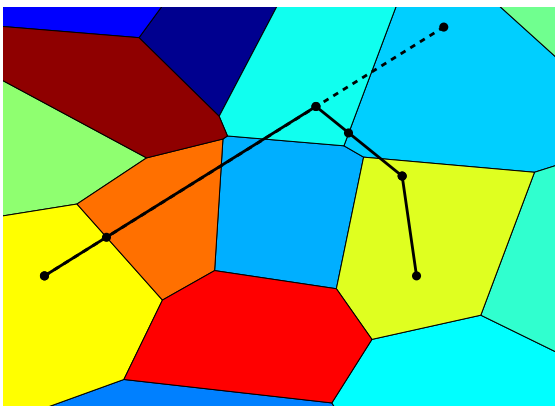
KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Parallel Algorithms for Optimization of Dynamic Systems in Real-Time



Janick Frasch

Dissertation presented in partial fulfillment of the requirements for the joint degree of “Doctor of Engineering Science” (KU Leuven) and “doctor rerum naturalium” (OVGU Magdeburg)

September 30, 2014

Parallel Algorithms for Optimization of Dynamic Systems in Real-Time

Janick FRASCH

Examination committee:
Prof. Dr. A. Pott, chair
Prof. Dr. S. Sager, supervisor
Prof. Dr. P. Benner
Prof. Dr. W. Kahle
(all OVGU Magdeburg)
Prof. Dr. M. Diehl, supervisor
Prof. Dr. J. Suykens
Prof. Dr. J. Swevers
Prof. Dr. S. Vandewalle
(all KU Leuven)
Prof. Dr. Dres. h.c. H. G. Bock, supervisor
(U Heidelberg)
Prof. Dr. C. N. Jones
(EPF Lausanne)

Dissertation presented in partial fulfillment of the requirements for the joint degree of "Doctor of Engineering Science" (KU Leuven) and "doctor rerum naturalium" (OVGU Magdeburg)

September 30, 2014

© 2013 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Janick Frasch, Kasteelpark Arenberg 10, bus 2446, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

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 the publisher.

ISBN XXX-XX-XXXX-XXX-X

D/XXXX/XXXX/XX

Acknowledgements

This thesis thrived under the influence of numerous persons. Their contributions added up to something that is fundamentally more than what I could have achieved without them and I would like to extend my deeply felt gratitude to every single one of them for pushing, pointing and guiding me, for trusting me and sharing with me, for advising and teaching me, for challenging me and scrutinizing my work and thoughts when necessary, and for all the other forms of help and support they have offered.

Among these persons, I would like to single out Sebastian Sager, Moritz Diehl, and Georg Bock, who contributed each in their own individual way to ensure the excellent supervision that I have received.

It is substatially due to Georg Bock that the Interdisciplinary Center for Scientific Computing (IWR) in Heidelberg forms such an inspiring environment, attracting dozens of excelling minds, whom I had the opportunity to learn from. A good share of the ideas that forms the basis of this thesis originates in the fertile soil of IWR.

But I would never even have benefited from this environment if it were not for Sebastian Sager, who sparked my enthusiasm for dynamic optimization, and who trusted in me besides my lack of previous experience in this domain. I am moreover grateful to Sebastian for his continued and active support for all of my plans, and for that he always gave me the freedom to pursue my own research ideas, even despite the fact that they were not always that well aligned with his own directions of research. He truly was and continues to be much more to me than a supervisor — a vigilant mentor and an inspiring role model far beyond research.

I owe my sincere gratitude to Moritz Diehl, not only for agreeing to become my co-supervisor at a very early stage of my PhD, but also for being this inexhaustible source (not to say this tremendous volcano) of ideas and inspiration, for inviting me multiple times, for stimulating, motivating, and unhesitatingly sharing with

me, and for his kindness and patience with me despite his (gazillions of) other obligations.

Moreover I am grateful to Alexander Domahidi and to Francesco Borrelli for hosting me for short and not so short research stays at ETH Zurich and UC Berkeley, respectively.

Many thanks are also in order to Joachim Ferreau, Christian Kirches, and Leonard Wirsching, who mentored me excellently in the early (and not so early anymore) days of my PhD, and from whom I obtained many of the tools and much of the knowledge that I needed to get acquainted with efficient algorithms and software for dynamic optimization.

I would like to thank my colleagues and fellow researchers in Heidelberg, Berkeley, Leuven, and Magdeburg (and all the other places some of them moved to by now), for the inspiration, the motivation, and the fruitful discussions we have had. I owe particular thanks to Joel Andersson, Yiqi Gao, Andrew Gray, Sébastien Gros, Boris Houska, Tony Huschto, Dennis Janka, Robert Kircheis, Attila Kozma, Rien Quirynen, Kristine Rinke, Robin Verschueren, Milan Vukov, and Mario Zanon for sharing their thoughts and ideas with me. It was a great pleasure for me working with them, and even though we did not manage to pursue all ideas to the stage of publication (yet), the discussions advanced my knowledge for certain.

I am indebted to Joel Andersson, Dimitris Kouzoupis, Jan Krümpelmann, Rien Quirynen, Kristine Rinke, Sebastian Sager, Tobias Weber, and Mario Zanon for providing valuable comments on earlier versions of this manuscript.

I gratefully acknowledge funding by the EU FP7 project EMBOCON and the Flemish IWT SBO project LeCoPro for large parts of my PhD studies. Paricularly, I would like to thank the initiators and the members of EMBOCON, for making this project possible in the first place, for all the inspiring ideas about embedded optimization for control — pardon, resource constrained platforms — I could soak up in the past years, and the great times we have had together at the multitude of EMBOCON events, conferences, and other occasions.

Last in this list, but surely not in priority, I would like to thank Leen Cuypers, Jacqueline De bruyn, and Susanne Hess for all the efforts they underwent on numerous occasions to excellently guide me through the administrative jungle that sometimes stood between me and the plans I had taken into my head.

Janick Frasch

Abstract

This thesis proposes optimization-based methods for control and estimation of dynamic systems. Due to their versatility, optimization-based schemes range among the most advanced and generic methods, and enjoy increasing popularity. The computational burden associated with the online solution of an optimization problem, however, still constitutes a major limiting factor on the control and estimation performance in a range of real-time applications that impedes the further practical success of this class of methods.

We focus on problems featuring long prediction or estimation horizons, with both linear and nonlinear dynamics, and constraints. Such problems are of great practical interest for a variety of reasons including stability guarantees and control or estimation performance. Particularly in the pursuit of economic objectives their formulation can be a key necessity. We develop algorithms that exploit the inherent sparsity of these problems with a focus on concurrency of the major computational steps, so as to ensure parallelizability on current and future computational architectures.

In particular, a hierarchical re-linearization procedure for nonlinear model predictive control (MPC) based on the well-established Real-Time Iteration scheme is given, thus further bridging the gap between linear and nonlinear MPC. An improved, reduced-complexity condensing algorithm for dimension reduction of the arising structured quadratic programming problems (QPs) is presented.

Moreover, a novel structure-exploiting sparse QP algorithm based on dual decomposition and a semismooth Newton method is proposed, aiming at a combination of sparsity-exploitation characteristics from interior-point methods and warmstarting capabilities from common active-set methods for dense problems. We prove finite convergence and rigorous infeasibility detection of the algorithm, and identify its tailored factorization routine with the Riccati recursion for linear-quadratic control. Most notably, we analyze various

theoretical and practical numerical aspects of this method, leading to an efficient implementation that improves the state-of-the-art in QP solution approaches by factors of more than ten in computation time on a variety of benchmark problems. The implementation has been publicly released as open-source code as part of this thesis.

Generalizations of this dual Newton idea to distributed quadratic programming and to nonlinear programming are examined. The development of a high-fidelity real-time feasible nonlinear MPC scheme for autonomous driving, which can be applied for real-time collision avoidance in dangerous driving conditions as well as for time-optimal driving, concludes this thesis.

Zusammenfassung

In dieser Arbeit werden optimierungsbasierte Verfahren zur Regelung und zur kombinierten Parameter- und Zustandsschätzung dynamischer Systeme vorgestellt. Aufgrund ihrer Vielseitigkeit zählen optimierungsbasierte Ansätze zu den universellsten und mächtigsten Verfahren und sind daher von großer praktischer Relevanz. Die Echtzeit-Lösung der zugrundeliegenden Optimierungsprobleme stellt jedoch nach wie vor eine kritische Herausforderung beim Einsatz dieser Methoden dar.

Wir betrachten hier insbesondere Probleme mit langen Prädiktions-, bzw. Schätzhorizonten, die sowohl durch lineare als auch durch nichtlineare Dynamiken, sowie durch zusätzliche Pfadbeschränkungen charakterisiert sein können. Das Formulieren solcher Probleme ist häufig aus Performanz- und Stabilitätsgesichtspunkten interessant und ist nimmt insbesondere im Zusammenhang mit Nicht-Tracking Zielfunktionalen eine zentrale Rolle ein. Ziel der in dieser Arbeit entwickelten Algorithmen ist eine effiziente Ausnutzung der probleminhärenten Dünnbesetztheit unter Berücksichtigung von Parallelisierungsaspekten auf aktuellen und zukünftigen Rechnerarchitekturen.

Insbesondere entwickeln wir ein hierarchisches Relinearisierungsverfahren für die nichtlineare modellprädiktive Regelung (Model Predictive Control, MPC), das auf dem etablierten Real-Time Iteration Scheme basiert und eine Brücke zwischen nichtlinearer und linearer MPC schlägt. Weiterhin wird ein verbesserter Condensing-Algorithmus zur Dimensionsreduktion der auftretenden strukturierten quadratischen Optimierungsprobleme (Quadratic Programming Problem, QP) vorgestellt, der eine Rechenzeitkomplexität von lediglich quadratischer Ordnung in der Horizontlänge aufweist.

Darüber hinaus wird ein strukturausnutzender Algorithmus zur direkten Lösung der auftretenden QPs entwickelt, der auf dem Prinzip der Dual Decomposition und einem semiglatten Newton-Verfahren basiert. Die Absicht dahinter ist es, Vorteile von Innere-Punkte Methoden bezüglich der Strukturausnutzung mit

den Warmstart-Fähigkeiten gewöhnlicher Active-Set Methoden für dichtbesetzte Probleme zu verbinden. Wir beweisen die Termination des Verfahrens nach endlich vielen Schritten und analysieren die Detektion von primal unzulässigen Problemen. Weiterhin wird das verwendete Verfahren zur Faktorisierung der auftretenden linearen Gleichungssysteme mit der aus der linear-quadratischen Regelung bekannten Riccati Rekursion identifiziert und numerische Teilaspekte des Algorithmus werden analysiert. Die gewonnenen Erkenntnisse wurden zur Entwicklung eines effizienten Open-Source Codes verwendet, der auf mehreren betrachteten Benchmarkproblemen den State-of-the-art strukturausnutzender QP-Löser um einen Faktor zehn und mehr verbessert.

Im weiteren Verlauf der Arbeit werden Möglichkeiten der Verallgemeinerung dieser Methode zur Lösung verteilter quadratischer Optimierungsprobleme und zur Lösung allgemeiner nichtlinearer Optimierungsprobleme untersucht. Zudem wird ein echtzeitfähiges nichtlineares MPC Schema auf Basis eines detaillierten Fahrzeugmodells zum Zwecke des autonomen Fahrens entwickelt. Dieses kann beispielsweise bei der Kollissionsverhütung unter schwierigen Straßen- und Witterungsverhältnissen oder beim zeitoptimalen Fahren Anwendung finden.

Beknopte Samenvatting

In deze thesis worden optimalisatie-gebaseerde methoden voor het regelen en meten van dynamische systemen voorgesteld. Inzake hun veelzijdigheid behoren optimalisatie-gebaseerde schema's tot de algemeenste en capabelste methoden, en ze vertonen een groeiende populariteit. Omdat een optimalisatieprobleem in real time moet opgelost worden is de computationele kost echter nog steeds een beperkend factor voor de uitbreiding van deze klasse van methoden.

Onze focus in deze thesis is op problemen met een lange horizon, die door lineaire en niet-lineaire dynamische systemen, en door beperkingen gekenmerkt kunnen zijn. Zulke problemen zijn van groot praktisch belang, bijvoorbeeld om stabiliteit te waarborgen, voor hoge regel- of meet-prestaties, of onder economische doelstellingen. We ontwikkelen algoritmen in deze thesis die kunnen profiteren van de inherente spaarsheid van deze klasse van problemen met een focus op parallellisme en actuele en toekomstige computer architecturen.

Met name wordt een hiërarchisch herlinearisatie schema voor niet-lineaire Model Predictieve Regelaars (Model Predictive Control, MPC) op basis van het befaamde Real-Time Iteration Scheme gegeven, die de kloof tussen lineaire en niet-lineaire MPC vermindert. Verder wordt een verbeterd Condensing algoritme van verminderde complexiteit voor reductie van de dimensionaliteit van de bekomen kwadratische programmeringsproblemen (Quadratic Programming Problem, QP) voorgesteld.

Bovendien wordt een nieuwe structuur exploiterend QP algoritme ontwikkeld, die gebaseerd is op dual decomposition en een semismooth versie van Newton's methode. Het doel is om het spaarsheid exploitatie vermogen van interior-point methoden te combineren met het warmstarting vermogen van gewone active-set methoden voor dense problemen. We analyseren een diversiteit van theoretische en numerieke aspecten van deze methode, en gebruiken het inzicht hieruit om een efficiënte implementatie van dit algoritme te ontwikkelen. De implementatie verbetert de state-of-the-art in QP oplossingen met factoren van meer dan tien

in rekentijd op een verscheidenheid van benchmark problemen en is publiekelijk verkrijgbaar als open source code als deel van deze thesis.

Verder worden generalisaties van dit dual Newton idee naar gedistribueerde QPs en algemene niet-lineaire problemen onderzocht. Op het einde van deze thesis wordt nog een high-fidelity real time feasible niet-lineaire MPC regeling voor het autonome rijden van personenwagens ontwikkelt. Deze regeling kan worden gebruikt voor het vermijden van ongevallen in gevaarlijke rijomstandigheden of voor het tijdsoptimaal rijden.

Contents

List of Figures

List of Tables

Abbreviations

AS	Active set (method)
BDF	Backwards differentiation formula
BVP	Boundary value problem
CG	Conjugate gradient (method)
CoG	Center of gravity
DDMS	Distributed Direct Multiple Shooting
DMS	Direct Multiple Shooting
DoF	Degree of freedom
DOP	Dynamic optimization problem
EKF	Extended Kalman filter
END	External Numerical Differentiation
EP	State and parameter estimation problem
FLOP	Floating point operation
iff	if and only if
IND	Internal Numerical Differentiation
IP	Interior point (method)
IVP	Initial value problem
KKT	Karush-Kuhn-Tucker (conditions)
LGPL	GNU Lesser General Public License
LTI	Linear time-invariant (system)
LTV	Linear time-varying (system)

MHE	Moving horizon estimation
MLE	Maximum likelihood estimator
MPC	Model predictive control
NLP	Nonlinear programming (problem)
OCP	Optimal control problem
ODE	Ordinary differential equation
PDE	Partial differential equation
QP	Quadratic programming (problem)
RTI	Real-Time Iteration (scheme)
SCP	Sequential convex programming
SLP	Sequential linear programming
SQP	Sequential quadratic programming
w.l.o.g.	without loss of generality

List of Symbols

General Notation

$\mathcal{E}$	Index set of equality constraints
$\mathcal{I}$	Index set of inequality constraints
$\cdot := \cdot$	Definition operator
$\cdot \cong \cdot$	Isomorphism relation between vector spaces/structural equivalence
$\cdot \uplus \cdot$	Disjoint union
$\cdot \equiv \cdot$	Congruence relation
$\cdot \odot \cdot$	Generalized scalar multiplication
$\cdot \oplus \cdot$	Generalized summation
$\mathbb{N}$	The space of natural numbers
$\mathbb{R}$	The space of real numbers

Dynamic Systems and Optimal Control

$\mathcal{J}$	Index set of point constraints
$\mathcal{U}$	Set of control functions
$\mathcal{X}$	Set of state trajectories
$\mathcal{S}$	Index set of discretization points on time horizon
$\mathcal{T}$	Time horizon of the dynamic system

$\boldsymbol{d}$	Vector of path constraint functions
$\boldsymbol{f}$	Continuous time system dynamics
$\boldsymbol{p}$	Time-constant parameters
$\boldsymbol{r}$	Vector of point constraints
$\boldsymbol{u}$	Vector of control functions
$\boldsymbol{x}$	Vector of differential states
N	Number of discretization intervals/stages along time horizon
n_d	Number of constraint functions
n_u	Number of control functions
n_x	Number of differential states
t_0	Beginning of time horizon
t_f	End of time horizon

Dynamic Estimation

$\boldsymbol{y}$	Vector of system output measurements
M	Number of measurement points
n_y	Number of measured outputs

Dual Newton Strategy

$\boldsymbol{\mathcal{G}}$	Dual gradient
$\boldsymbol{\mathcal{M}}$	Dual Hessian
$\mathcal{A}^*$	Optimal primal active set
$\boldsymbol{\mathcal{P}^*}$	Nullspace elimination matrix for active primal constraints
$\tilde{\boldsymbol{\mathcal{M}}}$	Regularized dual Hessian
A	Region of the dual space

Chapter 0

Introduction

A famous quote among mathematical optimization researchers reads “Every problem is an optimization problem”. Despite the natural bias of quoters and quotee, the core of this bold statement stresses the versatility of mathematical optimization. Whenever an (approximate) quantitative representation of a phenomenon is available, tools from mathematical optimization can be used to analyze the available data and draw conclusions — whether it is directly in the form of computing optimal configurations of modifiable variables for a certain task, or indirectly via, for example, a regression or another form of computational simplification that makes a problem more comprehensible.

This versatility stretches out to dynamic systems, where, in replacement of simplistic controller and observer designs, any kind of model granularity up to first principles models from physics, chemistry, biology, economics, or sociology (to name only a few areas) can be employed in the mathematical optimization framework to pursue an ultimate objective. Even though theoretical guarantees (regarding stability, for example) are sometimes harder to obtain in more complex model settings, the additional insight from the consideration of detailed system knowledge may often result in a better control and estimation performance, or in a higher quality of the analysis.

Static optimization, i.e., the solution of mathematical optimization problems without explicitly considering dynamic evolutions, can be considered well established for many applications. To give a few examples, just think of the calculation of set-points for optimal operation of chemical reactors, the arrangement patterns of goods in a warehouse, or a regression analysis for the identification of system characteristics. Often, however, the formulation of a static problem itself may be a simplification. Instead of calculating a

set-point for a reaction, for example, the ultimate objective in many cases may be of a different flavor — maximum yield, minimum waste, or highest (energy-) efficiency to only name a few. Since many phenomena are dynamic by nature, taking a model of the dynamic evolution into account may result (by way of computing optimal trajectories instead of steady states, for example) in an increased performance regarding the ultimate objective.

A prerequisite for this to work well is the existence of models that characterize the behavior of the underlying dynamic system rather accurately. But even the most accurate models will exhibit a certain mismatch with reality, the so-called *model-plant mismatch*, for almost every practical problem. A powerful concept to deal with this mismatch is the concept of *feedback* or, coined at the optimization lingo, the concept of *online optimization*. Instead of obtaining all data in bulk and solving the optimization problem *offline*, solely based on this data, it is the idea of these concepts to re-solve the optimization problem *online*, i.e., while the dynamic process is "running" (evolving), based on newly acquired data from the operation of the process. This allows to react to disturbances in a control task, or to adjust the result of a regression based on the latest observations.

Online-optimization based feedback and regression is particularly interesting as it allows to combine three very important aspects around the operation and analysis of a dynamic process. First, it takes the dynamically changing characteristics of the system explicitly into account, in form of a model. Second, it allows to flexibly formulate objectives that are of primary, original interest. And third, it allows to directly formulate restrictions, on the operation of the process for example, which then have to be respected in the pursuit of the primary objective.

The main limitation of online-optimization based feedback and regression approaches, however, is the computational burden associated with the solution of the optimization problem in real-time. It is the central goal of this thesis to push that frontier, by proposing both new and enhanced computational methods for the solution of optimization problems of dynamic systems in real-time. The instruments to achieve this goal are bifocal: Above all, characteristic structures that present themselves rather generically in optimization problems of dynamic systems are identified and exploited. Then, in the design and analysis of tailored algorithms, a particular emphasis is put on implementability on multi- and many-core parallel computational architectures, which become increasingly available even on cheap and embedded devices, and which are expected to represent the largest origin of growth in computational power over the coming years (or even decades) due to physical limitations on the maximum achievable clock speed.

The following figure visualizes more specifically a few of the keywords which are relevant for this thesis:

In particular, we focus our considerations on nonlinear systems, for which we apply a multiple shooting discretization to obtain a structured nonlinear programming problem. We develop inexact variants of sequential quadratic programming schemes for the solution of these problems in real-time and then focus on a computational bottleneck which is particularly pronounced in parallel computing environments[1] — the solution of the underlying quadratic programming problems.

Our emphasis is on gaining particular efficiency for problems with long prediction horizons, i.e., problems with many discretization points in the time domain. Such problems are desirable to be formulated in practice for a variety of reasons:

- Important among these reasons are safety concerns. For an illustration just think of a vehicle being operated autonomously at a high speed. It is imperative that the vehicle is always operated at a state such that it can safely come to a complete stop without straying off the track. Due to the involved nonlinear dynamics, the formulation of so-called *terminal regions* in state space (a common practice in control engineering to issue similar guarantees) may become very involved and may lead to overly conservative control behavior. On the other hand, a long prediction horizon can equip the controller with sufficient freedom of action while being still able to

[1] This is due to the fact that in so-called direct simultaneous methods like multiple shooting (and even more so in collocation) the linearization step is naturally parallelizable.

decelerate the vehicle completely within the foreseeable future (i.e., the predicted horizon). Also more generic and formal stability guarantees can be established without any requirement of terminal cost or constraint for sufficiently long prediction horizons, cf. [Grü13].

- Along similar lines can be the economic operation of a system. Let us think of a robotic manipulator for illustration purposes, that is to reach a certain final position at a fixed point in time, consuming the lowest possible amount of energy. Whenever the dynamics of this system do not permit a (tractable) closed-form representation of the so-called *cost-to-go* (cf. Section 1.2) — which typically is the case — considering the full prediction horizon until the time of the desired arrival may be required to ensure optimality of the operation with respect to the economic objective.

- The quality of online parameter estimation (identification) results depends largely on the sensitivity of the system with respect to the parameter (observability) and on the noise on the measurement data. If the sensitivity is small and/or the measurement noise is strong, a large number of measurement points along the time grid need to be considered for a reliable estimate.

- Direct methods, in general, only verify feasibility with respect to path constraints (e.g., state bounds) on the discretization grid. For a strict adherence to path constraints, a large number of discretization points along the prediction horizon is therefore essential. Even methods that make use of a continuous system trajectory representation (like collocation methods, for example) can only guarantee adherence to the path constraints up to the accuracy of the continuous representation between the checkpoints and therefore also require a fine discretization grid for reliable constraint satisfaction, which, in turn, results in a long band-structure of the resulting nonlinear programming problem.

While being practically relevant for the above reasons, optimization problems of dynamic systems on long prediction horizons are also particularly challenging due to the entailed large numbers of optimization variables. This sparks the demand for fast solution algorithms.

On the upside, optimization problems of dynamic systems exhibit very characteristic, band-structured sparsity patterns, which become increasingly pronounced with growth of the length of the prediction horizon. This presents an opportunity for research on efficient, sparsity exploiting algorithms that render also long-horizon problems tractable in real-time.

0.1 Contributions

Before detailing the structure of this thesis, let us accentuate a few constitutive contributions of this thesis:

A hierarchical update scheme for nonlinear MPC In Section 2.4, we present an algorithmic framework that extends the Real-Time Iteration scheme from [DBS+02] and the Multi-Level Iteration scheme from [BDKS07] to a more general concept that allows to tackle nonlinear model predictive control (MPC) and moving horizon estimation (MHE) problems by a combination of linear MPC and hierarchical re-linearization updates that can be triggered flexibly in accordance with the available computational infrastructure. A corresponding, adapted condensing procedure accounting for the flexibility of the updates is introduced in Section 3.3.

A structure-exploiting, parallelizable active-set method for QP We introduce a new algorithm for the solution of band-structured, strictly convex quadratic programming (QP) problems, as they appear both in linear and nonlinear MPC and MHE, in Chapter 4. The algorithm is based on a two-level approach, solving parametric, reduced-size inequality constrained QPs for each discretization stage and a dual unconstrained consensus problem on top of that. Numerical details regarding the efficient implementation are given and finite convergence guarantees are established. We give an in-depth analysis of the specific dual function at hand that leads to rigorous algorithmic infeasibility certificates. Furthermore, an $\mathcal{O}(\log N)$ parallelizable factorization algorithm, where N is the length of the prediction horizon, is proposed for the consensus problem, and the warmstarting capabilities (which are significantly superior to state-of-the-art structure-exploiting interior-point methods) of the algorithm are illustrated.

An efficient open-source QP solver for MPC and MHE As part of this thesis, the open-source structure-exploiting QP solver qpDUNES has been developed. This software is based on the ideas developed in Chapter 4 and is released as free open-source software under the GNU LGPL license [GNU11]. A presentation of the software design is given in Chapter 6. This package has already been successfully used in a variety of applications from control and estimation.

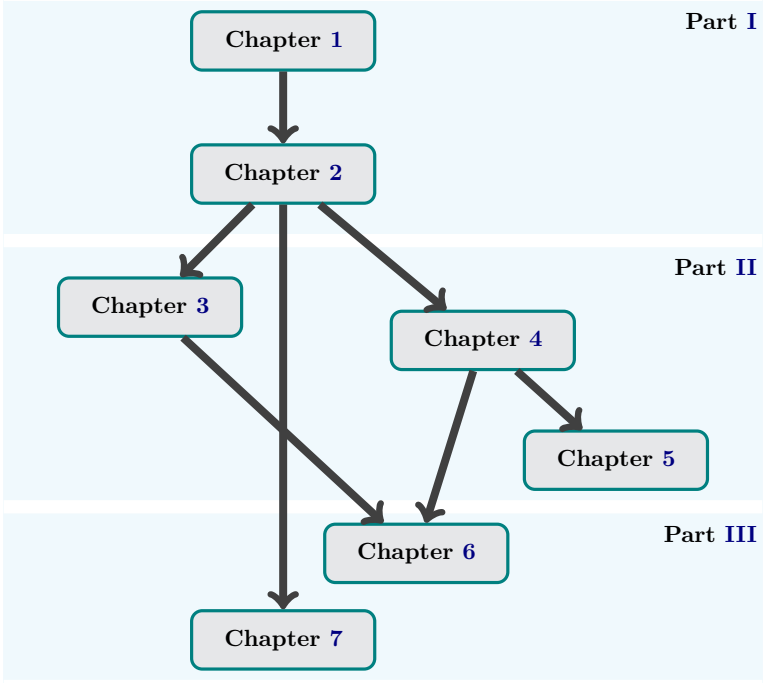
A structure-exploiting, parallelizable nonlinear programming method The ideas from Chapter 4 are carried further in Section 5.2 to be able to treat also non-

convex structured optimization problems by modifying the stage subproblems in an ADMM-inspired augmented-Lagrangian fashion.

A real-time feasible high-fidelity nonlinear MPC for autonomous driving In Chapter 7, we consider the application problem of autonomous driving in challenging situations, such as real-time collision avoidance or time-optimal driving. We present a detailed vehicle model that is usable for nonlinear model predictive control in real-time and includes significantly more dynamic effects than the previous state-of-the-art. A spatial representation of the vehicle dynamics permits a straightforward modeling of road boundaries and obstacles.

0.2 Outline and Structure of this Thesis

The main part of this thesis is structured in three parts, which are further subdivided in seven chapters; this introduction and a conclusion complement the main part to a total of nine chapters. The rough dependency structure of the individual chapters can be seen from the following sketch:



Part I is about control, estimation, and the nonlinear programming perspective of it. Therein Chapter 1 introduces the control and estimation setting we consider in this thesis, details the treatment of these problems by the framework of nonlinear programming, and reviews the required theoretical foundations. Chapter 2 gives the online perspective on the dynamic optimization problems at hand, in form of introducing model predictive control (MPC) and moving horizon estimation (MHE). The fundamental concept of the Real-Time Iteration scheme is reviewed, and algorithmic extensions thereof are presented, which result in a flexible framework bridging linear and nonlinear MPC.

Part II is about identifying and exploiting structures that arise in quadratic programming (QP) subproblems from the application of the above algorithms applied to dynamic optimization problems. In particular, Chapter 3 points out these structures and presents the classical condensing approach — however, in a revised fashion that results in a more efficient condensing algorithm. Chapter 4 introduces an alternative to the condensing approach for solving the structured (strictly convex) quadratic programming problems: a two-level sparse QP algorithm that performs Newton steps on the dual problem, but maintains the flavor of an active-set method and is highly parallelizable. Numerical aspects, theoretical issues, and the efficient application in MPC and MHE are discussed. Chapter 5 takes the lessons learned from Chapter 4 one step further and proposes ideas of how to apply this algorithmic frame (a) in more generic coupling structures, as they appear, for example, in distributed optimization-based control and estimation, and (b) in a more general problem setting that allows to treat convex programming problems, positive semidefinite QPs, and even general non-convex nonlinear programming problems (NLPs).

Part III is about software, benchmarking and applications. Chapter 6 introduces the open-source band-structured QP solver qpDUNES, which was developed as part of this thesis, and demonstrates its efficiency in benchmarks from linear and nonlinear MPC against other state-of-the-art QP solving approaches. Chapter 7 considers the application case of autonomous driving and presents an encompassing vehicle model based on a spatial reformulation of the dynamic system that models safety-critical driving behavior with significantly higher fidelity than the previous state-of-the-art, but is still comfortably real-time feasible for control and estimation.

Part I

Model-Based Control and Estimation

Chapter 1

Nonlinear Dynamic Optimization

1.1 Control and Estimation of Dynamic Systems

Let us begin by giving a formal introduction to the considered problem classes in the following.

1.1.1 Controlled dynamic systems

The common core of all problems central to this thesis are dynamic systems. These systems can be of physical, chemical, biological, economic or social origin, to name only the most common sources (cf. [Lue79]). The term *dynamic* therein refers to time-changing characteristics which are the dominant feature in all the considered problems.

A common way to represent a dynamic system mathematically on a continuous time horizon $\mathcal{T} := [t_0, t_f] \subseteq \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ is through its so called *(differential) state* denoted by $\boldsymbol{x} \in \mathcal{X} := \{\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_x}\}$. We assume that the considered systems can be influenced by some sort of manipulation or actuation that is denoted mathematically by a vector of *control functions* $\boldsymbol{u} : \mathcal{T} \to \mathbb{R}^{n_u}$, which we assume to be measurable and of bounded variation, i.e., $\boldsymbol{u} \in \mathcal{U} := \{\boldsymbol{u} : \mathcal{T} \to \mathbb{R}^{n_u} \mid \boldsymbol{u} \text{ measurable of bounded variation}\}$. The law determining the temporal evolution of our dynamic system is commonly

represented as a coupled system of *ordinary differential equations* (ODEs)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}. \tag{1.1}$$

The vector $\boldsymbol{p} \in \mathbb{R}^{n_p}$ denotes *time-constant model parameters* that influence the *system dynamics* $\boldsymbol{f} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ (also simply known as *the right-hand-side function*).

Often, we simply characterize a dynamic system by $\boldsymbol{f}$, when $\mathcal{T}$, $\mathcal{X}$, and $\mathcal{U}$ are canonical, clear from the context, or merely irrelevant. Some basic systems theoretic definitions in Sections 1.1.2 and 1.1.3 however will require the following, more formal definition based on [Son98]:

Definition 1.1 (Dynamic system) Formally, a tuple $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ is called a *(dynamic) system.* ⌟

We address several aspects of the dynamic system (1.1) that relate to computational properties and problem formulations in the following.

Assumption 1.2 Throughout this thesis we assume the system dynamics $\boldsymbol{f} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ to be piecewise Lipschitz continuous in $\boldsymbol{x}(t)$ and continuous in t and $\boldsymbol{u}(t)$.

Assumption 1.2 is crucial, as it forms the basis for existence and uniqueness[1] guarantees of $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_x}$, our representation of the dynamic system. These guarantees can be established by the well-known Picard-Lindelöf theorem and the Carathédory theorem, but may depend on the actual control function representation and are therefore omitted in this general setting. It is noted however, that for the choice of a piecewise polynomial control parameterization (as introduced with the general algorithmic setting in Section 1.2.1) the Picard-Lindelöf theorem is (at least) piecewise applicable on each interval where $\boldsymbol{u}(t)$ is continuous. The corresponding, uniquely determined state trajectory is then denoted by $\boldsymbol{x}(t; \boldsymbol{x}(t_0) = \hat{\boldsymbol{x}}_{\boldsymbol{0}}, \boldsymbol{u}, \boldsymbol{p})$, where each of the conditional arguments may be omitted if they are clear from the context or irrelevant.

With regard to the derivative-based numerical methods considered and developed within this thesis, we additionally demand

Assumption 1.3 The system model representation $\boldsymbol{f} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ is assumed to be twice continuously differentiable with respect to $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{p}$, if not stated otherwise.

[1] Note that for uniqueness we particularly need a fixed initial condition of the system, which is typically given by $\boldsymbol{x}(t_0) = \hat{\boldsymbol{x}}_{\boldsymbol{0}}$, where $\hat{\boldsymbol{x}}_{\boldsymbol{0}} \in \mathbb{R}^{n_x}$.

Remark 1.4 In some contexts (most notably in estimation) it is notationally favorable to express the system behavior explicitly at discrete time points by the law

$$\boldsymbol{x}(t_{k+1}) = \boldsymbol{F_k}(\boldsymbol{x}(t_k), \boldsymbol{u}(\cdot), \boldsymbol{p}) \qquad \forall\, k \in \{0, \dots, N-1\} =: \mathcal{S}_N.$$

Due to uniqueness (and existence, of course) of the system representation $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_x}$, the *state transition mappings* $\boldsymbol{F_k} : \mathbb{R}^{n_x} \times \mathcal{U} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x},\ k \in \mathcal{S}_N$ are an equivalent representation of the system behavior that is given in explicit form by

$$\boldsymbol{F_k}(\boldsymbol{x}(t_k), \boldsymbol{u}(\cdot), \boldsymbol{p}) = \boldsymbol{x}(t_k) + \int_{t_k}^{t_{k+1}} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p})\, \mathrm{d}\tau \qquad \forall\, k \in \mathcal{S}_N.$$

The integral is understood component-wise in this context.

Remark 1.5 The parameter vector can be hidden by appending n_p entries to the state that are uniquely determined over the full time horizon $\mathcal{T}$ by

$$\dot{x}_{n_x+i}(t) = 0 \qquad \forall\, i \in \{1, \dots, n_p\},\ t \in \mathcal{T},$$

$$x_{n_x+i}(t_0) = p_i \qquad \forall\, i \in \{1, \dots, n_p\}.$$

Remark 1.6 System (1.1) can be transformed into an *autonomous* system

$$\dot{\boldsymbol{x}}(t) = \tilde{\boldsymbol{f}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) \qquad \forall\, t \in \mathcal{T},$$

i.e., one without explicit time-dependency, by introducing one additional state

$$\dot{x}_{n_x+1}(t) = 1 \qquad \forall\, t \in \mathcal{T},$$

$$x_{n_x+1}(t_0) = t_0.$$

We note that both reformulations above are mainly introduced for notational convenience. In an algorithmic implementation, a separate treatment of states, parameters, and time-dependency may lead to increased computational efficiency.

Remark 1.7 Even though dynamic system (1.1) is stated for a fixed time horizon $\mathcal{T} = [t_0, t_f]$, a time horizon of variable length $T := t_f - t_0$ can easily be considered by introducing a parameterization σ of t. We define $t(\sigma) := t_0 + \sigma \cdot T,\ \sigma \in [0, 1]$ and restate (1.1) accordingly as

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t(\sigma), \boldsymbol{x}(t(\sigma)), \boldsymbol{u}(t(\sigma)), \boldsymbol{p}) \cdot \frac{\partial t(\sigma)}{\partial \sigma} \\ &= \boldsymbol{f}(t(\sigma), \boldsymbol{x}(t(\sigma)), \boldsymbol{u}(t(\sigma)), \boldsymbol{p}) \cdot T, \qquad \sigma \in [0, 1]. \end{aligned}$$

In particular, some applications may require a generalization of the notion of time in the dynamic system context to a *pseudo-time* or simply a different *independent variable* of the ODE system in lieu of time.

As a final remark to this brief section on dynamic systems, let us note that our understanding of a dynamic system as an ODE through (1.1) is not exhaustive in the domains of optimal control and dynamic estimation. Several extensions are conceivable:

- Some mechanical or chemical systems may require the generalization of the ODE system to a system of *differential algebraic equations* (DAEs). The works [Lei99] and [Die02] elaborate on the treatment of DAEs with numerical methods that are compatible with the ones presented in this work; [Lei99] covers offline aspects, while [Die02] focuses on online computations.
- Systems that feature switched inputs (i.e., inputs that can only attain values from a discrete set) lead to *mixed-integer optimal control problems* (MIOCPs). Computational methods for such problems that build on the same algorithmic core as this thesis and are therefore largely compatible with the algorithms presented here can be found in [Sag05] for offline problems, and in [Kir11] for online problems.
- For phenomena that have a dynamic evolution not only with respect to time, but also with respect to space (or, in general, that have coupled multidimensional dynamics), *partial differential equation* (PDE) models are required. [Pot11b] discusses a computational approach that is related to the basis of the methods discussed here (direct multiple shooting discretization).
- Some systems with uncertain dynamic behavior can be modelled by *stochastic differential equations* (SDEs). We refer to [HS14] and the references therein for an overview of related numerical methods treating SDEs.

1.1.2 Optimal control

Since the control functions in dynamic system (1.1) can be manipulated and are thus degrees of freedom, one is naturally interested in choosing them in an optimal fashion, leading to the problem of *optimal control*. Here, optimality is desired with respect to some performance index $\Phi : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^{n_p} \to \mathbb{R}$, that

maps (at least locally) favorable state and control trajectories onto smaller values than less favorable ones.

A common, generic form of the performance index is the so called *Bolza-type objective*, which consists of an integral term and an end-point contribution:

$$\Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}) = \int_{t_0}^{t_f} \ell(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) \, \mathrm{d}t + e(t_f, \boldsymbol{x}(t_f), \boldsymbol{p}).$$

A performance index that only consists of the integral part is usually referred to as a *Lagrange-type objective*, while a performance index that only features an end-point contribution is known as a *Mayer-type objective*.

In the context of model predictive control (MPC) so called *least-squares objectives*, given by

$$\Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}) = \int_{t_0}^{t_f} \|\boldsymbol{\ell}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\|_2^2 \, \mathrm{d}t + \|\boldsymbol{e}(t_f, \boldsymbol{x}(t_f))\|_2^2. \tag{1.2}$$

are occurring frequently, for example in stabilizing control, where the control objective is to drive a system to an equilibrium (cf. Definition 1.12), or more generally in tracking control, where any predefined *reference trajectory* is tracked. This special case is displayed separately since very powerful algorithms exploiting the particular structure of this objective exist (most notably the so-called *generalized Gauss-Newton method*, cf. Section 1.4.4).

In the domain of optimal control the following definitions are essential and give a handle on the well-definedness of the control problem. Our formulations are based on [Son98, RM09].

Definition 1.8 (Reachability) Let $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ be a dynamic system and let $\boldsymbol{x}$ be implicitly defined by $\boldsymbol{f}$ through (1.1) and a known initial state. A state $\boldsymbol{x_2} \in \mathbb{R}^{n_x}$ is said to be *reachable* on $\mathcal{T}$ from the *event* $\boldsymbol{x}(t_1) = x_1$, where $t_1 \in \mathcal{T}$ and $x_1 \in \mathbb{R}^{n_x}$, if either

a) there is a $t_2 \in \mathcal{T}$, $t_2 \geq t_1$ and a $\boldsymbol{u} \in \mathcal{U}$ such that $\boldsymbol{x}(t_2) = \boldsymbol{x_2}$, or

b) $t \in \mathcal{T}$ for all $t > t_1$ and there is a $\boldsymbol{u} \in \mathcal{U}$ such that $\lim_{t \to \infty} \boldsymbol{x}(t) = \boldsymbol{x_2}$. ⌟

Definition 1.9 (Controllability) We say a dynamic system $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ with $\boldsymbol{x}$ implicitly defined by $\boldsymbol{f}$ through (1.1) and a known initial state is *controllable* on $\mathcal{T}$, if for every event $\boldsymbol{x}(t_1) = \boldsymbol{x_1}$, with $t_1 \in \mathcal{T}$ and $\boldsymbol{x_1} \in \mathbb{R}^{n_x}$, and $\boldsymbol{x_2} \in \mathbb{R}^{n_x}$ we have that $\boldsymbol{x_2}$ is reachable from $\boldsymbol{x}(t_1) = \boldsymbol{x_1}$. ⌟

Assumption 1.10 Throughout this thesis we assume in the context of optimal control that the dynamic system $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ defined by (1.1) is controllable on $\mathcal{T}^\infty := [t_0, \infty[$, if not explicitly mentioned otherwise.

Remark 1.11 We note that for systems theoretic considerations weaker assumptions than 1.10, such as splitting (1.1) in controllable and uncontrollable modes, may be advantageous. Since the focus of this thesis however lies on computational methods, we remain with Assumption 1.10 for reasons of simplicity and refer to [Son98] for complementary reading.

We further have the important notion of an *equilibrium state*, which, colloquially, is a state of the system that, once attained, is never left, cf. [Lue79].

Definition 1.12 (Equilibrium) We call a state $\bar{\boldsymbol{x}} \in \mathbb{R}^{n_x}$ an *equilibrium* (sometimes also *steady state*), if there is a control law $\boldsymbol{u}^*(t;\bar{\boldsymbol{x}}) \in \mathcal{U}$ such that

$$\boldsymbol{f}(t, \bar{\boldsymbol{x}}, \boldsymbol{u}^*(t;\bar{\boldsymbol{x}}), \boldsymbol{p}) = \boldsymbol{0} \qquad \forall\, t \in \mathcal{T}. \qquad \lrcorner$$

Often in control theory, an equilibrium is a desirable state. To characterize convergence to an equilibrium, we make use of the following notion.

Definition 1.13 (Asymptotic stability) For a given control law $\boldsymbol{u}^*(t) \in \mathcal{U}$, we call a dynamic system $\boldsymbol{f}$ *asymptotically stable* at an equilibrium $\bar{\boldsymbol{x}}$ if for its corresponding state trajectory $\boldsymbol{x}$ defined implicitly by (1.1) it holds that both

a) for all $\epsilon > 0$, there is a $\delta > 0$, such that $\|\boldsymbol{x}(t_0) - \bar{\boldsymbol{x}}\| < \delta$ implies $\|\boldsymbol{x}(t) - \bar{\boldsymbol{x}}\| < \epsilon$ for all $t \geq t_0$ (*Lyapunov stability*), and

b) there is a $\eta > 0$, such that $\|\boldsymbol{x}(t_0) - \bar{\boldsymbol{x}}\| < \eta$ implies $\lim_{t\to\infty} \|\boldsymbol{x}(t) - \bar{\boldsymbol{x}}\| = 0$ (*local attractiveness*). $\lrcorner$

We note that Definition 1.13 is only one out of a large variety of characterizations of stability. We refer to [RM09] and the references therein for an overview.

1.1.3 Dynamic estimation

The dynamics of a real-world system, and even the state the system is currently in might not always be (perfectly) known. For illustration just think of a pressurized tank reactor in chemical engineering. While the precursor volumes might be known well, the reaction rate might only be known approximately or might be subject to unmodeled influences, and in consequence the current product volumes, which could be part of the state, are unknown as they cannot be measured during the reaction.

In situations like this, one is interested in *estimating* the current system state $\boldsymbol{x}(t_f)$ and system parameters $\boldsymbol{p}$ from observations $\boldsymbol{y_k} \in \mathbb{R}^{n_y}$ made at time t_k

for each $k \in \{1, \ldots, M\}$. Typically, the estimation performance index is of the form

$$\Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}) = \ell_0(\boldsymbol{x}(t_0), \boldsymbol{p}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) + \sum_{k=1}^{M} \ell_k(\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), \boldsymbol{p}, \boldsymbol{y_k}),$$

where the ℓ_0 term is used to include a priori information $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{p}}$.

A special case, that is however frequently appearing, is the so-called *least-squares estimator*, which is characterized by the performance index

$$\Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}) = \left\| \begin{bmatrix} \boldsymbol{x}(t_0) - \bar{\boldsymbol{x}} \\ \boldsymbol{p} - \bar{\boldsymbol{p}} \end{bmatrix} \right\|_{\boldsymbol{P}}^2 + \sum_{k=1}^{M} \|\boldsymbol{y_k} - \boldsymbol{h}(\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), \boldsymbol{p})\|_{\boldsymbol{V_k}}^2 .$$

In this context $\boldsymbol{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$ denotes the so-called *output function*, which relates dependent system variables to observable system quantities, the so-called *outputs*. We assume that $\boldsymbol{h}$ is (at least) twice continuously differentiable for algorithmic reasons. The shorthand $\|\boldsymbol{a}\|_{\boldsymbol{A}} := \sqrt{\boldsymbol{a}^\top \boldsymbol{A}\, \boldsymbol{a}}$ denotes the norm induced by positive semidefinite, symmetric weighting matrices $\boldsymbol{P} \in \mathbb{R}^{(n_x+n_p)\times(n_x+n_p)}$ and $\boldsymbol{V_k} \in \mathbb{R}^{n_y \times n_y}$. We refer to [RM09] for additional details on the formulation of estimation problems. In particular, so-called *process noise* variables that allow to model uncertainty in the system dynamics are introduced there. We address the issue of how to incorporate these terms into the vectors of discretized control variables in Section 2.2.

Definition 1.14 (Distinguishability) Let $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ be a dynamic system with the output function $\boldsymbol{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$, and let $\boldsymbol{x}$ be implicitly defined by $\boldsymbol{f}$ through (1.1). A pair of events $(\boldsymbol{x}(t_1) = \boldsymbol{x_1}, \boldsymbol{p_1})$ and $(\boldsymbol{x}(t_1) = \boldsymbol{x_2}, \boldsymbol{p_2})$, where $t_1 \in \mathcal{T}$, is said to be *distinguishable* on $\mathcal{T}$, if there is at least one control input $\tilde{\boldsymbol{u}} \in \mathcal{U}$ such that there exists a $t \in [t_0, t_1] \subseteq \mathcal{T}$ with

$$\boldsymbol{h}(\boldsymbol{x}(t; \boldsymbol{x}(t_1) = \boldsymbol{x_1}), \tilde{\boldsymbol{u}}(t), \boldsymbol{p_1}) \neq \boldsymbol{h}(\boldsymbol{x}(t; \boldsymbol{x}(t_1) = \boldsymbol{x_2}), \tilde{\boldsymbol{u}}(t), \boldsymbol{p_2}). \qquad \lrcorner$$

Definition 1.15 (Observability) We say a dynamic system $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ with the output function $\boldsymbol{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$ and $\boldsymbol{x}$ implicitly defined by $\boldsymbol{f}$ through (1.1) is observable if all distinct pairs of events $(\boldsymbol{x}(t_f) = \boldsymbol{x_f}, \boldsymbol{p_1})$ and $(\boldsymbol{x}(t_f) = \boldsymbol{x_2}, \boldsymbol{p_2})$ are distinguishable on $\mathcal{T} = [t_0, t_f]$. $\lrcorner$

Assumption 1.16 We assume throughout this thesis in the context of dynamic estimation that a dynamic system $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ defined by (1.1) with its corresponding output function $\boldsymbol{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$ is observable on $\mathcal{T}^\infty :=]-\infty, t_f]$, if not explicitly mentioned otherwise.

Assumption 1.16 is rather weak, and only ensures well-definedness in the sense that the estimation task is not completely utopian. Instead, due to continuity of $\boldsymbol{x}$ and $\boldsymbol{h}$, we can expect that the current state and parameter vector of the system can be recovered for a sufficiently fine measurement grid and a sufficiently long measurement history, if the system is excited in the right way (i.e., if the right control inputs were chosen). For an overview of further observability characterizations and related concepts we refer to [RM09] and [Son98]. In the Section 2.2 we will present a more practical definition of observability based on the actually available measurements.

1.1.4 Dynamic optimization problems

The combination of a performance index as defined in Sections 1.1.2 and 1.1.3 and a constraining dynamic system (1.1) leads to the notion of a *dynamic optimization problem* (DOP).

Definition 1.17 (Dynamic Optimization Problem) The standard form of a dynamic optimization problem assumed throughout this thesis reads

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}} \quad \Phi(\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}) \tag{DOP1}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}) \qquad \forall\, t \in \mathcal{T} \tag{DOP2}$$

$$\boldsymbol{0} \geq \boldsymbol{d}(t,\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}) \qquad \forall\, t \in \mathcal{T} \tag{DOP3}$$

$$\boldsymbol{0} = \boldsymbol{r_i}(\boldsymbol{x}(t_i),\boldsymbol{p}) \qquad \forall\, i \in \mathcal{J}. \tag{DOP4}$$

The objective (DOP1) strives to minimize the performance index $\Phi : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^{n_p} \to \mathbb{R}$ for the dynamic system (DOP2), while respecting path constraints $\boldsymbol{d} : \mathcal{T} \times \mathcal{X} \times \mathcal{U} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ through (DOP3) and point constraints $\boldsymbol{r_i} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{r_i}}$, $\forall\, i \in \mathcal{J}$ at discrete time points $t_i \in \mathcal{T}$ from a finite index set $\mathcal{J} = \{1,\ldots,n_{\mathcal{J}}\}$ through (DOP4). ⌟

Depending on whether the objective functional is of control or estimation nature, we will call the respective instance of (DOP) *optimal control problem* (OCP) or *state/parameter estimation problem* (EP). In an optimal control context, usually the control functions are regarded as the actual degrees of freedom in the sense of the optimization, while the state trajectories are seen as dependent (infinite dimensional) variables and the parameters are usually assumed to be fully known. In an estimation context on the other hand, one usually sees parameters, terminal (current) state and possibly the noise vectors as degrees of freedom in the sense of the optimization, while states and controls

are dependent variables. We will see in Section 1.2 however that the differences between control and estimation problems become rather subtle on the numerical level when applying direct solution methods, which is why we introduce the major algorithmic concepts more generally for dynamic optimization problems.

Some previous works (e.g., [Lei99, Die02, Sag05]) consider a to some extent more general setting than Definition 1.17, that allows to formulate so-called *multi-stage optimal control problems*, which may appear naturally, e.g., in some applications from chemical engineering. Other applications, like the optimization of flight orbits, may require an initial-value-free *periodic solution* of an OCP, cf. [Hou07, Hou11]. Problems that fall in these more general classes can, in principle, still be modeled by (DOP), but at the cost of introducing additional auxiliary states. An efficient solution algorithms for such problems, which would need to treat the arising structures specifically, is however beyond the scope of this thesis.

We conclude this section with an assumption that is relevant for the numerical methods to be developed:

Assumption 1.18 In regard to derivative-based numerical methods, we also assume Φ, $\boldsymbol{c}$, and $\boldsymbol{r_i}$, $i \in \mathcal{J}$ to be twice continuously differentiable.

1.2 Numerical Methods for Dynamic Optimization

Dynamic optimization problems of the form (DOP) contrast standard optimization problems (e.g., those classically formulated in the domain of operations research) through two distinctive features. Firstly, the dynamic system constraints (DOP2) define the state $\boldsymbol{x}$ only implicitly through a coupling relation of the state and its derivative; in general, we cannot expect to find an exact closed form representation of $\boldsymbol{x}$. And secondly, since the optimization variables in Formulation (DOP) live in function space, our dynamic optimization problem is an *infinite dimensional optimization problem*, and therefore requires additional effort to be tractable by present-day numerical computing architecture.

Commonly, the literature distinguishes between three fundamentally different approaches to tackle (DOP), cf. Figure 1.1. We only give a very brief overview here, and refer to [Bie10, Sag05, And13] for further reading.

