asymptotically stable if and only if the following condition is satisfied:*

$$T^2 \hat{\mu}\hat{\gamma} - (1 - \kappa)(1 - (1 - T\bar{a})^{\frac{1}{q}}) < 0, \tag{3.87}$$

which is satisfied for any sufficiently small sampling time $T \leq T_5 := \frac{\beta(1-\kappa)}{\hat{\mu}}$. *Moreover, the function* $V_l(w) := \hat{w}^\top w$, *where*

$$\hat{w} = \begin{bmatrix} 1 \\ \beta \end{bmatrix}, \quad \text{with} \quad \beta := \frac{1}{2}\frac{\bar{a}}{q\hat{\gamma}}, \tag{3.88}$$

is a Lyapunov function for (3.81) *in* $\mathbb{R}_{\geq 0}^2$.

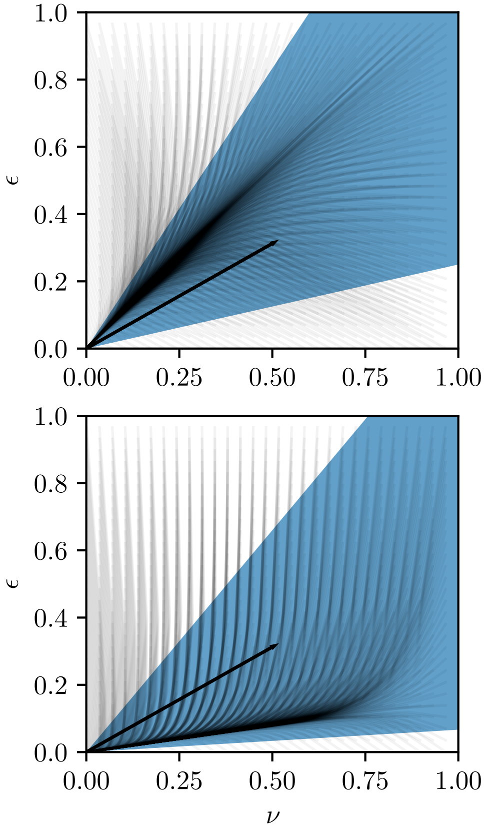


Figure 3.5: Trajectories of the auxiliary dynamics (3.81) for different initial conditions ($\kappa = 0.4$, $\bar{a} = 0.5$, $\hat{\gamma} = 0.2$, $\hat{\mu} = 0.1$) - $T = 1.0$ (top) and $T = 0.4$ (bottom). The black vector describes the direction defined by $\hat{w}$ as in Theorem 3.2.27, while the shaded area defines the cone that contains all the vectors that would satisfy (3.89), i.e., all the vectors $\hat{w}$ that define a valid Lyapunov function $V_l(w) = \hat{w}^\top w$.

Proof. In order to prove that $\hat{w}^\top w$ is a Lyapunov function for (3.81) it suffices to apply Theorem 3.2.26. The system of inequalities

$$\begin{aligned} -1 + (1 - T\bar{a})^{\frac{1}{q}} + T\hat{\gamma}\,\beta &< 0, \\ T\hat{\mu} + (\kappa - 1)\,\beta &< 0, \\ \beta &> 0 \end{aligned} \tag{3.89}$$

admits solutions if

$$T\frac{\hat{\mu}}{1-\kappa} < \beta < \frac{1 - (1 - T\bar{a})^{\frac{1}{q}}}{T\hat{\gamma}}. \tag{3.90}$$

This condition can always be satisfied for a sufficiently small T. In fact, it is easy to show that the limits for $T \to 0$ of the upper and lower bounds on β are 0 and $\frac{\bar{a}}{q\hat{\gamma}} > 0$, respectively, such that, by continuity, there must exist a strictly positive constant T_5, such that (3.87) is satisfied for any $T \leq T_5$. However, in order to make β independent of T, we note that, due to convexity, $1 - (1 - T\bar{a})^{\frac{1}{q}} \geq \frac{\bar{a}}{q}T$, for any $T \geq 0$. Using such lower bound we can simplify the upper bound in (3.90):

$$\beta < \frac{\frac{\bar{a}}{q}T}{\hat{\gamma}T} = \frac{\bar{a}}{q\hat{\gamma}} \leq \frac{1 - (1 - T\bar{a})^{\frac{1}{q}}}{T\hat{\gamma}}. \tag{3.91}$$

Choosing β to be half of this upper bound, i.e., $\beta := \frac{1}{2}\frac{\bar{a}}{q\hat{\gamma}}$, we obtain that (3.90) is satisfied for any $T \leq T_5 := \frac{\beta(1-\kappa)}{\hat{\mu}}$, which concludes the proof. □

Theorem 3.2.27 shows that (exponential) asymptotic stability of the auxiliary dynamics holds under the condition that the sampling time T satisfies inequality (3.87) for given $\hat{\mu}$, $\bar{a}$ and κ. Figure 3.5, illustrates the meaning of Theorem 3.2.27, by showing the trajectories of the auxiliary system in a phase plot, for fixed values of the parameters $\hat{\mu}, \bar{a}, \kappa$ and $\hat{\gamma}$, two different values of the sampling time T and for different initial conditions. In the following, we establish the main result of the section by exploiting the Lyapunov decrease established in Theorem 3.2.27 for the auxiliary dynamics to construct a Lyapunov function for the combined system-optimizer dynamics (3.49).

Theorem 3.2.28. *Let Assumptions 3.2.2, 3.2.4, 3.2.8, 3.2.11, 3.2.12 and 3.2.16 hold. Then, for any $T \leq \min\{T_4, T_5\}$, the origin is an asymptotically stable equilibrium with region of attraction Σ for the combined system-optimizer dynamics* (3.49)*. In particular, the function*

$$V_{\mathrm{so}}(\xi) := \hat{w}^\top \begin{bmatrix} V(x)^{\frac{1}{q}} \\ \|z - \bar{z}(x)\| \end{bmatrix}, \tag{3.92}$$

where $\hat{w}$ is defined according to Theorem 3.2.27, is a Lyapunov function in Σ for the system (3.49) *and the origin* $(x, z) = \xi = 0$.

Proof. We can derive an upper bound for $V_{\mathrm{so}}(\xi)$ as follows:

$$\begin{aligned} V_{\mathrm{so}}(\xi) &= V(x)^{\frac{1}{q}} + \beta\|z - \bar{z}(x)\| \\ &\leq a_2^{\frac{1}{q}}\|x\| + \beta\|z - \bar{z}(x)\| \\ &\leq a_2^{\frac{1}{q}}\|x\| + \beta(\|z\| + \|\bar{z}(x)\|) \\ &\leq (a_2^{\frac{1}{q}} + \sigma\beta)\|x\| + \beta\|z\| \\ &\leq \underbrace{\max\{a_2^{\frac{1}{q}} + \sigma\beta, \beta\}}_{=:\tilde{w}_2} \cdot (\|x\| + \|z\|) \\ &\leq \tilde{w}_2 \cdot (\|x\|_1 + \|z\|_1) = \tilde{w}_2 \cdot \|\xi\|_1 \\ &\leq \tilde{w}_2\sqrt{n_x + n_z} \cdot \|\xi\|. \end{aligned} \tag{3.93}$$

In order to derive a lower bound, we proceed as follows. Using the reverse triangle inequality and Lipschitz continuity of $\bar{z}(x)$, we obtain

$$\begin{aligned} V_{\mathrm{so}}(\xi) &\geq V(x)^{\frac{1}{q}} + \beta(\|z\| - \sigma\|x\|) \\ &\geq a_1^{\frac{1}{q}}\|x\| + \beta(\|z\| - \sigma\|x\|) \\ &= (a_1^{\frac{1}{q}} - \beta\sigma)\|x\| + \beta\|z\|. \end{aligned} \tag{3.94}$$

If $a_1^{\frac{1}{q}} - \beta\sigma > 0$, then we can readily compute a lower bound:

$$\begin{aligned} V_{\mathrm{so}}(\xi) &\geq (a_1^{\frac{1}{q}} - \beta\sigma)\|x\| + \beta\|z\| \\ &\geq \frac{a_1^{\frac{1}{q}} - \beta\sigma}{\sqrt{n_x}}\|x\|_1 + \frac{\beta}{\sqrt{n_z}}\|z\|_1 \\ &\geq \underbrace{\min\left\{\frac{a_1^{\frac{1}{q}} - \beta\sigma}{\sqrt{n_x}}, \frac{\beta}{\sqrt{n_z}}\right\}}_{=:\tilde{w}_{1,1}}(\|x\|_1 + \|z\|_1) \\ &\geq \tilde{w}_{1,1} \cdot \|\xi\|. \end{aligned} \tag{3.95}$$

If instead $a_1^{\frac{1}{q}} - \beta\sigma \leq 0$, we define the auxiliary lower bound

$$\hat{V}_{\mathrm{so}}(x,z) := a_1^{\frac{1}{q}} \|x\| + \beta \|z - \bar{z}(x)\|. \tag{3.96}$$

Since $V_{\mathrm{so}}(\xi) \geq \hat{V}_{\mathrm{so}}(x,z)$, if we can show that $\hat{V}_{\mathrm{so}}(x,z)$ can be lower bounded by a properly constructed function of z, we can use this function to lower bound $V_{\mathrm{so}}(\xi)$ too. To this end, we first observe that, since we assumed that $a_1^{\frac{1}{q}} - \beta\sigma \leq 0$, for any x such that $\|x\| \leq \frac{1}{\sigma}\|z\|$, we have that

$$\begin{aligned} \hat{V}_{\mathrm{so}}(x,z) &\geq (a_1^{\frac{1}{q}} - \beta\sigma)\|x\| + \beta\|z\| \\ &\geq \min_{x\,\mathrm{s.t.}\ \|x\| \leq \frac{1}{\sigma}\|z\|} (a_1^{\frac{1}{q}} - \beta\sigma)\|x\| + \beta\|z\| \\ &\geq \frac{a_1^{\frac{1}{q}}}{\sigma}\|z\|, \end{aligned} \tag{3.97}$$

where, for the minimization, we have used the fact that the objective $(a_1^{\frac{1}{q}} - \beta\sigma)\|x\| + \beta\|z\|$ is monotonically nonincreasing in $\|x\|$ such that the minimum is attained at the boundary of the interval for $\|x\| = \frac{1}{\sigma}\|z\|$. Similarly, for any x such that $\|x\| \geq \frac{1}{\sigma}\|z\|$, we can use the fact that

$$\hat{V}_{\mathrm{so}}(x,z) = a_1^{\frac{1}{q}}\|x\| + \beta\|z - \bar{z}(x)\| \geq a_1^{\frac{1}{q}}\|x\| \geq \frac{a_1^{\frac{1}{q}}}{\sigma}\|z\|.$$

Hence, we can conclude that

$$V_{\mathrm{so}}(\xi) \geq \hat{V}_{\mathrm{so}}(x,z) \geq \frac{a_1^{\frac{1}{q}}}{\sigma}\|z\| \tag{3.98}$$

for any x. Summing this last inequality and $V_{\mathrm{so}}(\xi) \geq a_1^{\frac{1}{q}}\|x\|$, we obtain

$$\begin{aligned} V_{\mathrm{so}}(\xi) &\geq \frac{a_1^{\frac{1}{q}}}{2}\|x\| + \frac{a_1^{\frac{1}{q}}}{2\sigma}\|z\| \\ &\geq \frac{a_1^{\frac{1}{q}}}{2\sqrt{n_x}}\|x\|_1 + \frac{a_1^{\frac{1}{q}}}{2\sigma\sqrt{n_z}}\|z\|_1 \\ &\geq \underbrace{\min\left\{\frac{a_1^{\frac{1}{q}}}{2\sqrt{n_x}}, \frac{a_1^{\frac{1}{q}}}{2\sigma\sqrt{n_z}}\right\}}_{=:\tilde{w}_{1,2}}(\|x\|_1 + \|z\|_1) \\ &\geq \tilde{w}_{1,2} \cdot \|\xi\|. \end{aligned} \tag{3.99}$$

Together with (3.95), we can define

$$\tilde{w}_1 := \begin{cases} \tilde{w}_{1,1}, & \text{if} \quad a_1^{\frac{1}{q}} - \beta\sigma > 0, \\ \tilde{w}_{1,2}, & \text{otherwise}, \end{cases} \tag{3.100}$$

and conclude that $V_{\mathrm{so}}(\xi) \geq \tilde{w}_1 \cdot \|\xi\|$. Finally, the Lyapunov decrease can be derived. For given x and z, let $\epsilon = E(x,z)$ and $\nu = V(x)^{\frac{1}{q}}$. Then the following holds.

$$\begin{aligned} V_{\mathrm{so}}(\xi_+) &= V_+(T;x,z)^{\frac{1}{q}} + \beta E_+(T;x,z) \\ &\overset{\text{Remark 3.2.25}}{\leq} \nu_+ + \beta\epsilon_+ \\ &\overset{\text{Theorem 3.2.26}}{\leq} \nu + \beta\epsilon - \hat{d} \cdot \|(\nu,\epsilon)\|_1 \\ &= V(x)^{\frac{1}{q}} + \beta E(x,z) - \hat{d} \cdot \|(\nu,\epsilon)\|_1 \\ &= V_{\mathrm{so}}(\xi) - \hat{d} \cdot (V(x)^{\frac{1}{q}} + \|z - \bar{z}(x)\|), \end{aligned} \tag{3.101}$$

where we have used the ℓ_1 norm due to the intermediate result in the proof of Theorem 3.2.26. Let $\Delta V_{\mathrm{so}}(\xi) := -\hat{d} \cdot (V(x)^{\frac{1}{q}} + \|z - \bar{z}(x)\|)$ denote the Lyapunov decrease. Using the same procedure used to derive the lower bound for $V_{\mathrm{so}}(\xi)$, we can show that, if $a_1^{\frac{1}{q}} - \sigma > 0$, we can write

$$\begin{aligned} \Delta V_{\mathrm{so}}(\xi) &\leq -\hat{d} \cdot \left((a_1^{\frac{1}{q}} - \sigma)\|x\| + \|z\| \right) \\ &\leq \underbrace{-\hat{d} \cdot \min\left\{ \frac{a_1^{\frac{1}{q}} - \sigma}{\sqrt{n_x}}, \frac{1}{\sqrt{n_z}} \right\}}_{=:\tilde{w}_{3,1}} (\|x\|_1 + \|z\|_1) \\ &\leq -\tilde{w}_{3,1} \cdot \|\xi\|. \end{aligned}$$

Else, if $a_1^{\frac{1}{q}} - \sigma \leq 0$, we can obtain the following bound:

$$\Delta V_{\mathrm{so}}(\xi) = -\hat{d} \cdot \left(V(x)^{\frac{1}{q}} + \|z - \bar{z}(x)\| \right) \leq -\frac{\hat{d} a_1^{\frac{1}{q}}}{\sigma} \|z\|.$$

Summing this last inequality and $\Delta V_{\mathrm{so}}(\xi) \leq -\hat{d} a_1^{\frac{1}{q}} \|x\|$, we obtain

$$\begin{aligned}
\Delta V_{\mathrm{so}}(\xi) &\leq -\frac{\hat{d} a_1^{\frac{1}{q}}}{2} \left(\|x\| + \frac{1}{\sigma} \|z\| \right) \\
&\leq -\frac{\hat{d} a_1^{\frac{1}{q}}}{2} \left(\frac{1}{\sqrt{n_x}} \|x\|_1 + \frac{1}{\sigma \sqrt{n_z}} \|z\|_1 \right) \\
&\leq -\underbrace{\frac{\hat{d} a_1^{\frac{1}{q}}}{2} \cdot \min \left\{ \frac{1}{\sqrt{n_x}}, \frac{1}{\sigma \sqrt{n_z}} \right\}}_{=: \tilde{w}_{3,2}} (\|x\|_1 + \|z\|_1) \\
&\leq -\tilde{w}_{3,2} \cdot \|\xi\|.
\end{aligned}$$

We define

$$\tilde{w}_3 := \begin{cases} \tilde{w}_{3,1}, & \text{if} \quad a_1^{\frac{1}{q}} - \sigma > 0, \\ \tilde{w}_{3,2}, & \text{otherwise}, \end{cases} \tag{3.102}$$

and conclude that $\Delta V_{\mathrm{so}}(\xi) \leq -\tilde{w}_3 \cdot \|\xi\|$. Hence, we can define the $\mathcal{K}_\infty$ functions $\alpha_{\mathrm{so},1}(\|\xi\|) := \tilde{w}_1 \cdot \|\xi\|$ and $\alpha_{\mathrm{so},2}(\|\xi\|) := \tilde{w}_2 \cdot \|\xi\|$ and the positive definite function $\alpha_{\mathrm{so},3}(\|\xi\|) := \tilde{w}_3 \cdot \|\xi\|$, such that

$$\begin{aligned}
\alpha_{\mathrm{so},1}(\|\xi\|) \leq V_{\mathrm{so}}(\xi) \leq \alpha_{\mathrm{so},2}(\|\xi\|) \\
V_{\mathrm{so}}(\xi_+) - V_{\mathrm{so}}(\xi) \leq -\alpha_{\mathrm{so},3}(\|\xi\|),
\end{aligned} \tag{3.103}$$

i.e., $V_{\mathrm{so}}(\xi)$ is a Lyapunov function in Σ for the system-optimizer dynamics $\xi_+ = \Phi(T; \xi)$ and the equilibrium $\xi = 0$, for any $T \leq \min\{T_4, T_5\}$. ☐

3.2.4 Illustrative example

In this section, in order to illustrate Theorem 3.2.28, we use a variant of the classical example from [Chen and Allgöwer, 1998]. In particular, we regard an optimal control problem of the form in (2.67) and we define the continuous-time dynamics as

$$\phi(x, u) := \begin{bmatrix} x_2 + u(\mu + (1-\mu)x_2) \\ x_1 + u(\mu - 4(1-\mu)x_2) \end{bmatrix}. \tag{3.104}$$

In order to compute an LQR-based terminal cost, the linearized dynamics are defined as

$$A_{\mathrm{c}} := \frac{\partial \phi}{\partial x}(0,0), \quad B_{\mathrm{c}} := \frac{\partial \phi}{\partial u}(0,0), \tag{3.105}$$

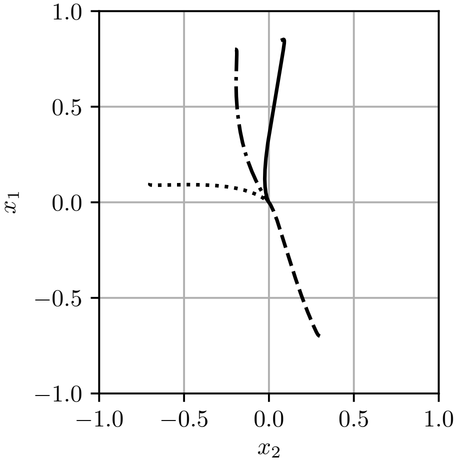


Figure 3.6: Illustrative example adapted from [Chen and Allgöwer, 1998] - closed-loop state trajectories obtained using the approximate feedback policy computed with a single iteration of a Gauss-Newton real-time algorithm (solid) and contour lines of $\frac{\mathrm{d}V(x(t))}{\mathrm{d}t}$ (dashed).

and discretized using exact discretization with piece-wise constant parametrization of the control trajectories:

$$A := \exp{(A_\mathrm{c}T_\mathrm{d})}, \quad B := \left(\int_{\tau=0}^{T_\mathrm{d}} \exp{(A_\mathrm{c}\tau)}\mathrm{d}\tau\right) B_\mathrm{c},$$

where T_d denotes the discretization time. We compute the solution P to the discrete-time algebraic Riccati equation

$$P = A^\top PA - (A^\top PB)(R + B^\top PB)^{-1}(B^\top PA) + Q,$$

where $Q = 0.1 \cdot \mathbb{I}_2$, $R = 0.1$, such that, the cost functionals can be defined as

$$L(x,u) := x^\top Qx + u^\top Ru, \quad m(x) := x^\top Px \tag{3.106}$$

and we impose simple bounds on the input $-2 \leq u \leq 2$.

We set the prediction horizon $T_\mathrm{f} = 1.0$ and discretize the resulting continuous-time optimal control problem using direct multiple shooting with $N = 5$ shooting nodes. Euler discretization is used for the cost integral and explicit RK4 is used to discretize the dynamics. In order to solve the resulting discretized

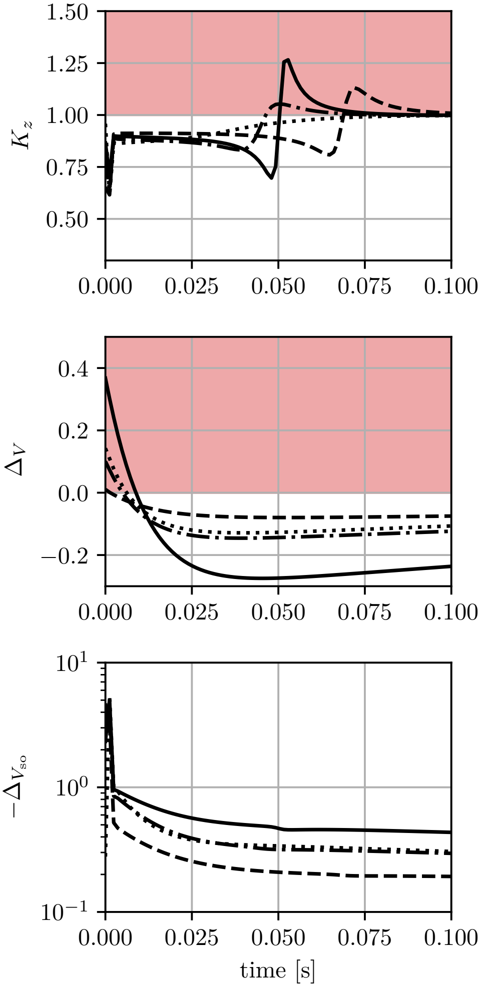


Figure 3.7: Illustrative example adapted from [Chen and Allgöwer, 1998] - although the numerical error does not necessarily decrease monotonically over time, the Lyapunov function for the combined system-optimizer dynamics $V_{\mathrm{so}}(\xi)$ does decrease over time.

optimal control problem, we use the standard RTI approach, with Gauss-Newton iterations and a fixed Levenberg-Marquardt-type term. A single SQP iteration per sampling time is carried out.

In order to compute an estimate for the constants involved in the definition of the Lyapunov function in Theorem 3.2.28, we regard six different initial conditions, and control the system using the feedback policy associated with the exact solution to the discretized optimal control problem. For every state x in the obtained trajectories, we evaluate the optimal cost $V(x)$ and the primal-dual optimal solution $\bar{z}(x)$. With these values, we can estimate the constants a_1, a_2, a_3 in Assumption 3.2.2, the constant $\tilde{\mu}$ in Assumption 3.2.4 and the constant σ in Theorem 3.2.13. Moreover, for any state visited, we carry out a limited number of iterations of the optimizer in order to estimate the contraction rate $\hat{\kappa}$. Choosing the sampling time $T = 0.0012$, we obtain the estimates $\kappa = 0.882, \bar{a} = 1.157, \hat{\gamma} = 70.23$, and $\hat{\mu} = 0.360$, such that the parameter involved in the definition of the Lyapunov function for the combined system-optimizer dynamics takes the value $\beta = 0.0041$ and we have $T_5 = 0.037 \geq T$. All the computations were carried out using CasADi [Andersson et al., 2019] and its interface to Ipopt [Wächter and Biegler, 2009] and the code for the illustrative example is made available at `https://github.com/zanellia/nmpc_system_optimizer_lyapunov`.

Figure 3.6 shows the state trajectories obtained controlling the system using the approximate feedback law starting from the selected initial conditions. Figure 3.7 shows the behavior of $\|z - \bar{z}(x)\|$, $V(x)$ and $V_{\mathrm{so}}(\xi)$ over time through the compact metrics

$$K_z := \frac{\|z_+ - \bar{z}(x_+)\|}{\|z - \bar{z}(x)\|}, \tag{3.107}$$

$$\Delta_V := \frac{V(x_+) - V(x)}{T_s}, \tag{3.108}$$

$$\Delta_{V_{\mathrm{so}}} := \frac{V_{\mathrm{so}}(\xi_+) - V_{\mathrm{so}}(\xi)}{T_s}. \tag{3.109}$$

In particular, it is shown that, although the numerical error $\|z - \bar{z}(x)\|$ and the value function does not necessarily decrease over time, the constructed Lyapunov function $V_{\mathrm{so}}(\xi)$ does decrease.

3.2.5 Conclusions

In this section, we presented a novel asymptotic stability results for inexact MPC relying on a limited number of iterations of an optimization algorithm. A class of optimization methods with Q-linearly convergent iterates has been regarded and, under the assumption that the ideal feedback law is stabilizing, we constructed a Lyapunov function for the system-optimizer dynamics. These results extend the attractivity proofs present in the literature which rely instead on the assumption that inequality constraints are either absent or inactive in the attraction region considered. Moreover, although under more stringent regularity assumptions, with respect to more general results on suboptimal MPC (cf. [Scokaert et al., 1999], [Pannocchia et al., 2011], [Allan et al., 2017]), we analyzed in deeper detail the interaction between system and optimizer.

3.3 Chapter summary and outlook

In this chapter, contraction and stability properties of real-time methods for NMPC have been derived and discussed. First, in Section 3.1, properties of a general class of numerical algorithms for nonconvex parametric programs have been proved. In particular, it was shown that, under the assumption that the underlying parameter undergoes sufficiently small changes across iterations of the optimizer, a Q-linear contraction estimate can be obtained. Moreover, concrete examples of algorithms that satisfy the assumptions made (including Newton-type generalized methods, ADMM and a truncated SQP method) have been discussed.

These contraction estimates were then used to derive asymptotic stability guarantees in Section 3.2. The section extends the attractivity result from [Diehl et al., 2005, Diehl et al., 2007] in the sense that *i)* asymptotic stability rather than attractivity was proved *ii)* the inequality constrained setting is covered *iii)* a Lyapunov function for the system-optimizer dynamics is explicitly constructed *iv)* a general framework that abstracts away from the specific numerical method and Lyapunov function chosen is introduced.

Extensions of the results presented in this chapter in several directions would contribute to their improved generality and applicability and are subject of ongoing research. Among other, an interesting extension would be the one of relaxing the assumptions of Lipschitz continuity of $\bar{z}$ and Q-linear contraction of the optimizer's iterations, namely Assumptions 3.2.11 and 3.2.12. In this way, a broader class of problems and algorithms could be addressed. In the case of Assumption 3.2.11, this would allow one to regard parametric programs that, for example, do not satisfy SSOSC and have, e.g., set-valued dual (or even primal) solutions or Hölder continuous primal optimal solutions.

Chapter 4

Zero-order methods for NMPC with stability guarantees

In this chapter we introduce a class of inexact NMPC methods, that we call *zero-order* methods, that can be used to compute cheap approximate solutions to nonlinear programs. The main idea behind zero-order methods originates in [Bock et al., 2007] and has strong connections with early work on second-order corrections in [Fletcher, 1982] and projection methods in [Sargent and Murtagh, 1973]. It lies in fixing part of the first- and second-order derivative information that is generally required to construct the QP subproblems within an SQP strategy. In this way, iterative algorithms for nonlinear programming that only require the computation of zero-order information (i.e., function evaluations) are obtained. As we will see, this type of algorithms can largely speed up the computations associated with the construction and solution of the QP subproblems in at least two different ways: *i)* in applications where the computation of first-order derivatives is a bottleneck in itself, freezing the sensitivities can lead to a considerable improvement of the computation times *ii)* the use of fixed first-order information generally leads to faster algorithms that can exploit precomputation of computationally intensive linear algebra operations (generally level 3 BLAS and LAPACK operations).

Zero-order methods are inexact methods, not only in the sense that they carry out approximate SQP iterations, but also, as we will see, due to the fact that they converge to inexact solutions. For this reason, apart from their computational benefits, we are interested in investigating the properties of the inexact solutions and the system theoretic properties of the resulting inexact feedback policies.

Outline

In Section 4.1, we introduce zero-order NMPC and investigate some of the fundamental numerical properties of the underlying numerical algorithm. In Sections 4.2 and 4.2.2, which are based on [Zanelli et al., 2016] and [Zanelli et al., 2019a], respectively, we introduce a stability analysis of zero-order NMPC in the equality and inequality constrained setting, respectively. Moreover, in Section 4.3, we discuss how the structure of the nonconvex programs arising from NMPC formulations can be exploited in order to derive an efficient algebraic elimination strategy to be combined with a dense active-set solver.

4.1 Zero-order SQP for NMPC

In this initial section, we will introduce the main idea behind zero-order NMPC as first described in [Bock et al., 2007]. Consider the following finite-horizon discrete-time optimal control problem:

$$\begin{aligned} V(x) := \min_{\substack{s_0,\dots,s_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l(s_i,u_i) + m(s_N) \\ \text{s.t.} \quad & s_0 - x = 0, \\ & \psi(s_i,u_i) - s_{i+1} = 0, \quad i = 0,\dots,N-1, \\ & \pi(s_i,u_i) \leq 0, \qquad\qquad i = 0,\dots,N-1, \\ & \pi_N(s_N) \leq 0, \end{aligned} \tag{4.1}$$

with optimization variables $s_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$. Here, $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_+$ and $m : \mathbb{R}^{n_x} \to \mathbb{R}_+$ define the objective and the functions $\pi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\pi}$ and $\pi_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{\pi_N}}$ describe the stage and terminal constraints, respectively.

If the NLP in (4.1) is solved with an SQP strategy [Bock, 1983], the resulting subproblems take the form

$$\begin{aligned}\min_{\substack{\Delta s_0,\cdots,\Delta s_N\\ \Delta u_0,\cdots,\Delta u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1}\begin{bmatrix}\Delta s_i\\ \Delta u_i\\ 1\end{bmatrix}^\top M_i \begin{bmatrix}\Delta s_i\\ \Delta u_i\\ 1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\Delta s_N\\ 1\end{bmatrix}^\top M_N \begin{bmatrix}\Delta s_N\\ 1\end{bmatrix}\\
\text{s.t.} \quad & \Delta s_0 - r_{\lambda_0} = 0\\
& \Delta s_{i+1} - A_i\Delta s_i - B_i \Delta u_i - r_{\lambda_{i+1}} = 0, \quad i = 0,\cdots,N-1,\\
& \pi(\hat{s}_i,\hat{u}_i) + C_i\Delta s_i + D_i \Delta u_i \leq 0, \qquad i = 0,\cdots,N-1,\\
& \pi_N(\hat{s}_N) + C_N \Delta s_N \leq 0,\end{aligned} \tag{4.2}$$

with dynamics and inequality constraints linearized at the current approximate solution $\hat{s} := (\hat{s}_0,\ldots,\hat{s}_N)$ and $\hat{u} := (\hat{u}_0,\ldots,\hat{u}_{N-1})$:

$$A_i := \frac{\partial \psi}{\partial s}(\hat{s}_i,\hat{u}_i), \qquad B_i := \frac{\partial \psi}{\partial u}(\hat{s}_i,\hat{u}_i)$$

and

$$C_i := \frac{\partial \pi}{\partial s}(\hat{s}_i,\hat{u}_i), \qquad D_i := \frac{\partial \pi}{\partial u}(\hat{s}_i,\hat{u}_i).$$

Similarly, for the terminal constraint, we have defined $C_N := \frac{\partial \pi_N}{\partial s}(\hat{s}_N)$. For ease of notation, we have introduced the Hessians M_i, for $i = 0,\cdots,N-1$

$$M_i := \begin{bmatrix} M_i^{ss} & M_i^{su} & \nabla_s l_i\\ M_i^{us} & M_i^{uu} & \nabla_u l_i\\ \nabla_s l_i^\top & \nabla_u l_i^\top & 1\end{bmatrix} \tag{4.3}$$

and

$$M_N := \begin{bmatrix} M_N^{ss} & \nabla_s m\\ \nabla_s m^\top & 1\end{bmatrix}, \tag{4.4}$$

where the matrices M_i^{ss}, $M_i^{su} = M_i^{us,\top}$ and M_i^{uu} denote the approximations of the Hessians of the Lagrangian with respect to input and state variables at stage i. Moreover, we have introduced the shorthands $\nabla_s l_i := \nabla_{s_i} l(s_i,u_i)$ and $\nabla_u l_i := \nabla_{u_i} l(s_i,u_i)$. Finally, we have introduced the vectors r_{λ_i} for $i = 0,\cdots,N$ to denote the residuals of the equality constraints.

A possible way of solving the subproblems in (4.2) is by eliminating the state variables s resulting in a smaller and dense problem that can be efficiently solved by state-of-the-art QP solvers [Ferreau et al., 2014]. However, the computational burden introduced by the QP solver can be rather high. A second source of computational effort, regardless of the QP formulation used, is the fact that the matrices and vectors involved in the QP need to be computed at every iteration.

The main motivation behind an inexact strategy that exploits fixed derivatives is to reduce the computational burden associated with sensitivity generation and linear algebra operations by using the Jacobians evaluated at a reference $(s_\mathrm{r}, u_\mathrm{r})$:

$$A_i = A = \frac{\partial \psi}{\partial s}(s_\mathrm{r}, u_\mathrm{r}), \qquad B_i = B = \frac{\partial \psi}{\partial u}(s_\mathrm{r}, u_\mathrm{r}),$$

$$C_i = C = \frac{\partial \pi}{\partial s}(s_\mathrm{r}, u_\mathrm{r}), \qquad D_i = D = \frac{\partial \pi}{\partial u}(s_\mathrm{r}, u_\mathrm{r}).$$

Moreover, following the algorithm proposed in [Bock et al., 2007], we set M_i, for $i = 0, \cdots, N-1$

$$M_i = \begin{bmatrix} Q & S & r_i \\ S^\top & R & q_i \\ r_i^\top & q_i^\top & 1 \end{bmatrix} \tag{4.5}$$

and

$$M_N = \begin{bmatrix} Q_N & q_N \\ q_N^\top & 1 \end{bmatrix}, \tag{4.6}$$

where Q, R, S and Q_N define the chosen approximation of the Hessian blocks at the linearization point $(s_\mathrm{r}, u_\mathrm{r})$. Finally, at every iteration, we update the gradient vectors based on the approximate solution as $q_i := \nabla_s l(s_\mathrm{r}, u_\mathrm{r}) + Q(\hat{s}_i - s_\mathrm{r}) + S(\hat{u}_i - u_\mathrm{r})$, $r_i := \nabla_u l(s_\mathrm{r}, u_\mathrm{r}) + S^\top(\hat{s}_i - s_\mathrm{r}) + R(\hat{u}_i - u_\mathrm{r})$, for $i = 0, \dots, N-1$, and $q_N := \nabla_s m(s_\mathrm{r}) + Q_N(\hat{s}_N - s_\mathrm{r})$.

Remark 4.1.1. *Notice that, in principle, a stage-dependent reference $s_{\mathrm{r},i}, u_{\mathrm{r},i}$ for $i = 0, \dots, N-1$ and $s_{\mathrm{r},N}$ can be considered without affecting the convergence and computational properties of the algorithm. However, for the sake of notational simplicity we restrict our attention to a fixed linerization point across stages.*

Overall, introducing $y := (s, u)$ to refer to the primal solution and $\hat{C} := [\mathbb{I}\ 0]^\top$, we can rewrite (4.1) in the compact form

$$\begin{aligned} \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(y) + \hat{C}x = 0, \\ & h(y) \leq 0, \end{aligned} \tag{4.7}$$

where $y \in \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^n \to \mathbb{R}^{n_h}$.

With this compact notation, a generic QP subproblem reads as

$$\begin{aligned} \min_{\Delta y} \quad & \nabla_y f(\hat{y})^\top \Delta y + \frac{1}{2}\Delta y^\top M \Delta y \\ \text{s.t.} \quad & g(\hat{y}) + \hat{C}x + \nabla_y g(\hat{y})^\top \Delta y = 0, \\ & h(\hat{y}) + \nabla_y h(\hat{y})^\top \Delta y \leq 0. \end{aligned} \tag{4.8}$$

Here, we have introduced M to denote the chosen approximation of the Hessian of the Lagrangian and λ and μ are the Lagrange multipliers associated with the equality and inequality constraints, respectively. Accordingly, the zero-order SQP iterates are defined by the solution to the following QP:

$$\begin{aligned} \min_{\Delta y} \quad & a^\top \Delta y + \frac{1}{2}\Delta y^\top M \Delta y \\ \text{s.t.} \quad & g(\hat{y}) + \hat{C}x + G\Delta y = 0, \\ & h(\hat{y}) + H\Delta y \leq 0, \end{aligned} \tag{4.9}$$

where $a = \nabla_y f(y_\mathrm{r}) + P(\hat{y} - y_\mathrm{r})$, with $P \approx \nabla_y^2 \mathcal{L}(y_\mathrm{r}, \lambda_\mathrm{r}, \mu_\mathrm{r})$. Here, G and H represent the constant Jacobians of the equality and inequality constraints, respectively.

Remark 4.1.2. *Notice that, unlike in [Bock et al., 2007], we do not require $P = M$. This becomes particularly useful in the iterative setting described in Section 4.1.1. In that case, we will see that, by choosing P to be the exact Hessian of the Lagrangian, we can recover local quadratic convergence, even when $M \neq P$. In this way, we can leverage the computational benefits associated with choosing a positive-definite Hessian for the QP subproblems. At the same time, we recover quadratic contraction typical of exact Hessian SQP, which would however require in general a convexification strategy to handle potentially indefinite Hessians.*

If convergence to a point $(\tilde{y}, \tilde{\lambda}, \tilde{\mu})$ is achieved, from the first-order optimality conditions of the QPs, we obtain

$$\begin{aligned} \nabla_y f(\tilde{y}) + G^\top \tilde{\lambda} + H^\top \tilde{\mu} &= 0, \\ g(\tilde{y}) + \hat{C}x &= 0, \\ h(\tilde{y}) &\leq 0, \\ h_i(\tilde{y})\tilde{\mu}_i &= 0, \quad i = 1, \ldots, n_h, \end{aligned} \tag{4.10}$$

which can be interpreted as the first-order optimality conditions of the nonlinear program [Bock et al., 2007]

$$\begin{aligned} \min_{y} \quad & f(y) + \xi(x)^\top y \\ \text{s.t.} \quad & g(y) + \hat{C}x = 0, \\ & h(y) \leq 0, \end{aligned} \tag{4.11}$$

with

$$\xi(x) := \left(G^\top - \nabla g(\tilde{y}(x))\right) \tilde{\lambda}(x) + \left(H^\top - \nabla h(\tilde{y}(x))\right) \tilde{\mu}(x).$$

From this last perturbed nonlinear program, we see that the iterates of the proposed strategy converge to a feasible, but suboptimal solution.

The idea of exploiting new function evaluations and fixed sensitivities in order to reduce constraint violation of the iterates of an SQP-like algorithm is not entirely new and has strong connections with the popular second-order corrections introduced in [Fletcher, 1982]. Second-order corrections were introduced to help mitigate the Maratos effect which takes place in the presence of strong curvature of the constraints. In this situation, it can be difficult to achieve decrease in the merit function along the tangent to the linearized constraints. A second-order correction is a correction applied to the search direction computed by solving a modified QP where the residuals of the constraints are evaluated at the uncorrected new iterate (see [Fletcher, 1982] or [Nocedal and Wright, 2006] for further details). From this point of view, the strategy discussed here can be seen as an iterative application of second-order corrections.

4.1.1 Convergence analysis

In this section, we analyze the algorithmic properties of the proposed strategy. In particular, we are interested in *i)* studying convergence of the iterates associated with the solution to instances of (4.9) for a fixed reference $(y_\mathrm{r}, \lambda_\mathrm{r}, \mu_\mathrm{r})$ *ii)* deriving convergence guarantees for an iterative procedure in which $(y_\mathrm{r}, \lambda_\mathrm{r}, \mu_\mathrm{r})$ is updated in an outer loop. In the second case, the main idea lies in solving a series of problems with frozen sensitivities as the one in (4.11) and, after convergence, updating the sensitivities in order to construct a new subproblem with fixed derivatives. With reference to this second setting, we will assume in the following that $(y_\mathrm{r}, \lambda_\mathrm{r}, \mu_\mathrm{r})$ is updated according to the solution $(\tilde{y}, \tilde{\lambda}, \tilde{\mu})$. Moreover, we will refer to the iterations carried out on an instance of (4.11) as *inner iterations*. Similarly, we will refer to the iterates defined by the solutions the inner iterates converge to as *outer iterates*.

The analysis of this algorithm becomes relevant in at least two settings. First, because the zero-order methods described in Sections 4.1, 4.2 and 4.2.2 will typically be embedded into a hierarchical strategy, as in the so-called multi-level iterations in [Bock et al., 2007], with potentially asynchronous updates of the derivative information at lower frequency. Second, an inherent advantage of this type of algorithmic strategy would lie in the fact that any intermediate solution would be feasible with respect to both equality and inequality constraints. In this way, the approximate solutions obtained with a limited number of iterations could become very useful in contexts where feasibility is crucial.

In order to derive a convergence result for the inner iterations, we will interpret (4.9) as the QP associated with the linearized generalized equation obtained by applying the generalized Newton-type method to the following (nonlinear) generalized equation

$$0 \in \tilde{F}(z) + \mathcal{N}_K(z). \tag{4.12}$$

Here, we have omitted the dependency on x which we will assume to be fixed throughout the rest of this section. Moreover, we have introduced $z := (y, \lambda, \mu)$,

$$\tilde{F}(z) := \begin{bmatrix} \nabla_y f(\tilde{y}) + P(y - \tilde{y}) + G^\top \lambda + H^\top \mu \\ -g(y) \\ -h(y) \end{bmatrix} \tag{4.13}$$

and $K = \mathbb{R}^n \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h}_+$. For consistency, we have followed the notation and formulation used in [Robinson, 1980].

It can be easily verified that the linearized generalized equation

$$0 \in \tilde{F}(\hat{z}) + J(z - \hat{z}) + \mathcal{N}_K(z), \tag{4.14}$$

with

$$J := \begin{bmatrix} M & G^\top & H^\top \\ -G & 0 & 0 \\ -H & 0 & 0 \end{bmatrix} \approx \frac{\partial \tilde{F}}{\partial z}(\hat{z}) = \begin{bmatrix} P & G^\top & H^\top \\ -\frac{\partial g}{\partial y}(\hat{y}) & 0 & 0 \\ -\frac{\partial h}{\partial y}(\hat{y}) & 0 & 0 \end{bmatrix}, \tag{4.15}$$

represents the first-order optimality conditions associated with (4.9). We make the following assumptions.

Assumption 4.1.3. *Let $\tilde{z}$ be a solution to* (4.12). *Assume that* (4.12) *is strongly regular at $\tilde{z}$ with Lipschitz constant σ over the neighborhood $\mathcal{B}(\tilde{z}, \tilde{r}_z)$, with $\tilde{r}_z > 0$.*

Assumption 4.1.4. *Assume that there exist a non-empty neighborhood $\mathcal{B}(\tilde{z}, \hat{r}_z)$ of $\tilde{z}$ with radius $\hat{r}_z \leq \tilde{r}_z$ and a positive constant $\tilde{\kappa}$, with $\sigma\tilde{\kappa} < \frac{1}{2}$ such that, for any $z \in \mathcal{B}(\tilde{z}, \hat{r}_z)$, the following holds:*

$$\left\| \frac{\partial \tilde{F}}{\partial z}(\hat{z}) - J \right\| \leq \tilde{\kappa}. \tag{4.16}$$

Theorem 4.1.5 (Convergence of inner iterations). *Let* $\tilde{z}$ *be a solution to* (4.12) *such that Assumptions 4.1.3 and 4.1.4 hold. Let* $\hat{z}_+(\hat{z})$ *denote the solution to* (4.14) *constructed at* $\hat{z}$. *Then, there exist a strictly positive constant* $r_z \leq \hat{r}_z$ *and a positive constant* $\kappa < 1$, *such that, for any* $\hat{z}$ *in* $\mathcal{B}(\tilde{z}, r_z)$, *the following holds:*

$$\|\hat{z}_+(\hat{z}) - \tilde{z}\| \leq \kappa \|\hat{z} - \tilde{z}\|. \tag{4.17}$$

Proof. The proof follows the same arguments as the one of Lemma 2.1.36 □

4.1.2 Convergence of outer iterations

In the following, we analyze the convergence of the outer iterates associated with the solution to the zero-order problems in the setting where subsequent linearizations are carried out at the solutions to (4.12).

In order to simplify the derivations, we will regard the equality constrained formulation, which can be naturally extended to the inequality constrained case whenever it can be assumed that strict complementarity holds. Under this setting, the generalized equation (4.12) reduces to the nonlinear root-finding problem

$$\tilde{F}(z, \tilde{z}) := \begin{bmatrix} \nabla_y f(\tilde{y}) + P(y - \tilde{y}) + G(\tilde{y})^\top \lambda \\ g(y) \end{bmatrix} = 0, \tag{4.18}$$

where we have added the explicit dependence on $\tilde{z}$, which is now allowed to vary. The following result provides local contraction for the iterates obtained by solving a series of subproblems of the form (4.18), where G is updated according to the current approximate solution.

Assumption 4.1.6. *Let* $\bar{z}$ *be a solution to the equality constrained variant of* (4.7). *Assume that LICQ and SOSC hold at* $\bar{z}$.