Dynamic programming approaches base on Bellman's principle of optimality [Bel57] and proceed by propagating a so-called *cost-to-go* function backwards in time. This procedure generally leads to the *Hamilton-Jacobi-Bellman partial differential equation* or to a dynamic programming recursion if the propagation

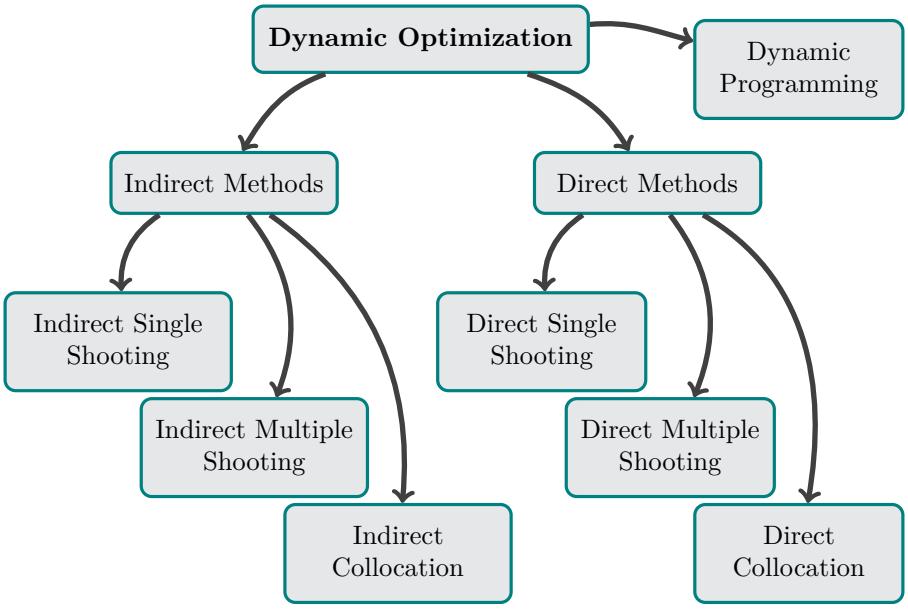


Figure 1.1: Categorization of solution methods for dynamic optimization problems.

is performed in discrete time steps. Since for the application to general dynamic systems a tabulation of the cost-to-go function on a state space discretization grid is required, this approach is well-known to suffer from the so-called curse of dimensionality. Except for special cases (cf. Sections 2.1 and 2.2) dynamic programming approaches are therefore only applicable to systems with few states. We refer the interested reader to [Loc01, Ber07] for further details on this approach.

The so-called *indirect approach* (also known as *first-optimize-then-discretize* scheme) is to analytically formulate the (infinite dimensional) first-order necessary optimality conditions of (DOP) — typically by way of the so-called *maximum principle*, cf. [PBGM62, BH75] — and to then (in general) solve the resulting *multi-point boundary value problem* (BVP) numerically (cf., e.g., [Osb69, Boc78]). While the accuracy of the solution obtained by the indirect approach directly depends on the accuracy of the BVP solution, this approach has the significant disadvantage that mathematical insight is required to formulate the optimality conditions analytically for each problem (or even each modification) individually. Indirect methods can, however, play an important role for theoretical considerations.

As an alternative, the so-called *direct approach* — following a *first-discretize-then-optimize* scheme — applies a discretization procedure to transform (DOP) into a finite-dimensional *nonlinear programming problem* (NLP), which is subsequently solved via its (finite-dimensional) optimality conditions. We note that, even though the direct approach only finds an optimum to a seemingly different problem, namely a discretized version of (DOP), it was shown (for example in [Fre79]) that for decreasing discretization error the discrete solution converges to a solution of the continuous problem in the primal and dual variables.

This thesis is focused on direct methods, and in particular variants of Bock's Direct Multiple Shooting (DMS) method [BP84]. While an in-detail comparison of direct and indirect methods for dynamic optimization is beyond the scope of this thesis, we state motivating cornerstones in the following. Slightly more elaborate discussions on this topic, together with further references, can be found in [Sag05] and [Kir11].

Direct methods have the advantage that they can be fully automated and, in principle, can tackle any dynamic optimization problem that is casted into their admissible standard form without the need for additional expert insight[2]. This is viable when trying to address several applications at the same time, for each of which dynamic system models might have to be adapted frequently as a result of the observed real-world system performance[3].

Since the driving spirit behind this thesis are real-time applications, the focus needs to be on methods that can repeatedly solve dynamic optimization problems at very high rates. These problems are typically very similar in the sense that their solutions are proximal in an appropriately defined metric space of admissible solutions – a fact which particularly DMS methods can exploit better than many other methods. DMS is also a particular form of *variable lifting*, which might reduce the problem nonlinearity and thus increase the domain of fast local convergence (cf. [AD10a, Pot11a]).

Finally, among direct methods, DMS can be seen as a hybrid approach between *single shooting methods* on the one hand, which, per iteration, are characterized by one (expensive) function evaluation but rather low costs on the actual NLP solution end, and *collocation methods* on the other hand, where, simply put, function evaluations are very cheap and almost all computational effort is shifted to the NLP. DMS methods require a medium number of function evaluations (along the time horizon) per iteration, and result in a medium-sized NLP. Typically, the function evaluations along the time horizon can be parallelized in

[2] This is notwithstanding the fact that certain problem characteristics might require tuning of the method at hand to cope with numerical difficulties.

[3] In general we cannot assume to have an "exact" model in virtually any real-world application.

a straight-forward manner, while NLP solution methods exhibit only limited parallelization possibilities. The expectation behind using DMS is therefore to profit from parallel computing architectures overproportionally.

We note that DMS and direct collocation may actually be considered very similar from a nonlinear programming point of view if fixed stepsize integrators are chosen in DMS. In contrast to single shooting, where the evaluation of the dynamic system has to be performed in *sequential* order over the time horizon, the term *direct simultaneous methods* was coined for DMS and collocation. We will come back to this similarity in Chapter 3.

1.2.1 Bock's Direct Multiple Shooting method

Multiple Shooting has its origin in the solution of Boundary Value Problems (cf. [MRZ62, Osb69, Bul71, SB08]). Bock and Plitt [BP84] adopted these ideas in a direct method (originally for the solution of optimal control problems), the Direct Multiple Shooting (DMS) method. Some extensions of the original idea and further details can be found in [Boc87, Lei99, LBS+03, Alb10a]. We largely follow similar presentations of the DMS method that can be found in [Lei95, Lei99, Die02, Sag05, Kir11].

Solving (DOP) with a DMS approach we first choose $N+1$ fixed control discretization points (the so-called *multiple shooting grid*)

$$t_0 < t_1 < \ldots < t_N := t_f,$$

which define a partitioning of the time horizon

$$\mathcal{T} = \bigcup_{k=0}^{N-1} \mathcal{T}_k,$$

into N (not necessarily need to be equidistant) *shooting intervals* $\mathcal{T}_k := [t_k, t_{k+1}]$, or, more generally, discrete time *stages*. We denote the induced stage index set by $\mathcal{S} := \{0, 1, \ldots, N\}$, and the induced interval index set by $\mathcal{S}_N := \mathcal{S} \backslash \{N\}$.

Next, the control functions $\boldsymbol{u} \in \mathcal{U}$ are approximated on each interval $k \in \mathcal{S}_N$ in a suitably chosen finite dimensional space $\mathcal{V}_k \cong \mathbb{R}^{n_{q_k}}$ spanned by n_{q_k} basis functions $\boldsymbol{\nu_{k,j}} : \mathcal{T}_k \to \mathbb{R}^{n_u},\ j \in \{1, \ldots, n_{q_k}\}$, such that

$$\boldsymbol{u}(t) \approx \boldsymbol{\nu_k}(t; \boldsymbol{q_k}) := \bigoplus_{j=1}^{n_{q_k}} \left((q_k)_j \odot \boldsymbol{\nu_{k,j}} \right) \qquad \forall\ t \in \mathcal{T}_k.$$

A common choice for $\boldsymbol{\nu_k}$ are vectors of polynomials [LBBS03, Kir11]. For notational simplicity we assume, w.l.o.g., monomials of degree zero as basis

functions throughout this thesis, i.e., $n_{q_k} = n_u,\ \forall\, k \in \mathcal{S}_N$ and

$$\boldsymbol{u}(t) \approx \boldsymbol{q_k} \qquad \forall\ t \in \mathcal{T}_k,\ k \in \mathcal{S}_N,$$

if not explicitly stated otherwise. In the domain of control engineering, this is also known as *zero-order hold*. Note that in particular any control function approximation by polynomials of higher degree can be hidden in the state vector by a state augmentation analogously to Remark 1.6.

The shooting grid induces a natural separation of the system dynamics into N independent *Initial value problems* (IVPs), each given by

$$\boldsymbol{x}(t_k) = \boldsymbol{s_k}\,; \qquad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{q_k}) \qquad \forall\ t \in \mathcal{T}_k \tag{1.4}$$

for $k \in \mathcal{S}_N$. Here, we introduce the additional *shooting node variables* $\boldsymbol{s_k} \in \mathbb{R}^{n_x},\ \forall\, k \in \mathcal{S}$ (the node variable $\boldsymbol{s_N}$ is not an initial condition of an IVP, but only used to check terminal costs and constraints in the NLP formulation) and we assume that time-constant parameters are hidden in the state vector for notational convenience. Keep in mind that by Assumption 1.2 the state trajectory $\boldsymbol{x}(t; \boldsymbol{s_k}, \boldsymbol{q_k})$ is uniquely determined by $\boldsymbol{s_k}$ and $\boldsymbol{q_k}$ on each interval $\mathcal{T}_k,\ k \in \mathcal{S}_N$.

The control discretization together with the state trajectory separation are the two key ingredients of Bock's multiple shooting method. Substituting both parameterizations into (DOP) renders (DOP1) and (DOP2) dependent on only a finite number of optimization variables. For consistency of the transformation we still need to remove the unphysical degrees of freedom $\boldsymbol{s_k},\ k \in \mathcal{S}_N$ by introducing the *continuity constraints* (also known as *matching conditions*)

$$\boldsymbol{s_{k+1}} = \boldsymbol{x}(t_{k+1}; \boldsymbol{s_k}, \boldsymbol{q_k}) \qquad \forall\ k \in \mathcal{S}_N,$$

which ensure continuity of the state trajectory.

To complete the transition from Problem (DOP) to a finite-dimensional NLP we additionally relax the path constraints (DOP3) to only be enforced on a finite grid. For notational simplicity, we choose, without loss of generality, this grid identical to the shooting grid, i.e., we demand

$$\mathbf{0} \leq \boldsymbol{d}(t_k, \boldsymbol{s_k}, \boldsymbol{q_k}) \qquad \forall\ k \in \mathcal{S}$$

instead of (DOP3); we emphasize, however, that in principle the choice of path constraint discretization points is independent from the control discretization grid. We note that, theoretically, this path constraint relaxation does permit arbitrarily large violations of (DOP3) on the strict interior of the intervals $\mathcal{T}_k,\ k \in \mathcal{S}_N$; for a sufficiently fine shooting grid however, this problem is virtually irrelevant in practice. Some interesting concepts on how to inhibit

violations of the infinite dimensional path constraints in a DMS setting can be found in [Pot06, PBS09].

Altogether, the above steps of Bock's multiple shooting procedure lead to the following (finite-dimensional) NLP formulation:

$$\min_{\boldsymbol{s},\boldsymbol{q}} \quad \sum_{k\in\mathcal{S}} \ell(\boldsymbol{s_k}, \boldsymbol{q_k}) \tag{MS-NLP1}$$

$$\text{s.t.} \quad \boldsymbol{s_{k+1}} = \boldsymbol{x_k}(t_{k+1}; \boldsymbol{s_k}, \boldsymbol{q_k}) \qquad \forall\ k \in \mathcal{S}_N \tag{MS-NLP2}$$

$$\mathbf{0} \geq \boldsymbol{d}(t_k, \boldsymbol{s_k}, \boldsymbol{q_k}) \qquad \forall\ k \in \mathcal{S} \tag{MS-NLP3}$$

$$\mathbf{0} = \boldsymbol{r_k}(\boldsymbol{s_k}, \boldsymbol{q_k}) \qquad \forall\ k \in \mathcal{S}. \tag{MS-NLP4}$$

Note that, as for the discretized path constraints, we here assumed that $\mathcal{J} \subseteq \mathcal{S}$, again for notational simplicity. While mathematically the transfer to the more general formulation is straightforward by simply replacing s_k through $\boldsymbol{x_k}(t_i; \boldsymbol{s_k}, \boldsymbol{q_k})$ in Equation (MS-NLP4) for $t_i \in \mathcal{T}_k$, where $i \in \mathcal{J}$, the efficient computational treatment requires special numerical integration routines with support for continuous output (or related features). For details of these methods we kindly refer the reader to [Alb10a, QGD13] for complementary information.

In Formulation (MS-NLP) we introduced the discretized performance indices $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$, whose evaluation, on each stage $k \in \mathcal{S}_N$ (i.e., each except for the last), may depend on the solution of an initial value problem (which, in turn, is fully determined by $\boldsymbol{s_k}, \boldsymbol{q_k}$, though). The discretized stage performance indices ℓ can most efficiently be evaluated alongside the evaluation of the state trajectory IVPs (1.4), cf. [Lei95, Lei99]. Along the time horizon, these evaluations are fully independent and thus can be performed in perfect concurrency on up to N computational threads. Note however that the practically observed speedup largely depends on the chosen numerical integration routine. While the integration routines used classically in the domain of optimal control are so-called *adaptive integrators* (cf., e.g., [Lei95, Lei99, DLS01, PLCS06, HF11, Alb10a, HFD11a, Bei12]), that have a very much data-dependent workload and thus may, in principle, lead to a parallel efficiency as bad as $\frac{1}{N}$, recently so-called *fixed-stepsize integrators*, that have a predetermined workload and thus allow for a theoretical worst-case parallel efficiency of 1 if the mulitple shooting grid is chosen suitably, have gained significant attention in the area of real-time dynamic optimization, cf. [HFD11b, FHGD11b, FKV+12, QVD12, VLH+12, QGD13]. These two classes of methods obviously lead to very different properties of the computed approximations to the exact solutions of IVPs (1.4). In the context of real-time control and estimation, the higher computational efficiency of fixed-stepsize methods may outweigh the (principally

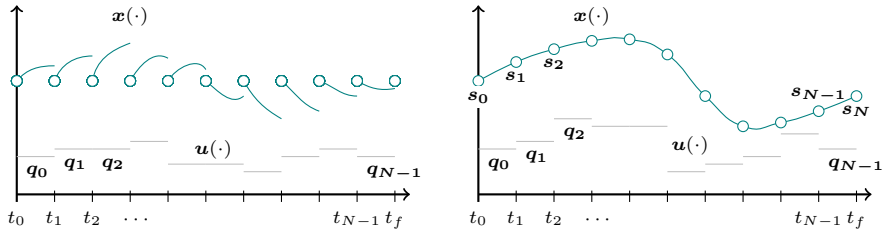


Figure 1.2: Visualization of unconverged (left) and converged (right) state and control variables of a direct multiple shooting method. Inspired by [Kir11].

possible) higher precision of adaptive methods in terms of control/estimation performance for practical purposes. A rigorous comparison between these two classes is beyond the scope of this thesis, though, if possible at all.

The NLP variables of the multiple shooting reformulation are visualized in Figure 1.2. Particularly note the distinctive characteristic of DMS that the state trajectory over the time horizon $\mathcal{T}$ is, in general, *not* continuous for infeasible (e.g., unconverged) NLP variable configurations.

For future reference, we introduce the shorthand notation $\boldsymbol{w} \in \mathbb{R}^n$, where $n = N \cdot (n_x + n_u) + n_x$ in this context, to summarize all optimization variables of (MS-NLP). In doing so, we assume the ordering

$$\boldsymbol{w} = (\boldsymbol{s_0}, \boldsymbol{q_0}, \boldsymbol{s_1}, \cdots, \boldsymbol{q_{N-1}}, \boldsymbol{s_N}) .$$

Concluding this short presentation of DMS let us note that, while the introduction of the additional shooting node variables may at first seem a steep price to pay for better NLP convergence properties, (MS-NLP) has very beneficial sparsity structures. The exploitation of these sparsity structures is absolutely essential for efficient numerical methods, and at the core of this thesis. We give details in Chapters 3 and 4.

1.3 A Primer in Nonlinear Programming Theory

For a general discussion of properties of and solution methods for Problem (MS-NLP), we summarize it in the generic standard form

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} \quad f(\boldsymbol{w}) \tag{1.6a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{w}) = 0 \tag{1.6b}$$

$$\boldsymbol{h}(\boldsymbol{w}) \leq 0, \tag{1.6c}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{eq}}}$, and $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{ieq}}}$. We assume that $\boldsymbol{g}$ and $\boldsymbol{h}$ are indexed by distinct index sets $\mathcal{E} \equiv \{1, \ldots, n_{\text{eq}}\}$ and $\mathcal{I} \equiv \{1, \ldots, n_{\text{ieq}}\}$, respectively.

In consequence of Assumptions 1.3 and 1.18 we also assume twice continuous differentiability of f, $\boldsymbol{g}$, and $\boldsymbol{h}$.

Definition 1.19 (Convex problem) Under the additional assumptions that $f : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{ieq}}}$ are convex, and that $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{eq}}}$ is affine, we call (1.6) a *convex problem.* ⌟

We are interested in characterizing well-known properties of the — not necessarily convex — NLP (1.6) throughout this section. For this purpose we follow [NW06, Kir11, Pot11a] in our presentation. We begin with some elementary definitions:

Definition 1.20 (Feasible point) A vector $\bar{\boldsymbol{w}} \in \mathbb{R}^n$ is called *feasible*, iff it holds

$$\boldsymbol{g}(\bar{\boldsymbol{w}}) = \boldsymbol{0} \qquad \wedge \qquad \boldsymbol{h}(\bar{\boldsymbol{w}}) \leq \boldsymbol{0}. \qquad ⌟$$

We call (1.6) *feasible* if a feasible point exists.

Definition 1.21 (Feasible set) The set consisting of all feasible points is called *feasible set* and is denoted by

$$\mathcal{F} := \{\boldsymbol{w} \mid \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0},\ \boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0}\}. \qquad ⌟$$

Definition 1.22 (Global Optimum) A vector $\boldsymbol{w}^* \in \mathcal{F}$ is called *global solution* (or *global optimum*), iff

$$f(\boldsymbol{w}^*) \leq f(\boldsymbol{w}) \qquad \forall\ \boldsymbol{w} \in \mathcal{F}. \qquad ⌟$$

Definition 1.23 (Local Optimimum) A vector $\boldsymbol{w}^* \in \mathcal{F}$ is called *local solution* (or *local optimum*), iff there exists an open ball of radius ϵ around $\boldsymbol{w}^*$, denoted by $B_\epsilon(\boldsymbol{w}^*) \subset \mathbb{R}^n$, with

$$f(\boldsymbol{w}^*) \leq f(\boldsymbol{w}) \qquad \forall\ \boldsymbol{w} \in \mathcal{F} \cap B_\epsilon(\boldsymbol{w}^*). \qquad \lrcorner$$

Remark 1.24 We note that finding a global solution to (1.6) is, in general, significantly more challenging than finding a local one (cf., e.g., [FAC+05]). In the scope of this thesis, we merely focus on computational methods for obtaining local solutions for two imperative reasons:

1. Real-time global dynamic optimization simply is computationally still intractable, even for (non-trivial real-world) small-scale systems.
2. For non-pathetic (i.e., well-posed, realistic) dynamic systems (and problem formulations), it seems reasonable to assume a one-to-one correspondence of desirable local solutions from one sampling time to the next, as well as a proximity of these solutions to one another in the space of optimization variables, given that the sampling rate for re-optimization is sufficiently high. The initialization techniques discussed in Section 2.3.1 then typically infer convergence of the here-considered local methods to the corresponding local solutions. In practical situations, an initialization in a global optimum is therefore most often sufficient for having one of the considered local methods actually "follow" a global optimum.

Definition 1.25 (Active constraint) A constraint $h_i : \mathbb{R}^n \to \mathbb{R},\ i \in \mathcal{I}$ is called *active* in a feasible point $\bar{\boldsymbol{w}} \in \mathcal{F}$, iff

$$h_i(\bar{\boldsymbol{w}}) = 0.$$

Consequently, all equality constraints $g_i : \mathbb{R}^n \to \mathbb{R},\ i \in \mathcal{E}$ are active in any feasible point $\bar{\boldsymbol{w}} \in \mathcal{F}$. $\lrcorner$

Definition 1.26 (Active Set) The set of all active constraints at a feasible point $\bar{\boldsymbol{w}} \in \mathcal{F}$, the so called *active set*, is denoted by[4]

$$\mathcal{A}(\bar{\boldsymbol{w}}) := \mathcal{E} \cup \{i \mid h_i(\bar{\boldsymbol{w}}) = 0\} \subseteq \mathcal{E} \cup \mathcal{I}. \qquad \lrcorner$$

Definition 1.27 (Linear independence constraint qualification) A feasible vector $\bar{\boldsymbol{w}} \in \mathcal{F}$ is said to fulfill the linear independence constraint qualification (LICQ), if the gradients $\boldsymbol{\nabla g_i}(\bar{\boldsymbol{w}}),\ i \in \mathcal{E}$ and the gradients $\boldsymbol{\nabla h_i}(\bar{\boldsymbol{w}})$ of all active inequality constraints $i \in \mathcal{I} \cap \mathcal{A}(\bar{\boldsymbol{w}})$ are linearly independent. $\lrcorner$

[4] Recall that we assumed $\mathcal{E}$ and $\mathcal{I}$ to be two disjoint index sets.

Remark 1.28 LICQ is a rather strong constraint qualification, but convenient to be used for our purposes. For weaker constraint qualifications and comparisons we refer to the pertinent literature, e.g., [NW06].

Definition 1.29 (Lagrangian function, Lagrange multipliers) For NLP (1.6), the *Lagrangian* function is defined as

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{w}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{w}) + \boldsymbol{\mu}^\top \boldsymbol{h}(\boldsymbol{w}).$$

The vectors $\boldsymbol{\lambda} \in \mathbb{R}^{n_{\text{eq}}}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_{\text{ieq}}}$ are referred to as *Lagrange multipliers* or *dual variables*, and can be interpreted as shadow prices, see [NW06]. ⌟

Remark 1.30 Contrasting the term of dual variables, $\boldsymbol{w}$ are often also referred to as *primal variables*.

Definition 1.31 (Dual problem) The problem

$$\max_{\substack{\boldsymbol{\lambda}\in\mathbb{R}^{n_{\text{eq}}},\\ \boldsymbol{\mu}\in\mathbb{R}^{n_{\text{ieq}}}}} \quad \inf_{\boldsymbol{w}\in\mathbb{R}^n} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{1.7a}$$

$$\text{s.t.} \quad \boldsymbol{\mu} \geq \mathbf{0} \tag{1.7b}$$

is called the *dual problem* of (1.6). Contrasting this, (1.6) is called the *primal problem*. ⌟

Theorem 1.32 (Concavity of the dual function) *The dual function*

$$(\boldsymbol{\lambda}, \boldsymbol{\mu}) \to \inf_{\boldsymbol{w}\in\mathbb{R}^n} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$$

is concave.

Proof See [NW06]. □

Theorem 1.33 (Weak duality) *For any $\bar{\boldsymbol{w}} \in \mathbb{R}^n$ feasible, $\bar{\boldsymbol{\lambda}} \in \mathbb{R}^{n_{\text{eq}}}$, and $\mathbf{0} \leq \bar{\boldsymbol{\mu}} \in \mathbb{R}^{n_{\text{ieq}}}$ we have*

$$\inf_{\boldsymbol{w}\in\mathbb{R}^n} \mathcal{L}(\boldsymbol{w}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}) \leq f(\bar{\boldsymbol{w}}).$$

Proof See [NW06]. □

Definition 1.34 (Strong duality) If

$$\inf_{\boldsymbol{w}\in\mathbb{R}^n} \mathcal{L}(\boldsymbol{w}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}) = f(\bar{\boldsymbol{w}})$$

we say that *strong duality* holds. ⌟

The motivation for formulating dual problems is mainly to obtain an alternative problem formulation that is easier to solve under some circumstances. In fact, the method we present in Chapter 4 is based on this motivation. In general, the dual problem is particularly useful for convex primal problems, where typically strong duality holds, cf. Section 1.3.1.

1.3.1 Finite-dimensional optimality conditions

We have the following necessary first-order characterization of locally optimal points:

Theorem 1.35 (Karush-Kuhn-Tucker (KKT) Conditions) *Let $\boldsymbol{w}^* \in \mathbb{R}^n$ be a local solution to (1.6) that fulfills the LICQ. Then there exist Lagrange multipliers $\boldsymbol{\lambda}^* \in \mathbb{R}^{n_{\text{eq}}}$ and $\boldsymbol{\mu}^* \in \mathbb{R}^{n_{\text{ieq}}}$ such that it holds:*

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{0} \qquad \text{(dual feasibility)}$$

$$\left.\begin{aligned} \boldsymbol{g}(\boldsymbol{w}^*) &= \mathbf{0} \\ \boldsymbol{h}(\boldsymbol{w}^*) &\leq \mathbf{0} \end{aligned}\right\} \qquad \text{(primal feasibility)}$$

$$\left.\begin{aligned} \boldsymbol{\mu}^* &\geq \mathbf{0} \\ \boldsymbol{\mu}^{*\top} \boldsymbol{h}(\boldsymbol{w}^*) &= 0 \end{aligned}\right\} \qquad \text{(complementarity)}$$

Proof See [NW06]. □

Definition 1.36 (KKT point) A triple $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ in the sense of Theorem 1.35 is called *KKT point* or *primal-dual solution* of (1.6). ⌟

Definition 1.37 (Weakly active constraint) Let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a KKT point of (1.6). If for an index $i \in \mathcal{I}$ it holds $\mu_i = 0$ while the corresponding constraints satisfies $h_i(\boldsymbol{w}^*) = 0$, we say that h_i is *weakly active*. If alternatively $\mu_i > 0$, we say that h_i is *binding* or *strictly active*. ⌟

Definition 1.38 (Strict complementarity) Let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a KKT point of (1.6). If all active constraints are binding, i.e., if $\mu_i > 0$ for all $i \in \mathcal{I} \cap \mathcal{A}(\boldsymbol{w}^*)$, we say that *strict complementarity* holds. ⌟

Theorem 1.39 (Uniqueness of Lagrange Multipliers) *Let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a KKT point of (1.6). If strict complementarity and LICQ hold, then the dual variables $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ are unique.*

Proof See [Bie10]. □

Theorem 1.35 only characterizes critical points. Second derivative information allows us to further filter out undesirable candidates for local optimality (saddle points, cf. [NW06]) and eventually can be used to state sufficient requirements to a local solution. We have:

Definition 1.40 (Reduced Hessian) Let $(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}^n \times \mathbb{R}^{n_{\text{eq}}} \times \mathbb{R}^{n_{\text{ieq}}}$, and let n_{act} denote the number of constraints of (1.6) which are active at $\boldsymbol{w}$. Let $\boldsymbol{Z}(\boldsymbol{w}) \in \mathbb{R}^{n \times (n - n_{\text{act}})}$ be a basis matrix for the nullspace of the active constraints of (1.6) linearized at $\boldsymbol{w}$. By *reduced Hessian* (or *projected Hessian*), we refer to the projection of the Hessian of the Lagrangian onto the space of free variables, formally defined by

$$\boldsymbol{H}^{\mathbf{P}}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \boldsymbol{Z}(\boldsymbol{w})^\top \, \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{ww}} \boldsymbol{\mathcal{L}}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \, \boldsymbol{Z}(\boldsymbol{w}). \qquad \lrcorner$$

Remark 1.41 If LICQ holds at $\boldsymbol{w}$, a nullspace basis matrix $\boldsymbol{Z}(\boldsymbol{w})$ as introduced in Definition 1.40 can be constructed as

$$\boldsymbol{Z}(\boldsymbol{w}) := \begin{bmatrix} -\boldsymbol{B}^{-1}\boldsymbol{N} \\ \mathbf{I} \end{bmatrix}$$

by partitioning the Jacobian matrix of active constraint rows in $\boldsymbol{w}$, denoted by $\boldsymbol{D}^{\text{act}} =: \begin{bmatrix} \boldsymbol{B} & \boldsymbol{N} \end{bmatrix} \in \mathbb{R}^{n_{\text{act}} \times n_z}$, into an invertible matrix $\boldsymbol{B} \in \mathbb{R}^{n_{\text{act}} \times n_{\text{act}}}$ (w.l.o.g. the first n_{act} columns by reordering) and a $n_{\text{act}} \times (n_z - n_{\text{act}})$ matrix $\boldsymbol{N}$. We refer to [GM78, NW06] for more details.

Theorem 1.42 (Second-order necessary optimality conditions) *Let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a KKT point of (1.6) that satisfies the LICQ. Then the Hessian of the Lagrangian is positive semidefinite on the nullspace of the strictly active constraints, i.e.,*

$$\boldsymbol{d}^\top \, \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{ww}} \boldsymbol{\mathcal{L}}(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \, \boldsymbol{d} \geq 0 \qquad \forall \; \boldsymbol{d} \in \{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{Z}(\boldsymbol{w}^*)^\top \boldsymbol{d} = \mathbf{0}\}$$

Proof See [NW06]. □

Theorem 1.43 (Second-order sufficient optimality conditions) *Let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a KKT point of (1.6). If the Hessian of the Lagrangian is strictly positive definite on the nullspace of the strictly active constraints, i.e., if*

$$\boldsymbol{d}^\top \, \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{ww}} \boldsymbol{\mathcal{L}}(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \, \boldsymbol{d} > 0 \qquad \forall \; \boldsymbol{d} \in \{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{Z}(\boldsymbol{w}^*)^\top \boldsymbol{d} = \mathbf{0}\} \setminus \{\mathbf{0}\},$$

then $\boldsymbol{w}^*$ is a local minimum of (1.6).

Proof See [NW06]. □

Remark 1.44 From Theorems 1.42 and 1.43 we can conclude that for a convex NLP (1.6) every local minimizer is also a global minimizer. If the reduced Hessian is even strictly positive definite, there exists a unique KKT point, which is the unique global minimizer. Formal proofs can be found, e.g., in [BV04, NW06]).

In the convex case, we have the following useful relationships between the primal and the dual problem.

Theorem 1.45 *Let (1.6) be convex, and let $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be a (not necessarily unique) KKT point of (1.6). Then $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solves the dual problem (1.7).*

Proof See [NW06]. □

Theorem 1.46 *Let (1.6) be convex, and let $\boldsymbol{w}^*$ be a solution of (1.6) at which LICQ holds. Let further be $(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$ a solution of the dual problem (1.7), and let $\hat{\boldsymbol{w}}$ such that $\inf_{\boldsymbol{w} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{w}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$ is attained at $\hat{\boldsymbol{w}}$.*

If the reduced Hessian $\boldsymbol{H}^{\mathrm{P}}(\boldsymbol{w}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$ is strictly positive definite in $(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$ for all $\boldsymbol{w}$ (e.g., if f is strictly convex), then $\hat{\boldsymbol{w}} = \boldsymbol{w}^*$ and strong duality holds and equality of primal and dual objective function value is attained at $(\boldsymbol{w}^*, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$.

Proof See [NW06]. □

1.4 Selected Methods for Nonlinear Programming

Nonlinear Optimization is a broad field that has sparked many researchers, who have subsequently developed various concepts for solving (1.6) in one variant or another. Again, we only briefly cover the algorithmic classes which are relevant for this thesis; for a broader picture we kindly refer the interested reader to [NW06] and the references therein.

1.4.1 Interior-point methods

Interior-point or *Barrier methods* are among the most popular classes of NLP algorithms. The general idea is to treat the combinatorial complexity introduced by the inequality constraints (1.6c) through smoothening and to obtain iterates that remain in the strict interior of the feasible region. We only prototypically state the concept of a *primal-dual interior-point* method here, therein following [NW06]; a more comprehensive overview can, for example, be found in [Wri97].

Primal-dual methods operate by applying Newton's method to a parametric system of nonlinear equations, which originates in the KKT conditions of (1.6) stated in Theorem 1.35. Introducing slack variables $\boldsymbol{s} \in \mathbb{R}^{n_{\mathrm{ieq}}}$, the KKT

conditions are restated as

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0} \tag{1.8a}$$

$$\boldsymbol{g}(\boldsymbol{w}) = \mathbf{0} \tag{1.8b}$$

$$\boldsymbol{h}(\boldsymbol{w}) = -\boldsymbol{s} \tag{1.8c}$$

$$\boldsymbol{\mu}^\top \boldsymbol{s} = 0 \tag{1.8d}$$

$$\boldsymbol{\mu} \geq \mathbf{0} \tag{1.8e}$$

$$\boldsymbol{s} \geq \mathbf{0} \tag{1.8f}$$

and, introducing the barrier parameter $\tau > 0$, subsequently weakened[5] to the following system of equations:

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0} \tag{1.9a}$$

$$\boldsymbol{g}(\boldsymbol{w}) = \mathbf{0} \tag{1.9b}$$

$$\boldsymbol{h}(\boldsymbol{w}) + \boldsymbol{s} = \mathbf{0} \tag{1.9c}$$

$$\boldsymbol{\mu}^\top \boldsymbol{s} = \tau. \tag{1.9d}$$

Primal-dual methods proceed by repeatedly solving approximations to system (1.9) for a sequence of barrier parameters $\tau \to 0$. More precisely, stating from an initial guess $(\boldsymbol{w^0}, \boldsymbol{\lambda^0}, \boldsymbol{\mu^0}, \boldsymbol{s^0})$ that satisfies $(\boldsymbol{\mu^0}, \boldsymbol{s^0}) > \mathbf{0}$ (component-wise), we iterate

$$(\boldsymbol{w^{k+1}}, \boldsymbol{\lambda^{k+1}}, \boldsymbol{\mu^{k+1}}, \boldsymbol{s^{k+1}}) := (\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k}, \boldsymbol{s^k}) + \alpha^k (\boldsymbol{\Delta w^k}, \boldsymbol{\Delta \lambda^k}, \boldsymbol{\Delta \mu^k}, \boldsymbol{\Delta s^k}),$$

where $(\boldsymbol{\Delta w^k}, \boldsymbol{\Delta \lambda^k}, \boldsymbol{\Delta \mu^k}, \boldsymbol{\Delta s^k})$ are solutions to the linear system of equations

$$\begin{bmatrix} \boldsymbol{\mathcal{H}^k} & \nabla \boldsymbol{g}(\boldsymbol{w^k}) & \nabla \boldsymbol{h}(\boldsymbol{w^k}) & \mathbf{0} \\ \nabla \boldsymbol{g}(\boldsymbol{w^k}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla \boldsymbol{h}(\boldsymbol{w^k}) & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{S^k} & \boldsymbol{M^k} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta w^k} \\ \boldsymbol{\Delta \lambda^k} \\ \boldsymbol{\Delta \mu^k} \\ \boldsymbol{\Delta s^k} \end{bmatrix} = \begin{bmatrix} -\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k}) \\ -\boldsymbol{g}(\boldsymbol{w^k}) \\ -\boldsymbol{h}(\boldsymbol{w^k}) - \boldsymbol{s^k} \\ -\boldsymbol{S^k M^k e} + \tau^k \boldsymbol{e} \end{bmatrix}.$$

Here, $\boldsymbol{\mathcal{H}^k} \approx \nabla^{\mathbf{2}}_{\boldsymbol{ww}} \mathcal{L}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k})$ is a suitable approximation to the Hessian of the Lagrangian that is positive definite on the nullspace of the constraint matrix

$$\begin{bmatrix} \nabla \boldsymbol{g}(\boldsymbol{w^k}) & \mathbf{0} \\ \nabla \boldsymbol{h}(\boldsymbol{w^k}) & \mathbf{I} \end{bmatrix},$$

[5] Precisely speaking, (1.9) only is a true relaxation of (1.8) for $\tau = 0$.

while $\boldsymbol{S^k} = \mathbf{diag}(s_1^k, \ldots, s_{n_{\text{ieq}}}^k)$, $\boldsymbol{M^k} = \mathbf{diag}(\mu_1^k, \ldots, \mu_{n_{\text{ieq}}}^k)$, and $\boldsymbol{e} = (1, \ldots, 1) \in \mathbb{R}^{n_{\text{ieq}}}$.

Suitable initialization and choice of step sizes $\alpha^k \in (0, 1]$ ensures that the thus generated homotopy of iterates $\{(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k}, \boldsymbol{s^k})\}_{k=1,2,\ldots}$ always satisfies $(\boldsymbol{\mu^k}, \boldsymbol{s^k}) > \mathbf{0}$, i.e., always remains inside the feasible region and eventually converges to a KKT point. We refer to [NW06] for an overview of specific criteria.

1.4.2 Augmented Lagrangian methods

Augmented Lagrangian methods in their original form are not among the most popular methods for nonlinear dynamic optimization problems of the here considered kind. Some aspects of and ideas from augmented Lagrangian methods have however made it into algorithms that are relevant for dynamic optimization; in particular, the dual-decomposition based NLP algorithm presented in Section 5.2 exhibits flavors of an augmented Lagrangian method, which is why we include a brief general introduction to this class of methods here, loosely following [NW06]. We also refer to [NW06] and the references therein for a more complete picture and further details.

For an introduction of the central idea of augmented Lagrangian methods, it is helpful to consider the rewritten NLP formulation

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \quad f(\boldsymbol{w}) \tag{1.10a}$$

$$\text{s.t.} \quad g_j(\boldsymbol{w}) = \mathbf{0} \qquad \forall\, j \in \mathcal{E} \tag{1.10b}$$

$$\boldsymbol{w} \leq \mathbf{0}. \tag{1.10c}$$

NLP (1.6) can be rewritten in the form of (1.10) by introducing slack variables, similar to the reformulation in Section 1.4.1. The augmented Lagrangian function $\mathcal{L}_{\text{A}}(\boldsymbol{w}, \boldsymbol{\lambda}; \rho)$ for a positive penalty weight $\rho > 0$ is a combination of the objective function, the linear Lagrangian penalty on the equality constraints, as well as a quadratic penalty term of these constraints. We have

$$\mathcal{L}_{\text{A}}(\boldsymbol{w}, \boldsymbol{\lambda}; \rho) := f(\boldsymbol{w}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{w}) + \frac{\rho}{2} \sum_{j \in \mathcal{E}} g_j^2(\boldsymbol{w}).$$

A basic augmented Lagrangian method solves (1.10) by iteratively minimizing the augmented Lagrangian function (approximately) in $\boldsymbol{w}$, subject to the bound constraints (1.10c), for repeatedly updated guesses of multipliers $\boldsymbol{\lambda^i}$ and penalty

weights ρ^i. The i-th subproblem therefore reads

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} \quad \mathcal{L}_{\mathrm{A}}(\boldsymbol{w}, \boldsymbol{\lambda^i}; \rho^i) \tag{1.11a}$$

$$\text{s.t.} \quad \boldsymbol{w} \leq \mathbf{0}. \tag{1.11b}$$

Due to the box-structure of the constraints, approximate solutions to (1.11) can be computed rather efficiently, for example by gradient projection methods (cf. [NW06]).

Denoting the approximate primal-dual minimizer of (1.11) by $\left(\boldsymbol{w^i}, \boldsymbol{\mu^i}\right)$, we can see from the KKT optimality conditions of (1.11), that

$$\begin{aligned}\mathbf{0} &\approx \boldsymbol{\nabla_w} \boldsymbol{\mathcal{L}_{\mathbf{A}}}(\boldsymbol{w^i}, \boldsymbol{\lambda^i}; \rho^i) + \boldsymbol{\mu^i} \\ &= \boldsymbol{\nabla f}(\boldsymbol{w^i}) + \sum_{j\in\mathcal{E}} \left(\lambda_j^i + \rho^i\, g_j(\boldsymbol{w^i})\right) \boldsymbol{\nabla g_j}(\boldsymbol{w^i}) + \boldsymbol{\mu^i}.\end{aligned}$$

Comparing this to the KKT conditions for (1.10) gives rise to the multiplier update rule

$$\boldsymbol{\lambda^{i+1}} := \boldsymbol{\lambda^i} + \rho^i\, \boldsymbol{g}(\boldsymbol{w^i}),$$

so as to have $\boldsymbol{\lambda^{i+1}} \approx \boldsymbol{\lambda^*}$, where $\boldsymbol{\lambda^*}$ are the optimal dual variables associated with the equality constraints of (1.10).

It can be established (under the typical regularity assumptions for nonlinear programming) that a minimizer $\boldsymbol{w^*}$ of (1.10) also solves (1.11) exactly if the exact multipliers $\boldsymbol{\lambda^*}$ are used and ρ is large enough; furthermore, it can be shown that NLP solution and augmented Lagrangian subproblem solution are close to one another already when either ρ is large enough or the used multiplier vector $\boldsymbol{\lambda}$ is a sufficiently accurate guess. In contrast to a pure quadratic penalty formulation, for example, augmented Lagrangian methods ensure stricter satisfaction of constraints, and avoid ill-conditioning of the subproblems at more or less the same computational expesenses, cf. [NW06].

1.4.3 Sequential quadratic programming

Somewhat similar to augmented Lagrangian methods, and in contrast to interior-point methods, so called *sequential quadratic programming* (SQP) methods (sometimes also referred to as *successive quadratic programming strategies*) pass on the combinatorial complexity of determining the correct active set to a simpler subproblem rather than smoothening it out. This simpler problem typically is a convex *quadratic programming problem* (QP), but there are also related methods that employ linear programming subproblems (*sequential linear*

programming, SLP), or more general convex subproblems (*sequential convex programming*, SCP). Still, QP subproblems have some very favorable properties and characteristics, and SQP remains one of the most successful methods for nonlinear programming to date. With respect to the repeated solution of similar NLPs in the context of real-time optimization, some of these favorable features have an even stronger effect. Due to this relevance, we focus on SQP in this introduction. Complementary information and further references can be found in [NW06], as well as in a recent PhD thesis on SCP, [TD12]. Our presentation of SQP follows [NW06] and [Kir11].

Newton- and Newton-type methods rank among the highest performing algorithms for unconstrained nonlinear optimization. SQP is motivated by the observation that for a fixed active set, Newton iterations on the KKT optimality conditions can be interpreted as repeatedly solving an equality-constrained QP, whose data corresponds to a local quadratic approximation of the NLP Lagrangian in the objective subject to a linearization of the active constraints in the current primal-dual NLP iterate.

Passing on the active-/inactiveness decision in the constraints, SQP repeatedly solves the following convex QP

$$\min_{\boldsymbol{\Delta w^k}\in\mathbb{R}^n} \quad \boldsymbol{\Delta w^k}^\top \boldsymbol{\mathcal{H}^k} \boldsymbol{\Delta w^k} + \boldsymbol{\nabla f}(\boldsymbol{w^k})^\top \boldsymbol{\Delta w^k} \tag{1.12a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{w^k}) + \boldsymbol{\nabla g}(\boldsymbol{w^k})\boldsymbol{\Delta w^k} = \mathbf{0} \quad | \ \lambda^k_{\mathrm{QP}} \tag{1.12b}$$

$$\boldsymbol{h}(\boldsymbol{w^k}) + \boldsymbol{\nabla h}(\boldsymbol{w^k})\boldsymbol{\Delta w^k} \leq \mathbf{0} \quad | \ \mu^k_{\mathrm{QP}} \ , \tag{1.12c}$$

updating the NLP primal-dual iterates by

$$\boldsymbol{w^{k+1}} = \boldsymbol{w^k} + \alpha^k \boldsymbol{\Delta w^k} \tag{1.13a}$$

$$\boldsymbol{\lambda^{k+1}} = \boldsymbol{\lambda^k} + \alpha^k (\boldsymbol{\lambda^k_{QP}} - \boldsymbol{\lambda^k}) \tag{1.13b}$$

$$\boldsymbol{\mu^{k+1}} = \boldsymbol{\mu^k} + \alpha^k (\boldsymbol{\mu^k_{QP}} - \boldsymbol{\mu^k}) \, . \tag{1.13c}$$

Here, $\boldsymbol{\mathcal{H}^k} \approx \boldsymbol{\nabla^2_{ww}\mathcal{L}}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k})$ is again a suitable approximation to the Hessian of the Lagrangian which is positive definite on the nullspace of the equality constraints $\boldsymbol{\nabla g_i}(\boldsymbol{w^k})$, $i \in \mathcal{E}$ and the active inequality constraints $\boldsymbol{\nabla h_i}(\boldsymbol{w^k})$, $i \in \mathcal{I} \cap \mathcal{A}(\boldsymbol{w^k})$. Regarding the QP Lagrangian it is easy to see that by using $\boldsymbol{\nabla f}(\boldsymbol{w^k})$ in lieu of $\boldsymbol{\nabla_w \mathcal{L}}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k})$ as the linear QP objective term, the QP multipliers $\boldsymbol{\lambda^k_{QP}}$ and $\boldsymbol{\mu^k_{QP}}$ approximate their NLP counterparts $\boldsymbol{\lambda^{k+1}}$ and $\boldsymbol{\mu^{k+1}}$, instead of just their increments, which, among other advantages, avoids the necessity of a dual initial guess[6], cf. [NW06]. It is easy to verify that

[6] In the first iteration, we can just use $\boldsymbol{\lambda^0} = \boldsymbol{\lambda^1_{QP}}$ and $\boldsymbol{\mu^0} = \boldsymbol{\mu^1_{QP}}$.

the Lagrangian of QP (1.12) is identical with the Lagrangian of NLP (1.6) up to a second-order expansion for an exact Hessian $\boldsymbol{\mathcal{H}^k} = \boldsymbol{\nabla^2_{ww}\mathcal{L}}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k})$, which is why regular minima of NLP (1.6) qualify as QP solutions for (1.12).

The step size $\alpha^k \in (0, 1]$ damps the Newton-type step and, if chosen appropriately, ensures *globalized convergence* to a KKT point under certain conditions, regardless of the initial guess. In general, *line search*, *trust-region*, or *filter methods* are commonly used to determine a desirable α^k. Depending on the method, the precise formulation of the subproblems (1.12) and the update rules (1.13) may vary slightly (they are formulated from a line search viewpoint here), but the big picture can be cast into the above form. Global convergence proofs can be established for a large variety of in-detail step size rules under some additional, typically rather mild assumptions. For an overview [Fle87] and [NW06] may serve as a starting point.

1.4.4 Hessian approximations and convergence results

More than global convergence, local convergence results are of particular interest in the context of real-time dynamic optimization, since one can, in general, assume a good initial guess in the respective algorithm's iterates which is close to a desirable KKT point. A good indicator for the performance of the above optimization algorithms, which belong the the class of *iterative methods*, is given by the *rate of convergence*, which characterizes the progress made by an algorithm towards a solution in each iteration. We have the following important categories, where $\|\cdot\|$ denotes an appropriate norm. Classifications of convergence rates using weaker criteria can be found, for example, in [NW06, OR70].

Definition 1.47 (q-sublinear convergence rate) Let $\{\boldsymbol{w^k}\} \subset \mathbb{R}^n$ with $\boldsymbol{w^k} \to \boldsymbol{w^*}$ for $k \to \infty$. We say $\{\boldsymbol{w^k}\}$ converges *(q-)sublinearly* if

$$\frac{\|\boldsymbol{w_{k+1}} - \boldsymbol{w^*}\|}{\|\boldsymbol{w_k} - \boldsymbol{w^*}\|} \leq 1 \qquad \forall\ k \geq \bar{k} \in \mathbb{N}$$

⌟

Definition 1.48 (q-linear convergence rate) Let $\{\boldsymbol{w^k}\} \subset \mathbb{R}^n$ with $\boldsymbol{w^k} \to \boldsymbol{w^*}$ for $k \to \infty$. We say $\{\boldsymbol{w^k}\}$ converges *(q-)linearly* if there is an $r \in (0, 1)$ such that

$$\frac{\|\boldsymbol{w_{k+1}} - \boldsymbol{w^*}\|}{\|\boldsymbol{w_k} - \boldsymbol{w^*}\|} \leq r \qquad \forall\ k \geq \bar{k} \in \mathbb{N}$$

⌟

Definition 1.49 (q-superlinear convergence rate) Let $\{\boldsymbol{w^k}\} \subset \mathbb{R}^n$ with $\boldsymbol{w^k} \to \boldsymbol{w^*}$ for $k \to \infty$. We say $\{\boldsymbol{w^k}\}$ converges *(q-)superlinearly* if

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

⌟

Definition 1.50 (q-quadratic convergence rate) Let $\{\boldsymbol{w^k}\} \subset \mathbb{R}^n$ with $\boldsymbol{w^k} \to \boldsymbol{w^*}$ for $k \to \infty$. We say $\{\boldsymbol{w^k}\}$ converges *(q-)quadratically* if there is a $K \in (0, \infty)$ such that

$$\frac{\|\boldsymbol{w_{k+1}} - \boldsymbol{w^*}\|}{\|\boldsymbol{w_k} - \boldsymbol{w^*}\|^2} \leq K \qquad \forall\, k \geq \bar{k} \in \mathbb{N} \qquad \lrcorner$$

The convergence rate of the presented algorithms depends strongly on the chosen approximation $\boldsymbol{\mathcal{H}^k} \approx \boldsymbol{\nabla^2_{ww}\mathcal{L}}(\boldsymbol{w^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k})$ to the Hessian of the Lagrangian. In general, positive definiteness of $\boldsymbol{\mathcal{H}^k}$ on the nullspace of the active (in-) equality constraints is essentially sufficient to guarantee convergence inside a small neighborhood of a KKT point. A variety of choices for $\boldsymbol{\mathcal{H}^k}$ therefore exists, including the identity matrix, approximations from the Broyden class that includes the famous BFGS update Formula [Bro70, Fle70, Gol70, Sha70], and the exact Hessian of the Lagrangian. Useful references to obtain an overview can be found in classical textbooks on nonlinear optimization including [Fle87, NW06, Bie10].

We present a rather abstract local convergence result in the following that can be instantiated to a variety of Newton-type methods. In analyzing local convergence rates of SQP methods it is common practice to assume that the correct active set at a desirable solution $\boldsymbol{w^*}$ is already identified [Rob74, NW06]. In a sufficiently small neighborhood of a local solution $\boldsymbol{w^*}$, the following theorem for root-finding problems is then applicable in a straight-forward manner to optimization algorithms that follow the generic update rule

$$\boldsymbol{w^{k+1}} = \boldsymbol{w^k} + \boldsymbol{\Delta w^k} = \boldsymbol{w^k} - \boldsymbol{M}(\boldsymbol{w^k})\, \boldsymbol{q}(\boldsymbol{w^k}). \tag{1.14}$$

Here, $\boldsymbol{w^k} \in \mathbb{R}^m$ denote the primal-dual iterates of the respective algorithm, while $\boldsymbol{M} : \mathcal{D} \to \mathbb{R}^{m \times m}$ and $\boldsymbol{q} : \mathcal{D} \to \mathbb{R}^m$ for $\mathcal{D} \subseteq \mathbb{R}^m$ denote the inverse Newton-like matrix and residual vector[7]. Details can be found in [Boc87, Lei99, Die02].

Theorem 1.51 (Local contraction theorem) *Let the sequence* $\{\boldsymbol{w^k}\}_{k=1,2,\ldots} \subset \mathbb{R}^m$ *be generated by* (1.14) *and let* $\boldsymbol{M}$ *be such that*

- *a finite "nonlinearity bound"* $\omega < \infty$ *exists, satisfying*

$$\left\|\boldsymbol{M}(\boldsymbol{w^{k+1}}) \left(\boldsymbol{\nabla q}(\boldsymbol{w^k} + t(\boldsymbol{w^{k+1}} - \boldsymbol{w^k})) - \boldsymbol{\nabla q}(\boldsymbol{w^k})\right)(\boldsymbol{w^k} - \boldsymbol{w^{k+1}})\right\| \leq \omega\, t \left\|\boldsymbol{w^{k+1}} - \boldsymbol{w^k}\right\|^2$$

for all $t \in [0, 1]$ *and* $(\boldsymbol{w^k}, \boldsymbol{w^{k+1}}) \in \mathcal{D} \times \mathcal{D}$, *and*

[7] For illustration, just think of Newton's method for unconstrained minimization, where $\boldsymbol{M}(\boldsymbol{w^k}) = \alpha^k \left(\boldsymbol{\nabla^2 f}(\boldsymbol{w^k})\right)^{-1}$ and $\boldsymbol{q}(\boldsymbol{w^k}) = \boldsymbol{\nabla f}(\boldsymbol{w^k})$.

- *an "incompatibility constant"* $\kappa < 1$ *exists with*

$$\left\| \boldsymbol{M}(\boldsymbol{w}^{\boldsymbol{k+1}}) \left(\mathbf{I} - \boldsymbol{\nabla q}(\boldsymbol{w}^{\boldsymbol{k}}) \boldsymbol{M}(\boldsymbol{w}^{\boldsymbol{k}}) \right) \boldsymbol{q}(\boldsymbol{w}^{\boldsymbol{k}}) \right\| \leq \kappa \left\| \boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}} \right\|.$$

Then, if $\boldsymbol{w}^{\mathbf{0}} \in \mathcal{D}$ *such that the closed ball* $\overline{\mathcal{B}}_\epsilon(\boldsymbol{w}^{\mathbf{0}})$ *is contained in* $\mathcal{D}$ *for* $\epsilon := \|\boldsymbol{w}^{\mathbf{1}} - \boldsymbol{w}^{\mathbf{0}}\|/(1-c^0)$*, where we define* $c^k := (\kappa + \frac{\omega}{2}\|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}}\|)$*, we have that all* $\boldsymbol{w}^{\boldsymbol{k}} \in \overline{\mathcal{B}}_\epsilon(\boldsymbol{w}^{\mathbf{0}})$ *and* $\boldsymbol{w}^{\boldsymbol{k}} \to \boldsymbol{w}^*$ *for a* $\boldsymbol{w}^* \in \overline{\mathcal{B}}_\epsilon(\boldsymbol{w}^{\mathbf{0}})$ *with*

$$\|\boldsymbol{w}^{\boldsymbol{k+2}} - \boldsymbol{w}^{\boldsymbol{k+1}}\| \leq c^k \cdot \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}}\|$$

and

$$\|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| \leq \frac{c^k}{1-c^k} \cdot \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}}\|. \tag{1.15}$$

We further have

$$\boldsymbol{M}(\boldsymbol{w}^*)\, \boldsymbol{q}(\boldsymbol{w}^*) = \mathbf{0}$$

in the solution, and if $\boldsymbol{M}$ *is additionally continuous and nonsingular, we have*

$$\boldsymbol{q}(\boldsymbol{w}^*) = \mathbf{0}.$$

Proof See, e.g., [Boc87] or [Pot11a]. □

For our purposes, three categories of approximations $\boldsymbol{\mathcal{H}}^{\boldsymbol{k}}$ are of particular interest. The first class are exact Hessian SQP methods, where $\boldsymbol{\mathcal{H}}^{\boldsymbol{k}} = \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{ww}} \boldsymbol{\mathcal{L}}(\boldsymbol{w}^{\boldsymbol{k}}, \boldsymbol{\lambda}^{\boldsymbol{k}}, \boldsymbol{\mu}^{\boldsymbol{k}})$. If $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a KKT point that satisfies the LICQ and the second-order sufficient optimality conditions of Theorem 1.43, then in terms of Theorem 1.51 we have $\boldsymbol{M}(\boldsymbol{w}^{\boldsymbol{k}}) = \left(\boldsymbol{\nabla q}(\boldsymbol{w}^{\boldsymbol{k}})\right)^{-1}$ in a sufficiently small neighborhood of the KKT point, and we can choose $\kappa = 0$. Then (1.15) implies

$$\|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| \leq \frac{c^k}{1-c^k} \cdot \left(\|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| + \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\| \right)$$

$$\Leftrightarrow \qquad (1 - 2\,c^k) \cdot \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| \leq c^k \cdot \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\|. \tag{1.16}$$

Plugging in the definition of c^k and using the triangle inequality once more we obtain

$$\left(1 - \omega \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}}\|\right) \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\|$$

$$\leq \frac{\omega}{2} \left(\|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| + \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\| \right) \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\|$$

$$\Leftrightarrow \quad \left(1 - \omega \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^{\boldsymbol{k}}\| - \frac{\omega}{2} \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\|\right) \|\boldsymbol{w}^{\boldsymbol{k+1}} - \boldsymbol{w}^*\| \leq \frac{\omega}{2} \|\boldsymbol{w}^{\boldsymbol{k}} - \boldsymbol{w}^*\|^2.$$

By Theorem 1.51 it holds $\left(1-\omega\left\|\boldsymbol{w}^{\boldsymbol{k+1}}-\boldsymbol{w}^{\boldsymbol{k}}\right\|-\frac{\omega}{2}\left\|\boldsymbol{w}^{\boldsymbol{k}}-\boldsymbol{w}^{*}\right\|\right)>0$ for a sufficiently large k, which means eventual quadratic convergence. An extended proof under slightly more general assumptions can be found in [Fle87].

The second category is formed by the so-called *Gauss-Newton* Hessian approximation, which is a particularly popular choice in the contexts of estimation and real-time dynamic optimization. The resulting SQP method is sometimes also referred to as *Generalized Gauss-Newton Method* [Boc83]. Here, we assume that the NLP objective (1.6a) has the partially separable quadratic structure

$$f(\boldsymbol{w}^{\boldsymbol{k}})=\frac{1}{2}\|\boldsymbol{r}(\boldsymbol{w}^{\boldsymbol{k}})\|_2^2\,,$$

where $\boldsymbol{r}:\mathbb{R}^n\to\mathbb{R}^{n_r}$. The Hessian of the Lagrangian is then given by

$$\begin{aligned}\boldsymbol{\nabla}^{\boldsymbol{2}}_{\boldsymbol{ww}}\boldsymbol{\mathcal{L}}(\boldsymbol{w}^{\boldsymbol{k}},\boldsymbol{\lambda}^{\boldsymbol{k}},\boldsymbol{\mu}^{\boldsymbol{k}})=\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{\boldsymbol{k}})\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{\boldsymbol{k}})^{\top}+\sum_{i=1}^{n_r}r_i(\boldsymbol{w}^{\boldsymbol{k}})\boldsymbol{\nabla}^{\boldsymbol{2}}\boldsymbol{r}_{\boldsymbol{i}}(\boldsymbol{w}^{\boldsymbol{k}})\\+\sum_{i=1}^{n_{\mathrm{eq}}}\lambda_i^k\boldsymbol{\nabla}^{\boldsymbol{2}}\boldsymbol{g}_{\boldsymbol{i}}(\boldsymbol{w}^{\boldsymbol{k}})+\sum_{i=1}^{n_{\mathrm{ieq}}}\mu_i^k\boldsymbol{\nabla}^{\boldsymbol{2}}\boldsymbol{h}_{\boldsymbol{i}}(\boldsymbol{w}^{\boldsymbol{k}}).\end{aligned}$$

Assuming that $\boldsymbol{r}(\cdot)$ will be small[8] close to a desirable solution $\boldsymbol{w}^{*}$ the Gauss-Newton Hessian approximation takes

$$\boldsymbol{\mathcal{H}}^{\boldsymbol{k}}=\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{\boldsymbol{k}})\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{\boldsymbol{k}})^{\top}.$$

One can indeed show that

$$\left\|\boldsymbol{\nabla}^{\boldsymbol{2}}_{\boldsymbol{ww}}\boldsymbol{\mathcal{L}}(\boldsymbol{w}^{*},\boldsymbol{\lambda}^{*},\boldsymbol{\mu}^{*})-\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{*})\boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^{*})^{\top}\right\|=\mathcal{O}(\|\boldsymbol{r}(\boldsymbol{w}^{*})\|)\,,$$

cf. [Die02]. If $\boldsymbol{r}(\boldsymbol{w}^{*})=\boldsymbol{0}$ we therefore can conclude a superlinear convergence rate from (1.16) using Theorem 1.51, since we can choose $\kappa\to 0$ with increasing k. In the general case of $\boldsymbol{r}(\boldsymbol{w}^{*})\neq\boldsymbol{0}$ (but still small) one obtains that the Gauss-Newton method converges only at a linear rate, cf. [Boc87]; however, the practically observed rate is typically still fast. In the context of estimation, it can further be shown that in the presence of multiple local minima a (full-step) SQP method with a Gauss-Newton Hessian approximation is only attracted by those minima which are stable under small data perturbations and therefore more desirable and in particular meaningful from a statistical point of view, cf. [BKS07].

Third, we briefly cover gradient methods with $\boldsymbol{\mathcal{H}}^{\boldsymbol{k}}=\gamma^k\cdot\mathbf{I}$ for $\gamma^k\in\mathbb{R}$. In this case, the Hessian compatibility is generally poor and typically prohibits

[8] Using the dual feasibility conditions one can show that also $\boldsymbol{\lambda}^{*}$ and $\boldsymbol{\mu}^{*}$ are small in this case, cf. [Die02].

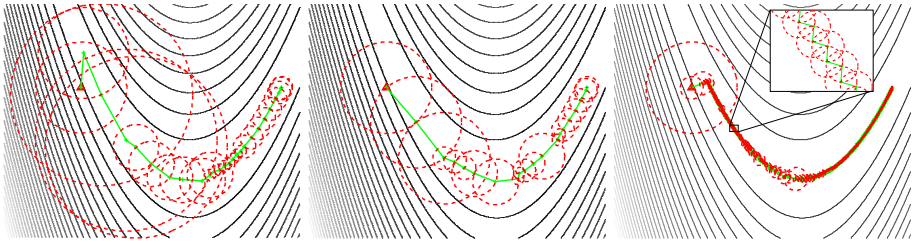


Figure 1.3: Exact Hessian, Gauss-Newton-Hessian, and identity Hessian SQP Method (from left to right) with a Dogleg trust-region globalization strategy on the Rosenbrock problem. Green solid lines visualize the Newton path and red dashed circles indicate the trust radius at each iterate. The minimum is found on the right side of each figure.

vanishing sequences of $\kappa \to 0$ in Theorem 1.51, which diminishes our hope for fast local convergence. Indeed, it is well known that the gradient method converges at a linear rate r, which cannot be bounded away from 1 with increasing condition number of $\boldsymbol{\nabla}^2_{\boldsymbol{ww}}\boldsymbol{\mathcal{L}}(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, where $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a KKT point. In particular, if $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ does not satisfy the second-order sufficient optimality conditions of Theorem 1.43, the asymptotic convergence rate is in general not better than sublinear. We refer to [Nes04] and [NW06] for details.

While these general results seem dissatisfying, gradient methods are still of interest for the solution of convex problems in the area of model predictive control (see also Section 1.5). The simple structure of the Hessian approximation allows for methods that only require matrix-vector products (but no matrix inversions) and therefore are particularly easy to implement on embedded hardware; furthermore, rather tight hard runtime bounds can be established, cf. [Ric12, PB14].

We include an illustrating example in the following. Figure 1.3 compares the Newton path of three different SQP methods on the famous Rosenbrock problem[9]. All three use the same *Dogleg trust region method* for globalization (see [NW06] for details). Circles visualize the trust radius at each iteration, which can be seen as an indicator for the compatibility between the respective Hessian approximation and the actual Hessian of the Lagrangian at each iterate. In particular, we can see from Figure 1.3 that the trust radius stays constant towards the end of the Newton path while full Newton-type steps are taken, both for the exact Hessian SQP methods and the Gauss-Newton method. The

[9]The problem description can be found, for example, in [NW06].

fact, that the trust radius for the Gauss-Newton method is comparable to the trust radius of the exact Hessian SQP method indicates the high quality of the Gauss-Newton approximation for problems with small objective residuals in the solution. In case of the gradient method, characteristic zigzagging behavior is observed. In the considered example, the exact Hessian SQP method converged within 28 iterations to a local solution with a moderate stationarity tolerance of 10^{-4}, while the Gauss-Newton method took 21 iterations, and the gradient method took 10286 iterations.

1.4.5 Derivative generation

The algorithms presented throughout this section have in common that they all require at least first-order derivatives of the NLP functions f, $\boldsymbol{g}$, and $\boldsymbol{h}$. In the context of dynamic optimization problems, these functions depend on the solution of IVPs. A closed-form representation of first- and second-order derivatives is then typically impossible or at least computationally intractable due to exploding size of the expressions involved when automatically generated, e.g., by computer algebra systems.

Therefore, we typically rely on *finite differences* or *algorithmic differentiation* (AD), which is sometimes also referred to as *automatic differentiation*, for the computation of derivative information. The so-called *complex-step derivative approximation* essentially falls in the category of AD [NAW98, MSA03].

Using finite differences is arguably the most straightforward way to obtain derivative information. Therein, a directional derivative $\dot{\boldsymbol{\nu}}_{\boldsymbol{i}} \in \mathbb{R}^m$,

$$\dot{\boldsymbol{\nu}}_{\boldsymbol{i}} = \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{w}}(\boldsymbol{w}) \cdot \dot{\boldsymbol{w}}_{\boldsymbol{i}}\,, \tag{1.17}$$

of a function $\boldsymbol{\Phi} : \mathbb{R}^n \to \mathbb{R}^m$ evaluated at $\boldsymbol{w}$ in direction $\dot{\boldsymbol{w}}_{\boldsymbol{i}}$, where $i \in \{1, \ldots, p\} \subset \mathbb{N}$, is approximated by the forward, backward, or central difference quotient of the function perturbed in the desired direction. It is a well-known fact that due to truncation errors on the one side and cancellation errors on the other side, in general, a precision of not more than half the number of significant digits (two thirds in the case of central differences) of the respective function evaluation can be expected for this class of derivatives, cf. [Alb10b].

AD is essentially applicable when a source code representation of the function by so-called *elemental functions*, like additions, multiplications, sines, etc., for which simple closed-form derivative expressions exist, is available. Such functions are called *factorable functions* (see [Alb10b] for a precise definition). AD is essentially based on the chain rule of differentiation and proceeds by traversing the source code propagating the perturbed elemental functions alongside the

regular function evaluation. In so-called *forward mode*, the derivative evaluation path follows the regular function evaluation path and obtains p directional derivatives (1.17) of a function at the cost of no more than $1 + 1.5\,p$ times the cost of a function evaluation [Gri00], thus comparable to the cost of finite differences. The advantage, however, is that derivatives are accurate up to machine precision.

In *reverse mode* AD can be used to compute p so-called *adjoint directional derivatives* $\bar{\boldsymbol{w}}_{\boldsymbol{i}} \in \mathbb{R}^n$,

$$\bar{\boldsymbol{w}}_{\boldsymbol{i}}{}^{\top} = \bar{\boldsymbol{\nu}}_{\boldsymbol{i}}{}^{\top} \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{w}}(\boldsymbol{w}),$$

of $\boldsymbol{\Phi}$ at $\boldsymbol{w} \in \mathbb{R}^n$ in adjoint direction $\bar{\boldsymbol{\nu}}_i \in \mathbb{R}^m$ for $i \in \{1, \ldots, p\}$ at the cost[10] of no more than $1.5 + 2.5p$ times the cost of an evaluation of $\boldsymbol{\Phi}$ [Gri00]. This is particularly remarkable, since this factor is independent from the dimension n of the input vector of $\boldsymbol{\Phi}$. We refer to [Gri00, GW08, And13, Alb10b] for an in-detail reading on derivative generation in general, and AD in particular.

We will revisit the issue of sensitivity generation in the specific context of dynamic optimization in Section 3.1, where we will also address solution methods for the underlying IVPs.

1.5 Quadratic and Convex Programming

SQP methods require the solution of a convex quadratic subproblem in each iteration. Even beyond SQP, QPs form a very important problem class that appears frequently in real-time optimal control, most notably in the rather large field of linear model predictive control (MPC) where one assumes linearity of the state transition mapping, see Section 2.1.2. For this reason we will consider the specific properties of QPs additionally to the characterizations of the superordinate class of NLPs in Sections 1.3 and 1.4.

Formally, a QP is given by the standard form

$$\min_{\boldsymbol{z} \in \mathbb{R}^n} \quad \frac{1}{2}\, \boldsymbol{z}^{\top} \boldsymbol{H}\, \boldsymbol{z} + \boldsymbol{m}^{\top} \boldsymbol{z} \tag{1.18a}$$

$$\text{s.t.} \quad \boldsymbol{C}\, \boldsymbol{z} = \boldsymbol{c} \tag{1.18b}$$

$$\boldsymbol{D}\, \boldsymbol{z} \leq \boldsymbol{d}, \tag{1.18c}$$

[10] Here, we only concern ourselves with costs in terms of computation time and note that AD in reverse mode may have additional, non-negligible memory requirements.

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ symmetric, $\boldsymbol{m} \in \mathbb{R}^n$, $\boldsymbol{C} \in \mathbb{R}^{n_{\text{eq}} \times n}$, $\boldsymbol{c} \in \mathbb{R}^{n_{\text{eq}}}$, $\boldsymbol{D} \in \mathbb{R}^{n_{\text{ieq}} \times n}$, and $\boldsymbol{d} \in \mathbb{R}^{n_{\text{ieq}}}$. We generally assume that a QP is convex, i.e., that $\boldsymbol{H} \succeq 0$ holds, if not explicitly stated otherwise.

Convex QPs belong to the NLP subclass of so-called *Convex Programming Problems* (CPs), which can be described by (1.6) under the additional assumptions of Definition 1.19. Aside from QPs (at least) two other forms of CPs are relevant in the context of real-time dynamic optimization.

The first class are so-called *quadratically constrained quadratic programming problems* (QCQPs), which extend (1.18) by a set of convex quadratic constraints

$$\boldsymbol{z}^\top \boldsymbol{Q_i} \boldsymbol{z} + \boldsymbol{q_i}^\top \boldsymbol{z} \leq 0 \qquad \forall\, i \in \{1, \dots, m\}\,,$$

where $0 \preceq \boldsymbol{Q_i} \in \mathbb{R}^{n \times n}$ symmetric and $\boldsymbol{q_i} \in \mathbb{R}^n$ for all $i \in \{1, \dots, m\}$. QCQPs play a role in some linear MPC applications for example, where they can be used to efficiently formulate terminal sets, cf., e.g., [BM99, McG00, ZJRM09, DZZ+12].

The second class are *Semidefinite Programming Problems* (SDPs), which can, for example, be used in robust (dynamic) optimization to guarantee worst-case satisfaction of quadratic constraints over a set of uncertain inputs, cf., e.g., [BGFB94, KBM96, Löf03a, Löf03b]. A general form is given through (1.18), extended by a so-called *linear matrix inequality* (LMI)

$$\boldsymbol{Q_0} + \sum_{i=1}^{n} z_i \boldsymbol{Q_i} \preceq 0,$$

where $\boldsymbol{Q_i} \in \mathbb{R}^{n \times n},\ i \in \{0, \dots, n\}$ are symmetric. We note that SDPs are the most general class of CPs considered here, and we have the inclusions that every QP is also a QCQP (trivial) and that every QCQP is an SDP (via a Schur complement transformation, cf. [BV04])[11].

We do not detail the theoretic properties of these CPs here, but refer to the pertinent textbooks, e.g., [BV04]. It is sufficient for our purposes to merely stress that

a) every local solution of a CP is also a global solution (cf. Remark 1.44), and

b) CPs are generally considered significantly more tractable than (non-convex) NLPs.

In the following, we introduce major classes of solution algorithms for CPs that are relevant in the context of real-time dynamic optimization.

[11] A derivation that SDPs indeed are CPs can also be found in [BV04].

1.5.1 Interior-point methods

Identically to Section 1.4.1 we can introduce slack variables $\boldsymbol{s} \in \mathbb{R}^{n_{\text{ieq}}}$ to transform (1.18)[12] into an equality constrained problem with non-negativity requirements on the slack variables $\boldsymbol{s}$ and the inequality multipliers $\boldsymbol{\mu} \in \mathbb{R}^{n_{\text{ieq}}}$.

Just as in the generic nonlinear case, interior-point methods for convex problems iteratively improve an initial primal-dual guess $(\boldsymbol{z^0}, \boldsymbol{\lambda^0}, \boldsymbol{\mu^0}, \boldsymbol{s^0})$, that satisfies $(\boldsymbol{\mu^0}, \boldsymbol{s^0}) > 0$ component-wise, by the update rule

$$(\boldsymbol{z^{k+1}}, \boldsymbol{\lambda^{k+1}}, \boldsymbol{\mu^{k+1}}, \boldsymbol{s^{k+1}}) := (\boldsymbol{z^k}, \boldsymbol{\lambda^k}, \boldsymbol{\mu^k}, \boldsymbol{s^k}) + \alpha^k (\boldsymbol{\Delta z^k}, \boldsymbol{\Delta \lambda^k}, \boldsymbol{\Delta \mu^k}, \boldsymbol{\Delta s^k}).$$

For convex QPs the step directions $(\boldsymbol{\Delta z^k}, \boldsymbol{\Delta \lambda^k}, \boldsymbol{\Delta \mu^k}, \boldsymbol{\Delta s^k})$ are computed from

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{C}^\top & \boldsymbol{D}^\top & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\mathrm{I}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{S^k} & \boldsymbol{M^k} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta z^k} \\ \boldsymbol{\Delta \lambda^k} \\ \boldsymbol{\Delta \mu^k} \\ \boldsymbol{\Delta s^k} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{r_d} \\ -\boldsymbol{r_{\text{eq}}} \\ -\boldsymbol{r_{\text{ieq}}} - \boldsymbol{s^k} \\ -\boldsymbol{S^k M^k e} + \tau^k \boldsymbol{e} \end{bmatrix}, \tag{1.19}$$

with the dual and primal residual terms

$$\boldsymbol{r_d} = \boldsymbol{m} + \boldsymbol{H}\, \boldsymbol{z^k} + \boldsymbol{C}^\top \boldsymbol{\lambda^k} + \boldsymbol{D}^\top \boldsymbol{\mu^k}$$

$$\boldsymbol{r_{\text{eq}}} = \boldsymbol{C}\, \boldsymbol{z^k} - \boldsymbol{c}$$

$$\boldsymbol{r_{\text{ieq}}} = \boldsymbol{D}\, \boldsymbol{z^k} - \boldsymbol{d}\,.$$

Here, $\boldsymbol{S}$, $\boldsymbol{M}$, and $\boldsymbol{e}$ are as defined in Section 1.4.1. The barrier parameter τ^k is gradually driven to zero.

For more general convex Problems, step directions have to be computed from a perturbed KKT system with $\boldsymbol{H}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ contribution updated in each iteration, as detailed in Section 1.4.1. Note however, that using the exact Hessian of the CP Lagrangian is feasible in this case due to convexity.

The performance of an interior-point method of course depends crucially on the applied solution method for the structured linear system and the step size selections. Detailed insights can be found in [Wri97, NW06] as well as in [Dom13] regarding the structure-exploiting solution of KKT systems arising in the context of dynamic optimization.

[12] We specifically discuss the convex QP case here. It will become clear below that the generic convex case is directly covered by Section 1.4.1.

1.5.2 First-order methods

The solution of a KKT system similar to (1.19) by direct linear algebra routines (possibly after structure-exploiting transformations and reorderings) generally involves factorizations of the Hessian matrix $\boldsymbol{H}$ or of projections of it. In a sense, methods requiring the solution of KKT-like systems can therefore be seen as the analogy of Newton's method in the unconstrained case and are consequently often subsumed under the term *second-order methods*.

In some situations, it may be intractable, too expensive, or for other reasons undesirable to factorize (or even to compute entirely) the second-order term $\boldsymbol{H}$. In analogy to the gradient method in unconstrained optimization one then seeks to only use first-order information (i.e., gradients, or subgradients) of the data of the convex problem, obtaining a *first-order method*. (In-) equality constraints may be treated in this context by penalty terms, projection of the computed descent direction onto a feasible set, or by dual decomposition (see also Chapter 4).

A large variety of first-order methods exist. The major advantages of first-order methods are that they are typically rather easy to implement, and that rather tight upper bounds on the required number of iterations to achieve a solution of a certain accuracy exist for several methods. These methods are, however, most efficient when only a solution of low or medium accuracy is required, and when the inequality constraint set is simple (e.g., consisting only of variable bounds). We refer to [Nes04] for a deeper theoretical analysis, as well as to [TD12, KCD13, Koz13] for an overview of existing first-order methods that are relevant in the context of dynamic optimization.

1.5.3 Active-set methods

Our presentation is loosely based on [NW06] and [Fer11]. We adopt a primal point of view in the following[13] and briefly sketch the main concepts. A more thorough and detailed discussion can be found, for example, in [Fle87, NW06].

The central idea behind active-set methods for quadratic programming is very similar to the Simplex method for linear programming. A major difference, however, is that, due to the curvature in the objective function, the optimal QP solution does not necessarily need to be found in a corner, but may also lie uniquely within a facet, or even completely in the interior of the feasible region.

[13] Note that a dual active-set method can, in essence, be identified with a primal active-set method applied to the dual problem.

The main procedure nevertheless is — identically to the Simplex method — to iteratively improve an initial guess of the optimal active set by adding or dropping individual inequality constraints $j \in \mathcal{I}$ given by

$$\boldsymbol{D}_{\boldsymbol{j},.}\, \boldsymbol{z} \leq d_j$$

in (1.18) until the correct active set is identified. For each fixed so-called *working set*[14] $\mathcal{W}^i \subseteq \mathcal{I}$ we solve the associated equality-constrained QP

$$\min_{\boldsymbol{v} \in \mathbb{R}^n} \quad \frac{1}{2}\, \boldsymbol{v}^\top \boldsymbol{H} \boldsymbol{v} + \boldsymbol{m}^\top \boldsymbol{v} \tag{1.20a}$$

$$\text{s.t.} \quad \boldsymbol{C}_{\boldsymbol{j},.}\, \boldsymbol{v} = c_j \qquad \forall\, j \in \mathcal{E} \tag{1.20b}$$

$$\boldsymbol{D}_{\boldsymbol{j},.}\, \boldsymbol{v} = d_j \qquad \mid\ \mu_j^i \qquad \forall\, j \in \mathcal{W}^i\,. \tag{1.20c}$$

Since (1.20) does not feature any inequality constraints, its primal-dual solution is directly given as the solution of the KKT system

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{C}^\top & \boldsymbol{D}_{\boldsymbol{\mathcal{W}}}{}^\top \\ \boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}_{\boldsymbol{\mathcal{W}}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v^i} \\ \boldsymbol{\lambda^i} \\ \boldsymbol{\mu^i_{\mathcal{W}}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{m} \\ \boldsymbol{c} \\ \boldsymbol{d_{\mathcal{W}}} \end{bmatrix}. \tag{1.21}$$

Here, $\boldsymbol{D_{\mathcal{W}}}$ and $\boldsymbol{d_{\mathcal{W}}}$ summarize the rows and entries of $\boldsymbol{D}$ and $\boldsymbol{d}$ corresponding to indices in the working set. Conversely, the inequality multipliers $\boldsymbol{\mu}$ of the original problem (1.18) associated with the current subproblem are given by a vector of all zeros except for those indices from the working set, where the corresponding entry from $\boldsymbol{\mu^i_{\mathcal{W}}}$ applies. The particular structure of (1.21) can be exploited for an efficient solution. One can distinguish (at least) two important classes of algorithms by the way (1.21) is solved. The first to be mentioned here are so-called *range-space methods* (sometimes also referred to as *Schur-complement methods*), where the Schur complement of the system matrix is formed to the end of only solving a system in the size of the number of multipliers. Range-space methods are most effective when only few inequality constraints are in the working set. The second class we introduce here are so-called *null-space methods*, which form a basis matrix for the null-space of the fixed constraints, and use this to project the QP Hessian onto the free variables in order to determine the step in the primal variables. The null-space basis matrix may be expensive to form initially, but may be quite effective when the number of the degrees of freedom of (1.21) is small. More details can be found in [NW06].

[14] For notational simplicity we only consider inequality constraints for the working set; all equality constraints are clearly active in any case.

Let us return to our high-level presentation of the active-set algorithm. It is clear that if we had identified the optimal active set, i.e., if $\mathcal{W}^i \cup \mathcal{E} \equiv \mathcal{A}(\boldsymbol{z^*})$[15], the solution $\boldsymbol{v^i}$ of subproblem (1.20) would coincide with the optimal solution of (1.18), $\boldsymbol{z^*}$.

Assume now that $\mathcal{W}^i \cup \mathcal{E} \not\equiv \mathcal{A}(\boldsymbol{z^*})$, and let $\boldsymbol{z^i} \in \mathbb{R}^n$ be feasible for (1.18) and such that (1.20b) and (1.20c) are fulfilled (details on a so-called *Phase-I* procedure to obtain such an initial feasible solution can, for example, be found in [NW06]). If $\boldsymbol{v^i} \neq \boldsymbol{z^i}$ we can move along the direction $\boldsymbol{v^i} - \boldsymbol{z^i}$ and set

$$\boldsymbol{z^{i+1}} := \boldsymbol{z^i} + \alpha^i(\boldsymbol{v^i} - \boldsymbol{z^i})$$

for a carefully chosen step-length parameter $\alpha^i \in [0, 1]$. Observe that $\boldsymbol{z^{i+1}}$ satisfies (1.20b) and (1.20c) for any α^i. Analogously to the well-known Simplex method, the step-length α^i is chosen such that $\boldsymbol{z^{i+1}}$ remains feasible with respect to all constraints of (1.18), i.e., as large as possible until the first *blocking constraint* $\bar{j} \in \mathcal{I}\backslash\mathcal{W}^i$ is reached or $\alpha^i = 1$. If $\alpha^i < 1$ we add the blocking constraint to the working set, i.e., we set[16]

$$\mathcal{W}^{i+1} := \mathcal{W}^i \cup \{\bar{j}\}.$$

The (primal) active-set method continues adding constraints until eventually an iteration index i with $\boldsymbol{v^i} = \boldsymbol{z^i}$ is reached. It can easily be verified that in this case $\boldsymbol{z^i}$ is a KKT point (and therefore the solution of (1.18)) if the inequality multipliers μ^i_j for all constraints $j \in \mathcal{W}^i$ from the working set are non-negative. If there is a multiplier associated with a constraint $\bar{j}$ from the working set with $\mu^i_{\bar{j}} < 0$, we can simply drop this constraint from the working set, i.e., set

$$\mathcal{W}^{i+1} := \mathcal{W}^i\backslash\{\bar{j}\}$$

and continue to iterate.

It can be shown that the active-set method sketched here converges under certain, mild assumptions if (1.18) is strictly convex. Care needs to be taken to handle degeneracy (i.e., facets or corners defined by a linearly dependent subset of constraints) and the resulting danger of cycling appropriately. Details can be found in [Fle87, NW06] and the references therein.

It is crucial to note that, in contrast to interior-point methods for example, the KKT systems (1.21) to be solved in subsequent iterations of the active-set method only differ by one row and column (either a constraint is added or

[15] Recall that inequality constraints index set $\mathcal{I}$ and equality constraints index set $\mathcal{E}$ are assumed to be disjoint.

[16] To be precise, particular care needs to be taken to ensure linear independence of the constraint gradients in the working set. We refer to [NW06] for details.

removed). The matrix factorizations required for the solution of theses systems can therefore be reused and cheaply updated, alongside with the null-space basis matrix (if a null-space method is used for the solution of the equality-constrained subproblems); details of the updates are discussed, for example, in [GMSW84, NW06]. These matrix updates are absolutely essential for the efficiency and competitiveness of active-set methods.

1.5.4 Parametric active-set methods

Parametric active-set methods for quadratic programming have been proposed and analyzed in [Bes96, Fer06, FBD08, Fer11, KPBS13, PKBS10]. We follow [FBD08] in our presentation. The open-source implementation qpOASES [FBD08] of a particular parametric active-set method, the so-called *Online Active-Set Strategy*, is used and referred to occasionally throughout this thesis.

The motivation for parametric QP solvers, or parametric programming in general, is to solve a sequence of related problems that only differ by some of their data. In the context of dynamic optimization, such sequences of problems may arise, for example, in SQP methods or even more so in model predictive control (MPC) of linear time-invariant (LTI) systems (cf. Section 2.1.2), where only the initial system state changes. Instead of solving a new problem with the changed data, the idea of parametric active-set methods like the Online Active-Set Strategy is to gradually "morph" a QP with known optimal solution (e.g., the previously solved QP) to the desired problem, maintaining primal and dual optimality of the solution. The key observation behind this procedure is that the primal and dual solution of Problem (1.18) depends piecewise affinely on changes in the linear objective term coefficient $\boldsymbol{m}$ and the constraint right-hand sides $\boldsymbol{c}$ and $\boldsymbol{d}$, a fact that is well-known and heavily exploited, for example, in the area of so-called *explicit MPC*, cf. [Zaf90, BMDP02]. Kinks in this solution homotopy may occur exclusively (and generally do occur) at active-set changes.

To make this brief introduction slightly more formal, let us assume that we have solved a QP

$$\min_{\boldsymbol{z}\in\mathbb{R}^n} \quad \frac{1}{2}\,\boldsymbol{z}^\top \boldsymbol{H}\,\boldsymbol{z} + \boldsymbol{m}^\top \boldsymbol{z} \tag{1.22a}$$

$$\text{s.t.} \quad \boldsymbol{C}\,\boldsymbol{z} = \boldsymbol{c} \tag{1.22b}$$

$$\boldsymbol{D}\,\boldsymbol{z} \leq \boldsymbol{d}\,, \tag{1.22c}$$

i.e., that we have a primal-dual solution $(\boldsymbol{z}^{*,\mathbf{0}}, \boldsymbol{\lambda}^{*,\mathbf{0}}, \boldsymbol{\mu}^{*,\mathbf{0}})$. With the aim of finding the (yet unknown) solution $(\boldsymbol{z}^{*,\mathbf{1}}, \boldsymbol{\lambda}^{*,\mathbf{1}}, \boldsymbol{\mu}^{*,\mathbf{1}})$ of the modified problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^n} \quad \frac{1}{2}\,\boldsymbol{z}^\top \boldsymbol{H}\,\boldsymbol{z} + \tilde{\boldsymbol{m}}^\top \boldsymbol{z} \tag{1.23a}$$

$$\text{s.t.} \quad \boldsymbol{C}\,\boldsymbol{z} = \tilde{\boldsymbol{c}} \tag{1.23b}$$

$$\boldsymbol{D}\,\boldsymbol{z} \leq \tilde{\boldsymbol{d}}, \tag{1.23c}$$

we regard the solution homotopy $(\boldsymbol{z}^*(\tau), \boldsymbol{\lambda}^*(\tau), \boldsymbol{\mu}^*(\tau))$ for a homotopy parameter $\tau \in [0,1]$ alongside the vector term parametrization

$$\begin{aligned}
\boldsymbol{m}(\tau) &:= \boldsymbol{m} + \tau\boldsymbol{\Delta m}\,, && \text{where } \boldsymbol{\Delta m} := \tilde{\boldsymbol{m}} - \boldsymbol{m} \\
\boldsymbol{c}(\tau) &:= \boldsymbol{c} + \tau\boldsymbol{\Delta c}\,, && \text{where } \boldsymbol{\Delta c} := \tilde{\boldsymbol{c}} - \boldsymbol{c} \\
\boldsymbol{d}(\tau) &:= \boldsymbol{d} + \tau\boldsymbol{\Delta d}\,, && \text{where } \boldsymbol{\Delta d} := \tilde{\boldsymbol{d}} - \boldsymbol{d}\,.
\end{aligned}$$

Denoting the constraints which are active in a solution $(\boldsymbol{z}^*(\tau), \boldsymbol{\lambda}^*(\tau), \boldsymbol{\mu}^*(\tau))$ by $\mathcal{A}(\tau)$, we know that in each τ a solution is characterized by the KKT conditions

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{C}^\top & \boldsymbol{D}_{\boldsymbol{\mathcal{A}(\tau)}}{}^\top \\ \boldsymbol{C} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{D}_{\boldsymbol{\mathcal{A}(\tau)}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}^*(\tau) \\ \boldsymbol{\lambda}^*(\tau) \\ \boldsymbol{\mu}^*_{\boldsymbol{\mathcal{A}(\tau)}}(\tau) \end{bmatrix} = \begin{bmatrix} -\boldsymbol{m}(\tau) \\ \boldsymbol{c}(\tau) \\ \boldsymbol{d}_{\boldsymbol{\mathcal{A}(\tau)}}(\tau) \end{bmatrix}.$$

Since the active set is constant in a small neighborhood of $(\boldsymbol{z}^{*,\mathbf{0}}, \boldsymbol{\lambda}^{*,\mathbf{0}}, \boldsymbol{\mu}^{*,\mathbf{0}})$ (which may actually be of diameter 0 in the degenerate case of weakly active constraints), we have the local relationship

$$\begin{bmatrix} \boldsymbol{z}^*(\tau) \\ \boldsymbol{\lambda}^*(\tau) \\ \boldsymbol{\mu}^*_{\boldsymbol{\mathcal{A}(\tau)}}(\tau) \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}^{*,\mathbf{0}} \\ \boldsymbol{\lambda}^{*,\mathbf{0}} \\ \boldsymbol{\mu}^{*,\mathbf{0}}_{\boldsymbol{\mathcal{A}(0}} \end{bmatrix} + \tau \begin{bmatrix} \boldsymbol{\Delta z}^* \\ \boldsymbol{\Delta\lambda}^* \\ \boldsymbol{\Delta\mu}^*_{\boldsymbol{\mathcal{A}(0)}} \end{bmatrix},$$

where the increments satisfy

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{C}^\top & \boldsymbol{D}_{\boldsymbol{\mathcal{A}(0)}}{}^\top \\ \boldsymbol{C} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{D}_{\boldsymbol{\mathcal{A}(0)}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta z}^*(\tau) \\ \boldsymbol{\Delta\lambda}^*(\tau) \\ \boldsymbol{\Delta\mu}^*_{\boldsymbol{\mathcal{A}(0)}}(\tau) \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\Delta m} \\ \boldsymbol{\Delta c} \\ \boldsymbol{\Delta d}_{\boldsymbol{\mathcal{A}(0)}} \end{bmatrix} \tag{1.24}$$

for all τ up to a certain $\tau_{\max} \geq 0$, which is determined by the first active-set change. Analogously to regular active-set methods, $\tau_{\max}$ can be computed as the point at which either a previously inactive constraint becomes active, or a previously positive (inequality) multiplier vanishes (indicating the removal of a constraint from the active set/working set). Just as in a regular active-set

method, the auxiliary system (1.24) can be solved in a structure-exploiting fashion by a null-space or a range-space approach and the factorizations can be updated cheaply. We refer to [Fer11] for details.

With the updated active set, we continue following the solution homotopy until eventually $\tau = 1$, which means that (1.23) is solved. Details on how to handle degeneracy in this context can be found in [Fer11].

Regarding the issue of how to find an initial optimal solution, we note that $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ is trivially optimal if we start with $\boldsymbol{m} = \mathbf{0}$, $\boldsymbol{c} = \mathbf{0}$, $\boldsymbol{d} = \mathbf{0}$. This starting point could, in principle, also be used whenever matrix data changes and the factorizations need to be updated. In [Fer11], however, more sophisticated approaches to warm-start parametric active-set methods from an initial guess of the active set or the primal-dual solution are detailed.

Chapter 2

Dynamic Optimization in Real-time

This chapter introduces dynamic estimation and optimal control problems in the light of *online* process surveillance and control, where new measurements arrive periodically, and state and parameter estimates as well as the applied control law need to be updated based on these observations. First, the control theoretic frameworks of *model predictive control* (MPC) and *moving horizon estimation* (MHE) are introduced. At its core, this chapter focuses on algorithmic developments that are tailored to the online solution of the arising dynamic optimization problems. Important developments from the last 15 years, such as the *Real-Time Iteration scheme*, are put into context, and some new extensions are presented. A parallelization scheme for the proposed algorithm class is given, and the prototyping software implementation *CHUCS*[1] is presented. The effectiveness of the proposed algorithms is demonstrated at the example of a benchmark problem from the area of autonomous driving at the end of the chapter.

Acknowledgement Parts of this chapter have been published in the paper "Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control" by Janick Frasch, Leonard Wirsching, Sebastian Sager, and Hans Georg Bock [FWSB12]. Janick Frasch and Leonard Wirsching are the main authors of this paper. Janick Frasch contributed the original conception and software design, while Leonard Wirsching contributed the formalization. The adapted

[1] *CHUCS* stands for *C*oncurrently *H*ierarchically *U*pdating *C*ontroller *S*chemes.

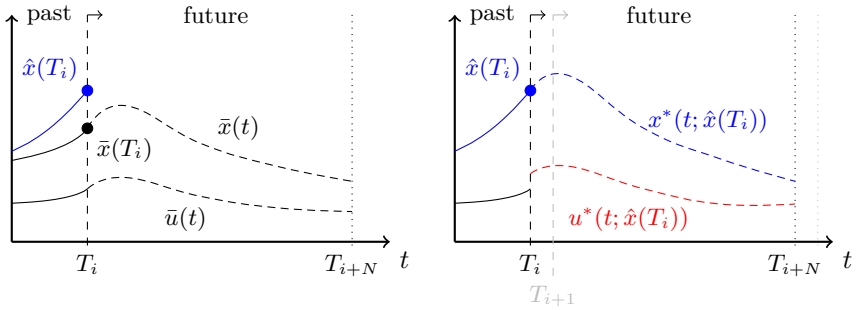


Figure 2.1: The principle of model predictive control: Based on an online state estimate and a model prediction, the control signal is reoptimized; this procedure is repeated periodically.

condensing algorithm from the original paper was conceived in joint discussions. Sebastian Sager and Hans Georg Bock served as advisors and vital discussion partners.

2.1 Model Predictive Control

For any real-world dynamic system, process model (1.1) is, in general, only an approximate description. Systematic modeling errors and unforeseen external influences may therefore cause the system to miss the desired optimal performance when applying a precomputed input signal $u \in \mathcal{U}$ *open-loop*; even significantly more severe consequences may arise, e.g., when the real-world system, referred to as *plant*[2], exceeds a desired limitation (constraint), or is not stabilized at the desired reference.

In general one can deal with this challenge in two ways. On the one hand, we can include safety margins of various kinds into the optimal control problem and/or the resulting NLP. The research areas of *robust optimal control* and *robust optimization*, as well as *stochastic optimal control* investigate such approaches. For an introduction to robust control and optimization, we refer to [Hou11], [BTEGN09], and the references therein; for further reading on stochastic control we recommend [Ber95].

In this thesis, on the other hand, we focus on *feedback* approaches that operate a dynamic system in *closed-loop*. Among feedback control approaches, *model*

[2]We adopt a very generic understanding of *plant* here, meaning any realization of system (1.1).

predictive control (MPC) is a rather generic principle and therefore one of the most powerful concepts[3]. MPC generates a regulating system input *online*, i.e., while the plant is running, by repeatedly solving a dynamic optimization problem, therein minimizing an optimal control objective. Figure 2.1 visualizes the essential concept. Applying the control signal $\bar{u} \in \mathcal{U}$, the dynamic system x is observed at time[4] T_i, where $i \in \mathbb{N}$, to be in state $\hat{x}(T_i)$ instead of the planned/desired state $\bar{x}(T_i)$ due to unmodeled dynamics or perturbations. The MPC approach is to re-solve an optimal control problem on the (typically finite) *prediction horizon* $[T_i, T_{i+N}] \equiv [t_0, t_f] = \mathcal{T}$, yielding an updated optimal predicted state trajectory x^* that is to be realized by applying the updated control signal u^*. After a certain period of time $\Delta T_i := T_{i+1} - T_i$, the so-called *sampling period*, this procedure is repeated for a new state observation $\hat{x}(T_{i+1})$. In general, the sampling grid $T_0, T_1, \ldots$ does not need to be equidistant; however, aligning it with the discretization grid of a direct method to be applied to the problem may facilitate algorithm warmstarting, cf. Section 2.3.1.

Depending on the problem, the prediction horizon at each sampling time may either always end at the same fixed time T_{end}, i.e., be of *shrinking* length $T_{\text{end}} - T_i,\ i = 0, 1, \ldots, N$, or may be infinitely *receding*, i.e., $\mathcal{T}_i \equiv [T_i, T_{i+N}]$ is used at each sampling time $T_0, T_1, \ldots$. Shrinking horizon problems may appear, for example, in batch operation of certain processes in chemical engineering, while classical receding horizon problems may appear in systems that are intended to be operated for time periods significantly longer than the length of the prediction horizon (or even of indetermined length), like, for example, the operation of a power plant.

We note that in the receding horizon context the finiteness of the prediction horizon length in combination with the MPC feedback strategy of only applying part of the computed control signal may actually result in unexpected and undesirable system behavior, even in the nominal case, where we assume perfect knowledge of the system dynamics. Such effects, among other things, are a strong motivation for stability investigations of MPC schemes, which are, however, in their full spectrum beyond the scope of this thesis.

Still, to give only a very brief summary, it is often possible to establish stability guarantees, which can be seen as a kind of "well-behavedness" guarantees on the system-MPC interaction, if suitable terminal constraints or penalties are formulated in the MPC scheme, or if the prediction horizon is sufficiently long. We refer to [RM09, GP11, Grü12] for more details.

[3] For a more general introduction and an overview of other feedback concepts, we kindly refer the reader to standard textbooks, like[FPEN09].

[4] Note, also for future reference, that in the context of online computations we use upper case $T_i,\ i \in \mathbb{N}$ to denote time in a global frame, while lower case $t_i,\ i \in \mathbb{N}$ denote time in a local, i.e., algorithm- or prediction-centered frame.

2.1.1 MPC problem formulations

In a generic form, the MPC problem to be solved at each sampling time T_i is summarized by

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{k=0}^{N-1} \ell_k(\boldsymbol{x_k}, \boldsymbol{u_k}) + \ell_N(\boldsymbol{x_N}) \tag{MPC1}$$

$$\text{s.t.} \quad \boldsymbol{x_{k+1}} = \boldsymbol{F_k}(\boldsymbol{x_k}, \boldsymbol{u_k}) \qquad \forall\ k \in \mathcal{S}_N \tag{MPC2}$$

$$\mathbf{0} \geq \boldsymbol{d_k}(\boldsymbol{x_k}, \boldsymbol{u_k}) \qquad \forall\ k \in \mathcal{S} \tag{MPC3}$$

$$\boldsymbol{x_0} = \hat{\boldsymbol{x}}_\mathbf{0}. \tag{MPC4}$$

Here, we follow the presentation commonly adopted in the literature (cf., e.g., [CB07, RM09, GP11]) and introduce (MPC) as a discretized variant of (DOP), anticipating the solution by a direct method, cf. Section 1.2. This means in particular that the continuous time control functions $\boldsymbol{u}(t)$ are parameterized on each stage $k \in \mathcal{S}_N$ by a finite dimensional vector $\boldsymbol{u_k}$; w.l.o.g. we assume $\boldsymbol{u_k} \in \mathbb{R}^{n_u}\ \forall k \in \mathcal{S}_N$. The (fully determined) discretized state variables $\boldsymbol{x_k} := \boldsymbol{x}(t_k) \in \mathbb{R}^{n_x}$, where $\{t_k\}_{k\in\mathcal{S}}$ defines a grid on $\mathcal{T}$, are kept for clarity of the presentation and for algorithmic exploitation. Since we are only going to consider discretized optimization variables for the remainder of this section, we introduce the shorthands $\boldsymbol{x} := (\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_N})$ and $\boldsymbol{u} := (\boldsymbol{u_0}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_{N-1}})$ in this context. The *initial value* of the system, $\hat{\boldsymbol{x}}_\mathbf{0} := \hat{\boldsymbol{x}}(t_0) = \hat{\boldsymbol{x}}(T_i)$, is assumed to be fully known, for example through an MHE, cf. Section 2.2; (partially) free initial conditions are not considered here, even though the presented algorithms would, in principle, support it. State constraints $\boldsymbol{d_k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_d},\ k \in \mathcal{S}$ are only enforced on a finite grid; we refer to the discussion in Section 1.2.1. The separable objective function contributions $\ell_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R},\ k \in \mathcal{S}_N$ and $\ell_N : \mathbb{R}^{n_x} \to \mathbb{R}$ are referred to as *stage costs*; ℓ_N is in particular also known as terminal cost. W.l.o.g. we assume an identical grid for control and constraint (and objective) discretization here, but stress that, in principle, these grids may be chosen independently from one another.

Classically, MPC is particularly concerned with so-called *tracking* objectives, which are generally considered computationally more tractable, and for which considerable theoretical foundations exist, cf. [RM09, GP11]. In tracking MPC, one tries to minimize the deviation of the actual system state $\boldsymbol{x}(\cdot)$ from a desired *reference state* $\bar{\boldsymbol{x}}(\cdot)$ and the deviation of the control input $\boldsymbol{u}(\cdot)$ from a *reference input* signal $\bar{\boldsymbol{u}}(\cdot)$ over the prediction horizon $\mathcal{T}$. The corresponding stage costs

are of least-squares form and read

$$\ell_k(\boldsymbol{x_k}, \boldsymbol{u_k}) = \frac{1}{2}\left(\|\boldsymbol{x_k} - \bar{\boldsymbol{x}}_{\boldsymbol{k}}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u_k} - \bar{\boldsymbol{u}}_{\boldsymbol{k}}\|^2_{\boldsymbol{R}}\right) \qquad \forall\, k \in \mathcal{S}_N$$

and

$$\ell_N(\boldsymbol{x_N}) = \frac{1}{2}\|\boldsymbol{x_N} - \bar{\boldsymbol{x}}_{\boldsymbol{N}}\|^2_{\boldsymbol{P}},$$

where the weighting matrices $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$, and $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$ are tuning parameters of the respective *MPC scheme* that are typically chosen positive semidefinite or positive definite. In this context, $\boldsymbol{P}$ is sometimes also referred to as *terminal cost*. The tracking reference signals $\bar{\boldsymbol{x}}(\cdot)$ and $\bar{\boldsymbol{u}}(\cdot)$ are often equilibria of the considered dynamic system given through $F_k,\ k \in \mathcal{S}_N$, but also precomputed, optimized trajectories or other setpoints are used.

Particularly in tracking MPC of equilibria the notion of *stability* is an important concept. A large variety of stability definitions exists, each with its own quantification of data and model uncertainty and corresponding concepts for establishing the respective stability guarantees for a certain MPC scheme. As investigations from this rather large research area are beyond the scope of this thesis, we kindly refer to the textbooks [RM09, GP11] and the references therein for complementary reading. For completeness, we only establish two concepts which very often are the key ingredient to establish stability guarantees for certain MPC schemes, the notion of a *terminal (point) constraint* and its generalization to a *terminal set constraint*. A terminal constraint is simply established by demanding the state $\boldsymbol{x}$ to attain a certain fixed value (often the origin of a suitably shifted system) at the end of the prediction horizon through

$$\boldsymbol{x_N} = \bar{\boldsymbol{x}},$$

which obviously can be expressed by (MPC3). A terminal set constraint

$$\boldsymbol{x_N} \in \bar{\mathcal{X}},$$

which demands the system state to attain a value from the terminal set $\bar{\mathcal{X}} \subset \mathbb{R}^{n_x}$ at the end of the prediction horizon, can, in priciple, also be expressed by (MPC3), under the assumption that a suitable representation of $\bar{\mathcal{X}}$ by a finite number of inequalities exists.

Besides tracking objective function formulations also *economic* objectives are of interest in MPC. Here, instead of following a precomputed reference, one aims to achieve a more sophisticated, direct goal such as maximizing yield, minimizing

time, minimizing operational costs, etc. by minimizing arbitrary[5] stage costs $\ell_k,\ k \in \mathcal{S}$ that are not restricted to a least-squares form. Nonwithstanding this, certain conditions may permit a positive definite reformulation of economic objectives, which can then in turn be approximated by a least-squares form (see for example [ARA11, ZGD14]).

2.1.2 Linear MPC

Instances of MPC where $\boldsymbol{F_k}$ and $\boldsymbol{d_k}$ are linear, and ℓ_k are quadratic functions in $\boldsymbol{x_k}$ and $\boldsymbol{u_k}$ for all $k \in \mathcal{S}$ and $k \in \mathcal{S}_N$, respectively, are called *linear MPC*. In its canonical form, a linear MPC problem is given by

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{k=0}^{N-1} \left(\frac{1}{2} \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{Q_k} & \boldsymbol{S_k} \\ \boldsymbol{S_k}^\top & \boldsymbol{R_k} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix} + \begin{bmatrix} \boldsymbol{q_k} \\ \boldsymbol{r_k} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix} \right) \tag{LMPC1}$$

$$+ \ \frac{1}{2} \boldsymbol{x_N}^\top \boldsymbol{P}\, \boldsymbol{x_N} + \boldsymbol{q_N}^\top \boldsymbol{x_N}$$

$$\text{s.t. } \boldsymbol{x_{k+1}} = \boldsymbol{A_k} \boldsymbol{x_k} + \boldsymbol{B_k} \boldsymbol{u_k} + \boldsymbol{c_k} \qquad \forall\ k \in \mathcal{S}_N \tag{LMPC2}$$

$$\underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k} \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix} \ \leq \ \overline{\boldsymbol{d}}_{\boldsymbol{k}} \qquad \forall\ k \in \mathcal{S}_N \tag{LMPC3}$$

$$\underline{\boldsymbol{d}}_{\boldsymbol{N}} \leq \boldsymbol{D_N} \boldsymbol{x_N} \ \leq \ \overline{\boldsymbol{d}}_{\boldsymbol{N}} \tag{LMPC4}$$

$$\boldsymbol{x_0} = \hat{\boldsymbol{x}}_{\boldsymbol{0}}, \tag{LMPC5}$$

where $\boldsymbol{Q_k} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{S_k} \in \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{R_k} \in \mathbb{R}^{n_u \times n_u}$, $\boldsymbol{A_k} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{B_k} \in \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{D_k} \in \mathbb{R}^{(n_x+n_u) \times n_d}$ for all $k \in \mathcal{S}_N$, and $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{D_N} \in \mathbb{R}^{n_x \times n_d}$. We further have $\boldsymbol{q_k} \in \mathbb{R}^{n_x}$, $\underline{\boldsymbol{d}}_{\boldsymbol{k}} \in \mathbb{R}^{n_d}$, and $\overline{\boldsymbol{d}}_{\boldsymbol{k}} \in \mathbb{R}^{n_d}$ for all $k \in \mathcal{S}$, as well as $\boldsymbol{r_k} \in \mathbb{R}^{n_u}$ and $\boldsymbol{c_k} \in \mathbb{R}^{n_x}$ for all $k \in \mathcal{S}_N$.

Often, one additionally assumes $\boldsymbol{R_k} \succ \boldsymbol{0}\ \forall k \in \mathcal{S}_N$ in linear MPC, which is sufficient to render the solution of (LMPC) unique (recall that only $\boldsymbol{u_k},\ k \in \mathcal{S}$ are true degrees of freedom), as well as $\begin{bmatrix} \boldsymbol{Q_k} & \boldsymbol{S_k} \\ \boldsymbol{S_k}^\top & \boldsymbol{R_k} \end{bmatrix} \succeq \boldsymbol{0}$ and $\boldsymbol{P} \succeq \boldsymbol{0}$ for algorithmic purposes.

If the problem data is independent of the time discretization index k we call (LMPC) a *linear time-invariant* (LTI) MPC problem. Otherwise (LMPC) is called *linear time-varying* (LTV).

[5] Assumption 1.18 remains unaffected.

If no stage constraints are present, i.e., if $n_d = 0$, the solution of (LMPC) via dynamic programming is tractable, since the exact solution of the dynamic programming recursion step, the so-called *cost to go function*, can be stated as a closed quadratic form in the state, cf. [RM09].