Theorem 4.1.7 (Convergence of outer iterations). *Let* $\tilde{z}_+(\tilde{z})$ *denote a solution to* (4.18) *for a given* $\tilde{z}$ *and let* $Z \in \mathbb{R}^{n \times n_g}$ *be a matrix whose columns span the nullspace of* $\frac{\partial g}{\partial y}(\bar{y})$, *i.e.,* $\frac{\partial g}{\partial y}(\bar{y})Z = 0$. *Let Assumption 4.1.6 hold. Then, if*

$$\kappa := \rho\left((Z^\top P Z)^{-1} Z^\top (\nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P) Z\right) < 1, \tag{4.19}$$

the outer iterations exhibit local linear convergence with asymptotic convergence rate κ. *If* $\kappa = 0$, *then the outer iterations converge quadratically locally.*

Proof. Expanding $\tilde{z}_+(\tilde{z})$ at $\bar{z}$ we obtain

$$\tilde{z}_+(\tilde{z}) = \bar{z} + \frac{\partial \tilde{z}_+}{\partial \tilde{z}}(\bar{z}) \cdot (\tilde{z} - \bar{z}) + O\left(\|\tilde{z} - \bar{z}\|^2\right), \tag{4.20}$$

which shows that, in order to study local contractivity, it suffices to compute the spectral radius of $\frac{\partial \tilde{z}_+}{\partial \tilde{z}}$. In particular, using the implicit function theorem, we can write

$$\frac{\partial \tilde{z}_+}{\partial \tilde{z}} = -\left(\frac{\partial \tilde{F}}{\partial z}\right)^{-1} \frac{\partial \tilde{F}}{\partial \tilde{z}} = \begin{bmatrix} P & G(\bar{y})^\top \\ G(\bar{y}) & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P & 0 \\ 0 & 0 \end{bmatrix},$$

which can be easily computed using the null space method [Nocedal and Wright, 2006] as follows. Since the second block column of the right-hand side is zero, we can regard the linear system

$$\begin{bmatrix} P & G(\bar{y})^\top \\ G(\bar{y}) & 0 \end{bmatrix} \begin{bmatrix} X \\ \star \end{bmatrix} = \begin{bmatrix} \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P \\ 0 \end{bmatrix}.$$

We partition X as

$$X = YX_1 + ZX_2, \tag{4.21}$$

where Y is any $n \times n_g$ matrix such that $[Y|Z]$ is nonsingular. By substituting X in the second equation and recalling that $GZ = 0$, we obtain

$$GYX_1 = 0 \tag{4.22}$$

and, due to the fact that GY is invertible, we obtain that

$$X_1 = (GY)^{-1}0 = 0. \tag{4.23}$$

Similarly, we can substitute (4.21) in the first equation in order to obtain

$$PYX_1 + PZX_2 + G^\top \star = \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P. \tag{4.24}$$

Premultiplying by $Z^\top$ and using the fact that $X_1 = 0$, we can write

$$Z^\top PZX_2 = Z^\top (\nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P), \tag{4.25}$$

such that we can compute X as

$$X = Z(Z^\top PZ)^{-1} Z^\top (\nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}) - P). \tag{4.26}$$

Observing that

$$\frac{\partial \tilde{z}_+}{\partial \tilde{z}} = \begin{bmatrix} X & 0 \\ \star & 0 \end{bmatrix}, \tag{4.27}$$

we can conclude that $\kappa = \rho\left(\frac{\partial \tilde{z}_+}{\partial \tilde{z}}\right) = \rho(X)$. Hence, using Ostrowski's Theorem [Ostrowski, 1966], we can conclude that the iterates converge linearly locally with asymptotic contraction rate $\rho(X)$. □

A second variant of the algorithm under analysis can be obtained assuming that the exact gradient of the cost $\nabla_y f$ is computed at every iteration, which is sometimes much cheaper than computing the derivatives associated with the constraints of the problem. In this way we obtain

$$\tilde{F}(z, \tilde{z}) := \begin{bmatrix} \nabla_y f(y) + G(\tilde{y})^\top \lambda \\ g(y) \end{bmatrix} = 0. \tag{4.28}$$

Corollary 4.1.8 (Convergence of outer iterations)**.** *Let $\tilde{z}_+(\tilde{z})$ denote a solution to* (4.28) *for a given $\tilde{z}$ and let $Z \in \mathbb{R}^{n \times n_g}$ be a matrix whose columns span the nullspace of $\frac{\partial g}{\partial y}(\bar{y})$, i.e., $\frac{\partial g}{\partial y}(\bar{y})Z = 0$. Let Assumption 4.1.6 hold. Then, if*

$$\kappa := \rho\left((Z^\top \nabla_y^2 f(\bar{y}) Z)^{-1} Z^\top \sum_i \bar{\lambda}_i \nabla_y^2 g_i(\bar{y}) Z \right) < 1, \tag{4.29}$$

the outer iterations exhibit local linear convergence with asymptotic convergence rate κ. If $\kappa = 0$, then the outer iterations converge quadratically

Proof. The proof follows the same arguments as the proof of Theorem 4.1.7 noting that expanding $\tilde{z}_+(\tilde{z})$ at $\bar{z}$ we obtain

$$\tilde{z}_+(\tilde{z}) = \bar{z} + \frac{\partial \tilde{z}_+}{\partial \tilde{z}}(\bar{z}) \cdot (\tilde{z} - \bar{z}) + O\left(\|\tilde{z} - \bar{z}\|^2\right), \tag{4.30}$$

which shows that, in order to study local contractivity, it suffices to compute the spectral radius of $\frac{\partial \tilde{z}_+}{\partial \tilde{z}}$. In particular, using the implicit function theorem, we can write

$$\frac{\partial \tilde{z}_+}{\partial \tilde{z}} = -\left(\frac{\partial \tilde{F}}{\partial z}\right)^{-1} \frac{\partial \tilde{F}}{\partial \tilde{z}} = \begin{bmatrix} \nabla_y^2 f(\bar{y}) & G(\bar{y})^\top \\ G(\bar{y}) & 0 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i \bar{\lambda}_i \nabla_y^2 g_i(\bar{y}) & 0 \\ 0 & 0 \end{bmatrix}.$$

The rest of the proof follows the same derivations in the proof of Theorem 4.1.7. □

Illustrative example

In the following, in order to illustrate the meaning of Theorem 4.1.7, a simple numerical example is presented. We will regard the optimization problem

$$\begin{aligned} \min_y \quad & \left\| \begin{matrix} (y_1 - \alpha)^3 \\ y_2 \end{matrix} \right\|_2^2 \\ \text{s.t.} \quad & (y_1 - 1)^2 + (y_1 - 1)^3 + (y_2 - 1) - 1 = 0, \end{aligned} \tag{4.31}$$

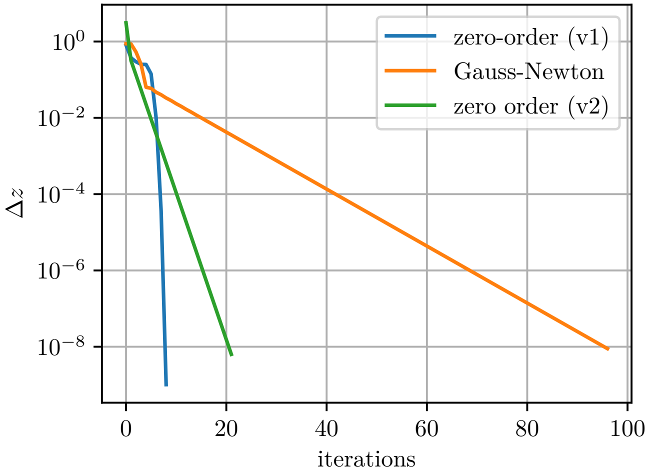


Figure 4.1: Illustrative example: comparison of convergence rate of feasible SQP with iterated second-order corrections and Gauss-Newton on the nonlinear least squares problem (4.31) - Euclidean norm of the primal-dual steps. Convergence rate computed according to Theorem 4.1.7: $\kappa = 0.412$. "Empirical" convergence rate computed with linear regression: $\tilde{\kappa} = 0.408$. Gauss-Newton asymptotic convergence rate: 0.842.

where $\alpha \in \mathbb{R}$ is a tuning parameter. Problem (4.31) is solved with both variants of the proposed algorithm and the generalized Gauss-Newton method. For both variants of the zero-order algorithm, we choose M to be the Gauss-Newton Hessian. The iterates in the primal space and the Euclidean of the primal-dual steps are reported in Figure 4.2 and 4.1, respectively. Figure 4.2 shows that the resulting feasible SQP iterates are always feasible, while in Figure 4.1 the result on the asymptotic convergence rate in Theorem 4.1.7 is validated and we see that feasible SQP achieves a better convergence rate than generalized Gauss-Newton. Moreover, choosing P to be the exact Hessian of the Lagrangian at $(\tilde{y}, \tilde{\lambda})$, the first variant of the algorithm achieves quadratic convergence as stated by Theorem 4.1.7. The same does not hold for the second variant, where the asymptotic contraction rate is proportional to the multipliers and to the Hessians of the constraints, as stated in Corollary 4.1.8.

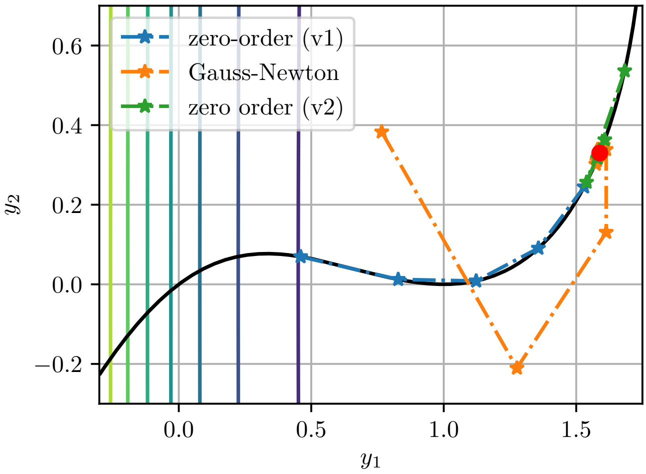


Figure 4.2: Illustrative example: iterates in the primal space obtained with the proposed zero-order sequential programming algorithm and with Gauss-Newton. The iterates obtained with zero-order sequential programming are always feasible with respect to the nonlinear equality constraint.

Efficient solution of the quadratic subproblems

In order to obtain an efficient implementation of the feasible SQP, an efficient solver for the quadratic subproblems involved in the solution of problem (4.9) is of fundamental importance. In particular, given that the matrices appearing in the subproblems are fixed until a solution to (4.12) is found, using an active-set strategy, it is possible to maintain a factorization of the KKT systems to be updated with low-rank updates as the working set changes across iterations. Notice that, in this context, interior-point methods cannot in general exploit the fact that the sensitivities are fixed because the update of the multipliers and slack variables determines a high-rank update of the Hessians. For structured problems arising in optimal control a tailored elimination algorithm to be used in combination with the active-set strategy described in [Ferreau et al., 2008] and implemented in `qpOASES` is proposed in Section 4.3.1. In the context of large unstructured problems, we propose to use an active-set solver that exploits the strategy presented in [Janka et al., 2016] and implemented in `qpOASES` in order to speed up the solution of the underlying linear systems. The algorithm presented in [Janka et al., 2016] is based on a Schur complement strategy [Bisschop and Meeraus, 1977, Gill et al., 1987, Gould and Toint, 2002, Gill and Wong, 2015]. Such an algorithm and its implementation in `qpOASES` can be readily used to obtain an efficient solution of the quadratic subprograms in the feasible SQP strategy and require no further development.

A proof-of-concept implementation of the proposed algorithm based on `CasADi`'s code generation and `qpOASES` is available at `https://github.com/zanellia/feasible_sqp`.

4.1.3 Conclusions

In this section, we have introduced zero-order NMPC and analyzed its numerical properties. The proposed algorithm relies on the solution of a series of subproblems with zero-order iterations (or second-order corrections) such that feasibility can be achieved by reevaluating the residuals of the constraints at each new iteration. In this way, a feasible, but suboptimal solution can be obtained, while considerably reducing the computational footprint. The algorithm can be embedded into an iterative strategy that updates the sensitivities in an outer loop such that convergence towards a locally optimal solution is obtained. We analyzed the contraction properties of the inner and outer iterations and illustrated the meaning of the convergence results on a toy example.

In the context of unstructured nonconvex programming, the proposed algorithm can be used in combination with the active-set Schur complement strategy proposed in [Janka et al., 2016]. Such a method can exploit the fact that the QP matrices are kept unchanged across several iterations to update the factorization of the KKT system. A proof-of-concept implementation has been made available while extensive benchmarking of the proposed strategy is subject of undergoing research.

4.2 Asymptotic stability of zero-order NMPC

In this section, we analyze the stabilizing properties of the zero-order suboptimal feedback policy. The analysis will be carried out in the equality constrained setting first and later extended to the inequality constrained setting in Section 4.2.2. Without loss of generality, we will assume that the system has a steady state at the origin and that the fixed derivatives are evaluated at $(s, u) = (0, 0)$.

We will refer to an equality constrained formulation of the form

$$
\begin{aligned}
V(x) := \min_{\substack{s_0,\dots,s_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l(s_i,u_i) + m(s_N) \\
\text{s.t.} \quad & s_0 - x = 0, \\
& \psi(s_i,u_i) - s_{i+1} = 0, \;\; i = 0,\dots,N-1.
\end{aligned}
\tag{4.32}
$$

Regard the first-order optimality conditions for the QP subproblem (4.2)

$$
\begin{aligned}
& -\Delta s_0 && = -r_{\lambda_0} \\
& -\lambda_0 + M_0^{ss}\Delta s_0 + M_0^{su}\Delta u_0 + A_0^\top \lambda_1 && = -\nabla_s l_0 \\
& M_0^{us}\Delta s_0 + M_0^{uu}\Delta u_0 + B_0^\top \lambda_1 && = -\nabla_u l_0 \\
& \quad\cdots && \quad\cdots \\
& -\Delta s_N + A_{N-1}\Delta s_{N-1} + B_{N-1}\Delta u_{N-1} && = -r_{\lambda_N} \\
& -\lambda_N + M_N^{ss}\Delta s_N && = -\nabla_s l_N,
\end{aligned}
\tag{4.33}
$$

where $\lambda_0, \cdots, \lambda_N$ are the Lagrange multipliers associated with the equality constraints. Within the proposed zero-order strategy the optimality conditions take the form

$$
\begin{aligned}
& -\Delta s_0 && = -r_{\lambda_0} \\
& -\lambda_0 + Q\Delta s_0 + S\Delta u_0 + A^\top \lambda_1 && = -q_0 \\
& S^\top \Delta s_0 + R\Delta u_0 + B^\top \lambda_1 && = -r_0 \\
& \quad\cdots && \quad\cdots \\
& -\Delta s_N + A\Delta s_{N-1} + B\Delta u_{N-1} && = -r_{\lambda_N} \\
& -\lambda_N + Q_N\Delta s_N && = -q_N.
\end{aligned}
\tag{4.34}
$$

It is immediately clear from (4.34), that, when convergence is achieved, the residuals associated with the equality constraints must be zero, such that we recover a feasible solution satisfying the following conditions:

$$\begin{aligned}
s_0 - x &= 0 \\
q_0 + A^\top \lambda_1 - \lambda_0 &= 0 \\
r_0 + B^\top \lambda_1 &= 0 \\
\dots \quad & \dots \\
s_N - \psi(s_{N-1}, u_{N-1}) &= 0 \\
q_N - \lambda_N &= 0.
\end{aligned} \tag{4.35}$$

The solution recovered by the inexact strategy is feasible, as the equality constraints appear unchanged in the nonlinear root-finding problem (4.35). The approximation introduced by fixing the sensitivities affects instead optimality of the solution. For this reason, when analyzing the stability of the closed-loop strategy, the question naturally arises of how this approximation degrades the properties of the optimal value function which is commonly used as a Lyapunov function in tracking NMPC.

The nominal stability proof will be based on a sensitivity analysis of the solution to the nonlinear root-finding problem (4.35) with respect to the initial value x at the origin. These considerations will be then exploited to prove that, in a neighbourhood of the origin, the standard Lyapunov stability arguments must hold for the inexact strategy as well. First, a simple fact on the sensitivity of both optimal and suboptimal solutions with respect to x will be recalled. Regard the first-order optimality conditions of (4.32):

$$\begin{aligned}
s_0 - x &= 0 \\
\nabla_s l_0 + \nabla_s \psi(s_0, u_0)\lambda_1 - \lambda_0 &= 0 \\
\nabla_u l_0 + \nabla_u \psi(s_0, u_0)\lambda_1 &= 0 \\
s_1 - \psi(s_0, u_0) &= 0 \\
\dots \quad & \dots \\
s_N - \psi(s_{N-1}, u_{N-1}) &= 0 \\
\nabla_s m - \lambda_N &= 0.
\end{aligned} \tag{4.36}$$

Equations (4.35) and (4.36) will be referred to in the compact forms

$$\tilde{F}(z, x) = 0 \tag{4.37}$$

and

$$F(z, x) = 0, \tag{4.38}$$

respectively, where $z := (s, u, \lambda)$.

Assumption 4.2.1. *Assume that there exists a non-empty neighborhood $\bar{\mathbb{X}}$ of the origin such that, for any $x \in \bar{\mathbb{X}}$, equations* (4.37) *and* (4.38) *have at least a solution. Moreover, assume that $\nabla_z F(0, 0)$ is nonsingular.*

Assumption 4.2.2. *Assume that l and m are positive definite functions and that $\nabla_s l(0, 0) = 0$, $\nabla_u l(0, 0) = 0$ and $\nabla_s m(0) = 0$. Moreover, assume that there exists positive constants $a_1 \leq a_2$ such that $a_1 \|x\|^2 \leq l(x, u)$ for all u and all $x \in \bar{\mathbb{X}}$ and $V(x) \leq a_2 \|x\|^2$ for all $x \in \bar{\mathbb{X}}$.*

Assumption 4.2.3. *Assume that, for any $x \in \bar{\mathbb{X}}$, there exists a $u^\dagger(x)$ such that*

$$m(\psi(x, u^\dagger(x))) - m(x) + l(x, u^\dagger(x)) \leq 0. \tag{4.39}$$

Lemma 4.2.4. *Let Assumptions 4.2.1 and 4.2.2 hold. Then there exists a single-valued localization $\tilde{z}(x)$, with $\tilde{z}(0) = 0$, of the solution map of* (4.37). *Similarly, there exists a single-valued localization $\bar{z}(x)$, with $\bar{z}(x) = 0$, of the solution map of* (4.38). *Moreover, the following holds:*

$$\|\tilde{z}(x) - \bar{z}(x)\| = O\left(\|x\|^2\right).$$

Proof. First, it can be trivially verified that, for $x = 0$, $z = 0$ solves both (4.37) and (4.38). The existence of the single-valued localizations $\bar{z}(x)$ and $\tilde{z}(x)$ is guaranteed by Dini's Theorem (cf. Theorem 2.1.22). Moreover, it is possible to express their derivative with respect to the parameter x around the origin respectively as

$$\frac{\partial z}{\partial x} = -\frac{\partial F}{\partial z}^{-1} \frac{\partial F}{\partial x} \quad \text{and} \quad \frac{\partial \tilde{z}}{\partial x} = -\frac{\partial \tilde{F}}{\partial z}^{-1} \frac{\partial \tilde{F}}{\partial x},$$

where invertibility of $\frac{\partial F}{\partial z}$ and $\frac{\partial \tilde{F}}{\partial z}$ is guaranteed in a neighbourhood of the origin due to Assumption 4.2.1. It is possible to see that the derivative matrices coincide at the origin, i.e

$$\frac{\partial \tilde{F}}{\partial z}(0, 0) = \frac{\partial F}{\partial z}(0, 0) \quad \text{and} \quad \frac{\partial \tilde{F}}{\partial x}(0, 0) = \frac{\partial F}{\partial x}(0, 0) \tag{4.40}$$

due to the fact that

$$A = \frac{\partial \psi}{\partial s}(0, 0) \quad \text{and} \quad B = \frac{\partial \psi}{\partial u}(0, 0). \tag{4.41}$$

Moreover, since $\tilde{z}(0) = \bar{z}(0) = 0$, we can write

$$\|\tilde{z}(x) - \bar{z}(x)\| = O\left(\|x\|^2\right). \tag{4.42}$$

□

Let $\tilde{y}(x)$ and $\bar{y}(x)$ denote the primal solution associated with $\tilde{z}(x)$ and $\bar{z}(x)$, respectively. In order to be able to use fundamental properties such as monotonicity of the optimal value function and global optimality, which are used in the argumentation of standard stability proofs for NMPC (see, e.g., [Rawlings et al., 2017]), we will make the following assumption.

Assumption 4.2.5. *Assume that, for any $x \in \bar{\mathbb{X}}$, $\bar{z}(x)$ attains the global minimum for* (4.32)*, i.e., $V(x) = f(\bar{y}(x))$.*

Now regard (4.32) in compact form

$$\begin{aligned} V(x) := \min_{y} \quad & f(y) \\ \text{s.t.} \quad & g(x) + \hat{C}x = 0. \end{aligned} \tag{4.43}$$

The result from Lemma 4.2.4 can be used to quantify the suboptimality of $\tilde{y}(x)$ as a function of $\|x\|$.

Lemma 4.2.6. *Let Assumptions 4.2.1 and 4.2.2 hold. Then, the following holds:*

$$f\left(\tilde{y}(x)\right) - f\left(\bar{y}(x)\right) = O\left(\|x\|^4\right). \tag{4.44}$$

Proof. The proof exploits the fact that $\tilde{y}(x)$ is a feasible solution, hence $g(\tilde{y}(x)) + \hat{C}x = 0$. Thus, objective and Lagrangian coincide, i.e., $f(\tilde{y}(x)) = \mathcal{L}(\tilde{y}(x), \lambda)$, where:

$$\mathcal{L}(y, \lambda) := f(y) - \lambda^\top (g(y) + \hat{C}x). \tag{4.45}$$

The Taylor expansion of the Lagrangian at the solution $\left(\bar{y}(x), \bar{\lambda}(x)\right)$ reads

$$\mathcal{L}(y, \bar{\lambda}(x)) = \mathcal{L}(\bar{y}(x), \bar{\lambda}(x)) + \nabla_y \mathcal{L}(\bar{y}(x), \bar{\lambda}(x))^\top (y - \bar{y}(x)) + O\left(\|y - \bar{y}(x)\|^2\right).$$

Using the fact that $\bar{y}(x)$ is an optimal solution, i.e., $\nabla_y \mathcal{L}(\bar{y}(x), \bar{\lambda}(x)) = 0$, the following is obtained for $y = \tilde{y}(x)$:

$$\begin{aligned} f(\tilde{y}(x)) \quad & = \mathcal{L}(\tilde{y}(x), \bar{\lambda}(x)) \\ & = \mathcal{L}(\bar{y}(x), \bar{\lambda}(x)) + O\left(\|(\tilde{y}(x) - \bar{y}(x))\|^2\right) \\ & = f(\bar{y}(x)) + O\left(\|(\tilde{y}(x) - \bar{y}(x))\|^2\right). \end{aligned} \tag{4.46}$$

Together with the fact that $\|\tilde{z}(x) - \bar{z}(x)\| = O\left(\|x\|^2\right)$, this implies that the suboptimality grows with the fourth power of the norm of x, i.e.,

$$\|f(\tilde{y}(x)) - f(\bar{y}(x))\| = O\left(\|x\|^4\right), \tag{4.47}$$

which concludes the proof. □

The observations made until now can be exploited in order to build a Lyapunov function for the system controlled by applying the suboptimal solution in a receding horizon fashion. The following theorem states the main stability result for the inexact strategy in the equality constrained setting.

Theorem 4.2.7. *Let Assumptions 4.2.1, 4.2.2, 4.2.3 and 4.2.5 hold. Then the origin is a locally exponentially stable equilibrium for the closed-loop system*

$$x_{\text{next}} = \psi(x, \tilde{u}_0(x)) \tag{4.48}$$

obtained by applying the suboptimal control input $\tilde{u}_0(x)$ associated with $\tilde{y}(x)$.

Proof. In order to prove the above result we show that that the cost

$$\tilde{V}(x) := f(\tilde{y}(x)) \tag{4.49}$$

is a valid Lyapunov function. Define $V_{N-1}(x)$ as the optimal value function associated with a modified version of (4.32), with horizon $N-1$. For any x sufficiently close to the origin, the following holds:

$$\begin{aligned}
\tilde{V}(x) &= l(x, \tilde{u}_0(x)) + \sum_{i=1}^{N-1} l(\tilde{s}_i(x), \tilde{u}_i(x)) + m(\tilde{s}_N(x)) \\
&\overset{\text{Assumption 4.2.2}}{\geq} a_1\|x\|^2 + \sum_{i=1}^{N-1} l(\tilde{s}_i(x), \tilde{u}_i(x)) + m(\tilde{s}_N(x)) \\
&\overset{\text{optimality}}{\geq} a_1\|x\|^2 + V_{N-1}(\psi(x, \tilde{u}_0(x))) \\
&\overset{\text{monotonicity}}{\geq} a_1\|x\|^2 + V(\psi(x, \tilde{u}_0(x))),
\end{aligned} \tag{4.50}$$

where, for the last inequality, we have exploited the monotonicity property of the value function [Rawlings et al., 2017]. Moreover, due to Lemma 4.2.6, we can write

$$\tilde{V}(x) \geq a_1\|x\|^2 + \tilde{V}(\psi(x, \tilde{u}_0(x))) + O\left(\|\tilde{s}_1(x)\|^4\right). \tag{4.51}$$

This last inequality shows that $\tilde{V}(\cdot)$ decreases for $\|\tilde{s}_1(x)\|$ small enough. Since, due the Dini's Theorem, the primal solution is Lipschitz in x, i.e., $\tilde{s}_1(x) = O\left(\|x\|\right)$, the following holds:

$$\tilde{V}(x) \geq a_1\|x\|^2 + \tilde{V}(\psi(x, \tilde{u}_0(x))) + O\left(\|x\|^4\right). \tag{4.52}$$

Hence, there exists a strictly positive constant $\tilde{a}_1$ such that

$$\tilde{V}(\psi(x, \tilde{u}_0(x))) - \tilde{V}(x) \leq -\tilde{a}_1 \left\|x\right\|^2, \tag{4.53}$$

in a non-empty neighborhood of the origin.

Moreover, lower and upper bounds can be trivially constructed using Assumption 4.2.2 such that $\tilde{V}(x)$ is a valid Lyapunov function for the closed-loop system over a nonempty neighborhood of the origin. □

4.2.1 Illustrative example

In the following, in order to illustrate Theorem 4.2.14, the proposed zero-order strategy will be applied to a simple example.

We regard an OCP of the form in (4.32), with an additional terminal equality constraint as in [Zanelli et al., 2016]. In this case, the function ψ represents discretized dynamics obtained by applying the explicit Runge-Kutta strategy of order 4 with fixed step-size $h = 0.1$ to the following ordinary differential equation:

$$\dot{x} = \phi(x, u) := \begin{bmatrix} x_1^3 + (1 + x_2)u_1 \\ x_2^3 + x_1 + u_2 \end{bmatrix}. \tag{4.54}$$

A control horizon $T = 1$ is used and the trajectories are discretized using $N = 10$ shooting nodes. A quadratic cost is used and the cost matrices are chosen to be equal to the identity matrix $Q = R = \mathbb{I}_2$.

The full-step Gauss-Newton algorithm is used for the comparison and both the exact and inexact SQP-type algorithms are iterated until either convergence or failure. Two possible causes of failure are taken into account: either the algorithm has not converged after a maximum number of iterations $\tau_{\max} = 100$ or an infeasible QP has arisen.

The state space region $\mathcal{X} = \{-1.2 \leq x_1 \leq 1.2, -1.2 \leq x_2 \leq 1.2\}$ is discretized with an equally spaced grid and the OCP is solved with both methods. For any initial condition x and input computed $\bar{u}_0(x)$, let ΔV be the cost difference defined as

$$\Delta V = V\left(\psi(x, \bar{u}_0(x))\right) - V(x) \tag{4.55}$$

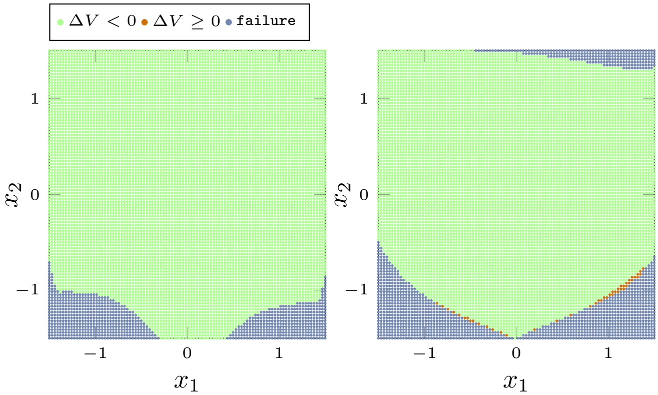


Figure 4.3: ΔV for optimal value function (left) and $\tilde{V}(x)$ (right). Decreasing cost $\Delta V < 0$ in green, non-decreasing cost $\Delta V \geq 0$ in red, maximum number of iterations reached or infeasible SQP step in blue. The optimal value function is guaranteed to decrease by construction. For the inexact strategy, the cost can be non-decreasing due to the approximation introduced. However, $\Delta V < 0$ holds in a non-negligible region around the origin.

for exact NMPC and

$$\Delta V = \tilde{V}\left(\psi(x, \tilde{u}_0(x))\right) - \tilde{V}(x) \tag{4.56}$$

for zero-order NMPC.

Figure 4.3 shows the regions where $\Delta V < 0$, $\Delta V \geq 0$ or a failure is encountered, comparing the results obtained with the two strategies. In particular, it is shown that Lyapunov decrease can be obtained for zero-order NMPC in a rather large neighbourhood of the origin.

The relative suboptimality $\epsilon_{\%} = \frac{\tilde{V}(x) - V(x)}{V(x)} \cdot 100$ of the trajectories obtained with the inexact strategy is plotted in Figure 4.4 for different initial conditions. The strategy can become largely suboptimal for points sufficiently distant from the origin, where the fixed sensitivities might not be good approximations of the exact ones. However, closed-loop feasibility is always guaranteed and stability can be guaranteed in a non-negligible region as illustrated in Figure 4.3.

Finally, in Figure 4.5, points on a line that passes through the origin parametrized with the scalar coordinate x_d are considered $[x_1\ x_2] = x_d\,[1\ 1.71]$ and the absolute suboptimality $\tilde{V}(x) - V(x)$ is compared with the stage cost $\frac{1}{2}x^\top Qx$. Due to Lemma 4.2.6, the suboptimality is of fourth order in $\|x\|$, hence the inequality $\tilde{V}(x) - V(x) < \frac{1}{2}x^\top Qx$ holds in a neighbourhood of the origin.

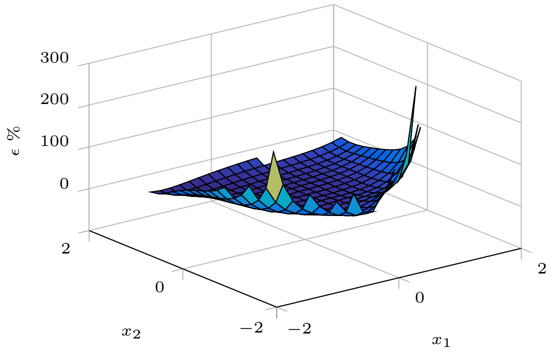


Figure 4.4: Relative suboptimality $\epsilon_{\%} = \frac{\tilde{V}(x)-V(x)}{V(x)} \cdot 100$. The inexact strategy gives rise to largely suboptimal policies in certain regions of the state space, however, as shown in Figure 4.3 stability is guaranteed in a non-negligible region of the state space.

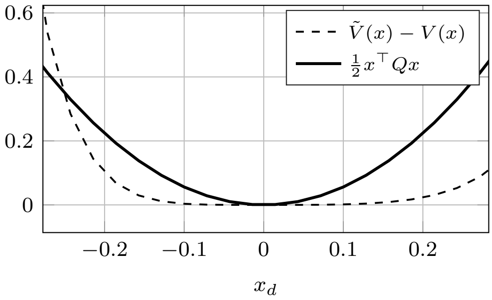


Figure 4.5: Suboptimality as a function of the directional coordinate x_d. $\tilde{V}(x) - V(x)$ is compared with $\frac{1}{2}x^\top Qx$. Due to Lemma 4.2.6, the suboptimality is dominated by the quadratic stage cost.

4.2.2 Asymptotic stability - inequality constrained

In order to extend Theorem 4.2.14 to the inequality constrained setting, we replace equations (4.37) and (4.38) with the generalized equations

$$0 \in \tilde{F}(z, x) + \mathcal{N}_K(z) \tag{4.57}$$

and

$$0 \in F(z, x) + \mathcal{N}_K(z), \tag{4.58}$$

respectively and we adapt Assumption 4.2.1 as follows.

Assumption 4.2.8. *Assume that, for $x = 0$, the generalized equations* (4.57) *are strongly regular at $z = 0$.*

Proposition 4.2.9. *Let Assumption 4.2.8 hold. Then there exists a single-valued localization $\tilde{z}(x)$, with $\tilde{z}(0) = 0$, of the solution map of* (4.57)*. Similarly, there exists a single-valued localization $\bar{z}(x)$, with $\bar{z}(x) = 0$, of the solution map of* (4.58)*. Moreover, let $\tilde{y}(x)$ and $\bar{y}(x)$ denote the primal solution associated with $\tilde{z}(x)$ and $\bar{z}(x)$, respectively.*

Proof. The result is a direct consequence of Assumption 4.2.8. □

Let $\tilde{y}(x)$ and $\bar{y}(x)$ denote the primal solution associated with $\tilde{z}(x)$ and $\bar{z}(x)$, respectively.

Proposition 4.2.10. *Regard problem* (4.11) *and let Assumption 4.2.8 hold. Then the following holds:*

$$\xi(x) = O\left(\|x\|^2\right). \tag{4.59}$$

Proof. Since $\tilde{z}(0) = 0$, due to strong regularity, $\tilde{\mu}(x) = O(\|x\|)$ and $\tilde{y}(x) = O(\|x\|)$. Moreover, since $G^\top - \nabla g(\tilde{y}(x)) = O(\|x\|)$ and $H^\top - \nabla h(\tilde{y}(x)) = O(\|x\|)$, then $\xi(x) = O(\|x\|^2)$. □

Consider now the following parametrization of problem (4.11):

$$\begin{aligned} \min_{y} \quad & f(y) + \xi^\top y \\ \text{s.t.} \quad & g(y) + \hat{C}x = 0, \\ & h(y) \leq 0, \end{aligned} \tag{4.60}$$

where ξ is regarded as a parameter. Let the following auxiliary generalized equation represents the first-order necessary conditions of (4.60):

$$0 \in \hat{F}(z, x, \xi) + \mathcal{N}_K(z), \tag{4.61}$$

where $\hat{F}(z, x, 0) = F(z, x)$.

Proposition 4.2.11. *Let Assumption 4.2.8 hold. Then there exists a single-valued localization $\hat{z}(x, \xi)$ of the solution map of* (4.61)*. Moreover, for x sufficiently close to 0, the following holds:*

$$\hat{z}(x, \xi) - \bar{z}(x) = O(\|\xi\|). \tag{4.62}$$

Proof. Existence of the single-valued localization $\hat{z}(x,\xi)$ is guaranteed by Assumption 4.2.8. Moreover, due to Corollary 2.1.33, we can write

$$\|\hat{z}(x,\xi) - \hat{z}(x,0)\| = O\left(\left\|\begin{bmatrix}x\\ \xi\end{bmatrix} - \begin{bmatrix}x\\ 0\end{bmatrix}\right\|\right) = O(\|\xi\|). \tag{4.63}$$

Moreover, since for x sufficiently small, due to strong regularity, (4.61) has a unique solution in a neighborhood of $z = 0$, $\bar{z}(x) = \hat{z}(x,0)$ holds and we can write

$$\hat{z}(x,\xi) - \bar{z}(x) = O\left(\|\xi\|\right). \tag{4.64}$$

□

Lemma 4.2.12. *Let Assumption 4.2.8 hold. Then the following is true:*

$$\bar{z}(x) - \tilde{z}(x) = O\left(\|x\|^2\right). \tag{4.65}$$

Proof. The result is a direct consequence of Propositions 4.2.10 and 4.2.11.

□

Lemma 4.2.13. *Let Assumption 4.2.8 hold. Then, the following holds:*

$$f(\bar{y}(x)) - f(\tilde{y}(x)) = O\left(\|x\|^4\right). \tag{4.66}$$

Proof. Given that $\tilde{y}(x)$ is a local minimizer for (4.11), it holds that

$$f(\tilde{y}(x)) + \xi(x)^\top \tilde{y}(x) \leq f(y) + \xi(x)^\top y, \tag{4.67}$$

for any feasible y in $\mathcal{B}(\tilde{y}(x), r_y)$ for some $r_y > 0$ and, in particular, for $\|x\|$ sufficiently small, we can write

$$f(\tilde{y}(x)) - f(\bar{y}(x)) \leq \xi(x)^\top (\bar{y}(x) - \tilde{y}(x)). \tag{4.68}$$

Moreover, due to Proposition 4.2.10 and Lemma 4.2.12, we have that $\xi(x) = O\left(\|x\|^2\right)$ and $\bar{y}(x) - \tilde{y}(x) = O\left(\|x\|^2\right)$, which implies that

$$f(\tilde{y}(x)) - f(\bar{y}(x)) = O\left(\|x\|^4\right). \tag{4.69}$$

□

Lemma 4.2.13 provides a result similar to Lemma 4.2.6, which can be used to establish local asymptotic stability stability for the zero-order NMPC strategy in the inequality constrained setting.

In order to ensure that the optimal value function is a Lyapunov function, we will rely on the standard Assumptions 2.2.16, 2.2.17, 2.2.18 and 2.2.19.

Theorem 4.2.14. *Let Assumptions 2.2.16, 2.2.17, 2.2.18, 2.2.19, 4.2.2, 4.2.3, 4.2.5 and 4.2.8 hold. Then the origin is a locally exponentially stable equilibrium for the closed-loop system*

$$x_{\text{next}} = \psi(x, \tilde{u}_0(x)) \tag{4.70}$$

obtained by applying the suboptimal control input $\tilde{u}_0(x)$ associated with $\tilde{y}(x)$.

Proof. The proof follows the same arguments as the proof of Theorem 4.2.14. ☐

4.2.3 Asymptotic stability of system-optimizer dynamics

In the following, a stability analysis for the combined system-optimizer dynamics with zero-order optimization is proposed. In particular, we show how the result from Theorem 3.2.28 can be used in order to guarantee asymptotic stability of the combined closed-loop dynamics. In particular, in Theorem 4.1.5 we have shown that a Q-linear contraction result can be derived for the zero-order iterates in the inequality constrained setting.

Theorem 4.2.15. *Let the Assumptions of Theorem 3.2.28 hold. Then, there exists a ("sufficiently" short) sampling time $T > 0$, such that the origin $(x, z) = (0, 0)$ is a locally asymptotically stable equilibrium for the combined system-optimizer dynamics obtained by controlling a system in receding horizon with the feedback policy obtained with the proposed zero-order real-time iteration strategy:*

$$\begin{aligned} x_+ &= \psi(T; x, M_{u,z} z), \\ z_+ &= \varphi(\psi(T; x, M_{u,z} z), z), \end{aligned} \tag{4.71}$$

where $\varphi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ denotes the solution map of the linearized generalized equation associated with the SQP iterates.

Proof. The proof follows the same arguments as the one of Theorem 3.2.28 choosing as Lyapunov function $f(\tilde{y}(x))$. ☐

4.3 Implementation details and benchmarking

In this section, the zero-order algorithm algorithm will be specialized to the structure obtained by using the direct collocation discretization strategy and an efficient elimination strategy based on lifted integrators [Quirynen et al., 2017] and the condensing routines proposed in [Frison and Jørgensen, 2013] will be described. The main underlying idea, as in the numerical strategy for unstructured problem proposed in Section 4.1, is to exploit the fact that the sensitivities are frozen. In this way, several computations that need in general to be carried out online can be performed offline.

The structure-exploiting algorithm is based on a two-level algebraic elimination procedure that, starting from the large and sparse direct collocation formulation, allows one to solve lower dimensional quadratic programs (QPs), with a prefactorized KKT system. First, the exact lifted integrators proposed in [Quirynen et al., 2017] which, in this context, can be seen as a strategy to eliminate the internal variables associated with implicit integrators, are used to bring the linear systems into multiple shooting form. Second, a condensing procedure is used to eliminate the state variables, such that a dense QP with fewer variables needs to be solved. Due to the fact that the sensitivities are kept constant, most computations needed for both elimination procedures can be carried out offline. These computations include factorizations needed to solve the collocation equations and the computation of the dense Hessian and its factorization. The only computations left to be carried out online consist in the evaluation of the (explicit) nonlinear functions describing cost and constraints of the NLP, condensing and expansions of right-hand sides and solution of the dense QP. For the latter, efficient rank one updates can be exploited to speed up the required computations [Ferreau et al., 2008].

Remark 4.3.1. *Although in some cases it might be more computationally efficient to use explicit integrators, the two-level elimination strategy will be described for the case where implicit integrators are used to discretize the continuous-time dynamics. This is done on purpose, since the proposed algorithm can significantly speed up the computations associated with the solution of implicit collocation equations as well.*

4.3.1 Two-level algebraic elimination

Consider the following discrete-time optimal control problem in direct collocation form:

$$\begin{aligned}
\min_{s,u,v} \quad & \sum_{i=0}^{N-1} l(s_i,u_i) + m(s_N) \\
\text{s.t.} \quad & s_0 - x = 0, \\
& \phi(s_i,u_i,v_i) = 0, \qquad i = 0,\ldots,N-1, \\
& s_i + Ev_i - s_{i+1} = 0, \quad i = 0,\ldots,N-1, \\
& \pi(s_i,u_i) \leq 0, \qquad i = 0,\ldots,N-1, \\
& \pi_N(s_N) \leq 0,
\end{aligned} \tag{4.72}$$

where $s_i \in \mathbb{R}^{n_x}$, for $i = 0,\ldots,N$ and $u_i \in \mathbb{R}^{n_u}$, $v_i \in \mathbb{R}^{n_v}$, for $i = 0,\ldots,N-1$, are the states, controls and collocation variables, respectively. The equation $\phi(s_i,u_i,v_i) = 0$ represents the collocation equations

$$\phi(s_i,u_i,v_i) := \begin{bmatrix} \psi_c(v_i^1, s_i + T_{\text{int}} \sum_{s=1}^{q} a_{1,s} v_i^s, u_i) \\ \vdots \\ \psi_c(v_i^q, s_i + T_{\text{int}} \sum_{s=1}^{q} a_{q,s} v_i^s, u_i) \end{bmatrix} \tag{4.73}$$

associated with stage i, where q denotes the number of collocation nodes and the scalars $a_{i,j}$ with $i,j = 1,\ldots,q$ are the coefficients of the collocation method. The integration step size is represented by T_{int} and E in (4.72) is a constant matrix that depends on T_{int} and the collocation nodes. The function ψ_c in (4.73) characterizes the fully implicit continuous-time dynamics

$$0 = \psi_c(\dot{x}(t), x(t), \bar{u}). \tag{4.74}$$

Remark 4.3.2. *As described in [Quirynen et al., 2017], the presented algorithm can be easily extended to the case where a differential-algebraic equation describes the dynamics, and to the case where more than one intermediate integration step is carried out per shooting node. For the sake of brevity, we will restrict ourselves to the less general formulation* (4.72)-(4.73)*.*

Algorithm 3 Level-1 elimination: lifted integrators

```
input: current primal iterate $(\hat{s}, \hat{u}, \hat{v})$
output: updated primal iterate $(s_+, u_+, v_+)$
L1-Condensing procedure
for $i = 0, \dots, N-1$ do
    $\Delta\tilde{v}_i \leftarrow \frac{\partial\phi}{\partial v}^{-1}\phi(\hat{s}_i, \hat{u}_i, \hat{v}_i)$,
    $r_{\lambda_{i+1}} \leftarrow \hat{s}_i + E\hat{v}_i - \hat{s}_{i+1} + E\Delta\tilde{v}_i$
end for
QP solution (Algorithm 4)
for $i = 0, \dots, N-1$ do
    $s_{i,+} \leftarrow \hat{s}_i + \Delta s_i$
    $u_{i,+} \leftarrow \hat{u}_i + \Delta u_i$
end for
$s_{N,+} \leftarrow \hat{s}_N + \Delta s_N$
L1-Expansion procedure
for $i = 0, \dots, N-1$ do
    $v_{i,+} \leftarrow \hat{v}_i + \Delta\tilde{v}_i + V_s\Delta s_i + V_u\Delta u_i$
end for
```

The proposed algorithm solves, at every iteration, QP subproblems of the form (4.2) with fixed sensitivities, which we report below for improved readability:

$$\begin{aligned}
\min_{\substack{\Delta s_0,\cdots,\Delta s_N\\ \Delta u_0,\cdots,\Delta u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1}\begin{bmatrix}\Delta s_i\\ \Delta u_i\\ 1\end{bmatrix}^\top M_i \begin{bmatrix}\Delta s_i\\ \Delta u_i\\ 1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\Delta s_N\\ 1\end{bmatrix}^\top M_N \begin{bmatrix}\Delta s_N\\ 1\end{bmatrix} \\
\text{s.t.} \quad & \Delta s_0 - r_{\lambda_0} = 0 \\
& \Delta s_{i+1} - A\Delta s_i - B\Delta u_i - r_{\lambda_{i+1}} = 0, \quad i = 0,\cdots,N-1, \\
& \pi(\hat{s}_i, \hat{u}_i) + C\Delta s_i + D\Delta u_i \leq 0, \qquad i = 0,\cdots,N-1, \\
& \pi_N(\hat{s}_N) + C_N\Delta s_N \leq 0.
\end{aligned} \tag{4.75}$$

Here, due to the presence of the collocation equality constraints, the matrices A and B are defined as follows:

$$A := I + EV_s, \quad B := EV_u, \tag{4.76}$$

where

$$V_s := -\frac{\partial\phi}{\partial v}^{-1}\frac{\partial\phi}{\partial s}, \quad V_u := -\frac{\partial\phi}{\partial v}^{-1}\frac{\partial\phi}{\partial u}, \quad V := [V_s,\, V_u]. \tag{4.77}$$

Algorithm 4 Level-2 elimination: states condensing

```
input: QP (4.75)
output: $\Delta s$ and $\Delta u$
1: L2-Condensing procedure
2:     update condensed QP ([Frison, 2015] - Algorithms 6, 8, 14)
3: Condensed QP solution
4:     compute $\Delta u$
5: L2-Expansion procedure
6:     compute $\Delta s$ ([Frison, 2015] - Equation 9.1)
```

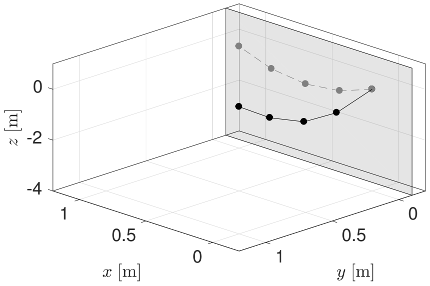


Figure 4.6: Nonlinear hanging chain benchmark for $n_m = 5$ masses [Kouzoupis et al., 2018]. The dashed sketch describes the equilibrium at which the fixed quantities used by Algorithm 3 and 4 are computed.