If (LMPC) additionally is an LTI problem, also the infinite horizon problem[6]

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \frac{1}{2} \sum_{k=0}^{\infty} \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_k} \\ \boldsymbol{u_k} \end{bmatrix} \tag{LQR1}$$

$$\text{s.t. } \boldsymbol{x_{k+1}} = \boldsymbol{A x_k} + \boldsymbol{B u_k} \qquad \forall\, k \in \mathcal{S}_\infty \tag{LQR2}$$

$$\boldsymbol{x_0} = \hat{\boldsymbol{x}}_\mathbf{0}, \tag{LQR3}$$

which is often referred to as *linear quadratic regulator* (LQR), is tractable, and its feedback law is given by $\boldsymbol{u^*}(\boldsymbol{x}) = -\boldsymbol{K}\,\boldsymbol{x}$, with the so-called *LQR gain*

$$\boldsymbol{K} = \left(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{B}\right)^{-1} \left(\boldsymbol{S} + \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A}\right).$$

In this context, $\boldsymbol{P}$ denotes the (positive definite) solution to the *discrete time algebraic Riccati equation* of the underlying system,

$$\boldsymbol{P} = \boldsymbol{Q} - \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} - \left(\boldsymbol{S}^\top + \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B}\right) \left(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{B}\right)^{-1} \left(\boldsymbol{S} + \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A}\right).$$

2.2 Moving Horizon Estimation

In (MPC), we assumed a fully known initial condition of the dynamic system, $\hat{\boldsymbol{x}}_\mathbf{0} \in \mathbb{R}^{n_x}$, as well as a known parameter vector $\boldsymbol{p} \in \mathbb{R}^{n_p}$. *Full state observation*, however, can in general not be guaranteed in real-world systems, and parameters may actually vary during operation.

Moving horizon estimation (MHE) is a technique to recover $\hat{\boldsymbol{x}}_\mathbf{0}$ and $\boldsymbol{p}$ from a series of measurements $\{\boldsymbol{y_j}\}_{k=1,2,\ldots} \subset \mathbb{R}^{n_y}$, taking the system dynamics and possibly physical limitations (in form of constraints) into account. In contrast to a generic *full information estimator* which considers all available measurements (cf. Section 1.1.3), MHE only considers the M most recent measurements (w.l.o.g. denoted by $\{\boldsymbol{y_1}, \ldots, \boldsymbol{y_M}\}$) explicitly in the dynamic optimization problem, while all earlier measurements are either simply ignored, or considered through a statistic, the so-called *prior terms* $\bar{\boldsymbol{x}}_\mathbf{0} \in \mathbb{R}^{n_x}$ and $\bar{\boldsymbol{p}} \in \mathbb{R}^{n_p}$. Figure

[6] We drop the first-order objective terms in this context only for notational convenience; formally correct, this may be obtained, e.g., by a translation of the optimization variables.

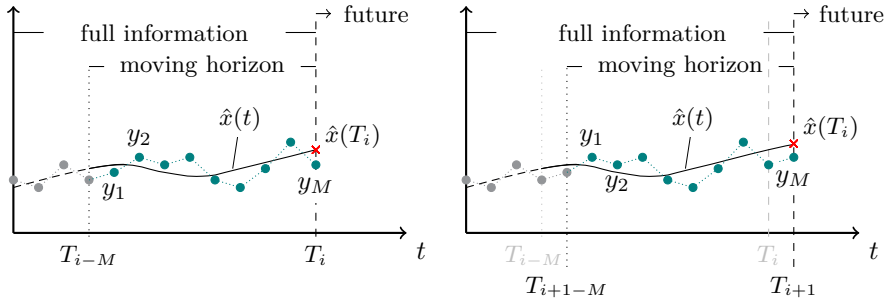


Figure 2.2: Sliding time window in moving horizon estimation at two subsequent sampling times. Green dots show explicitly considered measurements $y_\cdot$, while a red x visualizes the estimated current state $\hat{x}(\cdot)$. Inspired by [RM09].

2.2 visualizes this principle. The motivation for only considering a sliding window of the time horizon containing a fixed (sufficiently large) number of measurements is mainly to render the estimation problem computationally tractable. As the amount of measurements increases while the plant is evolving, the dynamic optimization problem to be solved for the full state estimator becomes increasingly challenging.

One notable exception however is the so-called *Kalman filter*, a pair of one-step recursion formulae for the state/parameter estimate and their corresponding covariance estimate, that solve the dynamic programming problem of a least-squares full information estimator with linear dynamics and without path constraints exactly[7]. The details of this rather well-known algorithm can be found, for example, in [Ste94, RM09].

[7] In some sense, the Kalman filter can be seen as the analogy of the explicit LQR solution for estimation (see [RM09] for details).

2.2.1 MHE problem formulation

In a generic mathematical form, the estimation problem to be solved in an MHE scheme at each sampling time reads

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}} \quad \ell_0(\boldsymbol{x_0},\boldsymbol{p},\bar{\boldsymbol{x}},\bar{\boldsymbol{p}}) + \sum_{j=1}^{M} \ell_j(\boldsymbol{x}(t_j),\boldsymbol{u}(t_j),\boldsymbol{p},\boldsymbol{y_j}) \tag{MHE1}$$

$$\text{s.t.} \quad \boldsymbol{x_{k+1}} = \boldsymbol{F_k}(\boldsymbol{x_k},\boldsymbol{u_k},\boldsymbol{p}) \qquad \forall\ k \in \mathcal{S}_N \tag{MHE2}$$

$$\mathbf{0} \geq \boldsymbol{d_k}(\boldsymbol{x_k},\boldsymbol{u_k},\boldsymbol{p}). \qquad \forall\ k \in \mathcal{S} \tag{MHE3}$$

Analogously to (MPC), we present (MHE) as a discretized instance of (DOP) in anticipation of the solution using direct simultaneous methods. This presentation is consistent with standard literature, e.g., [RM09]. Again, we group the optimization variables introducing $\boldsymbol{x} := (\boldsymbol{x_0},\boldsymbol{x_1},\ldots,\boldsymbol{x_N})$ and $\boldsymbol{u} := (\boldsymbol{u_0},\boldsymbol{u_1},\ldots,\boldsymbol{u_{N-1}})$ for notational convenience. As in the context of MPC, $\boldsymbol{x_k} \in \mathbb{R}^{n_x},\ k \in \mathcal{S}$ denote the discretized state variables, and $\boldsymbol{u_k},\ k \in \mathcal{S}_N$ (w.l.o.g. $\boldsymbol{u_k} \in \mathbb{R}^{n_u}\ \forall k \in \mathcal{S}_N$) denote the control parameterization variables. In the interplay with MPC (or another control scheme), the (optimal or approximately optimal) terminal state vector $\boldsymbol{x_N}^{(\mathrm{MHE})}$, which is the estimate of the current state of the system, $\hat{\boldsymbol{x}}(T_i)$, at sampling time T_i, serves as initial condition $\hat{\boldsymbol{x}}_{\mathbf{0}}^{(\mathrm{MPC})}$ for the prediction of the controller.

As in (MPC), transition functions $\boldsymbol{F_k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ model the system dynamics, and discretized path constraints $\boldsymbol{d_k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ contain physical limitations of the system that guide the estimator.

In general, the evaluation of the objective function terms $\ell_j : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_y} \to \mathbb{R},\ j \in \{1,\ldots,M\}$ (more precisely the evaluation of $\boldsymbol{x}(t_j)$) requires the use of numerical integration routines that return a continuous representation of the system dynamics (see also Section 3.1). Efficient implementations of such methods have been described, e.g., in [Alb10a] and [QGD13]. While using coarser grids for control/state discretization can be useful in case of *multi-rate measurements*, where some measurements $\{\boldsymbol{y_j^i}\}_{j=1,2,\ldots},\ i \in \{1,\ldots n_y\}$ are available at higher rates than others, we note that theoretically, measurement and algorithmic grid could be aligned, i.e., we could choose $N := M$.

In the context of MHE, the deviations of the system's initial state $\boldsymbol{x_0}$ and the parameter vector $\boldsymbol{p}$ from the *priors* $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{p}}$, respectively, are penalized by a so-called *arrival cost* $\ell_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$. Section 2.2.2 will outline ideas for subsuming the not explicitly considered measurement history into the prior information terms.

The stage costs $\ell_j,\ j \in \{0, \dots M\}$ are typically convex in MHE. In particular, a very common choice is simply the least-squares form

$$\ell_j(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), \boldsymbol{p}, \boldsymbol{y_j}) = \|\boldsymbol{y_j} - \boldsymbol{h_j}(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), \boldsymbol{p})\|^2_{\boldsymbol{V_j}} \tag{2.5}$$

for $k \in \{1, \dots, M\}$, and

$$\ell_0(\boldsymbol{x_0}, \boldsymbol{p}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{p}}) = \left\| \begin{bmatrix} \boldsymbol{x_0} - \bar{\boldsymbol{x}} \\ \boldsymbol{p} - \bar{\boldsymbol{p}} \end{bmatrix} \right\|^2_{\boldsymbol{P}} \tag{2.6}$$

for the arrival cost, where $\boldsymbol{V_j} \in \mathbb{R}^{n_y \times n_y},\ j \in \{1, \dots, M\}$ and $\boldsymbol{P} \in \mathbb{R}^{(n_x+n_p)\times(n_x+n_p)}$ are appropriately chosen weighting matrices, and $\boldsymbol{h_j} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$ is a suitable output function, cf. Section 1.1.3. We will shed light on appropriate choices of the weighting matrices in Section 2.2.3.

Some MHE problem formulations assume the presence of so-called *state noise*. Formally, this corresponds to having system dynamics

$$\boldsymbol{x_{k+1}} = \boldsymbol{F_k}(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{p}) + \boldsymbol{\zeta_{k+1}} \qquad \forall\ k \in \mathcal{S}_N$$

in lieu of (MHE2), where $\boldsymbol{\zeta_k} : \Omega \to \mathbb{R}^{n_x},\ k \in \mathcal{S}_1$ are independent random variables on a suitably chosen probability space[8] $(\Omega, \mathcal{F}, \mathcal{P})$. While generally the state noise formulation of the system dynamics can simply be cast in form (MHE) by introducing auxiliary variables (e.g., in form of additional discretized controls)[9], the corresponding least-squares stage cost (2.5) becomes

$$\ell_j(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), \boldsymbol{\zeta_j}, \boldsymbol{p}, \boldsymbol{y_j}) = \|\boldsymbol{y_j} - \boldsymbol{h_j}(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), \boldsymbol{p})\|^2_{\boldsymbol{V_j}} + \|\boldsymbol{\zeta_j}\|^2_{\boldsymbol{W_j}}\,. \tag{2.7}$$

Here, we assumed $M = N$ for notational convenience.

While (2.5)/(2.7) negotiate a (weighted) "mean" fit of the state trajectory to the measurements that is rather stable (i.e., shows little sensitivity towards small data changes), it tends to get influenced strongly by outliers. In an attempt to increase robustness against measurement outliers, the so-called *Huber penalty function*[10] [Hub81] is occasionally used as a robust alternative in praxis. The Huber penalty $\mathcal{H}_\sigma : \mathbb{R} \to \mathbb{R}_{0+}$ combines the local "smoothness" of the ℓ^2 norm

[8] We omit details here, but refer the reader to introductory literature on statistics, like [KCM05], for complementary reading.

[9] An efficient implementation may need, of course, to exploit the specific structures of these controls, e.g., during the derivative generation.

[10] The Huber penalty is sometimes also known as Huber norm; since, mathematically speaking, this measure however only switches between two norms, but itself lacks the positive homogeneity requirement of a norm, we stick to the term penalty here.

with the robustness of the ℓ^1 norm in a convex, continuously differentiable function given by

$$\mathcal{H}_\sigma(a) := \begin{cases} \frac{1}{2}a^2 & |a| \leq \sigma \\ \sigma\left(|a| - \frac{1}{2}\sigma\right) & |a| > \sigma \end{cases}.$$

A visualization and further interpretations can be found, e.g., in [BV04]. We also refer to [BV04, GD13] for details regarding the efficient implementation of Huber penalties in optimization problems.

For MHE based on M measurements of the dynamic system defined by $\boldsymbol{F_k}$, $k \in \mathcal{S}_N$ with the output function $\boldsymbol{h_j}$, $j \in \{1, \dots, M\}$ we can instantiate Definition 1.15 to an easier to handle criterion, cf. [Fia83, MG95, SJ11]. For notational convenience we again assume $M = N$ here.

Lemma 2.1 *Let $(\mathcal{T}, \mathcal{X}, \mathcal{U}, \boldsymbol{f})$ be a system that implicitly defines $\boldsymbol{F_k}$, $k \in \mathcal{S}_N$, and let $\boldsymbol{h_j}$, $j \in \mathcal{S}_0$ be the corresponding output function. Further let $\boldsymbol{F_k^u} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ be given by $\boldsymbol{F_k^u}(\boldsymbol{x}, \boldsymbol{p}) := \boldsymbol{F_k}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$. Now, let us consider the mapping $\boldsymbol{\Phi} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^{N n_u} \to \mathbb{R}^{N n_y}$ given by*

$$\boldsymbol{\Phi}(\boldsymbol{x_0}, \boldsymbol{p}, \boldsymbol{u}) := \begin{bmatrix} \boldsymbol{h_1} \circ \boldsymbol{F_0^{u_0}} \\ \vdots \\ \boldsymbol{h_N} \circ \boldsymbol{F_{N-1}^{u_{N-1}}} \circ \cdots \circ \boldsymbol{F_0^{u_0}} \end{bmatrix} (\boldsymbol{x_0}, \boldsymbol{p}).$$

If there is a $\boldsymbol{u} \in \mathcal{U}$ such that for all $\boldsymbol{x} \in \mathcal{X}$ the Jacobian $\frac{\partial \boldsymbol{\Phi}}{\partial(\boldsymbol{x}, \boldsymbol{p})}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$ has full column rank (equal to $n_x + n_p$), then the system is N-observable. In particular, all distinct pairs of events $(\boldsymbol{x}(t_f) = \boldsymbol{x_f}, \boldsymbol{p_1})$ and $(\boldsymbol{x}(t_f) = \boldsymbol{x_2}, \boldsymbol{p_2})$ are distinguishable.

Proof See, e.g., [SJ11]. □

We can therefore use Lemma 2.1 to characterize when the current state and parameter vector is recoverable by an MHE scheme using a limited number of measurements.

In our basic setting, we assumed that the implemented control actions $\boldsymbol{u}$ are known exactly. We note that some practical applications may require to also treat $\boldsymbol{u}$ as additional degrees of freedom (that are fitted to corresponding "measurements") in the optimization problem (MHE) to account for actuator uncertainty.

2.2.2 Arrival cost updates

The conceptual idea behind the arrival cost $\ell_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is to summarize (*filter*) all *prior information* that cannot be considered explicitly

in (MHE) for computational reasons. In principle, *zero prior weighting*, i.e., completely discounting all a priori information, e.g., by choosing $\boldsymbol{P} \equiv \mathbf{0}$ in Equation (2.6) as proposed in [MM95] would be possible, and existence and stability proofs can be established also for this case under some additional, rather restrictive assumptions, cf. [MM95, RM09]. However, doing so may render the horizon lengths of the MHE problem required to recover the true system state overly long in practice.

A more common choice is to use Kalman-Filter-based updates on $\boldsymbol{P}$ and/or $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{p}}$, which can be motivated by dynamic programming arguments [MRL93, Mus95, RL95, RLR96, RRL01]. If the underlying system dynamics are nonlinear, a natural choice for updating the arrival cost are variants of the so-called *extended Kalman filter* (EKF), which essentially is a Kalman filter that utilizes a system linearization at the most recent estimate for state and covariance prediction (see, e.g., [RM09] for details). A smoothed variant of the EKF, which chooses the most recent MHE estimate for $\boldsymbol{x_0}$ and $\boldsymbol{p}$ as linearization point, appears to have particularly favorable properties, cf. [RL95, RLR96, Fin97, RRL01, KKWB08, KDK+11a]. Further details and update formulae tailored to Multiple-Shooting-style methods can be found in [Kra07, KDK+11a]. It should be stressed at this point that, in contrast to its linear variant, the EKF does, in general, not obtain the full information estimates if the system dynamics are nonlinear — even in the absence of path constraints. MHE with EKF-based updates of its arrival cost therefore tends to be seen as superior to a single EKF as it considers more measurements explicitly and chooses a possibly more accurate linearization point before the information is filtered into the arrival cost terms. In fact, by choosing an estimation history of only $M = 1$, a MHE scheme with suitably chosen weighting matrices and Gauss-Newton Hessian approximation is identical to the EKF, cf., e.g., [Rob96, KDK+11a].

2.2.3 Choice of weighting matrices

Even though MHE is a deterministic method for minimizing measurement residuals, the probabilistic insight can be very helpful for choosing suitable penalty terms. Here, we particularly focus on the choice of weighting matrices $\boldsymbol{V_j} \in \mathbb{R}^{n_y \times n_y}$, $j \in \mathcal{S}_0$ and $\boldsymbol{P} \in \mathbb{R}^{(n_x+n_p)\times(n_x+n_p)}$ for least-squares fitting objective functions. Some further aspects in the discussion on the implications of a probabilistic versus a deterministic perspective on MHE can be found, e.g., in [RLR96, RM09, KDK+11b].

Under the assumption that the sensor noise perturbing the observation of the system outputs follows independent normal distributions, i.e., $\boldsymbol{y_j} := \boldsymbol{h_j}(\boldsymbol{x}^{\#}(t_j), \boldsymbol{u}(t_j), \boldsymbol{p}^{\#}) + \boldsymbol{\eta_j}$ for all $j \in \mathcal{S}_0$, where $\boldsymbol{x}^{\#}(t_j)$ and $\boldsymbol{p}^{\#}$ denote the

true system state and parameter vector at time t_j, and $\boldsymbol{\eta_j} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{j}}^{\boldsymbol{y}})$ with $\boldsymbol{\Sigma}_{\boldsymbol{j}}^{\boldsymbol{y}} \succ 0$, it can be shown that choosing $\boldsymbol{V_j} := (\boldsymbol{\Sigma}_{\boldsymbol{j}}^{\boldsymbol{y}})^{-1}$ ensures the equivalence of the solution of MHE and the maximum-likelihood estimator (MLE) in the unconstrained case, cf. [RLR96, KDK+11b]. This result holds either for zero prior weighting, i.e., if $\boldsymbol{P} \equiv \mathbf{0}$, or under the assumptions that the sliding window's initial value $\boldsymbol{x_0}$ and $\boldsymbol{p}$ are independently normally distributed with $\boldsymbol{x_0} \sim \mathcal{N}(\bar{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\mathbf{0}}^{\boldsymbol{x}})$ and $\boldsymbol{p} \sim \mathcal{N}(\bar{\boldsymbol{p}}, \boldsymbol{\Sigma}^{\boldsymbol{p}})$ if $\boldsymbol{P} = \operatorname{block\,diag}((\boldsymbol{\Sigma}_{\mathbf{0}}^{\boldsymbol{x}})^{-1}, (\boldsymbol{\Sigma}^{\boldsymbol{p}})^{-1})$. If the presence of state noise is assumed, additionally $\boldsymbol{\zeta_j} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{j}}^{\boldsymbol{x}})$ and $\boldsymbol{W_j} = (\boldsymbol{\Sigma}_{\boldsymbol{k}}^{\boldsymbol{x}})^{-1}$, $j \in \mathcal{S}_0$ is required for the MLE property to hold. A discussion on conditions under which the MLE property is preserved if stage constraints (in particular on states and parameters) become active in the MHE solution can be found in [Rob96].

2.3 Real-time Dynamic Optimization Algorithms

The online re-computation of solutions of the dynamic optimization problems (MPC) and (MHE) at each sampling time permits to react to consequences from unmodeled influences in an attempt to operate the dynamic system at hand in an optimal fashion. For this to be successful, however, the challenge of actuation delays introduced by the online feedback computations needs to be overcome.

Applying one of the nonlinear programming algorithms mentioned in Section 1.4 will typically require a significant share of each sampling period (or even exceed it). This has several implications. First of all, the computational delay introduced by solving (MHE) with a standard method for nonlinear programming, $\Delta t^{\mathrm{MHE-NLP}}$, will cause the estimate of the "current" state $\hat{\boldsymbol{x}}(T_i)$ to only be available at time $T_i + \Delta t^{\mathrm{MHE-NLP}}$. A control scheme for computing a feedback law at sampling time T_i can therefore either choose to use a prediction $\tilde{\boldsymbol{x}}(T_i)$, which in particular does not yet employ the most recent observation $\boldsymbol{y}(T_i)$ (we can however expect a control action computed based on this prediction to be almost identical[11] with the control action computed at sample time T_{i-1} for time T_i), or to wait for a more accurate estimate $\hat{\boldsymbol{x}}(T_i)$, which however only becomes available with a delay of $\Delta t^{\mathrm{MHE-NLP}}$. Additionally, the solution of (MPC) by a standard NLP algorithm will introduce a delay of $\Delta t^{\mathrm{MPC-NLP}}$ itself. This means that the updated control action $\boldsymbol{u_0^*}$ will only become available at time $T_i + \Delta t^{\mathrm{MHE-NLP}} + \Delta t^{\mathrm{MPC-NLP}}$. In the meantime, however, the system state of the plant may have changed, and internal as well as external unmodeled influences may have caused additional deviations of the actual system state from

[11] If the prediction horizon is chosen sufficiently long, we can expect the effect from the shifted horizon to be rather small.

its predicted value. On top of this, the control performance may be suboptimal if it bases on the assumption that the optimized control law is implemented immediately, i.e., if the optimal control problem did not account for its own *computational dead-time* (which often can only be estimated in an extremely conservative fashion if iterative algorithms are used for the NLP solution, and therefore lead itself to a significant delay in the control law).

A variety of algorithmic ideas has been proposed to overcome these challenges. To speed up computations beyond the regular algorithmic and computational advancement, essentially one can either aim to precompute certain results, or to compute approximate solutions. *Explicit MPC* largely falls in the former category. The main idea of explicit MPC is to precompute (approximations to) the solution manifold[12] of (MPC) for variable initial conditions $\hat{\boldsymbol{x}}_{\mathbf{0}}$ offline and to to store it for online retrieval based on the observed/estimated system state. We refer, e.g., to [BMDP02, BBM03, Bor03, PRW07, BBM14] and the references therein for an introduction to explicit MPC. Despite the fact that the algorithmic ideas are typically presented from the MPC viewpoint, we note that the main algorithmic concepts are, in principle, also applicable to other dynamic optimization problems, like MHE.

Particularly for linear MPC the explicit approach can be very efficient in terms of online computation time. The main disadvantage, however, is that offline computation time and storage requirements typically grow exponentially in the dimension of the initial state (or, more general, in the number of parameters the solution depends upon), which limits the applicability of explicit MPC to rather small-scale systems for which all data is known offline (i.e., no online changing references, etc.).

Other popular methods that aim specifically at a particularly short feedback delay are the so-called *Advanced Step controller*, see [ZB09], and the so-called *Real-Time Iteration (RTI) scheme*, see [DBS+02]. Both exploit the similarity between subsequent dynamic optimization problems to be solved in the online context for a prediction of the following initial condition, at the aim of preparing significant parts of the NLP solution even before the initial condition is known.

This thesis is particularly focused on variants of the RTI scheme, which has proven to be computationally very efficient, in particular in combination with automatic code generation, while retaining sufficient accuracy (see, e.g., [HFD11b, FHGD11b, VLH+12, FKV+12, FGZ+13]). A detailed comparison between the Advanced Step controller and the RTI scheme, as well as a broader overview of other initial value embeddings and the resulting first-order predictors can be found in the survey [DFH09].

[12] To be precise, only the first control action $\boldsymbol{u}_{\mathbf{0}}^{\star}(\hat{\boldsymbol{x}}_{\mathbf{0}})$ of each solution needs to be computed and stored.

2.3.1 The Real-Time Iteration scheme

The *Real-Time Iteration* (RTI) scheme is an algorithmic concept that is specifically designed to reduce the computation time delay between observation of a measurement $\boldsymbol{y}(T_i)$ and implementation of the re-optimized control law $\boldsymbol{u}_{\mathbf{0}}^*$. It goes back to [DBS+02, Die02] and was originally introduced as an adaptation of the SQP procedure within Bock's direct multiple shooting method. Our presentation is loosely based on [Die02, Kir11].

The essential idea of the RTI scheme is that, instead of doing SQP iterations until convergence, only one QP solution is performed at each sampling time (each in MHE and in MPC) based on a linearization computed before observation of the measurement $\boldsymbol{y}(T_i)$. This way, the sampling period can be reduced to roughly the computational cost of two full SQP iterations[13] (one each for MHE and MPC), consisting each of a linearization procedure and a subsequent QP solution. Furthermore, the *feedback delay* reduces to the time required for two QP solutions (which may, depending on the problem, be significantly shorter than the computation time required for linearization), since the full QP, except for the initial value/measurement embedding, can already be set up beforehand. Convergence guarantees to an optimal solution can be retained in the real-time notion of contractivity, cf. Section 2.3.3 and [Die02, DBS+02, DFA07].

Figure 2.3 summarizes the RTI scheme. Therein, the *shift* step prepares a good initial guess for linearization of the nonlinear components at the next sampling time. We will give details in Section 2.3.2. It is important to observe that both, shifting and linearization for the DOP solution at sampling time T_{i+1} are performed at sampling time T_i only *after* the updated control strategy/state estimate has been sent to the plant/the controller, but finish *before* the new state estimate/observation is received from the estimator/the plant. These two steps together form the so-called *preparation phase* of the RTI scheme. The second, so-called *feedback phase*, which essentially is the only source of feedback delay, consists solely of the initial value/measurement embedding and the QP solution.

In detail, the RTI scheme computes an (approximate) solution of the discretized dynamic optimization problem (i.e., the MPC/MHE NLP) based on a primal-dual initial guess $(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ as

$$(\boldsymbol{w}^+, \boldsymbol{\lambda}^+, \boldsymbol{\mu}^+) := (\boldsymbol{w} + \boldsymbol{z}^*, \boldsymbol{\lambda}_{\mathbf{QP}}^*, \boldsymbol{\mu}_{\mathbf{QP}}^*),$$

where $(\boldsymbol{z}^*, \boldsymbol{\lambda}_{\mathbf{QP}}^*, \boldsymbol{\mu}_{\mathbf{QP}}^*)$ is the primal-dual QP subproblem solution, cf. Section 1.4.3. Note that typically no globalization strategy is employed in the RTI scheme to determine the step size of the update to the primal-dual NLP variables.

[13] on a sequential computational architecture

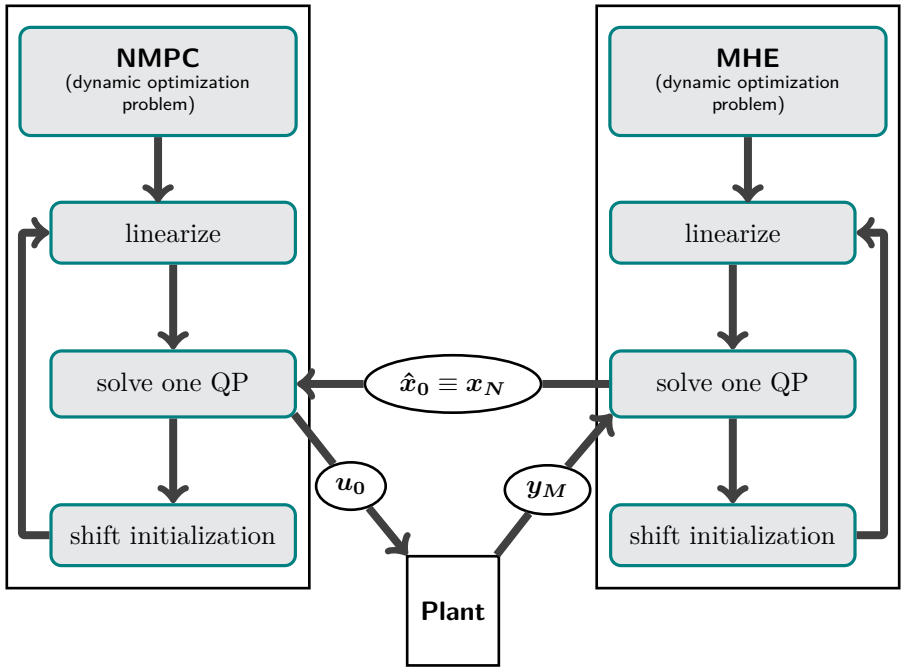


Figure 2.3: Sketch of the real-time iteration scheme.

Instead, full steps are taken, i.e., $\alpha^k := 1$ is chosen constantly in (1.13). The motivation behind this approach is two-fold. One the one hand, the RTI scheme is designed to provide very fast feedback under the assumption that a very good initialization (i.e., close to a regular KKT point) of the NLP variables is available, e.g., by a shift of the previous optimal solution (assuming a not too inaccurate system model). One in particular assumes that the initialization is good enough to still be in the region of attraction of a full-step SQP method, cf. Section 1.4.4. On the other hand, performing a full SQP-type step is the only possibility to ensure that the initial value embedding constraint (MPC4) is satisfied already after only one SQP-type iteration[14].

For both, MPC and MHE, we have the observation that the instance of QP (1.12) to be solved in each iteration of an SQP method applied to an MPC/MHE problem structurally resembles (LMPC)[15]. Therefore, the RTI scheme can be

[14]Note that (MPC4) is identical to its linearized variant (LMPC5).

[15]Here, the only significant difference between (LMPC) and its MHE counterpart is the initial value embedding (LMPC5), which is replaced by an embedding of the terminal measurement $\boldsymbol{y_M}$ in MHE. For clarity of the presentation we therefore restrict ourselves to the MPC-centered viewpoint.

seen as a linear MPC controller, whose data is updated in each iteration based on a re-linearization of the nonlinear dynamics and constraints, and a new (possibly inexact) quadratic fit to the NLP Lagrangian. To reflect this in notation, we will stick to the convention in the following, that NLP stage variables are denoted by $\boldsymbol{w_k} := (\boldsymbol{s_k}, \boldsymbol{q_k}),\ k \in \mathcal{S}$ and that QP/LMPC stage variables (particularly in Part II) are denoted by $\boldsymbol{z_k} := (\boldsymbol{x_k}, \boldsymbol{u_k}),\ k \in \mathcal{S}$. W.l.o.g., we assume parameters $\boldsymbol{p}$ hidden in the state vector, cf. Section 1.1.1, and, for notational convenience, redundant optimization vectors for control parameterization on the final stage, $\boldsymbol{q_N} \in \mathbb{R}^0$ and $\boldsymbol{u_N} \in \mathbb{R}^0$, are introduced.

2.3.2 Initialization in the Real-Time Iteration scheme

Since the RTI feedback at each sampling time is computed based on only a single QP approximation, a sensible choice of the linearization point is crucial for the RTI scheme's performance. Usually, the RTI scheme prepares the initial guess for the NLP variables at a subsequent sampling time by performing a time shift that corresponds to one sampling period. This means, if the NLP iterates at sampling time T_i (after the feedback phase) were

$$\boldsymbol{w}|_{T_i} = (\boldsymbol{s_0}, \boldsymbol{q_0}, \boldsymbol{s_1}, \ldots, \boldsymbol{q_{N-1}}, \boldsymbol{s_N}),$$

the linearization for the LMPC problem to be solved at sampling time T_{i+1} is, assuming that the sampling grid is identical with the discretization grid, performed at the initial guess

$$\boldsymbol{w}|_{T_{i+1}} := (\boldsymbol{s_1}, \boldsymbol{q_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{q_{N-1}}, \boldsymbol{s_N}, \boldsymbol{q}^{\mathbf{new}}_{\boldsymbol{N-1}}, \boldsymbol{s}^{\mathbf{new}}_{\boldsymbol{N}}).$$

Several possibilities exist for choosing the linearization points $\boldsymbol{q}^{\mathbf{new}}_{\boldsymbol{N-1}}$ and $\boldsymbol{s}^{\mathbf{new}}_{\boldsymbol{N}}$ on the last shooting interval, cf. [Die02, Kir11]:

Extrapolation The control law $\boldsymbol{q_{N-1}}$ is kept on the last control interval, and the new terminal value $\boldsymbol{s}^{\mathbf{new}}_{\boldsymbol{N}}$ is obtained by forward simulation, i.e.,

$$\boldsymbol{q}^{\mathbf{new}}_{\boldsymbol{N-1}} := \boldsymbol{q_{N-1}}, \qquad \text{and} \qquad \boldsymbol{s}^{\mathbf{new}}_{\boldsymbol{N}} := \boldsymbol{F_N}(\boldsymbol{s_N}, \boldsymbol{q}^{\mathbf{new}}_{\boldsymbol{N-1}}),$$

where $\boldsymbol{F_N}$ denotes the state transition mapping from sampling time T_{i+N} to T_{i+N+1}. This rule leads to a continuous trajectory initialization on the last stage, but path, and particularly terminal constraints (in case of MPC) may be violated by this initialization.

Nominal extrapolation Similarly to the extrapolation initialization, $\boldsymbol{s}^{\mathbf{new}}_{\boldsymbol{N}}$ is obtained by forward simulation; however, instead of repeating the

terminal control action from the previous sampling time, a (possibly time dependent) nominal control law $\bar{\boldsymbol{q}}(T_{(i+1)+(N-1)}, \boldsymbol{s_N})$ is applied. We have

$$\boldsymbol{q}_{\boldsymbol{N-1}}^{\mathbf{new}} := \bar{\boldsymbol{q}}(T_{i+N}, \boldsymbol{s_N}), \qquad \text{and} \qquad \boldsymbol{s}_{\boldsymbol{N}}^{\mathbf{new}} := \boldsymbol{F_N}(\boldsymbol{s_N}, \boldsymbol{q}_{\boldsymbol{N-1}}^{\mathbf{new}}).$$

Again, we obtain a continuous trajectory initialization on the last stage, but may violate path constraints.

State feedback extrapolation Instead of applying a fixed nominal control law, we may adapt this law depending on the terminal value $\boldsymbol{s_N}$, e.g., by using a precomputed gain matrix (or one that is based on the most recent QP subproblem solution) $\boldsymbol{K}$. We then have

$$\boldsymbol{q}_{\boldsymbol{N-1}}^{\mathbf{new}} := \boldsymbol{K}\boldsymbol{s_N}, \qquad \text{and} \qquad \boldsymbol{s}_{\boldsymbol{N}}^{\mathbf{new}} := \boldsymbol{F_N}(\boldsymbol{s_N}, \boldsymbol{q}_{\boldsymbol{N-1}}^{\mathbf{new}}).$$

If MPC and MHE are deeply interleaved, we may also use the control signal computed in MPC and sent to the plant for extrapolation in MHE (or even use the MPC prediction of the resulting state for $\boldsymbol{s}_{\boldsymbol{N}}^{\mathbf{new}}$). Once more, a continuous trajectory initialization on the last stage and violated path constraints are possible consequences from such choices.

Duplication Both, terminal control law and terminal state are kept from the previous sampling time, i.e., we have

$$\boldsymbol{q}_{\boldsymbol{N-1}}^{\mathbf{new}} := \boldsymbol{q_{N-1}}, \qquad \text{and} \qquad \boldsymbol{s}_{\boldsymbol{N}}^{\mathbf{new}} := \boldsymbol{s_N}.$$

This way, time constant path and terminal constraints are fulfilled, but continuity of the initialization will, in general, be violated.

Independently of the primal initialization strategy for the terminal stage, the dual variables (if required) are typically shifted alongside the primal variables, and simply kept on the terminal stage. This means, if

$$\boldsymbol{\lambda}|_{T_i} = (\boldsymbol{\lambda_0}, \ldots, \boldsymbol{\lambda_{N-1}})$$

denotes the dual variables of the coupling constraints (MPC2), and

$$\boldsymbol{\mu}|_{T_i} = (\boldsymbol{\mu_0}, \ldots, \boldsymbol{\mu_N})$$

denotes the dual variables of the stage constraints (MPC3), we use

$$\boldsymbol{\lambda}|_{T_{i+1}} := (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_{N-1}}, \boldsymbol{\lambda_{N-1}})$$

and

$$\boldsymbol{\mu}|_{T_{i+1}} := (\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_N}, \boldsymbol{\mu_N}).$$

Depending on the problem at hand, the solution of the dynamic optimization problem solved at time T_{i+1} may also resemble the solution of the dynamic optimization problem solved at time T_i, rather than its shifted version[16]. Then, a simple warm-start may be performed instead of the shift, i.e., the linearization for the setup of the linear MPC problem to be solved at time T_{i+1} is performed directly based on the primal-dual solution of the dynamic optimization problem at time T_i.

2.3.3 Contractivity of the Real-Time Iteration scheme

The RTI scheme, in general, does not solve nonlinear dynamic optimization problems exactly. We state, however, central important results (due to [Die02, DFA+05, DFA07]) that link the solutions computed by the RTI scheme to the exact solutions of the series of dynamic optimization problems in the following.

The analysis is based on local convergence theory for Newton's method, in particular Theorem 1.51. The general assumption is that the dynamic optimization problem to be solved is initialized sufficiently close to a KKT point such that the combinatorics of the active set selection can be neglected[17] for the analysis, i.e., we only have to consider an equality constrained NLP

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} \quad f(\boldsymbol{w}) \qquad \text{s.t.} \qquad \boldsymbol{g}(\boldsymbol{w}) = 0. \tag{2.8}$$

The first-order optimality conditions then read, cf. Theorem 1.35,

$$\boldsymbol{r}(\boldsymbol{w},\boldsymbol{\lambda}) := \begin{bmatrix} \boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{\nabla} \boldsymbol{g}(\boldsymbol{w})^\top \boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{w}) \end{bmatrix} = 0 \tag{2.9}$$

and an approximate solution is computed by the RTI scheme — based on an initial guess $(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}})$ — by performing one Newton-type iteration

$$\begin{bmatrix} \boldsymbol{w^{(i+1)}} \\ \boldsymbol{\lambda^{(i+1)}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w^{(i)}} \\ \boldsymbol{\lambda^{(i)}} \end{bmatrix} - \boldsymbol{J}\big(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}}\big)^{-1} \boldsymbol{r}\big(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}}\big), \tag{2.10}$$

where $\boldsymbol{J}(\boldsymbol{w},\boldsymbol{\lambda}) \approx \frac{\partial \boldsymbol{r}}{\partial(\boldsymbol{w},\boldsymbol{\lambda})}(\boldsymbol{w},\boldsymbol{\lambda})$ is an invertible approximation of the derivative of the KKT residual (also known as *KKT matrix*), that depends on the chosen Hessian approximation, cf. Section 1.4.4.

[16] This may, e.g., be the case in MPC with rather short prediction horizons, cf. [Die02], or when sampling periods are significantly shorter than the control discretization time step.

[17] Nevertheless it can be shown that the RTI scheme employs *generalized* tangential predictors that incorporate the effects from active-set changes at least in an approximate fashion, cf. [Die02, DFH09].

The first statements here address feasibility and suboptimality of the solutions computed by the RTI scheme in comparison with an exact solution algorithm. Since the problems to be solved in MPC and MHE change from iteration to iteration, we quantify suboptimality in a per-interval notion in the following. The technique to do this is to consider real-time iterations on a shrinking horizon. Based on the discretized DOP (2.8) we derive a series of reduced horizon problems $P^{(i)}$ by incrementally fixing the controls on the first $i-1$ stages. Formally, we therefore have[18]

$$\begin{aligned} P^{(i)}: \qquad & \min_{\boldsymbol{w}\in\mathbb{R}^n} \quad f(\boldsymbol{w}) \\ & \text{s.t.} \quad \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0} \\ & \qquad\quad \boldsymbol{q_j} = \bar{\boldsymbol{q}}_{\boldsymbol{j}}, \qquad\qquad \forall\ j \in \{0,\dots,i-1\} \end{aligned}$$

where $\bar{\boldsymbol{q}}_{\boldsymbol{j}}$ are some fixed (in terms of the optimization) control parameterizations, which are defined successively with $\bar{\boldsymbol{q}}_{\boldsymbol{i}}$ being the solution computed for $\boldsymbol{q_i}$ by the RTI scheme applied to problem $P^{(i)}$, cf. [Die02]. When $i > N$ exceeds the horizon length, we define $P^{(i)} := P^{(N)}$.

We use the shorthands $\boldsymbol{p^{(i)}} := \begin{bmatrix} \boldsymbol{w^{(i)}} \\ \boldsymbol{\lambda^{(i)}} \end{bmatrix}$ to denote the primal-dual RTI solutions for problem $P^{(i)}$. The RTI updates $\boldsymbol{\Delta p^{(i)}}$ are, in analogy to (2.9) and (2.10), defined by

$$\boldsymbol{\Delta p^{(i)}} := \boldsymbol{J^{(i)}}\left(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}}\right)^{-1} \boldsymbol{r^{(i)}}\left(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}}\right),$$

where

$$\boldsymbol{r^{(i)}}(\boldsymbol{w},\boldsymbol{\lambda}) := \begin{bmatrix} \boldsymbol{\Pi^{(i)}}\left(\boldsymbol{\nabla f}(\boldsymbol{w}) + \boldsymbol{\nabla g}(\boldsymbol{w})^\top \boldsymbol{\lambda}\right) \\ \boldsymbol{g}(\boldsymbol{w}) \\ \boldsymbol{q_0} - \bar{\boldsymbol{q}}_{\boldsymbol{0}} \\ \vdots \\ \boldsymbol{q_{i-1}} - \bar{\boldsymbol{q}}_{\boldsymbol{i-1}} \end{bmatrix} = 0$$

and

$$\boldsymbol{J^{(i)}}(\boldsymbol{w},\boldsymbol{\lambda}) \approx \frac{\partial \boldsymbol{r^{(i)}}}{\partial(\boldsymbol{w},\boldsymbol{\lambda})}(\boldsymbol{w},\boldsymbol{\lambda})$$

are invertible approximations. Here, $\boldsymbol{\Pi^{(i)}} \in \mathbb{R}^{(n-m_i)\times n}$, where m_i is the dimension of $(\boldsymbol{q_0},\dots,\boldsymbol{q_{i-1}})$, are suitable projection matrices to eliminate the rows of the Lagrange gradient corresponding to fixed controls. We refer to [Die02] for further details. Note that it holds in particular that $\boldsymbol{r^{(i)}}(\boldsymbol{w},\boldsymbol{\lambda}) \in \mathbb{R}^{n+l}$,

[18] Recall that $\boldsymbol{w}$ is the vector of stacked state and control variables, i.e., it contains $\boldsymbol{q_j}$, $j \in \{0,\dots,i-1\}$.

where l is the number of rows of $\boldsymbol{g}$, is for all $i = 1, 2, \ldots$ of fixed dimension, and we have $\boldsymbol{r^{(0)}}(\boldsymbol{w}, \boldsymbol{\lambda}) = \boldsymbol{r}(\boldsymbol{w}, \boldsymbol{\lambda})$ and accordingly $\boldsymbol{J^{(0)}}(\boldsymbol{w}, \boldsymbol{\lambda}) = \boldsymbol{J}(\boldsymbol{w}, \boldsymbol{\lambda})$.

We have now set the stage to address (fast) asymptotic feasibility of the RTI iterates[19].

Theorem 2.2 *Let $\boldsymbol{p}^{(0)} \in \mathcal{D} \subseteq \mathbb{R}^n$, let $\boldsymbol{J}(\boldsymbol{p})$ be invertible with a bounded inverse on $\mathcal{D}$, i.e., $\|\boldsymbol{J}(\boldsymbol{p})^{-1}\| \leq \beta$ for all $\boldsymbol{p} \in \mathcal{D}$. Assume further, that $\boldsymbol{J}(\boldsymbol{p})$ is Lipschitz-continuous on $\mathcal{D}$ with Lipschitz constant ω/β , and that the incompatibility of the KKT matrix approximation is sufficiently bounded on $\mathcal{D}$, i.e., $\|\boldsymbol{J}(\boldsymbol{p}) - \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}\| \leq \kappa/\beta$ with $\kappa < 1$. If the initial guess $\boldsymbol{p}^{(0)}$ is sufficiently good, such that*

$$\delta^{(0)} := \kappa + \frac{\omega}{2} \| - \boldsymbol{J}(\boldsymbol{p^{(0)}})^{-1} \boldsymbol{r}(\boldsymbol{p^{(0)}})\| < 1, \tag{2.11}$$

and

$$\mathcal{D}^{(0)} := \left\{ \boldsymbol{p} \in \mathbb{R}^n \mid \|\boldsymbol{p} - \boldsymbol{p^{(0)}}\| \leq \frac{\| - \boldsymbol{J}(\boldsymbol{p^{(0)}})^{-1} \boldsymbol{r}(\boldsymbol{p^{(0)}})\|}{1 - \delta^{(0)}} \right\} \subseteq \mathcal{D},$$

then the sequence of RTI iterates $\{\boldsymbol{p^{(i)}}\}_{0,1,\ldots}$ converges towards a feasible point $\boldsymbol{p^{*_1}} \in \mathcal{D}^{(0)}$. In particular, the contraction property of Theorem 1.51 holds, i.e.,

$$\|\boldsymbol{\Delta p^{(i+1)}}\| \leq \left(\kappa + \frac{\omega}{2} \|\boldsymbol{\Delta p^{(i)}}\|\right) \|\boldsymbol{\Delta p^{(i)}}\|.$$

Proof See [Die02]. □

Theorem 2.2 indicates in particular that we can expect the RTI scheme to produce iterates that are nonlinearly feasible both with respect to the stage constraints (MPC3)/(MHE3) and with respect to the continuity constraints (MPC2)/(MHE2) on stages that are sufficiently far from the end of the prediction horizon (back in the receding horizon setting), i.e., stages whose stage variables have already been improved by several Newton-type iterations.

Next we give a quantification of suboptimality of the RTI scheme with respect to the fully converged solution, again in the setting of shrinking horizons. To this end, we introduce $\boldsymbol{p_i^{*\infty}}$, the fully converged solution of $P^{(i)}$ computed by applying sufficiently many Newton-type steps.

Theorem 2.3 *Let the conditions of Theorem 2.2 be satisfied. Then, the distance between the limit point of the RTI scheme, $\boldsymbol{p^{*_1}}$, and the rigorous solution $\boldsymbol{p_i^{*\infty}}$ of the i-th reduced horizon dynamic optimization problem $P^{(i)}$ can be bounded by*

$$\|\boldsymbol{p^{*_1}} - \boldsymbol{p_i^{*\infty}}\| \leq \frac{2(\delta_0)^{i+1} \|\boldsymbol{\Delta p^0}\|}{1 - \delta_0},$$

[19] Note that in the following we switch from $(\boldsymbol{w}, \boldsymbol{\lambda})$ to the notationally more convenient argument representation $\boldsymbol{p}$ for $\boldsymbol{J}$ and $\boldsymbol{r}$.

where δ_0 is defined in Equation (2.11).

Proof See [Die02]. □

Theorem 2.3 gives a handle on how fast the solutions computed by the RTI scheme approach the fully converged solutions. While this observation is trivial for $i \geq N$, we still can conclude from this result that the RTI limit solution $\boldsymbol{p}^{*_1}$ is close to the rigorous solutions of problems with a reduced horizon; in fact the distance between the two solutions shrinks with the reduction of the horizon length at rates similar to the convergence rates of Newton-type methods, cf. Theorem 1.51.

If we assume that the RTI scheme is initialized in a KKT point (that is, e.g., precomputed offline), we can quantify how fast a perturbation of the initial state $\hat{\boldsymbol{x}}_{\mathbf{0}}$ (with respect to the assumed initial state) is rejected in comparison with a fully converged nonlinear MPC controller. Recall to this end, that $P^{(0)} =: P(\hat{\boldsymbol{x}}_{\mathbf{0}})$ implicitly depends on $\hat{\boldsymbol{x}}_{\mathbf{0}}$ via the initial value embedding, which is part of $\boldsymbol{g}$. Note that a similar disturbance rejection result can be established for the MHE context.

Theorem 2.4 *Let $\bar{\boldsymbol{p}}^*$ be an initialization of $P(\hat{\boldsymbol{x}}_{\mathbf{0}})$ that is a KKT point. Let us assume that the true initial value $\hat{\boldsymbol{x}}_{\mathbf{0}}$ deviates (slightly) from this initialization, i.e., we have a disturbance of size $\epsilon := \|\hat{\boldsymbol{x}}_{\mathbf{0}} - \bar{\boldsymbol{x}}_{\mathbf{0}}^*\|$ of the parametric dependency of $P(\hat{\boldsymbol{x}}_{\mathbf{0}})$, where $\bar{\boldsymbol{x}}_{\mathbf{0}}^*$ are the first n_x variables of $\bar{\boldsymbol{p}}^*$. Then, the distance between the limit point of the RTI scheme applied to the disturbed problem, $\boldsymbol{p}^{*_1}$, and the rigorous solution $\boldsymbol{p}_{\mathbf{0}}^{*_\infty}$ of the disturbed dynamic optimization problem is of first order in ϵ,*

$$\|\boldsymbol{p}^{*_1} - \boldsymbol{p}^{*_\infty}\| \leq 2\frac{\kappa + \frac{\omega}{2}\beta\epsilon}{1 - (\kappa + \frac{\omega}{2}\beta\epsilon)}\,\beta\epsilon$$

for general Newton-type methods, and even of second order for an exact Newton method:

$$\|\boldsymbol{p}^{*_1} - \boldsymbol{p}^{*_\infty}\| \leq \frac{\omega}{1 - \frac{\omega}{2}\beta\epsilon}\,\beta^2\epsilon^2.$$

The loss of optimality is of second order in ϵ for a general Newton method:

$$f(\boldsymbol{p}^{*_1}) - f(\boldsymbol{p}^{*_\infty}) \leq 2\,B_r\left(\frac{\kappa + \frac{\omega}{2}\beta\epsilon}{1 - (\kappa + \frac{\omega}{2}\beta\epsilon)}\,\beta\right)^2\epsilon^2,$$

and fourth order for an exact Newton method:

$$f(\boldsymbol{p}^{*_1}) - f(\boldsymbol{p}^{*_\infty}) \leq \frac{B_r\omega^2}{2\,(1 - \frac{\omega}{2}\beta\epsilon)^2}\,\beta^4\epsilon^4.$$

Here, β, ω, and κ are defined as in Theorem 2.2, and B_r is a bound on the norm of the exact derivative of the KKT residual function which satisfies

$$\left\| \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}(\boldsymbol{p}) \right\| \leq B_r \qquad \forall\ \boldsymbol{p} \in \mathcal{D}^{(0)}.$$

Proof See [Die02]. □

In the remainder of this section, we briefly address the question of MPC stability under application of the RTI scheme. Since the RTI scheme does not solve each dynamic optimization problem exactly, the standard proofs of nominal stability for nonlinear MPC are not applicable here. In [DFA+05] and [DFA07] stability proofs for the RTI scheme are given, which rely on a series of rather technical assumptions. We give a characteristic summary of the central results here, and refer to the original papers for the technical details.

The general setting assumed for the stability considerations is a dynamic system

$$\boldsymbol{x_{i+1}} = \boldsymbol{F}(\boldsymbol{x_i}, \boldsymbol{u}(\boldsymbol{x_i})) \tag{2.12}$$

with shifted coordinates such that (w.l.o.g.) the origin is a steady state, i.e., $\boldsymbol{F}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}$.

The RTI scheme is applied to the following nonlinear MPC problems without stage constraints:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{k=0}^{N} \ell_k(\boldsymbol{x_k}, \boldsymbol{u_k}) \tag{2.13a}$$

$$\text{s.t.} \qquad \boldsymbol{x_{k+1}} = \boldsymbol{F}(\boldsymbol{x_k}, \boldsymbol{u_k}) \qquad \forall\ k \in \mathcal{S}_N \tag{2.13b}$$

$$\boldsymbol{x_0} = \hat{\boldsymbol{x}}_{\boldsymbol{0}} \tag{2.13c}$$

where the cost function $V(\hat{\boldsymbol{x}}_{\boldsymbol{0}}) := \sum_{k=0}^{N} \ell_k(\boldsymbol{x}_{\boldsymbol{k}}^{*}(\hat{\boldsymbol{x}}_{\boldsymbol{0}}), \boldsymbol{u}_{\boldsymbol{k}}^{*}(\hat{\boldsymbol{x}}_{\boldsymbol{0}}))$ is assumed to feature a unique optimal solution (which is also assumed to be the only KKT point) for all initial values in a neighborhood of the origin $\boldsymbol{0} \in \mathbb{R}^{n_x}$. Furthermore, $V(\hat{\boldsymbol{x}}_{\boldsymbol{0}})$ is assumed to be bounded from below and from above by quadratic functions in the initial value $m\|\boldsymbol{x}\|^2 \leq V(\hat{\boldsymbol{x}}_{\boldsymbol{0}}) \leq M\|\boldsymbol{x}\|^2$.

Again, regarding regularity of the Newton-type iterations, an incompatibility bound of the used Hessian approximation as in Theorem 2.2 is assumed, as well as a nonlinearity bound like the one from Theorem 1.51. Both bounds are assumed to hold also under shifting and under reduction of the horizon length[20].

[20] These assumptions are rather mild, if the prediction horizon is long enough, and the sampling periods are short enough.

Proposition 2.5 (Nominal stability of the shifted RTI scheme) *There exists an environment $\Xi \subseteq \mathbb{R}^{n_x}$ around the origin such that the RTI feedback with shift computed from MPC problem (2.13) with the terminal constraint $\boldsymbol{F}(\boldsymbol{x_N}, \boldsymbol{u_N}) = \mathbf{0}$ drives the closed loop system (2.12) to the origin for all $\boldsymbol{x^0} \in \Xi$, if the sampling period is sufficiently short, the prediction horizon is sufficiently long, and the objective and the dynamics are sufficiently aligned*[21].

We refer to [DFA+05] for the precise technical assumptions and a formal proof.

To establish nominal stability of the RTI scheme without shifting, additionally a joint Lipschitz continuity condition in $\boldsymbol{x}$ and $\boldsymbol{u}$ on $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{x}$ featuring not too big Lipschitz constants needs to be assumed (see [DFA07] for details). We then have the following result:

Proposition 2.6 (Nominal stability of the RTI scheme without shift) *There exist environments Ξ and $\bar{\Xi} \supseteq \Xi$ around the origin such that the RTI feedback without shift computed from MPC problem (2.13) with a sufficiently strong terminal penalty on $\boldsymbol{F}(\boldsymbol{x_N}, \boldsymbol{u_N})$ drives the closed loop system (2.12) to the origin for all $\boldsymbol{x^0} \in \Xi$, while the state remains in the bounded region $\bar{\Xi}$, if the sampling period is sufficiently short and the prediction horizon is sufficiently long.*

We refer to [DFA07] for further details and a formal proof.

2.4 Hierarchical QP Updates

The key advantage of the RTI scheme is the reduction of the feedback delay from the solution time of a full NLP to essentially the solution time of a single QP. However, the computational effort for the preparation phase may, depending on the problem, still be significantly larger than the effort for the feedback phase and therefore limit the feedback rate decisively. On the other hand, linear MPC — whose feedback rate is only limited by the solution time of one QP — is known to often work well in praxis even if the plant dynamics are inherently nonlinear. We present an approach that closes the gap between nonlinear MPC and linear MPC in the following. The central idea of this approach is to limit the effort per iteration of the RTI scheme by permitting *inexact* first-order approximations in the linear MPC problem. The considerations are mainly due to [BDKS07], where the ideas were introduced in form of so-called *Multi-Level Real-Time*

[21] By sufficiently aligned we essentially mean that the Lagrange multipliers of steady state dynamics in the solution of the dynamic optimization problem do vanish with increasing horizon length, cf. [DFA+05].

Iteration schemes. We adopt a slightly different presentation here, that stresses the flexibility of this idea and eventually leads to the concept of *Mixed-Level Iteration schemes* coined in [FWSB12].

For notational convenience we represent the data of real-time dynamic optimization problems (MPC) and (MHE) in a more compact fashion by introducing

$$\ell(\boldsymbol{w}) := \sum_{k=0}^{N} \ell_k(\boldsymbol{s_k}, \boldsymbol{q_k}),$$

$$\boldsymbol{c}(\boldsymbol{w}) := \begin{bmatrix} \boldsymbol{F_0}(\boldsymbol{s_0}, \boldsymbol{q_0}) - \boldsymbol{s_1} \\ \vdots \\ \boldsymbol{F_{N-1}}(\boldsymbol{s_{N-1}}, \boldsymbol{q_{N-1}}) - \boldsymbol{s_N} \end{bmatrix},$$

$$\boldsymbol{d}(\boldsymbol{w}) := \begin{bmatrix} \boldsymbol{d_0}(\boldsymbol{s_0}, \boldsymbol{q_0}) \\ \vdots \\ \boldsymbol{d_N}(\boldsymbol{s_N}, \boldsymbol{q_N}) \end{bmatrix},$$

and

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := \ell(\boldsymbol{w}) + \boldsymbol{\lambda}^\top \boldsymbol{c}(\boldsymbol{w}) + \boldsymbol{\mu}^\top \boldsymbol{d}(\boldsymbol{w}).$$

In the following we regard for each time index $i = 0, 1, \ldots$ the linear MPC problem given by

$$\min_{\boldsymbol{z}} \quad \frac{1}{2} \boldsymbol{z}^\top \boldsymbol{H^{(i)}} \boldsymbol{z} + \boldsymbol{m^{(i)}}^\top \boldsymbol{z} \tag{2.14a}$$

$$\text{s.t.} \qquad \boldsymbol{0} = \boldsymbol{C^{(i)}}\, \boldsymbol{z} + \boldsymbol{c^{(i)}} \tag{2.14b}$$

$$\boldsymbol{0} \geq \boldsymbol{D^{(i)}}\, \boldsymbol{z} - \boldsymbol{d^{(i)}}\,, \tag{2.14c}$$

which needs to be solved at each sampling time T_i to obtain an update of the feedback law. Here, we assume the linear initial value embedding (in MPC)

$$\boldsymbol{L}\, \boldsymbol{z} = \boldsymbol{s_0^{(i)}} - \hat{\boldsymbol{x}}(T_i),$$

where $\boldsymbol{L} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n_x \times N n_z + n_x}$, to be hidden in the remaining constraints for notational convenience only.

The standard RTI scheme does a full update of the QP's data in each iteration, i.e., chooses

$$\boldsymbol{H^{(i)}} \approx \boldsymbol{\nabla^2_{ww}} \mathcal{L}(\boldsymbol{w^{(i)}}, \boldsymbol{\lambda^{(i)}}, \boldsymbol{\mu^{(i)}}),$$

where the kind of approximation is typically one from the categories of Section 1.4.4, alongside

$$\begin{aligned}
\boldsymbol{m}^{(i)} &:= \boldsymbol{\nabla}_{\boldsymbol{w}} \boldsymbol{\ell}(\boldsymbol{w}^{(i)}) \\
&= \boldsymbol{\nabla}_{\boldsymbol{w}} \boldsymbol{\mathcal{L}}(\boldsymbol{w}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{\mu}^{(i)}) - \boldsymbol{C}^{(i)\top} \boldsymbol{\lambda}^{(i)} - \boldsymbol{D}^{(i)\top} \boldsymbol{\mu}^{(i)}, \\
\boldsymbol{c}^{(i)} &:= \boldsymbol{c}(\boldsymbol{w}^{(i)}), \\
\boldsymbol{C}^{(i)} &:= \boldsymbol{\nabla}_{\boldsymbol{w}} \boldsymbol{c}(\boldsymbol{w}^{(i)}), \\
\boldsymbol{d}^{(i)} &:= \boldsymbol{d}(\boldsymbol{w}^{(i)}),
\end{aligned}$$

and

$$\boldsymbol{D}^{(i)} := \boldsymbol{\nabla}_{\boldsymbol{w}} \boldsymbol{d}(\boldsymbol{w}^{(i)}).$$

The remainder of this section is dedicated to developing a more flexible concept for choosing $\boldsymbol{H}^{(i)}, \boldsymbol{m}^{(i)}, \boldsymbol{C}^{(i)}, \boldsymbol{c}^{(i)}, \boldsymbol{D}^{(i)}$, and $\boldsymbol{d}^{(i)}$.

2.4.1 Linear MPC/MHE with hierarchical updates

Even for systems with inherently nonlinear dynamics the feedback law computed by solving the linear MPC problem (2.14) repeatedly with fixed problem data (except for the initial value embedding $\hat{\boldsymbol{x}}(T_i)$) may still be sufficient to reject smaller disturbances and to cope well with small model uncertainties. This is particularly observed when $\boldsymbol{H}^{(i)}, \boldsymbol{m}^{(i)}, \boldsymbol{C}^{(i)}, \boldsymbol{c}^{(i)}, \boldsymbol{D}^{(i)}$, and $\boldsymbol{d}^{(i)}$ correspond to a linearization in a steady state; but also on an arbitrary trajectory path, a fixed linearization may suffice to yield a good control/estimation performance for a certain period, if the sampling rate is sufficiently high. In particular, the tangential predictor corresponding to the linear MPC law may still approximate the nonlinear problem reasonably well, even if active-set changes occur, cf. [DFH09], and perturbations in the initial value are rejected up to first or even second order (depending on the chosen Hessian matrix approximation), cf. Theorem 2.4.

The idea of partial QP data updates is to have a linear MPC controller running at a certain (high) sampling rate whose data is updated cascadingly. The most rigorous update is a full re-linearization, as it is done in the preparation phase of the original RTI scheme. However, we also may perform only partial updates of the QP data, which then may be performed at a higher rate. We motivate

this idea by regarding the KKT conditions of (2.14) for a fixed (i.e., known) optimal active set, which implicitly defines $\boldsymbol{D}_{\mathbf{act}}^{(i)}$ and $\boldsymbol{d}_{\mathbf{act}}^{(i)}$:

$$\begin{bmatrix} \boldsymbol{H}^{(i)} & \boldsymbol{C}^{(i)\top} & \boldsymbol{D}_{\mathbf{act}}^{(i)\top} \\ \boldsymbol{C}^{(i)} & & \\ \boldsymbol{D}_{\mathbf{act}}^{(i)} & & \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu}_{\mathbf{act}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{m}^{(i)} \\ -\boldsymbol{c}^{(i)} \\ -\boldsymbol{d}_{\mathrm{act}}^{(i)} \end{bmatrix} \tag{2.15}$$

From the perspective of (2.15), the RTI scheme corresponds to solving this linear system once per sampling time (i.e., performing one Newton-type iteration on the KKT conditions of the NLP). Note, however, that the row configuration of $\boldsymbol{D}_{\mathbf{act}}^{(i)}$ and $\boldsymbol{d}_{\mathbf{act}}^{(i)}$ may be altered by active set changes during the QP solution process.

Partial QP updates originate in the observation that the left-hand side matrix of (2.15) is rather expensive to evaluate, and updating it typically makes up for the larger share of the computational effort of the preparation phase. On the other hand, Newton's method is known to converge also for perturbed choices of the left-hand side matrix in the linear system[22]. Inspired by this, we may use adaptations of the RTI scheme that work with inexact derivatives and only perform updates of the right-hand side vector of the objective and constraint residuals.

We remark that feedback can be obtained with an even shorter delay than the solution time of a QP by fixing the active set in (2.15) and only performing one linear system solution (which may even be prepared before a new $\hat{\boldsymbol{x}}(T_i)$ is obtained). This feedback policy is referred to as *local feedback law* in [KWSB10]. It may however lead to an arbitrary violation of path constraints, as constraints which are not binding at sampling time T_{i-1} are simply disregarded at time T_i. Also, if a parametric active set strategy (cf. Section 1.5.4) is employed for the solution of the linear MPC problem, the feedback delay caused by the QP solution is typically not drastically longer than the delay caused by the local feedback law under the assumption that no active set changes occur.

We propose to consider the following partial updates, cf. [BDKS07]:

Feasibility-improving updates The cost of a nonlinear function evaluation corresponds roughly to the cost of one forward simulation of the nonlinear dynamic system (over the full prediction horizon), which is still significantly cheaper than performing a full re-linearization. The data obtained from a

[22] In a sufficiently small neighborhood of a solution, such that we can ignore step size considerations.

forward simulation can be used to update

$$\boldsymbol{c}_{\boldsymbol{k}}^{(\boldsymbol{i})} := \boldsymbol{s}_{\boldsymbol{k+1}}^{(\boldsymbol{i})} - \boldsymbol{F}_{\boldsymbol{k}}(\boldsymbol{s}_{\boldsymbol{k}}^{(\boldsymbol{i})}, \boldsymbol{q}_{\boldsymbol{k}}^{(\boldsymbol{i})}) \tag{2.16a}$$

$$\boldsymbol{d}_{\boldsymbol{k}}^{(\boldsymbol{i})} := \boldsymbol{d}_{\boldsymbol{k}}(\boldsymbol{s}_{\boldsymbol{k}}^{(\boldsymbol{i})}, \boldsymbol{q}_{\boldsymbol{k}}^{(\boldsymbol{i})}). \tag{2.16b}$$

Alongside the nonlinear constraint residuals, we correct the quadratic model of the NLP Lagrangian by the shift in the nonlinear variables, using

$$\boldsymbol{m}^{(\boldsymbol{i})} := \boldsymbol{m}^{(\boldsymbol{i-1})} + \boldsymbol{H}^{(\boldsymbol{i-1})}(\boldsymbol{w}^{(\boldsymbol{i})} - \boldsymbol{w}^{(\boldsymbol{i-1})}). \tag{2.17}$$

Matrix-vector products and the forward simulation step are the most expensive operations of this update.

In the particular case where we choose a Gauss-Newton approximation of the Hessian matrix of the Lagrangian function, we can even use a slightly better correction of the QP objective at roughly the same computational costs. If the nonlinear MPC/MHE objective function is given by

$$f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{w})\|_2^2,$$

we have that (2.17) corresponds to

$$\boldsymbol{m}^{(\boldsymbol{i})} := \boldsymbol{\nabla r}^{(\boldsymbol{i-1})} \left(\boldsymbol{r}^{(\boldsymbol{i-1})} - \boldsymbol{\nabla r}^{(\boldsymbol{i-1})}(\boldsymbol{w}^{(\boldsymbol{i})} - \boldsymbol{w}^{(\boldsymbol{i-1})}) \right),$$

where $\boldsymbol{\nabla r}^{(\boldsymbol{i-1})} = \boldsymbol{\nabla r}(\boldsymbol{w}^{(\boldsymbol{i-1})})$ and $\boldsymbol{r}^{(\boldsymbol{i-1})} = \boldsymbol{r}(\boldsymbol{w}^{(\boldsymbol{i-1})})$ if a full Gauss-Newton re-linearization was performed in iteration $i-1$, and recursively $\boldsymbol{\nabla r}^{(\boldsymbol{i-1})} = \boldsymbol{\nabla r}^{(\boldsymbol{i-2})}$ and $\boldsymbol{r}^{(\boldsymbol{i-1})} = \boldsymbol{r}^{(\boldsymbol{i-2})}$, and so on, if the objective was not updated. Instead of using this correction term for an extrapolation of the expected change in the objective's least-squares residuals, we can directly reevaluate the least squares residuals and use

$$\boldsymbol{m}^{(\boldsymbol{i})} := \boldsymbol{\nabla r}^{(\boldsymbol{i-1})}\boldsymbol{r}(\boldsymbol{w}^{(\boldsymbol{i})}), \tag{2.18}$$

as we expect the cost for evaluating the objective residuals to be rather negligible when performed alongside the dynamic system simulation.

Optimality-improving updates In addition to a correction of the nonlinear constraint residuals through (2.16) we may also make use of the exact gradient of the NLP Lagrangian at little extra cost. Instead of using (2.17) or (2.18), the update rule

$$\begin{aligned}\boldsymbol{m}^{(i)} &:= \boldsymbol{\nabla \ell}(\boldsymbol{w}) + \left(\boldsymbol{C}^{(\boldsymbol{i})} - \boldsymbol{C}^{(\boldsymbol{i-1})}\right)^\top \boldsymbol{\lambda}_{\mathrm{QP}}^{(\boldsymbol{i-1})} + \left(\boldsymbol{D}^{(\boldsymbol{i})} - \boldsymbol{D}^{(\boldsymbol{i-1})}\right)^\top \boldsymbol{\mu}_{\mathrm{QP}}^{(\boldsymbol{i-1})} \\ &= \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{\mathcal{L}}(\boldsymbol{w}^{(\boldsymbol{i})}, \boldsymbol{\lambda}^{(\boldsymbol{i})}, \boldsymbol{\mu}^{(\boldsymbol{i})}) - \boldsymbol{C}^{(\boldsymbol{i-1})^\top}\boldsymbol{\lambda}_{\mathrm{QP}}^{(\boldsymbol{i-1})} - \boldsymbol{D}^{(\boldsymbol{i-1})^\top}\boldsymbol{\mu}_{\mathrm{QP}}^{(\boldsymbol{i-1})}\end{aligned} \tag{2.19}$$

is applied[23]. It is important to observe here that the matrices $\boldsymbol{C}^{(i)}$ and $\boldsymbol{D}^{(i)}$ are never formed explicitly in an efficient implementation. Instead, we rely on the adjoint mode of automatic differentiation to compute $\boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{\mathcal{L}}(\boldsymbol{w}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{\mu}^{(i)})$ at the cost of a small constant multiple of a forward simulation, cf. Section 1.4.5. Since the matrices $\boldsymbol{C}^{(i-1)}$ and $\boldsymbol{D}^{(i-1)}$ are already known at sampling time T_i, ${\boldsymbol{C}^{(i-1)}}^\top \boldsymbol{\lambda}_{\mathrm{QP}}^{(i-1)}$ and ${\boldsymbol{D}^{(i-1)}}^\top \boldsymbol{\mu}_{\mathrm{QP}}^{(i-1)}$ are also rather cheap to evaluate by simple matrix-vector products.

2.4.2 Convergence guarantees

With inexact and incomplete data updates, we have the following convergence results due to [BDKS07].

Theorem 2.7 (Limit of feasibility-improving updates) *Consider QP (2.14), with initial data* $\boldsymbol{H}^{(0)} := \boldsymbol{\nabla}^2_{\boldsymbol{ww}}\boldsymbol{\mathcal{L}}(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$, $\boldsymbol{m}^{(0)} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{\ell}(\bar{\boldsymbol{w}})$, $\boldsymbol{C}^{(0)} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{c}(\bar{\boldsymbol{w}})$, $\boldsymbol{c}^{(0)} := \boldsymbol{c}(\bar{\boldsymbol{w}})$, $\boldsymbol{D}^{(0)} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{d}(\bar{\boldsymbol{w}})$, *and* $\boldsymbol{d}^{(0)} := \boldsymbol{d}(\bar{\boldsymbol{w}})$, *where* $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ *is a primal-dual reference point*[24]. *Let the QP data be updated inexactly by (2.16) and (2.17) in each iteration on all stages* $k \in \mathcal{S}$. *Assume that* $\hat{\boldsymbol{x}}(T_i) := \tilde{\boldsymbol{x}}$ *is constant for all* $i \geq 0$. *Then, every limit* $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*_{\mathbf{QP}}, \boldsymbol{\mu}^*_{\boldsymbol{QP}})$ *(if existing) of the implicitly defined nonlinear iterates* $(\boldsymbol{w}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{\mu}^{(i)})$ *is a KKT point of the perturbed, but nonlinearly feasible problem*

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}(\boldsymbol{w} - \bar{\boldsymbol{w}})^\top \boldsymbol{H}^{(0)}(\boldsymbol{w} - \bar{\boldsymbol{w}}) + (\boldsymbol{m}^{(0)} + \boldsymbol{e})^\top \boldsymbol{w} \tag{2.20a}$$

$$\text{s.t.} \qquad \mathbf{0} = \boldsymbol{c}(\boldsymbol{w}) \tag{2.20b}$$

$$\mathbf{0} \geq \boldsymbol{d}(\boldsymbol{w}) \tag{2.20c}$$

$$\boldsymbol{L}\,\boldsymbol{w} = \tilde{\boldsymbol{x}}, \tag{2.20d}$$

where $\boldsymbol{e} := \left(\boldsymbol{C}^{(0)} - \boldsymbol{\nabla}\boldsymbol{c}(\boldsymbol{w}^*)\right)^\top \boldsymbol{\lambda}^*_{\mathbf{QP}} + \left(\boldsymbol{D}^{(0)} - \boldsymbol{\nabla}\boldsymbol{d}(\boldsymbol{w}^*)\right)^\top \boldsymbol{\mu}^*_{\mathbf{QP}}$.

Proof See [BDKS07]. □

Corollary 2.8 (Limit of Gauss-Newton feasibility-improving updates) *Let the objective of the nonlinear MPC/MHE problem be of least-squares type* $\ell(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{w})\|_2^2$. *Consider QP (2.14) with the initial data* $\boldsymbol{H}^{(0)} := {\boldsymbol{R}^{(0)}}^\top \boldsymbol{R}^{(0)}$, *where* $\boldsymbol{R}^{(0)} := \boldsymbol{\nabla}\boldsymbol{r}(\bar{\boldsymbol{w}})$, $\boldsymbol{m}^{(0)} := {\boldsymbol{R}^{(0)}}^\top \boldsymbol{r}(\bar{\boldsymbol{w}})$, $\boldsymbol{C}^{(0)} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{c}(\bar{\boldsymbol{w}})$,

[23] Recall that $\boldsymbol{\lambda}_{\mathrm{QP}}^{(i-1)} = \boldsymbol{\lambda}_{\mathrm{NLP}}^{(i)}$ and $\boldsymbol{\mu}_{\mathrm{QP}}^{(i-1)} = \boldsymbol{\mu}_{\mathrm{NLP}}^{(i)}$, cf. Section 1.4.3.

[24] $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ could, for example (but not necessarily), be a KKT point corresponding to an equlibrium.

$\boldsymbol{c}^{(\mathbf{0})} := \boldsymbol{c}(\bar{\boldsymbol{w}})$, $\boldsymbol{D}^{(\mathbf{0})} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{d}(\bar{\boldsymbol{w}})$, *and* $\boldsymbol{d}^{(\mathbf{0})} := \boldsymbol{d}(\bar{\boldsymbol{w}})$, *where* $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ *is a primal-dual reference point. Let the QP data be updated inexactly by* (2.16) *and* (2.18) *in each iteration on all stages* $k \in \mathcal{S}$. *Assume that* $\hat{\boldsymbol{x}}(T_i) := \tilde{\boldsymbol{x}}$ *is constant for all* $i \geq 0$. *Then, every limit* $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*_{\mathbf{QP}}, \boldsymbol{\mu}^*_{\boldsymbol{QP}})$ *(if existing) of the implicitly defined nonlinear iterates* $(\boldsymbol{w}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{\mu}^{(i)})$ *is a KKT point of the perturbed, but nonlinearly feasible problem*

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{w})\|_2^2 + \boldsymbol{e}^\top \boldsymbol{w} \tag{2.21a}$$

$$\text{s.t.} \qquad \mathbf{0} = \boldsymbol{c}(\boldsymbol{w}) \tag{2.21b}$$

$$\mathbf{0} \geq \boldsymbol{d}(\boldsymbol{w}) \tag{2.21c}$$

$$\boldsymbol{L}\,\boldsymbol{w} = \tilde{\boldsymbol{x}}, \tag{2.21d}$$

where

$$\boldsymbol{e} := \left(\boldsymbol{C}^{(\mathbf{0})} - \boldsymbol{\nabla}\boldsymbol{c}(\boldsymbol{w}^*)\right)^\top \boldsymbol{\lambda}^*_{\mathbf{QP}} + \left(\boldsymbol{D}^{(\mathbf{0})} - \boldsymbol{\nabla}\boldsymbol{d}(\boldsymbol{w}^*)\right)^\top \boldsymbol{\mu}^*_{\mathbf{QP}} + \left(\boldsymbol{R}^{(\mathbf{0})} - \boldsymbol{\nabla}\boldsymbol{r}(\boldsymbol{w}^*)\right)^\top \boldsymbol{r}(\boldsymbol{w}^*).$$

Proof See [BDKS07]. □

Theorem 2.9 (Limit of optimality-improving updates) *Consider QP* (2.14) *with the initial data* $\boldsymbol{H}^{(\mathbf{0})} := \boldsymbol{\nabla}^{\mathbf{2}}_{\boldsymbol{ww}}\boldsymbol{\mathcal{L}}(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$, $\boldsymbol{m}^{(\mathbf{0})} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{\ell}(\bar{\boldsymbol{w}})$, $\boldsymbol{C}^{(\mathbf{0})} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{c}(\bar{\boldsymbol{w}})$, $\boldsymbol{c}^{(\mathbf{0})} := \boldsymbol{c}(\bar{\boldsymbol{w}})$, $\boldsymbol{D}^{(\mathbf{0})} := \boldsymbol{\nabla}_{\boldsymbol{w}}\boldsymbol{d}(\bar{\boldsymbol{w}})$, *and* $\boldsymbol{d}^{(\mathbf{0})} := \boldsymbol{d}(\bar{\boldsymbol{w}})$, *where* $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ *is a primal-dual reference point. Let the QP data be updated inexactly by* (2.16) *and* (2.19) *in each iteration on all stages* $k \in \mathcal{S}$. *Assume that* $\hat{\boldsymbol{x}}(T_i) := \tilde{\boldsymbol{x}}$ *is constant for all* $i \geq 0$. *Then, every limit* $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*_{\mathbf{QP}}, \boldsymbol{\mu}^*_{\boldsymbol{QP}})$ *(if existing) of the implicitly defined nonlinear iterates* $(\boldsymbol{w}^{(i)}, \boldsymbol{\lambda}^{(i)}, \boldsymbol{\mu}^{(i)})$ *is a KKT point of the original MPC/MHE problem.*

Proof See [BDKS07]. □

With the limits of the different update rules in place, we can establish conditions, under which we achieve convergence in analogy to Section 2.3.3. To this end, we reduce the nonlinear optimization problem — under the assumption of an already identified optimal active set — to the root finding problem

$$\boldsymbol{F}(\boldsymbol{y}) = \mathbf{0}.$$

Here, $\boldsymbol{y} = (\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}_{\mathbf{act}})$ summarize the relevant primal-dual iterates for a fixed active set. Since, depending on whether we perform updates in the sense of

Theorem 2.7, Corollary 2.8, or Theorem 2.9, the nonlinear problems which are solved by the limit solutions are different, the nonlinear system $\boldsymbol{F}$ takes one of the three instantiations

$$\boldsymbol{F}_{\mathbf{B}}(\boldsymbol{y}) := \begin{bmatrix} \boldsymbol{H}^{(\mathbf{0})}(\boldsymbol{w} - \bar{\boldsymbol{w}}) + \boldsymbol{m}^{(\mathbf{0})} + \boldsymbol{C}^{(\mathbf{0})^\top}\boldsymbol{\lambda} + \boldsymbol{D}_{\mathbf{act}}^{(\mathbf{0})^\top}\boldsymbol{\mu}_{\mathbf{act}} \\ \boldsymbol{c}(\boldsymbol{w}) \\ \boldsymbol{d}_{\mathbf{act}}(\boldsymbol{w}) \end{bmatrix},$$

$$\boldsymbol{F}_{\mathbf{B_{GN}}} := \begin{bmatrix} \boldsymbol{R}^{(\mathbf{0})^\top}\boldsymbol{r}(\boldsymbol{w}) + \boldsymbol{C}^{(\mathbf{0})^\top}\boldsymbol{\lambda} + \boldsymbol{D}_{\mathbf{act}}^{(\mathbf{0})^\top}\boldsymbol{\mu}_{\mathbf{act}} \\ \boldsymbol{c}(\boldsymbol{w}) \\ \boldsymbol{d}_{\mathbf{act}}(\boldsymbol{w}) \end{bmatrix},$$

$$\boldsymbol{F}_{\mathbf{C}} := \begin{bmatrix} \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}_{\mathbf{act}}) \\ \boldsymbol{c}(\boldsymbol{w}) \\ \boldsymbol{d}_{\mathbf{act}}(\boldsymbol{w}) \end{bmatrix}.$$

Note that we have implicitly defined $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}_{\mathbf{act}})$ as the variant of $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ where all multipliers of inactive constraints are fixed to 0. The indexing of $\boldsymbol{F}_{\cdot}$ refers to the fact that in [BDKS07] updates in the sense of Theorem 2.7 or Corollary 2.8 followed by a feedback-generating QP solution are referred to as *Level B iterations*, whereas updates in the sense of Theorem 2.9 followed by a feedback-generating QP solution are referred to as *Level C iterations*.

The inexact RTI variants defined by Theorem 2.7, Corollary 2.8, and Theorem 2.9 are, for a fixed active set, given by the Newton-type iterations

$$\boldsymbol{J}\,\boldsymbol{\Delta y}^{(\boldsymbol{i})} = \boldsymbol{F}_{\cdot}(\boldsymbol{y}^{(\boldsymbol{i})})$$

and $\boldsymbol{y}^{(\boldsymbol{i}+\mathbf{1})} = \boldsymbol{y}^{(\boldsymbol{i})} + \boldsymbol{\Delta y}^{(\boldsymbol{i})}$. For all considered variants, the Newton-type matrix $\boldsymbol{J}$ (again assuming the active set does not change) is identically given by:

$$\boldsymbol{J} := \begin{bmatrix} \boldsymbol{H}^{(\mathbf{0})} & \boldsymbol{C}^{(\mathbf{0})^\top} & \boldsymbol{D}_{\mathbf{act}}^{(\mathbf{0})^\top} \\ \boldsymbol{C}^{(\mathbf{0})} & & \\ \boldsymbol{D}_{\mathbf{act}}^{(\mathbf{0})} & & \end{bmatrix}$$

We assume in the following that the constraint matrix $\begin{bmatrix} \boldsymbol{C}^{(\mathbf{0})} \\ \boldsymbol{D}_{\mathbf{act}}^{(\mathbf{0})} \end{bmatrix}$ is of full row rank and that $\boldsymbol{H}^{(\mathbf{0})}$ is positive definite on the nullspace of these (active) constraints[25]. This allows us to state the following theorem on convergence:

Theorem 2.10 (Convergence of inexact RTI) *Let* $\boldsymbol{y}^* :\equiv (\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ *be a regular KKT point of* (MPC)/(MHE) *(or, respectively,* (2.20) *or* (2.21)*) that*

[25] Note that $\boldsymbol{J}$ is therefore invertible.

fulfills the strict complementarity condition. Then there is a neighborhood $\mathcal{U}$ *of* $\boldsymbol{y}^*$ *such that the optimal active set is constant. If we can furthermore choose* $\mathcal{U}$ *such that a compatibility condition in the form of*

$$\left\| \mathbf{I} - \boldsymbol{J}^{-1} \frac{\partial \boldsymbol{F}_\cdot}{\partial \boldsymbol{y}}(\boldsymbol{y}) \right\| \leq \kappa$$

for $\kappa < 1$ *is fulfilled for all* $\boldsymbol{y} \in \mathcal{U}$*, we have that the iterations defined by Theorem 2.9 (or, respectively, by Theorem 2.7 or Corollary 2.8) converge to the optimal solution* $\boldsymbol{y}^*$ *of* (MPC)/(MHE) *(or, respectively,* (2.20) *or* (2.21)*) for all initial guesses* $\boldsymbol{y}^{(\mathbf{0})}$ *that satisfy*

$$\|\boldsymbol{y}^* - \boldsymbol{y}^{(\mathbf{0})}\| + \frac{\|\boldsymbol{J}^{-1} \boldsymbol{F}_\cdot(\boldsymbol{y}^{(\mathbf{0})})\|}{1-\kappa} \in \mathcal{U}.$$

Proof See [BDKS07]. □

2.4.3 Assembly of Multi- and Mixed-Level Iteration schemes

In [BDKS07] it was proposed to assemble the different update rules for the linear MPC problem (2.14), namely the feasibility-improving, the optimality-improving, and the full re-linearization updates to fixed schemes. In doing so, feedback-generating solutions of the linear MPC problem with a simple initial value embedding were referred to as *level A iterations*, while feasibility-improving updates followed by a feedback-generating QP solution were labeled *level B iterations*, optimality-improving updates with a subsequent QP solution *level C iterations*, and full RTI scheme iterations *level D iterations*.

A *Multi-Level Iteration (MLI) scheme* can be assembled from these components by specifying rates (through multiples of the sampling time) at which the individual levels are executed. For example, an $\mathrm{D}^6\mathrm{C}^\infty\mathrm{B}^2\mathrm{A}^1$ scheme would correspond to solve a feedback-generating linear MPC problem at each sampling time, whose data is updated according to the appropriate feasibility improving update rules every second sampling time, and by a full re-linearization every six sampling times. Optimality-improving data updates are not performed in this case.

The principle of MLI schemes can be carried further, as proposed in [FWSB12], by using different update rules for each individual stage at each sampling time. This approach is motivated by the observation that we essentially solve an open-loop OCP at each sampling time in MPC. It therefore can be expected that the (optimal) controls computed for a certain stage become more and more outdated as the process evolves, since disturbances or measurement errors add up and therefore require a re-optimization at later sampling times. Hence, when

solving a single instance of OCP in an MPC loop, it is more important to model the process well in earlier parts of the prediction horizon than in later parts, where predictions are less accurate and the computed controls can be refined at a later point in time. We therefore propose two modifications to the MLI approach:

Fractional-level iterations Apply one of the update levels described above only to the first $N_{\text{frac}} < N$ stages. For example, a fractional-D (or D′) iteration performs a full re-linearization only on the first N_{frac} stages and reuses the old data for the remaining intervals.

Mixed-level iterations Apply one of the update levels D or C to the first N_{frac} stages and another (computationally less expensive) update level for the other intervals. For example, a D/B iteration computes full re-linearization information on the first N_{frac} stages, but only evaluates constraint residuals (and approximates the objective function gradient accordingly) on the remaining stages.

Analogously, we can argue in MHE that the estimate a state on an earlier stage is initialized with is already close to its true value. Therefore, an update of the full derivative information on an earlier stage may lead to little additional quality in the most recent state and parameter estimate, particularly in comparison to an update of the derivative information on a later (i.e., more recent) stage. We therefore define fractional-level iterations for MHE as applying a certain update level only to the last N_{frac} stages; mixed-level iterations for MHE consequently are the mixture of applying a certain higher-fidelity update level to the last N_{frac} stages and a lower-complexity one to the remaining stages.

All in all, by using the more flexible concept of multi-, fractional-, and mixed-level iteration schemes in the just outlined fashion, we may obtain a significant reduction of the preparation phase, as costly full derivative evaluations can be avoided to a smaller or larger extent, depending on the specific scheme. Still, as long as higher-fidelity updates are performed at a sufficiently high rate, we can expect a satisfactory control or estimation performance.

2.4.4 Parallelization of feedback and data updates

On a parallel, multi-threaded computing architecture, we can even avoid the rigid concept of fixing a specific scheme (as in the notion of multi- or mixed-level iteration schemes) beforehand. We observe that the feedback-generating

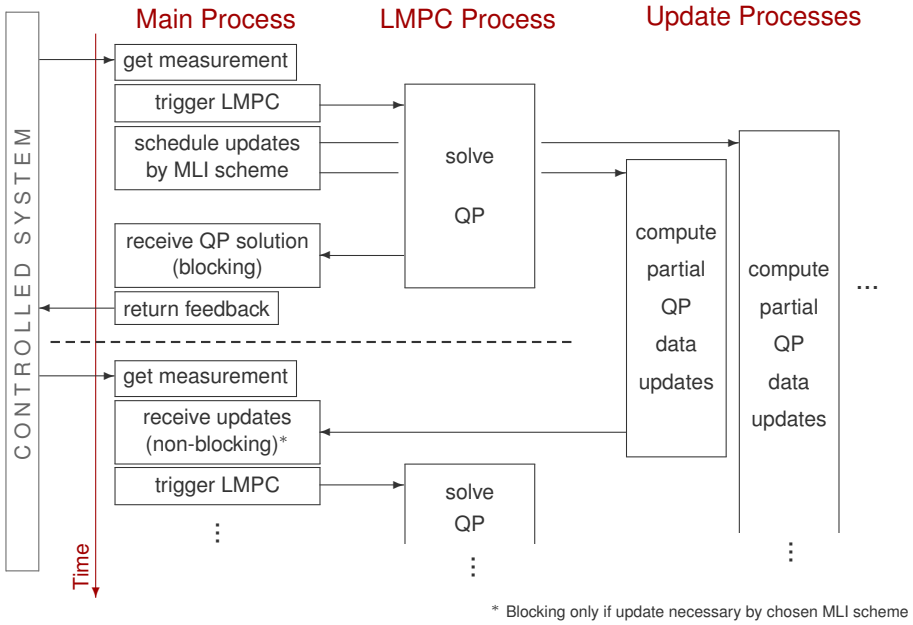


Figure 2.4: Communication diagram for a concurrent implementation of an RTI scheme with hierarchical data updates.

linear MPC iterations and the different QP data updates can be run largely independently from one another in separated-memory processes on different computational threads, only updating their information at the start of each respective iteration. In particular, the different update schemes proposed in Section 2.4.1 can be run concurrently. As optimality-improving updates come at significantly smaller computational costs than full re-linearizations[26], we can, for example, improve the feedback-generating linear MPC problem already by using the updates computed from (2.16) and (2.17)/(2.18) on a separate thread, before a full re-linearization is available; while the full re-linearization update is still being computed, system disturbances can already be rejected with the guarantees of Theorems 2.4 and 2.9.

We essentially propose to orchestrate the feedback-generating linear MPC and the hierarchical data updates as sketched in Figure 2.4. We have a main process that acts as scheduler for the desired scheme, triggering feedback and update iterations as needed. After a state measurement is observed from the dynamic system, a feedback iteration is triggered using the most recent

[26]For identical problem data, linearization points, and integrator error control (in case of adaptive integrators).

quadratic approximation of the nonlinear problem which is available. While the QP is solved by the linear MPC process, update iterations are triggered in accordance with the specified MLI scheme and available computational power, which limits the number of update processes to be run in parallel by (at most) the number of available threads. Also, quality criteria measuring the accuracy of the current quadratic model may be used to trigger the appropriate data update routine. Such criteria may be estimates of the incompatibility factor κ, as introduced in Theorem 1.51, which is, however, typically difficult to achieve in practical real-time computations, cf. [Pot11b]. Alternatively, a comparison between predicted and actually observed system states may indicate when an update of the linearization information is beneficial. We may even use feasibility-improving updates for probing of the accuracy of the current quadratic model after a larger step in the nonlinear variables is performed, e.g., to figure out which stages require an update of the linearization information most urgently in situations where the available computational power is limited and the cost of a full re-linearization is high.

For (partial) re-linearization, the current set of NLP variables is sent to a triggered update process in order to be used as reference point. Meanwhile, after the feedback-generating QP solution is completed, the first (nonlinear) control action q_0 is immediately sent to the controlled system. At the next sampling time, after observing a new measurement, QP data updates are received — if available. Only if the respective data updates are needed in compliance with the specified MLI scheme the updates are waited for. A feedback iteration is triggered, computing a solution to a potentially updated QP. Again update iterations are triggered as needed and after completion of the feedback iteration the first control action is immediately returned.

2.4.5 The controller package CHUCS

The flexible mixed-level and multi-level iteration framework as sketched in Figure 2.4 has been implemented in the CasADi-based (see [AÅD12]) Python code CHUCS[27]. This implementation is MPC-centered and allows for the flexible assembly of feedback-generating and data-updating routines. MPC problems can be specified in the CasADi syntax. Function evaluations and sensitivities are computed in CHUCS using CVODES from the SUNDIALS integrator suite, see [HBG+05]. The repeated solution of quadratic programs is performed using the Online Active Set Strategy implemented in qpOASES (see [FBD08, FKP+13]). For sparsity exploitation a condensing procedure based on [Lei95, Lei99] is

[27] CHUCS stands for *Concurrently Hierarchically Updating Controller Schemes*.

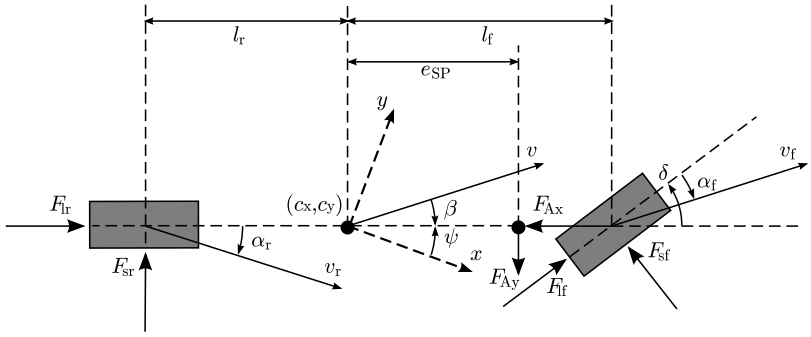


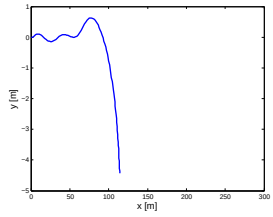
Figure 2.5: Coordinates and considered forces of the single-track vehicle model. The figure is aligned with the vehicle's local coordinate system, while the dashed pair of vectors $(\boldsymbol{x}, \boldsymbol{y})$ depicts the global coordinate system. Taken from [KSBS10, FWSB12].

available. The concurrent computational threads are managed using Python's `multiprocessing` package.

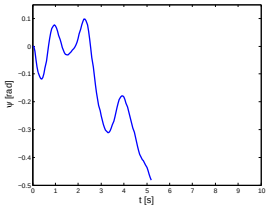
2.4.6 Numerical case study

We demonstrate the capabilities of the flexible mixed-level concept in the following in a disturbance rejection scenario for a ground vehicle traveling at high speed. This case study is essentially identical to the one from [ABK+09], where the effectiveness of the standard MLI framework from [BDKS07] was demonstrated. The nonlinear single-track car model from [Ger05, KSBS10], featuring a Pacejka-type tire model, is used to characterize the vehicle dynamics. It is governed by an ODE system in seven states $x = (c_x, c_y, \psi, v, \beta, w_z, \delta)$ describing the position, orientation, velocities, and the steering angle of the vehicle. Three control inputs $u = (w_\delta, F_B, \phi)$ are available, manipulating the steering rate, deceleration force, and throttle. The model is nonlinear in the states as well as in the control ϕ. Coordinates, angles, and forces are visualized in Figure 2.5. We refer to Chapter 7 for an in-depth discussion of the dynamic equations of ground vehicles.

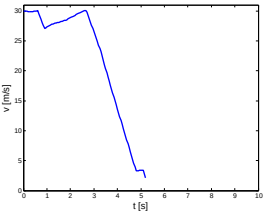
We consider a scenario, where a vehicle with a mass of 1.300 kg is driving on a straight lane at a speed of 30 m/s. An impulse of magnitude $2 \cdot 10^4$ N is acting on the rear axle perpendicular to the driving direction for 0.05 s. In open loop control, the impact is strong enough to make the car spin multiple times and push it off the lane. It is the aim of the controller to keep the vehicle on the



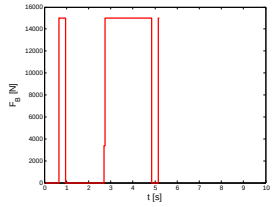
(a) Vehicle path (X-Y plane).



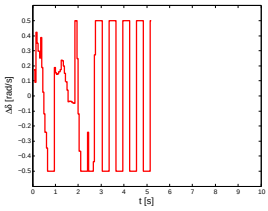
(b) Vehicle orientation



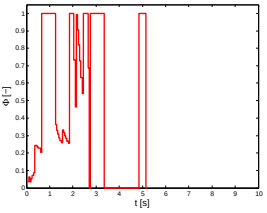
(c) Velocity of the vehicle.



(d) Vehicle braking force.



(e) Vehicle steering rate.



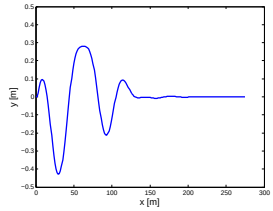
(f) Vehicle accelerator

Figure 2.6: Disturbance rejection scenario with a $\mathrm{D}^6\mathrm{A}^1$ controller. The vehicle strays disastrously from its lane and the controller crashes after roughly 5 s.

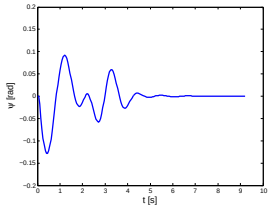
lane while retaining a speed of 30 m/s. To this end, we use a least-squares objective formulation, minimizing the deviation from the track reference (i.e., the center line) and the cruising speed, and regularizing the controls $w_\delta, F_\mathrm{B}, \phi$. All states and controls have two-sided physical limits. The full system state information is available at a resolution of 50 ms and we use a prediction horizon length of 16 sampling times.

We demonstrate the effectiveness of fractional full re-linearizations by comparing two different controllers. Both solve a feedback QP at every sampling time and in parallel compute a full re-linearization update over the whole prediction horizon every 6 samples. Additionally, the second controller features full re-linearization updates on the first $N_\mathrm{frac} = 2$ stages every two samples. As can be seen from Figure 2.6, the first controller without fractional updates fails to stabilize the car, but crashes after roughly 5 s of simulation. The second controller stabilizes the vehicle after roughly 5 s, cf. Figure 2.7. The maximum deviation is less than 0.5 m from the desired straight driving line and less than 1.2 m/s from the reference velocity.

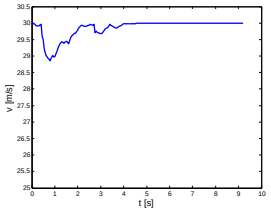
We obtained the following computational results on a standard computer featuring a dual-core Intel i7 CPU at 2.7 GHz clock speed and 8 GB of RAM. We stress that these computation times only serve the purpose of permitting



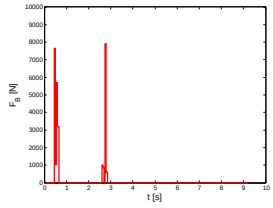
(a) Vehicle path (X-Y plane).



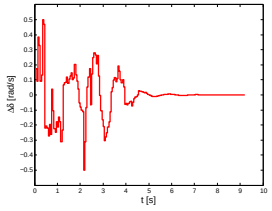
(b) Vehicle orientation



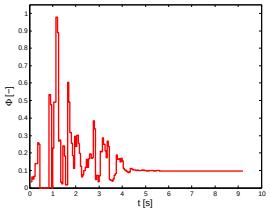
(c) Velocity of the vehicle.



(d) Vehicle braking force.



(e) Vehicle steering rate.



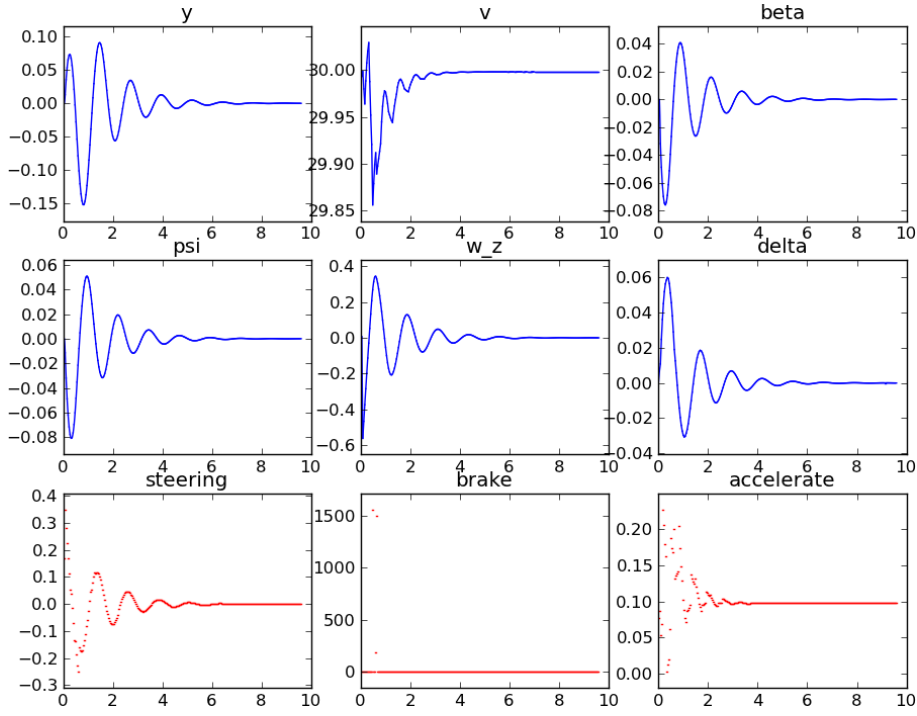
(f) Vehicle accelerator

Figure 2.7: Disturbance rejection scenario with a $\mathrm{D}^6\mathrm{D}'^2\mathrm{A}^1$ controller. The controller is able to reject the shock, and stabilizes the vehicle at its reference after roughly 5 s.

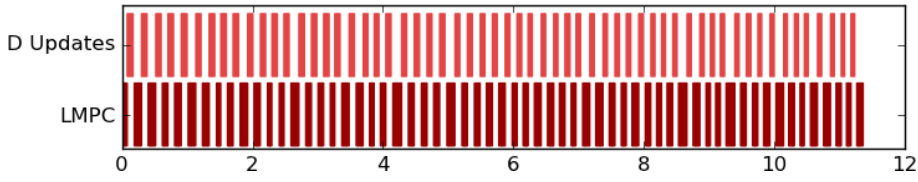
a relative comparison of the costs for the different computations. Using a lower-level programming language such as C/C++ would, most certainly, lead to significant improvements in the absolute time scales. The average clock time for one feedback-generating QP solution was 1.8 ms, while an average full re-linearization update took 117.7 ms (thereof condensing 20.6 ms). The average clock time for one D' update iteration was 27.1 ms (thereof condensing 12.8 ms).

In a second comparison, we analyze the effect of a concurrent execution of data updates and feedback-generating linear MPC solutions. We regard a $\mathrm{D}^4\mathrm{A}^1$ controller in sequential and in parallel execution on the same vehicle benchmark problem. Observe that the control performance of a parallel MLI controller may be worse than the performance of a sequential MLI controller (at least from the viewpoint of this non-real-time, simulation-based analysis), as the nonlinear iterate on which a full re-linearization is based in the parallel execution may correspond to an older sampling time compared to the sequential execution, where the re-linearization is only triggered when requested by the chosen scheme and based on the most-recent nonlinear iterate.

On the positive side, however, we can see from Figures 2.8 and 2.9 that the



(a) Sequential controller performance.

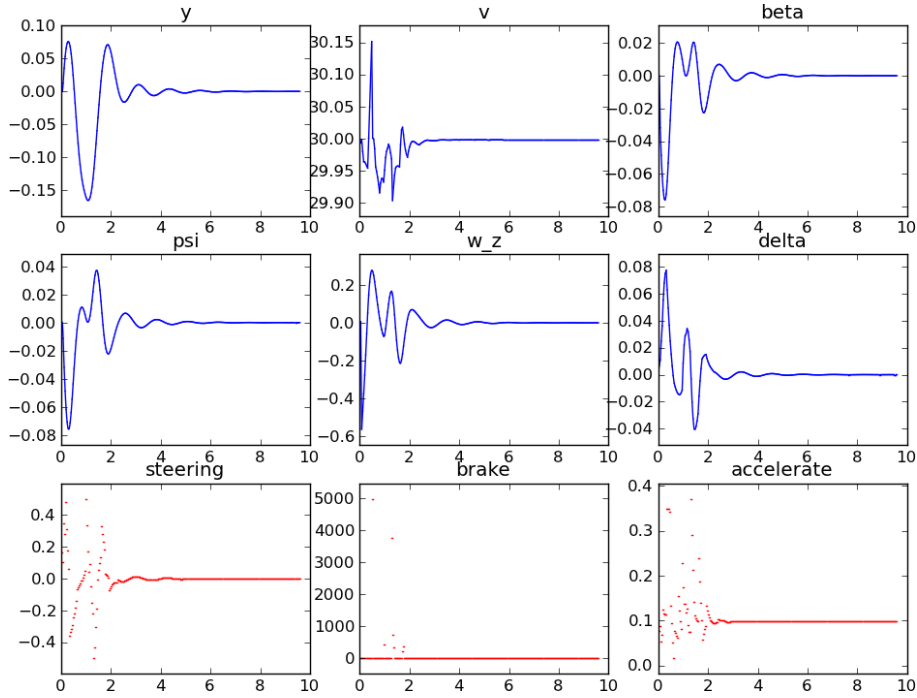


(b) Sequential execution time diagram.

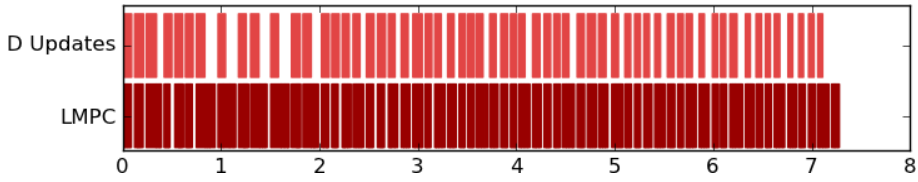
Figure 2.8: Controller performance and computation time diagram of a D^4A^1 scheme in sequential execution.

difference between the sequential and parallel controller execution in terms of control performance is rather small. Considering the execution time diagrams[28], we observe a roughly 60% parallel efficiency of the parallel controller. A slightly higher parallel efficiency would have been possible by switching from a blocking to a non-blocking QP data update reception strategy, i.e., by not forcing the

[28]The diagrams show time on the horizontal axis. Red means the corresponding computational process is computing, while white means it is idle.



(a) Parallel controller performance.



(b) Parallel execution time diagram.

Figure 2.9: Controller performance and computation time diagram of a D^4A^1 scheme in parallel execution.

linear MPC to wait for a data update when specified by the MLI scheme. This would, however, come at the cost of a deteriorated control performance.

In practice, a tradeoff needs to be found on this scale between feedback rate and control performance. A more sophisticated scheduling routine, which takes into account a prediction of the execution times of each routine on a real-time platform, may help in this respect.

Part II

Structure-Exploiting Quadratic Programming

Chapter 3

Distinctive Structures in Dynamic Optimization

It is clear from Chapter 2 that both (MHE) and (MPC) are structurally equivalent to (MS-NLP). In the following, we analyze the characteristic structures of these dynamic optimization NLPs in detail. Specifically, we focus on the QP subproblems, which appear when applying based SQP methods in general or the RTI scheme in particular, and which inherit the distinctive dynamic optimization sparsity patterns from their NLP origins.