While the terms $r_{\lambda_i},\ i = 1, \ldots, N$ are updated using Algorithm 3, the terms $\pi(\hat{s}_i, \hat{u}_i)$ for $i = 0, \ldots, N-1$, and $\pi_N(\hat{s}_i)$ are computed by evaluating the constraint functions at the current iterate.

After problem (4.75) is formed through steps $1 - 5$ of Algorithm 3, a (states) condensing routine is used to update a condensed QP whose solution delivers the Newton step in the input variables Δu as described in Algorithm 4. Notice that, since the QP matrices in (4.75) are constant throughout the iterations, the step 5 of Algorithm 4, only involves the update of gradients and right-hand sides resulting in a tailored condensing routine that is significantly cheaper than the standard implementation.

4.3.2 Numerical results

The zero-order strategy described in Algorithms 3 and 4 has been implemented in C within the framework for nonlinear embedded optimization `acados` [Verschueren et al., 2021] using the high-performance linear algebra library

`BLASFEO` [Frison et al., 2018]. The QP solver `qpOASES` [Ferreau et al., 2008] is used in order to exploit hotstarting of its active-set strategy and avoid the necessity of factorizing the condensed Hessian at every iteration. The efficient condensing routines implemented in `HPIPM` [Frison, 2017] are used in order to carry out steps 1 and 5 in Algorithm 4. In the following, a numerical case-study based on a scalable example is presented where it is shown that considerable speedups can be achieved. All benchmarks have been run on a Dell XPS13-9360 equipped with an Intel i7-7560U with maximum and minimum frequency set to the nominal value of 2.40 GHz in order to avoid thermal throttling.

Nonlinear hanging chain - Timings

The system as presented in [Wirsching et al., 2006] and [Ferreau et al., 2008] consists in a hanging chain of masses connected by springs described by a differential equation with $n_x = 6(n_m - 2) + 3$ states, where n_m represents the number of masses in the chain. The chain is controlled by adjusting the velocities of the mass at one of its ends resulting in $n_u = 3$ controls, while the opposite end is fixed. Figure 1 shows a sketch of the system under consideration.

Following the notation in (4.72), a tracking formulation with

$$l = \frac{1}{2}(s - x_{\text{ss}})^\top Q(s - x_{\text{ss}}) + \frac{1}{2}(u - u_{\text{ss}})^\top R(u - u_{\text{ss}}) \tag{4.78}$$

and

$$m = \frac{1}{2}(s - x_{\text{ss}})^\top Q_N(s - x_{\text{ss}}) \tag{4.79}$$

is be used, where $Q = Q_N = 100 \cdot \mathbb{I}_{n_x}$ and $R = \mathbb{I}_{n_u}$. Double-sided box constraints are imposed on the inputs $u_{\min} \leq u \leq u_{\max}$ and single-sided constraints for the states $x_{\min} \leq x$ are included that represent the wall on the side of the chain. Implicit lifted collocation integrators [Quirynen et al., 2017] of type Gauss-Legendre with order $2n_s$ are used to discretize the dynamics of the system.

Table 4.1 shows the worst-case CPU time in milliseconds obtained in a closed-loop simulation using the standard (RTI) and the proposed (0-RTI) real-time iteration strategies. Especially for a large number of masses n_m and number of stages of the collocation integrators n_s, a large speedup can be achieved with respect to the standard strategy. For these benchmarks, the maximum increase in the closed-loop cost with respect to the standard RTI is below 0.1%.

N	10	10	10	20	20	20	30	30	30
n_m	5	6	7	5	6	7	5	6	7
n_s	2	4	6	2	4	6	2	4	6
RTI	1.20	4.65	19.96	2.54	9.97	40.41	4.80	16.04	61.41
0-RTI	0.43	0.44	0.94	0.81	1.32	2.34	1.84	2.65	4.23
speedup	2.79	10.56	**21.37**	3.13	7.55	**17.27**	2.61	6.05	14.52

Table 4.1: Closed-loop worst-case computation time, in milliseconds, for the standard (RTI) and zero-order (0-RTI) real-time iteration strategies for different prediction horizons N, numbers of masses n_m and numbers of stages for the collocation integrators n_s. In all simulations the system is steered to the steady state. Using the 0-RTI strategy, a maximum increase of less than 0.1% in the closed-loop cost is incurred with respect to the standard RTI.

Nonlinear hanging chain - Control performance

In order to illustrate the benefits of the proposed strategy in terms of control performance, a slight adaptation of the original formulation used in [Wirsching et al., 2006] will be taken into account. In particular, a convex quadratic constraint that requires the position of the actuated mass to be within a ball of a fixed radius $\bar{\rho}$ centered around $\bar{p}$ is introduced:

$$\|p - \bar{p}\|_2^2 - \bar{\rho}^2 \leq 0, \tag{4.80}$$

where p represents the position of the actuated mass.

Figure 2 shows the open-loop trajectories obtained by solving the exact NLP and the approximate zero-order and the linear-quadratic formulations obtained by using the required fixed quantities computed at the steady-state. Although the zero-order trajectories are clearly suboptimal, the advantage over the linear-quadratic formulation is evident due to the fact that the additional nonlinear constraint (4.80) can be satisfied when using the proposed approach.

Finally, in order to illustrate the superior asymptotic approximation of the optimal cost, Figure 4.8 shows the deviation of the open-loop costs obtained with the zero-order and linear-quadratic formulations.

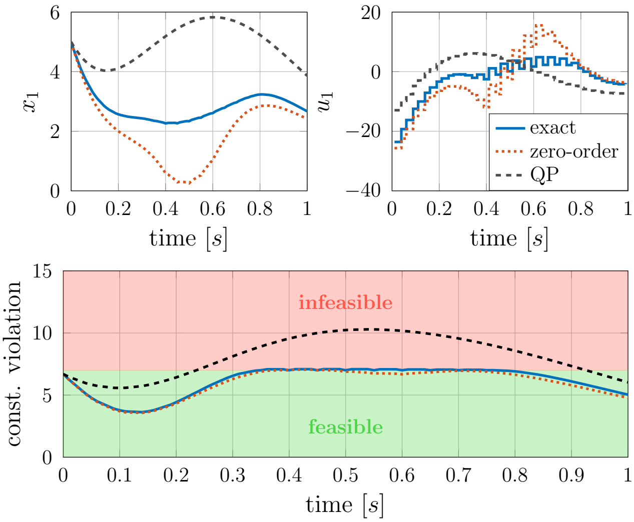


Figure 4.7: Open-loop trajectories obtained with the original (solid), zero-order (dotted) and linear-quadratic (dashed) formulations. An additional constraint is added to the problem used in [Wirsching et al., 2006] that requires the position of the actuated mass to be within a ball. Although the zero-order strategy is clearly suboptimal, the obtained trajectories satisfy the nonlinear constraint (unlike with the linear-quadratic formulation).

4.3.3 Conclusions

In Sections 4.2 and 4.3 we have presented system theoretic and computational considerations on zero-order NMPC. In particular, we introduced a proof of local asymptotic stability of the closed-loop system controlled with the feasible, but suboptimal, solutions recovered by the zero-order iterates. Section 4.3 focuses instead on computational aspects by proposing a structure exploiting algebraic elimination strategy that leverages the frozen sensitivities in the underlying QPs.

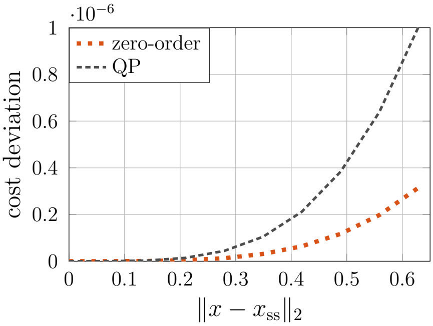


Figure 4.8: Asymptotic suboptimality of zero-order and linear-quadratic formulations in open-loop as a function of the deviation of x from the steady-state. The zero-order strategy provides a superior approximation of the optimal cost.

4.4 Chapter summary and outlook

In this chapter, we investigated numerical and system theoretic properties of algorithmic strategies based on a special case of the multi-level real-time iterations proposed in [Bock et al., 2007]. The main idea relies on the fact that, reevaluating the residuals of the constraint functions, we can enforce feasibility of properly constructed SQP iterates that can be efficiently computed. Since the QP subproblems only need function evaluations, or "zero-order" information, to be constructed, we named the class of numerical algorithms *zero-order* methods.

We analyzed how this simple, and yet powerful, idea can be specialized into different strategies for NMPC and numerical optimization in general. In Section 4.2, a stability analysis of the closed-loop system obtained using the suboptimal, but feasible, solution recovered by a zero-order SQP strategy is proposed. In particular, we showed that, under the assumption that the optimal cost is a Lyapunov function for the closed-loop system, locally, the cost associated with the suboptimal solutions is also a Lyapunov function. Moreover, an efficient structure exploiting strategy has been proposed that relies on a two-level algebraic elimination. In fact, relying on the fact that the sensitivities are held constant across the SQP iterates, it is possible to reuse many of the numerical quantities involved in the step computations. First the variables associated with implicit integrators, can be eliminated using prefactorized Jacobians using what we might refer to as a zero-order variant of the lifted integrators proposed in [Quirynen et al., 2013]. Second, the state variables can be eliminated using the condensing algorithm proposed in [Frison and Jørgensen, 2013]. The resulting routines do not update matrices and only the vector recursions need to be carried out in order to formulate and solve the QP subproblems. Finally, a condensed QP needs to be solved whose KKT system can be prefactorized in order to speed up the computations of an active-set solver. In this way, altogether, the computational footprint of one iteration becomes much closer to the one of linear-quadratic MPC, and yet the approximate solution the iterates converge to retains feasibility with respect to the (potentially) nonlinear constraints.

Undergoing research involves an efficient implementation of the feasible SQP solver using the Schur complement strategy described in [Janka et al., 2016] and available in `qpOASES` and its extensive benchmarking against state-of-the-art solvers for large-scale nonlinear programming.

Chapter 5

Progressive tightening methods for NMPC with stability guarantees

In this chapter, we analyze a class of NMPC formulations that rely on a rather intuitive concept called *progressive tightening*. In particular, we will study the system theoretic properties of progressive tightening NMPC and design numerical algorithms that can speed up computations by leveraging such formulations. The main concept behind progressive tightening is to use *stage-varying*, rather than *time-varying*, costs and constraints that, loosely speaking, lead to costs that monotonically increase in the prediction horizon. By doing so, the optimal control formulation relies on an increasing penalization of state and input deviations from their steady state values the farther into the future we look. As we will see, the use of such formulations can be motivated directly by system theoretic considerations and can be advantageously adopted to mitigate numerical challenges associated with specific ingredients of the underlying optimization problems.

Outline

The chapter is organized as follows. In Section 5.1, asymptotic stability of general progressive tightening NMPC formulations is proved under the assumption of subsequent inclusion of what will be referred to as cost-constraint epigraphs of the underlying optimal control problem. In Section 5.2, we introduce a partial tightening real-time iteration strategy that exploits a particular type of tightening based on logarithmic barrier functions on the terminal section of the prediction horizon. The resulting algorithm is interpreted within the framework of generalized Newton-type methods and a convergence and stability proof for the real-time variant of the numerical strategy is derived. Finally, in Section 5.2.5, numerical results obtained with the proposed partial tightening algorithm are presented and discussed.

5.1 Asymptotic stability of progressive tightening model predictive control

Regard the following standard optimal control problem:

$$\begin{aligned} \min_{\substack{s_0,\dots,s_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l_i(s_i,u_i) + m(s_N) \\ \text{s.t.} \quad & s_0 - x = 0, \\ & \psi(s_i,u_i) - s_{i+1} = 0, \quad i = 0,\dots,N-1, \\ & \pi_i(s_i,u_i) \le 0, \qquad\quad\ \ i = 0,\dots,N-1, \\ & \pi_N(s_N) \le 0, \end{aligned} \tag{5.1}$$

where $s \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ represent the states and controls of the system. The functions $l_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_+$, and $m : \mathbb{R}^{n_x} \to \mathbb{R}_+$ define the cost terms. The constraint functions are denoted by $\pi_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\pi}$ and $\pi_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{\pi_N}}$, while $\psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ describes the dynamics of the system. Finally, the parameter x represents the initial state of the system. We will denote the optimal solution to (5.1) for a given x as $(\bar{u}(x), \bar{s}(x))$, where $\bar{u}(x) := (\bar{u}(x)_0, \dots, \bar{u}_{N-1}(x))$ and $\bar{s}(x) := (\bar{s}_0(x), \dots, \bar{s}_N(x))$.

Remark 5.1.1. *Since the costs and constraints in progressive tightening NMPC are not time-varying, but rather stage-varying, classical results on NMPC with time-varying costs and constraints cannot be trivially applied.*

In order to formalize the concept of progressive tightening and define the class of optimal control problems that we will refer to, we introduce in the following the concept of *cost-constraint epigraphs.*

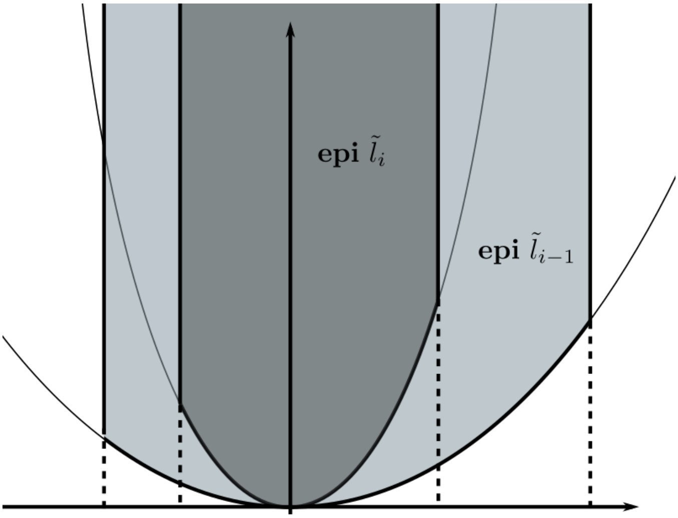


Figure 5.1: Simplified illustration of the progressive tightening Assumption 5.1.3 based on the cost-constraint epigraph inclusion.

Definition 5.1.2 (Cost-constraint epigraphs)**.** *Regard the following modified cost terms:*

$$\tilde{l}_i(s,u) := l_i(s,u) + \mathcal{I}_{\Pi_i}(s,u), \quad i = 0, \dots, N-1, \tag{5.2}$$

$$\tilde{m}(s) := m(s) + \mathcal{I}_{\Pi_N}(s), \tag{5.3}$$

where

$$\Pi_i = \{(s,u) : \pi_i(s,u) \leq 0\}, \quad i = 0, \dots, N-1, \tag{5.4}$$

$$\Pi_N = \{s : \pi_N(s) \leq 0\}. \tag{5.5}$$

Here

$$\mathcal{I}_\Omega(v) := \begin{cases} 0, & \text{if} \quad v \in \Omega \\ \infty, & \text{otherwise} \end{cases} \tag{5.6}$$

denotes the indicator function of a set Ω*. We will refer to the epigraphs of the modified cost terms as cost-constraint epigraphs.*

Assumption 5.1.3 (Progressive tightening)**.** *For the cost-constraint epigraphs associated with the costs and constraints in* (5.1)*, the following holds:*

$$\mathbf{epi}\ \tilde{l}_i \subseteq \mathbf{epi}\ \tilde{l}_{i-1}, \quad i = 1, \dots, N-1. \tag{5.7}$$

In the following, we present a stability proof for nonlinear model predictive control with stage-varying costs and constraints satisfying Assumption 5.1.3.

The main result is based on an extension of standard stability arguments [Rawlings et al., 2017] which we adapt to the problem formulation under analysis in the following. For consistency, we will refer to the notation and definitions used in [Rawlings et al., 2017]. To this end, let $\mathbb{Z}_i := \{(s,u) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mid \pi_i(s,u) \leq 0\}$, $\mathbb{U}_i(s) := \{u \in \mathbb{R}^{n_u} \mid (s,u) \in \mathbb{Z}_i\}$, $\mathbb{X}_i := \{s \in \mathbb{R}^{n_x} \mid \mathbb{U}_i(s) \neq \emptyset\}$ and $\mathbb{X}_f := \{s \in \mathbb{R}^{n_x} | \pi_N(s) \leq 0\}$. Moreover, let $\bar{\mathbb{X}} \subseteq \mathbb{X}_0$ denote the set containing all the x for which (5.1) has a solution. We require the following Assumptions to hold:

Assumption 5.1.4 (Continuity of system and cost)**.** *Assume that the origin is a steady state with* $\psi(0,0) = 0$, $l_i(0,0) = 0$, $i = 0, \dots, N-1$ *and* $m(0) = 0$*. Moreover, assume that* l_i *for* $i = 0, \dots, N-1$ *and* m *are continuous and positive definite and that* π_i *for* $i = 0, \dots, N$ *and* ψ *are continuous.*

Assumption 5.1.5 (Properties of constraint sets)**.** *For* $i = 0, \dots, N-1$*, the sets* $\mathbb{Z}_i$ *are closed and the sets* $\mathbb{U}_i(s)$ *are compact and uniformly bounded in each corresponding* $\mathbb{X}_i$*. The set* $\mathbb{X}_f \subseteq \mathbb{X}_{N-1}$ *is compact and each set contains the origin.*

Assumption 5.1.6 (Basic stability assumption)**.** *The terminal cost* m*, the set* $\mathbb{X}_f$ *and the cost terms* l_{N-1} *and* l_0 *satisfy the following properties:*

1. *For all* $x \in \mathbb{X}_f$*, there exists a* $u^\dagger(x)$*, such that* $(x, u^\dagger(x)) \in \mathbb{Z}_{N-1}$*, satisfying*
$$\psi(x, u^\dagger(x)) \in \mathbb{X}_f,$$
$$m(\psi(x, u^\dagger(x))) - m(x) \leq -l_{N-1}(x, u^\dagger(x)).$$
2. *There exist a* $\mathcal{K}_\infty$ *function* α_l *such that*
$$l_0(x,u) \geq \alpha_l(\|x\|), \ \forall x \in \bar{\mathbb{X}},$$
for any u *such that* (x,u) *is in* $\mathbb{Z}_0$*.*

Assumption 5.1.7 (Weak controllability)**.** *There exists a* $\mathcal{K}_\infty$ *function* $\alpha_2(\cdot)$ *such that, for the optimal cost associated with* (5.1)
$$V(x) := \sum_{i=0}^{N-1} l_i(\bar{s}_i(x), \bar{u}_i(x)) + m(\bar{s}_N(x)), \tag{5.8}$$
the following holds:
$$V(x) \leq \alpha_2(\|x\|), \forall x \in \bar{\mathbb{X}}. \tag{5.9}$$

Under the requirement that Assumptions 5.1.3, 5.1.4, 5.1.5, 5.1.6 and 5.1.7 hold, the main stability result can be derived. In particular, it can be shown that $V(x)$ is a Lyapunov function for the closed-loop system obtained by controlling the system with the optimal feedback law $M_{u,y}\bar{y}(x)$ in a receding horizon fashion [Rawlings et al., 2017]:

Theorem 5.1.8 (Asymptotic stability of progressive tightening NMPC)**.** *Suppose that Assumptions 5.1.3, 5.1.4, 5.1.5, 5.1.6 and 5.1.7 are satisfied. Then the following hold.*

1. *There exist $\mathcal{K}_\infty$ functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ and a positive definite function $\alpha_3(\cdot)$ such that*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|), \tag{5.10a}$$

$$V(\psi(x, M_{u,y}\bar{y}(x))) - V(x) \leq -\alpha_3(\|x\|), \tag{5.10b}$$

for all $x \in \bar{\mathbb{X}}$.

2. *The origin $x = 0$ is an asymptotically stable equilibrium with region of attraction $\bar{\mathbb{X}}$ for the closed-loop system*

$$x_{\text{next}} = \psi(x, \bar{u}_0(x)). \tag{5.11}$$

Proof. First, it is necessary to identify valid candidates for the functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ such that the inequality (5.10a) is satisfied. The fact that a $\mathcal{K}_\infty$ function $\alpha_2(\cdot)$ exists such that

$$V(x) \leq \alpha_2(\|x\|), \quad \forall x \in \bar{\mathbb{X}}$$

is a direct consequence of Assumption 5.1.7. Due to Assumption 5.1.6, we have

$$V(x) \geq l_0(x, \bar{u}(x)) \geq \alpha_l(\|x\|), \quad \forall x \in \bar{\mathbb{X}},$$

for any u such that (x, u) is in $\mathbb{Z}_0$, which implies that $\alpha_1(\cdot) = \alpha_l(\cdot)$ can be used.

Second, it is necessary to show that the decrease property in (5.10b) holds. To this end, consider the following control sequence obtained by shifting the solution obtained by solving (5.1) for a given x:

$$\tilde{u} = (\bar{u}_1(x), \ldots, \bar{u}_{N-1}(x), \tilde{u}_N),$$

where $\tilde{u}_N := u^\dagger(\bar{s}_N(x))$ is defined according to Assumption 5.1.6. Due to Assumption 5.1.3 on the recursive inclusion of the cost-constraint epigraphs, $\tilde{u}$

is feasible for the optimal control problem associated with the shifted horizon, since the inclusion **epi** $\tilde{l}_i \subseteq$ **epi** $\tilde{l}_{i-1}$ implies $\mathbb{Z}_i \subseteq \mathbb{Z}_{i-1}$. The cost associated with the suboptimal sequence $\tilde{u}$ reads

$$\begin{aligned}\tilde{V}(\bar{s}_1(x)) &= \sum_{i=1}^{N-1} l_{i-1}(\bar{s}_i(x), \bar{u}_i(x)) + l_{N-1}(\bar{s}_N(x), \tilde{u}_N) + m(\psi(\bar{s}_N(x), \tilde{u}_N)) \\ &= V(x) - l_0(x, \bar{u}_0(x)) - \sum_{i=1}^{N-1} l_i(\bar{s}_i(x), \bar{u}_i(x)) - m(\bar{s}_N(x)) \\ &\quad + \sum_{i=1}^{N-1} l_{i-1}(\bar{s}_i(x), \bar{u}_i(x)) + l_{N-1}(\bar{s}_N(x), \tilde{u}_N) + m(\psi(\bar{s}_N(x), \tilde{u}_N)).\end{aligned}$$

Again, due to the cost-constraint epigraph inclusion in Assumption 5.1.3, the following holds:

$$\sum_{i=1}^{N-1} l_{i-1}(\bar{s}_i(x), \bar{u}_i(x)) - \sum_{i=1}^{N-1} l_i(\bar{s}_i(x), \bar{u}_i(x)) \leq 0,$$

which implies

$$\tilde{V}(\bar{s}_1(x)) \leq V(x) - l_0(x, \bar{u}_0(x)) - m(\bar{s}_N(x)) + l_{N-1}(\bar{s}_N(x), \tilde{u}_N) + m(\psi(\bar{s}_N(x), \tilde{u}_N))$$

and, due to Assumption 5.1.6, we have that

$$\tilde{V}(\bar{s}_1(x)) - V(x) \leq -l_0(x, \bar{u}_0(x)) \leq \alpha_l(\|x\|). \tag{5.12}$$

This last inequality shows that (5.10b) holds with $\alpha_3(\cdot) = \alpha_l(\cdot)$.

Finally, in order to conclude the proof, Theorem 2.2.15 can be used. The set $\bar{\mathbb{X}}$ can be shown to be positive invariant for the closed-loop system following the arguments in the proof of [Rawlings et al., 2017, Proposition 2.10(b)]. Moreover, inequalities (2.80a) and (2.80b) are trivially satisfied through (2.83a) and (2.83b). This implies that Theorem 2.2.15 can be applied and the origin can be shown to be asymptotically stable in $\bar{\mathbb{X}}$. ☐

5.2 A partial tightening real-time method

In this section, we propose a numerical strategy based on progressive tightening that can be used to reduce the computational burden associated with the solution of problems arising in nonlinear model predictive control. The prediction horizon is split into two sections and the constraints associated with the terminal one are tightened using a barrier formulation. In this way, when using the RTI strategy, variables associated with such stages can be efficiently eliminated from the quadratic subproblems by a single backward Riccati sweep. After eliminating the tightening stages, a quadratic problem with a reduced horizon is solved where the original constraints are used. The solution is then expanded to the full horizon with a single forward Riccati sweep. By doing so, the online computational burden associated with the solution of the optimization problems can be largely reduced.

Regard the following optimal control problem:

$$\begin{aligned}\min_{\substack{s_0,\dots,s_N\\u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} l(s_i,u_i) + m(s_N)\\ \text{s.t.} \quad & s_0 - x = 0\\ & s_{i+1} = \psi(s_i,u_i), \quad i = 0,\dots,N-1,\\ & \pi(s_i,u_i) \leq 0, \qquad i = 0,\dots,N-1,\\ & \pi_N(s_N) \leq 0,\end{aligned} \tag{5.13}$$

where l, m, ψ, π and π_N are twice continuously differentiable and π and π_N are convex.

Remark 5.2.1. *While, from a computational point of view, the proposed algorithm can be applied to a more general problem formulation with potentially nonconvex inequality constraints, convexity of π and π_N will be assumed in order to be able to employ recentered barrier functions. These become in fact necessary to construct a Lyapunov function for the proposed method.*

We are interested in solving (5.13) with an SQP-type algorithm. In order to reduce the computational burden associated with several linearizations and QP solutions at each sampling instant, the RTI strategy can be used. This allows the algorithm to converge over time and achieve faster control feedback to the system. In the following we propose a numerical strategy based on progressive tightening that can be used to speed up the solution of the QPs associated with the RTI.

5.2.1 The partial tightening real-time strategy

In the following, we propose a partial tightening formulation that allows one to speed up the computations necessary to solve (5.13). For stages $i = M,\dots,N$, the constraints are removed and the stage costs are replaced with the modified costs

$$\begin{aligned} \tilde{l}(s,u) &= \hat{l}(s,u) + \rho(s,u), \\ \tilde{m}(s) &= \hat{m}(s) + \rho_N(s), \end{aligned} \tag{5.14}$$

where the logarithmic barriers

$$\begin{aligned} \rho(s,u) &:= -\tau \cdot \sum_{j=1}^{n_g} \log(-\pi_j(s,u)), \\ \rho_N(s) &:= -\tau \cdot \sum_{j=1}^{n_{g_N}} \log(-\pi_{Nj}(s)), \end{aligned} \tag{5.15}$$

have been introduced and where the cost terms have been adapted to include the associated gradient recentering terms

$$\begin{aligned} \hat{l}(s,u) &= l(s,u) - \rho(0,0) - \nabla_s \rho(0,0)^\top s - \nabla_u \rho(0,0)^\top u, \\ \hat{m}(s) &= m(s) - \rho_N(0) - \nabla_s \rho_N(0)^\top s. \end{aligned} \tag{5.16}$$

Remark 5.2.2. *Notice that the inclusion of the recentering terms is only necessary in order to be able to derive the system theoretic properties presented in Section 5.2.4 and one can in principle set $\hat{l}(s,u) = l(s,u)$ and $\hat{m}(s) = m(s)$ resulting in a completely valid algorithm.*

The resulting parametric nonlinear program reads

$$\begin{aligned} \min_{\substack{s_0,\ldots,s_N \\ u_0,\ldots,u_{N-1}}} \quad & \sum_{i=0}^{M-1} l(s_i,u_i) + \sum_{i=M}^{N-1} \tilde{l}(s_i,u_i) + \tilde{m}(s_N) \\ \text{s.t.} \quad & s_0 - x = 0 \\ & s_{i+1} = \psi(s_i,u_i), \quad i = 0,\ldots,N-1, \\ & \pi(s_i,u_i) \leq 0, \qquad i = 0,\ldots,M-1. \end{aligned} \tag{5.17}$$

Note that such a formulation is also used in interior-point methods in order to cope with the nonsmoothness of the complementarity conditions. In those methods, the so-called barrier parameter τ is shrunk as the algorithm proceeds in order to obtain a solution to the original problem as $\tau \to 0$ [Nocedal and Wright, 2006].

In the context of model predictive control, strategies that exploit a fixed value of the barrier parameter and tightening of the entire horizon have been investigated by the authors of [Wills and Heath, 2004] and [Feller and Ebenbauer, 2016]. In the first work, for nonlinear systems, the stabilizing properties are described for the feedback policy that is obtained by solving the tightening optimal control problems for a fixed value of the barrier parameter. The second work, for linear systems, presents an approximate strategy that requires, in the limit, performing a single iteration per subproblem. Convergence and stability of such an algorithm is analyzed in [Feller and Ebenbauer, 2016]. Finally, the approach in [Ohtsuka, 2015] shares some similarities with the work in [Wills and Heath, 2004] and [Feller and Ebenbauer, 2016] in the sense that the C/GMRES iterations can be interpreted as interior-point-like iterations on problems with a particular barrier formulation [Diehl et al., 2009].

In the following, a similar approach is used with the important difference that only stages from M to N will be tightened in order not to require a modification of the constraints in the first stages. Moreover, instead of solving exactly the tightening problems, a formulation is proposed that results in QP subproblems with linearized complementarity conditions for the tightening stages. In this way, the variables associated with such stages can be efficiently eliminated with a Riccati-like recursion. After elimination, a smaller QP is obtained, with a reduced horizon that can be efficiently solved with QP solvers tailored to MPC.

Real-time iterations with partial tightening

It can be easily shown that the first-order optimality conditions associated with problem (5.13), with stage costs modified according to the barrier formulation in (5.14) and (5.15) can be written as:

$$\begin{aligned}
&-s_0 + x &&= 0\\
&\nabla_s l(s_0,u_0) + \nabla_s \psi(s_0,u_0)\lambda_1 - \lambda_0 + \nabla_s \pi(s_0,u_0)\nu_0 &&= 0\\
&\nabla_u l(s_0,u_0) + \nabla_u \psi(s_0,u_0)\lambda_1 + \nabla_u \pi(s_0,u_0)\nu_0 &&= 0\\
&\pi(s_0,u_0) + t_0 &&= 0\\
&T_0\nu_0 &&= 0\\
&\qquad\dots
\end{aligned}\tag{5.18}$$

for stages 0 to $M-1$, and

$$\begin{aligned}
&\dots\\
&-s_M + \psi(s_{M-1}, u_{M-1}) && = 0\\
&\nabla_s \hat{l}(s_M, u_M) + \nabla_s \psi(s_M, u_M)\lambda_{M+1} - \lambda_M + \nabla_s \pi(s_M, u_M)\nu_M && = 0\\
&\nabla_u \hat{l}(s_M, u_M) + \nabla_u \psi(s_M, u_M)\lambda_{M+1} + \nabla_u \pi(s_M, u_M)\nu_M && = 0\\
&\pi(s_M, u_M) + t_M && = 0\\
&T_M \nu_M && = \tau \mathbf{1} \qquad (5.19)\\
&\dots\\
&-s_N + \psi(s_{N-1}, u_{N-1}) && = 0\\
&\nabla_s \hat{m}(s_N) - \lambda_N + \nabla_s \pi(s_N)\nu_N && = 0\\
&\pi(s_N) + t_N && = 0\\
&T_N \nu_N && = \tau \mathbf{1},
\end{aligned}$$

for stages M to N. Here ν_i, $t_i \geq 0$ are the Lagrange multipliers and slacks, respectively, associated with the stage inequalities and λ_i are the equality constraint multipliers. Moreover, T_i denotes the diagonal matrix having the elements of s_i on its diagonal. In order to compute a solution to (5.18)-(5.19), we regard a Newton-type method based on the following linearized problem:

$$\begin{aligned}
&-\Delta s_0 && = -r_{\lambda_0}\\
&-\Delta\lambda_0 + H_0^{ss}\Delta s_0 + H_0^{su}\Delta u_0 + C_0^\top \Delta\nu_0 + A_0^\top \Delta\lambda_1 && = -r_{s_0}\\
&H_0^{us}\Delta s_0 + H_0^{uu}\Delta u_0 + D_0^\top \Delta\nu_0 + B_0^\top \Delta\lambda_1 && = -r_{u_0} \qquad (5.20)\\
&C_0 \Delta s_0 + D_0 \Delta u_0 + \Delta t_0 && = -r_{\nu_0}\\
&(\hat{T}_0 + \Delta T_0)(\hat{\nu}_0 + \Delta\nu_0) && = 0\\
&\dots
\end{aligned}$$

for stages 0 to $M-1$, and

$$\begin{aligned}
&-\Delta s_M + A_{M-1}\Delta s_{M-1} + B_{M-1}\Delta u_{M-1} && = -r_{\lambda_M}\\
&-\Delta\lambda_M + H_M^{ss}\Delta s_M + H_M^{su}\Delta u_M + C_M^\top\Delta\nu_M + A_M^\top\Delta\lambda_{M+1} && = -r_{s_M}\\
&H_M^{us}\Delta s_M + H_M^{uu}\Delta u_M + D_M^\top\Delta\nu_M + B_M^\top\Delta\lambda_{M+1} && = -r_{u_M}\\
&C_M\Delta s_M + D_M\Delta u_M + \Delta t_M && = -r_{\nu_M}\\
&\hat{T}_M\Delta\nu_M + \hat{V}_M\Delta t_M && = -e_M\\
&\quad\dots\\
&-\Delta s_N + A_{N-1}\Delta s_{N-1} + B_{N-1}\Delta u_{N-1} && = -r_{\lambda_N}\\
&-\Delta\lambda_N + H_N^{ss}\Delta s_N + C_N^\top\Delta\nu_N && = -r_{s_N}\\
&C_N\Delta s_N + \Delta t_N && = -r_{\nu_N}\\
&\hat{T}_N\Delta\nu_N + \hat{V}_N\Delta t_N && = -e_N
\end{aligned} \tag{5.21}$$

for stages M to N. Here

$$\begin{aligned}
A_i &:= \nabla_s\psi(\hat{s}_i,\hat{u}_i)^\top, \quad B_i := \nabla_u\psi(\hat{s}_i,\hat{u}_i)^\top,\\
C_i &:= \nabla_s\pi(\hat{s}_i,\hat{u}_i)^\top, \quad D_i := \nabla_u\pi(\hat{s}_i,\hat{u}_i)^\top,
\end{aligned} \tag{5.22}$$

have been introduced that represent evaluations of the Jacobians of equality and inequality constraints at the linearization point $(\hat{s}_i,\hat{u}_i)$. The matrices $\hat{T}_i$ and $\hat{V}_i$ are the diagonal matrices having the elements of $\hat{t}_i$ and $\hat{\nu}_i$ on their diagonals. The matrices H_i^{ss}, $H_i^{su}=(H_i^{us})^\top$ and H_i^{uu} denote the approximations of the Hessians of the Lagrangian with respect to input and state variables at stage i. Finally, we have introduced the vectors r_{λ_i}, r_{s_i}, for $i=0,\dots,N$, r_{u_i} for $i=0,\dots,N-1$ and e_i for $i=M,\dots,N$, to denote the residuals defined according to (5.18)-(5.19).

With the standard elimination procedure described, for example, in [Rao et al., 1998] we can eliminate the multipliers and slack variables:

$$\begin{aligned}
\Delta t_N &= \hat{V}_N^{-1}(-\hat{T}_N\Delta\nu_N - e_N),\\
\Delta\nu_N &= \hat{V}_N\hat{T}_N^{-1}(r_{\nu_N} - \hat{V}_N^{-1}e_N + C_N\Delta s_N),
\end{aligned} \tag{5.23}$$

and, for $i = M, \dots, N-1$,

$$\begin{aligned} \Delta t_i &= \hat{V}_i^{-1}(-\hat{T}_i \Delta\nu_i - e_i), \\ \Delta\nu_i &= \hat{V}_i \hat{T}_i^{-1}(r_{\nu_i} - \hat{V}_i^{-1} e_i + C_i \Delta s_i + D_i \Delta u_i), \end{aligned} \tag{5.24}$$

such that we obtain the following simplified set of equations:

$$\begin{aligned} & -\Delta s_M + A_{M-1}\Delta s_{M-1} + B_{M-1}\Delta u_{M-1} && = -r_{\lambda_M} \\ & -\Delta\lambda_M + \tilde{H}_M^{ss}\Delta s_M + \tilde{H}_M^{su}\Delta u_M + A_M^\top \Delta\lambda_{M+1} && = -\tilde{r}_{s_M} \\ & \tilde{H}_M^{us}\Delta s_M + \tilde{H}_M^{uu}\Delta u_M + B_M^\top \Delta\lambda_{M+1} && = -\tilde{r}_{u_M} \\ & \quad \dots && \dots \\ & -\Delta s_N + A_{N-1}\Delta s_{N-1} + B_{N-1}\Delta u_{N-1} && = -r_{\lambda_N} \\ & -\Delta\lambda_N + \tilde{H}_N^{ss}\Delta s_N && = -\tilde{r}_{s_N}, \end{aligned} \tag{5.25}$$

with the following updated Hessian and right-hand side terms:

$$\begin{aligned} \tilde{H}_i^{ss} &:= H_i^{ss} + C_i^\top \hat{V}_i \hat{T}_i^{-1} C_i, & \tilde{H}_i^{su} &:= H_i^{su} + C_i^\top \hat{V}_i \hat{T}_i^{-1} D_i, \\ \tilde{H}_i^{us} &:= H_i^{ss} + C_i^\top \hat{V}_i \hat{T}_i^{-1} C_i, & \tilde{H}_i^{uu} &:= H_i^{uu} + D_i^\top \hat{V}_i \hat{T}_i^{-1} D_i, \end{aligned} \tag{5.26}$$

and

$$\begin{aligned} \tilde{r}_{s_i} &:= r_{s_i} + C_i^\top V_i T_i^{-1} r_{\nu_i} - C_i^\top T_i^{-1} e_i, \\ \tilde{r}_{u_i} &:= r_{u_i} + D_i^\top V_i T_i^{-1} r_{\nu_i} - D_i^\top T_i^{-1} e_i, \end{aligned} \tag{5.27}$$

for $i = M, \dots, N-1$, and

$$\begin{aligned} \tilde{H}_N^{ss} &:= H_N^{ss} + C_N^\top \hat{V}_N \hat{T}_N^{-1} C_N, \\ \tilde{r}_{s_N} &:= r_{s_N} + C_N^\top V_N T_N^{-1} r_{\nu_N} - C_N^\top T_N^{-1} e_N. \end{aligned} \tag{5.28}$$

Notice that, if an SQP step is applied to the partial tightening problem, where logarithmic barriers are used explicitly in the cost, analogously, a set of linear equations would be obtained for the terminal section of the horizon and the strategy would naturally fall in the standard SQP framework. However, for numerical reasons, it might be convenient instead to use the formulation proposed above. Although different intermediate iterations would be taken by the two strategies in general, if convergence is achieved, a solution to (5.18)-(5.19) is obtained by both algorithms.

Notice that, when applying the standard RTI strategy, the presence of constraints gives rise to nonsmooth equations that require special treatment. If, for example, the resulting QP is solved with an interior-point method, the complementarity conditions for the first M stages need to be relaxed and iteratively solved within one SQP iteration. In the proposed strategy, the smoothed complementarity conditions for stages $i = M, \dots, N$ have been linearized in (5.20)-(5.21) such that the computational burden associated with one RTI iteration can be largely reduced. In particular, a single backward Riccati recursion [Rao et al., 1998] can be used to factorize the part of the KKT matrix associated with the tightening stages. These implementation aspects are discussed further in Section 5.2.2.

5.2.2 Efficient implementation

In this section, the implementation of the proposed partial tightening algorithm is discussed. An efficient Riccati recursion based on the algorithm proposed in [Rao et al., 1998] is used to eliminate variables associated with the tightening stages in order to obtain, after elimination, a reduced QP with a shorter horizon. Note that the solution of the resulting QP is expanded back into the solution for the original long horizon problem.

The backward Riccati recursion

To perform an RTI, system (5.20)-(5.21) needs to be solved, including the positivity constraints for the slack variables and Lagrange multipliers. For stages M to N, it can be shown that the linear system associated with a Newton step has the form of the KKT system that arises from a linear-quadratic problem [Rao et al., 1998]. In particular, after eliminating slack variables and inequality multipliers, this leads to a system with a special band diagonal structure that can be exploited in order to reduce the computational burden. For example, for

$N = 4$ and $M = 2$, these equations, for stages M to N, would read

$$\begin{bmatrix} & -I & & & & & & \\ -I & \tilde{H}_2^{ss} & \tilde{H}_2^{su} & A_2^\top & & & & \\ & \tilde{H}_2^{us} & \tilde{H}_2^{uu} & B_2^\top & & & & \\ & A_2 & B_2 & & -I & & & \\ & & & -I & \tilde{H}_3^{ss} & \tilde{H}_1^{su} & A_3^\top & \\ & & & & \tilde{H}_3^{us} & \tilde{H}_3^{uu} & B_3^\top & \\ & & & & A_3 & B_3 & & -I \\ & & & & & & -I & \tilde{H}_4^{ss} \end{bmatrix} \begin{bmatrix} \Delta\lambda_2 \\ \Delta s_2 \\ \Delta u_2 \\ \Delta\lambda_3 \\ \Delta s_3 \\ \Delta u_3 \\ \Delta\lambda_4 \\ \Delta s_4 \end{bmatrix} = - \begin{bmatrix} r_{\lambda_2} \\ \tilde{r}_{s_2} \\ \tilde{r}_{u_2} \\ r_{\lambda_3} \\ \tilde{r}_{s_3} \\ \tilde{r}_{u_3} \\ r_{\lambda_4} \\ \tilde{r}_{s_4} \end{bmatrix}. \tag{5.29}$$

It is possible to factorize the matrix in (5.29) starting from the block corresponding to stage N using the standard Riccati recursion

$$P_i = \tilde{H}_i^{ss} + A_i^\top P_{i+1} A_i + \Sigma_i (\tilde{H}_i^{us} + B_i^\top P_{i+1} A_i) \tag{5.30}$$

and

$$p_i = \tilde{r}_{s_i} + A_i^\top (P_{i+1} r_{\lambda_{i+1}} + p_{i+1}) + \Sigma_i (\tilde{r}_{u_i} + B_i^\top P_{i+1} r_{\lambda_{i+1}} + B_i^\top p_{i+1}), \tag{5.31}$$

where

$$\Sigma_i := -(\tilde{H}_i^{su} + A_i^\top P_{i+1} B_i)(\tilde{H}_i^{uu} + B_i^\top P_{i+1} B_i)^{-1}, \tag{5.32}$$

for $i = N-1, \dots, M$ and initialized with $P_N = \tilde{H}_N^{ss}$ and $p_N = \tilde{r}_{s_N}$, in order to obtain P_M and p_M.

The reduced QP subproblem and forward expansion

Once the variables associated with stages M to N have been eliminated, the following reduced QP with shorter horizon is left to be solved:

$$\begin{aligned} \min_{\substack{\Delta s_0, \dots, \Delta s_M \\ \Delta u_0, \dots, \Delta u_{M-1}}} \quad & \sum_{i=0}^{M-1} \begin{bmatrix} \Delta s_i \\ \Delta u_i \\ 1 \end{bmatrix}^\top H_i \begin{bmatrix} \Delta s_i \\ \Delta u_i \\ 1 \end{bmatrix} + m_M(\Delta s_M) \\ \text{s.t.} \quad & \Delta s_0 - r_{\lambda_0} = 0, \\ & \Delta s_{i+1} - A_i \Delta s_i - B_i \Delta u_i - r_{\lambda_i} = 0, \quad i = 0, \dots, M-1, \\ & \pi(\hat{s}_i, \hat{u}_i) + C_i \Delta s_i + D_i \Delta u_i \leq 0, \qquad i = 0, \dots, M-1, \end{aligned} \tag{5.33}$$

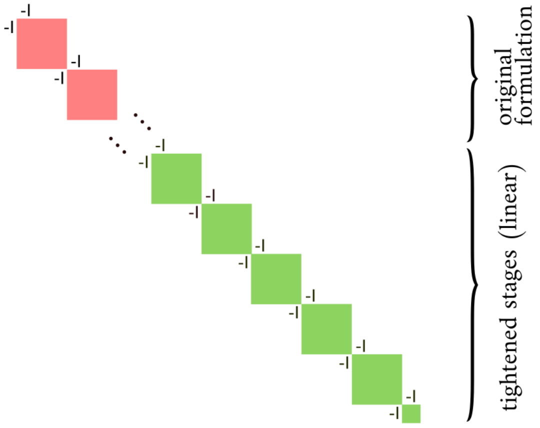


Figure 5.2: Backward and forward Riccati recursion. The variables associated with stages M to N (in green) can be efficiently eliminated due to the (relaxed) linearized complementarity conditions.

where the terminal cost for stage M is defined as

$$m_M(\Delta s_M) := \Delta s_M^\top P_M \Delta s_M + p_M^\top \Delta s_M, \tag{5.34}$$

with P_M and p_M both resulting from the backward recursion on stages N to M in (5.30) and (5.31). Moreover, for ease of notation, we have introduced the Hessians H_i, for $i = 0, \dots, M-1$, defined as follows:

$$H_i := \begin{bmatrix} H_i^{ss} & H_i^{su} & \nabla_s l_i \\ H_i^{us} & H_i^{uu} & \nabla_u l_i \\ \nabla_s l_i^\top & \nabla_u l_i^\top & 1 \end{bmatrix}. \tag{5.35}$$

Remark 5.2.3. *The procedure described so far can be interpreted as a way of building a terminal cost $m_M(s_M)$ for the MPC problem* (5.33) *with shorter horizon. Compared to the original OCP in* (5.13), *$m_M(s_M)$ incorporates an approximate contribution due to stages M to N, which can be efficiently computed thanks to the partial tightening formulation that is adopted.*

The resulting QP in (5.33) can be readily solved with a standard QP solver tailored to MPC such as `qpOASES` [Ferreau et al., 2014], `HPMPC` [Frison et al., 2014] or `FORCES PRO` [Domahidi et al., 2012b]. After solving the QP subproblem, a forward recursion is used to update the solution for the tightening stages. In order to use the standard formulation of the Riccati recursion, we can rewrite

(5.29) as

$$\begin{bmatrix} \tilde{H}_2^{uu} & B_2^\top & & & & \\ B_2 & & -I & & & \\ & -I & \tilde{H}_3^{ss} & \tilde{H}_3^{su} & A_3^\top & \\ & & \tilde{H}_3^{us} & \tilde{H}_3^{uu} & B_3^\top & \\ & & A_3 & B_3 & & -I \\ & & & & -I & \tilde{H}_4^{ss} \end{bmatrix} \begin{bmatrix} \Delta u_2 \\ \Delta\lambda_3 \\ \Delta s_3 \\ \Delta u_3 \\ \Delta\lambda_4 \\ \Delta s_4 \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{u_2} + \tilde{H}_2^{us}\Delta s_2 \\ r_{\lambda_1} + A_2 r_{\lambda_2} \\ \tilde{r}_{s_3} \\ \tilde{r}_{u_3} \\ r_{\lambda_4} \\ \tilde{r}_{s_4} \end{bmatrix}, \tag{5.36}$$

such that we can compute $\Delta\lambda_M$ and Δu_M by solving the following linear system:

$$\begin{bmatrix} \tilde{H}_2^{uu} & B_2^\top & \\ B_2 & & -I \\ & -I & P_3 \end{bmatrix} \begin{bmatrix} \Delta u_2 \\ \Delta\lambda_3 \\ \Delta s_3 \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{u_2} + \tilde{H}_2^{us}\Delta s_2 \\ r_{\lambda_3} + A_2 r_{\lambda_2} \\ p_3 \end{bmatrix}. \tag{5.37}$$

Finally, we complete the forward recursion with

$$\begin{aligned} \Delta u_i \quad &= K_i \Delta s_i + k_i, \\ \Delta s_{i+1} \quad &= A_i \Delta s_i + B_i \Delta u_i + r_{\lambda_{i+1}} \\ \Delta\lambda_i \quad &= P_i \Delta s_i + p_i, \end{aligned} \tag{5.38}$$

for $i = M+1, \ldots, N-1$, where

$$\begin{aligned} K_i \quad &= -\Gamma_i \, B_i^\top P_{i+1} A_i, \\ k_i \quad &= -\Gamma_i \left(\tilde{r}_{u_i} + B_i^\top (P_{i+1} r_{\lambda_{i+1}} + p_{i+1}) \right) \\ \Gamma_i \quad &= (\tilde{H}_i^{uu} + B_i^\top P_{i+1} B_i)^{-1}. \end{aligned} \tag{5.39}$$

Using equations (5.23) and (5.24), we can expand the solution to the full space. Once the solution $\Delta\bar{z} := (\Delta\bar{\lambda}, \Delta\bar{s}, \Delta\bar{u}, \Delta\bar{\nu}, \Delta\bar{s})$ to equations (5.20)-(5.21) has been computed, an inexact backtracking line-search is used to obtain a feasible step:

$$\hat{z}_+ = \hat{z} + \alpha \Delta\bar{z}, \tag{5.40}$$

where $\alpha \in (0, 1]$ such that the positivity constraints on ν_i and s_i are satisfied. Notice that the positivity constraints for stages 0 to $M-1$ are always satisfied, if a feasible solution to the reduced QP is obtained. Hence, for a practical implementation, the line-search might be limited to the tightening stages. The proposed strategy is summarized in Algorithm 5 and the special block-banded structure with linear stages is visualized in Figure 5.2.

Remark 5.2.4. *Notice that x is only necessary at line 10 of Algorithm 5. For this reason, when splitting the computations in preparation and feedback phase according to [Diehl et al., 2007], the feedback phase consists only of the solution of the reduced QP, and forward expansion which can further reduce the feedback delays.*

Algorithm 5 Partial tightening real-time iteration

```
 1: input: $\hat{z} := (\hat{\lambda}, \hat{s}, \hat{u}, \hat{\nu}, \hat{t})$, $\tau$
 2: linearization of (5.18)-(5.19)
 3: - compute
 4:         $A_i, B_i, D_i,$              for $i = 0, \dots, N-1,$
 5:         $C_i, r_{\nu_i}, r_{s_i}, r_{\lambda_i},$  for $i = 0, \dots, N,$
 6:         $\hat{T}_i, \hat{V}_i, e_i,$              for $i = M, \dots, N,$
 7: reduction to equality constrained form $(N \rightarrow M)$:
 8: - eliminate $\Delta s_i, \Delta \nu_i$, for $i = M, \dots, N$ according to [Rao et al., 1998]
 9: backward Riccati sweep $(N \rightarrow M)$:
10: - compute $P_M$ and $p_M$ using (5.30) and (5.31)
11: estimate new initial state $x$
12: QP solution (5.33):
13: - compute $\Delta\bar{s}_i, \Delta\bar{\lambda}_i,$   for $i = 0, \dots, M$
14: - and $\Delta\bar{u}_i, \Delta\bar{\nu}_i,$        for $i = 0, \dots, M-1$
15: forward Riccati sweep $(M \rightarrow N)$:
16: - compute $\Delta\bar{s}_i, \Delta\bar{\lambda}_i,$ for $i = M, \dots, N$
17: - and $\Delta\bar{u}_i,$ for $i = M, \dots, N-1$ using (5.38) and (5.39)
18: expansion $(M \rightarrow N)$:
19: - compute $\Delta\bar{t}_i, \Delta\bar{\nu}_i,$ for $i = M, \dots, N$ according to [Rao et al., 1998]
20: line-search:
21: - compute $z_+$ using (5.40)
22: output: $z_+$
```

5.2.3 Convergence

In order to analyze the contraction properties of the proposed partial tightening strategy, we will interpret the iterations described in Algorithm 5 as the iterations of a generalized Newton-type method applied to a properly defined generalized equation. In fact, although a similar algorithm applied to an "unlifted" formulation (5.17) where barrier functions appear explicitly in the cost can be easily interpreted as an SQP-type algorithm, the same does not hold in this case. The presence of both smoothed and unsmoothed complementarity constraints in (5.20)-(5.21) makes it more natural to analyze instead the iterates using the framework of generalized Newton methods. This allows one to avoid (implicitly or explicitly) referring to the underlying optimization problem whose first-order optimality conditions are expressed by (5.20)-(5.21). To this end, we rewrite the conditions expressed in (5.18)-(5.19) together with the associated positivity constraints on slack and inequality multipliers as

$$
\begin{aligned}
& x - s_0 && = 0 \\
& \nabla_s l(s_0, u_0) + \nabla_s \psi(s_0, u_0)\lambda_1 - \lambda_0 + \nabla_s \pi(s_0, u_0)\mu_0 && = 0 \\
& \nabla_u l(s_0, u_0) + \nabla_u \psi(s_0, u_0)\lambda_1 + \nabla_u \pi(s_0, u_0)\mu_0 && = 0 \\
& \mu_0 - \nu_0 && = 0 \\
& \pi(s_0, u_0) + t_0 && = 0 \\
& -t_0 && \leq 0 \\
& -\nu_0 && \leq 0 \\
& T_0 \nu_0 && = 0 \\
& \cdots && \\
& x_M - \psi(s_{M-1}, u_{M-1}) && = 0 \\
& \nabla_s \hat{l}(s_M, u_M) + \nabla_s \psi(s_M, u_M)\lambda_{M+1} - \lambda_M + \nabla_s \pi(s_M, u_M)\mu_M && = 0 \\
& \nabla_u \hat{l}(s_M, u_M) + \nabla_u \psi(s_M, u_M)\lambda_{M+1} + \nabla_u \pi(s_M, u_M)\mu_M && = 0 \\
& \pi(s_M, u_M) + t_M && = 0 \\
& T_M \mu_M - \tau \mathbf{1} && = 0 \\
& \cdots && \\
& s_N - \psi(s_{N-1}, u_{N-1}) && = 0 \\
& \nabla_s \hat{m}(s_N) - \lambda_N + \nabla_s \pi(s_N)\mu_N && = 0 \\
& \pi(s_N) + t_N && = 0 \\
& T_N \mu_N - \tau \mathbf{1} && = 0,
\end{aligned}
\tag{5.41}
$$

where we have introduced explicitly the equalities $\pi(s_i, u_i) + t_i = 0$, for $i = 0, \ldots, M-1$ with associated multipliers μ_i. Following [Robinson, 1980, Section

4], we can reformulate the necessary optimality conditions into the following generalized equation:

$$0 \in F_p(z) + \mathcal{N}_K(z) + Cx, \tag{5.42}$$

where $z := (\lambda, s, u, \mu, \nu, t)$, the vector valued function in the generalized equation is defined as $F_p(z) :=$

$$\begin{bmatrix} -s_0 \\ \nabla_s l(s_0, u_0) + \nabla_s \psi(s_0, u_0)\lambda_1 - \lambda_0 + \nabla_s \pi(s_0, u_0)\mu_0 \\ \nabla_u l(s_0, u_0) + \nabla_u \psi(s_0, u_0)\lambda_1 + \nabla_u \pi(s_0, u_0)\mu_0 \\ \mu_0 - \nu_0 \\ \pi(s_0, u_0) + t_0 \\ -t_0 \\ \dots \\ s_M - \psi(s_{M-1}, u_{M-1}) \\ \nabla_s \hat{l}(s_M, u_M) + \nabla_s \psi(s_M, u_M)\lambda_{M+1} - \lambda_M + \nabla_s \pi(s_M, u_M)\mu_M \\ \nabla_u \hat{l}(s_M, u_M) + \nabla_u \psi(s_M, u_M)\lambda_{M+1} + \nabla_u \pi(s_M, u_M)\mu_M \\ \pi(s_M, u_M) + t_M \\ T_M \mu_M - \tau \mathbf{1} \\ \dots \\ s_N - \psi(s_{N-1}, u_{N-1}) \\ \nabla_s \hat{m}(s_N) - \lambda_N + \nabla_s \pi(s_N)\mu_N \\ \pi(s_N) + t_N \\ T_N \mu_N - \tau \mathbf{1} \end{bmatrix} \tag{5.43}$$

and $\mathcal{N}_K(z)$ is the normal cone to the set

$$K := \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_g} \times \mathbb{R}_+^{n_g} \times \cdots \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_{g_N}} \times \mathbb{R}_+^{n_{g_N}}. \tag{5.44}$$

Finally, the parameter x enters the generalized equation linearly through $C := [I \ 0]^\top$. In order to compute an iterate of the generalized Newton's method applied to (5.43), we need to solve the following linearized generalized equation, with unknown z_+ and linearization point $\hat{z}$:

$$0 \in F_p(\hat{z}) + \nabla_z F_p(\hat{z})^\top (z_+ - \hat{z}) + \mathcal{N}_K(z_+) + Cx. \tag{5.45}$$

The following theorem provides a convergence result for the iterates generated by the generalized Newton iterations associated with the linearized generalized equation (5.45).

Theorem 5.2.5 (Convergence of partial tightening Newton iterations)**.** *Let* $\bar{z}(x)$ *be a strongly stable solution to* (5.42) *for a given parameter value* x*. Then, there exists a non-empty neighborhood* Z *of* $\bar{z}(x)$ *and positive constants* $0 \leq \kappa < 1$ *and* $\omega \geq 0$*, such that, for any linearization point* $\hat{z} \in Z$*, the generalized Newton iterates associated with the solution of the linearized generalized equation* (5.45) *satisfy the following contraction estimate:*

$$\|z_+ - \bar{z}(x)\| \leq \kappa \|\hat{z} - \bar{z}(x)\| + \frac{\omega}{2} \|\hat{z} - \bar{z}(x)\|^2. \tag{5.46}$$

Proof. The result is a direct consequence of the assumption of strong regularity at $\bar{z}(x)$ following the proof of Lemma 2.1.36. □

A note on strong regularity

The convergence result stated in Theorem 5.2.5, relies on the assumption that the generalized equation (5.42) used to define the partial tightening iterations is strongly regular at a solution. Since (5.42) cannot be trivially interpreted as the generalized equation associated with the first-order optimality conditions of *some* optimization problem, we cannot directly apply the strong sufficient second order conditions from [Robinson, 1980, Theorem 4.1]. Alternatively, one could resort to the reduction technique in [Robinson, 1980, Theorem 3.1], which applies to generic generalized equations. However, we can exploit a simpler argument, namely that, exploiting the Riccati-like elimination strategy described above, the linearized generalized equation associated with (5.42), can be easily reduced into an auxiliary problem which corresponds to the first-order optimality conditions of the reduced QP in (5.33). Hence, strong regularity of (5.42) at a given solution would be implied by the fact that strong second-order conditions hold for the reduced QP.

5.2.4 Asymptotic stability of partial tightening real-time iterations

It is possible to show asymptotic stability of the combined system-optimizer dynamics obtained when controlling a system in closed-loop with the proposed partial tightening real-time iteration strategy. To this end, it suffices to rely on the result from Theorem 5.1.8, together with the contraction result in Theorem 5.2.5 and some additional assumptions required to fit the setup to the one of Theorem 3.2.28. In particular, using the gradient recentered formulation proposed in [Wills and Heath, 2004], it is easy to show that Assumptions 5.1.3, 5.1.4, 5.1.5, 5.1.6 and 5.1.7 hold for the partial tightening NMPC formulation

5.17 such that the optimal value function $V(x)$ associated with it, is a valid Lyapunov function. At this point, making the assumptions necessary for Theorem 3.2.28, we can state the following result.

Theorem 5.2.6. *Let the Assumptions of Theorems 5.1.8, 5.2.5 and 3.2.28 hold. Then, there exists a ("sufficiently" short) sampling time* $T > 0$, *such that the origin* $(x, z) = (0, 0)$ *is a locally asymptotically stable equilibrium for the combined system-optimizer dynamics obtained by controlling a system in receding horizon with the feedback policy obtained with the proposed partial tightening real-time iteration strategy:*

$$\begin{aligned} x_+ &= \psi(T; x, M_{u,z}z), \\ z_+ &= \varphi(\psi(T; x, M_{u,z}z), z), \end{aligned} \tag{5.47}$$

where $\varphi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ *denotes the solution map of the linearized generalized equation* (5.45)*.*

Proof. The proof follows the same arguments as the one of Theorem 3.2.28. □

5.2.5 Illustrative example and benchmarking

In order to illustrate the proposed partial tightening approach, we regard the problem of controlling a simple system of the form

$$\begin{aligned} \dot{x}_1 &= 0.5 \cdot \sin(x_2) - x_2 \\ \dot{x}_2 &= 0.1 \cdot x_2 + x_1 + u, \end{aligned} \tag{5.48}$$

subject to the input constraint $-1 \leq u \leq 1$. We discretize this ODE with the explicit Runge-Kutta method of order 4 and formulate an OCP with $N = 30$ shooting nodes. A quadratic cost with $Q = Q_N = 10 \cdot \mathbb{I}_2$ and $R = 1$ is used. Figures 5.3 and 5.4 show the open-loop trajectories obtained for different values of M and for $\tau = 0.1$ and $\tau = 1.0$, respectively. For $\tau = 10.0$, Figure 5.5 shows instead the resulting closed-loop trajectories obtained when using the partial tightening real-time strategy with different numbers of iterations.

Finally, the proposed strategy has been implemented within `acados` using the open-source interior-point solver `HPMPC` [Frison et al., 2014] that exploits high-performance linear algebra package `BLASFEO` [Frison et al., 2017]. In order to validate this implementation and highlight the computational benefits of the proposed partial tightening strategy, we will consider the problem of controlling

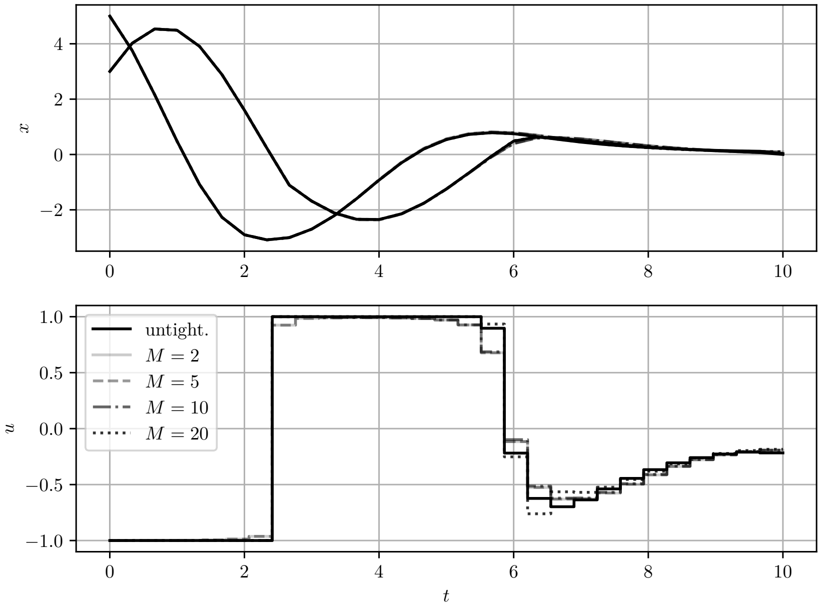


Figure 5.3: Converged open-loop trajectories for $\tau = 0.1$.

the following ordinary differential equations describing the dynamics of an inverted pendulum:

$$\begin{aligned}
\dot{p} &= v \\
\dot{v} &= \frac{-l_\mathrm{p} m_\mathrm{p} \sin(\theta)\omega^2 + F + g m_\mathrm{p} \cos(\theta)\sin(\theta)}{M_\mathrm{p} + m_\mathrm{p} - m_\mathrm{p}\cos(\theta)^2} \\
\dot{\theta} &= \omega \\
\dot{\omega} &= \frac{-l_\mathrm{p} m_\mathrm{p} \cos(\theta)\sin(\theta)\omega^2 + F\cos(\theta) + g m_\mathrm{p}\sin(\theta) + M_\mathrm{p} g \sin(\theta)}{l_\mathrm{p}(M_\mathrm{p} + m_\mathrm{p} - m_\mathrm{p}\cos(\theta)^2)},
\end{aligned} \tag{5.49}$$

where $x = (p, \theta, v, \omega)$ is the state of the system, where p and v are the linear position and velocity of the cart and θ and ω are the angle and angular velocity of the pendulum. The input to the system is the force F applied to the cart, while m_p, M_p, l_p and g are fixed parameters representing the mass of the pendulum, the mass of the cart, the length of the pendulum and gravitational acceleration respectively.

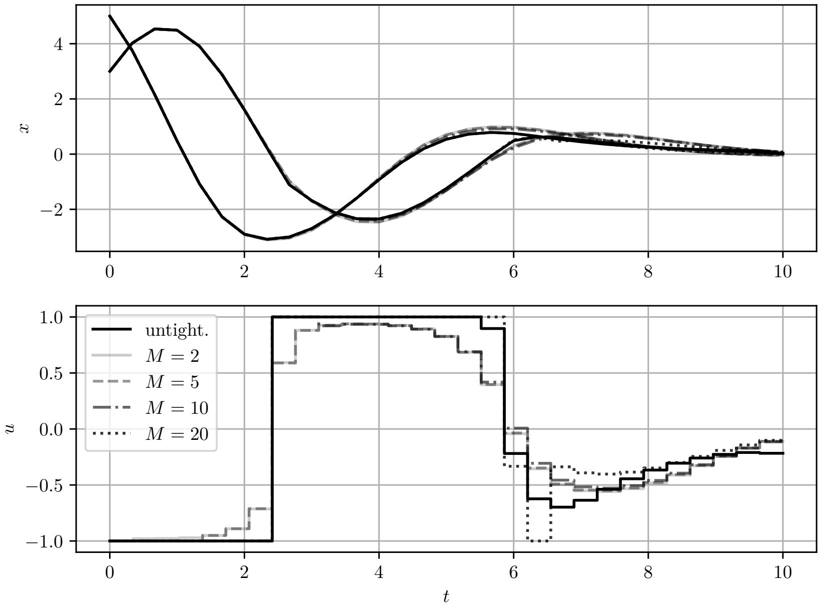


Figure 5.4: Converged open-loop trajectories for $\tau = 1.0$.

An OCP of the form in (5.13) is considered, where $\psi(\cdot)$ represents the discretized dynamics obtained by applying the explicit Runge-Kutta strategy of order four with fixed step-size $h = 0.01s$. A control horizon $T = 1s$ is used and the trajectories are discretized using $N = 100$ shooting nodes. Simple bounds are imposed on the input $F_{\text{max}} = -F_{\text{min}} = 12\,N$ and the cost matrices have been chosen as follows:

$$Q = \text{diag}(1 \cdot 10^{-1},\, 1,\, 1 \cdot 10^{-1},\, 2 \cdot 10^{-3}) \;\text{ and }\; R = 5 \cdot 10^{-4},$$

and an LQR-based terminal cost is used.

Figure 5.6 shows the closed-loop trajectories obtained with three different setups. An RTI strategy with standard formulation is used with $N = 50$ and $N = 100$ and the proposed partial tightening formulation is used with $M = 20$, $N = 100$ and $\tau = 1$. The partial tightening strategy stabilizes the system, keeping the original form of the constraints only in few stages. In this way, a long prediction horizon can be used while reducing the computational burden associated with the iterations with respect to the standard RTI strategy. If a shorter horizon of e.g. $N = 50$ shooting nodes is used, the system cannot be stabilized. Timing results are reported in Table 5.1 for the inverted pendulum swing-up example,

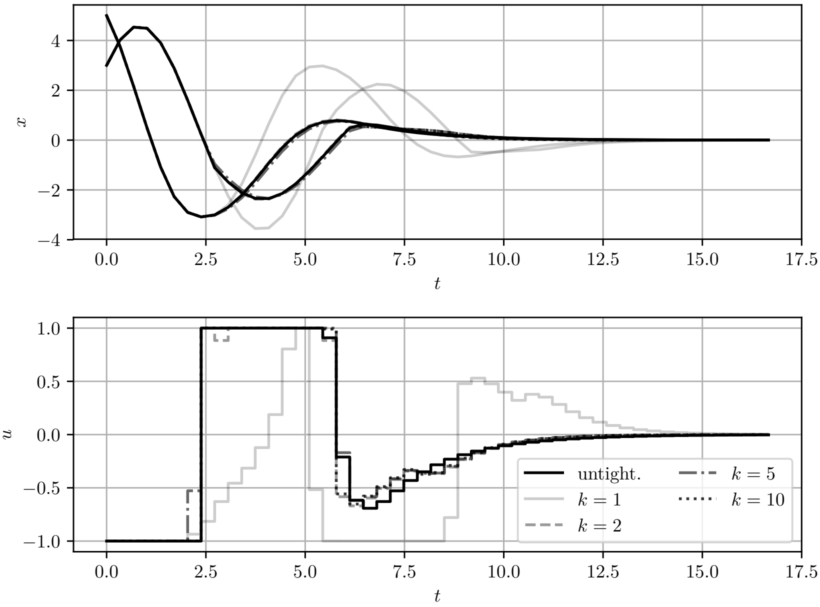


Figure 5.5: Closed-loop trajectories obtained with the proposed partial tightening real-time iteration strategy for different number of iterations k per sampling time and $\tau = 10.0$.

using a varying number of untightened stages M and $\tau = 1$. For small values of M, a large speedup can be achieved with respect to the standard formulation with $N = 100$.

5.2.6 Conclusions

An efficient partial tightening RTI strategy for NMPC has been presented. The algorithm uses a barrier formulation to approximate stage constraints in the terminal part of the prediction horizon. In this way a large part of the variables in the QP subproblems can be eliminated with a single backward Riccati recursion sweep. After solving a reduced QP for the initial part of the horizon, the solution is expanded back to the full horizon. We proved convergence of the partial tightening generalized Newton iterations and asymptotic stability of the system-optimizer dynamics. A numerical case study is presented that shows both closed-loop simulations and detailed timing results.

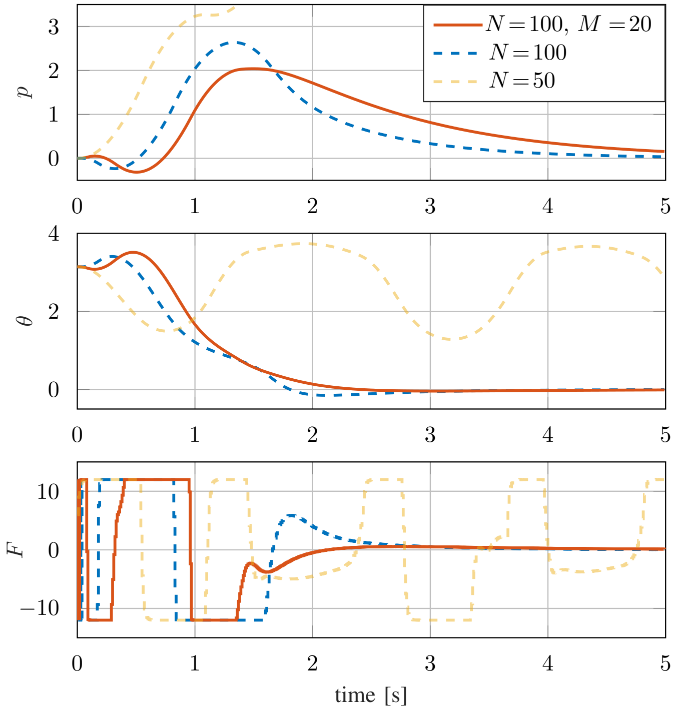


Figure 5.6: Inverted pendulum swing-up: comparison of RTI strategies. Partial tightening RTI strategy (bold solid red) with $N = 100$, $M = 20$ and $\tau = 1$, standard formulation with $N = 100$ (dashed blue) and with $N = 50$ (dashed yellow). The partial tightening formulation stabilizes the system keeping the original constraints only in the initial 20 stages. The same does not hold for a standard formulation with a shortened horizon of $N = 50$ nodes.

5.3 Progressive tightening NMPC for attitude control of a quadcopter

This section discusses the design, implementation and deployment of an attitude controller for a quadrotor based on partial tightening NMPC on a low-power embedded system equipped with a Cortex-A9 CPU running at 800 MHz. Simulation results that show the improvement in performance obtained by using NMPC over standard control techniques are discussed and experimental results using the proposed implementation are presented.

	$N=100$	$M=50$	$M=20$	$M=10$	$M=5$	$M=2$
QP [ms]	4.405	2.571	1.034	0.566	0.313	**0.237**
lin. [ms]	0.190	0.190	0.190	0.190	0.190	0.190
total [ms]	4.595	2.761	1.224	0.756	0.503	**0.427**
i.p. iter.	37	32	29	26	17	17
c.l. cost	1037	1005	1160	1128	1338	1288
$\text{speedup}_{\text{QP}}$	–	1.71	4.26	7.78	14.07	**18.59**
$\text{speedup}_{\text{RTI}}$	–	1.66	3.75	6.07	9.13	**10.76**

Table 5.1: Pendulum example: worst-case computation times for the swing-up closed-loop scenario (1000 sampling steps) in milliseconds and closed-loop cost with $N = 100$, $\tau = 1$ and decreasing values of M (standard RTI with $N = 100$ in the first column).

Background on unmanned aerial vehicles

Unmanned aerial vehicles (UAVs) are finding their way into several application fields such as inspection, surveillance and rescuing and are drawing considerable interest in the control engineering community. Furthermore, a few applications have emerged in which quadrotor- and multirotor-like systems are used as personal air vehicles [EHa, 2014, Kit, 2010]. Due to the highly nonlinear dynamics exhibited by quadrotors, the performance of linearization-based controllers can be affected when the vehicles are operated far from the linearization point. Moreover, with classical control solutions, it is non-trivial to deal with constraints on states and inputs, when they are present. Although such approaches have been successfully applied to the problem of controlling the attitude of quadrotors (see [Bouabdallah et al., 2004], [Yang, 2012] to cite only a few applications), NMPC could in principle provide a more direct approach in treating nonlinearity and constraints. In [Kamel et al., 2015] an NMPC attitude controller for a multicopter that operates on the rotation group $\mathbf{SO}(3)$ using the RTI strategy [Diehl et al., 2002b] is proposed.

In this section, NMPC is used to design an attitude controller for a human-sized quadrotor equipped with a low-power embedded processor running at 800 MHz. The designed controller is required to be able to compute the control action within 10 ms, while sparing enough CPU time for the other routines running on the embedded platform to be performed (e.g telemetry, logging, low-level controller, sensing and estimation, fault detection, etc). In order to meet the required execution time, partial tightening NMPC introduced in Section 5.2 is used. The OCP employed utilizes dynamics in quaternion form and a nonlinear least-squares cost that enables direct tracking of references in the Euler angles space.

The main purpose of the section is, rather than building on top of state-of-the-art control techniques for UAVs, to show that recent advances in algorithms and software implementations enable one to reduce the computational burden associated with optimization-based control techniques. In particular, applications can be tackled where short sampling times need to be met on resource constrained hardware.

5.3.1 Quadcopter model and optimal control formulation

In order to control the quadcopter's attitude we will use the tracking NMPC formulation with least-squares cost terms

$$\begin{aligned}
\min_{\substack{s_0,\dots,s_N \\ u_0,\dots,u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(s_i,u_i)\|_W^2 + \frac{1}{2}\|\eta_N(s_N)\|_{W_N}^2 \\
\text{s.t.} \quad & s_0 - x = 0, \\
& s_{i+1} = \psi(s_i,u_i), \quad i = 0,\dots,N-1, \\
& \pi(s_i,u_i) \leq 0, \qquad i = 0,\dots,N-1, \\
& \pi_N(s_N) \leq 0,
\end{aligned} \tag{5.50}$$

where $x_i \in \mathbb{R}^{n_x}$ and $u_i \in \mathbb{R}^{n_u}$ are the states and inputs of the system respectively and ψ, π, π_N, η and η_N are twice continuously differentiable functions. The nonlinear residual functions η and η_N are weighted by W, $W_N \in \mathbb{S}_+$, respectively and the initial state of the system is denoted by x.

The dynamics of the quadcopter to be controlled are described by the following model [Betsch and Siebert, 2009]:

$$\dot{q} = \frac{1}{2}S^\top \Omega, \quad \dot{\Omega} = J^{-1}(T - \Omega \times J\Omega), \tag{5.51}$$

where q and Ω describe the orientation of the quadrotor expressed in quaternion representation and its angular velocity, respectively, and with

$$S := \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}. \tag{5.52}$$

Parameter	Value	Description
ρ	$1.225\,\mathrm{kg/m^3}$	air density
A	$0.1\,\mathrm{m^2}$	propeller area
C_l	0.125	lift coefficient
C_d	0.075	drag coefficient
m	$10\,\mathrm{kg}$	quadrotor mass
g	$9.81\,\mathrm{m/s^2}$	gravitational acceleration
$J_1 = J_2 = 4 \cdot J_3$	$0.25\,\mathrm{kg \cdot m/s^2}$	moments of inertia

Table 5.2: Quadrotor model - values of the parameter used for the simulation results in Section 5.3.2. Notice that these values are fictitious. They have been used for simulation purpose and do not correspond to the parameters of the physical system.

It is assumed that angular velocities of the propellers ω can be tracked instantaneously, hence they are considered as inputs to the system. Moreover, the angular momentum contribution of the propellers is ignored in order to simplify the model. The matrix J denotes the inertia matrix of the vehicle, while the torques applied to the system are described by $T := [T_1\ T_2\ T_3]^\top$, with

$$T_1 := \frac{AC_\mathrm{l}\rho(\omega_2^2 - \omega_4^2)}{2}, \quad T_2 := \frac{AC_\mathrm{l}\rho(\omega_1^2 - \omega_3^2)}{2},$$

$$T_3 := \frac{AC_\mathrm{d}\rho(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{2},$$

where ρ is the air density, C_d and C_l are the drag and lift coefficients and A is the area of the propellers. The values of the parameters appearing in the model are listed in Table 5.2. In the next section, numerical simulations will be presented that compare the computation times and the closed loop performance of different attitude controllers among which the partial tightening and the standard RTI strategy.

5.3.2 Simulation results

In this section, the performance of different control strategies will be assessed in simulation. In particular, a proportional-derivative controller (PD), a linear-quadratic regulator (LQR) and different variants of an NMPC-based strategy will be taken into account.

PD controller

The first controller taken into account is a PD acting separately on the three Euler coordinates and using a fixed torque allocation as described in [Bouabdallah et al., 2004]. The main idea consists in defining the torque applied to the vehicle using an a priori fixed parametrization that relies on the observation that torques along the three axis can be obtained, loosely speaking, by adjusting the propeller speeds in a “differential” fashion:

$$\begin{aligned} \tau_1 &= \frac{AC_\mathrm{l}\rho}{2J_1}(\omega_2^2-\omega_4^2), \quad \tau_2 = \frac{AC_\mathrm{l}\rho}{2J_2}(\omega_3^2-\omega_1^2), \\ \tau_3 &= \frac{AC_\mathrm{d}\rho}{2J_3}(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2). \end{aligned} \tag{5.53}$$

Additionally, the equation $F_r = (\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)\frac{AC_\mathrm{l}\rho}{2}$ is used to specify the desired total thrust.

Remark: notice that the torque parametrization used completely neglects the inertial terms in the dynamics which depend on the angular velocities of the vehicle. However, for small angular velocities, it can be expected to provide a reasonable approximation of the actual torques.

The chosen approximate parametrization allows one to design three fully decoupled PD controllers that control the attitude on a separate axis each:

$$\tau_i = K_\mathrm{p} e_i + K_\mathrm{d}\dot{e}_i, \quad \text{for,} \quad i \in \{1,2,3\}, \tag{5.54}$$

where e_i and $\dot{e}_i$ are estimates of angle errors and their derivatives, respectively, and K_p and K_d are the proportional and derivative diagonal matrix gains. Once the desired torque vector τ has been computed according to (5.54), the squared rotor speeds ω_s can be efficiently computed by solving the linear system

$$\frac{A\rho}{2}M\omega_s = t, \tag{5.55}$$

where

$$M := \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \; t := \begin{bmatrix} \frac{\tau_1 J_1}{C_\mathrm{l}} & \frac{\tau_2 J_2}{C_\mathrm{l}} & \frac{\tau_3 J_3}{C_\mathrm{d}} & \frac{gm}{C_\mathrm{l}} \end{bmatrix}^\top.$$

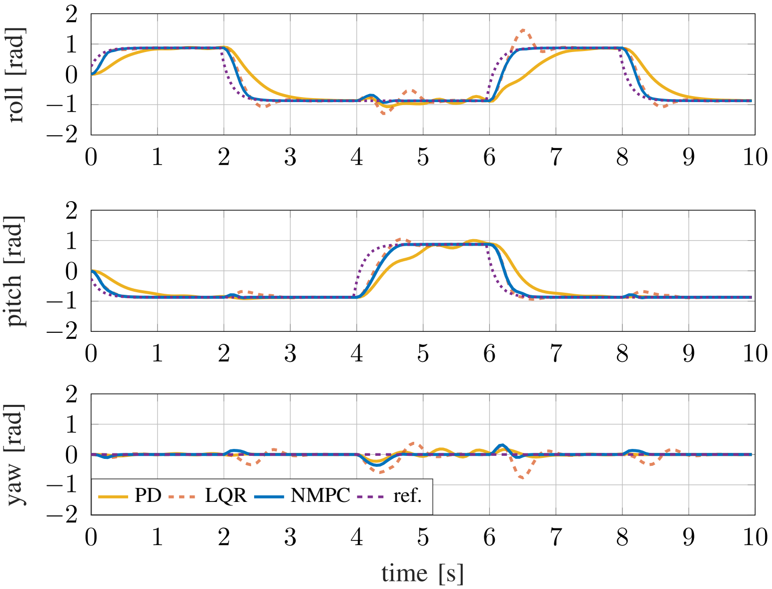


Figure 5.7: Attitude tracking simulation results comparing different control strategies - closed-loop state trajectories: PD in solid yellow, LQR in dashed red and converged-NMPC in solid blue. The NMPC controller achieves a more accurate tracking of the reference attitude.

	max CPU time [ms]	avg CPU time [ms]	subopt. [%]
`Ipopt`	131.40	43.40	-
RTI	1.04	0.52	3.69
pt-RTI	0.22	0.18	17.67

Table 5.3: Maximum and average CPU time and relative closed-loop suboptimality with respect to converged NMPC of the RTI and partial tightening RTI (pt-RTI) strategies. Using the pt-RTI strategy a speedup of about a factor 5 can be achieved with a moderate increase in suboptimality.

Notice that the matrix $\frac{A\rho}{2}M$ is constant and it can be pre-factorized offline in order to speed up the solution of the linear system. The actual rotor speeds can be finally obtained by computing the element-wise square root of ω_s: $\omega_i = \sqrt{\omega_{s,i}}, \quad i = 1, \dots, 4$.

LQR controller

The second controller that will be taken into account is based on a reduced space LQR. Since the attitude dynamics in quaternion coordinates are not controllable, the dynamics will be first projected onto a controllable subspace as proposed in [Yang, 2012]. In particular, using the fact that $q_0 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2}$, the first component of the quaternion vector can be eliminated yielding a differential equation which, together with the angular velocity dynamics in (5.51), will be used to design the LQR static gain. To this end, the dynamics are linearized around the hovering steady state and input ($\bar{x}$, $\bar{u}$) and discretized using an explicit RK4 integration strategy:

$$x_{k+1} - \bar{x} = A(x_k - \bar{x}) + B(u_k - \bar{u}). \tag{5.56}$$

At this point, a discrete-time LQR controller can be designed by solving the discrete time algebraic Riccati equation

$$A^\top PA - P - (A^\top PB)(B^\top PB + R)^{-1}(B^\top PA) + Q = 0,$$

which provides a static state feedback $u = Kx + \bar{u}$ with

$$K = (B^\top PB + R)^{-1}(B^\top PA), \tag{5.57}$$

which, once an estimate of the state of the system has been computed, allows one to readily compute the input to be applied to the system.

NMPC controller

Three different variants of NMPC-based controllers will be compared in simulation. All of them use a nonlinear least-squares formulation with residual functions

$$\eta(x,u) := \begin{bmatrix} \alpha(x) - \alpha_r \\ \beta(x) - \beta_r \\ \gamma(x) - \gamma_r \\ x - x_r \\ u - u_r \end{bmatrix}, \quad \eta_N(x) := \begin{bmatrix} \alpha(x) - \alpha_r \\ \beta(x) - \beta_r \\ \gamma(x) - \gamma_r \\ x - x_r \end{bmatrix}, \tag{5.58}$$

where α, β and γ define the attitude of the quadrotor in Euler angles (roll, pitch and yaw) as functions of q. The quantities in (5.58) with the r subscript denote the desired references associated with each residual output.

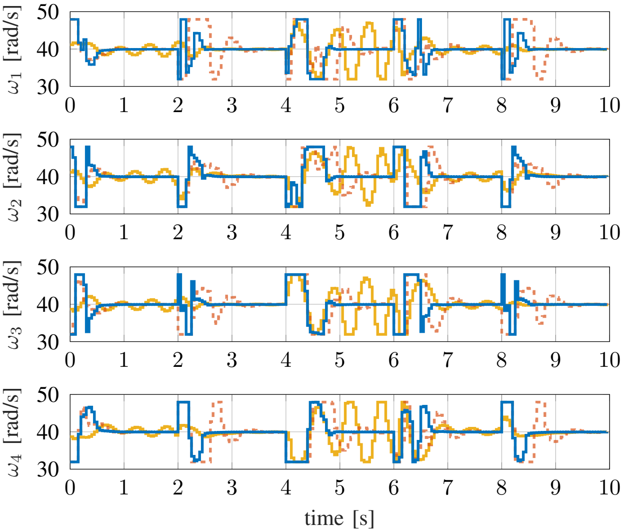


Figure 5.8: Attitude tracking simulation results comparing different control strategies - closed-loop input trajectories. PD in solid yellow, LQR in dashed red and converged-NMPC in solid blue.

Converged NMPC strategy

The problem formulation in (5.50) has been implemented using `CasADi` [Andersson et al., 2019] with a prediction horizon of $T = 1.0\,\mathrm{s}$ and $N = 20$ shooting nodes, discretizing the dynamics using the explicit RK4 integration strategy. The obtained OCPs are solved using the interior-point solver `Ipopt` [Wächter and Biegler, 2006]. The `Ipopt` interface available in `CasADi` is used where the possibility of *just-in-time* compiling function evaluations is exploited in order to speed up the computations. The linear system solver `ma57` [Duff, 2004] is used linked against a single threaded build of the high-performance BLAS implementation `OpenBLAS` [OpenBLAS, 2011]. Although real-time implementations of nonlinear interior-point methods are present in the literature [Zanelli et al., 2017a], the computation times obtained when solving the OCPs to a local minimum are often longer than the ones obtained with approximate strategies like the RTI. However, the closed loop trajectories obtained with this approach will be used as a reference to assess the suboptimality associated with the approximate strategies described below.

Standard RTI strategy

In order to reduce the computation times associated with the solution of the OCPs, the RTI strategy has been implemented using the software package `acados` [Verschueren et al., 2021]. The same number of shooting nodes and the same tuning has been used for the implementation. The chosen QP solver is `HPMPC` [Frison et al., 2014] which relies on the hardware-tailored linear algebra package `BLASFEO` [Frison et al., 2017].

Partial tightening RTI strategy

Finally, the partial tightening RTI strategy (pt-RTI) has been implemented in `acados` using an untightened horizon of $M = 2$ stages and an overall horizon of $N = 20$. The same tuning used for the previous two strategies is used and a fixed barrier value $\tau = 10$ is used. As for the standard RTI strategy, the solver `HPMPC` will be used to solve the reduced QPs as well as to perform the Riccati-based elimination for the terminal section of the horizon described in [Zanelli et al., 2017b].

The code in the following simulations is set up to use the ANSI C implementation `BLASFEO RF` [Frison et al., 2018] in order to better resemble the CPU load distribution between different routines (e.g. linearization and QP solution) expected on the embedded hardware.

Comparison

The controllers described above are used in the following to track a periodic attitude reference. For the NMPC formulations the following weights are chosen:

$$\begin{aligned} W &= \mathrm{blkdiag}(5 \cdot 10^2 \cdot \mathbb{I}_3,\ 1 \cdot 10^{-3} \cdot \mathbb{I}_{11}) \\ W_N &= \mathrm{blkdiag}(5 \cdot 10^2 \cdot \mathbb{I}_3,\ 1 \cdot 10^{-3} \cdot \mathbb{I}_7). \end{aligned}$$

In order to tune the PD controller, the parametrization $K_\mathrm{p} = \kappa_\mathrm{p} \cdot \mathbf{I}_3$ and $K_\mathrm{d} = \kappa_\mathrm{d} \cdot \mathbf{I}_3$, with $\kappa_\mathrm{p} \in [1,\ 60]$ and $\kappa_\mathrm{d} \in [1,\ 20]$ has been chosen. After discretizing each parameter interval into 100 equidistant values, a simulation has been run for each combination of values $(\kappa_\mathrm{p}, \kappa_\mathrm{d})$ and the squared deviation from the reference trajectories in the Euler space has been taken into account as a performance metric. The values $\kappa_\mathrm{p} = 23$ and $\kappa_\mathrm{d} = 9$ have been chosen, which provide a reasonable trade-off between deviation from the reference trajectory and chattering of the input trajectories. The LQR controller could be in principle tuned by exploiting a linearization of the transformation from Euler to the controllable quaternion subspace. In this way, the control policy obtained would be locally equivalent to the NMPC one. However, possibly due to nonlinearity and to the presence of constraints, the simulations showed the necessity to detune the controller in order to achieve acceptable performance. To this end, the LQR weighting matrices have been chosen as follows:

$$Q = \text{blkdiag}(1 \cdot 10^{2} \cdot \mathbb{I}_3,\, 1 \cdot 10^{-3} \cdot \mathbb{I}_3),\; R = 1 \cdot 10^{-3} \cdot \mathbb{I}_4.$$

A simulation with sampling time $T_s = 0.05\,\mathrm{s}$ is performed where the input bounds are imposed on the propeller velocities: $\omega_{\mathrm{ss}} - \Delta\omega_{\max} \leq \omega_i \leq \omega_{\mathrm{ss}} + \Delta\omega_{\max}$, with $\Delta\omega_{\max} := 8\,\mathrm{rad\,s^{-1}}$ and where $\omega_{\mathrm{ss}} := 39.939\,\mathrm{rad\,s^{-1}}$ denotes the steady-state input associated with a mass of $10\,\mathrm{kg}$ (although the controller only regulates the vehicle's attitude, it is meant to be used in a cascaded architecture, where also position is controlled).

The closed-loop trajectories obtained with PD, LQR and NMPC controllers are reported in Figure 5.7 and 5.8. Only trajectories obtained with `Ipopt` are shown for the sake of clarity, as the results obtained with the RTI and pt-RTI strategies do not differ much from the ones obtained using converged NMPC. For the two approximate strategies, computation times and closed-loop suboptimality are reported in Table 5.3. From Figure 5.7 and 5.8, it can be seen that the converged NMPC controller performs better than the other two control strategies in the sense that smaller overshoots and faster response to references changes can be achieved. This might be due to the fact that nonlinearity and the presence of constraints can degrade the performance of the PD and LQR controllers for large reference changes like the ones used in the benchmark.