To this end, let us consider a generic quadratic subproblem of the form

$$\min_{\boldsymbol{z}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{z}^\top \boldsymbol{H}\boldsymbol{z} + \boldsymbol{m}^\top \boldsymbol{z} \tag{3.1a}$$

$$\text{s.t.} \quad \boldsymbol{0} = \boldsymbol{C}\boldsymbol{z} + \boldsymbol{c} \tag{3.1b}$$

$$\boldsymbol{D}\boldsymbol{z} \leq \boldsymbol{d}\,. \tag{3.1c}$$

We assume that parametric dependencies, like initial value embedding constraints (MPC4) for example, are hidden in the stage constraints (3.1c).

Independently of whether QP (3.1) stems from an online or an offline dynamic optimization problem, a control or an estimation problem, it exhibits the

following characteristic matrix sparsity patterns:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H_0} & & & \\ & \boldsymbol{H_1} & & \\ & & \ddots & \\ & & & \boldsymbol{H_N} \end{bmatrix} \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C_0} & -\boldsymbol{E_0} & & & \\ & \boldsymbol{C_1} & -\boldsymbol{E_1} & & \\ & & \ddots & \ddots & \\ & & & \boldsymbol{C_{N-1}} & -\boldsymbol{E_{N-1}} \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D_0} & & & \\ & \boldsymbol{D_1} & & \\ & & \ddots & \\ & & & \boldsymbol{D_N} \end{bmatrix}.$$

Matching these patterns, we have

$$\boldsymbol{m} = (\boldsymbol{m_0}, \boldsymbol{m_1}, \ldots, \boldsymbol{m_N}),$$

$$\boldsymbol{c} = (\boldsymbol{c_0}, \boldsymbol{c_1}, \ldots, \boldsymbol{c_{N-1}}),$$

and

$$\boldsymbol{d} = (\boldsymbol{d_0}, \boldsymbol{d_1}, \ldots, \boldsymbol{d_N}).$$

The block entries are, for $n_z := n_x + n_u$, $\boldsymbol{H_k} \in \mathbb{R}^{n_z \times n_z}$, $\boldsymbol{C_k} \in \mathbb{R}^{n_x \times n_z}$, $\boldsymbol{E_k} := \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$, $\boldsymbol{D_k} \in \mathbb{R}^{n_d \times n_z}$ (w.l.o.g. the numbers of affine constraints are identical on all QP stages), $\boldsymbol{m_k} \in \mathbb{R}^{n_z}$, $\boldsymbol{c_k} \in \mathbb{R}^{n_x}$, and $\boldsymbol{d_k} \in \mathbb{R}^{n_d}$ (again without loss of generality) for all $k \in \mathcal{S}_N$. We further have $\boldsymbol{H_N} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{D_N} \in \mathbb{R}^{n_d \times n_x}$, and $\boldsymbol{d_N} \in \mathbb{R}^{n_d}$. We assume that parameters are hidden in the state vector in this context, cf. Section 1.1.1.

Particularly for long horizon lengths N (relative to the number of states and controls, n_x, and n_u) the involved matrices obviously are very sparse. It is therefore crucial for the performance of any optimization algorithm to exploit this sparsity in the best possible way. In the SQP-type frameworks central to this thesis, each iteration can be divided in two parts: a setup phase, where the quadratic problem data is generated, and a solution phase, where QP (3.1) is solved. We analyze in the following, how each phase can benefit from the existing sparsity patterns. Section 3.1 covers the setup phase, while Section 3.2 presents a new, computationally more efficient algorithm for the classical problem dimension reduction technique commonly referred to as *condensing* [BP84, Lei99]. In Section 3.3, we adapt this algorithm to suit partial QP data updates in the spirit of Section 2.4.1. Additionally, we present a whole new sparse QP algorithm in Chapter 4.

On a side note we remark that also direct collocation methods lead to structurally similar sparsity patterns to the ones seen above. In particular, the utilization of an SQP framework with a block-structured QP solution method such as the one to be presented in Chapter 4 is conceivable. Care must be taken however in the collocation context to exploit the specific internal sparsity patterns of the matrix blocks $\boldsymbol{H_k}$, $\boldsymbol{C_k}$, $\boldsymbol{E_k}$, and $\boldsymbol{D_k}$, which become slightly more pronounced in this context, cf., e.g., [SBS98, Bie10, And13].

Acknowledgement Section 3.2 of this chapter is partly based on the paper "A Condensing Algorithm for Nonlinear MPC with a Quadratic Runtime in Horizon Length" by Joel Andersson, Janick Frasch, Milan Vukov, and Moritz Diehl [AFVD13]. Joel Andersson contributed the central idea that lead to the lower complexity condensing algorithm. Janick Frasch helped formalize the algorithmic idea, while Milan Vukov contributed a prototypic implementation for numerical testing (not part of this thesis). Moritz Diehl served as a vital discussion partner. This thesis features a revised and more detailed description of the algorithm, and gives details to its applicability in the real-time context.

The algorithmic extensions for Mixed-Level Iterations Schemes in Section 3.3 are new, previously unpublished results, but we acknowledge that a similar analysis of the classical condensing algorithm has been performed for the paper "Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control" by Janick Frasch, Leonard Wirsching, Sebastian Sager, and Hans Georg Bock [FWSB12]. For that publication, the analysis has been carried out jointly by Janick Frasch and Leonard Wirsching.

3.1 IVP Solution and Sensitivity Generation

By definition of the SQP method in Section 1.4.3, the data of QP (3.1) is obtained by (possibly inexact) linearization of the NLP constraint functions and a (possibly inexact) quadratic approximation of the NLP Lagrangian. In case of the dynamic optimization problems (MS-NLP), (MPC), and (MHE) however, the original NLP data is not available in closed form, but rather depends on the solution of IVPs in the form of

$$\boldsymbol{x}(t_0) = \boldsymbol{s}\,; \qquad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{p}) \qquad \forall\, t \in \mathcal{T}, \tag{3.2}$$

cf. Equation (1.4). Here, we subsume all parametric dependencies of the IVP (3.2), i.e., the control discretization variables as well as the actual system parameters, into one vector $\boldsymbol{p}$ for notational convenience.

Only for ODE systems of special structure a compact closed-form analytic solution representation is available. Therefore one relies on *numerical integration routines* to compute an approximate solution for the state trajectories $\boldsymbol{x}(t; \boldsymbol{s}, \boldsymbol{p})$, $t \in \mathcal{T}$ with given initial value $\boldsymbol{s}$, and parametric dependency $\boldsymbol{p}$. Depending on the chosen method, the solution may be a continuous representation of the (approximate) state trajectory, or simply the (approximate) trajectory value at some desired point t_j, $j \in \mathbb{N}$. Among numerical integration routines one typically distinguishes between *explicit methods*, such as the popular family of *explicit Runge-Kutta methods*, and *implicit methods*, such as *collocation integration methods* or *backwards differentiation formulae (BDF)*. Explicit methods are typically employed for non-stiff[1] ODE systems, while implicit methods often have a more desirable region of absolute stability and hence are better suited for stiff ODE systems. We omit further details here, and refer the reader to [SB08, Alb10b] and the references therein for a more thorough analysis.

As mentioned in Section 1.2.1, both adaptive and fixed-stepsize integration schemes are being used in the context of real-time estimation and control. The essential difference is that the latter class of methods fixes the integrator discretization grid beforehand, e.g., by user specifications and/or semi-automatic heuristics, while the former class uses error estimates (typically derived from a comparison with an increased-order method) to refine the initially coarse integrator discretization grid where needed. We refer to [SB08, Alb10b] for details, but note that the adaptivity of the integrator discretization grid has severe consequences when it comes to the calculation of derivatives, as we will see further below.

The evaluation of the integrals of the DOP objective can be performed alongside the trajectory integration, cf. [Lei95, Lei99]. Besides function evaluations, the data setup of QP (3.1) requires the computation of sensitivity information of the IVP solution with respect to the system's initial value as well as its parametric dependency in form of the *variational trajectories*

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{s}}(t; \boldsymbol{s}, \boldsymbol{p}) \qquad \text{and} \qquad \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}(t; \boldsymbol{s}, \boldsymbol{p})$$

of System (3.2). Also, second-order derivatives may be required, if an exact Hessian approximation is chosen, cf. Section 1.4.4.

Clearly, sensitivity information can be computed by applying the method of finite differences to the IVP solutions $\boldsymbol{x}(t; \boldsymbol{s}, \boldsymbol{p})$. It is well-known, however, that this

[1] There is no generally agreed upon unique system-based characterization of stiffness, but a system linearization featuring both fast and slow modes, i.e., a large ratio of the smallest and largest negative Eigenvalue, is often a good indicator for a stiff system, see also [HNW96].

approach, which is sometimes referred to as *external numerical differentiation (END)*[2], is generally inferior to the so-called approach of *internal numerical differentiation (IND)*[3], where one solves perturbed IVPs alongside the nominal IVP using an identical integrator discretization grid, cf. [Boc81]. In fact, one can in general expect the accuracy of derivatives computed by END to be at most in the order of the square-root of the accuracy of the IVP solution, while the (local) approximation error of derivatives computed by the IND approach initially improves at the same rate as the (local) approximation error of the original IVP solution. Even with a frozen integrator discretization grid, a finite differences derivative approximation still suffers from cancellation errors, which limit the accuracy of the obtained derivatives to roughly the square-root of machine precision, cf. Section 1.4.5. To overcome these shortcomings, one often formulates the IVP sensitivities as *variational differential equations*

$$\begin{aligned} \boldsymbol{x}(t_0) &= \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{d}} \\ \frac{\partial}{\partial \boldsymbol{t}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{d}}(t) &= \frac{\partial \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{x}(\boldsymbol{t}), \boldsymbol{p})}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{d}}(t) + \frac{\partial \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{x}(\boldsymbol{t}), \boldsymbol{p})}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{d}} \qquad \forall\, t \in \mathcal{T}, \end{aligned}$$

which are derived by differentiating IVP (3.2) with respect to a direction $\boldsymbol{d}$ and interchanging the differentiation order on the left-hand side of the differential equation. If the variational differential equations are solved alongside the nominal IVP (3.2) using the same integrator discretization grid, the principle of IND is again satisfied and the accuracy of the sensitivity information is only limited by the accuracy of the IVP solver whenever exact derivatives of the system model $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ and $\boldsymbol{p}$ are available, e.g., through algorithmic differentiation. We refer to [MP96, Alb10b, Qui12, And13] and the references therein for a more thorough discussion. There, also the computation of adjoint sensitivities is covered, and insights for a white-box integration of AD and IVP solver can be found.

It is important to observe that, while it is advantageous to compute trajectories and sensitivities alongside, these computations can be performed independently from one another over all discretization stages $k \in \mathcal{S}$. In particular, we can execute fixed-stepsize integrators fully parallel on N computational nodes with an efficiency of 1 if an identical number of integrator steps is used on each stage. For adaptive integrators it is straightforward to construct worst-case counterexamples, where an arbitrarily large number of integrator steps is

[2]The name *external* numerical differentiation stems from the fact that the finite difference approximation is taken *outside* the integrator discretization grid.

[3]The name *internal* numerical differentiation refers to the fact that freezing the integrator grid can be seen as differentiating the components of the integration algorithm itself with respect to the initial perturbations.

required on one stage, while the required accuracy on all other stages is satisfied already with one step of the respective method.

Nonetheless, by sensibly choosing the discretization grid, the resulting stage IVPs can become quite linear (i.e., require few integrator steps per stage problem) and a very high parallel efficiency can be obtained. This comes, of course, at the price of larger NLPs and ultimately QPs, which then become the computational bottleneck. Motivated by this, we focus on the structure-exploiting solution of the resulting QPs and parallel solution methods therefor in the remainder of this part.

3.2 Dimension Reduction by Condensing

The main idea of condensing, which was established in the context of direct multiple shooting mainly through [BP84], is to use the stage coupling (equality) constraints (3.1b) of the dynamic optimization QP subproblem for an elimination of the dependent state variables from the vector of optimization variables. This leads to dense QPs of a significantly reduced dimensionality if the number of states n_x is rather large compared to the horizon length N and the dimensionality of control parameterization n_u.

Details of the original condensing algorithm from [BP84] can be found, e.g., in [Lei95, Lei99]. The computational complexity of this established algorithm is $\mathcal{O}(N^3\, n_x\, n_u^2 + N^2\, n_x^2\, n_u)$ and therefore in particular cubic in the horizon length (cf. [BNS95, Lei95, Lei99, CWTB00, LBS+03, DFH09, FWSB12, KWBS12, VDF+13]). Only very recently a Riccati recursion point of view (adopted in [FJ13], based on observations in [AM12]), and, independently, an algorithmic differentiation point of view ([AFVD13, And13], inspired by [AD10b]) lead to a faster condensing algorithm with a quadratic runtime in horizon length. We base our presentation on [AFVD13] in the following.

Recalling that the stage variable vectors are given by $\boldsymbol{z_k} = (\boldsymbol{x_k}, \boldsymbol{u_k})$ for each $k \in \mathcal{S}_N$, and $\boldsymbol{z_N} = \boldsymbol{x_N}$, we can regroup the vector of optimization variables into $\bar{\boldsymbol{x}} := (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N})$, $\boldsymbol{u} := (\boldsymbol{u_0}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_N})$, and $\boldsymbol{x_0}$. Accordingly, (3.1b) can equivalently be written as

$$\boldsymbol{A}\,\bar{\boldsymbol{x}} + \boldsymbol{B}\,\boldsymbol{u} + \boldsymbol{c} + \boldsymbol{L}\,\boldsymbol{x_0} = 0,$$

where

$$\boldsymbol{A} = \begin{bmatrix} -\mathbf{I} & & & \\ \boldsymbol{A_1} & -\mathbf{I} & & \\ & \ddots & \ddots & \\ & & \boldsymbol{A_{N-1}} & -\mathbf{I} \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_0} & & & \\ & \boldsymbol{B_1} & & \\ & & \ddots & \\ & & & \boldsymbol{B_{N-1}} \end{bmatrix},$$

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{A_0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \qquad \text{and} \qquad \tilde{\boldsymbol{c}} = \begin{bmatrix} \boldsymbol{c_0} \\ \boldsymbol{c_1} \\ \vdots \\ \boldsymbol{c_{N-1}} \end{bmatrix}.$$

We have

$$\boldsymbol{A_k} = \frac{\partial \boldsymbol{F_k}}{\partial \boldsymbol{s_k}}(\boldsymbol{s_k}, \boldsymbol{\nu_k})$$

and

$$\boldsymbol{B_k} = \frac{\partial \boldsymbol{F_k}}{\partial \boldsymbol{\nu_k}}(\boldsymbol{s_k}, \boldsymbol{\nu_k}),$$

where the linearization points $\boldsymbol{s_k}$ and $\boldsymbol{\nu_k}$ depend on the specific SQP-type method. Based on this separation of variables, we first rewrite the objective (3.1a) as

$$\frac{1}{2}\left(\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{S}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{S}^\top \boldsymbol{u} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}\right) + \boldsymbol{q}^\top \boldsymbol{x} + \boldsymbol{r}^\top \boldsymbol{u},$$

where $\boldsymbol{x} = (\boldsymbol{x_0}, \bar{\boldsymbol{x}})$. The block matrices $\boldsymbol{Q} \in \mathbb{R}^{(N+1)n_x \times (N+1)n_x}$, $\boldsymbol{S} \in \mathbb{R}^{Nn_u \times (N+1)n_x}$, and $\boldsymbol{R} \in \mathbb{R}^{Nn_u \times Nn_u}$ are given by

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q_0} & & & \\ & \boldsymbol{Q_1} & & \\ & & \ddots & \\ & & & \boldsymbol{Q_N} \end{bmatrix}, \qquad \boldsymbol{S} = \begin{bmatrix} \boldsymbol{S_0} & & & \\ & \boldsymbol{S_1} & & \\ & & \ddots & \\ & & \boldsymbol{S_{N-1}} & \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_0} & & & \\ & \boldsymbol{R_1} & & \\ & & \ddots & \\ & & & \boldsymbol{R_{N-1}} \end{bmatrix}.$$

The stage constraints (3.1c) accordingly become

$$\boldsymbol{D^x}\boldsymbol{x} + \boldsymbol{D^u}\boldsymbol{u} \leq \boldsymbol{d},$$

where

$$\boldsymbol{D^x} = \begin{bmatrix} \boldsymbol{D_0^x} & & & \\ & \boldsymbol{D_1^x} & & \\ & & \ddots & \\ & & & \boldsymbol{D_N^x} \end{bmatrix} \quad \text{and} \quad \boldsymbol{D^u} = \begin{bmatrix} \boldsymbol{D_0^u} & & & \\ & \boldsymbol{D_1^u} & & \\ & & \ddots & \\ & & & \boldsymbol{D_{N-1}^u} \\ & & & \boldsymbol{0} \end{bmatrix}.$$

Since, by inspection, $\boldsymbol{A}$ is invertible, we can use

$$\bar{\boldsymbol{x}} = -\boldsymbol{A}^{-1}\left(\boldsymbol{B}\boldsymbol{u} + \boldsymbol{c} + \boldsymbol{L}\,\boldsymbol{x_0}\right)$$

to eliminate the dependent state vector from the QP optimization variables. We introduce the *transition matrix* $\boldsymbol{G} \in \mathbb{R}^{(N+1)n_x \times N n_u}$ and the *transition vector* $\boldsymbol{g} \in \mathbb{R}^{(N+1)n_x}$, as well as the condensed embedding of $\boldsymbol{x_0}$, denoted by $\boldsymbol{G^e} \in \mathbb{R}^{(N+1)n_x \times n_x}$:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{0} & & & \\ \boldsymbol{G_{1,0}} & & & \\ \boldsymbol{G_{2,0}} & \boldsymbol{G_{2,1}} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{G_{N,0}} & \boldsymbol{G_{N,1}} & \cdots & \boldsymbol{G_{N,N-1}} \end{bmatrix}, \; \boldsymbol{G^e} = \begin{bmatrix} \mathbf{I} \\ \boldsymbol{G_1^e} \\ \boldsymbol{G_2^e} \\ \vdots \\ \boldsymbol{G_N^e} \end{bmatrix}, \text{ and } \boldsymbol{g} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \\ \vdots \\ \boldsymbol{g_N} \end{bmatrix},$$

where the nonzero blocks from $\boldsymbol{G}$ and $\boldsymbol{g}$ are given by $-\boldsymbol{A}^{-1}\boldsymbol{B}$ and $-\boldsymbol{A}^{-1}\boldsymbol{c}$, respectively, and the non-trivial blocks of $\boldsymbol{G^e}$ are given by $-\boldsymbol{A}^{-1}\boldsymbol{L}$.

We then yield a *condensed QP*[4]

$$\min_{\boldsymbol{u},(\boldsymbol{x_0})} \quad \frac{1}{2}\boldsymbol{u}^\top \bar{\boldsymbol{R}}\,\boldsymbol{u} + \bar{\boldsymbol{r}}^\top \boldsymbol{u} + e(\boldsymbol{x_0}, \boldsymbol{u}) \tag{3.3a}$$

$$\text{s.t.} \quad \bar{\boldsymbol{D}}\,\boldsymbol{u} + \bar{\boldsymbol{D}}^{\boldsymbol{e}}\boldsymbol{x_0} \leq \bar{\boldsymbol{d}}, \tag{3.3b}$$

where

$$\bar{\boldsymbol{r}} := \boldsymbol{r} + \boldsymbol{G}^\top(\boldsymbol{q} + \boldsymbol{Q}\,\boldsymbol{g}) + \boldsymbol{S}\,\boldsymbol{g}\,,$$

$$\bar{\boldsymbol{R}} := \boldsymbol{R} + \boldsymbol{G}^\top \boldsymbol{Q}\,\boldsymbol{G} + \boldsymbol{S}\,\boldsymbol{G} + \boldsymbol{G}^\top \boldsymbol{S}^\top\,,$$

$$\bar{\boldsymbol{D}} := \boldsymbol{D^u} + \boldsymbol{D^x}\,\boldsymbol{G}\,,$$

$$\bar{\boldsymbol{d}} := \boldsymbol{d} - \boldsymbol{D^x}\,\boldsymbol{g}\,.$$

[4]Whether $\boldsymbol{x_0}$ is an optimization variable or not depends on the specific problem; essentially it is one for MHE, but in MPC it is often fully determined.

The initial value embedding is given by

$$\bar{\boldsymbol{D}}^{\boldsymbol{e}} := \boldsymbol{D}^{\boldsymbol{x}}\, \boldsymbol{G}^{\boldsymbol{e}}$$

and

$$e(\boldsymbol{x_0}, \boldsymbol{u}) := \frac{1}{2}\, \boldsymbol{x_0}\, \bar{\boldsymbol{Q}}^{\boldsymbol{e}}\, \boldsymbol{x_0} + \boldsymbol{u}^\top \bar{\boldsymbol{S}}^{\boldsymbol{e}}\, \boldsymbol{x_0} + \bar{\boldsymbol{q}}^{\boldsymbol{e}\top} \boldsymbol{x_0}\,, \tag{3.4}$$

where

$$\bar{\boldsymbol{Q}}^{\boldsymbol{e}} := \boldsymbol{G}^{\boldsymbol{e}\top} \boldsymbol{Q}\, \boldsymbol{G}^{\boldsymbol{e}}$$

$$\bar{\boldsymbol{S}}^{\boldsymbol{e}} := \boldsymbol{G}^{\top} \boldsymbol{Q}\, \boldsymbol{G}^{\boldsymbol{e}} + \boldsymbol{S}\, \boldsymbol{G}^{\boldsymbol{e}}$$

$$\bar{\boldsymbol{q}}^{\boldsymbol{e}} := \boldsymbol{G}^{\boldsymbol{e}\top} \boldsymbol{Q}\, \boldsymbol{g} + \boldsymbol{G}^{\boldsymbol{e}\top} \boldsymbol{q}\,.$$

We initially consider the MPC case with fully determined initial value, where the initial value embedding $e(\boldsymbol{x_0}, \boldsymbol{u})$ reduces to

$$e(\boldsymbol{x_0}, \boldsymbol{u}) \equiv \boldsymbol{u}^\top \bar{\boldsymbol{S}}^{\boldsymbol{e}}\, \boldsymbol{x_0}\,,$$

as the remaining terms are constant. Note that here only $\boldsymbol{u}$ are optimization variables, while $\boldsymbol{x_0}$ is a fixed parameter after the initial value embedding.

The essential idea of the N^2 complexity condensing algorithm is to compute $\boldsymbol{G}^\top \boldsymbol{Q}\, \boldsymbol{G}$ exploiting the banded structure of $\boldsymbol{A}$ and $\boldsymbol{B}$ separately. The computational order can be seen as $\boldsymbol{B}^\top(-\boldsymbol{A}^{-\top}(\boldsymbol{\Pi}\, \boldsymbol{Q}\, \boldsymbol{G}))$, rather than the order $\boldsymbol{G}^\top(\boldsymbol{Q}\, \boldsymbol{G})$ from the classical condensing algorithm. Here,

$$\boldsymbol{\Pi} := \begin{bmatrix} \mathbf{0} & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{N n_x \times (N+1) n_x}$$

is only needed to formally eliminate the first block row of zeros from $\boldsymbol{Q}\, \boldsymbol{G}$ for a dimension-consistent multiplication. In an analogous fashion, the computational complexity for obtaining $\bar{\boldsymbol{r}}$ can be reduced by one order of magnitude in N.

In detail, we suggest the following algorithms for computing $\bar{\boldsymbol{r}}$, $\bar{\boldsymbol{R}}$, $\bar{\boldsymbol{D}}$, and $\bar{\boldsymbol{d}}$.

Algorithm 3.1: Computation of nonzero blocks of $\boldsymbol{g}$

$\boldsymbol{g_1} := \boldsymbol{c_0}$
for $i = 1 : N-1$ **do**
  $\boldsymbol{g_{i+1}} := \boldsymbol{c_i} + \boldsymbol{A_i}\,\boldsymbol{g_i}$

Algorithm 3.2: Computation of nonzero blocks of $\boldsymbol{G}$

for $j = 0 : N-1$ **do**
  $\boldsymbol{G_{j+1,j}} := \boldsymbol{B_j}$
  for $i = j+1 : N-1$ **do**
    $\boldsymbol{G_{i+1,j}} := \boldsymbol{A_i}\,\boldsymbol{G_{i,j}}$

Algorithm 3.3: Blockwise computation of $\bar{\boldsymbol{r}}$

$\tilde{\boldsymbol{w}} := \boldsymbol{q_N} + \boldsymbol{Q_N}\,\boldsymbol{g_N}$
for $i = N-1 : 1$ **do**
  $\bar{\boldsymbol{r}}_{\boldsymbol{i}} := \boldsymbol{r_i} + \boldsymbol{B_i}^\top \tilde{\boldsymbol{w}} + \boldsymbol{S_i}\,\boldsymbol{g_i}$
  $\tilde{\boldsymbol{w}} := \boldsymbol{A_i}^\top\,\tilde{\boldsymbol{w}} + (\boldsymbol{q_i} + \boldsymbol{Q_i}\,\boldsymbol{g_i})$
$\bar{\boldsymbol{r}}_{\boldsymbol{0}} := \boldsymbol{r_0} + \boldsymbol{B_0}^\top \tilde{\boldsymbol{w}}$

3.2.1 Condensed QP objective

First, the nonzero blocks of the transition matrix $\boldsymbol{G}$ and the transition vector $\boldsymbol{g}$ are computed by block-forward substitution from Algorithms 3.1 and 3.2. The computational costs can be seen from Table 3.1 in Section 3.2.6[5].

In computing $\bar{\boldsymbol{r}}$ and $\bar{\boldsymbol{R}}$, we make use of the intermediate results $\boldsymbol{w} := -\boldsymbol{A}^{-\top}(\boldsymbol{q} + \boldsymbol{Q}\,\boldsymbol{g})$ and $\boldsymbol{W} := -\boldsymbol{A}^{-\top}(\boldsymbol{Q}\,\boldsymbol{G})$, respectively. Note, however, that these results do not need to be stored completely, but only one vector block $\tilde{\boldsymbol{w}} \in \mathbb{R}^{n_x}$ and one matrix block $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n_x \times n_u}$ of memory are sufficient in a sequential implementation[6].

Both, $\boldsymbol{w}$ and $\boldsymbol{W}$ are computed by blockwise backwards substitution in Algorithms 3.3 and 3.4, respectively. The desired QP data, $\bar{\boldsymbol{r}}$ and $\bar{\boldsymbol{R}}$, is computed alongside by a premultiplication with the corresponding $\boldsymbol{B}$. block and an addition of the corresponding $\boldsymbol{R}$. ($\boldsymbol{r}$. in case of $\bar{\boldsymbol{r}}$) and $\boldsymbol{S}$. components, where applicable.

[5] We note, also for future reference, that we only count additions and multiplications or copy operations in this context.

[6] In a parallel implementation, this becomes one vector and one matrix block each per computational thread.

Algorithm 3.4: Blockwise computation of $\bar{\boldsymbol{R}}$

```
for j = 0 : N − 1 do
    W̃ := Q_N G_{N,j}
    for i = N − 1 : j + 1 do
        R̄_{i,j} := B_i^⊤ W̃ + S_i G_{i,j}
        W̃ := A_i^⊤ W̃ + Q_i G_{i,j}
    R̄_{j,j} := R_{j,j} + B_j^⊤ W̃
```

Algorithm 3.5: Blockwise computation of $\bar{\boldsymbol{D}}$

```
D̄_{0,0} := D_0^u
for i = 1 : N − 1 do
    for j = 0 : i − 1 do
        D̄_{i,j} := D_i^x G_{i,j}
    D̄_{i,i} := D_i^u + D_i^x G_{i,j}
for j = 0 : N − 1 do
    D̄_{N,j} := D_N^x G_{N,j}
```

3.2.2 Condensing of affine stage constraints

The condensed matrix of path constraints $\bar{\boldsymbol{D}}$ can be computed in a straightforward manner as given by Algorithm 3.5. Analogously, $\bar{\boldsymbol{d}}$ can be computed as suggested by Algorithm 3.6.

3.2.3 Initial value embedding in MPC

Algorithm 3.3 only computes the "offline" part of $\bar{\boldsymbol{r}}$ (essentially assuming $\boldsymbol{x_0} = \boldsymbol{0}$). Once the true value for $\boldsymbol{x_0}$ is known, the result from Algorithm 3.3 needs to be corrected by $\bar{\boldsymbol{r}}^{\boldsymbol{e}} := \bar{\boldsymbol{S}}^{\boldsymbol{e}} \boldsymbol{x_0} = (\boldsymbol{G}^\top \boldsymbol{Q}\, \boldsymbol{G}^{\boldsymbol{e}} + \boldsymbol{S}\, \boldsymbol{G}^{\boldsymbol{e}})\, \boldsymbol{x_0}$.

Depending on whether the focus lies on an algorithm with minimal feedback delay or on a minimal overall computational effort, we propose to use either Algorithms 3.7 and 3.8 or Algorithms 3.9 and 3.10. Algorithm 3.8 precomputes an Nn_u by n_x matrix $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$, based on the influence matrix of $\boldsymbol{x_0}$, $\boldsymbol{G}^{\boldsymbol{e}}$, which has to be computed before, e.g., by Algorithm 3.7. The overall computational cost

Algorithm 3.6: Blockwise computation of $\bar{\boldsymbol{d}}$

$\bar{\boldsymbol{d}}_{\boldsymbol{0}} := \boldsymbol{d}_{\boldsymbol{0}}$
for $i = 1 : N$ **do**
  $\bar{\boldsymbol{d}}_{\boldsymbol{i}} := \boldsymbol{d}_{\boldsymbol{i}} - \boldsymbol{D}_{\boldsymbol{i}}^{\boldsymbol{x}}\, \boldsymbol{g}_{\boldsymbol{i}}$

Algorithm 3.7: Blockwise computation of $\boldsymbol{G}^{\boldsymbol{e}}$

$\boldsymbol{G}_{\boldsymbol{0}}^{\boldsymbol{e}} := \mathbf{I}$
for $i = 0 : N - 1$ **do**
  $\boldsymbol{G}_{\boldsymbol{i+1}}^{\boldsymbol{e}} := \boldsymbol{A}_{\boldsymbol{i}}\, \boldsymbol{G}_{\boldsymbol{i}}^{\boldsymbol{e}}$

of Algorithms 3.7 and 3.8 is

$$n_x + n_x^2 + (N-1) \cdot 2\, n_x^3 + 2\, n_x^3 + (N-1) \cdot (4\, n_x^2\, n_u + 4\, n_x^3) + 2\, n_x^2\, n_u$$

$$= (6\, N - 4)\, n_x^3 + (4\, N - 2)\, n_x^2\, n_u + n_x^2 + n_x$$

FLOPs. In an online context, these computations can, however, be performed in the preparation phase before $\boldsymbol{x}_{\boldsymbol{0}}$ is known. The cost of the operations in the feedback phase then reduces to $N\, n_u\, n_x$ FLOPs for the addition of the matrix-vector product $\bar{\boldsymbol{r}}^{\boldsymbol{e}} = \bar{\boldsymbol{S}}^{\boldsymbol{e}} \boldsymbol{x}_{\boldsymbol{0}}$ to the precomputed linear objective term $\bar{\boldsymbol{r}}$.

Algorithm 3.10 directly computes $\bar{\boldsymbol{r}}^{\boldsymbol{e}}$ as $(\boldsymbol{G}^\top \boldsymbol{Q} + \boldsymbol{S})\, \boldsymbol{v}^{\boldsymbol{e}}$ based on the propagation of $\boldsymbol{x}_{\boldsymbol{0}}$, $\boldsymbol{v}^{\boldsymbol{e}} := \boldsymbol{G}^{\boldsymbol{e}} \boldsymbol{x}_{\boldsymbol{0}}$, which is computed by Algorithm 3.9. The total computational costs of Algorithms 3.9 and 3.10 add up to

$$N \cdot 2\, n_x^2 + n_x + 2\, n_x^2 + (N-1) \cdot (4\, n_x\, n_u + 4\, n_x^2) + 4\, n_x\, n_u$$

$$= 4\, N\, n_x\, n_u + (6\, N - 2)\, n_x^2 + n_x$$

FLOPs. Here, however, all operations can only be performed after $\boldsymbol{x}_{\boldsymbol{0}}$ is known, therefore slightly increasing the feedback delay at the gain of reduced overall computational costs.

Like for the objective terms, two possibilities exist for embedding the initial value into the path constraints:

Algorithm 3.11 directly computes $\bar{\boldsymbol{d}}^{\boldsymbol{e}} := \bar{\boldsymbol{D}}^{\boldsymbol{e}} \boldsymbol{x}_{\boldsymbol{0}}$ at an overall computational cost of $2\,(N+1)\, n_x\, n_d$ FLOPS. However, Algorithm 3.11 requires the execution of Algorithm 3.9 during the feedback phase, which comes at the cost of $N \cdot 2\, n_x^2 + n_x$ FLOPs, as outlined above.

Algorithm 3.12 on the other hand prepares the embedding $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$ at the cost of $n_x\, n_d + 2\, N\, n_x^2\, n_d$ FLOPS, but can directly be executed during the preparation

Algorithm 3.8: Blockwise computation of $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$

$\tilde{\boldsymbol{W}} := \boldsymbol{Q_N}\,\boldsymbol{G_N^e}$
for $i = N-1:1$ **do**
  $\bar{\boldsymbol{S}}_{\boldsymbol{i}}^{\boldsymbol{e}} := \boldsymbol{B_i}^{\top}\tilde{\boldsymbol{W}} + \boldsymbol{S_i}\,\boldsymbol{G_i^e}$
  $\tilde{\boldsymbol{W}} := \boldsymbol{A_i}^{\top}\tilde{\boldsymbol{W}} + \boldsymbol{Q_i}\,\boldsymbol{G_i^e}$
$\bar{\boldsymbol{S}}_{\boldsymbol{0}}^{\boldsymbol{e}} := \boldsymbol{B_0}^{\top}\tilde{\boldsymbol{W}} + \boldsymbol{S_0}$

Algorithm 3.9: Blockwise computation of $\boldsymbol{v}^{\boldsymbol{e}}$

$\boldsymbol{v_0^e} := \boldsymbol{x_0}$
for $i = 0:N-1$ **do**
  $\boldsymbol{v_{i+1}^e} := \boldsymbol{A_i}\,\boldsymbol{v_i^e}$

phase. The computational cost of the subtraction $\bar{\boldsymbol{d}} - \bar{\boldsymbol{d}}^{\boldsymbol{e}} = \bar{\boldsymbol{d}} - \bar{\boldsymbol{D}}^{\boldsymbol{e}}\boldsymbol{x_0}$ during the feedback phase again sum up to $2\,(N+1)\,n_x\,n_d$ FLOPs. Note that Algorithm 3.11 requires a previous execution of Algorithm 3.9, while Algorithm 3.12 requires a previous execution of Algorithm 3.7.

3.2.4 Online measurement embedding in MHE

If the dynamic real-time optimization problem at hand is of MHE origin, $\boldsymbol{x_0}$ remains an optimization variable. Accordingly, Algorithms 3.8 and 3.12 need to be employed to compute $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$ and $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$ explicitly (both relying on a precomputed $\boldsymbol{G^e}$ by Algorithm 3.7). Furthermore, the $\boldsymbol{u}$-free terms in (3.4) are no longer negligible for the condensed QP (3.3). We therefore have Algorithms 3.13 and 3.14 to compute $\tilde{\boldsymbol{q}}^{\boldsymbol{e}} = \boldsymbol{g}^{\top}\boldsymbol{Q}\,\boldsymbol{G^e} + \boldsymbol{q}^{\top}\boldsymbol{G^e}$ and $\bar{\boldsymbol{Q}}^{\boldsymbol{e}} = \boldsymbol{G^e}^{\top}\boldsymbol{Q}\,\boldsymbol{G^e}$. Note that $\boldsymbol{G_N^e}^{\top}\boldsymbol{q_N}$ is omitted in Algorithm 3.13 on purpose, since it is not yet determined during the preparation phase.

It is easy to verify that the final measurement $\boldsymbol{y_N}$, which is still unknown during the preparation phase, only enters in the last block component of the linear QP objective term, $\boldsymbol{q_N}$ (cf., e.g., [KDK+11a]). Therefore, the result of Algorithm 3.13 (which is only correct if the new measurement $\boldsymbol{y_N}$ is identical with the model prediction, i.e., if $\boldsymbol{q_N} = \boldsymbol{0}$) needs to be corrected during the feedback phase by $\bar{\boldsymbol{q}}^{\boldsymbol{e}} := \tilde{\boldsymbol{q}}^{\boldsymbol{e}} + \boldsymbol{G_N^e}^{\top}\boldsymbol{q_N}$ at the computational cost of $2\,n_x^2$. Furthermore, Algorithm 3.3 as well needs to be performed in the feedback phase, as all its computations are dependent on $\boldsymbol{q_N}$. The overall computational costs of

Algorithm 3.10: Blockwise direct computation of $\bar{\boldsymbol{r}}^{\boldsymbol{e}}$

$\tilde{\boldsymbol{w}} := \boldsymbol{Q_N}\, \boldsymbol{v_N^e}$
for $i = N-1:1$ **do**
  $\bar{\boldsymbol{r}}_{\boldsymbol{i}}^{\boldsymbol{e}} := \boldsymbol{B_i}^\top \tilde{\boldsymbol{w}} + \boldsymbol{S_i}\, \boldsymbol{v_i^e}$
  $\tilde{\boldsymbol{w}} := \boldsymbol{A_i}^\top \tilde{\boldsymbol{w}} + \boldsymbol{Q_i}\, \boldsymbol{v_i^e}$
$\bar{\boldsymbol{r}}_{\boldsymbol{0}}^{\boldsymbol{e}} := \boldsymbol{B_0}^\top \tilde{\boldsymbol{w}} + \boldsymbol{S_0} \boldsymbol{v_0^e}$

Algorithm 3.11: Blockwise direct computation of $\bar{\boldsymbol{d}}^{\boldsymbol{e}}$

for $i = 0:N$ **do**
  $\bar{\boldsymbol{d}}_{\boldsymbol{i}}^{\boldsymbol{e}} := \boldsymbol{D_i^x}\, \boldsymbol{v_i^e}$

Algorithm 3.12: Blockwise computation of $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$

$\bar{\boldsymbol{D}}_{\boldsymbol{0}}^{\boldsymbol{e}} := \boldsymbol{D_0^x}$
for $i = 1:N$ **do**
  $\bar{\boldsymbol{D}}_{\boldsymbol{i}}^{\boldsymbol{e}} := \boldsymbol{D_i^x}\, \boldsymbol{G_i^e}$

Algorithm 3.3 sum up to

$$2\,n_x^2 + (N-1)\cdot(4\,n_x\,n_u + 4\,n_x^2) + 2\,n_x\,n_u$$
$$= (4\,N-2)\cdot(n_x^2 + n_x\,n_u).$$

3.2.5 Recovery of the full-space primal-dual solution

Since the condensed QP (3.3) was obtained from (3.1) by propagation of the equality constraints (3.1b), both solutions are equivalent. In particular, the primal-dual solution of (3.3), $(\boldsymbol{u}^*, \boldsymbol{x_0^*}, \boldsymbol{\mu}^*)$ corresponds to the optimal state, initial value and stage constraint multipliers of (3.1)[7]. To recover the full state vector $\boldsymbol{x}^* \in \mathbb{R}^{(N+1)n_x}$, a simple forward simulation, as given by Algorithm 3.15, is sufficient. To recover the multipliers of the coupling constraints $\boldsymbol{\lambda}^* := (\boldsymbol{\lambda_1^*}, \boldsymbol{\lambda_2^*}, \ldots, \boldsymbol{\lambda_N^*}) \in \mathbb{R}^{Nn_x}$ we make use of the fact that the gradient of the Lagrangian of (3.1) is stationary in an optimal primal-dual solution

[7] W.l.o.g., we assume that $\boldsymbol{x_0^*}$ is a (possibly fixed) optimization variable here.

Algorithm 3.13: Additive computation of $\tilde{\boldsymbol{q}}^{\boldsymbol{e}}$

$\tilde{\boldsymbol{q}}^{\boldsymbol{e}} := \boldsymbol{q_0}$
for $i = 1 : N-1$ **do**
  $\tilde{\boldsymbol{q}}^{\boldsymbol{e}} := \tilde{\boldsymbol{q}}^{\boldsymbol{e}} + \boldsymbol{G_i^e}^\top(\boldsymbol{q_i} + \boldsymbol{Q_i}\,\boldsymbol{g_i})$
$\tilde{\boldsymbol{q}}^{\boldsymbol{e}} := \tilde{\boldsymbol{q}}^{\boldsymbol{e}} + \boldsymbol{G_N^e}^\top(\boldsymbol{Q_N}\,\boldsymbol{g_N})$

Algorithm 3.14: Additive computation of $\bar{\boldsymbol{Q}}^{\boldsymbol{e}}$

$\bar{\boldsymbol{Q}}^{\boldsymbol{e}} := \boldsymbol{Q_0}$
for $i = 1 : N$ **do**
  $\bar{\boldsymbol{Q}}^{\boldsymbol{e}} := \bar{\boldsymbol{Q}}^{\boldsymbol{e}} + \boldsymbol{G_i^e}^\top \boldsymbol{Q_i}\,\boldsymbol{G_i^e}$

$(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$. Since the Lagrangian of (3.1) is given by

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\left(\boldsymbol{x}^\top \boldsymbol{Q}\,\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{S}\,\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{S}^\top \boldsymbol{u} + \boldsymbol{u}^\top \boldsymbol{R}\,\boldsymbol{u}\right) + \boldsymbol{q}^\top \boldsymbol{x} + \boldsymbol{r}^\top \boldsymbol{u}$$
$$+ \boldsymbol{\lambda}^\top \left(\begin{bmatrix}\boldsymbol{L} & \boldsymbol{A}\end{bmatrix} \boldsymbol{x} + \boldsymbol{B}\,\boldsymbol{u} + \boldsymbol{c}\right) + \boldsymbol{\mu}^\top \left(\boldsymbol{D^x}\,\boldsymbol{x} + \boldsymbol{D^u}\,\boldsymbol{u} - \boldsymbol{d}\right),$$

we have

$$\boldsymbol{\nabla_x}\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \boldsymbol{Q}\,\boldsymbol{x}^* + 2\,\boldsymbol{S}^\top \boldsymbol{u}^* + \boldsymbol{q} + \begin{bmatrix}\boldsymbol{L} & \boldsymbol{A}\end{bmatrix}^\top \boldsymbol{\lambda}^* + \boldsymbol{D^x}^\top \boldsymbol{\mu}^* = \boldsymbol{0}. \quad (3.5)$$

The block-banded structure of $\boldsymbol{L}$ and $\boldsymbol{A}$ in Equation (3.5) can be used to obtain $\boldsymbol{\lambda}^*$ from the known vectors $\boldsymbol{x}^*$, $\boldsymbol{u}^*$, and $\boldsymbol{\mu}^*$ by the block-backwards substitution given in Algorithm 3.16[8].

3.2.6 Computational complexity analysis

The number of floating point operations required by each of the condensing algorithms detailed here can be seen from Table 3.1. As mentioned before, we only count additions, multiplications and copy operations. Particular structures, such as, e.g., multiplication with an identity matrix, are exploited. We assume an optimized order of operations, which means, for example, that an addition and a matrix-vector multiplication $\boldsymbol{c} := \boldsymbol{A}\,\boldsymbol{a} + \boldsymbol{b}$ are assumed to be performed jointly at the cost of $2\,n^2$ FLOPs, if all dimensions are n.

For clarity, we summarize the asembly choices of the individual algorithms for MPC and MHE in Table 3.2. We can conclude that in MPC the computational

[8] To be precise, only the last N block rows of (3.5) are used for recovering $\boldsymbol{\lambda}$.

Algorithm 3.15: Recovery of $\boldsymbol{x}^*$

```
for i = 1 : N do
    x*_i := A_{i-1} x*_{i-1} + B_{i-1} u*_{i-1} + c_{i-1}
```

Algorithm 3.16: Recovery of $\boldsymbol{\lambda}^*$

```
λ*_N := Q_N x*_N + q_N + D_N^{x⊤} μ*_N
for i = N − 1 : 1 do
    λ*_i := Q_i x*_i + S_i^⊤ u*_i + q_i + A_i^⊤ λ_{i+1} + D_i^{x⊤} μ*_i
```

complexity of the condensing procedure on a sequential computational architecture ranges between

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right) + N\left(n_x^3 + n_x^2\, n_d\right)\right)$$

and

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right)\right)$$

in the preparation phase, and between

$$\mathcal{O}\left(N\left(n_x\, n_u + n_x\, n_d\right)\right)$$

and

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_u + n_x\, n_d\right)\right)$$

in the feedback phase[9], depending on whether the feedback delay minimizing or the overall-effort minimizing implementation is chosen. In MHE, the computational complexity is

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right) + N\left(n_x^3 + n_x^2\, n_d\right)\right)$$

in the preparation phase and

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_u\right)\right)$$

in the feedback phase, due to the additional degree of freedom in the condensed QP.

On a parallel computational architecture with at least $\left\lceil \frac{N}{2} \right\rceil$ threads[10], the computational complexity of the preparation phase reduces to

$$\mathcal{O}\left(N\left(n_x^2 n_u + n_x n_u^2 + n_x n_u n_d\right)\right)$$

[9] We assume that the expansion step for recovering the full-space solution is performed during the subsequent preparation phase.

[10] Taking advantage of the block-triangular shape of the critical result matrices we specifically discuss the use of $\left\lceil \frac{N}{2} \right\rceil$ threads here for a high parallel efficiency, although $\mathcal{O}(N)$ threads would be a sufficient requirement to uphold our complexity results.

Algorithm	Result	Number of FLOPs
Algorithm 3.1	$\boldsymbol{g}$	$2\,(N-1)\,n_x^2 + n_x$
Algorithm 3.2	$\boldsymbol{G}$	$N^2\,n_x^2\,n_u - N\,n_x^2\,n_u + N\,n_x\,n_u$
Algorithm 3.3	$\bar{\boldsymbol{r}}$	$4\,N\,(n_x^2 + n_x\,n_u) - 2\,n_x^2 - 2\,n_x\,n_u$
Algorithm 3.4	$\bar{\boldsymbol{R}}$	$2\,N^2\,(n_x^2\,n_u + n_x\,n_u^2)$
Algorithm 3.5	$\bar{\boldsymbol{D}}$	$(N^2 + 3\,N - 2)\,n_x\,n_u\,n_d + n_u\,n_d$
Algorithm 3.6	$\bar{\boldsymbol{d}}$	$2\,N\,n_x\,n_d + n_d$
Algorithm 3.7	$\boldsymbol{G}^{\boldsymbol{e}}$	$2\,(N-1)\,n_x^3 + n_x^2 + n_x$
Algorithm 3.8	$\bar{\boldsymbol{S}}^{\boldsymbol{e}}$	$4\,N(n_x^3 + n_x^2\,n_u) - 2\,n_x^3 - 2\,n_x^2\,n_u$
Algorithm 3.9	$\boldsymbol{v}^{\boldsymbol{e}}$	$2\,N\,n_x^2 + n_x$
Algorithm 3.10	$\bar{\boldsymbol{r}}^{\boldsymbol{e}}$	$4\,N(n_x^2 + n_x\,n_u) - 2\,n_x^2$
Algorithm 3.11	$\bar{\boldsymbol{d}}^{\boldsymbol{e}}$	$2\,(N+1)\,n_x\,n_d$
Algorithm 3.12	$\bar{\boldsymbol{D}}^{\boldsymbol{e}}$	$2\,N\,n_x^2\,n_d + n_x\,n_d$
Algorithm 3.13	$\tilde{\boldsymbol{q}}^{\boldsymbol{e}}$	$4\,N\,n_x^2$
Algorithm 3.14	$\bar{\boldsymbol{Q}}^{\boldsymbol{e}}$	$4\,N\,n_x^3$
Algorithm 3.15	$\boldsymbol{x}^*$	$2\,N\,(n_x^2 + n_x\,n_u)$
Algorithm 3.16	$\boldsymbol{\lambda}^*$	$N\,(4\,n_x^2 + 2\,n_x\,n_u + 2\,n_x\,n_d) - 2\,(n_x^2 - n_x\,n_u)$

Table 3.1: Computational complexities (FLOPs) of the condensing algorithms.

in MPC when choosing the overall-effort minimizing implementation, and

$$\mathcal{O}\left(N\,(n_x^3 + n_x^2 n_u + n_x n_u^2 + n_x n_u n_d) + n_x^2 n_d\right)$$

in the feedback delay minimizing implementation or in MHE. The complexity of the operations of the feedback phase reduces to

$$\mathcal{O}\,(n_x n_u + n_x n_d)$$

in the feedback delay minimizing implementation of condensing in MPC, while it reduces only slightly to

$$\mathcal{O}\left(N\,(n_x^2 + n_x n_u) + n_x n_d\right)$$

in the overall-effort minimizing implementation. In MHE, the complexity of the condensing operations of the the feedback phase remains

$$\mathcal{O}\left(N\,(n_x^2 + n_x n_u)\right).$$

	MPC with minimal feedback delay	MPC with minimal overall effort	MHE
Preparation phase	Algorithm 3.1 Algorithm 3.2 Algorithm 3.3 Algorithm 3.4 Algorithm 3.5 Algorithm 3.6 Algorithm 3.7 Algorithm 3.8 Algorithm 3.12	Algorithm 3.1 Algorithm 3.2 Algorithm 3.3 Algorithm 3.4 Algorithm 3.5 Algorithm 3.6	Algorithm 3.1 Algorithm 3.2 Algorithm 3.4 Algorithm 3.5 Algorithm 3.6 Algorithm 3.7 Algorithm 3.8 Algorithm 3.12 Algorithm 3.13 Algorithm 3.14
Feedback phase	$\bar{\boldsymbol{r}} := \bar{\boldsymbol{r}} + \bar{\boldsymbol{S}}^{\boldsymbol{e}}\, \hat{\boldsymbol{x}}_{\boldsymbol{0}}$ $\bar{\boldsymbol{d}} := \bar{\boldsymbol{d}} - \bar{\boldsymbol{D}}^{\boldsymbol{e}}\, \hat{\boldsymbol{x}}_{\boldsymbol{0}}$	Algorithm 3.9 Algorithm 3.10 Algorithm 3.11 $\bar{\boldsymbol{r}} := \bar{\boldsymbol{r}} + \bar{\boldsymbol{r}}^{\boldsymbol{e}}$ $\bar{\boldsymbol{d}} := \bar{\boldsymbol{d}} - \bar{\boldsymbol{d}}^{\boldsymbol{e}}$	Algorithm 3.3 $\bar{\boldsymbol{q}}^{\boldsymbol{e}} := \tilde{\boldsymbol{q}}^{\boldsymbol{e}} + {\boldsymbol{G}_N^e}^{\top} \boldsymbol{q}_{\boldsymbol{N}}$
Expansion	Algorithm 3.15 Algorithm 3.16	Algorithm 3.15 Algorithm 3.16	Algorithm 3.15 Algorithm 3.16

Table 3.2: Condensing schemes for MPC and MHE.

3.3 Re-Condensing for Partial QP Data Updates

The whole condensing procedure, as detailed in Section 3.2, only needs to be executed if all QP data was updated by a re-linearization, i.e., during a standard RTI. In the following, we analyze the effect of partial and inexact QP data updates, as introduced in Chapter 2.4, on the condensing procedure. We are in particular interested in the consequences arising from "vector updates", like the feasibility-improving and the optimality improving updates, as well as in the consequences from updating only the data on the first N_{frac} stages (dubbed *Mixed-Level Iterations* in [FWSB12]).

It should be noted that obviously, if no re-linearization is performed, all QP data remains unchanged except for the parametric embedding ($\boldsymbol{x_0}$ in the MPC case, and $\boldsymbol{q_N}$ in the MHE case). Then, only the matrix-vector product $\bar{\boldsymbol{r}}^{\boldsymbol{e}} = \bar{\boldsymbol{S}}^{\boldsymbol{e}} \boldsymbol{x_0}$ needs to be recomputed to prepare the feedback-generating QP in MPC; in

MHE, only $\bar{\boldsymbol{r}}$ needs to be recomputed from Algorithm 3.3 and $\tilde{\boldsymbol{q}}^{\boldsymbol{e}}$ needs to be corrected by $\bar{\boldsymbol{q}}^{\boldsymbol{e}} = \tilde{\boldsymbol{q}}^{\boldsymbol{e}} + {\boldsymbol{G}_{\boldsymbol{N}}^{\boldsymbol{e}}}^{\top} \boldsymbol{q}_{\boldsymbol{N}}$.

3.3.1 Complete and partial inexact re-linearizations

If an optimality-improving update or an feasibility-improving update is performed on a stage $k \in \mathcal{S}$, only the vectors $\boldsymbol{q}_{\boldsymbol{k}}$, $\boldsymbol{r}_{\boldsymbol{k}}$, $\boldsymbol{c}_{\boldsymbol{k}}$, and $\boldsymbol{d}_{\boldsymbol{k}}$ change, while all QP matrix data (of the uncondensed problem) remains. Consequently, $\bar{\boldsymbol{R}}$, $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$, $\bar{\boldsymbol{D}}$, $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$, and $\bar{\boldsymbol{Q}}^{\boldsymbol{e}}$ (if appearing at all) in (3.3), as well as the auxiliary matrices $\boldsymbol{G}$ and $\boldsymbol{G}^{\boldsymbol{e}}$ do not need to be recomputed, as it was already observed in [KWBS12].