5.3.3 Experimental results

The controller based on the pt-RTI strategy has been deployed to the on-board embedded hardware of the quadrotor which features a Xilinx Zynq system-on-chip with dual-core Cortex-A9 clocked at 800 MHz. Notice that, although the instruction set available on such a CPU provides vectorized instructions, they are only available in single precision. Hence, the `GENERIC` implementation of the `BLASFEO` package has been used, which exploits a panel-major format, but does not make explicit usage of vectorized instructions.

For the embedded implementation, a horizon $T = 1.0\,\mathrm{s}$ is used with $N = 10$ shooting nodes and untightened horizon $M = 2$. In order to achieve faster response to changes in the reference and disturbance rejection, and to improve the convergence of the pt-RTI strategy, the controller is run at a sampling $T_s = 10\,\mathrm{ms}$. Notice that, although to the knowledge of the authors a formal

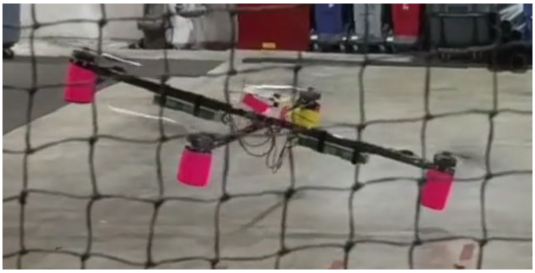


Figure 5.9: Human-sized quadrotor equipped with a low-power Xilinx Zynq SoC with a dual-core ARM Cortex-A9 running at 800 MHz: snapshot from the experiment video (`https://www.youtube.com/watch?v=-dsezQa7nzk&feature=youtu.be`).

stability proof for this setup does not exist, the “over-sampled” implementation of NMPC strategies is rather common among practitioners.

Similarly to what obtained in simulations, using the pt-RTI strategy with $M = 2$ gives rise to a considerable speedup reducing the average computation times from about 6 ms for the standard RTI to about 2 ms. In this way, enough computational time can be spared to carry out other tasks such as telemetry, logging and executing lower level controllers without approaching high CPU loads which might lead to faults.

Figures 5.10 and 5.11 show the attitude and actuators trajectories obtained during a test flight. Notice that, since no position control has been implemented for the test, the attitude reference during the test flight is provided by a human pilot who is making sure that the vehicle is hovering safely above the ground.

5.3.4 Conclusions

In this section, the design, implementation and deployment of an NMPC attitude controller for a human-sized quadrotor has been presented. The partial tightening method proposed in Section 5.2 is used in order to reduce the computation times. Simulation results are discussed which show that considerable speedups can be achieved with a moderate increase in suboptimality with respect to standard approaches. The NMPC controller is deployed to the on-board computer of the vehicle and flight test results are reported and discussed.

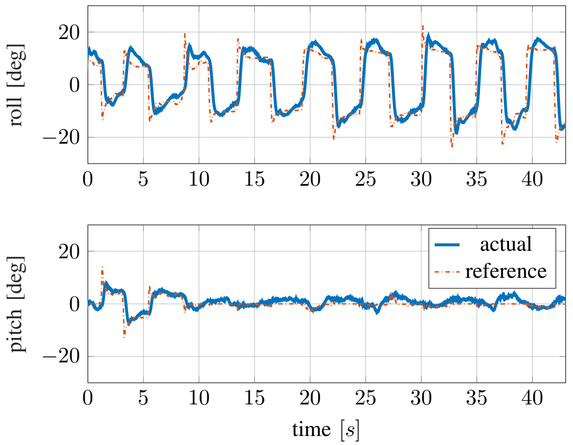


Figure 5.10: Experimental results - attitude in Euler angles.

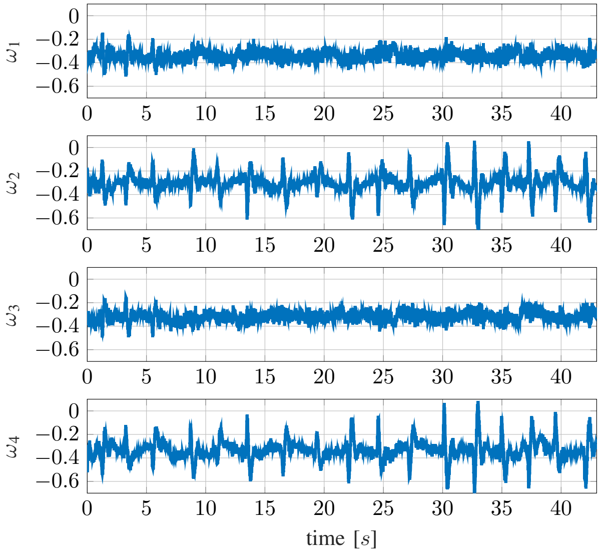


Figure 5.11: Experimental results - actuators.

5.4 Chapter summary and outlook

In this chapter, we introduced a class of progressive tightening NMPC formulations that rely on stage-varying costs and constraints that become

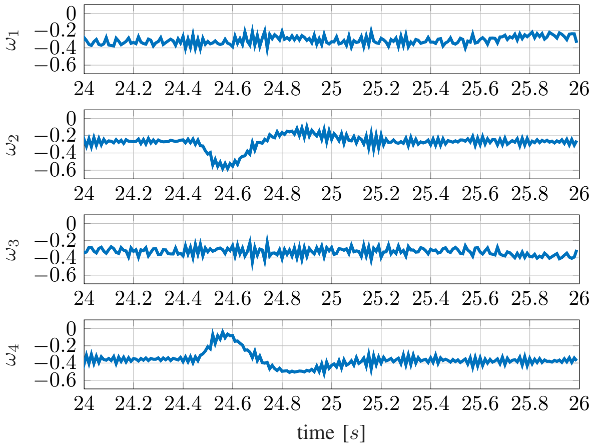


Figure 5.12: Experimental results - actuators (zoom in).

increasingly “tight” as the stage index increases. This idea encodes the fact that increasingly conservative assessments are made as we predict farther into the future. We showed that standard arguments can be adapted in order to construct a Lyapunov function for NMPC formulations that satisfy the cost-constraint epigraph inclusion assumption that formalizes the concept of progressive tightening.

In the second part of the chapter, we introduced a partial tightening RTI strategy that exploits an underlying progressive tightening formulation based on barrier functions in order to speed up the computations. In this case, the main idea is to split the prediction horizon in two sections and exploit a smooth, and “tightened”, approximation of the feasible sets for the terminal section. In this way, the numerical difficulties associated with inequalities are mitigated and, in particular, it is possible to use an efficient Riccati-like algebraic elimination to reduce the horizon of the underlying QP subproblems. Moreover, although a suboptimal solution is recovered, since the exact constraints are used in the initial section of the prediction horizon, the full feasible input set can be exploited by the proposed strategy. This is in contrast with most barrier-based MPC strategies present in the literature. We showed that, under the assumption of strong regularity, the iterates generated by the proposed algorithm converge Q-linearly to a KKT point of the partial tightening OCP. In this way, the results from Chapter 3 can be applied in order to guarantee asymptotic stability of the system- optimizer dynamics. The partial tightening RTI strategy has been implemented in the software package `acados` and its numerical performance has been validated on a nontrivial example showing that considerable speedups can be achieved.

The asymptotic stability proof presented in Section 5.1 is rather general and it can cover in principle a large class NMPC formulations. At the same time, it can inspire the development of novel formulations and numerical methods such as the partial tightening RTI strategy. The revisitation of existing NMPC methods and their interpretation as progressive tightening NMPC is subject of undergoing research. On the numerical side, strategies that make use of barrier formulations with an increasing barrier parameter could offer numerical benefits and provide an interesting research direction. Finally, a user-friendly implementation of the partial tightening RTI method in `acados` is yet to be developed and could be relatively easily obtained using the building blocks provided by the QP solver `HPIPM` [Frison et al., 2020].

Chapter 6

Continuous control set NMPC of reluctance synchronous machines

In this chapter we describe the design and implementation of a current controller for a reluctance synchronous machine based on continuous set nonlinear model predictive control. A computationally efficient grey box model of the flux linkage map is employed in a tracking formulation which is implemented using the high-performance framework for nonlinear model predictive control `acados`. The resulting controller is validated in simulation and deployed on a `dSPACE` real-time system connected to a physical reluctance synchronous machine. Experimental results are presented where the proposed implementation can reach sampling times in the range typical for electrical drives and can achieve large improvements in terms of control performance with respect to state-of-the-art classical control strategies.

6.1 Nonlinear model predictive control for electrical drives

In recent years, reluctance synchronous machines (RSMs) have emerged as a competitive alternative to classical synchronous machines (SMs) with permanent magnet (PMSMs) or direct current excitation. In addition to the favourable properties of SMs in general, e.g., high efficiency, reliability and compact

design, RSMs are often easier to manufacture and comparably cheap due to the absence of magnets. Moreover, their anisotropic magnetic path in the rotor, makes them particularly suitable for saliency-based encoderless control [Landsmann et al., 2010a, Landsmann et al., 2010b].
However, a major drawback of the RSM concerning control is its characteristic nonlinearity of the flux linkage, caused by magnetic saturation and cross-coupling effects in the rotor. As a consequence, the machines' inductances vary significantly with the stator currents. Moreover, additional coupling between the stator d- and q-currents is imposed by the cross-coupling inductances and the coupling of the nonlinear back electro-motive force in the synchronous reference frame, which generally requires further measurements to be carried out online.
Regarding the control of RSMs, two main concepts have been pursued in the past: (i) Direct torque control (DTC) [Boldea et al., 1991, Lagerquist et al., 1994] and (ii) field-oriented control (FOC) [Matsuo and Lipo, 1993, Xu et al., 1991, Betz et al., 1993]. While DTC is known for its robustness and fast dynamics [Bolognani et al., 2011], it produces a high current distortion leading to torque ripples [Chikhi et al., 2010]. In contrast, vector control improves the torque response [Rashad et al., 2004] and the efficiency of the system [Kamper et al., 1996], but good knowledge of the system parameters is required for implementation. In [Hackl, 2015], a completely parameter-free adaptive PI controller is proposed which guarantees tracking with prescribed transient accuracy. The controller is applied to current control of (reluctance) synchronous machines, but measurement results are not provided.
In [Rashad et al., 2004] and [Yamamoto et al., 2009], the inductances are tracked online in order to adjust the current *references* thus achieving a higher control accuracy. In [Hackl et al., 2016], a FOC control scheme is proposed, where the PI control parameters are continuously adapted to the actual system state, which improves the overall current dynamics.

An alternative to classical control approaches is the use of optimization-based control techniques such as model predictive control (MPC). When using MPC, a parametric optimization problem is formulated that exploits a model of the plant to be controlled and enforces constraints while minimizing a certain objective function. Although MPC can in principle improve the control performance and ease the controller design [Geyer et al., 2009], meeting the required sampling times is in general a challenging task due to the high computational burden associated with the solution of the underlying optimization problems.

In order to circumvent this difficulty, several algorithmic strategies have been proposed over the past decade that use different approaches and (potentially) different formulations of the optimal control problems to be solved. Among the possible classifications of methods present in the literature, in the fields of electrical drives and power electronics, a fundamental distinction can be made

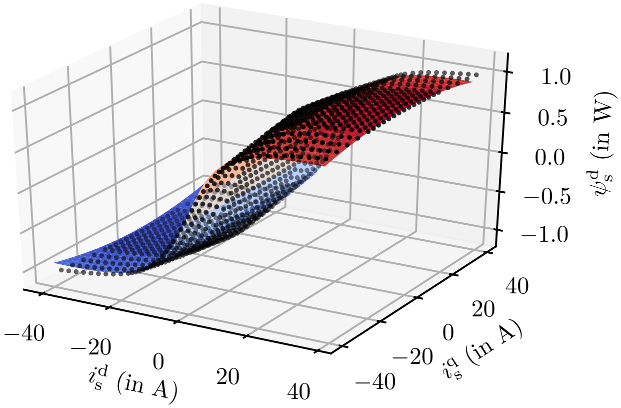


Figure 6.1: Nonlinear flux linkage of a real RSM obtained from FEM data $\hat{\Psi}_s^d$ (solid surface) and fitted grey box model Ψ_s^d (dotted) - d-component. The worst-case relative error amounts to less than 10%.

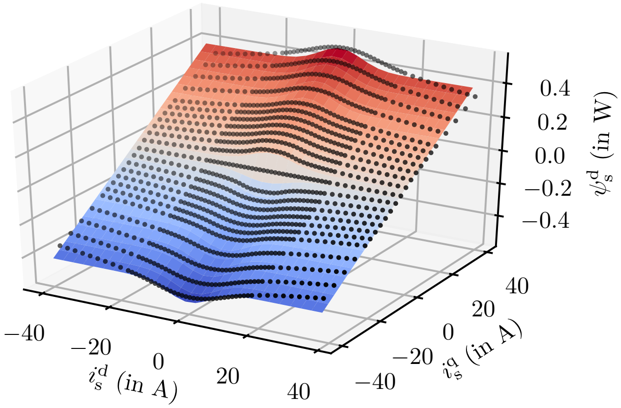


Figure 6.2: Nonlinear flux linkage of a real RSM obtained from FEM data $\hat{\Psi}_s^d$ (solid surface) and fitted grey box model Ψ_s^d (dotted) - q-component. The worst-case relative error amounts to less than 10%.

between what is sometimes referred to as *finite* (FS-) and *continuous control set* (CS-) MPC [Quevedo et al., 2019], [Cortes et al., 2008].

In FS-MPC, the switch positions of the power converter are regarded as optimization variables leading to mixed-integer programs. In this way, the need for an external modulator is eliminated and the switching sequences are directly determined by the solution to the optimal control problem (hence the name *“direct”* MPC used in some of the literature on MPC for electrical drives and power converters [Geyer, 2016]).

When using CS-MPC instead, we delegate the determination of switching sequences to an external modulator in order to obtain a continuous optimization problem. For this reason, CS-MPC is sometimes referred to as *“indirect”* MPC [Geyer, 2016]. Although the computation times associated with this latter approach scale favourably with prediction horizon length and number of control variables (typically complexity $\mathcal{O}\big(N(n_u + n_x)^3\big)$ can be achieved, where N, n_u and n_x represent horizon length, number of inputs and states, respectively), for short horizons, strategies based, e.g., on sphere decoding algorithms applied to FS-MPC formulations can achieve sufficiently short computation times. On the contrary, CS-MPC is generally regarded as more computationally expensive and it is still, arguably for this reason, largely unexplored [Geyer, 2016].

Among the experimental results in the literature obtained with CS-MPC, in [Besselmann et al., 2015] a DC-excited synchronous motor is controlled using the real-time iteration method. In [Englert and Graichen, 2018], a fixed-point iteration scheme is used to control a permanent magnet synchronous machine. Among applications leveraging linear-quadratic CS-MPC we mention the work in [Domahidi et al., 2012a] in which permanent magnet synchronous machines and induction machines are controlled using explicit model predictive control. Finally, in the recent work in [Cimini et al., 2020], an active-set algorithm is used to solve the convex QPs arising from a linear-quadratic CS-MPC formulation to control a PMSM.

6.1.1 Contribution

In this section, we describe the design and implementation details together with simulation and experimental results of a nonlinear CS-MPC controller (CS-NMPC) for an RSM. The contributions of the present work are:

- We describe the design and implementation details of a tracking CS-NMPC formulation that relies on the software package `acados`, which is capable of achieving timings in the microsecond time scale necessary to control the electrical drive.
- We propose the use of a simple grey box model for the flux maps of RSMs with a small number of parameters that can be used for online applications where computation times are of key importance.
- Finally, we present simulation and experimental results that confirm the validity of the proposed control formulation and its implementation and

its superior performance in comparison with state-of-the-art methods from the field of classical control. This is, to the best of the authors' knowledge, one of the earliest experimentally validated applications of CS-NMPC to an RSM.

6.1.2 Background on RSMs and NMPC

In order to facilitate the discussion of the design and implementation of the proposed controller, in the following, mathematical models of RSMs and voltage source inverters (VSI) will be derived and numerical methods for NMPC will be introduced. Note that the argument (t), used to denote dependence on time, is sometimes dropped for the sake of readability.

6.1.3 Generic model of the RSM

The machine model in the synchronously rotating (d,q)-reference frame is given by [Hackl, 2017, Chap. 14]

$$\begin{aligned} u_{\mathrm{s}} &= R_{\mathrm{s}} i_{\mathrm{s}} + \omega \overbrace{\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}}^{=:J} \psi_{\mathrm{s}}\big(i_{\mathrm{s}}\big) + \tfrac{\mathrm{d}}{\mathrm{d}t}\psi_{\mathrm{s}}\big(i_{\mathrm{s}}\big), \\ \tfrac{\mathrm{d}}{\mathrm{d}t}\omega &= \frac{n_{\mathrm{p}}}{\Theta}\Big[m_{\mathrm{m}}(i_{\mathrm{s}}) - m_{\mathrm{l}}\Big], \qquad \tfrac{\mathrm{d}}{\mathrm{d}t}\phi = \omega, \end{aligned} \tag{6.1}$$

where $u_{\mathrm{s}} := (u_{\mathrm{s}}^{\mathrm{d}},\, u_{\mathrm{s}}^{\mathrm{q}})^\top$ are the applied stator voltages, R_{s} is the stator resistance, $i_{\mathrm{s}} := (i_{\mathrm{s}}^{\mathrm{d}},\, i_{\mathrm{s}}^{\mathrm{q}})^\top$ are the stator currents and $\psi_{\mathrm{s}} := (\psi_{\mathrm{s}}^{\mathrm{d}},\, \psi_{\mathrm{s}}^{\mathrm{q}})^\top$ are the stator flux linkages (functions of i_{s}). The (d,q)-reference frame rotates with electrical angular velocity $\omega = n_{\mathrm{p}}\,\omega_{\mathrm{m}}$ of the rotor, where n_{p} is the number of pole pairs and ω_{m} denotes the mechanical angular velocity of the machine. Furthermore, Θ is the total moment of inertia,

$$m_{\mathrm{m}}(i_{\mathrm{s}}) := \tfrac{3}{2} n_{\mathrm{p}}\,(i_{\mathrm{s}})^\top J \psi_{\mathrm{s}}\big(i_{\mathrm{s}}\big) \tag{6.2}$$

is the electro-magnetic machine torque, and m_{l} represents an external (time-varying) bounded load torque.

In order to formulate an optimal control problem, the flux dynamics can be described, based on (6.1), by the following differential algebraic equation (DAE):

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}\psi_{\mathrm{s}} &= u_{\mathrm{s}} - R_{\mathrm{s}} i_{\mathrm{s}} - \omega J \psi_{\mathrm{s}} + v, \\ 0 &= \psi_{\mathrm{s}} - \Psi_{\mathrm{s}}(i_{\mathrm{s}}), \end{aligned} \tag{6.3}$$

where $\Psi_{\mathrm{s}} := (\Psi_{\mathrm{s}}^{\mathrm{d}},\, \Psi_{\mathrm{s}}^{\mathrm{q}})^\top : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defines the algebraic constraints based on the identified flux maps and $v := (v^{\mathrm{d}},\, v^{\mathrm{q}})^\top$ are additive disturbances which will be used in an offset-free NMPC setting (see Section 6.1.7).

Based on the available flux maps computed through the finite element method (FEM), we obtained a continuously differentiable model by fitting a simple grey box model. Due to their low number of parameters and simple structure, we propose the following parametrization of the flux maps:

$$\begin{aligned}&\Psi_{\mathrm{s}}^{\mathrm{d}}(i_{\mathrm{s}}^{\mathrm{d}},i_{\mathrm{s}}^{\mathrm{q}},\theta_{\mathrm{d}}) = \\ &\qquad \frac{c_0^{\mathrm{d}}}{\sqrt{2\pi\sigma_{\mathrm{q}}^2}} \exp\left(-\gamma\left(i_{\mathrm{s}}^{\mathrm{q}},\sigma_{\mathrm{q}}\right)\right) \operatorname{atan}(c_1^{\mathrm{d}}\, i_{\mathrm{s}}^{\mathrm{d}}) + c_2^{\mathrm{d}}\, i_{\mathrm{s}}^{\mathrm{d}}\end{aligned} \tag{6.4}$$

and

$$\begin{aligned}&\Psi_{\mathrm{s}}^{\mathrm{q}}(i_{\mathrm{s}}^{\mathrm{d}},i_{\mathrm{s}}^{\mathrm{q}},\theta_{\mathrm{q}}) = \\ &\qquad \frac{c_0^{\mathrm{q}}}{\sqrt{2\pi\sigma_{\mathrm{d}}^2}} \exp\left(-\gamma\left(i_{\mathrm{s}}^{\mathrm{d}},\sigma_{\mathrm{d}}\right)\right) \operatorname{atan}(c_1^{\mathrm{q}}\, i_{\mathrm{s}}^{\mathrm{q}}) + c_2^{\mathrm{q}}\, i_{\mathrm{s}}^{\mathrm{q}},\end{aligned} \tag{6.5}$$

with

$$\gamma(x,y) := \frac{1}{2}\left(\frac{x}{y}\right)^2 \tag{6.6}$$

and where the unknown parameters involved are

$$\theta_{\mathrm{d}} := (c_0^{\mathrm{d}},\, c_1^{\mathrm{d}},\, c_2^{\mathrm{d}},\, \sigma_{\mathrm{d}}) \tag{6.7}$$

and

$$\theta_{\mathrm{q}} := (c_0^{\mathrm{q}},\, c_1^{\mathrm{q}},\, c_2^{\mathrm{q}},\, \sigma_{\mathrm{q}}). \tag{6.8}$$

This parametrization of the flux maps is, to the authors' best knowledge, novel and it is able to capture the main features of the flux maps with only 4 parameters per flux component. The numerical values of the coefficients can be computed by solving the following (decoupled) nonlinear least-squares problems:

$$\begin{aligned}&\min_{\theta_{\mathrm{d}}} \sum_{j=1}^{m}\sum_{k=1}^{n} \left(\Psi_{\mathrm{s}}^{\mathrm{d}}(\bar{i}_{\mathrm{s},j}^{\mathrm{d}},\bar{i}_{\mathrm{s},k}^{\mathrm{q}},\theta_{\mathrm{d}}) - \hat{\Psi}_{\mathrm{s}}^{\mathrm{d}}(\bar{i}_{\mathrm{s},j}^{\mathrm{d}},\bar{i}_{\mathrm{s},k}^{\mathrm{q}})\right)^2 \\ &\min_{\theta_{\mathrm{q}}} \sum_{j=1}^{m}\sum_{k=1}^{n} \left(\Psi_{\mathrm{s}}^{\mathrm{q}}(\bar{i}_{\mathrm{s},j}^{\mathrm{d}},\bar{i}_{\mathrm{s},k}^{\mathrm{q}},\theta_{\mathrm{q}}) - \hat{\Psi}_{\mathrm{s}}^{\mathrm{q}}(\bar{i}_{\mathrm{s},j}^{\mathrm{d}},\bar{i}_{\mathrm{s},k}^{\mathrm{q}})\right)^2\end{aligned} \tag{6.9}$$

where $\bar{i}_{\mathrm{s},j}^{\mathrm{d}}$ and $\bar{i}_{\mathrm{s},k}^{\mathrm{q}}$ are the j-th and k-th current data points associated with the flux values $\hat{\Psi}_{\mathrm{s}}^{\mathrm{d}}$ and $\hat{\Psi}_{\mathrm{s}}^{\mathrm{q}}$ obtained from FEM analysis. The fitting problems have been solved with the `MATLAB` Curve Fitting Toolbox and the resulting fitted model is shown in Figure 6.2.

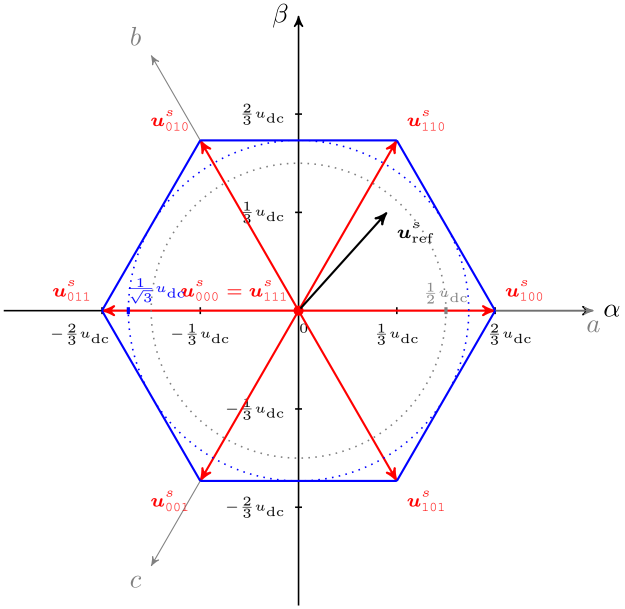

β
b
$\frac{2}{3}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{010}$
$\boldsymbol{u}^s_{110}$
$\frac{1}{3}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{\mathrm{ref}}$
$\boldsymbol{u}^s_{011}$
$\frac{1}{\sqrt{3}}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{000} = \boldsymbol{u}^s_{111}$
$\frac{1}{2}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{100}$
α
a
$-\frac{2}{3}u_{\mathrm{dc}}$
$-\frac{1}{3}u_{\mathrm{dc}}$
0
$\frac{1}{3}u_{\mathrm{dc}}$
$\frac{2}{3}u_{\mathrm{dc}}$
$-\frac{1}{3}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{001}$
$-\frac{2}{3}u_{\mathrm{dc}}$
$\boldsymbol{u}^s_{101}$
c

Figure 6.3: Voltage hexagon associated with the two-level VSI.

6.1.4 Model of the two-level VSI

The machine is supplied by a two-level voltage source inverter (VSI), which – on average over one switching period T_{s} – translates a given voltage reference

$$u^{\mathrm{s}}_{\mathrm{s,ref}} := (u^{\alpha}_{s,\mathrm{ref}}, u^{\beta}_{s,\mathrm{ref}}) \tag{6.10}$$

(in the stationary $s = (\alpha, \beta)$-reference frame) into the inverter output voltage $u^{\mathrm{s}}_{\mathrm{s}}$, i.e.

$$u^{\mathrm{s}}_{\mathrm{s}}(k\,T_{\mathrm{s}}) \approx u^{\mathrm{s}}_{\mathrm{s,ref}}((k-1)T_{\mathrm{s}}), \quad k \in \nabla. \tag{6.11}$$

Since a two-level voltage source inverter may produce a total of eight unique switching vectors, i.e. $s^{abc}_{\mathrm{s}} := (s^{\mathrm{a}}_{\mathrm{s}}, s^{\mathrm{b}}_{\mathrm{s}}, s^{\mathrm{c}}_{\mathrm{s}})^{\top} \in \{000, 001, 010, 100, 011, 101, 110, 111\}$, the typical voltage hexagon in the $\alpha\beta$-plane is obtained (see Figure 6.3), where

$$u^{\mathrm{s}}_{\mathrm{s}} = \kappa\, u_{\mathrm{dc}} \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} s^{abc}_{\mathrm{s}} \tag{6.12}$$

depends on the switching vector s^{abc}_{s} and the Clarke-factor $\kappa \in \{2/3, \sqrt{2/3}\}$ [Hackl, 2017, Chap. 14]. Using space-vector modulation (SVM) to generate the switching vector, any voltage reference within the circle of radius $u_{\mathrm{dc}}/\sqrt{3}$ can

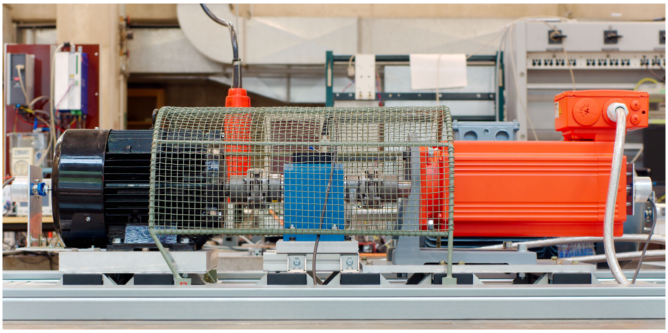


Figure 6.4: Laboratory setup including `dSPACE` real-time system, voltage-source inverters connected back-to-back, RSM, PMSM and torque sensor.

be realized, with u_{dc} denoting the (assumed constant) DC-link voltage. Finally, the inverter output voltage is transformed into the rotating (d,q)-reference frame using the inverse Park transformation, i.e.

$$u_{\mathrm{s}} = \begin{bmatrix} u_{\mathrm{s}}^{\mathrm{d}} \\ u_{\mathrm{s}}^{\mathrm{q}} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}}_{=:T_{\mathrm{P}}(\phi)^{-1}} u_{\mathrm{s}}^{\mathrm{s}}. \tag{6.13}$$

From now on, since we will only refer to currents, fluxes and voltages applied to the stator and in the (d-q)-frame, we will simplify the notation by dropping the associated subscript such that, for example, $i = (i_{\mathrm{s}}^{\mathrm{d}}, i_{\mathrm{s}}^{\mathrm{q}})$ denotes the stator currents in the (d-q)-frame.

6.1.5 Nonlinear model predictive control formulation

In this section, we will regard the following standard tracking formulation with prediction horizon T_h and N shooting nodes, where the squared deviation of fluxes ψ and voltages u from properly defined steady-state references are penalized:

$$
\begin{aligned}
\min_{\substack{\psi_0,\ldots,\psi_N\\ u_0,\ldots,u_{N-1}}} \quad & \frac{T_h}{2N}\sum_{i=0}^{N-1}\left\|\begin{matrix}\psi_i-\bar{\psi}\\ u_i-\bar{u}\end{matrix}\right\|_W^2+\frac{1}{2}\|\psi_N-\bar{\psi}\|_{W_N}^2\\
\text{s.t.} \quad & \psi_0-\psi_{\mathrm{e}}=0,\\
& g(\psi_i,u_i,\omega_{\mathrm{e}},v_{\mathrm{e}})-\psi_{i+1}=0, \quad i=0,\ldots,N-1,\\
& u_i^\top u_i\le\left(\frac{u_{\mathrm{dc}}}{\sqrt{3}}\right)^2, \qquad i=0,\ldots,N-1,\\
& \hat{C}u_i\le\hat{c}, \qquad i=0,\ldots,N-1,
\end{aligned}
\tag{6.14}
$$

where g describes the discretized dynamics obtained by integrating the differential-algebraic model in (6.3) using the Gauss-Legendre collocation method of order 2 assuming constant (estimated) angular velocity ω_{e} and disturbances v_{e}. The variables $\bar{\psi}$ and $\bar{u}$ denote the steady-state references computed for a given desired torque using a maximum-torque-per-Ampere (MTPA) criterion [Eldeeb et al., 2017]. Given the flux maps obtained from FEM data in Figure 6.2, it is possible to compute off-line lookup tables (LUTs) that contain the MTPA reference fluxes and voltages for a finite number of values of the target torque in a specified range. The LUTs are then interpolated online in order to compute approximate values of $\bar{\psi}$ and $\bar{u}$ associated with the specified target torque $\bar{m}$ (see Figure 6.5).

The convex quadratic constraint in (6.14) describes the circular input feasible set introduced in Section 6.1.4. Finally, $\hat{C}$ and $\hat{c}$ define polytopic constraints (which we will later refer to as "safety" constraints) that are meant to be always inactive at any local solution of (6.14) (apart from a finite number of points where they are locally equivalent to the linearized spherical constraint), but can mitigate constraint violation of intermediate SQP iterates. In particular, we define $\hat{C}$ and $\hat{c}$ such that the affine constraint defines an outer polytopic approximation with 6 facets as depicted in Figure 6.6. Notice that due to this formulation of the feasible set, linear independence constraint qualification can fail at a finite number of points where the linearization of the nonlinear constraint is equivalent to one of the affine constraints in (6.14). Although, this would violate a common assumption used in convergence theory for both SQP and some numerical methods for the solution of convex QPs, the active-set solver `qpOASES` that we employ for this application can handle redundant constraints through a strategy that determines which constraint needs to be removed from the working set [Ferreau et al., 2008].

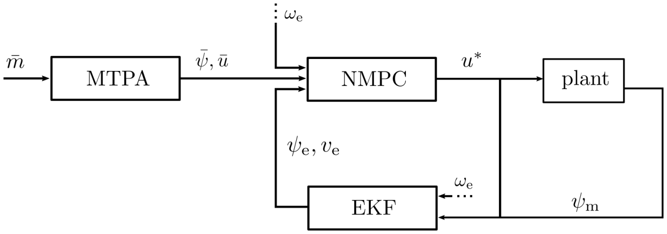


Figure 6.5: Control diagram: the MTPA LUTs provide the reference flux $\bar{\psi}$ and voltage $\bar{u}$ associated with a given reference torque $\bar{m}$. The NMPC controller computes the optimal control action based on the current state and disturbance estimate provided by an EKF.

Remark 6.1.1. *Notice that the actual dynamics of the system involve a coupling of mechanical (ω) and electrical states (ψ). It is however common, given the large difference between associated time constants, to assume a constant angular velocity ω when designing controllers. In our case, it allows us to use much shorter prediction horizons since we do not require the OCP in* (6.14) *to steer the speed of the motor to the desired reference, but only fluxes which directly map to currents and, for a given speed, to torques.*

Problem (6.14) is used to define an implicit feedback policy that requires the solution of an instance of the parametric NLP at every sampling time, where the value of the parameter ψ_e is given by the current estimate of the system's state. The resulting solutions are feasible with respect to the constraints and minimize (at least locally) the cost function. Nominal and inherently robust stability of the closed-loop system can be guaranteed in a neighborhood of a steady-state by properly choosing the terminal cost [Rawlings et al., 2017].

Remark 6.1.2. *Notice that formulations more general than* (6.14) *can in principle be used in the framework of NMPC. Among others, economic costs and more general nonlinear constraints and nonlinear cost terms, are features that can be included in the problem in order to better capture control design requirements. However, for the application discussed in this section, the nonlinear least-squares problem described in* (6.14) *is general enough.*

6.1.6 Numerical methods and software for NMPC

In order to be able to solve problem (6.14) within the available computation time, the use of efficient numerical methods is fundamental.

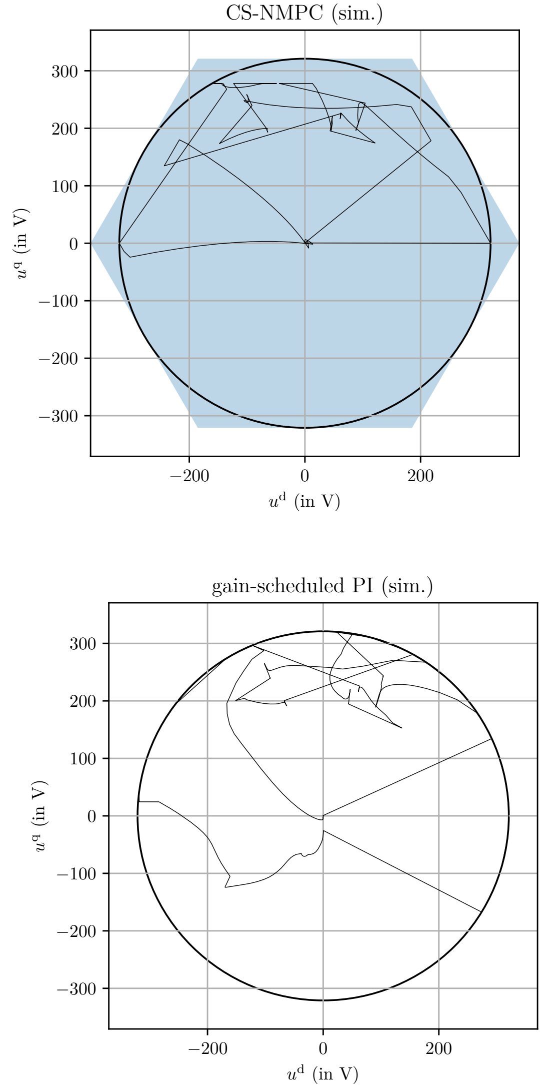


Figure 6.6: Current steps at $157\,\mathrm{rad\,s^{-1}}$ (simulation): results obtained using the CS-NMPC (left) and the gain-scheduled PI controller (right). The voltage spherical constraints are directly included into the control formulation using the SCQP strategy proposed in [Verschueren et al., 2016]. Additionally, a “safety” polytopic constraint is included which, due to its linearity, is always satisfied exactly.

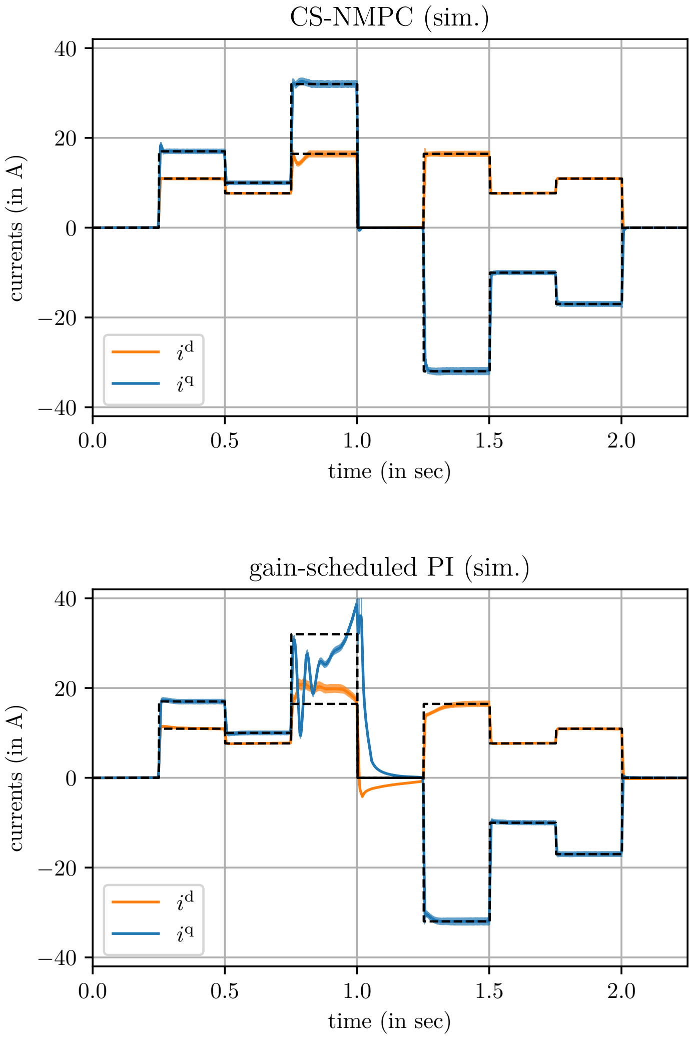


Figure 6.7: Current steps at 157 rad s^{-1} (simulation): results obtained using the CS-NMPC (left) and gain-scheduled PI controller (right). The CS-NMPC controller outperforms the PI controller, especially when the input constraints become active (e.g., between $t = 0.75\,\mathrm{s}$ and $t = 1.00\,\mathrm{s}$). At the same time, a faster transient can be achieved even when the constraints become active only for a short time.

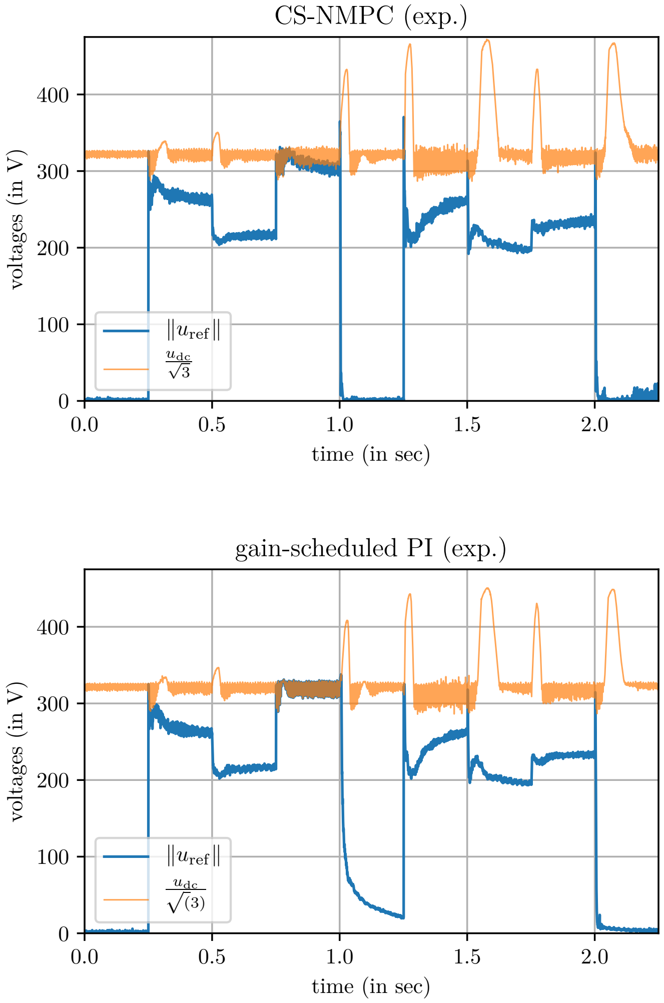


Figure 6.8: Current steps at $157\,\mathrm{rad\,s^{-1}}$ (experiment): two-norm of measured voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference. Notice that the input commanded by the PI controller remains saturated during the entire step. On the contrary, the CS-NMPC controller, after an initial saturation, steers the current to the (feasible) reference values.

Due to the computational burden associated with the solution of the QPs and re-linearization of the original NLP in (6.14), several approximate strategies can be used that can significantly reduce computation times (e.g., [Zanelli et al., 2019a], [Graichen and Kugi, 2010], [Feller and Ebenbauer, 2017]). In this work, we will use the RTI strategy [Diehl, 2001, Diehl et al., 2002a], which relies on a single SQP iteration in order to provide an approximate feedback law and whose stability properties have been analyzed in Chapter 3. In particular, the NMPC controller used both in simulations and experiments uses the implementation of the RTI available in `acados`, which we briefly describe below.

The `acados` framework

The high-performance software package `acados` [Verschueren et al., 2021] provides a modular framework for NMPC and moving horizon estimation (MHE). It consists of a C library that implements building blocks needed to solve NLPs arising from NMPC and MHE formulations. It relies on the high-performance linear algebra package `BLASFEO` [Frison et al., 2018] and on the quadratic program (QP) solver `HPIPM` [Frison et al., 2020] and contains efficient implementations of explicit and implicit integration methods. Moreover, it interfaces a number of QP solvers such as `qpOASES` [Ferreau et al., 2014] and `OSQP` [Stellato et al., 2020] and it provides high-level Python and `MATLAB` interfaces. Through these interfaces, one can conveniently specify optimal control problems and code-generate a self-contained C library that implements the desired solver and can be easily deployed onto embedded control units such as `dSPACE` using the automatically generated C wrapper and S-Function. The code-generation takes place through templated C code which is rendered by the `Tera` templating engine written in Rust. In this way, human-readable C code can be generated that facilitates the deployment on the target hardware.

6.1.7 NMPC offset-free tracking formulation

In order to achieve offset-free regulation, we adopt the standard strategies discussed, for example, in [Pannocchia and Rawlings, 2003]. In particular, we use the following augmented dynamics to design an extended Kalman filter (EKF):

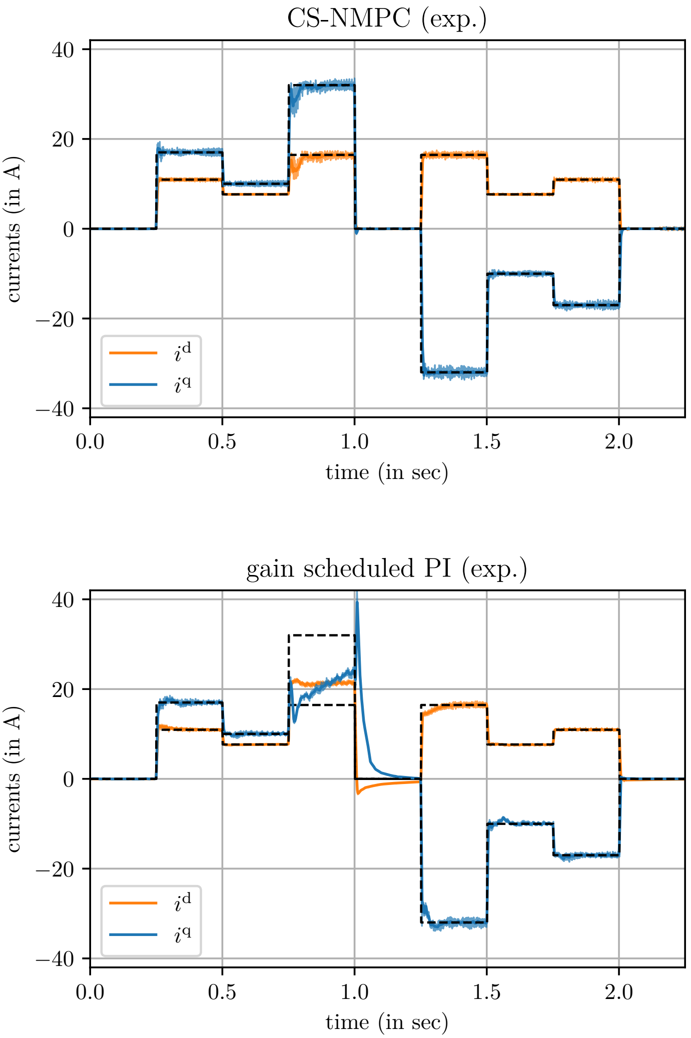


Figure 6.9: Current steps at $157\,\mathrm{rad\,s^{-1}}$ (experiment): results obtained using the proposed CS-NMPC controller (left) and gain-scheduled PI controller (right). The CS-NMPC controller outperforms the PI controller, especially when the input constraints become active (e.g., between $t = 0.75\,\mathrm{s}$ and $t = 1.00\,\mathrm{s}$). At the same time, as it can be seen especially between $t = 1.25\,\mathrm{s}$ and $t = 1.50\,\mathrm{s}$, a faster transient can be achieved, even when the constraints are active only for a short time.