Only Algorithms 3.1, 3.3, and 3.6 need to be executed during the preparation phase[11] to update $\bar{\boldsymbol{r}}$ and $\bar{\boldsymbol{d}}$ in the condensed QP (3.3). The overall computational complexity of the condensing step in the preparation phase then reduces from

$$\mathcal{O}\left(N^2 \left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right)\right)$$

or even

$$\mathcal{O}\left(N^2 \left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right) + N\left(n_x^3 + n_x^2\, n_d\right)\right)$$

to

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_u + n_x\, n_d\right)\right)$$

in MPC, and

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_d\right)\right)$$

in MHE. The complexity of the operations performed in the feedback phase of course remain unaffected. In a parallel implementation on $\mathcal{O}(N)$ threads, the time complexities reduce only slightly further to

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_u\right) + n_x\, n_d\right)$$

in MPC, and

$$\mathcal{O}\left(N\, n_x^2 + n_x\, n_d\right)$$

in MHE, since the main loops of Algorithms 3.1 and 3.3 cannot be parallelized due to the recursive dependency in $\boldsymbol{g}$ and $\boldsymbol{w}$, respectively.

If, in the spirit of mixed-level iterations, only the first N_{frac} stages are updated in their vector data, while the remaining stages remain unchanged, no further significant computational savings are possible in the proposed condensing algorithm. This can easily be seen by observing from Algorithm 3.1 that already an update in $\boldsymbol{c}_{\boldsymbol{0}}$ causes an update to all blocks of $\boldsymbol{g}$, which in turn

[11] Algorithm 3.3 again moves to the feedback phase if the QP is of MHE origin.

induces an update to all block components of $\bar{\boldsymbol{r}}$ and $\bar{\boldsymbol{d}}$. If, on the other hand, only the last N_{frac} stages receive an update in their vector data, the computational complexity reduces slightly to

$$\mathcal{O}\left(N\left(n_x^2 + n_x\, n_u\right) + N_{\text{frac}}\left(n_x\, n_d\right)\right)$$

in MPC, as only the last N_{frac} block components of $\bar{\boldsymbol{d}}$ change (but $\bar{\boldsymbol{r}}$, however, needs to be recomputed entirely), and to

$$\mathcal{O}\left(N_{\text{frac}}\left(n_x^2 + n_x\, n_d\right)\right)$$

in MHE. The latter however relies on the assumption, that the intermediate sum of the first $N - N_{\text{frac}}$ terms that add to $\bar{\boldsymbol{q}}^{\boldsymbol{e}}$ in Algorithm 3.13 is saved separately for warmstarting the summation loop.

3.3.2 Partial exact re-linearizations

If a full re-linearization is performed on a stage $k \in \mathcal{S}$, all[12] data matrices and vectors of this stage are recomputed, i.e., $\boldsymbol{Q_k}$, $\boldsymbol{R_k}$, $\boldsymbol{S_k}$, $\boldsymbol{q_k}$, $\boldsymbol{r_k}$, $\boldsymbol{A_k}$, $\boldsymbol{B_k}$, $\boldsymbol{c_k}$, $\boldsymbol{D_k^x}$, $\boldsymbol{D_k^u}$, and $\boldsymbol{d_k}$.

From the discussion in Section 3.3.1 it is clear that no computational savings arise in the vector data for a re-condensing step after an update to only the first N_{frac} stages. Savings are however possible in Algorithms 3.2, 3.4, and 3.5, since only the first N_{frac} block columns of $\boldsymbol{G}$, $\bar{\boldsymbol{R}}$, and $\bar{\boldsymbol{D}}$ depend on updated data[13]. The overall computational complexity of re-condensing in the preparation phase therefore reduces from

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right)\right)$$

to

$$\mathcal{O}\left(N\, N_{\text{frac}}\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right)\right)$$

in MPC, when no preparatory computations for the initial value embedding are performed, and from

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right) + N\left(n_x^3 + n_x^2\, n_d\right)\right)$$

to

$$\mathcal{O}\left(N\, N_{\text{frac}}\left(n_x^2\, n_u + n_x\, n_u^2 + n_x\, n_u\, n_d\right) + N\left(n_x^3 + n_x^2\, n_d\right)\right)$$

[12] W.l.o.g. we assume that redundant matrices $\boldsymbol{R_N}$, $\boldsymbol{S_N}$, $\boldsymbol{A_N}$, $\boldsymbol{B_N}$, $\boldsymbol{D_N^u}$, and vectors $\boldsymbol{r_N}$, $\boldsymbol{c_N}$ are defined for notational convenience.

[13] In case of Algorithm 3.5 this means the inner loop may be aborted after at most N_{frac} iterations.

when the initial value embedding is prepared by Algorithms 3.7, 3.8, and 3.12 in the preparation phase, or in MHE. Here, bear in mind that $\boldsymbol{G^e}$, $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$, and $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$ need to be recomputed completely as soon as $\boldsymbol{A_0}$ changes.

In a parallel implementation, only little savings are possible in the time complexity of the condensing algorithms compared to a full exact re-linearization. The time complexity becomes

$$\mathcal{O}\left(N\left(n_x^2\, n_u + n_x\, n_u^2\right) + N_{\text{frac}}\, n_x\, n_u\, n_d\right)$$

in MPC without preparatory computations for the initial value embedding, and

$$\mathcal{O}\left(N\left(n_x^3 + n_x^2\, n_u + n_x\, n_u^2\right) + N_{\text{frac}}\, n_x\, n_u\, n_d + n_x^2\, n_d\right)$$

in MPC, when preparing $\bar{\boldsymbol{S}}^{\boldsymbol{e}}$ and $\bar{\boldsymbol{D}}^{\boldsymbol{e}}$ during the preparation phase, as well as in MHE.

If, conversely, only the data of the last N_{frac} stages is updated by exact re-linearization, the computational savings are significantly smaller. The overall computational complexity reduces only slightly (compared to a full condensing step) to

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2\right) + N\, N_{\text{frac}}\, n_d\, n_x\, n_u\right)$$

for the overall-effort minimizing implementation, and to

$$\mathcal{O}\left(N^2\left(n_x^2\, n_u + n_x\, n_u^2\right) + N\, N_{\text{frac}}\, n_d\, n_x\, n_u + N\, n_x^3 + N_{\text{frac}}\, n_x^2\, n_d\right)$$

in the feedback delay minimizing implementation. This is due to the fact that Algorithm 3.4 needs to be executed completely due to its backwards substitution order; Algorithms 3.5, 3.7, and 3.12 however can be executed at lower cost, as only the respective last block components change.

In a parallel implementation, the time complexity of the overall-effort minimizing condensing algorithm assembly becomes

$$\mathcal{O}\left(N\left(n_x^2\, n_u + n_x\, n_u^2\right) + N_{\text{frac}}\, n_d\, n_x\, n_u\right),$$

and the time complexity of the feedback delay minimizing approach becomes

$$\mathcal{O}\left(N\left(n_x^3 + n_x^2\, n_u + n_x\, n_u^2\right) + N_{\text{frac}}\, n_d\, n_x\, n_u + n_x^2\, n_d\right).$$

Chapter 4

Block-Banded Quadratic Programming

In Section 3.2 we reviewed the classical approach to solving the band-structured quadratic programming problems arising in dynamic optimization. We observed that even in its new, more efficient implementation, the computational effort scales quadratically in the horizon length. These complexity theoretic considerations also reflect in practice. Particularly when using fixed-stepsize integrators that permit a well-distributed sharing of the workload among up to $N + 1$ computational threads, the bottleneck shifts towards the QP solution step with increasing horizon length and parallelization. Motivated by the fact that efficient solvers for the condensed dynamic optimization QPs, such as the Online Active Set Strategy, are in general not well suited for parallelization, we investigate alternatives to the condensing-based QP solution approach in this section. In particular, we present a novel algorithm, dubbed *dual Newton strategy*, that aims at overcoming shortcomings of the condensing-based approach for long-horizon problems.

Acknowledgement This Chapter is largely based based on the paper "A Parallel Quadratic Programming Method for Dynamic Optimization Problems" by Janick Frasch, Sebastian Sager, and Moritz Diehl [FSD13]. Moritz Diehl initiated the project that lead to this paper and contributed fundamental ideas. Janick Frasch is the main author of this paper and contributed the algorithmic design as well as the theoretical foundations and the open-source software implementation. Sebastian Sager served as a vital discussion partner,

in particular with respect to the convergence proofs. The extensions presented in Sections 4.6 and 4.7 are mostly unpublished work.

4.1 Dense vs. Sparse Solution

Historically, the application of a condensing procedure seemed highly beneficial. In linear time-invariant MPC, the costly condensing operations could be performed offline, and with sparse QP solvers still being in their early stages, the dimensionality reduction mattered significantly. Among the early nonlinear MPC problems were many from chemical engineering that often feature a significantly larger number of states n_x compared to the number of controls n_u and the horizon length N. There, the computational savings resulting from the reduced size of the condensed QP often exceeded the costs for the additional condensing step drastically.

When considering different applications, such as mechanical systems or systems with integer controls[1], as well as generally problems on long prediction horizons, this assumption often does not hold anymore, and the fact that the condensed problem is still (at least) of size Nn_u, cf. Section 3.2, may jeopardize the benefits from condensing.

A particular drawback of the condensing approach is that the reduced-size problem is entirely dense. Applying a second-order method for the solution of the condensed problem will therefore require the factorization of a dense matrix of size $\mathcal{O}(N)$ every time a re-linearization is performed on the NLP level. Using updates to the stored factors, some QP algorithms such as the Online Active Set Strategy may not require additional re-factorizations throughout their iterations. Still, these updates may become expensive themselves; in case of the Online Active Set Strategy the computational effort scales quadratically with the number of optimization variables for dense problems, cf. [Fer06, FBD08]. The effort of the initial factorization is generally even cubic in the number of variables, although it has been indicated in [AM12, FJ13] that, taking advantage of the dynamic origin, a symmetric factor of the condensed Hessian $\bar{\boldsymbol{R}}$ (cf. Problem (3.3)) can, in principle, be obtained at the cost of $\mathcal{O}(N^2)$ operations.

As an alternative, an active-set method with sparsity exploitation based on a *complementary condensing* approach was proposed in [KBSS11]. Here, the essential idea is to make use of the discretized dynamic coupling constraints

[1] Using an outer convexification approach in combinatation with integrality relaxation forms the basis of a very successful class of methods for tackling mixed-integer OCPs. This however comes at the price of additional artificial control variables, such that often $n_u \gg n_x$. We refer to [Sag05, Kir11] for details and complementary reading.

to eliminate the controls as degrees of freedom, as opposed to the classical condensing algorithm, where overparameterized states are eliminated. This makes the algorithm naturally well suited for applications that feature many controls, such as problems that stem from outer convexification treatment of integer control variables, cf. [Sag05, Kir11]. Due to the fact that each active set change calls for a new sparse matrix factorization the complementary condensing approach is however most useful only when few active set changes occur.

Even without major preprocessing steps, the block-banded structure that is exhibited by dynamic optimization QP subproblems is typically exploited well by tailored interior-point (e.g., [RWR98, MB09, DZZ+12]) and fast-gradient methods (e.g., [RJM09, BP12]) after a suitable reordering (cf. [RWR98, Dom13, Le14]). However, a well-known drawback of both classes of methods is their limited warm-starting capability to exploit the knowledge of similarity between the solutions of subsequently solved QPs, which is often critical in dynamic optimization in general, and even more so in algorithms from the family of the Real-Time Iteration scheme, where strong similarity links between the solutions of subsequently solved QPs exist.

Applying classical active set-methods, including parametric ones such as [FBD08, FKP+13], to the sparse (uncondensed) problem direction would leverage this similarity for higher efficiency. These methods, however, typically do not benefit from the problem-inherent sparsity as much as interior-point methods, and can generally be expected to perform much worse than on a condensed problem of reduced size.

In the following, we pursue a new idea for a QP algorithm based on a dual decomposition approach, that was introduced for linear MPC problems in [FKD12]. This method aims at combining the benefits of interior-point methods in terms of structure exploitation with the warm-starting capabilities of active-set methods, and comes at only a linear runtime complexity in the horizon length. Based on ideas from [LS97] and [DF06] the stage coupling constraints are dualized and the resulting QP is solved in a two level approach, using a non-smooth/semismooth Newton method in the multipliers of the stage coupling constraints on the higher level, and a primal active-set method in the decoupled parametric QPs of each stage on the lower level. Note that in contrast to classical active-set methods, this approach permits several active-set changes at the cost of one block-banded matrix factorization. Hereinafter, we refer to this procedure as a *dual Newton strategy*.

We give details for this method regarding the efficient numerical implementation in general, and in the online context in particular. Theoretical properties of the algorithm are investigated. Furthermore, we introduce a parallel algorithm for the solution of the structured Newton system, that reduces the runtime

complexity of the dual Newton strategy even further to $\mathcal{O}(\log N)$ per iteration in a sufficiently parallel computing environment.

4.2 The Dual Newton Strategy

For the presentation of our method, we adopt the following, rather generic band-structured QP setting:

$$\min_{\boldsymbol{z}} \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} \right) \tag{PQP1}$$

$$\text{s.t. } \boldsymbol{E_{k+1}}\, \boldsymbol{z_{k+1}} = \boldsymbol{C_k}\, \boldsymbol{z_k} + \boldsymbol{c_k} \qquad \forall\, k \in \mathcal{S}_N \tag{PQP2}$$

$$\underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \leq \overline{\boldsymbol{d}}_{\boldsymbol{k}} \qquad \forall\, k \in \mathcal{S}. \tag{PQP3}$$

The cost function on each stage consists of a positive definite second-order term $\mathbf{0} \prec \boldsymbol{H_k} \in \mathbb{R}^{n_z \times n_z}$ and a first-order term $\boldsymbol{g_k} \in \mathbb{R}^{n_z}$ for each $k \in \mathcal{S}$. Two subsequent stages $k \in \mathcal{S}_N$ and $k+1 \in \mathcal{S}_0$ are coupled by first-order terms $\boldsymbol{C_k}, \boldsymbol{E_{k+1}} \in \mathbb{R}^{n_x \times n_z}$ and a constant term $\boldsymbol{c_k}$. We assume that all $\boldsymbol{C_k}$ have full row rank, i.e., $\operatorname{rk}(\boldsymbol{C_k}) = n_x$, $\forall\, k \in \mathcal{S}_N$, and that all $\boldsymbol{E_k}$ have the special structure $\boldsymbol{E_k} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$, where $\mathbf{I} \in \mathbb{R}^{n_x \times n_x}$ is an identity matrix and $\mathbf{0}$ is a zero matrix of appropriate dimensions. Vectors $\underline{\boldsymbol{d}}_{\boldsymbol{k}}, \overline{\boldsymbol{d}}_{\boldsymbol{k}} \in \mathbb{R}^{n_d}$, and a matrix $\boldsymbol{D_k} \in \mathbb{R}^{n_d \times n_z}$ of full row rank denote affine stage constraints.

If the origin of (PQP) is, for example, an MPC problem, this setting could be achieved by grouping system states $\boldsymbol{x_k} \in \mathbb{R}^{n_x}$ and control inputs $\boldsymbol{u_k} \in \mathbb{R}^{n_u}$ in stage variables $\boldsymbol{z_k} = (\boldsymbol{x_k}, \boldsymbol{u_k}) \in \mathbb{R}^{n_z}$ for each stage $k \in \mathcal{S}_N$, and $\boldsymbol{z_N} = (\boldsymbol{x_N}, \mathbf{0}) \in \mathbb{R}^{n_z}$ for the terminal stage.

It is important to stress that we assume strict positive definiteness of the stage costs terms, which is slightly stronger than the typical requirement of positive semidefiniteness in quadratic programming and model predictive control. Still, for practical applications this can easily be achieved by a small primal regularization term added to the objective's second-order term. We investigate the effect of different primal regularization terms in Section 6.2.1.

If not stated differently, we assume in the following that a solution $\boldsymbol{z^*} := \left(\boldsymbol{z_0^*}, \ldots, \boldsymbol{z_N^*}\right)$ of (PQP) exists and fulfills the LICQ, cf. Definition 1.27. Section 4.5 discusses the consequences resulting from infeasibility of (PQP) and states how infeasible instances can be detected.

4.2.1 Dual decomposition

We decouple the QP stages by dualizing constraints (PQP2). Introducing

$$\boldsymbol{\lambda} := (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \ldots, \boldsymbol{\lambda_N}) \in \mathbb{R}^{N n_x} \tag{4.2}$$

we can express (PQP1) and (PQP2) by the partial Lagrangian function

$$\begin{aligned}\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) &:= \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} \right) \\ &\qquad + \sum_{k=0}^{N-1} \boldsymbol{\lambda_{k+1}}^\top (-\boldsymbol{E_{k+1}} \boldsymbol{z_{k+1}} + \boldsymbol{C_k} \boldsymbol{z_k} + \boldsymbol{c_k}) \\ &= \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} + \begin{bmatrix} \boldsymbol{\lambda_k} \\ \boldsymbol{\lambda_{k+1}} \end{bmatrix}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix} \boldsymbol{z_k} + \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{c_k} \right),\end{aligned}$$

where we define zero matrices $\boldsymbol{E_0} = \boldsymbol{C_N} = \boldsymbol{0} \in \mathbb{R}^{n_x \times n_z}$ and redundant multipliers $\boldsymbol{\lambda_0} = \boldsymbol{\lambda_{N+1}} := \boldsymbol{0} \in \mathbb{R}^{n_x}$ only for notational convenience in this context (note that they are not among the optimization variables of the dual problem defined below).

By Lagrangian duality, the solution of (PQP) can therefore be computed as

$$\begin{aligned}\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{z}} & \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} + \begin{bmatrix} \boldsymbol{\lambda_k} \\ \boldsymbol{\lambda_{k+1}} \end{bmatrix}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix} \boldsymbol{z_k} + \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{c_k} \right) \\ \text{s.t. } & \underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \leq \overline{\boldsymbol{d}}_{\boldsymbol{k}} \qquad \forall k = 0, \ldots, N.\end{aligned}$$

As this problem is separable in the stage variables $\boldsymbol{z_k}$, minimization and summation can be interchanged, and a solution to (PQP) is obtained by solving

$$\max_{\boldsymbol{\lambda}} f^*(\boldsymbol{\lambda}) := \max_{\boldsymbol{\lambda}} \sum_{k=0}^{N} f_k^*(\boldsymbol{\lambda}), \tag{DQP}$$

where

$$\begin{aligned} f_k^*(\boldsymbol{\lambda}) \;:=\; \min_{\boldsymbol{z_k}} \; & \frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \left(\boldsymbol{g_k}^\top + \begin{bmatrix} \boldsymbol{\lambda_k} \\ \boldsymbol{\lambda_{k+1}} \end{bmatrix}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix} \right) \boldsymbol{z_k} + \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{c_k} \\ \text{s.t.} \quad & \underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \;\leq\; \overline{\boldsymbol{d}}_{\boldsymbol{k}}. \end{aligned} \tag{QP$_k$}$$

We refer to (QP_k) as stage QP. Note that each (QP_k) depends on at most two block components of the vector of dual optimization variables $\boldsymbol{\lambda}$ defined in (4.2).

Remark 4.1 Since $\boldsymbol{\lambda}$ only enters in the objective of each (QP_k), feasibility of (QP_k), and thus existence of $f_k^*(\boldsymbol{\lambda})$ is independent of the choice of $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$. In particular, since the constraints of (QP_k) are a subset of (PQP3), feasibility of (PQP) implies feasibility of (QP_k).

Remark 4.2 Each $f_k^*(\boldsymbol{\lambda})$ implicitly defines a $\boldsymbol{z_k^*}(\boldsymbol{\lambda})$, the solution of ($\mathrm{QP}_k$).

4.2.2 Characterization of the dual function

It was shown in [FKD12] (based on results from [Fia83, Zaf90, BRT97]) that $f^*(\boldsymbol{\lambda})$ is concave, piecewise quadratic, and once continuously differentiable. We establish relevant findings in the following.

We begin by instantiating the definition of an active set in our specific context.

Definition 4.3 For a stage $k \in \mathcal{S}$, the optimal *active set* at $\boldsymbol{\lambda}$ is given by

$$\mathcal{A}_k^*(\boldsymbol{z_k^*}(\boldsymbol{\lambda})) := \left\{ 1 \leq i \leq n_d \mid (\boldsymbol{D_k})_{\boldsymbol{i},\cdot} \boldsymbol{z_k^*}(\boldsymbol{\lambda}) = (\underline{d}_k)_i \vee (\boldsymbol{D_k})_{\boldsymbol{i},\cdot} \boldsymbol{z_k^*}(\boldsymbol{\lambda}) = \left(\overline{d}_k\right)_i \right\},$$

i.e., the set of row indices of the constraints of (QP_k) that are fulfilled with equality. ⌟

Definition 4.3 naturally extends to a definition of the active set in the full space of primal variables by $\mathcal{A}^*(\boldsymbol{z^*}(\boldsymbol{\lambda})) := \mathcal{A}_0^*(\boldsymbol{z_0^*}(\boldsymbol{\lambda})) \times \ldots \times \mathcal{A}_N^*(\boldsymbol{z_N^*}(\boldsymbol{\lambda}))$. The finite number of disjoint active sets further induces a subdivision of the dual $\boldsymbol{\lambda}$ space:

Definition 4.4 Each active set defines a *region* $A \subseteq \mathbb{R}^{Nn_x}$ in the dual $\boldsymbol{\lambda}$ space. For a representative $\boldsymbol{\lambda}^{(j)} \in \mathbb{R}^{Nn_x}$ we have

$$A^{(j)} := \{\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x} \mid \mathcal{A}^*(\boldsymbol{z^*}(\boldsymbol{\lambda})) = \mathcal{A}^*(\boldsymbol{z^*}(\boldsymbol{\lambda}^{(j)}))\}.$$

By choosing representatives of pairwise distinct regions we can define an arbitrary, but fixed ordering that allows us to uniquely identify each region $A^{(j)}$. ⌟

The name *region* is anticipatory, but it will become clear from Corollary 4.7 that each set $A^{(j)}$ is indeed connected. The number of regions n_r clearly is finite, as there is only a finite number of distinct active sets. From Remark 4.1 we can conclude that each $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$ is contained in a region $A^{(j)}$, and thus

$$\bigcup_{1 \leq j \leq n_r} A^{(j)} = \mathbb{R}^{Nn_x}.$$

Remark 4.5 Two distinct regions $A^{(j_1)}$ and $A^{(j_2)}$ do not need to be disjoint. Values of $\boldsymbol{\lambda}$ that lead to weakly active stage constraints are contained in two or more regions. These values of $\boldsymbol{\lambda}$ form the seams of the regions.

Next, we substantiate Remark 4.2 by characterizing the nature of the dependency of $\boldsymbol{z}_k^*$ on the dual variables $\boldsymbol{\lambda}$ in the stage problems (QP_k).

Lemma 4.6 *Let (QP_k) be feasible. Then, the optimal solution of (QP_k), $\boldsymbol{z}_k^*(\boldsymbol{\lambda})$, is a piecewise affine and continuous function in $\boldsymbol{\lambda}$. In particular, the dependency is affine within each region $A^{(j)}$, $1 \leq j \leq n_r$.*

Proof (cf. [FKD12], Lem. 2; [Zaf90]) For stage Lagrange multipliers $\boldsymbol{\mu}_k \in \mathbb{R}^{2n_d}$ the solution of (PQP) is given by (see, e.g., [Fle87])

$$\begin{bmatrix} \boldsymbol{H}_k & -\underline{\boldsymbol{D}}_k^{*\top} & \overline{\boldsymbol{D}}_k^{*\top} \\ -\underline{\boldsymbol{D}}_k^* & \boldsymbol{0} & \boldsymbol{0} \\ \overline{\boldsymbol{D}}_k^* & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_k^* \\ \boldsymbol{\mu}_k^* \end{bmatrix} = \begin{bmatrix} -\left(\boldsymbol{g}_k{}^\top + \begin{bmatrix} \boldsymbol{\lambda}_k \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix}^\top \begin{bmatrix} -\boldsymbol{E}_k \\ \boldsymbol{C}_k \end{bmatrix} \right) \\ \underline{\boldsymbol{d}}_k^* \\ -\overline{\boldsymbol{d}}_k^* \end{bmatrix}, \tag{4.3}$$

where $\underline{\boldsymbol{D}}_k^*$, $\underline{\boldsymbol{d}}_k^*$ and $\overline{\boldsymbol{D}}_k^*$, $\overline{\boldsymbol{d}}_k^*$ consist of the rows of $\boldsymbol{D}_k$, $\underline{\boldsymbol{d}}_k$, and $\overline{\boldsymbol{d}}_k$ that correspond to the constraints that are active (i.e., fulfilled with equality) at the lower or, respectively, the upper bound in the solution $\boldsymbol{z}_k^*$, and $\boldsymbol{\mu}_k^*$ is the vector of consistent dimension that contains the corresponding multipliers. The remaining stage multiplier entries are 0 in the solution of (PQP). As $\boldsymbol{\lambda}$ enters affinely only on the right-hand side, it is clear that for identical active sets it holds that $\boldsymbol{z}_k^*$ depends affinely on $\boldsymbol{\lambda}$. Continuity has been shown in [Fia83]. □

Corollary 4.7 *Each region $A^{(j)}$, $1 \leq j \leq n_r$, of the dual space is convex and polyhedral.*

Proof Each stage problem (QP_k) is constrained by affine constraints. For a given representative $\boldsymbol{\lambda}^{(j)}$ the set $\mathcal{F}_k := \{\boldsymbol{z}_k \in \mathbb{R}^{n_z} \mid \mathcal{A}_k(\boldsymbol{z}_k) = \mathcal{A}_k^*(\boldsymbol{z}_k^*(\boldsymbol{\lambda}^{(j)}))\}$ is therefore convex and polyhedral. From Lemma 4.6 we have that $\boldsymbol{z}_k^*$ is affine in $\boldsymbol{\lambda}$ for a certain (fixed) active set. A region $A^{(j)}$ is therefore the intersection of $N+1$ (i.e., a finite number) preimages of convex sets, and therefore convex. □

Lemma 4.6 is the basis for the following exhaustive characterization of the dual function $f^*(\boldsymbol{\lambda})$, which we take from [FKD12] without proof.

Lemma 4.8 ([FKD12], Lem. 3) *If all stage QPs (QP_k) are feasible, then the dual function $f^*(\boldsymbol{\lambda})$ exists and is described by a concave, continuously differentiable, and piecewise quadratic spline in $\boldsymbol{\lambda}$ space.*

Remark 4.9 In particular $f^*(\boldsymbol{\lambda})$ is quadratic on each region $A^{(j)}$.

4.2.3 Solution by a (non-smooth) Newton method

Dual decomposition by itself is of course a rather well-known technique. It's early roots in the area of convex optimization go back to [Eve63]. More recent literature reviews can be found in [Ber99, NS08]. In the area of optimal control and particularly model predictive control, dual decomposition techniques have been employed, for example, in [GR10, GDK+13, NDS08, Ran09, RMJ11]. These works propose algorithms that exclusively make use of gradient information to maximize the continuously differentiable, but not twice continuously differentiable dual function $f^*(\boldsymbol{\lambda})$. Alternatively, works like [NS09] and [TDNSD12] employ a smoothed dual function to the end of applying an interior-point framework to the dual decomposition reformulation. A major drawback of these methods, however, is that typically a rather large number of iterations is required to achieve medium accuracy in the solution. This characteristic is fundamental to first-order and overly perturbed second-order methods, cf. Section 1.4.4.

In an attempt to overcome these shortcomings, our approach is to tackle (DQP) as an unconstrained optimization problem of a piecewise quadratic spline $f^*(\boldsymbol{\lambda})$, and to employ a non-smooth Newton method[2] to it, as originally proposed in [LS97], and also used in [FKD12]. We iterate

$$\boldsymbol{\lambda}^{\boldsymbol{i+1}} := \boldsymbol{\lambda}^{\boldsymbol{i}} + \alpha\, \boldsymbol{\Delta\lambda} \tag{4.4}$$

(the Newton iterates $\boldsymbol{\lambda}^{\boldsymbol{i}}$ are not to be confused with the region representatives $\boldsymbol{\lambda}^{(\boldsymbol{j})}$ from Section 4.2.2) for an initial guess $\boldsymbol{\lambda}^{\boldsymbol{0}}$ and a suitably chosen step size α until $f^*(\boldsymbol{\lambda}^{\boldsymbol{i}})$ is stationary. The step direction $\boldsymbol{\Delta\lambda}$ is computed from

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\, \boldsymbol{\Delta\lambda} = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}), \tag{4.5}$$

where $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) := -\frac{\partial^2 f^*}{\partial \boldsymbol{\lambda}^2}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ and $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) := \frac{\partial f^*}{\partial \boldsymbol{\lambda}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$. By Remarks 4.9 and 4.5, $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ is unique everywhere but on the seams of f^* (a null set), where an arbitrary, but fixed second derivative from the finite number of possible choices is used, ensuring well-definedness of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$.

Clearly, stationarity of $f^*(\boldsymbol{\lambda}^{\boldsymbol{i}})$ is equivalent to optimality of (DQP). Observe that by definition of (DQP), $\boldsymbol{z}^*(\boldsymbol{\lambda}^{\boldsymbol{i}})$ is always optimal and (PQP3) are always fulfilled in the spirit of an active-set method. Lemma 4.10 will show that feasibility of (PQP2) is identical with stationarity of $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$.

The complete QP solution method is given in Algorithm 4.1, where we denote the Lagrange multipliers of the stage constraints (PQP3) by $\boldsymbol{\mu_k} \in \mathbb{R}^{2n_d}$ for each

[2] More specifically, our problem fulfills the requirements for *semismoothness*, a term that was coined in the domain of optimization by [Mif77] and specifically in the context of Newton's method by [QS93].

Algorithm 4.1: Dual Newton Strategy

Input: Initial guess $\boldsymbol{\lambda^0}$, termination criteria n_{maxIt}, ϵ_λ
Output: Optimal solution $(\boldsymbol{z^*}, \boldsymbol{\lambda^*}, \boldsymbol{\mu^*})$

1 **for** $i = 0 : (n_{\text{maxIt}} - 1)$ **do**
2 Solve all $\text{QP}_k(\boldsymbol{\lambda^i})$ to obtain $[\boldsymbol{z_k^*}(\boldsymbol{\lambda^i}), \boldsymbol{\mu_k^*}(\boldsymbol{\lambda^i})]$
3 Set up gradient $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$
4 **if** $\left\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\right\| \leq \epsilon_\lambda$ **then**
5 **return** $[\boldsymbol{z_k^*}(\boldsymbol{\lambda^i}), \boldsymbol{\lambda^i}, \boldsymbol{\mu_k^*}(\boldsymbol{\lambda^i})]$
6 Set up Newton matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i})$
7 Solve Newton system (4.5)
8 Compute appropriate step size α
9 Update current iterate $\boldsymbol{\lambda^{i+1}} := \boldsymbol{\lambda^i} + \alpha\, \boldsymbol{\Delta\lambda}$

stage $k \in \mathcal{S}$. Note that the parametric solution of the stage problems (QP_k) for the current iterate $\boldsymbol{\lambda^i}$ in Step 2, as well as the setup of $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$ and $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i})$ in Steps 3 and 6 permits an independent, concurrent execution on all stages (see also Section 4.6). A convergence proof for Algorithm 4.1 is given in Section 4.4.

4.2.4 Characterization of the dual Newton iterations

A full QP solution by Algorithm 4.1 can be visualized as in Figure 4.1. Each cell corresponds to a region $A^{(j)}$ in $\boldsymbol{\lambda}$-space, for which the primal active set is constant. Starting from an initial guess $\boldsymbol{\lambda^0}$, a Newton step direction is computed from Equation (4.5) that leads to $\boldsymbol{\lambda^1_{FS}}$. Using a globalization strategy (see Section 4.3.6), a suitable step size $\boldsymbol{\lambda^1_{\bar{\alpha}}}$ is found. In contrast to classical active-set methods, *multiple* active set changes are possible *in one iteration*. For future reference we also indicate a minimum guaranteed step $\boldsymbol{\lambda^1_{\alpha_{\min}}}$. In the second iteration, $\boldsymbol{\lambda^2_{FS}}$ already provides sufficient progress, thus no globalization is applied. In the following iteration $\boldsymbol{\lambda^*}$ is found. We will proof in Lemma 4.19 that a one-step terminal convergence is guaranteed, once the correct region is identified.

4.3 Algorithmic Details

The dynamic optimization origin induces a specific structure in Problem (DQP), that we strive to exploit in the following.

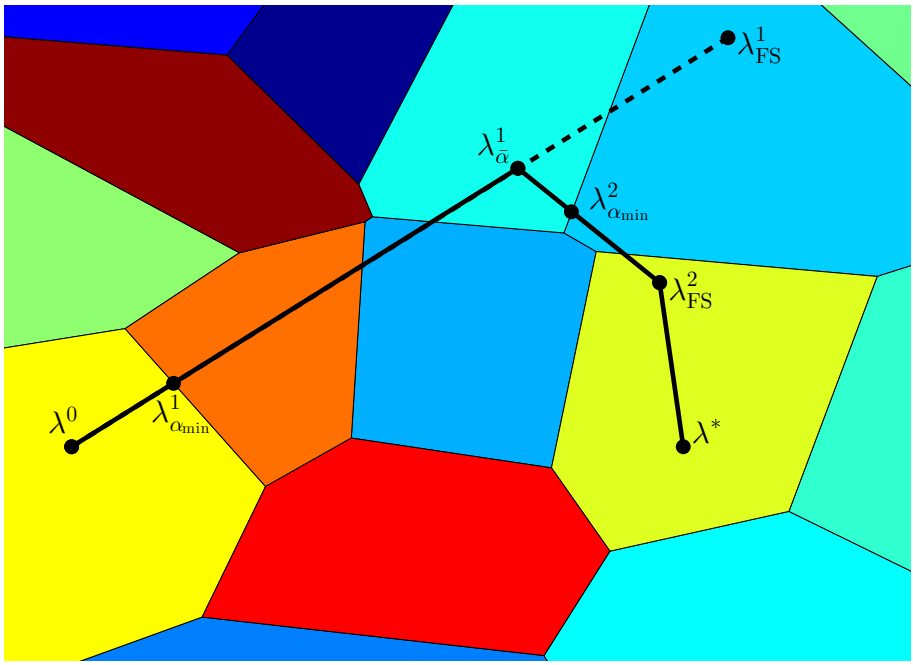


Figure 4.1: Steps of the dual Newton strategy in the $\boldsymbol{\lambda}$-space.

4.3.1 Solution of decoupled parametric stage QPs

On each stage $k \in \mathcal{S}$ we have to repeatedly solve a QP of size (n_z, n_d) that only changes in the first-order term (and the negligible constant term) with the current guess of $\boldsymbol{\lambda}$. We have QP_k given by

$$\min_{\boldsymbol{z_k}} \frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{m_k}(\boldsymbol{\lambda})^\top \boldsymbol{z_k} + p_k(\boldsymbol{\lambda})$$

$$\text{s.t. } \underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \leq \overline{\boldsymbol{d}}_{\boldsymbol{k}},$$

with $\boldsymbol{m_k}(\boldsymbol{\lambda})^\top := \boldsymbol{g_k}^\top - \boldsymbol{\lambda_k}^\top \boldsymbol{E_k} + \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{C_k}$, $p_k(\boldsymbol{\lambda}) := \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{c_k}$, and, in general, $\boldsymbol{H_k}$ and $\boldsymbol{D_k}$ dense. Such QPs can be solved efficiently (see [FBD08, Bes96]), by employing a parametric QP solver such as the Online Active Set Strategy, cf. Section 1.5.4, which is implemented in the open-source QP solver qpOASES [FKP+13].

In the special, yet practically relevant, case where $\boldsymbol{H_k}$ is a diagonal matrix and $\boldsymbol{D_k}$ is an identity matrix (i.e., only bounds on states and controls exist) the

optimal solution $\boldsymbol{z_k^*}$ can conveniently be computed by component-wise "clipping" of the unconstrained solution as it was presented in [FKD12]:

$$\boldsymbol{z_k^*} = \mathbf{max}(\underline{\boldsymbol{d}}_{\boldsymbol{k}}, \mathbf{min}(\boldsymbol{H_k}^{-1}\boldsymbol{m_k}, \overline{\boldsymbol{d}}_{\boldsymbol{k}})). \tag{4.6}$$

4.3.2 Structure of the Newton system

The right hand side vector $\boldsymbol{\mathcal{G}} : \mathbb{R}^{Nn_x} \to \mathbb{R}^{Nn_x}$ of Newton system (4.5) is easily seen to only depend on two neighboring stages in each block $\boldsymbol{\lambda_k}$. It holds that

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) := \frac{\boldsymbol{\partial f^*}}{\boldsymbol{\partial \lambda}}(\boldsymbol{\lambda})^\top = \begin{bmatrix} \frac{\boldsymbol{\partial f_0^*}}{\boldsymbol{\partial \lambda_1}}^\top + \frac{\boldsymbol{\partial f_1^*}}{\boldsymbol{\partial \lambda_1}}^\top \\ \frac{\boldsymbol{\partial f_1^*}}{\boldsymbol{\partial \lambda_2}}^\top + \frac{\boldsymbol{\partial f_2^*}}{\boldsymbol{\partial \lambda_2}}^\top \\ \vdots \\ \frac{\boldsymbol{\partial f_{N-1}^*}}{\boldsymbol{\partial \lambda_N}}^\top + \frac{\boldsymbol{\partial f_N^*}}{\boldsymbol{\partial \lambda_N}}^\top \end{bmatrix} (\boldsymbol{\lambda}). \tag{4.7}$$

The left-hand side Newton matrix $\boldsymbol{\mathcal{M}} : \mathbb{R}^{Nn_x} \to \mathbb{R}^{Nn_x \times Nn_x}$ has a block tri-diagonal structure, as only neighboring multipliers $\boldsymbol{\lambda_k}, \boldsymbol{\lambda_{k+1}}$ can have a joint contribution to f^*. At a fixed $\boldsymbol{\lambda}$ it holds

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) := -\frac{\boldsymbol{\partial^2 f^*}}{\boldsymbol{\partial \lambda^2}}(\boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{W_1} & \boldsymbol{U_1} & & \\ \boldsymbol{U_1}^\top & \boldsymbol{W_2} & \ddots & \\ & \ddots & \ddots & \boldsymbol{U_{N-1}} \\ & & \boldsymbol{U_{N-1}}^\top & \boldsymbol{W_N} \end{bmatrix} (\boldsymbol{\lambda}), \tag{4.8}$$

where the diagonal and off-diagonal block components are given by

$$\boldsymbol{W_k}(\boldsymbol{\lambda}) := -\frac{\boldsymbol{\partial^2 f^*}}{\boldsymbol{\partial \lambda_k^2}}(\boldsymbol{\lambda}) \qquad \text{and} \qquad \boldsymbol{U_k}(\boldsymbol{\lambda}) := -\frac{\boldsymbol{\partial^2 f^*}}{\boldsymbol{\partial \lambda_k \lambda_{k+1}}}(\boldsymbol{\lambda}). \tag{4.9}$$

4.3.3 Gradient and Hessian computation

Lemma 4.10 *(cf. [BT89, App. C]) Let all* (QP_k) *be feasible. Then the derivative of* f_k^* *with respect to the dual variables* $\boldsymbol{\lambda}$ *exists and is given by*

$$\begin{bmatrix} \frac{\boldsymbol{\partial f_k^*}}{\boldsymbol{\partial \lambda_k}} & \frac{\boldsymbol{\partial f_k^*}}{\boldsymbol{\partial \lambda_{k+1}}} \end{bmatrix} = \boldsymbol{z_k^*}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix}^\top + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{c_k} \end{bmatrix}^\top. \tag{4.10}$$

Proof The derivative $\frac{\partial f_k^*}{\partial \boldsymbol{\lambda}}$ exists by Lemma 4.8. We derive a closed form by regarding the stage QP Lagrangian

$$\mathcal{L}_k(\boldsymbol{z_k}, \boldsymbol{\mu_k}; \boldsymbol{\lambda}) := \frac{1}{2}\boldsymbol{z_k}^\top \boldsymbol{H_k} \boldsymbol{z_k} + \boldsymbol{m_k}(\boldsymbol{\lambda})^\top \boldsymbol{z_k} + p_k(\boldsymbol{\lambda}) + \boldsymbol{\mu_k}^\top \begin{bmatrix} \boldsymbol{D_k z_k} - \underline{\boldsymbol{d}}_{\boldsymbol{k}} \\ \overline{\boldsymbol{d}}_{\boldsymbol{k}} - \boldsymbol{D_k z_k} \end{bmatrix}.$$

Since $\boldsymbol{H_k} \succ \boldsymbol{0}$ and (QP_k) is feasible by assumption, it holds that (QP_k) has a (finite) optimal primal and dual solution $(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*})$, and, by Danskin's Theorem [Dan67], we can interchange optimization and derivation in the sense that

$$\frac{\partial \boldsymbol{f_k^*}}{\partial \boldsymbol{\lambda}} = \frac{\partial}{\partial \boldsymbol{\lambda}} \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})$$

holds. We then have

$$\begin{aligned}
\frac{\partial \boldsymbol{f_k^*}}{\partial \boldsymbol{\lambda}} &= \frac{\partial \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} + \frac{\partial \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})}{\partial \boldsymbol{z_k^*}} \cdot \frac{\partial \boldsymbol{z_k^*}}{\partial \boldsymbol{\lambda}} + \frac{\partial \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})}{\partial \boldsymbol{\mu_k^*}} \cdot \frac{\partial \boldsymbol{\mu_k^*}}{\partial \boldsymbol{\lambda}} \\
&= \frac{\partial \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} + \frac{\partial \boldsymbol{\mathcal{L}_k}(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*}; \boldsymbol{\lambda})}{\partial \boldsymbol{z_k^*}} \cdot \frac{\partial \boldsymbol{z_k^*}}{\partial \boldsymbol{\lambda}} + \begin{bmatrix} \boldsymbol{D_k z_k} - \underline{\boldsymbol{d}}_{\boldsymbol{k}} \\ \overline{\boldsymbol{d}}_{\boldsymbol{k}} - \boldsymbol{D_k z_k} \end{bmatrix}^\top \cdot \frac{\partial \boldsymbol{\mu_k^*}}{\partial \boldsymbol{\lambda}} \\
&= \left(\boldsymbol{z_k^*}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix}^\top + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{c_k} \end{bmatrix}^\top \right) + \boldsymbol{0} \cdot \frac{\partial \boldsymbol{z_k^*}}{\partial \boldsymbol{\lambda}} + \boldsymbol{0},
\end{aligned}$$

where the second and the third term vanish due to the stationarity and, respectively, the complementarity requirement of the optimal stage solution $(\boldsymbol{z_k^*}, \boldsymbol{\mu_k^*})$. □

Remark 4.11 We can see from Lemma 4.10 that $\| \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) \|$ is indeed a measure for both stationarity of $f^*(\boldsymbol{\lambda})$ and infeasibility of (PQP2), as claimed in Section 4.2.3.

The second derivative of f^* can be computed as follows:

Lemma 4.12 *Let $\boldsymbol{Z_k^*} \in \mathbb{R}^{n_z \times (n_z - n_k^{\mathrm{act}})}$, $k \in \mathcal{S}$ (where n_k^{act} denotes the number of active constraints) be a basis matrix for the nullspace of $\mathcal{A}_k^*(\boldsymbol{z_k^*}(\boldsymbol{\lambda}))$, the optimal active set of (QP_k) at $\boldsymbol{\lambda}$, and let $\boldsymbol{P_k^*} := \boldsymbol{Z_k^*}(\boldsymbol{Z_k^*}^\top \boldsymbol{H_k} \boldsymbol{Z_k^*})^{-1} \boldsymbol{Z_k^*}^\top \in \mathbb{R}^{n_z \times n_z}$ denote the elimination matrix for this nullspace. Then $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ is given by*

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{C}} \boldsymbol{\mathcal{P}} \boldsymbol{\mathcal{C}}^\top,$$

where $\mathcal{P} := \text{block diag}(\boldsymbol{P_0^*}, \boldsymbol{P_1^*}, \ldots, \boldsymbol{P_N^*})$ *and*

$$\boldsymbol{\mathcal{C}} := \begin{bmatrix} \boldsymbol{C_0} & -\boldsymbol{E_1} & & & \\ & \boldsymbol{C_1} & -\boldsymbol{E_2} & & \\ & & \ddots & \ddots & \\ & & & \boldsymbol{C_{N-1}} & -\boldsymbol{E_N} \end{bmatrix} \in \mathbb{R}^{Nn_x \times (N+1)n_z}.$$

Proof We compute the Hessian blocks in (4.8) explicitly. Differentiating (4.10) once more with respect to $\boldsymbol{\lambda}$, we obtain

$$\frac{\boldsymbol{\partial^2 f^*}}{\boldsymbol{\partial \lambda_k \lambda_{k+1}}} = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial \lambda_k}} \left(\frac{\boldsymbol{\partial f_k^*}}{\boldsymbol{\partial \lambda_{k+1}}} + \frac{\boldsymbol{\partial f_{k+1}^*}}{\boldsymbol{\partial \lambda_{k+1}}} \right) = \frac{\boldsymbol{\partial z_k^*}}{\boldsymbol{\partial \lambda_k}} \boldsymbol{C_k}^\top - \underbrace{\frac{\boldsymbol{\partial z_{k+1}^*}}{\boldsymbol{\partial \lambda_k}}}_{=0} \boldsymbol{E_{k+1}}^\top$$

and

$$\frac{\boldsymbol{\partial^2 f^*}}{\boldsymbol{\partial \lambda_k \lambda_k}} = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial \lambda_k}} \left(\frac{\boldsymbol{\partial f_{k-1}^*}}{\boldsymbol{\partial \lambda_k}} + \frac{\boldsymbol{\partial f_k^*}}{\boldsymbol{\partial \lambda_k}} \right) = \frac{\boldsymbol{\partial z_{k-1}^*}}{\boldsymbol{\partial \lambda_k}} \boldsymbol{C_{k-1}}^\top - \frac{\boldsymbol{\partial z_k^*}}{\boldsymbol{\partial \lambda_k}} \boldsymbol{E_k}^\top.$$

Within a fixed active set, the optimal solution of (QP_k) at $\boldsymbol{\lambda}$ is given by (cf., e.g., [NW00])

$$\boldsymbol{z_k^*}(\boldsymbol{\lambda}) = -\boldsymbol{P_k^*}^{-1} \boldsymbol{m_k}(\boldsymbol{\lambda}) = -\boldsymbol{P_k^*}^{-1} \left(\boldsymbol{g_k} + \boldsymbol{E_k}^\top \boldsymbol{\lambda_k} + \boldsymbol{C_k}^\top \boldsymbol{\lambda_{k+1}} \right).$$

Accordingly, the Hessian blocks (cf. Equation (4.9)) are computed as

$$\boldsymbol{U_k} = -\boldsymbol{E_k} \boldsymbol{P_k^*} \boldsymbol{C_k}^\top \tag{4.11}$$

and

$$\boldsymbol{W_k} = \boldsymbol{C_{k-1}} \boldsymbol{P_{k-1}^*} \boldsymbol{C_{k-1}}^\top + \boldsymbol{E_k} \boldsymbol{P_k^*} \boldsymbol{E_k}^\top, \tag{4.12}$$

which concludes the proof. □

Remark 4.13 It is important to note that $\boldsymbol{P_k^*}$ can be obtained relatively cheaply from a null-space QP solver like qpOASES [FKP+13] that directly provides $\boldsymbol{Z_k^*}$ and a Cholesky factor $\boldsymbol{R}$ for $\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{Z_k^*}^\top \boldsymbol{H_k} \boldsymbol{Z_k^*}$, see [FBD08]. For the special case of diagonal Hessian matrices $\boldsymbol{H_k} = \text{diag}(h_k^1, \ldots, h_k^{n_z})$ and simple bounds, the projection $\boldsymbol{P_k^*}$ is simply a diagonal matrix with either $1/h_k^i$ or 0 entries depending on whether the corresponding variable bound is inactive or active. The calculation of the Hessian blocks can then be accelerated even further using diadic products as proposed in [FKD12].

Remark 4.14 From the construction of $\boldsymbol{\mathcal{M}}$ in Lemma 4.12 we can see that the dual Newton strategy is, in principle, also capable of dealing with certain indefinite problems of the form (PQP), as long as the reduced Hessian blocks $\boldsymbol{Z_k^*}^\top \boldsymbol{H_k} \boldsymbol{Z_k^*}$ remain positive (semi-)definite for all $k \in \mathcal{S}$. This could, for example, be the case when equality stage constraints eliminate the negative definite directions from the stage Hessians. Basic requirement for this work is of course the stage QP solvers' support of such indefinite QPs.

4.3.4 Solution of the Newton system and regularization

By Lemma 4.8, $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ is positive semidefinite, as $f^*(\boldsymbol{\lambda})$ is concave. The block-tridiagonal structure of the $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$, cf. Equation (4.8), can be exploited for the efficient solution of the Newton system (4.5). Observing that a lower triangular factor $\boldsymbol{L}$ of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \boldsymbol{L}\,\boldsymbol{L}^\top$ possesses the same structural zero blocks below the diagonal, we suggest to employ a banded Cholesky decomposition. This factorization differs from a regular Cholesky decomposition (see, e.g., [NW00]) by skipping all redundant blocks left and below the subdiagonal block $\boldsymbol{U_k}^\top$ of each block column k, thus reducing the computational complexity from $\mathcal{O}(N^3 n_x^3)$ to $\mathcal{O}(N n_x^3)$ floating point operations (FLOPs).

In the case of jointly redundant active constraints in several (QP_k) via the stage coupling constraints (PQP2), $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ may become rank-deficient [LS97]. We propose to overcome this by applying regularization. In the software package qpDUNES, where we implemented the dual Newton strategy (see also Section 6.1), both a Levenberg-Marquadt-type regularization and a "on-the-fly" regularization are available. While the former one uses

$$\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i}) := \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i}) + \gamma \cdot \mathbf{I} \tag{4.13}$$

with a (small) constant regularization parameter $\gamma \in \mathbb{R}^+$ instead of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i})$ in the Newton system (4.5) on detection of singularity during the initial banded Cholesky factorization, the latter one only regularizes those diagonal elements for which the crucial division step in the Cholesky decomposition cannot be performed due to singularity (similarly to the modified Cholesky factorization described in [NW00], based on [GMW81]). We note that the "on-the-fly" regularization ensures positive definiteness of the resulting $\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})$ (as it has a unique Cholesky decomposition) and avoids the need of restarting the factorization, but may be numerically less stable.

Algorithm 4.2: Structure-exploiting reverse Cholesky factorization

Input: Newton Hessian matrix $\boldsymbol{\mathcal{M}}$
Output: Cholesky-like factor R for $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{R}}\boldsymbol{\mathcal{R}}^\top$

```
for k = N : 1 do                                   /* go by block columns */
    for j = k · n_x : (k − 1) · n_x + 1 do          /* go by columns */
        w = M_jj
        l̄ = min(N · n_x, (k + 1) · n_x)             /* end of row fill in */
        for l = j + 1 : l̄ do                        /* subtract row tail */
            w = w − R_jl^2
        R_jj = √w
        ī = max(1, (k − 2) · n_x + 1)               /* end of column fill in */
        for i = j − 1 : ī do                        /* write rest of column */
            w = M_ij
            if i > (k − 1) · n_x then               /* end of row fill in */
                l̄ = min(N · n_x, (k + 1) · n_x)
            else
                l̄ = min(N · n_x, k · n_x)
            for l = j + 1 : l̄ do                    /* subtract row tail */
                w = w − R_jl · R_il
            R_ij = w/R_jj
```

4.3.5 A reverse Cholesky factorization for improved stability

Particularly in the context of MPC, one expects rather many active constraints in the beginning of the control horizon, and few to none towards the end of the horizon. We aim to exploit this knowledge by applying a Cholesky factorization to $\boldsymbol{\mathcal{M}}$ (we omit the $\boldsymbol{\lambda}$-dependency in this section for notational convenience) in reverse order, i.e., starting from the last row/column, as detailed in Algorithm 4.2. Instead of a factorization $\boldsymbol{\mathcal{M}} = \boldsymbol{L}\,\boldsymbol{L}^\top$, we obtain a Cholesky-like factorization $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{R}}\,\boldsymbol{\mathcal{R}}^\top$, that is equally well suited for an efficient solution of the Newton system (4.5). To see this, observe that Algorithm 4.2 is equivalent to a standard Cholesky factorization applied to $\hat{\boldsymbol{\mathcal{M}}} := \boldsymbol{\Pi}\,\boldsymbol{\mathcal{M}}\,\boldsymbol{\Pi}^\top$ after a full row and column permutation through $\boldsymbol{\Pi} := \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$. The advantage of applying this reverse Cholesky factorization in the dual Newton strategy is twofold. First, observe that a diagonal block $\boldsymbol{W_k}$ only changes from one Newton iteration to the next if the active set on stage k or stage $k-1$ changes, and

an off-diagonal block $\boldsymbol{U_k}$ only changes if the active set on stage k changes (in particular note that $\boldsymbol{\mathcal{M}}$ only needs to be recomputed in blocks with active set changes). As Algorithm 4.2 only uses data from the last k block rows (and columns) in block iteration k, it is sufficient to restart the factorization from the block row that corresponds to the last active set change. Furthermore, we can also expect better numerical properties of $\boldsymbol{\mathcal{R}}$, as the principal submatrix corresponding to stages without active state constraints is positive definite (recall that rank-deficiency of $\boldsymbol{\mathcal{M}}$ can only arise from a redundancy in active stage constraints over several stages) and of similar conditioning as the original problem; a significant worsening of the conditioning can only appear in block-rows with active stage constraints, which, in a typical MPC setting, tend to appear rather on the earlier than on the later stages, and thus enter later in Algorithm 4.2 compared to the standard Cholesky factorization.