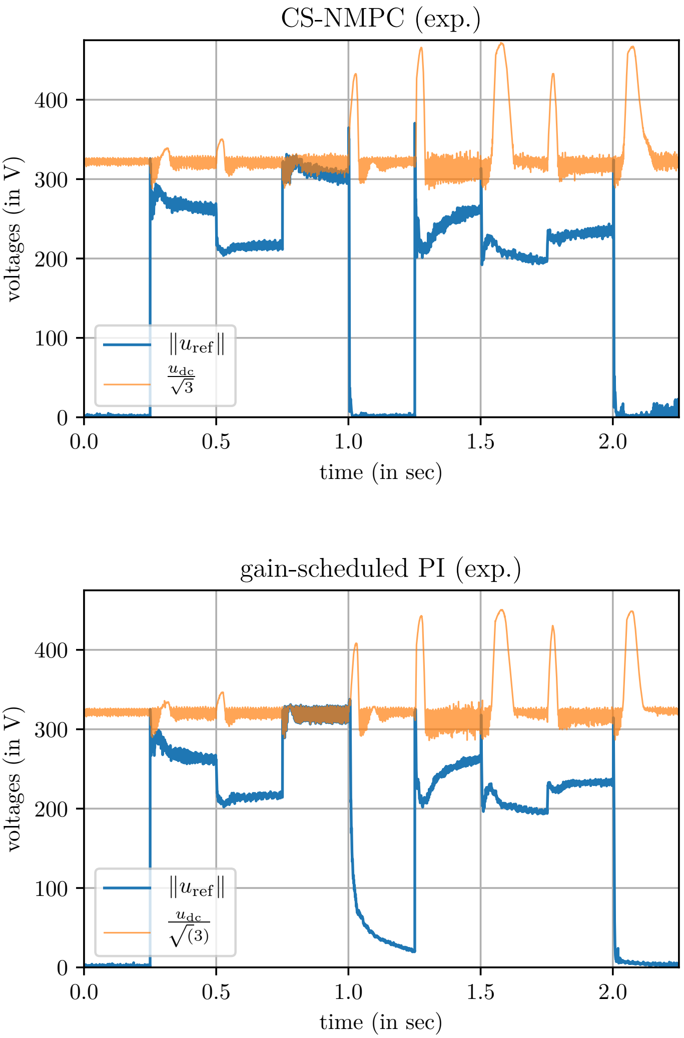


Figure 6.10: Current steps at 157 rad s^{-1} (experiment): two-norm of measured voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference. Notice that the input commanded by the PI controller remains saturated during the entire step. On the contrary, the CS-NMPC controller, after an initial saturation, steers the current to the (feasible) reference values.

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}\psi &= u - Ri - \omega J\psi + v, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}v &= 0, \\ 0 &= \psi - \Psi(i), \end{aligned} \tag{6.15}$$

where the disturbance state v is introduced and we assume that pseudo-measurements ψ_{meas} are available through the interpolated FEM flux maps:

$$\psi_{\mathrm{meas}} = \hat{\Psi}(i_{\mathrm{meas}}), \tag{6.16}$$

while current measurements i_{meas} are physically carried out on the machine. A standard EKF is designed using (6.15) and (6.16) which uses flux measurements to estimate fluxes ψ_{e} and disturbances v_{e}. Notice that the angular velocity ω_{e} is estimated externally and is considered as a constant-over-time parameter that is updated at every sampling time. In [Pannocchia et al., 2015, Theorem 14] a streamlined version of the results from [Muske and Badgwell, 2002, Pannocchia and Rawlings, 2003, Morari and Maeder, 2012] is presented, where under the assumptions, among others, of observability of the augmented dynamics (6.15) and asymptotically constant disturbances v, the steady-state of the closed-loop system can be proved to be offset-free.

6.2 Simulation and experimental results

An RTI strategy [Diehl et al., 2002a], where a single QP of an SQP algorithm is carried out per sampling time, is used to solve (6.14). In particular, the generalized Gauss-Newton Hessian approximation proposed in [Verschueren et al., 2021] is used. In this way, the (positive) curvature contribution due to the convex spherical constraints on voltages can be exploited in order to improve the Hessian approximation used in the QP subproblems. Although in our experience this improves a lot the convergence of the RTI iterates on this specific problem, the approximate feedback law can, from time to time, be largely infeasible with respect to the nonlinear spherical constraints (recall that the intermediate full-step SQP iterates are feasible only with respect to linear constraints). Since a-posteriori projection of the control actions onto the feasible set can largely deteriorate the control performance, we add extra polytopic "safety" constraints (defined by $\hat{C}$ and $\hat{c}$ in (6.14)) around the spherical ones in order to ensure that the constraint violation is bounded at any successfully computed iterate.

In order to be able to meet the short sampling times required to control the electrical drive, we use a prediction horizon of $T_h = 3.2$ ms obtained with 2 shooting nodes ($N = 2$) and we use the QP solver `qpOASES`, which is particularly suited for problems with short horizons [Kouzoupis et al., 2018]. For both simulation and experimental results the controller is run at 4 kHz.

Remark 6.2.1. *Notice that the discretization time* ($T_h/2 = 1.6\,\mathrm{ms}$) *used in the optimal control problem is not equivalent to the sampling time* $T_s = 0.25\,\mathrm{ms}$. *This setting, which we might call "oversampled" NMPC, allows one to obtain a longer prediction horizon without increasing the number of optimization variables. Although a theoretical analysis of this strategy is well beyond the scope of this section, we point the interested reader to [Grüne and Nesić, 2003] where it is shown that fundamental properties of the feedback policy hold for the oversampled setting. Moreover, notice that this setting is used in [Zanelli et al., 2020] and [Zanelli et al., 2021b] in order to prove asymptotic stability of the combined system-optimizer dynamics.*

We have tuned the weights in (6.14) until satisfactory closed-loop performance could be achieved in simulation resulting in $W = \mathrm{blkdiag}(312.5 \cdot \mathbb{I}_2, 1e^{-4} \cdot \mathbb{I}_2)$ and W_N set to the corresponding LQR cost obtained with the dynamics linearized at $i = 0$, $\psi = 0$, $u = 0$ and $\omega_\mathrm{m} = 0$. Although the terminal cost should in general be updated together with the desired reference, a fixed terminal cost was able to provide satisfactory control performance.

In the following, we discuss simulation and experimental results obtained using the above described RTI strategy to solve (6.14) with `acados`.

6.2.1 Simulation results

In order to validate, first in simulation and then experimentally, the proposed approach, we regard a setting where the RSM is connected to a permanent synchronous machine (PMSM) which can be used to simulate different load conditions. The CS-NMPC controller has been implemented in `acados` using its Python interface and integrated in a `Simulink` model that makes use of a high-fidelity model of the system to be controlled including a model of the PMSM and of the two-level VSI described in Section 6.1.2. Moreover, we have implemented an EKF based on the augmented model (6.15) using the implicit integrators available in `acados`.

We set the PMSM's controller such that it maintains a constant rotational speed and we change the torque reference fed to the RSM's controller to assess the tracking performance of the proposed controller. We compare the closed-loop trajectories obtained with the ones achieved when using instead the gain-scheduled PI controller with anti-windup presented in [Hackl, 2015]. The parameter tuning used in [Hackl, 2015] was used as baseline and we adapted

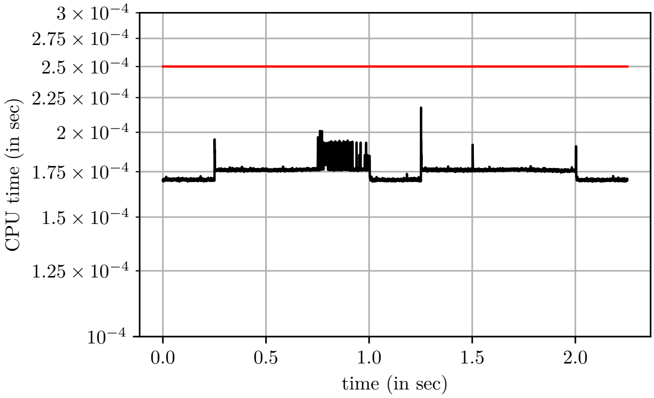


Figure 6.11: Current steps at 157 rad s^{-1}: overall control loop turnaround time (in black) obtained with the CS-NMPC controller using `acados` with `qpOASES` (available computation time of 250 µs in red). About 90% of the computation time is due the CS-NMPC controller (together with the EKF).

the parameters until the PI controller was able to stabilize the system and achieve satisfactory control performance for the scenario under analysis. In particular, we had to scale down the proportional and integral coefficients by a factor two. For the sake of reproducibility the entire simulation environment is made available at `https://github.com/zanellia/cs_nmpc_rsm`.

In order to highlight the advantages of using a controller which can handle constraints directly, we set the reference speed to a value that is close to the limit value $\omega_\text{m}^\star$ computed as follows:

$$\begin{aligned} \omega_\text{m}^\star := \quad & \frac{1}{2}\arg\max_{\omega} \quad \omega \\ & \qquad\qquad \text{s.t.} \quad \|R_s i_\text{ref} + \omega J \psi_\text{ref}\|_2 \leq \left(\frac{u_\text{dc}}{\sqrt{3}}\right). \end{aligned} \tag{6.17}$$

Since the optimal value is achieved at the boundaries of the feasible set, we can simply solve for ω the quadratic equation

$$\|R_s i_\text{ref} + \omega J \psi_\text{ref}\|_2^2 - \left(\frac{u_\text{dc}}{\sqrt{3}}\right)^2 = 0, \tag{6.18}$$

such that, for the values $i_\text{ref} = (16.45, 31.99)$ A and $\psi_\text{ref} = (0.819, 0.417)$ Wb associated with the torque value 58 N m and $u_\text{dc} = 556$ V, we obtain $\omega_\text{m}^\star = 169.32$ rad s^{-1}. Hence, we set $\omega_\text{m,ref} = 157$ rad s^{-1} $\approx \omega_\text{m,nom}$ for both the

par.	value	par.	value
R_s	$0.4\,\Omega$	$\omega_\mathrm{m,nom}$	$157.07\,\mathrm{rad\,s^{-1}}$
$m_\mathrm{m,nom}$	$61\,\mathrm{N\,m}$	$\hat{\imath}_\mathrm{s,max}$	$29.7\,\mathrm{A}$
$\hat{u}_\mathrm{s,max}$	$556\,\mathrm{V}$	–	

Table 6.1: Parameters of physical setup (used in simulation too).

simulation and experimental scenarios. Finally, the parameters used in the simulations match the ones of the physical setup and are reported in Table 6.1. The current trajectories obtained with the CS-NMPC and PI controller are reported in Figure 6.7 (similarly for input trajectories in Figure 6.6). It is clear from the plots that the tracking performance achieved by the CS-NMPC controller is largely superior to the one obtained by the PI controller, especially when the input constraints become active (e.g., between $t = 0.75\,\mathrm{s}$ and $t = 1.00\,\mathrm{s}$). At the same time, as it can be seen from the current trajectories in Figure 6.6 between $t = 1.25\,\mathrm{s}$ and $t = 1.50\,\mathrm{s}$, a faster transient can be achieved, even when the constraints are active only for a short time. In Appendix A.6 we report additional results obtained with a slightly increased reference speed $\omega_\mathrm{m,ref} = 165\,\mathrm{rad\,s^{-1}}$.

6.2.2 Experimental Results

The presented NMPC scheme has been deployed on a custom-built 9.6 kW RSM (Courtesy of Prof. Maarten Kamper, Stellenbosch University, South Africa) with the parameters reported in Table 6.1. and the nonlinear flux linkage maps as depicted in Figure 6.2 (maps were obtained from FEM). The overall laboratory setup is depicted in Figure 6.4 and comprises the `dSPACE` real-time system with processor board DS1007 and various extensions and I/O boards, two 22 kW SEW inverters in back-to-back configuration sharing a common DC-link. Moreover, it comprises the HOST-PC running `MATLAB/Simulink` with RCPHIL R2017 and `dSPACE ControlDesk` 6.1p4 for rapid-prototyping, data acquisition and evaluation, the custom-built 9.6 kW RSM as device under test and a 14.5 kW SEW PMSM as load machine. The DR2212 torque sensor allows to measure the mechanical torque too, but it was not used during the experiments. The CS-NMPC controller based on the formulation described in Section 6.2.1 and implemented using the `acados` framework has been deployed on the `dSPACE` unit connected to the physical RSM.

In order to validate the control performance of the proposed CS-NMPC controller we reproduced the scenario used for the simulation reported in Section 6.2.1, i.e., we used the PMSM to maintain the nominal rotational speed of the rotor

($157\,\mathrm{rad\,s^{-1}}$) and different torque references have been fed to the RSM controllers under analysis.

The closed-loop trajectories for the conducted experiments are reported in Figure 6.9-6.10. Similarly to the results obtained in simulation (see Figures 6.7 and 6.6), the proposed CS-NMPC controller achieves better tracking performance than the gain-scheduled PI controller, in particular when the voltage constraint becomes active (especially between $t = 0.75\,\mathrm{s}$ and $t = 1.00\,\mathrm{s}$). Notice that there is a non negligible discrepancy between simulation and experimental results potentially due to model mismatches. Among other possible causes of discrepancies, we mention the fact that the modelled flux maps might differ from the real ones. Moreover, we observe from Figure 6.10 that the DC-link voltage fluctuates around its nominal value. In fact, in the presence of a sudden change in the torque reference, the voltage of the DC-link capacitor can drop if the recharging rate is slower than the discharging rate (behavior not modelled in simulation). Although this behavior is not accounted for in the dynamics of the system used to design the controllers under analysis, in both controllers we exploit the measured DC-link voltage in order to adjust the feasible set. Notice that the u_{ref} computed by the CS-NMPC controller becomes sometimes infeasible. This infeasibility is consistent with the fact that the quadratic constraint can be violated during transient and will only be satisfied at the steady. However, due to the safety polytopic constraints introduced in (6.14), we can guarantee that the computed solution will not violate the outer voltage hexagon. At the same time, whenever a voltage reference that lies outside of the circular feasible set is fed to the modulator, a projection onto the disk of radius $\frac{u_{\mathrm{dc}}}{\sqrt{3}}$ is carried out.

6.3 Chapter summary and outlook

In this chapter we presented simulation and experimental results obtained with a CS-NMPC torque controller for RSMs. As opposed to most successful implementations present in the literature, that use instead FS-MPC/NMPC, we showed the effectiveness and real-time feasibility of the continuous control set approach. In particular, we showed that, using the software implementation of the *real-time iteration method* for NMPC available in the software package `acados`, it is possible to deploy the proposed controller on embedded hardware and to meet the challenging sampling times typically required to control electrical drives. We discussed implementation details and reported on simulation as well as experimental results which show that the proposed approach can largely outperform state-of-art control methods especially when the input constraints become active.

Future research will involve the investigation of novel numerical methods, e.g., the real-time first-order methods proposed in [Zanelli et al., 2019b] or the truncated SQP strategy outlined in Section 3.1, to speed up the computation times, which are currently still rather long and neither allow for extensions of the optimal control formulations (e.g., longer horizons, state or input spaces of higher dimension, etc) nor for deployment on hardware with lower computational power.

Chapter 7

`prometeo`: a domain specific language for embedded high-performance computing

Due to the considerable computational effort required to solve the nonconvex programs associated with NMPC, its successful application largely depends on the availability of computationally powerful hardware and high-performance implementations of algorithms for nonconvex optimization. At the same time, a second class of aspects that cannot be neglected when assessing applicability of optimization-based control solutions is the one of user-friendliness, scalability and maintainability. Although these aspects are crucial to the successful usage of any software, they are even more crucial in the context of this thesis due to the high algorithmic sophistication generally associated with NMPC, and embedded optimization in general, with respect to traditional control solutions.

7.1 Introducing `prometeo`

In this chapter, we introduce the experimental software package `prometeo` which provides a domain specific language (DSL) and a Python-to-C transpiler for embedded high-performance computing. The DSL made available with `prometeo` allows software developers to implement code for scientific computing using a subset of the Python language which can then be *transpiled* to high-performance and embeddable C code by `prometeo`'s transpiler. Its main features are:

- Python compatible syntax: `prometeo` is a DSL embedded into the Python language. `prometeo` programs can be executed from the Python interpreter.
- efficient: `prometeo` programs transpile to C code that relies on the high-performance library `BLASFEO`.
- statically typed: `prometeo` uses Python's native type hints to strictly enforce static typing.
- deterministic memory usage: a specific program structure is required and enforced through static analysis. In this way `prometeo` transpiled programs have a guaranteed maximum heap usage.
- fast memory management: thanks to its static analysis, `prometeo` can avoid allocating and garbage-collecting memory, resulting in faster and safer execution.
- self-contained and embeddable: unlike other similar tools and languages, `prometeo` targets specifically embedded applications and programs written in `prometeo` transpile to self-contained C code that does not require linking against the Python run-time library.

7.1.1 From throwaway to evolutionary prototyping

The main goal of `prometeo` is to provide a tool that can help to refine the commonly adopted paradigm adopted in the development of software for embedded high-performance computing which revolves around so-called *throwaway prototyping.* In fact, due to the nonnegligible complexity associated with the efficient implementation of numerical algorithms, it is common practice to prototype the algorithm of interest in a high-level programming language (e.g., Python, MATLAB or Julia [Bezanson et al., 2017]) that facilitates debugging and testing. In a second phase of the development, the entire implementation is carried out in a much lower level language (e.g., C) such that fundamental requirements on performance and deployability are met. This transition from high- to low-level programming languages often involves little to no code reuse and necessitates a rather error-prone process. In fact code that is meant to be deployable on embedded targets, and especially code that is meant to exploit the available computational capabilities at their best, often requires a tight interaction with hardware (and, quoting C++ 's creator Bjarne Stroustrup, *"fiddling with machine addresses and memory is rather unpleasant and not very productive"*). Fortunately, especially in the context of embedded high-performance computing, the most critical parts of the code base involve a number of patterns that repeat themselves. However, although repetitiveness can to some extent simplify the developer's job (e.g., developing and maintaining 10 variants of the same data structure is slightly easier than doing the same with 10 completely different data structures), this doesn't quite resolve the fundamental underlying issue. As it often happens when complexity is associated with a limited number of repetitive patterns, we argue that there is a large potential for abstraction of such patterns in this context and that this abstraction can be obtained with a dedicated DSL such as `prometeo`. The end goal of such a software tool is indeed to reduce the overhead associated with throwaway prototyping and to help to merge the prototyping and implementation phases of software development in the context of embedded high-performance computing.

7.1.2 Transpiling Python to C

A *transpiler* (also called *transcompiler* or source-to-source compiler) is a tool that is capable of translating a program to another language. Although this is, to some extent, also what a compiler does - in the sense that it translates source code into machine code - we make a distinction between compilers and transpilers based on the fact that the latter operate a conversion between languages of approximately the same level of abstraction. For example, `CFront`, the original compiler for C++ from around 1983, used to translate C++ code into C code, which, although is definitely a language that is "closer" to hardware than C++ , has approximately the same level of abstraction if compared to machine code (which would be the target language of a proper compiler such as `gcc`).

Although translating a program written in a language into another with a comparable level of abstraction can be significantly easier than translating to one with a very different level of abstraction (especially if the target language is of much lower level), translating Python programs into C programs still involves a considerable abstraction gap and it is not an easy task in general. Loosely speaking, the challenge lies in the necessity to reimplement features that are natively supported by the source language in the target language. In particular, when translating Python to C, the difficulty comes both from the different level of abstraction of the two languages and from the fact that the source and target languages are of two very different types: Python is an *interpreted*, *duck-typed* and *garbage-collected* language and C is a *compiled* and *statically typed* language.

The task of transpiling Python to C becomes even more challenging if we add the constraint that the generated C code must be efficient (even for small to medium scale computations) and deployable on embedded hardware. In fact these two requirements directly imply that the generated code cannot make use of: *i)* sophisticated runtime libraries, e.g., the Python runtime library, which are generally not available on embedded hardware *ii)* dynamic memory allocation that would make the execution slow and unreliable (exception made for memory that is allocated in a setup phase and whose size is known a priori).

Since source-to-source code transformation, or transpilation, and in particular transpilation of Python code into C code is not an unexplored realm, in the following, we mention a few existing projects that address it. In doing so, we highlight where and how they do not satisfy one of the two requirements outlined above, namely (small scale) efficiency and embeddability.

Related work

Several software packages exist that address Python-to-C translation in various forms.

In the context of high-performance computing, `Numba` [Lam et al., 2015] is a just-in-time compiler for numerical functions written in Python. As such, its aim is to convert properly annotated Python functions, not entire programs, into high-performance LLVM code such that their execution can be sped up. `Numba` uses an internal representation of the code to be translated and performs a (potentially partial) type inference on the variables involved in order to generate LLVM code that can be called either from Python or from C/C++ . In some cases, namely the ones where a complete type inference can be carried out successfully, code that does not rely on the C API can be generated (using the `nopython` flag). However, the emitted LLVM code would still rely on `Numpy` [Oliphant, 2006, Van Der Walt et al., 2011] for BLAS and LAPACK [Anderson et al., 1999] operations.

`Nuitka` [Nui, 2020] is a source-to-source compiler that can translate every Python construct into C code that links against the `libpython` library and it is therefore able to transpile a large class of Python programs. In order to do so, it relies on the fact that one of the most used implementations of the Python language, namely `CPython`, is written in C. In fact, `Nuitka` generates C code that contains calls to `CPython` that would normally be carried out by the Python parser. Despite its attractive and general transpilation approach, it cannot be easily deployed on embedded hardware due to its intrinsic dependency on `libpython`. At the same time, since it maps rather closely Python constructs to their `CPython` implementation, a number of performance issues can be expected

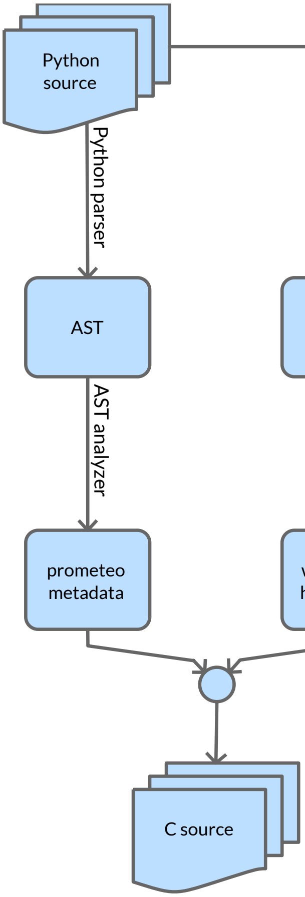


Figure 7.1: `prometeo`'s transpilation and worst case heap analysis.

when it comes to small to medium scale high-performance computing. This is particularly relevant due to the fact that operations associated with, for example, type checking, memory allocation and garbage collection that can slow down the execution are carried out by the transpiled program too.

`Cython` is a programming language whose goal is to facilitate writing C extensions for the Python language. In particular, it can translate (optionally) statically typed Python-like code into C code that relies on `CPython`. Similarly to the considerations made for `Nuitka`, this makes it a powerful tool whenever it is possible to rely on `libpython` (and when its overhead is negligible, i.e., when dealing with sufficiently large scale computations), but not in the context

```python
def function()
    a = 1
    return a
```

Listing 7.1: Simple Python program associated with the AST in Figure 7.2.

of interest here.

Finally, although it does not use Python as source language, we should mention that `Julia` too is just-in-time (and partially ahead-of-time) compiled into LLVM code. The emitted LLVM code relies however on the `Julia` runtime library such that considerations similar to the one made for `Cython` and `Nuitka` apply.

7.1.3 `prometeo`'s transpiler

Transpilation of programs written using a restricted subset of the Python language into C programs is carried out using `prometeo`'s transpiler. This source-to-source transformation tool analyzes abstract syntax trees (AST) associated with the source files to be transpiled in order to emit high-performance and embeddable C code. In order to do so, special rules need to be imposed on the Python code. This makes the otherwise extremely challenging task of transpiling an interpreted, high-level, and duck-typed language into a compiled low-level statically typed one possible. In doing so, we define what is sometimes referred to as an *embedded* DSL in the sense that the resulting language uses the syntax of a host language (Python itself) and, in `prometeo`'s case, it can also be executed by the standard Python interpreter.

Traversing Python abstract syntax trees

An AST is a representation of the abstract syntactic structure of a piece of source code written in a certain programming language. ASTs are commonly used in the context of compiled languages and, although it is not a compiled language, Python supports the generation and manipulation of ASTs and their compilation into bytecode through the built-in module `ast`. Figure 7.2 shows the AST associated with the simple Python program in Listing 7.1. Although most of the information needed to transpile the source code into C code is already contained in the AST, a fundamental piece of information is missing: types. In fact, since Python is a duck-typed language, types are never stated explicitly and the type of an object is instead determined by whether, at a given point during the execution, it has certain methods and properties. This

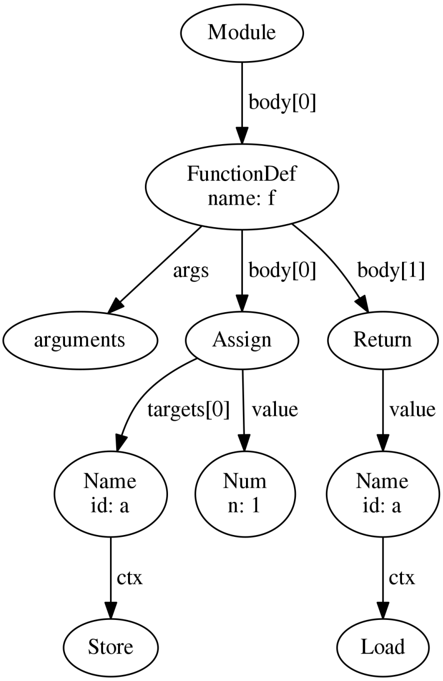


Figure 7.2: AST associated with Listing 7.1

extremely implicit and dynamic way of handling type checking, makes it difficult to transpile Python code into code written in a statically typed language. In particular, this operation requires that type inference is first carried out in order to determine the types of variables. However, type inference of an arbitrary Python program can be very challenging or computationally intractable. For this reason, we opt instead for a less flexible setup which goes in the direction of statically typed Python. To this end we leverage the so-called type hints and enforce their usage such that typing information can be easily extracted from programs and used to generate C code.

Type hints and static typing

Apart from other specific syntactic and structural restrictions that will be described in Section 7.2.1, a key restriction that allows the transpilation process to be greatly simplified is the one that every valid program needs to be *statically typed*. This is achieved by leveraging, and enforcing, the use of *type hints*, a feature defined by `PEP484` and available in Python3.6+ that allows the user to

```
a : int  = 1
```

Listing 7.2: Assignment with type annotation.

```
def function() -> int:
    a : int = 1
    return a
```

Listing 7.3: Simple Python program with type hints associated with the AST in Figure 7.3.

specify the type of variables involved in declarations and function calls. Listing 7.2 shows an assignment with a type hint that describes the type of the variable being declared. The information added through type hints is not used by the standard Python interpreter, but it is added to the AST. This means that a static analysis tool can in principle be used to determine the type of variables without executing the code. This is what is done, for example, by the static type checking tool `mypy`. Figure 7.3 shows how the typing information added in the adapted simple program in Listing 7.3 is incorporated into the AST.

Listing 7.4 shows instead the data structure associated with the AST of the program in Listing 7.1 generated by the `ast` module. `prometeo`'s transpiler walks these ASTs using a subclass of the `NodeVisitor` class provided with the `ast` module. In order to better understand how typed Python ASTs can be used to emit C code, let us consider the simple Python class described in Listing 7.5.

`prometeo` can traverse the typed AST associated with Listing 7.5 in order to generate the C code described in Listing 7.6 and 7.7. In particular, the class `simple_class` is mapped by `prometeo`'s transpiler into a C structure. In this structure, the class's attributes are translated into fields and its methods are implemented by means of functions pointed to by pointers that belong to the structure. Although the C++ -like function mangling used to allow overload makes the function names a little cryptic, in this simple example we can already appreciate some of the benefits of transpilation. First, overload itself would not be possible in C. Second, the emitted C code implements a class mimicking that is not very different from what is often done in hand-written C code.

```
Module(
   body=[
      FunctionDef(
         lineno=1,
         col_offset=0,
         name='f',
         args=arguments(args=[], ...),
         body=[
            AnnAssign(
               lineno=2,
               col_offset=4,
               target=Name(lineno=2, col_offset=4, id='a', ctx=Store()),
               annotation=Name(lineno=2, col_offset=8, id='int', ctx=Load()),
               value=Num(lineno=2, col_offset=14, n=1),
               simple=1,
            ),
            Return(
               lineno=3,
               col_offset=4,
               value=Name(lineno=3, col_offset=11, id='a', ctx=Load()),
            ),
         ],
         decorator_list=[],
         returns=Name(lineno=1, col_offset=11, id='int', ctx=Load()),
      ),
   ],
)
```

Listing 7.4: Data structure associated with the AST in Figure 7.3.

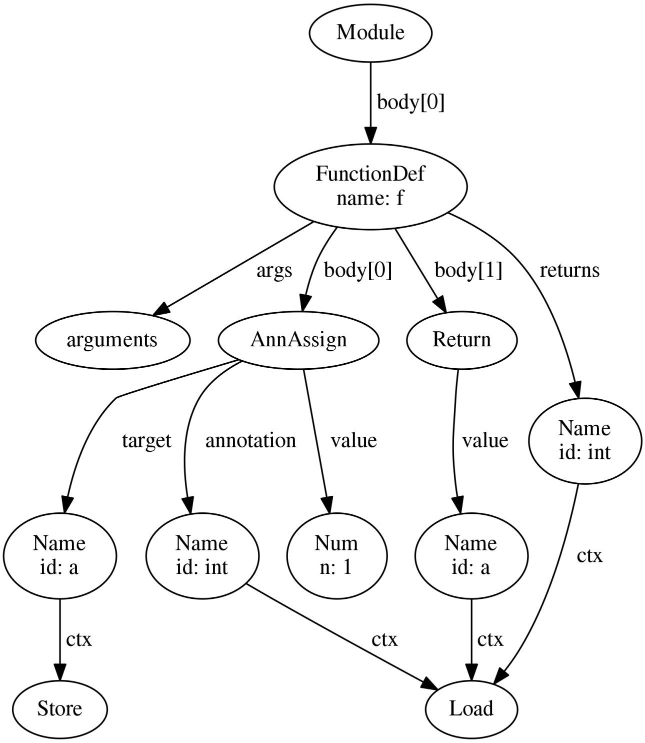


Figure 7.3: AST associated with Listing 7.3: typing information is added to the AST.

```
class Simple_class:
    def __init__(self) -> None:
        self.a: int = 1

    def method1(self) -> int:
        return self.a

    def method1(self, b : int) -> int:
        c : int = self.a + b
        return c
```

Listing 7.5: Simple Python class with type hints.

```c
typedef struct Simple_class Simple_class;

struct Simple_class{
    int a;
    void (*_Z8__init__)(Simple_class *self);
    int (*_Z7method1)(Simple_class *self);
    int (*_Z7method1int)(Simple_class *self, int b);
};

void (_Z8__init__Simple_class_impl)(Simple_class *self);
int (_Z7method1Simple_class_impl)(Simple_class *self);
int (_Z7method1intSimple_class_impl)(Simple_class *self, int b);
```

Listing 7.6: Simple Python class transpiled to C - header.

```c
void Simple_class_constructor(
  struct Simple_class *object){
    object->_Z8__init__ = &_Z8__init__Simple_class_impl;
    object->_Z7method1 = &_Z7method1Simple_class_impl;
    object->_Z7method1int = &_Z7method1intSimple_class_impl;
    object->_Z8__init__(object);
}

void _Z8__init__Simple_class_impl(Simple_class *self) {
    self->a = 1;
}

int _Z7method1Simple_class_impl(Simple_class *self) {
    return self->a;
}

int _Z7method1intSimple_class_impl(
  Simple_class *self, int b) {
    int c = self->a + b;
    return c;
}
```

Listing 7.7: Simple Python class transpiled to C - source.

```
from prometeo import *

n : dims = 10

def main() -> int:

    A: pmat = pmat(n, n)
    for i in range(n):
        for j in range(n):
            A[i, j] = 1.0

    B: pmat = pmat(n, n)
    for i in range(n):
        B[0, i] = 2.0

    C: pmat = pmat(n, n)

    pmat_print(A)
    pmat_print(B)
    C = A * B
    pmat_print(C)
    return 0
```

Listing 7.8: Simple program illustrating `prometeo`'s metadata.

7.1.4 Scope-dependent metadata

Although using typed ASTs it is possible to transpile most Python constructs using just local information, other tasks, such as heap usage analysis (see Section 7.1.5) and enforcement of type constraints, require knowledge which is typically not local. For example, when a variable is declared, it is necessary to check whether a variable with the same name was already declared in the current scope or, when calling a function or method, it is necessary to check that a function or method with the correct signature exists in the current scope. In particular, the transpiler creates dictionaries containing, e.g., scope-dependent information regarding the type of variables being declared and, for certain types of objects, their dimension. Such dictionaries are then stored in a cache folder such that are accessible at later stages of the transpilation and static analysis. Listings 7.8, 7.9 and 7.10 show a simple Python program and the associated metadata dictionaries containing type and dimension information.

```
"global": {},
"global@main": {
    "A": ["n","n"],
    "B": ["n","n"],
    "C": ["n","n"]
}
```

Listing 7.9: `prometeo` metadata dictionary containing dimensions to be used for heap analysis.

```
"global": {
    "n": "dims"
},
"global@main": {
    "A": "pmat",
    "B": "pmat",
    "C": "pmat"
}
```

Listing 7.10: `prometeo` metadata dictionary containing type information.

7.1.5 Static heap usage analysis

One of the key features of `prometeo` is its ability to simplify memory management without compromising efficiency. In fact, by enforcing a specific structure on the programs to be transpiled, it is possible to carry out a simple, but yet efficient static analysis that determines the worst-case heap usage. In this way the memory needed by the program can be allocated during a startup phase such that allocations and deallocations can be efficiently carried out on what we may call a preallocated *virtual heap*. The requirement on the program's structure is the one that no memory ever escapes a context with exception made for constructors. This entails that memory is always allocated by callers rather than callees. Although this is definitely a restrictive requirement in a general purpose programming language, we argue that in the context of embedded high-performance computing it is a reasonable compromise to be made. In particular, such a restriction allows one to carry out a static analysis of programs based on the associated call graphs. Loosely speaking, each scope is associated with a certain amount of memory which is independent of the value of mutable variables (i.e., branching is neglected) and we require instead that memory allocations only depend on special immutable *dimension* variables whose value can be easily determined at parse time. In this way, the worst-case heap usage

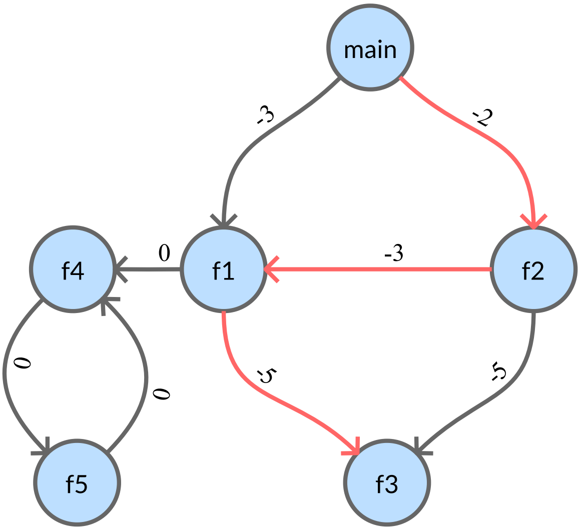


Figure 7.4: Call graph with memory-weighted edges. The shortest path from `main` to any node with the most negative cost determines the worst-case heap usage of the program. Only non negative cycles are allowed.

analysis boils down to solving a longest path problem on a graph whose nodes and edges represent scopes and their associated memory cost, respectively. Since the memory-weighted call graph is a directed graph which cannot have positive cycles (due to the program structure imposed by `prometeo`), we can transform the graph by flipping the sign of all weights and solve a simple shortest path problem using the Bellman-Ford [1] algorithm (Dijkstra's algorithm would require the weights in the transformed graph to be positive). Figure 7.4 shows the transformed graph associated with the simple program in Listing 7.11.

7.2 Usage and performance

In the following, we introduce `prometeo`'s syntax based on Python and present two numerical benchmark in which its performance is compared to the one of state-of-the-art alternatives for high-performance scientific computing.

---

[1] The Bellman-Ford algorithm has complexity $O(|V||E|)$, where V denotes the number of vertices of the graph and E denotes the number of its edges.

```
from prometeo import *

n :dims = 10

def f1() -> None:
    A : pmat = pmat(n,n)
    B : pmat = pmat(n,n)
    C : pmat = pmat(n,n)
    f3()
    f4()
    return

def f2() -> None:
    A : pmat = pmat(n,n)
    B : pmat = pmat(n,n)
    f1()
    f3()
    return

def f3() -> None:
    A : pmat = pmat(n,n)
    B : pmat = pmat(n,n)
    C : pmat = pmat(n,n)
    D : pmat = pmat(n,n)
    E : pmat = pmat(n,n)
    return

def f4() -> None:
    f5()
    return

def f5() -> None:
    f4()
    return

def main() -> int:
    f1()
    f2()
    return 0
```

Listing 7.11: Simple program illustrating `prometeo`'s static heap analysis.

7.2.1 Python restricted syntax

`prometeo` is an embedded domain specific language based on Python. Hence, its syntax is based on Python. Below we describe the details regarding the most common supported Python constructs that `prometeo` is able to transpile to C.

Variable declaration

A variable can be declared as follows

```
<var_name> : <type> = <value>
```

where `<var_name>` must be a valid identifier `<type>` must be a valid `prometeo` built-in type or a user-defined type and `<value>` must be a valid expression of type `<type>`.

Example:

```
a : int = 1
```

List declaration

A list can be declared using the `plist` constructor as follows

```
<list_name> : List = plist(<type>, <dim>)
```

Examples:

```
v : List = plist(int, n)
```

and

```
sizes : dimv = [[2,2], [2,2], [2,2], [2,2], [2,2]]
A : List = plist(pmat, sizes)
```

`if` statement

An `if` statement takes the form

```
if <cond>:
    ...
```

`for` loop

A `for` loop takes the form

```
for i in range([<start>], <end>)
    ...
```

where the optional parameter `<start>` must be an expression of type int (default value 0) and defines the starting value of the loop's index and `<end>` must be an expression of type `int` which defines its final value.

Function definition

Functions can be defined as follows

```
def <function_name> (<arg1> : <arg_1_type>, ...) -> <ret_type> :

    ...

    return <ret_value>
```

Class definition

`prometeo` supports basic classes of the following form

```
class <name>:
    def __init__(self, <arg1> : <type_1>, ...) -> None:
        self.<attr> : <type> = <value>
        ...

    def <method> (self, <arg1> : <type_1>, ...) -> <r_type>:
        ...

        return <ret_value>
```

`main` function

For consistency all main functions need to be defined as follows

```
def main() -> int:

    ...

return 0
```

pure Python blocks

In order to be able to use the full potential of the Python language and its vast pool of libraries, it is possible to write pure Python blocks that are run only when `prometeo` code is executed directly from the Python interpreter (when `-cgen` is set to `false`). In particular, any line that is enclosed within `# pure >` and `# pure <` will be run only by the Python interpreter, but completely discarded by `prometeo`'s parser.

```
# some prometeo code
A : pmat = pmat(n,n)
...

# pure >

# this is only run by the Python interpreter (--cgen=
    False)
# and will not be transpiled)

# some Python code

import numpy as np

M = np.array([[1.0, 2.0],[0.0, 0.5]])
print(np.linalg.eigvals(M))
...

# pure <

# some more prometeo code
for i in range(n):
    for j in range(n):
        A[i, j] = 1.0
...
```

7.2.2 BLAS and LAPACK API

A key aspect to user-friendliness of software for scientific computing is the availability of a simple, but yet expressive, interface to BLAS and LAPACK routines. The interface to such routines is based on the API of the software `BLASFEO` [Frison et al., 2020] and presents minor variations with respect to standard BLAS and LAPACK. Thanks to its static analysis engine, `prometeo` can combine the ease of use typical of APIs of high-level languages such as Python and the efficiency of high-performance C code. Below we report the main BLAS/LAPACK routines implemented in `prometeo` together with their API:

Level 2 BLAS

General matrix-vector multiplication (GEMV)

$$z \leftarrow \beta \cdot y + \alpha \cdot \mathrm{op}(A)x \tag{7.1}$$

```
pmt_gemv(A[.T], x, [y], z, [alpha=1.0], [beta=0.0])
```

Solve linear system with (lower or upper) triangular matrix coefficient (TRSV)

$$\mathrm{op}(A)\,x = b \tag{7.2}$$

```
pmt_trsv(A[.T], b, [lower=True])
```

Matrix-vector multiplication with (lower or upper) triangular matrix coefficient (TRMV)

$$z \leftarrow \mathrm{op}(A)\,x \tag{7.3}$$

```
pmt_trmv(A[.T], x, z, [lower=True])
```

LEVEL 3 BLAS

General matrix-matrix multiplication (GEMM)

$$D \leftarrow \beta \cdot C + \alpha \cdot \mathrm{op}(A)\,\mathrm{op}(B) \tag{7.4}$$

```
    pmt_gemm(A[.T], B[.T], [C], D, [alpha=1.0], [beta
        =0.0])
```

Symmetric rank k update (SYRK)

$$D \leftarrow \beta \cdot C + \alpha \cdot \mathrm{op}(A)\,\mathrm{op}(B) \tag{7.5}$$

with C and D lower triangular.

```
    pmt_syrk(A[.T], B[.T], [C], D, [alpha=1.0], [beta
        =0.0])
```

Triangular matrix-matrix multiplication (TRMM)

$$D \leftarrow \alpha \cdot B\,A^\top \tag{7.6}$$

with B upper triangular or

$$D \leftarrow \alpha \cdot A\,B \tag{7.7}$$

with A lower triangular.

```
    pmt_trmm(A[.T], B, D, [alpha=1.0], [beta=0.0])
```

LAPACK

Cholesky factorization (POTRF)

$$C = D\,D^\top \tag{7.8}$$

with D lower triangular and C symmetric and positive definite

```
    pmt_potrf(C, D)
```

LU factorization (GETRF)

$$C = L\,P\,U \tag{7.9}$$

```
    pmt_getr(C, D)
```

QR factorization (GEQRF)

$$C = Q\,R \tag{7.10}$$

```
pmt_geqrf(C, D)
```

7.2.3 Performance

Since `prometeo` programs transpile to pure C code that calls the high performance linear algebra library BLASFEO, execution time can be comparable to hand-written high-performance code. Next, we consider two benchmarks that highlight the computational benefits of using `prometeo`.

Riccati factorization

Figure 7.5 shows a comparison of the CPU time necessary to carry out a Riccati factorization using highly optimized hand-written C code with calls to BLASFEO and the ones obtained with `prometeo` transpiled code from this example. The computation times obtained with `NumPy` and Julia[2] and the hand-coded C implementation are added for comparison. Moreover, we include timings obtained with `Numpy` with the BLAS API of `BLASFEO` [Frison et al., 2020]. All the benchmarks have been run on a Dell XPS-9360 equipped with an i7-7560U CPU running at 2.30 GHz (to avoid frequency fluctuations due to thermal throttling).

For small matrix sizes, the CPU times obtained with `prometeo` are about an order of magnitude shorter than with `Numpy` and Julia. Moreover, for any size in the benchmark the computation times obtained with `prometeo` and with the hand-coded implementation are almost identical.

Fibonacci numbers

The performance highlighted in Figure 7.5 shows that Python programs transpiled with `prometeo` can achieve a performance substantially identical to the one of hand-written high-performance C code. In the following, we address the comparison with `Nuitka`, a state-of-the-art Python to C compiler, and

---

[2] notice however that these last two implementations of the Riccati factorization are not easily embeddable

```
from prometeo import *

nm:   dims = 10
nx:   dims = 2*nm
nu:   dims = nm
nxu:  dims = nx + nu
N:    dims = 5

def main() -> int:
    # number of repetitions for timing
    nrep : int = 10000

    Ac11 : pmat = pmat(nm,nm)
    for i in range(nm):
        Ac11[i,i] = 1.0

    # set up other matrices (similarly)
    ...

    # array-type Riccati factorization
    for i in range(nrep):
        pmt_potrf(Q, Lxx)
        M[nu:nu+nx,nu:nu+nx] = Lxx
        for i in range(1, N):
            pmt_trmm_rlnn(Lxx, BAt, w_nxu_nx)
            pmt_syrk_ln(w_nxu_nx, w_nxu_nx, RSQ, M)
            pmt_potrf(M, M)
            Lxx[0:nx,0:nx] = M[nu:nu+nx,nu:nu+nx]

    return 0
```

Listing 7.12: Riccati factorization implemented with `prometeo`.

```
from prometeo import *

def fib(n : int) -> int:
    a : int = 0
    b : int = 1
    c : int = 0
    for i in range(n):
        c = a + b
        a = b
        b = c
    return b


def main() -> int:

    res : int = 0

    for i in range(30):
        for j in range(1000000):
            res = fib(i)
    print('%i' %res)
    return 0
```

Listing 7.13: `prometeo` program that computes numbers in the Fibonacci series.

PyPy. To this end, we regard a substantially different benchmark in which the computationally intensive operations are not delegated to high-performance implementations (typically in C, Fortran or assembly) of BLAS and LAPACK routines. In particular, we will regard the simple program described by Listing 7.13 that computes numbers in the Fibonacci series. Table 7.1 shows the computation times obtained with the standard Python interpreter (CPython), `Nuitka`, PyPy and `prometeo`. The results show that `prometeo` is more than one order of magnitude faster than `Nuitka` and Python and about three times faster than PyPy.

7.3 Chapter summary and outlook

In this chapter, we introduced `prometeo`, a domain specific language and Python-to-C transpiler for embedded high-performance computing. `prometeo` is capable of translating programs written using a restricted subset of the Python language

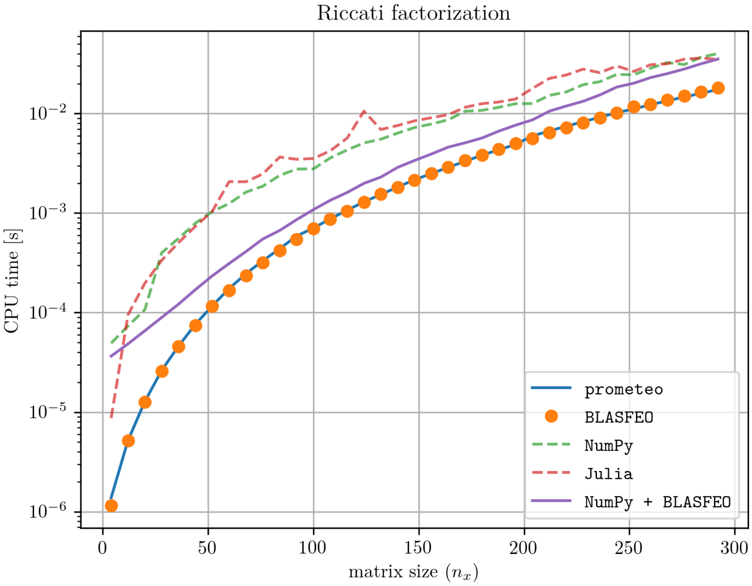


Figure 7.5: Computation time associated with a Riccati factorization for increasing matrix sizes. `prometeo`'s performance is almost identical to the one achieved with hand-coded C code. Although not embeddable, `Numpy` and `Julia` are included in the benchmark for reference.

parser/compiler	CPU time [s]
Python 3.7 (CPython)	11.787
`Nuitka`	10.039
PyPy 3.7	1.78
`prometeo`	0.657

Table 7.1: CPU time for the Fibonacci benchmark: `prometeo` is more than one order of magnitude faster than `Nuitka` and Python and about three times faster than PyPy.

into efficient and self-contained C programs. Unlike in most general purpose Python transpilers, the code emitted by `prometeo` does not depend on runtime libraries and it can be easily deployed on embedded hardware. Moreover, due to its tailored memory management system, it can outperform state-of-the-art alternatives while providing a sufficiently expressive language.

Ongoing research targets the improvement of the implementation of the features presented in this chapter as well as the development of new features. Among these, support for nonlinear functions and automatic differentiation through `CasADi` [Andersson et al., 2019] and interfacing with external C libraries are of primary interest.

Chapter 8

Conclusions and outlook

In this thesis we investigated methods for NMPC that, although lead to inexact feedback policies, preserve certain system theoretic and numerical properties, while reducing the computational burden associated with the solution of the underlying mathematical programs. The feedback policies obtained by these methods find their most intuitive interpretation in two distinct classes. The first is a class of methods that rely on feedback policies obtained from inexact solutions to optimal control problems with stabilizing properties. These are typically NMPC formulations for which we can easily construct a Lyapunov function using standard arguments. The second encompasses methods that make use of exact solutions to "perturbed" versions of standard optimal control problems. This qualitative classification, although to some extent artificial, can help us to distinguish the methods presented in the thesis and provides insights on how the derivation of stability guarantees can be carried out. In this respect, the real-time methods analyzed in Chapter 3 fit in the first class. In fact it is in general difficult to associate the obtained feedback policy with the optimal solution to a specific optimal control problem with certain stabilizing properties. Similarly, the progressive and partial tightening NMPC methods introduced in Chapter 5 together with the zero-order NMPC method proposed in Chapter 4 find their best interpretation within the second class. In both cases we can in fact more or less explicitly refer to the optimal solution to a perturbed optimal control problem in a useful way. Finally, as discussed in Chapters 4 and 5, it is often possible to regard inexact solutions to the perturbed problems, hence combining the two approaches, without jeopardizing the stabilizing properties of the obtained feedback policies. This is the case, e.g., for the real-time variants of the partial tightening and zero-order NMPC methods. Although not entirely surprising, the fact that stability properties can be preserved under specific modifications to the underlying formulations and numerical strategies opens the field to a number of interesting directions, which have been partially investigated in this thesis. Some of these methods, such as the SQP-based RTI, are already well established and constitute an extremely valuable tool in achieving real-time feasibility in many applications. This is the case, for example, for the continuous control set NMPC controller for reluctance synchronous machines presented in Chapter 6. It is in applications characterized by short sampling times and limited computational power that the potential of inexact methods becomes even more evident. Finally, the implementation of methods proposed in this thesis often relies on a nontrivial interaction between computational building blocks commonly present in standard algorithms for nonlinear programming for direct optimal control (e.g., QP solvers and integrators). Hence, their applicability hinges on the availability of a user-friendly framework for the implementation of numerical algorithms for embedded high-performance computing. This need has motivated the development of the paradigm described in Chapter 7 which `prometeo` is based on.

In the following we further elaborate on the conclusions and outlook of the present thesis, with special attention to the interaction between the different topics addressed.

Real-time methods

Methods inspired by the well established RTI strategy have been analyzed from an abstract point of view in Chapter 3 and specific algorithms have been proposed in Chapters 3, 4 and 5. In particular, we have shown how stability of the system-optimizer dynamics can be proved for a large number of algorithms, that need not be SQP-based (and not even optimization-based). The results of Chapter 3 both provide a proof of asymptotic stability for the classical RTI strategy in the inequality constrained setting and show that similar results can be obtained for similar strategies based on early termination. Notably, Theorem 3.2.28 applies to the real-time variants of the zero-order methods proposed in Chapter 4 and the partial tightening method proposed in Chapter 5 as well as to a real-time variant of the feasible SQP strategy proposed in Section 4.1.1.

Outlook. Interesting research directions could stem from relaxation of the assumptions made in Theorem 3.2.28. Among others, it would be interesting to remove the assumption of Lipschitz continuity and replace it with a weaker assumption, e.g., Hölder continuity, or regularity of set-valued solution maps, could constitute fruitful research directions. Analogously, it is not hard to envision extensions of the stability results to algorithms with sublinear convergence rates. Finally, since robustness issues have not been addressed yet, it would be interesting to further study properties such as input-to-state stability and robust constraint satisfaction in the presence of bounded disturbances. Although input-to-state stability could be easily obtained due to Lipschitz continuity of the system-optimizer dynamics, the details have not been worked out and this topic constitutes an interesting research direction.

Zero-order methods

Strategies based on the use of inexact fixed first- and second-order information that we call *zero-order* methods have been proposed and analyzed in Chapter 4. On the one hand, such methods can improve efficiency by avoiding computing derivatives and performing linear algebra operations otherwise necessary. On the other hand, they lead to feasible solutions that enjoy advantageous asymptotic suboptimality bounds. The properties of the solution the iterates converge to were used to prove local asymptotic stability of the resulting inexact feedback policy and, for the real-time variant of the algorithm, local asymptotic stability of the system-optimizer dynamics.

From an algorithmic point of view, in addition to the obvious advantage of not having to compute derivatives, solution of the QP subproblems can be largely sped up. We showed how factorizations of the KKT systems can be reused across iterations both for OCP structured and unstructured nonconvex programs. In the OCP structured case, it is possible to leverage a structure exploiting algebraic elimination in combination with a dense active-set QP solver (such as `qpOASES`) in order to avoid refactorization of the KKT system. Similarly, in the case of unstructured problems (or when dealing with large-scale OCP structured programs) it is possible to exploit a sparse active-set method based on a Schur complement strategy. Regardless of how the QPs are solved, solving a series of zero-order problems gives rise to a feasible SQP algorithm that we analyzed in Section 4.1.1.

Outlook. Although a proof-of-concept implementation of the feasible SQP solver using `qpOASES` and `CasADi`'s code-generation is available at `https://github.com/zanellia/feasible_sqp`, a thorough implementation and benchmarking is subject of undergoing research. Moreover, we believe that feasible SQP methods can be beneficially applied in the context of real-time optimal control. In fact feasibility of the intermediate iterates can be exploited in order to obtain cheap feedback policies or suboptimal path plans.

Progressive and partial tightening methods

In Chapter 5 we introduced *progressive tightening* NMPC methods with stage-varying costs and constraints. Using costs and constraints that are increasingly conservative the farther we look into the future, we extended the standard argumentation used for the stability proof of tracking NMPC. Based on this idea, we proposed a specific numerical strategy, that we called *partial tigthening*, that makes use of a barrier-based tightening of the constraints. The prediction horizon is split into two sections and, in the terminal section, the constraints are tightened in order to alleviate the numerical difficulties associated with the nonsmooth complementarity manifold. An iterative algorithm based on such formulation is proposed that can be classified as a generalized Newton-type method. In the resulting subproblems, the variables associated with the terminal section of the prediction horizon can be efficiently eliminated. In this way, a reduced linear-quadratic constrained OCP with shorter horizon needs to be solved leading to computational savings. A proof-of-concept implementation of partial tightening NMPC has been developed within `acados` using the building blocks of the Riccati-based interior-point QP solver `HPMPC` [Frison et al., 2014]. The implementation has been deployed on the embedded control unit of a quadcopter and experimentally validated.

Outlook. An implementation of partial tightening within `acados` using the successor of `HPMPC`, namely the QP solver `HPIPM` [Frison et al., 2020], is currently under consideration. Moreover, variants of partial and progressive tightening that also exploit concepts from zero-order methods could lead to interesting algorithms that can further speed up computations.

Continuous control set NMPC of electrical drives

Although MPC for electrical drives is an increasingly adopted control strategy, most of its applications rely on the so-called direct, or finite-set control, approach. A mixed-integer optimal control problem is formulated in which the switches' positions are regarded as input to the system. As stated in [Geyer, 2016], a much more unexplored approach is the one of indirect, or continuous control set, MPC (CS-MPC). In this case one assumes that the inputs to the system, namely the voltages applied to the windings, can take values in real-valued intervals. This second approach seems to have found much less widespread application due to the challenge of solving continuous optimization problems within the short time typically available between two successive sampling instants.

In Chapter 6, we presented the experimental validation of a CS-NMPC controller based on the RTI strategy and implemented with `acados`. The resulting controller was deployed on a `dSPACE` DS1007 unit and experimentally tested with a sampling time of 250 µs. Although the optimal control problem is discretized in only two shooting nodes, using a simple *oversampling* strategy, it is possible to predict about 10 sampling times ahead.

The simulation and experimental results showed that the proposed CS-NMPC controller can largely outperform state-of-the-art gain-scheduled field-oriented PI controllers, especially when the constraints become active. This situation occurs systematically when operating the machine close to, or slightly above, the nominal velocity and thus makes CS-NMPC a valid alternative to field-oriented PI controllers.

Outlook. Although it was possible to run the proposed solver based on `acados` on a `dSPACE` DS1007 unit, the computational burden associated with CS-NMPC is still rather high. In order to reduce it, it seems reasonable to explore algorithms similar to the real-time ADMM algorithm applied (in simulation only) to a permanent magnet synchronous machine in Section 3.1. In the context of nonconvex programming, such an algorithm could be based, e.g., on the truncated SQP method proposed in Section 3.1. Finally, an extensive simulation study and experimental validation of the proposed real-time ADMM algorithm for permanent magnet synchronous machines is definitely an interesting research direction.

Domain specific languages for embedded high-performance computing

The availability of user-friendly, maintainable and extensible software implementations of algorithms for numerical optimization is of fundamental importance for the successful deployment of NMPC on embedded hardware. However, combining computational efficiency and features of high-level programming languages is a challenging task. Most implementations of software for embedded optimization are nowadays carried out in low-level languages, such as C, resulting in a tedious and error prone development workflow. Although this difficulty does not necessarily impact user-friendliness, it definitely impacts the amount of work required by such implementations.

Based on the fact that the task of implementing software for embedded optimization often involves error prone, but repetitive, patterns, we presented in Chapter 7 the domain specific language `prometeo`. Thanks to its Python-to-C transpiler and its static analysis tools, `prometeo` aims at simplifying the development workflow for embedded optimization and high-performance computing in general.

Outlook. The implementation of `prometeo` is still at an early stage and will require further work. Undergoing research targets the improvement of the implementation of the features presented in Chapter 7 as well as the development of new features. Among these, support for nonlinear functions and automatic differentiation through `CasADi` [Andersson et al., 2019] and interfacing with external C libraries are of primary interest.

Appendix A

Appendix

A.1 Proof of Proposition 3.2.16

In the following, we will use the shorthand $u = M_{u,z}z$. First, notice that, since

$$\psi(t; x, u) = x + \int_0^t \phi(\psi(\tau; x, u), u)\,\mathrm{d}\tau, \tag{A.1}$$

for all $x \in X_{\bar{V}}$ and all $z \in \mathcal{B}(\bar{z}(x), r_z)$, there exists a $T' > 0$, such that, for all $t \leq T'$, we have $\psi(t; x, u) \in X_{\bar{V}+\rho}$.

Using the Gronwall Lemma and Assumption 3.2.16, we obtain that, for any $x \in X_{\bar{V}}$, all $z \in \mathcal{B}(\bar{z}(x), r_z)$ and all $T \leq T'$, the following holds:

$$\begin{aligned}\|\psi(T; x, u) - x\| &\leq \int_0^T \|\phi(\psi(\tau; x, u), u)\|\,\mathrm{d}\tau \\ &\leq \int_0^T L_{\phi,x}\|\psi(\tau; x, u) + x - x\|\,\mathrm{d}\tau \\ &\quad + \int_0^T L_{\phi,u}\|u\|\,\mathrm{d}\tau\end{aligned} \tag{A.2}$$

such that

$$\begin{aligned}\|\psi(T;x,u)-x\| &\leq T(L_{\phi,u}\|u\|+L_{\phi,x}\|x\|)\\ &\quad+\int_0^T L_{\phi,x}\|\psi(\tau;x,u)-x\|\,\mathrm{d}\tau \\ &\leq Te^{L_{\phi,x}T}(L_{\phi,u}\|u\|+L_{\phi,x}\|x\|)\end{aligned} \tag{A.3}$$

and we can define $L_{\psi,x} := e^{L_{\phi,x}T}L_{\phi,x}$ and $L_{\psi,u} := e^{L_{\phi,x}T}L_{\phi,u}$, such that

$$\|\psi(T;x,u)-x\| \leq T\cdot(L_{\psi,x}\|x\|+L_{\psi,u}\|u\|) \tag{A.4}$$

for any $x\in X_{\bar V}$ and all $z\in\mathcal{B}(\bar z(x),r_z)$. Similarly, in order to prove the second inequality, we first notice that there must exist a $T''>0$ such that, for all $x\in X_{\bar V}$, for all $u=M_{u,z}z$ and all $u'=M_{u,z}z'$ such that $z,z'\in\mathcal{B}(\bar z(x),r_z)$ and for all $t\leq T''$, we have $\psi(t;x,u),\psi(t;x,u')\in X_{\bar V+\rho}$. Hence, for any $x\in X_{\bar V}$ and all $z,z'\in\mathcal{B}(\bar z(x),r_z)$ and all $T\leq T''$ we can proceed as follows:

$$\begin{aligned}&\|\psi(T;x,u')-\psi(T;x,u)\|\\ &\leq\int_0^T\|\phi(\psi(\tau;x,u'),u')-\phi(\psi(\tau;x,u),u)\|\,\mathrm{d}\tau\\ &=\int_0^T\|\phi(\psi(\tau;x,u'),u')-\phi(\psi(\tau;x,u),u)\\ &\quad+\phi(\psi(\tau;x,u'),u)-\phi(\psi(\tau;x,u'),u)\|\,\mathrm{d}\tau\\ &=\int_0^T\|\phi(\psi(\tau;x,u'),u')-\phi(\psi(\tau;x,u'),u)\|\mathrm{d}\tau\\ &\quad+\int_0^T\|\phi(\psi(\tau;x,u),u)-\phi(\psi(\tau;x,u'),u)\|\,\mathrm{d}\tau\\ &\leq TL_{\phi,u}\|u'-u\|\\ &\quad+\int_0^T L_{\phi,x}\|\psi(\tau;x,u')-\psi(\tau;x,u)\|\,\mathrm{d}\tau\end{aligned}$$

and, applying the Gronwall Lemma, we obtain

$$\begin{aligned}\|\psi(T;x,u')-\psi(T;x,u)\| &\leq TL_{\phi,u}e^{TL_{\phi,x}}\|u'-u\|\\ &=TL_{\psi,u}\|u'-u\|.\end{aligned}$$

Finally, setting $T_1 := \min\{T',T''\}$ concludes the proof. □

A.2 Proof of Theorem 3.2.25

It suffices to show that $V_l(w)$ is a Lyapunov function for (3.84) in $\mathbb{R}^n_{\geq 0}$. Define $\hat{w}_{\min} := \min_{i=1,\dots,n} [\,\hat{w}\,]_i$ and $\hat{w}_{\max} := \max_{i=1,\dots,n} [\,\hat{w}\,]_i$. The following inequalities hold:

$$\hat{w}^\top w \geq \hat{w}_{\min} \cdot \mathbf{1}^\top w = \hat{w}_{\min} \cdot \|w\|_1 \geq \hat{w}_{\min} \cdot \|w\|_2$$

and

$$\begin{aligned} \hat{w}^\top w &\leq \hat{w}_{\max} \cdot \mathbf{1}^\top w \\ &\leq \hat{w}_{\max} \cdot \|\mathbf{1}\|_2 \|w\|_2 \\ &\leq \sqrt{n}\, \hat{w}_{\max} \cdot \|w\|_2 \end{aligned}$$

which shows that there exists $\mathcal{K}_\infty$ functions $\alpha_{l,1}(\|w\|) := \hat{w}_{\min} \cdot \|w\|$ and $\alpha_{l,2}(\|w\|) := \sqrt{n}\, \hat{w}_{\max} \cdot \|w\|$ such that

$$\alpha_{l,1}(\|w\|) \leq V_l(w) \leq \alpha_{l,2}(\|w\|). \tag{A.5}$$

Moreover, for any $w > 0$, we have that

$$\begin{aligned} V_l(w_+) - V_l(w) &= \hat{w}^\top A w - \hat{w}^\top w \\ &= \hat{w}^\top (A - \mathbb{I})\, w \\ &= w^\top (A^\top - \mathbb{I})\, \hat{w} \\ &\leq -\hat{d} \cdot \|w\|_1 \leq -\hat{d} \cdot \|w\|_2. \end{aligned} \tag{A.6}$$

Hence, there exists a positive definite and continuous function $\alpha_{l,3}(\|w\|) := \hat{d} \cdot \|w\|$ such that, for any $w \geq 0$, the following holds

$$V_l(w_+) - V_l(w) \leq -\alpha_{l,3}(\|w\|) \tag{A.7}$$

and $\alpha_{l,3}(0) = 0$, which concludes the proof. □

Remark A.2.1. *Notice that the original Theorem in [Kaczorek, 2008] requires the existence of a strictly positive vector $\hat{w} > 0$ such that*

$$(A - \mathbb{I})\hat{w} < 0. \tag{A.8}$$

Although the condition used in Theorem 3.2.26 is equivalent to one above, the resulting $\hat{w}$ can only be used to define a Lyapunov function for the dual system $w_+ = A^\top w$.

A.3 Lyapunov function for the system-optimizer dynamics in error form

In the following, we report a simplified version of the arguments presented in Section 3.2. In particular, we construct a Lyapunov function for the system-optimizer dynamics in error form defined as follows.

Definition A.3.1 (System-optimizer dynamics in error form)**.** *We will refer to the following discrete-time system as system-optimizer dynamics in error form:*

$$\begin{aligned} x_+ &= \psi(T; x, M_{u,z}e + \bar{z}(x)), \\ e_+ &= \varphi(\psi(T; x, M_{u,z}z), e + \bar{z}(x)) - \bar{z}(\psi(T; x, M_{u,z}z)), \end{aligned} \tag{A.9}$$

where we have introduced $e := z - \bar{z}(x)$*. Let* $\hat{\xi} := (x, e)$*. We will use*

$$\hat{\xi}_+ = \hat{\Phi}(T; \hat{\xi}), \tag{A.10}$$

to refer to (3.48) *in compact form.*

Corollary A.3.2. *Let Assumptions 3.2.2, 3.2.4, 3.2.8, 3.2.11, 3.2.12 and 3.2.16 hold. Then the origin is an asymptotically stable equilibrium with region of attraction* Σ *for the system-optimizer dynamics* (3.49)*. In particular, the function*

$$V_{\mathrm{so}}(\hat{\xi}) := \hat{w}^\top \begin{bmatrix} V(x)^{\frac{1}{q}} \\ \|e\| \end{bmatrix} \tag{A.11}$$

is a Lyapunov function in Σ *for the system* (3.49) *and the origin* $(x, e) = \hat{\xi} = 0$*.*

Proof. We can derive an upper bound for $V_{\mathrm{so}}(\hat{\xi})$ as follows.

$$\begin{aligned} V_{\mathrm{so}}(\hat{\xi}) &= V(x)^{\frac{1}{q}} + \beta\|e\| \leq a_2^{\frac{1}{q}}\|x\| + \beta\|e\| \\ &\leq \underbrace{\max\{a_2^{\frac{1}{q}}, \beta\}}_{=:\tilde{w}_2} \cdot (\|x\| + \|e\|) \\ &\leq \tilde{w}_2 \cdot (\|x\|_1 + \|e\|_1) = \tilde{w}_2 \cdot \|\hat{\xi}\|_1 \\ &\leq \tilde{w}_2\sqrt{n_x + n_z} \cdot \|\hat{\xi}\|. \end{aligned}$$

Similarly, a lower bound can be obtained as

$$\begin{aligned}
V_{\mathrm{so}}(\hat{\xi}) &= V(x)^{\frac{1}{q}} + \beta\|e\| \geq a_1^{\frac{1}{q}}\|x\| + \beta\|e\| \\
&\geq \frac{a_1^{\frac{1}{q}}}{\sqrt{n_x}}\|x\|_1 + \frac{\beta}{\sqrt{n_z}}\|e\|_1 \\
&\geq \underbrace{\min\left\{\frac{a_1^{\frac{1}{q}}}{\sqrt{n_x}}, \frac{\beta}{\sqrt{n_z}}\right\}}_{=:\tilde{w}_1} \cdot (\|x\|_1 + \|e\|_1) \\
&\geq \tilde{w}_1 \cdot \|\hat{\xi}\|.
\end{aligned}$$

Finally, the Lyapunov decrease can be derived as follows.

$$\begin{aligned}
V_{\mathrm{so}}(\hat{\xi}_+) &= V(x_+)^{\frac{1}{q}} + \beta\|e_+\| \\
&\leq \nu_+ + \beta\epsilon_+ \leq \nu + \epsilon - \hat{d} \cdot \left\|\begin{matrix}\nu \\ \epsilon\end{matrix}\right\| \\
&= V(x)^{\frac{1}{q}} + \beta E - \hat{d} \cdot \left\|\begin{matrix}\nu \\ \epsilon\end{matrix}\right\| \\
&= V_{\mathrm{so}}(\hat{\xi}) - \hat{d} \cdot \left\|\begin{matrix}v \\ E\end{matrix}\right\| \\
&\leq V_{\mathrm{so}}(\hat{\xi}) - \hat{d} \cdot \sqrt{(a_1^{\frac{1}{q}}\|x\|)^2 + \|e\|^2} \\
&\leq V_{\mathrm{so}}(\hat{\xi}) - \underbrace{\hat{d} \cdot \min\{a_1^{\frac{1}{q}}, 1\}}_{=:\tilde{w}_3} \sqrt{\|x\|^2 + \|e\|^2} \\
&\leq V_{\mathrm{so}}(\hat{\xi}) - \tilde{w}_3\|\hat{\xi}\|.
\end{aligned}$$

Hence we can define the $\mathcal{K}_\infty$ functions $\alpha_{\mathrm{so},1}(\|\hat{\xi}\|) := \tilde{w}_1 \cdot \|\hat{\xi}\|$ and $\alpha_{\mathrm{so},2}(\|\hat{\xi}\|) := \tilde{w}_2 \cdot \|\hat{\xi}\|$ and the positive definite function $\alpha_{\mathrm{so},3}(\|\hat{\xi}\|) := \tilde{w}_3 \cdot \|\hat{\xi}\|$, such that

$$\begin{aligned}
\alpha_{\mathrm{so},1}(\|\hat{\xi}\|) \leq V_{\mathrm{so}}(\hat{\xi}) &\leq \alpha_{\mathrm{so},2}(\|\hat{\xi}\|) \\
V_{\mathrm{so}}(\hat{\xi}_+) - V_{\mathrm{so}}(\hat{\xi}) &\leq -\alpha_{\mathrm{so},3}(\|\hat{\xi}\|),
\end{aligned} \tag{A.12}$$

i.e. $V_{\mathrm{so}}(\hat{\xi})$ is a Lyapunov function in Σ for the system-optimizer dynamics $\hat{\xi}_+ = \hat{\Phi}(T;\hat{\xi})$ and the equilibrium $\hat{\xi} = 0$. ☐

A.4 Asymptotic stability of the system-optimizer dynamics for equality constrained NMPC

Theorem 3.2.28 provides a Lyapunov function for the system-optimizer dynamics under general assumptions. In the following, based on [Zanelli et al., 2021b], we present a tailored result that can be applied in the context of equality constrained NMPC. In particular, we will make use of the following assumption.

Assumption A.4.1 (Second-order growth). *Assume that, for any $u \in \mathbb{R}^{n_u}$, for any $x \in X_{\bar{V}}$ and any $T \leq T_0$, the following holds:*

$$V(\psi(T;\, x, u)) - V(x) \leq -T \cdot a_3 \|x\|^2 + T \cdot O(\|u - \bar{u}(x)\|^2). \tag{A.13}$$

Remark A.4.2. *Assumption A.4.1 can be informally justified by analyzing the properties of an underlying continuous-time Lyapunov function V_c and using an argument similar to the one used in Assumption 2.18 in [Diehl et al., 2007] in a discrete-time setting. In particular, under suitable differentiability assumptions, for any $\delta > 0$, with a slight abuse of notation, we can introduce $u_\delta : \mathbb{R} \to \mathbb{R}^{n_u}$ with $u_\delta \in \mathcal{L}_2$ and write*

$$\begin{aligned} V_\mathrm{c}(x) &= \min_{u_\delta} \left\{ \int_0^\delta l(\psi(\tau, x, u_\delta(\tau)), u_\delta(\tau))\mathrm{d}\tau + \tilde{V}_\mathrm{c}(\psi(\delta, x, u_\delta)) \right\} \\ &= \tilde{V}_\mathrm{c}(\psi(\delta, x, u_\delta)) + \int_0^\delta l(\psi(\tau, x, u_\delta(\tau)), u_\delta(\tau))\mathrm{d}\tau \\ &\quad + O(\|u_\delta - \bar{u}_\delta(x)\|^2_{\mathcal{L}_2}), \end{aligned}$$

where $\tilde{V}_\mathrm{c}$ is the optimal value function for a problem with shrunk horizon $T_\mathrm{f} - \delta$ and $\bar{u}_\delta : \mathbb{R} \to \mathbb{R}^{n_u}$ with $\bar{u}_\delta \in \mathcal{L}_2$. Using the fact that $V_\mathrm{c}(x) \leq \tilde{V}_\mathrm{c}(x)$ and a quadratic lower bound on l we can conclude

$$\begin{aligned} V_\mathrm{c}(\psi(\delta, x, u_\delta)) &\leq \tilde{V}_\mathrm{c}(\psi(\delta, x, u_\delta)) \\ &\leq V_\mathrm{c}(x) - \int_0^\delta l(\psi(\tau, x, u_\delta(\tau)), u_\delta(\tau))\mathrm{d}\tau + O(\|u_\delta - \bar{u}_\delta(x)\|^2_{\mathcal{L}_2}) \\ &\leq V_\mathrm{c}(x) - \delta \cdot \tilde{a}_3 \|x\|^2 + O(\|u_\delta - \bar{u}_\delta(x)\|^2_{\mathcal{L}_2}). \end{aligned}$$

This last inequality, together with the fact that, for a piece-wise constant parametrization, $\|u_\delta - \bar{u}_\delta(x)\|^2_{\mathcal{L}_2} = \delta \cdot \|u - \bar{u}(x)\|$, justifies Assumption A.4.1, if V is a "sufficiently" good approximation of V_c in the sense of Remark 3.2.3.

Proposition A.4.3. *Let Assumptions 3.2.2, 3.2.8, 3.2.11 and A.4.1 hold. Then, there exist finite positive constants* $\mu, \bar{V}_q \leq \bar{V}$ *and* $r_q \leq r_z$*, such that, for any* $E \leq r_q$ *and any* x *in* $X_{\bar{V}_q}$*, where* $X_{\bar{V}_q} := \{x : V(x) \leq \bar{V}_q\}$*, the following holds:*

$$V_+ \leq (1 - T\bar{a})V + T\mu E^2, \tag{A.14}$$

with $\bar{a} := \frac{a_3}{a_2}$*.*

Proof. Assumption A.4.1 implies that there must exist strictly positive constants $\mu, \bar{V}_q \leq \bar{V}$ and $r_q \leq r_z$ such that the following holds

$$\begin{aligned} V(\psi(x, M_{u,z}z)) &\leq V(x) - Ta_3\|x\|^2 - T\mu E^2 \\ &\leq V(x) - T\frac{a_3}{a_2}V(x) - T\mu E^2 \\ &= (1 - T\bar{a})V(x) - T\mu E^2, \end{aligned} \tag{A.15}$$

for any $E \leq r_q$, any $x \in X_{\bar{V}_q}$ and any $T \leq T_2$. □

Definition A.4.4. *Define the following set:*

$$\Sigma := \{(x, z) : V(x) \leq \bar{V}_q, \|z - \bar{z}(x)\| \leq \tilde{r}_q\}, \tag{A.16}$$

where

$$\tilde{r}_q := \min\left\{r_q, \sqrt{\frac{\bar{a}\bar{V}_q}{\mu}}\right\}. \tag{A.17}$$

The following theorem shows positive invariance of Σ.

Lemma A.4.5 (Invariance of Σ)**.** *Let Assumptions 3.2.2, 3.2.8, 3.2.11, 3.2.12, 3.2.16 and A.4.1 hold. Define*

$$\gamma := \sigma\hat{\kappa}\eta \quad \text{and } T_3' := \frac{(1-\kappa)\tilde{r}_q\sqrt{a_1}}{\sqrt{\bar{V}}\gamma}. \tag{A.18}$$

Then, for any $(x, z) \in \Sigma$ *and any* $T \leq T_3 := \min\{T_3', T_2\}$*, it holds that* $(x_+, z_+) \in \Sigma$*. Moreover, the following coupled system-optimizer contractions hold:*

$$\begin{aligned} V_+ &\leq (1 - T\bar{a})V + T\mu E^2, \\ E_+ &\leq T\hat{\gamma}V^{\frac{1}{2}} + \kappa E, \end{aligned} \tag{A.19}$$

where $\hat{\gamma} = \frac{\gamma}{\sqrt{a_1}}$*.*

Proof. Given that $E \leq \tilde{r}_q \leq r_q$ and $x \in X_{\bar{V}_q}$, we can apply the inequality from Proposition A.4.3, such that

$$V_+ \leq (1 - T\bar{a})V + T\mu E^2, \tag{A.20}$$

holds. Moreover, due to the definition of $\tilde{r}_q$, $V_+ \leq \bar{V}_q$ holds, which implies that x_+ is in $X_{\bar{V}_q}$. Similarly, due to the fact that $E \leq \tilde{r}_q \leq r_z$ and $x \in X_{\bar{V}_q} \subseteq X_{\bar{V}}$, we can apply the result from Proposition 3.2.18, which shows that

$$E_+ \leq \ \kappa E + T\gamma\|x\| \quad \text{and} \quad E_+ \leq r_z \tag{A.21}$$

must hold. Using Assumption 3.2.2 in Equation (A.21), we obtain

$$E_+ \leq \ \kappa E + T\hat{\gamma}V^{\frac{1}{2}}. \tag{A.22}$$

Moreover, due to (A.18), $E_+ \leq \tilde{r}_q$ holds. □

Lemma A.4.5 shows that we can guarantee that the state of the combined system-optimizer dynamics (x, z) will not leave Σ under the assumption that the sampling time T is short enough. Moreover, due to subadditivity of the square root, the following holds:

$$V_+^{\frac{1}{2}} \leq (1 - T\bar{a})^{\frac{1}{2}} V^{\frac{1}{2}} + (T\mu)^{\frac{1}{2}} E \tag{A.23}$$

such that we can regard the following simpler dynamics:

Definition A.4.6 (Auxiliary dynamics)**.** *Regard the linear discrete-time positive dynamical system*

$$\begin{aligned} \nu_+ &= (1 - T\bar{a})^{\frac{1}{2}} \nu + (T\mu)^{\frac{1}{2}} \epsilon, \\ \epsilon_+ &= T\hat{\gamma}\nu + \kappa\epsilon \end{aligned} \tag{A.24}$$

with states $\nu, \epsilon \in \mathbb{R}$ or, in compact form, In the following, we exploit properties of positive systems in order to construct an explicit linear Lyapunov function for the auxiliary dynamics which can be rewritten in the compact form

$$w_+ = A_a w, \tag{A.25}$$

where

$$A_a := \begin{bmatrix} (1 - T\bar{a})^{\frac{1}{2}} & T\hat{\mu} \\ T\hat{\gamma} & \kappa \end{bmatrix}, \tag{A.26}$$

and $w := (\nu, \epsilon)$. We will refer to (A.25) *as auxiliary dynamics.*

Remark A.4.7. *Notice that the considerations made by [Diehl et al., 2007], in a similar setting, lead to the same type of coupled contraction from Lemma A.4.5. An attractivity proof that implicitly uses auxiliary dynamics that would be obtained directly from* (A.19) *is derived in [Diehl et al., 2007]. However, in that case, due to the fact that the auxiliary system-optimizer dynamics are not Lipschitz at* $(0,0)$*, it would not be possible to prove stability with standard linear analysis tools.*

Due to linearity and positivity of the auxiliary dynamics (A.24), we can study asymptotic stability as follows.

Theorem A.4.8. *The positive discrete-time linear system* (A.25) *is asymptotically stable if and only if the following condition is satisfied:*

$$T^{\frac{3}{2}}\sqrt{\mu}\hat{\gamma} - (1-\kappa)(1-(1-T\bar{a})^{\frac{1}{2}}) < 0, \tag{A.27}$$

which is satisfied for any sufficiently small sampling time $T \leq T_5 := \frac{\beta^2(1-\kappa)^2}{\mu}$*. Moreover, the function* $V_l(w) := \hat{w}^\top w$*, where*

$$\hat{w} = \begin{bmatrix} 1 \\ \beta \end{bmatrix}, \quad \text{with} \quad \beta := \frac{1}{4}\frac{\bar{a}}{\hat{\gamma}}, \tag{A.28}$$

is a Lyapunov function for (A.25) *in* $\mathbb{R}^2_{\geq 0}$*.*

Proof. The proof follows the same arguments used in the proof of Theorem 3.2.27, but based on the auxiliary system (A.24). A sufficient and necessary condition for asymptotic stability is that the eigenvalues of the matrix

$$A = \begin{bmatrix} (1-T\bar{a})^{\frac{1}{2}} & (T\mu)^{\frac{1}{2}} \\ T\hat{\gamma} & \kappa \end{bmatrix}, \tag{A.29}$$

are smaller than one in absolute value. □

Following an argumentation similar to the one in the proof Theorem 3.2.28 we can prove the following result.

Theorem A.4.9 (Asymptotic stability)**.** *Let Assumptions 3.2.2, 3.2.8, 3.2.11, 3.2.12, 3.2.16 and A.4.1 hold. Then, for any* $T \leq \min\{T_4, T_5\}$*, the origin is an asymptotically stable equilibrium with region of attraction* Σ *for the combined system-optimizer dynamics* (3.49)*. In particular, the function*

$$V_{\mathrm{so}}(\xi) := \hat{w}^\top \begin{bmatrix} V(x)^{\frac{1}{2}} \\ \|z - \bar{z}(x)\| \end{bmatrix}, \tag{A.30}$$

where $\hat{w}$ *is defined according to Theorem A.4.8, is a Lyapunov function in* Σ *for the system* (3.49) *and the origin* $(x,z) = \xi = 0$*.*

Illustrative example

In the following, although the results derived apply to a much more general class of problems (twice-continuously nonlinear dynamics and cost), we discuss an illustrative numerical example where we exploit a simplified setting in order to be able to explicitly compute all the constants used in the assumptions of Theorem A.4.9. In particular, we regard the following unconstrained, linear-quadratic optimal control problem:

$$\begin{aligned} \min_{s(\cdot),u(\cdot)} \quad & \int_0^\infty \begin{bmatrix} s(t) \\ u(t) \end{bmatrix}^\top \begin{bmatrix} Q_\mathrm{c} & 0 \\ 0 & R_\mathrm{c} \end{bmatrix} \begin{bmatrix} s(t) \\ u(t) \end{bmatrix} \\ \text{s.t.} \quad & s(0) - x = 0, \\ & \dot{s}(t) = A_\mathrm{c} s(t) + B_\mathrm{c} u(t), \quad t \in [0, \infty], \end{aligned} \tag{A.31}$$

where the continuous-time dynamics are defined by

$$A_\mathrm{c} := \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B_\mathrm{c} := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{A.32}$$

and the matrices

$$Q_\mathrm{c} := \mathbb{I}_2 \quad \text{and} \quad R_\mathrm{c} := 1 \tag{A.33}$$

define the cost.

Problem (A.34) is discretized using multiple shooting with a single shooting node and a fixed discretization time $T_\mathrm{d} = 0.1s$ as follows:

$$\begin{aligned} \min_{s_0, s_1, u_0} \quad & T_\mathrm{d} \begin{bmatrix} s_0 \\ u_0 \end{bmatrix}^\top \begin{bmatrix} Q_\mathrm{c} & 0 \\ 0 & R_\mathrm{c} \end{bmatrix} \begin{bmatrix} s_0 \\ u_0 \end{bmatrix} + s_1^\top P s_1 \\ \text{s.t.} \quad & s_0 - x = 0, \\ & s_1 = A_{T_\mathrm{d}} s_0 + B_{T_\mathrm{d}} u_0. \end{aligned} \tag{A.34}$$

Here the discrete-time dynamics are obtained using an exact discretization with piece-wise constant parametrization of the control trajectories:

$$A_{T_\mathrm{d}} := \exp(A_\mathrm{c} T_\mathrm{d}), \; B_{T_\mathrm{d}} := \left(\int_0^{T_\mathrm{d}} \exp(A_\mathrm{c} \tau) \mathrm{d}\tau \right) B_\mathrm{c}. \tag{A.35}$$

Finally, the symmetric positive-definite matrix P that defines the terminal cost for the discretized problem is computed by solving the discrete-time algebraic Riccati equation

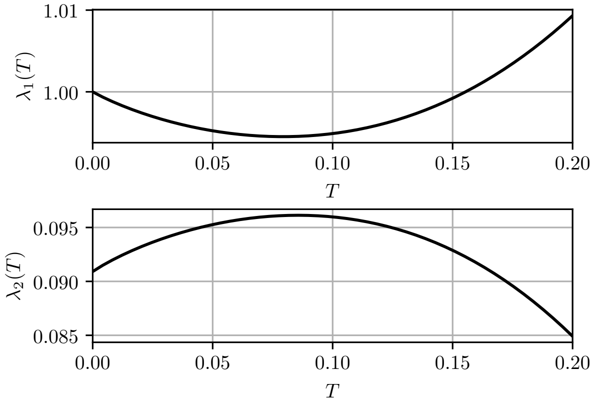


Figure A.1: Eigenvalues as a function of the sampling time T for problem (A.34). For sufficiently short sampling times, the auxiliary system in Definition A.4.6 is asymptotically stable.

$$P = A_{T_\mathrm{d}}^\top P A_{T_\mathrm{d}} - (A_{T_\mathrm{d}}^\top P B_{T_\mathrm{d}})(R + B_{T_\mathrm{d}}^\top P B_{T_\mathrm{d}})^{-1}(B_{T_\mathrm{d}}^\top P A_{T_\mathrm{d}}) + Q, \tag{A.36}$$

where $R := T_\mathrm{d} R_\mathrm{c}$ and $Q := T_\mathrm{d} Q_\mathrm{c}$. After elimination of s_0, the first-order optimality conditions of problem (A.34) read

$$Hu_0 + Gx = 0, \tag{A.37}$$

with

$$H := (T_\mathrm{d} R + B_{T_\mathrm{d}}^\top P B_{T_\mathrm{d}}), \quad G := B_{T_\mathrm{d}}^\top P A_{T_\mathrm{d}}. \tag{A.38}$$

We solve (A.37) with the following real-time gradient descent method:

$$u_{0,+} = -\tilde{H}^{-1}\left((H - \tilde{H})u_0 + Gx_+\right), \tag{A.39}$$

where $\tilde{H} = \rho\mathbb{I}$ for some positive constant $\rho > 1$.

Remark A.4.10. *Recall that the example under consideration has purely illustrative purpose and does not try to draw connections with numerical algorithms that would be used in practice. In fact here the optimization problem is a one dimensional unconstrained parametric quadratic program which can be solved in closed form. However, due to its simplicity, we will be able to better analyze the behavior of the otherwise far from trivial system-optimizer dynamics.*

Using standard arguments from convergence theory for Newton-type methods (see e.g [Diehl, 2016]), it is easy to show, that, for a fixed value of the parameter x, the following contraction estimate holds:

$$\|u_{0,+} - \bar{u}_0(x)\| \leq \hat{\kappa}\|u_0 - \bar{u}_0(x)\|, \tag{A.40}$$

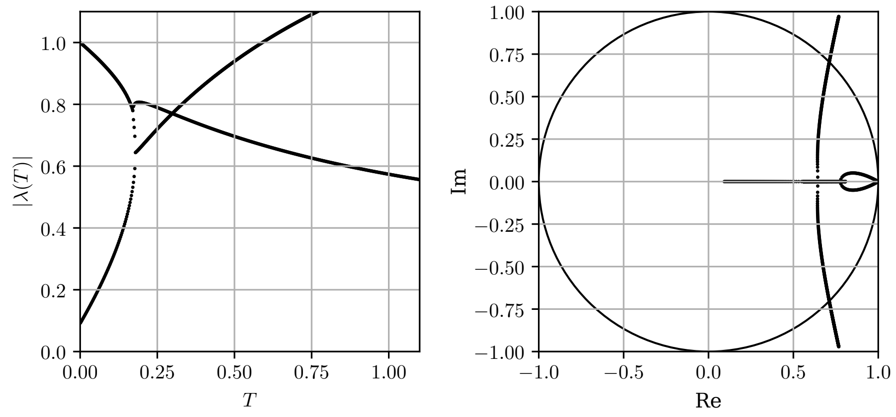


Figure A.2: Modulus of the eigenvalues associated with the system-optimizer dynamics and their trajectories in the complex plane. The plot reveals some conservatism of the analysis based on the auxiliary dynamics. In fact, the largest T for which the system-optimizer dynamics are stable is $T \approx 0.6$ as opposed to $T \approx 0.15$ obtained from Figure A.1.

with

$$\hat{\kappa} := \|\tilde{H}^{-1}(H - \tilde{H})\|. \tag{A.41}$$

Since $V(x) = x^\top Px$ and $\bar{u}_0(x) = -H^{-1}Gx$, we can compute exactly the constant $\sigma = \|H^{-1}G\|$ and approximately the constant $\mu \approx 0.5R$. Let

$$A_T := \exp(A_c T),\ B_T := \left(\int_{\tau=0}^{T} \exp(A_c\tau)\mathrm{d}\tau\right) B_c \tag{A.42}$$

describe the dynamics discretized according to the sampling time T. Given that

$$x_+ = A_T x + B_T u = x + \frac{T}{T}((A_T - \mathbb{I})x + B_T u), \tag{A.43}$$

we can compute the constants

$$L_{\psi,x} = \frac{1}{T}\|A_T - \mathbb{I}\| \quad \text{and} \quad L_{\psi,u} = \frac{1}{T}\|B_T\|. \tag{A.44}$$

Following the definitions in Propositions 3.2.18 and 3.2.18, we have $\hat{\gamma} = \frac{1}{\sqrt{a_1}}\sigma\hat{\kappa}\eta$ and $\kappa = \hat{\kappa}(1+T\sigma\theta)$, where $\eta = L_{\psi,u} + L_{\psi,x}\sigma$ and $\theta = L_{\psi,u}$. In order to validate Assumption 3.2.2 and compute an estimate for constant a_3, we compute the largest eigenvalue $\lambda_{\max}(\Delta P)$ of the matrix

$$\Delta P(T) := \frac{1}{T}(x_\dagger^\top P x_\dagger - x^\top Px), \tag{A.45}$$

with

$$x_{\dagger} = (A_T + B_T K)x \tag{A.46}$$

and $K = -H^{-1}G$. Choosing

$$a_3 = \min_{T} \lambda_{\max}(\Delta P(T)), \tag{A.47}$$

we obtain a value of a_3 that satisfies (3.37b) for any T such that $0 \leq T \leq T_{\mathrm{d}}$. Finally, we can compute the constants in (3.37a) as $a_1 = \lambda_{\min}(P)$ and $a_2 = \lambda_{\max}(P)$.

Given that we can numerically compute all the constants involved in the Assumptions of Theorem A.4.9, it is possible to compute the longest sampling time for which the auxiliary system in (A.24) is asymptotically stable. Figure A.1 shows the eigenvalues of the auxiliary system as a function of T.

Finally, Figure A.2 shows the eigenvalues of the system-optimizer dynamics defined as

$$\begin{bmatrix} x_+ \\ u_+ \end{bmatrix} = \begin{bmatrix} A_{T_{\mathrm{d}}} & B_{T_{\mathrm{d}}} \\ \tilde{H}^{-1} B_{T_{\mathrm{d}}}^{\top} P A_{T_{\mathrm{d}}} & \tilde{H}^{-1}(H - \tilde{H}) \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}. \tag{A.48}$$

The plot reveals some conservatism of the analysis based on the auxiliary dynamics. In fact, the largest T for which the system-optimizer dynamics are stable is $T \approx 0.6$ as opposed to $T \approx 0.15$ obtained from the analysis of the auxiliary dynamics (see Figure A.1).

A.5 A globalization heuristic for the feasible SQP strategy of Section 4.1

In order to enforce descent properties of the proposed method, a globalization strategy that guarantees contraction of the outer iterations is necessary. To this end, we propose the following modification of the gradient term:

$$a = \beta \cdot \nabla_y f(\tilde{y}) + \beta \cdot P(\hat{y} - \tilde{y}) + (1 - \beta) \cdot M(\hat{y} - \tilde{y}). \tag{A.49}$$

Let $\Delta = \|\hat{z}_+ - \hat{z}\|$ denote the norm of the primal-dual step associated with the inner iterations. The nonlinear root-finding problem (4.18) takes the following form:

$$\tilde{F}(z) := \begin{bmatrix} \beta \cdot \nabla_y f(\tilde{y}) + (\beta \cdot P + (1 - \beta) \cdot M)(y - \tilde{y}) + G(\tilde{y})^{\top} \lambda \\ g(y) \end{bmatrix} = 0. \tag{A.50}$$

The following Lemma shows that at any feasible point $\tilde{y}$, for sufficiently small values of the globalization parameter β, the step computed by the feasible SQP strategy is a descent direction.

Lemma A.5.1 (Descent properties)**.** *Let Assumption 4.1.6 hold and let $\tilde{y}$ be a feasible point, i.e. $g(\tilde{y}) = 0$. Moreover, let $\Delta y(\beta)$ denote the full step associated with $\tilde{y}$ as a function of β, i.e., $\Delta y(\beta) := \tilde{y}_+(\beta) - \tilde{y}$. Then there exist a strictly positive constants $\hat{\beta}$ and c_1 such that, for any $0 < \beta \leq \hat{\beta}$, the following holds:*

$$f(\tilde{y} + \Delta y(\beta)) - f(\tilde{y}) < -c_1 \cdot \|\Delta y(\beta)\|^2. \tag{A.51}$$

Proof. Regard the linearized problem

$$F_{\text{lin}}(z, \beta) := \begin{bmatrix} \beta \cdot \nabla_y f(\tilde{y}) + W(\beta)(y - \tilde{y}) + G(\tilde{y})^\top \lambda \\ G(\tilde{y})(y - \tilde{y}) \end{bmatrix} = 0, \tag{A.52}$$

with $W(\beta) := \beta \cdot P + (1 - \beta) \cdot M$ and let $\Delta y_{\text{lin}}(\beta)$ denote the primal step associated with its solution. First, notice that if β is sufficiently small and $M \succ 0$, LICQ and SOSC are satisfied for the QP associated with (A.52) such that a unique solution exists. Second, regard the inequality

$$f(\tilde{y} + \Delta y_{\text{lin}}(\beta)) - f(\tilde{y}) \leq \nabla_y f(\tilde{y})^\top \Delta y_{\text{lin}}(\beta) + \gamma \|\Delta y_{\text{lin}}(\beta)\|^2, \tag{A.53}$$

where γ is a positive constant. Due to (A.52), we can write

$$\begin{aligned} \nabla_y f(\tilde{y})^\top \Delta y_{\text{lin}}(\beta) \quad &= -\frac{1}{\beta}(\Delta y_{\text{lin}}(\beta)^\top W(\beta) \Delta y_{\text{lin}}(\beta) + \lambda^\top G(\tilde{y}) \Delta y_{\text{lin}}(\beta)) \\ &= -\frac{1}{\beta} \Delta y_{\text{lin}}(\beta)^\top W(\beta) \Delta y_{\text{lin}}(\beta), \end{aligned} \tag{A.54}$$

where the last equality follows from $G(\tilde{y}) \Delta y_{\text{lin}}(\beta) = 0$. Using this fact we obtain

$$f(\tilde{y} + \Delta y_{\text{lin}}(\beta)) - f(\tilde{y}) \leq -\frac{1}{\beta} \cdot \Delta y_{\text{lin}}(\beta)^\top W(\beta) \Delta y_{\text{lin}}(\beta) + \gamma \|\Delta y_{\text{lin}}(\beta)\|^2, \tag{A.55}$$

such that it becomes apparent that, for any sufficiently small β, $\Delta y_{\text{lin}}(\beta)$ attains the desired descent property. In order to prove the result, it suffices to show that $\Delta y(\beta)$ preserves such property. To this end, notice that, by straightforward application of Dini's Theorem to (A.52), we can show that $\Delta y_{\text{lin}}(\beta) = O(\beta)$. Similarly, for sufficiently small β and ξ, the perturbed problem

$$F_{\text{pert}}(z, \beta, \xi) := \begin{bmatrix} \beta \cdot \nabla_y f(\tilde{y}) + W(\beta)(y - \tilde{y}) + G(\tilde{y})^\top \lambda \\ g(y) + \xi \end{bmatrix} = 0, \tag{A.56}$$

has a unique solution in a properly defined nonempty neighborhood and the associated primal step satisfies $\Delta y_{\text{pert}}(\beta, \xi) = \Delta y(\beta) + O(\|\xi\|)$. Choosing $\xi = \bar{\xi} = G(\tilde{y})\Delta y_{\text{lin}}(\beta) - g(\tilde{y} + \Delta y_{\text{lin}}(\beta)) = O(\|\Delta y_{\text{lin}}(\beta)\|^2) = O(\beta^2)$, we obtain

$$\Delta y_{\text{lin}}(\beta) = \Delta y_{\text{pert}}(\beta, \bar{\xi}) = \Delta y(\beta) + O(\beta^2). \tag{A.57}$$

Finally, due to differentiability of f, we can write

$$\begin{aligned} f(\tilde{y}+&\Delta y(\beta)) - f(\tilde{y}) \\ &= f(\tilde{y} + \Delta y_{\text{lin}}(\beta)) - f(\tilde{y}) + O(\beta^2) \\ &\leq -\frac{1}{\beta} \cdot \Delta y_{\text{lin}}(\beta)^\top W(\beta) \Delta y_{\text{lin}}(\beta) + \gamma \|\Delta y_{\text{lin}}(\beta)\|^2 + O(\beta^2) \end{aligned} \tag{A.58}$$

and, given that $\Delta y_{\text{lin}} = O(\beta)$, there exist a strictly positive constant $\hat{\beta}$, such that, for all $0 < \beta \leq \hat{\beta}$, the desired descent property holds. □

A.6 CS-NMPC for RSMs: additional results

In the following, we report additional simulation and experimental (for CS-NMPC only) results carried out at a reference speed of $165\,\mathrm{rad\,s^{-1}}$. In this scenario, the impact of the input constraint is even stronger than for the one used in Sections 6.2.1 and 6.2.2. Since the results showed potentially damaging behavior when controlling the RSM with the gain-scheduled PI, we ran the corresponding experiments only with CS-NMPC. The simulation and experimental results are reported in Figures A.3, A.4 and A.5. These additional results confirm the observation made for the scenario reported in Sections 6.2.1 and 6.2.2.

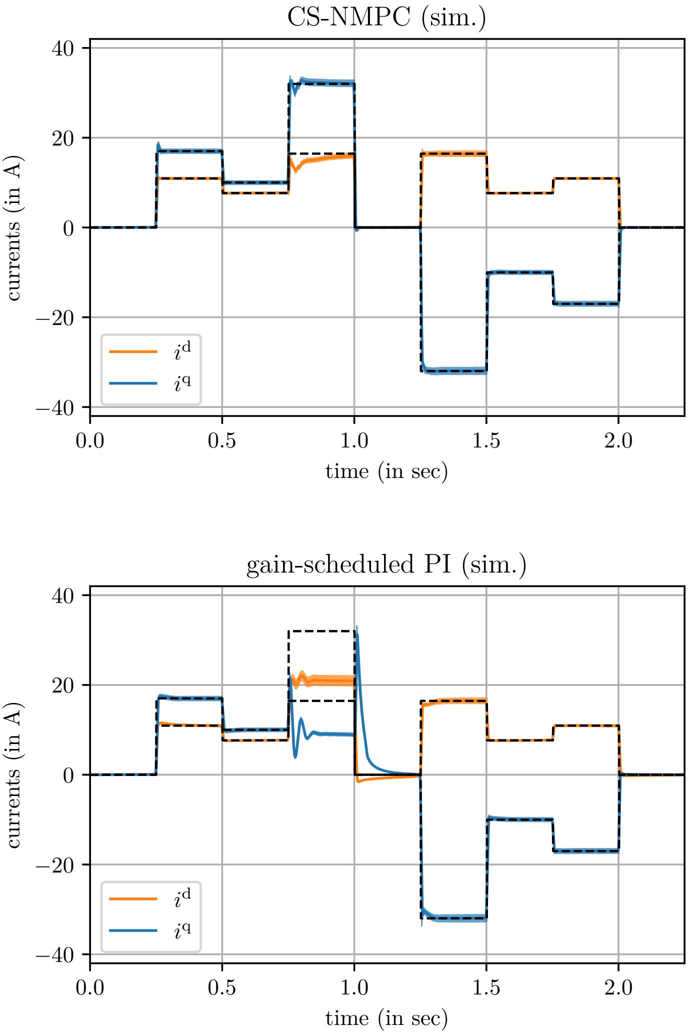


Figure A.3: Current steps at 165 rad s^{-1} (simulation): closed-loop trajectories obtained using CS-NMPC (left) and gain-scheduled PI controller (right).

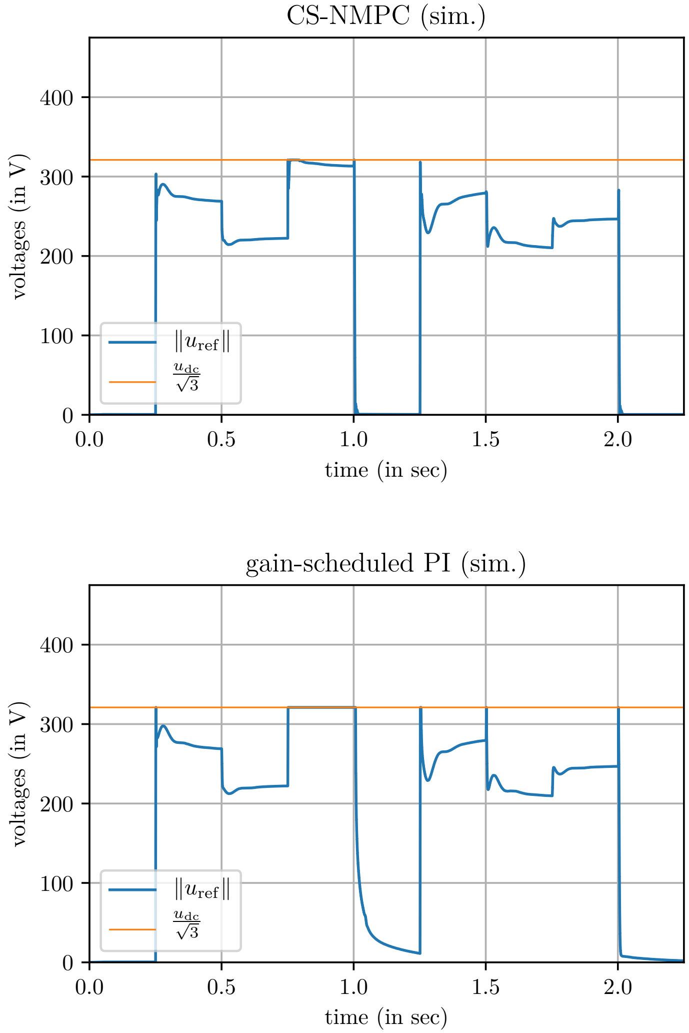


Figure A.4: Current steps at $165\,\mathrm{rad\,s^{-1}}$ (simulation): two-norm of voltage references u_{ref} commanded by the two controllers and u_{dc} over time. During the third current step, the PI controller saturates and does not steer the system to the desired reference.

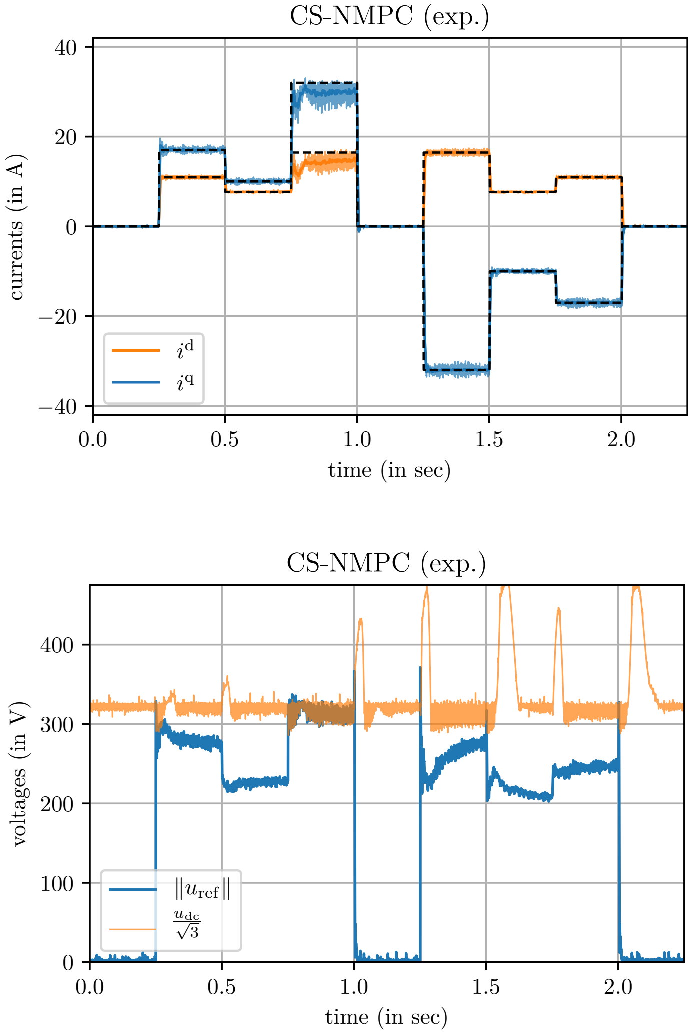


Figure A.5: Current steps at $165\,\mathrm{rad\,s^{-1}}$ (experiment): closed-loop current trajectories obtained using the two controllers under analysis. Due to the strong effect of input saturation observed in simulation, for this value of the reference speed it was not possible to run the experiment with the PI controller.

Bibliography

[Kit, 2010] (2010). Kitty Hawk, Mountain View, CA, USA. https://kittyhawk.aero.

[EHa, 2014] (2014). Ehang Inc. Guangzhou, China. http://www.ehang.com.

[Nui, 2020] (2020). Nuitka. https://github.com/Nuitka/Nuitka.

[Albersmeyer and Diehl, 2010] Albersmeyer, J. and Diehl, M. (2010). The lifted Newton method and its application in optimization. *SIAM Journal on Optimization*, 20(3):1655–1684.

[Albin et al., 2017] Albin, T., Ritter, D., Liberda, N., Quirynen, R., and Diehl, M. (2017). In-vehicle realization of nonlinear MPC for gasoline two-stage turbocharging airpath control. *IEEE Transactions on Control Systems Technology*, pages 1–13.

[Allan et al., 2017] Allan, D. A., Bates, C. N., Risbeck, M. J., and Rawlings, J. B. (2017). On the inherent robustness of optimal and suboptimal nonlinear MPC. *Systems & Control Letters*, 106:68–78.

[Allgower and Georg, 1990] Allgower, E. L. and Georg, K. (1990). *Introduction to Numerical Continuation Methods*. Colorado State University Press.

[Anderson et al., 1999] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition.

[Andersson et al., 2019] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

[Bertsekas, 1999] Bertsekas, D. P. (1999). *Nonlinear Programming.* Athena Scientific, 2nd edition.

[Besselmann et al., 2015] Besselmann, T. J., Van de moortel, S., Almér, S., Jörg, P., and Ferreau, H. J. (2015). Model predictive control in the multi-megawatt range. *IEEE Transactions on Industrial Electronics*, 63(7):4641–4648.

[Betsch and Siebert, 2009] Betsch, P. and Siebert, R. (2009). Rigid body dynamics in terms of quaternions: Hamiltonian formulation and conserving numerical integration. *International Journal for Numerical Methods in Engineering*, 79(4):444–473.

[Betz et al., 1993] Betz, R., Lagerquist, R., Jovanovic, M., Miller, T., and Middleton, R. (1993). Control of synchronous reluctance machines. *IEEE Transactions on Industry Applications*, 29(6):1110–1122.

[Bezanson et al., 2017] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98.

[Bisschop and Meeraus, 1977] Bisschop, J. and Meeraus, A. (1977). Matrix augmentation and partitioning in the updating of the basis inverse. *Mathematical Programming*, 13(1):241–254.

[Bock, 1983] Bock, H. G. (1983). Recent advances in parameter identification techniques for ODE. In *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, pages 95–121. Birkhäuser.

[Bock et al., 2007] Bock, H. G., Diehl, M., Kostina, E. A., and Schlöder, J. P. (2007). Constrained optimal feedback control of systems governed by large differential algebraic equations. In *Real-Time and Online PDE-Constrained Optimization*, pages 3–22. SIAM.

[Bock and Plitt, 1984] Bock, H. G. and Plitt, K. J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the IFAC World Congress*, pages 242–247. Pergamon Press.

[Boldea et al., 1991] Boldea, I., Fu, Z., and Nasar, S. (1991). Torque vector control (TVC) of axially-laminated anisotropic (ALA) rotor reluctance synchronous motors. *Electric machines and power systems*, 19(3):381–398.

[Bolognani et al., 2011] Bolognani, S., Peretti, L., and Zigliotto, M. (2011). Online MTPA control strategy for DTC synchronous-reluctance-motor drives. *IEEE Transactions on Power Electronics*, 26(1):20–28.

[Bonnans and Shapiro, 1998] Bonnans, J. F. and Shapiro, A. (1998). Optimization problems with perturbations: A guided tour. *SIAM Review*, 40:228–264.

[Bouabdallah et al., 2004] Bouabdallah, S., Noth, A., and Siegwart, R. (2004). Pid vs lq control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.

[Chen and Allgöwer, 1998] Chen, H. and Allgöwer, F. (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218.

[Chikhi et al., 2010] Chikhi, A., Djarallah, M., and Chikhi, K. (2010). A comparative study of field-oriented control and direct-torque control of induction motors using an adaptive flux observer. *Serbian Journal of Electrical Engineering*, 7(1):41–55.

[Cimini et al., 2020] Cimini, G., Bernardini, D., Levijoki, S., and Bemporad, A. (2020). Embedded model predictive control with certified real-time optimization for synchronous motors. *IEEE Transactions on Control Systems Technology*, pages 1–8.

[Cortes et al., 2008] Cortes, P., Kazmierkowski, M., Kennel, R., Quevedo, D., and Rodriguez, J. (2008). Predictive control in power electronics and drives. *IEEE Transactions on Industrial Electronics*, 55(12):4312–4324.

[Diehl, 2001] Diehl, M. (2001). *Real-Time Optimization for Large Scale Nonlinear Processes.* PhD thesis, University of Heidelberg.

[Diehl, 2002] Diehl, M. (2002). *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik.* VDI Verlag, Düsseldorf. PhD Thesis.

[Diehl, 2016] Diehl, M. (2016). *Lecture Notes on Numerical Optimization.* (Available online: http://cdn.syscop.de/publications/Diehl2016.pdf).

[Diehl et al., 2002a] Diehl, M., Bock, H., and Schlöder, J. (2002a). Newton-type methods for the approximate solution of nonlinear programming problems in real-time. In Pillo, G. D. and Murli, A., editors, *High Performance Algorithms and Software for Nonlinear Optimization*, pages 177–200. Kluwer Academic Publishers B.V.

[Diehl et al., 2002b] Diehl, M., Bock, H. G., Schlöder, J. P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002b). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585.

[Diehl et al., 2009] Diehl, M., Ferreau, H. J., and Haverbeke, N. (2009). Efficient numerical methods for nonlinear MPC and moving horizon estimation. In Magni, L., Raimondo, M., and Allgöwer, F., editors, *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 391–417. Springer.

[Diehl et al., 2007] Diehl, M., Findeisen, R., and Allgöwer, F. (2007). A stabilizing real-time implementation of nonlinear model predictive control. In Biegler, L., Ghattas, O., Heinkenschloss, M., Keyes, D., and van Bloemen Waanders, B., editors, *Real-Time and Online PDE-Constrained Optimization*, pages 23–52. SIAM.

[Diehl et al., 2005] Diehl, M., Findeisen, R., Allgöwer, F., Bock, H. G., and Schlöder, J. P. (2005). Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308.

[Diehl et al., 2003] Diehl, M., Findeisen, R., Allgöwer, F., Schlöder, J., and Bock, H. (2003). Stability of nonlinear model predictive control in the presence of errors due to numerical online optimization. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 1419–1424, Maui, Hawaii.

[Diehl and Gros, 2017] Diehl, M. and Gros, S. (2017). *Numerical Optimal Control - script draft.* (Available online: `https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf`).

[Diehl and Gros, 2018] Diehl, M. and Gros, S. (expected to be published in 2018). *Numerical Optimal Control.* –.

[Domahidi et al., 2012a] Domahidi, A., Mariethoz, S., and Morari, M. (2012a). High-bandwidth explicit model predictive control of electrical drives. *IEEE Transactions on Industry Applications*, 48(6):1980–1992.

[Domahidi et al., 2012b] Domahidi, A., Zgraggen, A., Zeilinger, M. N., Morari, M., and Jones, C. N. (2012b). Efficient interior point methods for multistage problems arising in receding horizon control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 668–674, Maui, HI, USA.

[Dontchev and Rockafellar, 2009] Dontchev, A. L. and Rockafellar, R. T. (2009). *Implicit Functions and Solution Mappings.* Springer.

[Duff, 2004] Duff, I. (2004). Ma57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2):118–144.

[Eldeeb et al., 2017] Eldeeb, H., Hackl, C. M., Horlbeck, L., and Kullick, J. (2017). A unified theory for optimal feedforward torque control of anisotropic synchronous machines. *International Journal of Control*, 91(10):2273–2302.

[Englert and Graichen, 2018] Englert, T. and Graichen, K. (2018). A fixed-point iteration scheme for model predictive torque control of PMSMs. In *Proceedings of the 6th IFAC Conference on Nonlinear Model Predictive Control*, volume 51, pages 568–573, Madison, Wisconsin, USA.

[Feller and Ebenbauer, 2016] Feller, C. and Ebenbauer, C. (2016). A stabilizing iteration scheme for model predictive control based on relaxed barrier functions. *arXiv preprint at arXiv:1603.04605.*

[Feller and Ebenbauer, 2017] Feller, C. and Ebenbauer, C. (2017). Relaxed logarithmic barrier function based model predictive control of linear systems. *IEEE Transactions on Automatic Control*, 62(3):1223–1238.

[Ferreau et al., 2008] Ferreau, H. J., Bock, H. G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830.

[Ferreau et al., 2014] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

[Fiacco, 1983] Fiacco, A. (1983). *Introduction to sensitivity and stability analysis in nonlinear programming.* Academic Press, New York.

[Fletcher, 1982] Fletcher, R. (1982). Second order corrections for non-differentiable optimization. *Numerical Analysis. Lecture Notes in Mathematics*, 912.

[Frison, 2015] Frison, G. (2015). *Algorithms and Methods for High-Performance Model Predictive Control.* PhD thesis, Technical University of Denmark (DTU).

[Frison, 2017] Frison, G. (2017). HPIPM, High-performance interior-point qp solvers. https://github.com/giaf/hpipm.

[Frison and Jørgensen, 2013] Frison, G. and Jørgensen, J. B. (2013). A fast condensing method for solution of linear-quadratic control problems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 7715–7720.

[Frison et al., 2017] Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., and Diehl, M. (2017). BLASFEO: Basic linear algebra subroutines for embedded optimization. *arXiv:1704.02457.*

[Frison et al., 2018] Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., and Diehl, M. (2018). BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):42:1–42:30.

[Frison et al., 2020] Frison, G., Sartor, T., Zanelli, A., and Diehl, M. (2020). The BLAS API of BLASFEO: Optimizing performance for small matrices. *ACM Transactions on Mathematical Software (TOMS)*, 46(2):15:1–15:36.

[Frison et al., 2014] Frison, G., Sorensen, H. B., Dammann, B., and Jørgensen, J. B. (2014). High-performance small-scale solvers for linear model predictive control. In *Proceedings of the European Control Conference (ECC)*, pages 128–133.

[Geyer, 2016] Geyer, T. (2016). *Model Predictive Control of High Power Converters and Industrial Drives*. John Wiley & Sons.

[Geyer et al., 2009] Geyer, T., Papafotiou, G., and Morari, M. (2009). Model predictive direct torque control – Part I: Concept, algorithm, and analysis. *IEEE Transaction on Industrial Electronics*, 56(6):1894–1905.

[Gill et al., 1987] Gill, P. E., Murray, W., Saunders, M. A., and H., W. M. (1987). A Schur-complement method for sparse quadratic programming. Technical report, Stanford Univ., CA (USA). Systems Optimization Lab.

[Gill and Wong, 2015] Gill, P. E. and Wong, E. (2015). Methods for convex and general quadratic programming. *Mathematical Programming Computation*, 7(1):71–112.

[Gould and Toint, 2002] Gould, N. I. M. and Toint, P. L. (2002). An iterative working-set method for large-scale nonconvex quadratic programming. *Applied Numerical Mathematics*, 43(1):109–128.

[Graichen and Kugi, 2010] Graichen, K. and Kugi, A. (2010). Stability and incremental improvement of suboptimal MPC without terminal constraints. *IEEE Transactions on Automatic Control*, 55(11):2576–2580.

[Grüne and Nesić, 2003] Grüne, L. and Nesić, D. (2003). Optimization-based stabilization of sampled-data nonlinear systems via their approximate discrete-time models. *SIAM J. Control Optim*, 42(1):98–122.

[Hackl et al., 2016] Hackl, C., Kamper, M., Kullick, J., and Mitchell, J. (2016). Current control of reluctance synchronous machines with online adjustment of the controller parameters. In *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*. IEEE.

[Hackl, 2015] Hackl, C. M. (2015). Dynamische Reibungsmodellierung: Das Lund-Grenoble (LuGre) Reibmodell *(available at the authors upon request)*. In Schröder, D., editor, *Elektrische Antriebe – Regelung von Antriebssystemen*, chapter 25, pages 1615–1657. Springer-Verlag.

[Hackl, 2017] Hackl, C. M. (2017). *Non-identifier Based Adaptive Control in Mechatronics*. Springer International Publishing.

[Janka et al., 2016] Janka, D., Kirches, C., Sager, S., and Wächter, A. (2016). An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal hessian matrix. *Mathematical Programming Computation*, 8(4):435–459.

[Kaczorek, 2008] Kaczorek, T. (2008). The choice of the forms of Lyapunov functions for a positive 2D Roesser model. *International Journal of Applied Mathematics and Computer Science*, 17(4):471–475.

[Kamel et al., 2015] Kamel, M., Alexis, K., Achtelik, M., and Siegwart, R. (2015). Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *IEEE Conference on Control Applications*, Sidney, Australia.

[Kamper et al., 1996] Kamper, M. J., van der Merwe, F., and Williamson, S. (1996). Direct finite element design optimisation of the cageless reluctance synchronous machine. *IEEE Transactions on Power Conversion*, 11(3):547–555.

[Kouzoupis et al., 2018] Kouzoupis, D., Frison, G., Zanelli, A., and Diehl, M. (2018). Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46(4):863–882.

[Lagerquist et al., 1994] Lagerquist, R., Boldea, I., and Miller, T. (1994). Sensorless-control of the synchronous reluctance motor. *IEEE Transactions on Industry Applications*, 30(3):673–682.

[Lam et al., 2015] Lam, S. K., Pitrou, A., and Seibert, S. (2015). Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, New York, NY, USA. Association for Computing Machinery.

[Landsmann et al., 2010a] Landsmann, P., Kennel, R., de Kock, H. W., and Kamper, M. J. (2010a). Fundamental saliency based encoderless control for reluctance synchronous machines. In *Proceedings of the XIX International Conference on Electrical Machines (ICEM)*, pages 1–7, Incheon, South Korea.

[Landsmann et al., 2010b] Landsmann, P., Paulus, D., Stolze, P., and Kennel, R. (2010b). Reducing the parameter dependency of encoderless predictive torque control for reluctance machines. In *Proceedings of the IEEE International Symposium on Sensorless Control for Electrical Drives (SLED)*, pages 93–99.

[Liao-McPherson et al., 2019] Liao-McPherson, D., Nicotra, M., and Kolmanovsky, I. (2019). Time distributed sequential quadratic programming for model predictive control: Stability and robustness. *arXiv preprint arXiv:1903.02605.*

[Matsuo and Lipo, 1993] Matsuo, T. and Lipo, T. (1993). Field oriented control of synchronous reluctance machine. In *Power Electronics Specialists Conference, 1993. PESC '93 Record., 24th Annual IEEE*, pages 425–431.

[Morari and Maeder, 2012] Morari, M. and Maeder, U. (2012). Nonlinear offset-free model predictive control. *Automatica*, 48(9):2059–2067.

[Muske and Badgwell, 2002] Muske, K. R. and Badgwell, T. A. (2002). Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12:617–632.

[Nishihara et al., 2015] Nishihara, R., Lessard, L., Recht, B., Packard, A., and Jordan, M. (2015). A general analysis of the convergence of admm. In *Proceedings of the 32nd International Conference on International Conference on Machine*, volume 37, pages 343–352.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, 2 edition.

[Ohtsuka, 2015] Ohtsuka, T. (2015). A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control. In *Control Conference (ECC), 2015 European*, pages 73–86.

[Oliphant, 2006] Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

[OpenBLAS, 2011] OpenBLAS (2011). OpenBLAS: An optimized BLAS library. http://www.openblas.net/.

[Ostrowski, 1966] Ostrowski, A. (1966). *Solutions of Equations and Systems of Equations.* Academic Press, New York.

[Pannocchia et al., 2015] Pannocchia, G., Gabiccini, M., and Artoni, A. (2015). Offset-free MPC explained: novelties, subtleties, and applications. In

Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control, volume 48, pages 342–251.

[Pannocchia and Rawlings, 2003] Pannocchia, G. and Rawlings, J. (2003). Disturbance Models for Offset-Free Model-Predictive Control. *AIChE Journal*, 49:426–437.

[Pannocchia et al., 2011] Pannocchia, G., Rawlings, J., and Wright, S. (2011). Conditions under which suboptimal nonlinear MPC is inherently robust. *System & Control Letters*, 60(9):747–755.

[Quevedo et al., 2019] Quevedo, D. E., Aguilera, R., and Geyer, T. (2019). Model predictive control for power electronics applications. In Rakovic, S. V. and Levine, W. S., editors, *Handbook of Model Predictive Control*, pages 551–580. Birkhäuser, Cham.

[Quirynen et al., 2013] Quirynen, R., Gros, S., and Diehl, M. (2013). Efficient NMPC for nonlinear models with linear subsystems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 5101–5106.

[Quirynen et al., 2015] Quirynen, R., Gros, S., and Diehl, M. (2015). Inexact Newton based lifted implicit integrators for fast nonlinear MPC. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, pages 32–38.

[Quirynen et al., 2017] Quirynen, R., Gros, S., Houska, B., and Diehl, M. (2017). Lifted collocation integrators for direct optimal control in ACADO toolkit. *Mathematical Programming Computation*, 9(4):527–571.

[Rao et al., 1998] Rao, C. V., Wright, S. J., and Rawlings, J. B. (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757.

[Rashad et al., 2004] Rashad, E., Radwan, T., and Rahman, M. (2004). A maximum torque per ampere vector control strategy for synchronous reluctance motors considering saturation and iron losses. In *Conference Record of the 2004 IEEE Industry Applications Conference (39th IAS Annual Meeting)*, volume 4, pages 2411–2417.

[Rawlings et al., 2017] Rawlings, J. B., Mayne, D. Q., and Diehl, M. M. (2017). *Model Predictive Control: Theory, Computation, and Design.* Nob Hill, 2nd edition.

[Robinson, 1980] Robinson, S. M. (1980). Strongly Regular Generalized Equations. *Mathematics of Operations Research, Vol. 5, No. 1 (Feb., 1980), pp. 43-62*, 5:43–62.

[Sargent and Murtagh, 1973] Sargent, R. and Murtagh, B. (1973). Projection methods for nonlinear programming. *Mathematical Programming*, 4(1):245–268.

[Scokaert et al., 1999] Scokaert, P. O. M., Mayne, D. Q., and Rawlings, J. (1999). Suboptimal Model Predictive Control (Feasibility Implies Stability). *IEEE Transactions on Automatic Control*, 44(3):648–654.

[Steinbach, 1996] Steinbach, M. (1996). Structured interior point SQP methods in optimal control. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76(S3):59–62.

[Stellato et al., 2020] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672.

[Tran-Dinh et al., 2012] Tran-Dinh, Q., Savorgnan, C., and Diehl, M. (2012). Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization. *SIAM J. Optimization*, 22(4):1258–1284.

[Van Der Walt et al., 2011] Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22.

[Van Parys and Pipeleers, 2018] Van Parys, R. and Pipeleers, G. (2018). Real-time proximal gradient method for linear MPC. In *Proceedings of the European Control Conference (ECC)*, Lymassol, Cyprus.

[Verschueren et al., 2021] Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., and Diehl, M. (2021). acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation.*

[Verschueren et al., 2016] Verschueren, R., van Duijkeren, N., Quirynen, R., and Diehl, M. (2016). Exploiting convexity in direct optimal control: a sequential convex quadratic programming method. In *Proceedings of the IEEE Conference on Decision and Control (CDC).*

[Wächter and Biegler, 2009] Wächter, A. and Biegler, L. (2009). IPOPT - an Interior Point OPTimizer. https://projects.coin-or.org/Ipopt.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

[Wills and Heath, 2004] Wills, A. G. and Heath, W. P. (2004). Barrier function based model predictive control. *Automatica*, 40(8):1415–1422.

[Wirsching et al., 2006] Wirsching, L., Bock, H. G., and Diehl, M. (2006). Fast NMPC of a chain of masses connected by springs. In *Proceedings of the IEEE International Conference on Control Applications, Munich*, pages 591–596.

[Xu et al., 1991] Xu, L., Xu, X., Lipo, T. A., and Novotny, D. W. (1991). Vector control of a synchronous reluctance motor including saturation and iron losses. *IEEE Transactions on Industrial Applications*, 27(5):977–985.

[Yamamoto et al., 2009] Yamamoto, S., Adawey, J., and Ara, T. (2009). Maximum efficiency drives of synchronous reluctance motors by a novel loss minimization controller considering cross-magnetic saturation. In *Proceedings of the 2009 IEEE Energy Conversion Congress and Exposition*, pages 288–293.

[Yang, 2012] Yang, Y. (2012). Spacecraft attitude determination and control: Quaternion based method. *Annual Reviews in Control*, 36(2):198–219.

[Zanelli et al., 2017a] Zanelli, A., Domahidi, A., Jerez, J. L., and Morari, M. (2017a). FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control.*

[Zanelli et al., 2018] Zanelli, A., Horn, G., Frison, G., and Diehl, M. (2018). Nonlinear model predictive control of a human-sized quadrotor. In *Proceedings of the European Control Conference (ECC)*, pages 1542–1547.

[Zanelli et al., 2021a] Zanelli, A., Kullick, J., Eldeeb, H., Frison, G., Hackl, C., and Diehl, M. (2021a). Continuous control set nonlinear model predictive control of reluctance synchronous machines. *IEEE Transactions on Control Systems Technology*, pages 1–12.

[Zanelli et al., 2016] Zanelli, A., Quirynen, R., and Diehl, M. (2016). An efficient inexact NMPC scheme with stability and feasibility guarantees. In *Proceedings of 10th IFAC Symposium on Nonlinear Control Systems*, Monterey, CA, USA.

[Zanelli et al., 2019a] Zanelli, A., Quirynen, R., and Diehl, M. (2019a). Efficient zero-order NMPC with feasibility and stability guarantees. In *Proceedings of the European Control Conference (ECC)*, Naples, Italy.

[Zanelli et al., 2017b] Zanelli, A., Quirynen, R., Frison, G., and Diehl, M. (2017b). A partially tightened real-time iteration scheme for nonlinear model predictive control. In *Proceedings of 56th IEEE Conference on Decision and Control*, Melbourne, Australia.

[Zanelli et al., 2019b] Zanelli, A., Tran-Dinh, Q., and Diehl, M. (2019b). Contraction estimates for abstract real-time algorithms for NMPC. In *Proceedings of the IEEE Conference on Decision and Control*, Nice, France.

[Zanelli et al., 2020] Zanelli, A., Tran-Dinh, Q., and Diehl, M. (2020). Stability analysis of real-time methods for equality constrained NMPC. In *Proceedings of the IFAC World Congress*, Berlin, Germany.

[Zanelli et al., 2021b] Zanelli, A., Tran-Dinh, Q., and Diehl, M. (2021b). A lyapunov function for the combined system-optimizer dynamics in inexact model predictive control. *Automatica*, 134:109901.

[Zavala and Anitescu, 2010] Zavala, V. and Anitescu, M. (2010). Real-Time Nonlinear Optimization as a Generalized Equation. *SIAM J. Control Optim.*, 48(8):5444–5467.

Curriculum vitae

Andrea Zanelli, born June 2, 1990, in Como, Italy.

Education

2015 - 2020: PhD in Engineering - Prof. Moritz Diehl, University of Freiburg, Germany.

2015: Research assistant - Dr. Alexander Domahidi, Dr. Juan Jerez, Prof. Manfred Morari, IfA, ETH Zurich, Switzerland - embotech AG (5 months).

2014 - 2015: Master thesis - Dr. Alexander Domahidi, Dr. Juan Jerez, Prof. Manfred Morari, IfA, ETH Zurich, Switzerland - embotech AG.

2012 - 2015: MSc in Robotics, Systems and Control, ETH Zurich, Switzerland.

2009 - 2012: BSc in Automation Engineering, Politecnico di Milano, Italy

Industrial experience

2016: Internship - Greg Horn, Kitty Hawk, Mountain View, California, USA (2 months).

2013 - 2014: Internship - Dr. Joachim Ferreau, ABB research center, Baden, Dättwil, Zurich, Switzerland (9 months).

List of publications

Journal articles (as main author)

1. Zanelli, A., Domahidi, A., Jerez, J. L., Morari, M. FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. International Journal of Control (2017).
2. Zanelli, A., Kullick, J., Eldeeb, H., Frison, G., Hackl, C., Diehl, M. Continuous control set nonlinear model predictive control of reluctance synchronous machines. IEEE Transactions on Control Systems Technology (2021)
3. Zanelli, A., Tran-Dinh, Q., Diehl, M. A Lyapunov function for the combined system-optimizer dynamics in nonlinear model predictive control. Automatica (2021).
4. Zanelli, A., Sartor, T., Rutquist, P., Frison, G., [...], Diehl, M. prometeo: a domain specific language and Python-to-C transpiler for embedded high-performance computing (in preparation).
5. Zanelli, A., [...], Diehl, M. Asymptotic stability of progressive tightening model predictive control (in preparation).
6. Zanelli, A., [...], Diehl, M. A feasible sequential quadratic programming strategy with iterated second-order corrections (in preparation).

Journal articles (as co-author)

1. Frison, G. Kouzoupis, D., Sartor, T., Zanelli, A., Diehl, M. BLASFEO: Basic linear algebra subroutines for embedded optimization. ACM Transactions on Mathematical Software (2018)

2. Kouzoupis, D., Frison, G., Zanelli, A., Diehl, M. Recent advances in quadratic programming algorithms for nonlinear model predictive control. Vietnam Journal of Mathematics (2018).
3. Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., Diehl, M. acados: a modular open-source framework for fast embedded optimal control. Mathematical Programming Computation (2021).
4. Frison, G., Sartor, T., Zanelli, A., Diehl, M. The BLAS API of BLASFEO: Optimizing performance for small matrices. ACM Transactions on Mathematical Software (2020).

Conference proceedings (as main author)

1. Zanelli, A., Quirynen, R., Diehl, M. An efficient inexact NMPC scheme with stability and feasibility guarantees. In Proceedings of the IFAC Symposium on Nonlinear Control Systems (Monterey, CA, USA, August 2016).
2. Zanelli, A., Quirynen, R., Jerez, J., Diehl, M. A homotopy-based nonlinear interior-point method for NMPC. In Proceedings of the IFAC World Congress (Toulouse, France, July 2017).
3. Zanelli, A., Quirynen, R., Frison, G., Diehl, M. A partially tightened real-time iteration scheme for nonlinear model predictive control. In Proceedings of the IEEE Conference on Decision and Control (Melbourne, Australia, December 2017).
4. Zanelli, A., Horn, G., Frison, G., Diehl, M. Nonlinear model predictive control of a human-sized quadrotor. In Proceedings of the European Control Conference (Limassol, Cyprus, June 2018).
5. Zanelli, A., Quirynen, R., Diehl, M. Efficient zero-order NMPC with feasibility and stability guarantees. In Proceedings of the European Control Conference (Naples, Italy, June 2019).
6. Zanelli, A., Tran-Dinh, Q., Diehl, M. Contraction estimates for abstract real-time algorithms for NMPC. In Proceedings of the IEEE Conference on Decision and Control (Nice, France, Dec 2019).
7. Zanelli, A., Tran-Dinh, Q., Diehl, M. Stability analysis of real-time methods for equality constrained NMPC. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).

8. Zanelli, A., Frey, J., Messerer, F., Plum, F., Diehl, M. Zero-order robust nonlinear model predictive control with ellipsoidal uncertainty sets. In Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (Bratislava, Slovakia, July 2021).

Conference proceedings (as co-author)

1. Kouzoupis, D., Zanelli, A. Peyrl, H., Ferreau, H.J. Towards proper assessment of QP algorithms for embedded model predictive control. In Proceedings of the European Control Conference (Linz, Austria, June 2015).
2. Frison, G., Quirynen, R., Zanelli, A., Diehl, M., Jørgensen, J.B. Hardware tailored linear algebra for implicit integrators in embedded NMPC. In Proceedings of the IFAC World Congress (Toulouse, France, July 2017).
3. Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., Diehl, M. Towards a modular software package for embedded optimization. In Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (Madison, Wisconsin, USA, August 2018).
4. Nurkanovic, A., Zanelli, A., Albrecht, S., Diehl, M., The advanced step real time iteration for NMPC. In Proceedings of the IEEE Conference on Decision and Control (Nice, France, December 2019).
5. Baumgärtner, K., Zanelli, A., Diehl, M. Zero-order moving horizon estimation. In Proceedings of the IEEE Conference on Decision and Control (Nice, France, December 2019).
6. Nurkanovic, A., Mesanovic, A., Zanelli, A., Frey, J., Frison, G., Albrecht, S., Diehl, M. Real-time nonlinear model predictive control for microgrid operation. In Proceedings of the American Control Conference (Denver, Colorado, USA, July 2020).
7. Gargiani, M., Zanelli, A., Tran-Dinh, Q., Diehl, M., Hutter, F. Transferring optimality across data distributions via homotopy methods. In Proceedings of the International Conference on Learning Representations (Addis Ababa, Ethiopia, April 2020).
8. Baumgärtner, K., Zanelli, A., Diehl, M. A gradient condition for the arrival cost in moving horizon estimation In Proceedings of the European Control Conference (Saint Petersburg, Russia, May 2020).
9. Gargiani, M., Zanelli, A., Diehl, M., Hutter F. On the promise of the stochastic generalized Gauss-Newton method for training DNNs. Technical report available on arXiv (2020).

10. Nurkanovic, A., Zanelli, A., Albrecht, S., Frison, G., Diehl, M. Contraction properties of the advanced step real-time iteration for NMPC. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).
11. Schoels, T., Rutquist, P., Palmieri, L., Zanelli, A., Arras, K.O., Diehl, M. CIAO: MPC-based safe motion planning in predictable dynamic environments. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).
12. Kloeser, D., Schoels, T., Sartor, T., Zanelli, A., Frison, G., Diehl, M. NMPC for racing using a singularity-free path-parametric model with obstacle avoidance. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).
13. Geweth, D., Zanelli, A., Frison, G., Vollmer, U, Diehl M. Field oriented economic model predictive control for permanent magnet synchronous motors. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).
14. Carlos, B.B., Sartor, T., Zanelli, A., Diehl, M., Oriolo, G. Least conservative linearized constraint formulation for real-time motion generation. In Proceedings of the IFAC World Congress (Berlin, Germany, July 2020).
15. Carlos, B.B., Sartor, T., Zanelli, A., Frison, G., Burgard, W., A., Diehl, M., Oriolo, G. An efficient real-time NMPC for quadrotor position control under communication time-delay. In Proceedings of the International Conference on Control, Automation, Robotics and Vision (Shenzhen, China, December 2020).

FACULTY OF ENGINEERING
DEPARTMENT OF MICROSYSTEMS ENGINEERING
SYSTEMS CONTROL AND OPTIMIZATION LABORATORY
Georges-Köhler-Allee 102
DE-79110 Freiburg i. Br.