To formalize this, we identify the reverse Cholesky factorization with the discrete time Riccati recursion in the following. Let us regard the (possibly regularized) Newton Hessian $\boldsymbol{\mathcal{M}} \succ \mathbf{0}$ in block form as defined in Equation (4.8). Then, the reverse Cholesky factorization (Algorithm 4.2) is easily seen to be given by the recursion

$$\begin{aligned} \boldsymbol{X_{k-1}} &= \boldsymbol{W_{k-1}} - \boldsymbol{U_{k-1}} \cdot \boldsymbol{X_k}^{-1} \cdot \boldsymbol{U_{k-1}}^\top \\ \boldsymbol{X_N} &= \boldsymbol{W_N}, \end{aligned} \tag{4.14}$$

where the Cholesky factor $\boldsymbol{\mathcal{R}}$ in block form is given by

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} \boldsymbol{\mathcal{R}_{1,1}} & \boldsymbol{\mathcal{R}_{1,2}} & & \\ & \boldsymbol{\mathcal{R}_{2,2}} & \ddots & \\ & & \ddots & \boldsymbol{\mathcal{R}_{N-1,N}} \\ & & & \boldsymbol{\mathcal{R}_{N,N}} \end{bmatrix}$$

with upper triangular blocks $\boldsymbol{\mathcal{R}_{k,k}}$ given implicitly (but uniquely) by $\boldsymbol{X_k} =: \boldsymbol{\mathcal{R}_{k,k}} \boldsymbol{\mathcal{R}_{k,k}}^\top \quad \forall\, k \in \mathcal{S}_0$ and dense blocks $\boldsymbol{\mathcal{R}_{k,k+1}} = \boldsymbol{U_k}\, \boldsymbol{\mathcal{R}_{k+1,k+1}} \quad \forall\, k \in \mathcal{S}_{0,N}$. Note that in this context the subscripts $\boldsymbol{\mathcal{R}_{k,k}}$ refer to the block entries of $\boldsymbol{\mathcal{R}}$ rather than to the individual entries (as used in Algorithm 4.2). We refer to $\boldsymbol{X_k}$ as Cholesky iterates in the following.

For LTI systems without active constraints it holds (cf. Lemma 4.12)

$$\begin{aligned} \boldsymbol{W_k} &= \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top + \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^\top \\ \boldsymbol{U_k} &= -\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{C}^\top \end{aligned}$$

for $k \in \mathcal{S}_{0,N}$, where — analogously to the notation of (LMPC) — $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \end{bmatrix}$ are the dynamics, $\boldsymbol{E} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$ is the state selection matrix, and $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S}^\top \\ \boldsymbol{S} & \boldsymbol{R} \end{bmatrix}$

is the quadratic objective weight. Due to a possibly different choice of the Hessian on the last interval ($\boldsymbol{H} \equiv \boldsymbol{P}$), it holds

$$\boldsymbol{W_N} = \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top + \boldsymbol{P}^{-1}. \tag{4.15}$$

Theorem 4.15 *If $\boldsymbol{P}$ is the solution to the discrete time algebraic Riccati equation*

$$\boldsymbol{P} = \boldsymbol{Q} + \boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{A} - (\boldsymbol{S} + \boldsymbol{A}^\top\boldsymbol{P}\boldsymbol{B})\left(\boldsymbol{R} + \boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B}\right)^{-1}(\boldsymbol{S}^\top + \boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{A}),$$

then the Cholesky iterates $\boldsymbol{X}$ are constant, i.e., recursion (4.14) is stationary. In particular, it holds $\boldsymbol{X_k} := \boldsymbol{P}^{-1} + \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top = \boldsymbol{W_N} \quad \forall\, k \in \mathcal{S}_0$.

Proof The proof is done by calculation. We start from the the Cholesky recursion property, apply the assumed relation between Cholesky and Riccati iterates, and transform the expression into the form of the Riccati recursion. We have

$$\begin{aligned}
&\boldsymbol{X_{k-1}} = \boldsymbol{W_{k-1}} - \boldsymbol{U_{k-1}}\boldsymbol{X_k}^{-1}\boldsymbol{U_{k-1}}^\top \\
\Leftrightarrow\quad &\boldsymbol{P}^{-1} + \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top = \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top + \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^\top \\
&\qquad\qquad - \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{C}^\top\left(\boldsymbol{P}^{-1} + \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top\right)^{-1}\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{E}^\top
\end{aligned}$$

and therefore

$$\boldsymbol{P}^{-1} = \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^\top - \boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{C}^\top\left(\boldsymbol{P}^{-1} + \boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top\right)^{-1}\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{E}^\top. \tag{4.16}$$

Using the Schur complement $\bar{\boldsymbol{Q}} = \boldsymbol{Q} - \boldsymbol{S}^\top\boldsymbol{R}^{-1}\boldsymbol{S}$ it is well known from elementary linear algebra that the inverse $\boldsymbol{H}^{-1}$ can be expressed by

$$\boldsymbol{H}^{-1} = \begin{bmatrix}\boldsymbol{Q} & \boldsymbol{S}^\top \\ \boldsymbol{S} & \boldsymbol{R}\end{bmatrix}^{-1} = \begin{bmatrix}\bar{\boldsymbol{Q}}^{-1} & -\bar{\boldsymbol{Q}}^{-1}\boldsymbol{S}^\top\boldsymbol{R}^{-1} \\ -\boldsymbol{R}^{-1}\boldsymbol{S}\bar{\boldsymbol{Q}}^{-1} & \boldsymbol{R}^{-1} + \boldsymbol{R}^{-1}\boldsymbol{S}\bar{\boldsymbol{Q}}^{-1}\boldsymbol{S}^\top\boldsymbol{R}^{-1}\end{bmatrix}.$$

Using this, $\boldsymbol{C} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\end{bmatrix}$, and the special structure of $\boldsymbol{E} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}$, we first see that the identities

$$\boldsymbol{E}\boldsymbol{H}^{-1}\boldsymbol{E}^\top = \bar{\boldsymbol{Q}}^{-1},$$

$$\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{E}^\top = \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}\right)\bar{\boldsymbol{Q}}^{-1} =: \bar{\boldsymbol{C}}\bar{\boldsymbol{Q}}^{-1}$$

and

$$\begin{aligned}
\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\top &= \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}\right)\bar{\boldsymbol{Q}}^{-1}\left(\boldsymbol{A}^\top - \boldsymbol{S}^\top\boldsymbol{R}^{-1}\boldsymbol{B}^\top\right) + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\top \\
&= \bar{\boldsymbol{C}}\bar{\boldsymbol{Q}}^{-1}\bar{\boldsymbol{C}}^\top + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^\top
\end{aligned}$$

hold. Thus, (4.16) can be written as

$$\boldsymbol{P}^{-1} = \bar{\boldsymbol{Q}}^{-1} - \bar{\boldsymbol{Q}}^{-1}\bar{\boldsymbol{C}}^{\top}\left(\boldsymbol{P}^{-1} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top} + \bar{\boldsymbol{C}}\bar{\boldsymbol{Q}}^{-1}\bar{\boldsymbol{C}}^{\top}\right)^{-1}\bar{\boldsymbol{C}}\bar{\boldsymbol{Q}}^{-1}$$

Applying the Woodbury matrix identity with $\boldsymbol{Y} := \boldsymbol{P}^{-1} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}$ we can express the right hand side term by

$$\boldsymbol{P}^{-1} = \left(\bar{\boldsymbol{Q}} + \bar{\boldsymbol{C}}^{\top}\boldsymbol{Y}^{-1}\bar{\boldsymbol{C}}\right)^{-1}$$

and thus

$$\begin{aligned}\boldsymbol{P} &= \bar{\boldsymbol{Q}} + \bar{\boldsymbol{C}}^{\top}\boldsymbol{Y}^{-1}\bar{\boldsymbol{C}}\\ &= \boldsymbol{Q} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{S}\\ &\qquad + \left(\boldsymbol{A}^{\top} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\right)\left(\boldsymbol{P}^{-1} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\right)^{-1}\left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}\right)\end{aligned}$$

holds. Note that the inverse matrices of $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ and $\boldsymbol{Y}$ exist (and are real-valued), since we assumed (PQP) strictly convex. Applying the Woodbury identity once again (however, in opposite direction) on $\left(\boldsymbol{P}^{-1} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\right)^{-1}$, and introducing $\bar{\boldsymbol{R}} := \left(\boldsymbol{R} + \boldsymbol{B}^{\top}\boldsymbol{P}\boldsymbol{B}\right)$, we get

$$\begin{aligned}\boldsymbol{P} &= \boldsymbol{Q} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{S}\\ &\quad + \left(\boldsymbol{A}^{\top} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\right)\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\left(\boldsymbol{R} + \boldsymbol{B}^{\top}\boldsymbol{P}\boldsymbol{B}\right)^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}\right)\\ &= \boldsymbol{Q} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{S} \qquad (4.17)\\ &\qquad + \boldsymbol{A}^{\top}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\boldsymbol{A}\\ &\qquad\quad - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\boldsymbol{A}\\ &\qquad\quad\quad - \boldsymbol{A}^{\top}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}\\ &\qquad\quad\quad\quad + \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}.\end{aligned}$$

Using the identity $\mathbf{I} = \bar{\boldsymbol{R}}\bar{\boldsymbol{R}}^{-1} = \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{R}}$, we further have

$$\boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}^{\top}\left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}\right)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^{\top}\boldsymbol{R}^{-1}\boldsymbol{S} =$$

$$
\begin{aligned}
&= \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{S} \\
&= \boldsymbol{S}^\top \boldsymbol{R}^{-1}\bar{\boldsymbol{R}}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&\quad - \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{R}}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= \boldsymbol{S}^\top \boldsymbol{R}^{-1}\left(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\right)\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&\quad - \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\left(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\right)\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{R}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} + \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&\quad - \boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&\quad\quad - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{S} - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= -\boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\boldsymbol{S}
\end{aligned}
$$

and

$$
\begin{aligned}
&- \boldsymbol{A}^\top \left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\right)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} = \\
&= -\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= -\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{R}}\boldsymbol{R}^{-1}\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= -\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\left(\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\right)\boldsymbol{R}^{-1}\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= -\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&\quad\quad - \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S} \\
&= -\boldsymbol{A}^\top \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{S}.
\end{aligned}
$$

Analogously it holds

$$
-\boldsymbol{S}^\top \boldsymbol{R}^{-1}\boldsymbol{B}^\top \left(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\right)\boldsymbol{A} = -\boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1}\boldsymbol{B}^\top \boldsymbol{P}\boldsymbol{A}.
$$

Therefore (4.17) is equivalent to

$$\begin{aligned}\boldsymbol{P} &= \boldsymbol{Q} - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1} \boldsymbol{S} \\ &\quad - \boldsymbol{S}^\top \bar{\boldsymbol{R}}^{-1} \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A} - \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B} \bar{\boldsymbol{R}}^{-1} \boldsymbol{S} + \boldsymbol{A}^\top \left(\boldsymbol{P} - \boldsymbol{P} \boldsymbol{B} \bar{\boldsymbol{R}}^{-1} \boldsymbol{B}^\top \boldsymbol{P} \right) \boldsymbol{A} \\ &= \boldsymbol{Q} + \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} - \left(\boldsymbol{S}^\top + \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{B} \right) \bar{\boldsymbol{R}}^{-1} \left(\boldsymbol{S} + \boldsymbol{B}^\top \boldsymbol{P} \boldsymbol{A} \right),\end{aligned}$$

which concludes the proof. □

This proof is easily seen to also extend to the LTV case without active constraints, where we have (cf. Lemma 4.12):

$$\begin{aligned}\boldsymbol{W}_{\boldsymbol{k}} &= \boldsymbol{C}_{\boldsymbol{k-1}} {\boldsymbol{H}_{\boldsymbol{k-1}}}^{-1} {\boldsymbol{C}_{\boldsymbol{k-1}}}^\top + \boldsymbol{E}_{\boldsymbol{k}} {\boldsymbol{H}_{\boldsymbol{k}}}^{-1} {\boldsymbol{E}_{\boldsymbol{k}}}^\top && \forall k \in \mathcal{S}_{0,N} \\ \boldsymbol{U}_{\boldsymbol{k}} &= -\boldsymbol{E}_{\boldsymbol{k}} \boldsymbol{H}_{\boldsymbol{k}}^{-1} {\boldsymbol{C}_{\boldsymbol{k}}}^\top && \forall k \in \mathcal{S}_{0,N} \\ \boldsymbol{W}_{\boldsymbol{N}} &= \boldsymbol{C}_{\boldsymbol{N-1}} {\boldsymbol{H}_{\boldsymbol{N-1}}}^{-1} {\boldsymbol{C}_{\boldsymbol{N}}}^\top + {\boldsymbol{H}_{\boldsymbol{N}}}^{-1},\end{aligned}$$

with $\boldsymbol{C}_{\boldsymbol{k}} = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{k}} & \boldsymbol{B}_{\boldsymbol{k}} \end{bmatrix}$, $\forall\, k \in \mathcal{S}_N$, $\boldsymbol{E}_{\boldsymbol{k}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$, $\forall\, k \in \mathcal{S}_N$, $\boldsymbol{H}_{\boldsymbol{k}} = \begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{k}} & {\boldsymbol{S}_{\boldsymbol{k}}}^\top \\ \boldsymbol{S}_{\boldsymbol{k}} & \boldsymbol{R}_{\boldsymbol{k}} \end{bmatrix}$, $\forall\, k \in \mathcal{S}_N$, and $\boldsymbol{H}_{\boldsymbol{N}} = \boldsymbol{Q}_{\boldsymbol{N}}$.

Corollary 4.16 *Let* $\boldsymbol{W}_{\boldsymbol{k}}$, $k \in \mathcal{S}_0$ *and* $\boldsymbol{U}_{\boldsymbol{k}}$, $k \in \mathcal{S}_{0,N}$ *be computed from an LTV system without (active) state constraints. Then, the Cholesky iterates* $\boldsymbol{X}_{\boldsymbol{k}}$, $k \in \mathcal{S}_0$ *from the Cholesky recursion* (4.14) *can be identified with the discrete time time-varying algebraic Riccati recursion*

$$\boldsymbol{P}_{\boldsymbol{N}} = \boldsymbol{Q}_{\boldsymbol{N}} \tag{4.18a}$$

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{k-1}} &= \boldsymbol{Q}_{\boldsymbol{k-1}} + {\boldsymbol{A}_{\boldsymbol{k-1}}}^\top \boldsymbol{P}_{\boldsymbol{k}} \boldsymbol{A}_{\boldsymbol{k-1}} \\ &\quad - \left(\boldsymbol{S}_{\boldsymbol{k-1}} + {\boldsymbol{A}_{\boldsymbol{k-1}}}^\top \boldsymbol{P}_{\boldsymbol{k}} \boldsymbol{B}_{\boldsymbol{k-1}} \right) \left(\boldsymbol{R}_{\boldsymbol{k-1}} + {\boldsymbol{B}_{\boldsymbol{k-1}}}^\top \boldsymbol{P}_{\boldsymbol{k}} \boldsymbol{B}_{\boldsymbol{k-1}} \right)^{-1} \\ &\qquad \left({\boldsymbol{S}_{\boldsymbol{k-1}}}^\top + {\boldsymbol{B}_{\boldsymbol{k-1}}}^\top \boldsymbol{P}_{\boldsymbol{k}} \boldsymbol{A}_{\boldsymbol{k-1}} \right)\end{aligned} \tag{4.18b}$$

via $\boldsymbol{X}_{\boldsymbol{k}} = {\boldsymbol{P}_{\boldsymbol{k}}}^{-1} + \boldsymbol{C}_{\boldsymbol{k-1}} {\boldsymbol{H}_{\boldsymbol{k-1}}}^{-1} {\boldsymbol{C}_{\boldsymbol{k-1}}}^\top \quad \forall\, k \in \mathcal{S}_0$.

Proof The proof follows exactly the lines of the proof to Theorem 4.15, yet keeping the matrix block indices. In particular, one has

$$\begin{aligned}\boldsymbol{P_{k-1}}^{-1} = \boldsymbol{E_{k-1}}\boldsymbol{H_{k-1}}^{-1}\boldsymbol{E_{k-1}}^{\top} \\ - \boldsymbol{E_{k-1}}\boldsymbol{H_{k-1}}^{-1}\boldsymbol{C_{k-1}}^{\top}\left(\boldsymbol{P_k}^{-1} + \boldsymbol{C_{k-1}}\boldsymbol{H_{k-1}}^{-1}\boldsymbol{C_{k-1}}^{\top}\right)^{-1} \\ \boldsymbol{C_{k-1}}\boldsymbol{H_{k-1}}^{-1}\boldsymbol{E_{k-1}}^{\top}\end{aligned}$$

in place of (4.16) and transforms it into (4.18b) using the same matrix identities. □

Remark 4.17 In the context of MHE it may actually be advantageous to keep the regular Cholesky factorization order. Even though it is still open whether it is possible to establish an analogy to Theorem 4.15 and Corollary 4.16 for the inverted factorization order in notion of estimation, e.g., based on a Kalman-like recursion, we may draw the motivation to do so from the nature of the RTI scheme. Since new data is embedded at the end of the considered horizon, we expect tentatively more active set changes here, while we expect the active sets of the stage problems to change rather little (at least in the nominal case) at the beginning of the horizon, where several Newton-type iterations drove the optimization variables already close to an exact solution for the nonlinear problem. Therefore, a higher effectiveness of the factorization warm-starts may be given in the regular Cholesky factorization order starting from the bottom-right matrix blocks.

4.3.6 Choice of the Newton step size

Due to the piecewise quadratic nature of $f^*(\boldsymbol{\lambda})$, a globalization strategy is needed. For computational efficiency close to the solution, where we assume the quadratic model of the dual function $f^*(\boldsymbol{\lambda})$ to be accurate, we propose to employ a line search technique to find an (approximate) solution to

$$\alpha^i := \underset{0\leq\alpha\leq 1}{\arg\max}\, f^*(\boldsymbol{\lambda^i} + \alpha\, \boldsymbol{\Delta\lambda^i})\,. \tag{4.19}$$

In contrast to general nonsmooth optimization, an exact line search is possible at reasonable cost in our context. In particular, $f^*(\boldsymbol{\lambda^i}+\alpha\,\boldsymbol{\Delta\lambda})$ is a one-dimensional piecewise quadratic function along the search direction $\boldsymbol{\Delta\lambda}$. An exact quadratic model in search direction can be built up by evaluating each $f_k^*(\boldsymbol{\lambda^i} + \alpha_j\boldsymbol{\Delta\lambda})$, $k \in \mathcal{S}$ at each value of $\alpha_j \in [0,1]$ that corresponds to an active set change on

this stage. Alongside, slope information in search direction can be obtained by

$$\begin{aligned}\frac{\partial f_k^*(\boldsymbol{\lambda^i} + \alpha_j\, \boldsymbol{\Delta\lambda})}{\partial \alpha} &= \frac{\boldsymbol{\partial f_k^*}}{\boldsymbol{\partial \lambda}}(\boldsymbol{\lambda^i} + \alpha_j\, \boldsymbol{\Delta\lambda}) \cdot \boldsymbol{\Delta\lambda} \\ &= \boldsymbol{z_k^*}(\boldsymbol{\lambda^i} + \alpha_j \boldsymbol{\Delta\lambda})^\top \left(\boldsymbol{C_k}^\top \boldsymbol{\Delta\lambda_{k+1}} - \boldsymbol{E_k}^\top \boldsymbol{\Delta\lambda_k} \right) + \boldsymbol{c_k}^\top \boldsymbol{\Delta\lambda_{k+1}},\end{aligned}$$

cf. Equation (4.10). As the second derivate is constant within each cell, it can be cheaply and accurately computed as the difference quotient of the slope at the left and the right side of the intersection of each region with the search direction $\boldsymbol{\Delta\lambda}$.

Note that a parametric active set strategy like qpOASES traverses the required points $\boldsymbol{z_k^*}(\boldsymbol{\lambda^i} + \alpha_j\, \boldsymbol{\Delta\lambda})$ for values of α_j corresponding to an active-set change on stage k naturally while computing $\boldsymbol{z_k^*}(\boldsymbol{\lambda^i} + \boldsymbol{\Delta\lambda})$ for the full step $\boldsymbol{\Delta\lambda}$ (cf. [FBD08]). When employing the clipping operation (4.6) for the solution of the stage problems (QP_k), the points of active-set changes along the search direction can analogously be determined by a simple ratio test.

With the piecewise quadratic model in search direction built up on each stage $k \in \mathcal{S}$, the dual objective value can be cheaply evaluated at each α_j, where an active set change occurs on *any* stage $k \in \mathcal{S}$. Taking the maximum over all these values (e.g., by performing a bisection search) identifies, in conjunction with the slope information, the region containing the maximum dual function value in search direction, and the optimal α^* can be found as the maximum of the one-dimensional quadratic model of this region.

Alternatively, also heuristic backtracking-based search strategies seem appropriate in this context. Particularly the fact that the gradient evaluation $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})$ comes almost at the same cost as a function evaluation $f^*(\boldsymbol{\lambda})$ can be exploited within the line search. A search strategy that seemed to perform particularly well in practice was a combination of a fast backtracking line search, allowing to quickly detect very small step sizes, with a bisection interval search for refinement.

While we make use of a backtracking line search to quickly decrease the maximum step size $\alpha_{\max}$, the minimum step size $\alpha_{\min}$ is given by the minimum scaling of the search direction that leads to an active set change on any stage. While this is intuitively clear, as each region with a constant active set is quadratic and Newton's method takes a step towards the minimum of a local quadratic function approximation, we give a formal proof for this in the following section, in the context of convergence (Lemma 4.19). This guaranteed minimum step size is indicated by $\boldsymbol{\lambda^i_{\alpha_{\min}}}$ in Figure 4.1.

Remark 4.18 As mentioned above, we obtain all α-values at which active set changes occur at no extra cost when employing an online active set strategy to solve each (QP_k). Therefore, taking a simple minimum over these α-values over all stages $k \in \mathcal{S}$ provides us with the lower bound $\alpha_{\min}$. If the solution to (QP_k) is computed by Equation (4.6), points of active-set changes can still be obtained cheaply by comparing the unconstrained to the clipped solution in each component.

4.4 Finite Convergence of the Algorithm

Convergence of non-smooth Newton methods has been proven before for functions with similar properties [QS93, LS97, GK08]. In the following we show convergence in our specific setting; this allows us to present a shorter, more straightforward proof. Based on these results we further establish an infeasibility detection mechanism in Section 4.5, which according to our knowledge is novel to QP solvers based on nonsmooth Newton methods.

A bit of notation is needed throughout this and the following section. By $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ we refer to the Hessian matrix of (DQP), as defined in Equations (4.8-4.9). The possibly regularized version of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ used in the solution step of the Newton system (4.5) is denoted by $\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda})$. We will omit the dependency on $\boldsymbol{\lambda}$ occasionally for notational convenience when it is clear from the context. The active set of stage constraints at $\boldsymbol{\lambda}$ is denoted by $\mathcal{A}^*(\boldsymbol{z}^*(\boldsymbol{\lambda}))$.

We start with a rather obvious result, that nonetheless is crucial for the practical performance of the Dual Newton Strategy.

Lemma 4.19 (Local one-step convergence) *Let* (PQP) *be feasible. Let* $\boldsymbol{\lambda}^{\boldsymbol{i}}$ *be the current dual iterate in Algorithm 4.1. Let* $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ *be positive definite, i.e., no regularization is needed during Step 7 in Algorithm 4.1, and let* $\boldsymbol{\Delta\lambda}$ *be the solution of the Newton equation* (4.5). *Then, if* $\mathcal{A}^*(\boldsymbol{z}^*(\boldsymbol{\lambda}^{\boldsymbol{i}})) = \mathcal{A}^*(\boldsymbol{z}^*(\boldsymbol{\lambda}^{\boldsymbol{i}} + \boldsymbol{\Delta\lambda}))$, *it holds that* $\boldsymbol{\lambda}^{\boldsymbol{i+1}} := \boldsymbol{\lambda}^{\boldsymbol{i}} + \boldsymbol{\Delta\lambda}$ *solves Problem* (DQP). *In particular it holds*

$$\arg\max_{0 \le \alpha \le 1} f^*(\boldsymbol{\lambda}^{\boldsymbol{i}} + \alpha\, \boldsymbol{\Delta\lambda}) = 1. \tag{4.20}$$

Proof Recall that f^* is piecewise quadratic in $\boldsymbol{\lambda}$ by Lemma 4.8. By the construction in Lemma 4.12 we know that $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ is constant within each region $A(\boldsymbol{\lambda}^{\boldsymbol{i}})$, since the active set is fixed. By its definition, the Newton step $\boldsymbol{\Delta\lambda}$ points to the maximum of the quadratic function characterizing $A(\boldsymbol{\lambda}^{\boldsymbol{i}})$. By concavity of f^* it follows that $\boldsymbol{\lambda}^{\boldsymbol{i}} + \boldsymbol{\Delta\lambda}$ has to be the maximum of f^* and thus solves (DQP). The claim (4.20) follows immediately. □

Lemma 4.19 is applied twofold in the dual Newton strategy. First, it allows us to make our line search smarter by only considering step sizes that lead to at least one active set change (or full steps) as mentioned above in Section 4.3.6. Second, it shows that once the correct region of the solution is identified we have a one-step convergence to the exact solution (up to numerical accuracy), cf. Section 4.2.4. Next, we show global convergence.

Theorem 4.20 (Global convergence) *Let* (PQP) *be feasible. Let* $\boldsymbol{\lambda^0} \in \mathbb{R}^{Nn_x}$ *and let* $\boldsymbol{\lambda^i} \in \mathbb{R}^{Nn_x}$ *be defined recursively by* $\boldsymbol{\lambda^{i+1}} := \boldsymbol{\lambda^i} + \alpha^i \boldsymbol{\Delta\lambda^i}$, *where* $\boldsymbol{\Delta\lambda^i}$ *is the (possibly regularized) solution to Equation* (4.5)*, and* α^i *is the solution to Equation* (4.19)*. Then the sequence* $\{\boldsymbol{\lambda^i}\}_{i\in\mathbb{N}_0} \subset \mathbb{R}^{Nn_x}$ *converges to the unique maximum* $\hat{\boldsymbol{\lambda}}$ *with* $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) = \mathbf{0}$.

Proof The sequence $\{\boldsymbol{\lambda^i}\}_{i\in\mathbb{N}_0}$ induces a sequence $\{f^i := f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0} \subseteq \mathbb{R}$. By definition of the exact line search (4.19) it holds that $f^{i+1} \geq f^i$, i.e., $\{f^i\}_{i\in\mathbb{N}_0}$ is monotonously increasing. Since (PQP) is feasible, $f^*(\boldsymbol{\lambda})$ is a bounded, concave function by Lemma 4.8 and duality theory. By the Bolzano-Weierstrass Theorem $\{f^i\}_{i\in\mathbb{N}_0}$ thus converges to an accumulation point $\hat{f}$.

Due to monotonicity of $\{f^i\}_{i\in\mathbb{N}_0}$ it holds that $\{\boldsymbol{\lambda^i}\}_{i\in\mathbb{N}_0}$ is contained in the superlevel set

$$\mathcal{F} := \{\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x} \mid f^*(\boldsymbol{\lambda}) \geq f^*(\boldsymbol{\lambda^0})\},$$

which is compact since $f^*(\boldsymbol{\lambda})$ is a bounded concave function. A convergent subsequence $\{\boldsymbol{\lambda^{i^{(1)}}}\} \subseteq \{\boldsymbol{\lambda^i}\}_{i\in\mathbb{N}_0}$ therefore has to exist and its limit $\hat{\boldsymbol{\lambda}}$ fulfills $f^*(\hat{\boldsymbol{\lambda}}) = \hat{f}$ due to the induced monotonicity of $f^*(\boldsymbol{\lambda^{i^{(1)}}})$.

What remains to show is that $\hat{\boldsymbol{\lambda}}$ indeed maximizes $f^*(\boldsymbol{\lambda})$, i.e. $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) = \mathbf{0}$. Assume contrarily $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) \neq \mathbf{0}$. Since $\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})$ is strictly positive definite and due to the fixed regularization even bounded away from 0 in norm, it holds that $\hat{\boldsymbol{\Delta\lambda}} = \tilde{\boldsymbol{\mathcal{M}}}(\hat{\boldsymbol{\lambda}})^{-1}\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) \neq \mathbf{0}$, cf. Equation (4.5), is an ascent direction. Then $\hat{\alpha} > 0$ holds for the solution of Equation (4.19), and by C^1-continuity of $f^*(\boldsymbol{\lambda})$ we can conclude that there is a $\delta > 0$ with

$$f^*(\hat{\boldsymbol{\lambda}} + \hat{\alpha}\,\hat{\boldsymbol{\Delta\lambda}}) \geq f^*(\hat{\boldsymbol{\lambda}}) + \delta.$$

Since $\{\boldsymbol{\lambda^{i^{(1)}}}\}$ converges to $\hat{\boldsymbol{\lambda}}$, an index $\bar{i} \in \mathbb{N}$ exists, such that for all $i^{(1)} \geq \bar{i}$ we have $\boldsymbol{\Delta\lambda^{i^{(1)}}}$ sufficiently close to $\hat{\boldsymbol{\Delta\lambda}}$ and $\boldsymbol{\lambda^{i^{(1)}}}$ close enough to $\hat{\boldsymbol{\lambda}}$ such that

$$f^*(\boldsymbol{\lambda^{i^{(1)}}} + \alpha^{i^{(1)}}\,\boldsymbol{\Delta\lambda^{i^{(1)}}}) \geq f^*(\boldsymbol{\lambda^{i^{(1)}}} + \hat{\alpha}\,\boldsymbol{\Delta\lambda^{i^{(1)}}}) \geq f^*(\hat{\boldsymbol{\lambda}}) + \delta/2,$$

where the first inequality holds by the maximum property of the line search in each iteration, and the second by continuity of $f^*(\boldsymbol{\lambda})$. This, however, would be a contradiction to $\hat{\boldsymbol{\lambda}}$ being an accumulation point of a monotonously increasing sequence, so $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) = \mathbf{0}$, and our claim holds. □

Lemma 4.21 *Let $z^*(\lambda^*)$ be a feasible solution for (PQP), that fulfills the LICQ. Then $\mathcal{M}(\lambda^*)$ is strictly positive definite.*

Proof From Lemma 4.12 we have that $\mathcal{M}(\lambda^*) = \mathcal{C}\,\mathcal{P}(\lambda^*)\,\mathcal{C}^\top$, where

$$\mathcal{P}(\lambda^*) = \mathcal{Z}^*(\mathcal{Z}^{*\top}\mathcal{H}\,\mathcal{Z}^*)^{-1}\mathcal{Z}^{*\top}$$

with $\mathcal{Z}^* := \text{block diag}(Z_0^*, Z_1^*, \ldots, Z_N^*)$ and $\mathcal{H} := \text{block diag}(H_0, H_1, \ldots, H_N)$. As in Lemma 4.12, each Z_k^*, $k \in \mathcal{S}$ denotes a basis matrix for the active constraints in the solution of (QP_k), in this context the solution given the subproblem parameter λ^*. Consider now

$$\lambda^\top \mathcal{M}(\lambda^*)\,\lambda = \lambda^\top \mathcal{C}\,\mathcal{Z}^*(\mathcal{Z}^{*\top}\mathcal{H}\,\mathcal{Z}^*)^{-1}\mathcal{Z}^{*\top}\mathcal{C}^\top\lambda. \tag{4.21}$$

Since $\mathcal{H}$ is positive definite and $\mathcal{Z}^*$, being a block diagonal composition of basis matrices, has full column rank, we have $\mathcal{Z}^{*\top}\mathcal{H}\,\mathcal{Z}^* \succ \mathbf{0}$, and thus $(\mathcal{Z}^{*\top}\mathcal{H}\,\mathcal{Z}^*)^{-1} \succ \mathbf{0}$. Using equation (4.21), this implies $\lambda^\top \mathcal{M}(\lambda^*)\,\lambda \geq 0$.

Assume $\lambda^\top \mathcal{M}(\lambda^*)\,\lambda = 0$. Since $(\mathcal{Z}^{*\top}\mathcal{H}\,\mathcal{Z}^*)^{-1} \succ \mathbf{0}$ this means $\lambda^\top \mathcal{C}\,\mathcal{Z}^* = \mathbf{0}$ has to hold.

The columns of $\mathcal{Z}^*$ however are linearly independent and span the nullspace of the active stage constraints, i.e., every vector from the nullspace of $\mathcal{Z}^*$ can be expressed by a linear combination of active stage constraints. If now $\lambda^\top\mathcal{C}$ lies in the nullspace of the active stage constraints this means there is a linear combination of active stage constraints that represents $\lambda^\top\mathcal{C}$, which is a linear combination of the stage coupling equality constraints. Since LICQ holds we can conclude that $\lambda = \mathbf{0}$, and thus $\mathcal{M}(\lambda^*) \succ \mathbf{0}$ holds. □

Corollary 4.22 (Finite termination of Algorithm 4.1) *Let $z^*(\lambda^*)$ be a feasible solution for (PQP) that fulfills the LICQ. Let $\lambda^0, \lambda^1, \ldots$ be computed from Algorithm 4.1 (in exact arithmetic). Then $\{\lambda^i\}_{i\in\mathbb{N}_0}$ becomes stationary after finitely many iterations, i.e., $\exists\,\bar{i}\ :\ \lambda^i = \lambda^*\ \forall\, i \geq \bar{i}$.*

Proof From Lemma 4.6 we know that $z_k^*(\lambda)$ depends continuously on λ. If no stage constraints of (QP_k) are weakly active in λ^*, then λ^* lies in the strict interior of a region A^* (with $\mathcal{M}(\lambda)$ constant on A^*) in the dual λ space. According to Theorem 4.20 there is a finite iteration index $\bar{i} < \infty$ with $\lambda^{\bar{i}} \in A^*$. Due to Lemma 4.21 we have that $\mathcal{M}(\lambda)$ is non-singular on A^*, and Lemma 4.19 guarantees convergence in the next iteration.

If there are weakly active constraints in λ^* (i.e., λ^* lies on the boundary between several, but a finite number of, nonempty regions $A^{(j_1)}, \ldots, A^{(j_n)}$ in Figure 4.1), then due to C^1 continuity of f^*, each quadratic function defining f^* on $A^{(j_i)}$, $i \in \{1, \ldots, n\}$ needs to have its maximum in λ^*. We can define a

ball $B_\epsilon(\boldsymbol{\lambda}^*)$ of fixed radius $\epsilon > 0$ around $\boldsymbol{\lambda}^*$ with the property that for every $\boldsymbol{\lambda} \in B_\epsilon(\boldsymbol{\lambda}^*)$ the dual function f^* on a region $A^{(j_i)}$ containing $\boldsymbol{\lambda}$ is again defined by a quadratic function having its maximum in $\boldsymbol{\lambda}^*$. By the identical argument as before we can conclude finite termination of Algorithm 4.1 due to Theorem 4.20, Lemma 4.21, and Lemma 4.19. □

4.5 Infeasibility Handling

If the primal problem (PQP) is infeasible, two possible consequences for the dual (DQP) arise. If a stage k and a selection of stage constraints $\left\{(\underline{\boldsymbol{d}}_{\boldsymbol{k}})_{\boldsymbol{i}} \leq (\boldsymbol{D}_{\boldsymbol{k}})_{\boldsymbol{i},\cdot}\boldsymbol{z}_{\boldsymbol{k}} \leq (\overline{\boldsymbol{d}}_{\boldsymbol{k}})_{\boldsymbol{i}}\right\}_{i\in\mathcal{I}_k}$, $\mathcal{I}_k \subseteq \{1,...,n_d\}$ exists that cannot be satisfied by any $\boldsymbol{z}_{\boldsymbol{k}} \in \mathbb{R}^{n_z}$, then and only then the dual problem (DQP) is infeasible as well, as no choice of $\boldsymbol{\lambda}$ will render (QP_k) feasible (see also Remark 4.1).

Otherwise, if all (QP_k) are feasible and yet (PQP) is infeasible, we have the following Lemma:

Lemma 4.23 *Let all* (QP_k) *be feasible. If* (PQP) *is infeasible, the dual function* $f^*(\boldsymbol{\lambda})$ *is unbounded and there exists (at least) one region in* $\boldsymbol{\lambda}$ *space,* $\emptyset \neq A^{\inf} \subseteq \mathbb{R}^{Nn_x}$*, with constant* $\boldsymbol{\mathcal{M}}_{\mathbf{inf}}$ *on* $A^{\inf}$ *and*

i) $\boldsymbol{\mathcal{M}}_{\mathbf{inf}}$ *singular, i.e.,* $\exists\, \boldsymbol{\lambda} \neq \mathbf{0} \;:\; \boldsymbol{\mathcal{M}}_{\mathbf{inf}}\boldsymbol{\lambda} = \mathbf{0}$,

ii) $\forall\, \bar{f} \in \mathbb{R}\ \exists\, \hat{\boldsymbol{\lambda}} \in A^{\inf} \;:\; f^*(\hat{\boldsymbol{\lambda}}) > \bar{f}$,

iii) for all $A^{(j)}$*, defined by a representative* $\boldsymbol{\lambda}^{(\boldsymbol{j})}$*, with* $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{(\boldsymbol{j})}) \succ \mathbf{0}$ *it holds*

$$\exists\, \hat{\boldsymbol{\lambda}} \in A^{\inf}\ \forall\, \bar{\boldsymbol{\lambda}} \in A^{(j)} \;:\; f^*(\hat{\boldsymbol{\lambda}}) > f^*(\bar{\boldsymbol{\lambda}}).$$

Proof Since all (QP_k) are feasible, $f^*(\boldsymbol{\lambda})$ exists and (DQP) is feasible. Thus, (DQP) has to be unbounded by duality theory.

Let $\boldsymbol{\lambda}^{(\boldsymbol{j})} \in A^{(j)}$ for any $A^{(j)}$. The mapping $\boldsymbol{\lambda} \to f^*(\boldsymbol{\lambda})$ is onto an interval that contains the half-open interval $[f^*(\boldsymbol{\lambda}^{(\boldsymbol{j})}), +\infty)$, since $f^*(\boldsymbol{\lambda})$ is continuous and unbounded. Since there is only a finite number of regions by Definition 4.4, property (ii) holds.

Assume now (i) is violated, i.e., $\exists\, A^{(j)}$ with $\boldsymbol{\mathcal{M}}(\cdot) \succ \mathbf{0}$ on $A^{(j)}$ and $\forall\, \bar{f} \in \mathbb{R}\ \exists\, \boldsymbol{\lambda} \in A^{(j)} : f^*(\boldsymbol{\lambda}) > \bar{f}$. Since $\boldsymbol{\mathcal{M}}(\cdot) \succ \mathbf{0}$ is constant on $A^{(j)}$ we have that $f^*(\boldsymbol{\lambda})$ is strictly, and even strongly concave on $A^{(j)}$. Therefore $\exists\, \bar{f} < \infty$ with $f^*(\boldsymbol{\lambda}) \leq \bar{f}\ \forall\, \boldsymbol{\lambda} \in \mathcal{A}^{(j)}$, a contradiction. Therefore (i) holds.

In particular $f^*(\boldsymbol{\lambda})$ is bounded (from above) on each $A^{(j)}$ with $\boldsymbol{\mathcal{M}}(\cdot) \succ \mathbf{0}$. Since there is only a finite number of regions, property (ii) implies (iii). □

Lemma 4.23 tells us that unboundedness of the dual objective function value can only occur in regions with singular Newton Hessian Matrix $\boldsymbol{\mathcal{M}}(\cdot)$. We further characterize these unbounded regions in the following.

Definition 4.24 (Infinite ray) For $\boldsymbol{\Delta\lambda} \neq \mathbf{0}$ we call a pair $(\bar{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ an *infinite ray*, if there is a region $A^{\inf} \subseteq \mathbb{R}^{Nn_x}$ represented by $\bar{\boldsymbol{\lambda}} \in A^{\inf}$ such that

1. the ray is contained in the region: $\bar{\boldsymbol{\lambda}} + \delta\, \boldsymbol{\Delta\lambda} \in A^{\inf} \quad \forall\, \delta > 0$,
2. the ray is in the nullspace of the region's Hessian: $\boldsymbol{\mathcal{M}}(\bar{\boldsymbol{\lambda}})\, \boldsymbol{\Delta\lambda} = \mathbf{0}$,
3. the ray is an ascent direction: $\boldsymbol{\mathcal{G}}(\bar{\boldsymbol{\lambda}})^\top \boldsymbol{\Delta\lambda} > 0$.

Clearly every region containing an infinite ray is an unbounded region in the sense of Lemma 4.23. We additionally have the following characterizations.

Remark 4.25 As $\boldsymbol{\Delta\lambda}$ lies in the nullspace of the negated dual Hessian $\boldsymbol{\mathcal{M}}(\cdot)$, the dual gradient $\boldsymbol{\mathcal{G}}(\cdot)$ along a ray $(\bar{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ is constant.

Lemma 4.26 *Let $(\bar{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ be an infinite ray contained in $A^{\inf}$. For every $\tilde{\boldsymbol{\lambda}} \in A^{\inf}$ it holds $\tilde{\boldsymbol{\lambda}} + \gamma\, \boldsymbol{\Delta\lambda} \in A^{\inf} \quad \forall\, \gamma > 0$, i.e., $(\tilde{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ is an infinite ray as well.*

Proof Since $A^{\inf}$ is convex by Lemma 4.7 and both $\tilde{\boldsymbol{\lambda}} \in A^{\inf}$ and $\bar{\boldsymbol{\lambda}} + \gamma\, \boldsymbol{\Delta\lambda} \in A^{\inf}$ for each choice of $\gamma > 0$, it holds

$$\beta\, \tilde{\boldsymbol{\lambda}} + (1 - \beta) \cdot (\bar{\boldsymbol{\lambda}} + \gamma\, \boldsymbol{\Delta\lambda}) \;\in A^{\inf} \quad \forall\, \beta \in [0, 1]. \tag{4.22}$$

From Lemma 4.7 we also have that $A^{\inf}$ is polyhedral and therefore closed in $\overline{\mathbb{R}}^{Nn_x}$, where $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$. Thus, the limit of (4.22) for $\gamma \to \infty$ is contained in $A^{\inf}$, and the claim holds. □

Lemma 4.26 is interesting as it tells us that $A^{\inf}$ has a cone- or beam-like shape. This will play a role in the following, when we characterize certificates for an unbounded dual problem. We have the following definition and theorem:

Definition 4.27 (Ridge) Let $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ be an infinite ray with $\boldsymbol{\Delta\lambda}^\dagger \neq \mathbf{0}$. We call $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ a *ridge* if it holds $\boldsymbol{\Delta\lambda}^\dagger = \gamma\, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$ for a $\gamma > 0$, i.e., if $\boldsymbol{\Delta\lambda}^\dagger$ is aligned with the gradient of the dual function f^* along the ray it is defining. ⌟

Theorem 4.28 *Let all* (QP_k) *be feasible. If* (PQP) *is infeasible, then a ridge $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ exists.*

Proof From Lemma 4.23 we know that a non-empty region A^{inf} exists. We further know that $f^*(\boldsymbol{\lambda})$ is an unbounded concave quadratic function on A^{inf}. An infinite ray $(\bar{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ in the sense of Definition 4.24 therefore has to exist in A^{inf}. From Lemma 4.26 we have that in this case $(\tilde{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ is an infinite ray as well for every $\tilde{\boldsymbol{\lambda}} \in A^{\text{inf}}$.

Let us regard $A^{\text{inf}}_{\cup} := A^{\text{inf},1} \cup A^{\text{inf},2} \cup \ldots \cup A^{\text{inf},n}$, the union of all unbounded regions in the sense of Lemma 4.23. Clearly for all $\bar{f} \in \mathbb{R}$ the superlevel set

$$\bar{A}^{\text{inf}}_{\cup} := \{\boldsymbol{\lambda} \in A^{\text{inf}}_{\cup} \mid f^*(\boldsymbol{\lambda}) \geq \bar{f}\} \tag{4.23}$$

is convex, as $f^*(\boldsymbol{\lambda})$ is concave. Since $f^*(\boldsymbol{\lambda})$ is continuous and unbounded on $\bar{A}^{\text{inf}}_{\cup}$, it holds that $\bar{A}^{\text{inf}}_{\cup}$ is (Nn_x)-dimensional, i.e., full-dimensional; otherwise a directional vector $\boldsymbol{e} \in \mathbb{R}^{Nn_x}$ would exist with $\tilde{\boldsymbol{\lambda}} \in \text{int}(\bar{A}^{\text{inf}}_{\cup})$, i.e., $f^*(\tilde{\boldsymbol{\lambda}}) > \bar{f}$ and $f^*(\tilde{\boldsymbol{\lambda}} + \epsilon\, \boldsymbol{e}) < \bar{f} \;\; \forall\, \epsilon > 0$, a violation of continuity.

Assume for the moment that every region $A^{\text{inf},j}$ only contains one infinite ray (up to translation and scaling). Consider the nonempty intersection of $\bar{A}^{\text{inf}}_{\cup}$ with a $(Nn_x - 1)$-dimensional Hyperplane $F \subset \mathbb{R}^{Nn_x}$. If $F \cap \bar{A}^{\text{inf}}_{\cup}$ does not contain a singular ray, $f^*(\boldsymbol{\lambda})$ has to be bounded on $F \cap \bar{A}^{\text{inf}}_{\cup}$, as $f^*(\boldsymbol{\lambda})$ is composed from only a finite number of concave quadratic functions. In particular $f^*(\boldsymbol{\lambda})$ attains a maximum $\hat{\boldsymbol{\lambda}}$ somewhere in the intersection.

This maximum $\hat{\boldsymbol{\lambda}}$ is characterized by the fact that the dual gradient $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}})$ is orthogonal to F if $\hat{\boldsymbol{\lambda}} \in \text{int}(F \cap \bar{A}^{\text{inf}}_{\cup})$. Since the dual $\boldsymbol{\lambda}$ space is (Nn_x)-dimensional $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}})/\|\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}})\|$ is uniquely defined by F and vice versa. Recall that $(\hat{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ is an infinite ray in every $\hat{\boldsymbol{\lambda}} \in \bar{A}^{\text{inf}}_{\cup}$ for one fixed $\boldsymbol{\Delta\lambda}$ as shown above. Since $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})$ is continuous, varying (i.e., "rotating") F eventually has to yield a $\hat{\boldsymbol{\lambda}}$ with $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}}) = \gamma\, \boldsymbol{\Delta\lambda}$ for some $\gamma > 0$.

If now there is a region A^{inf}_j with more than one infinite ray (i.e., the directional vectors $\boldsymbol{\Delta\lambda}$ of the infinite rays span a space of dimensionality $k > 1$) the same argument can be applied, but with a $(Nn_x - k)$-dimensional Hyperplane $F \subset \mathbb{R}^{Nn_x}$. In this case $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}})$ is not uniquely defined anymore, but lies in the normal space of F. By the same argument as above there has to be an F whose normal space coincides with the space spanned by the directional vectors $\boldsymbol{\Delta\lambda}$ of the infinite rays of A^{inf}_j, and thus again there is an infinite ray $(\hat{\boldsymbol{\lambda}}, \boldsymbol{\Delta\lambda})$ coinciding with the gradient $\boldsymbol{\mathcal{G}}(\hat{\boldsymbol{\lambda}})$ at its base point $\hat{\boldsymbol{\lambda}}$. □

Remark 4.29 Let $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)) \subseteq A^\dagger \subseteq \bar{A}^{\text{inf}}_{\cup}$ a ridge. Clearly, $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger)$ is rank deficient (and constant on $A^\dagger$) and thus regularized with $\delta\, \mathbf{I} \succ \mathbf{0}$ by Algorithm 4.1. By Theorem 4.28 and Definitions 4.24 and 4.27 it holds $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger)\, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) = \mathbf{0}$

and we have

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) = \frac{1}{\delta}\,\delta\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) + \frac{1}{\delta}\underbrace{\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger)\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)}_{=0} = \frac{1}{\delta}\,(\underbrace{\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger) + \delta\,\mathbf{I}}_{\succ 0})\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$$

$$\Leftrightarrow \quad \tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda}^\dagger)^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) = \frac{1}{\delta}\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$$

$$\Leftrightarrow \qquad \boldsymbol{\Delta\lambda}^\dagger = \frac{1}{\delta}\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger),$$

i.e., Algorithm 4.1 only performs gradient steps and thus remains on the ridge. Therefore we can see a ridge as the analogon to a fixed point in the case where the dual function $f^*(\boldsymbol{\lambda})$ is unbounded.

Lemma 4.30 (Minimality of the ridge gradient) *Let $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ be a ridge. Then*

$$\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)\|_2 \leq \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})\|_2 \tag{4.24}$$

for all $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$. Inequality (4.24) is furthermore strict except for those $\boldsymbol{\lambda}$ that lie on a ridge, i.e., for which there is a ridge $(\bar{\boldsymbol{\lambda}}, \overline{\boldsymbol{\Delta\lambda}})$ such that $\boldsymbol{\lambda} = \bar{\boldsymbol{\lambda}} + \gamma\,\overline{\boldsymbol{\Delta\lambda}}$ for a $\gamma \geq 0$. In particular the ridge $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger) \subseteq A^\dagger$ is unique up to scaling of $\boldsymbol{\Delta\lambda}^\dagger$ and translations from the nullspace of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger)$.

Proof Let $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$ and let $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ be a ridge. Due to concavity of $f^*(\cdot)$ we have $(\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger))^\top(\boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}) \geq 0$. Since $\boldsymbol{\mathcal{G}}(\cdot)$ is constant along the ridge, $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$, and $\boldsymbol{\Delta\lambda}^\dagger$ and $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$ only differ by a positive scalar factor, we also have

$$(\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger))^\top \frac{1}{\gamma}(\boldsymbol{\lambda}^\dagger + \gamma\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) - \boldsymbol{\lambda}) \geq 0$$

for all $\gamma > 0$. Since concavity of $f^*(\cdot)$ clearly also holds on the extended domain $\overline{\mathbb{R}}^{Nn_x}$, where $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, we have $(\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger))^\top \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) \geq 0$ in the limit for $\gamma \to \infty$. This is equivalent to

$$\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)\|_2^2 = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)^\top \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) \leq \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})^\top \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger) \leq \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})\|_2 \cdot \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)\|_2,$$

where the last inequality is the Cauchy-Schwarz inequality. Thus (4.24) holds.

For the proof of the second part of the Lemma, we note that the Cauchy-Schwarz inequality is strict unless $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})$ is a positive multiple of $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$, so $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})\|_2 = \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)\|_2$ implies $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$. Let us therefore consider an arbitrary $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$ with $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$. Due to concavity we have

$$\begin{aligned} f^*(\beta\,\boldsymbol{\lambda} + (1-\beta)\,\boldsymbol{\lambda}^\dagger) &\leq f^*(\boldsymbol{\lambda}) + \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})^\top(\beta\,\boldsymbol{\lambda} + (1-\beta)\,\boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}) \\ &= f^*(\boldsymbol{\lambda}) + (1-\beta)\cdot\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})^\top(\boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}) \end{aligned} \tag{4.25}$$

for $\beta \in [\,0\,,1\,]$, and analogously

$$f^*(\beta\,\boldsymbol{\lambda} + (1-\beta)\,\boldsymbol{\lambda}^\dagger) \leq f^*(\boldsymbol{\lambda}^\dagger) - \beta \cdot \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)^\top(\boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}). \tag{4.26}$$

We can multiply (4.25) by β and (4.26) by $(1-\beta)$, add both inequalities up, and, using $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$, yield

$$f^*(\beta\,\boldsymbol{\lambda} + (1-\beta)\,\boldsymbol{\lambda}^\dagger) \leq \beta\,f^*(\boldsymbol{\lambda}) + (1-\beta)\,f^*(\boldsymbol{\lambda}^\dagger)$$

Due to concavity of $f^*(\cdot)$ also the converse inequality holds and we have

$$f^*(\beta\,\boldsymbol{\lambda} + (1-\beta)\,\boldsymbol{\lambda}^\dagger) = \beta\,f^*(\boldsymbol{\lambda}) + (1-\beta)\,f^*(\boldsymbol{\lambda}^\dagger), \tag{4.27}$$

i.e., linearity of $f^*(\cdot)$ on the interval between $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^\dagger$. Once more since $\boldsymbol{\mathcal{G}}(\cdot)$ is constant along the ridge $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$, (4.27) holds analogously for all $\tilde{\boldsymbol{\lambda}}^\dagger := \boldsymbol{\lambda}^\dagger + \gamma\,\boldsymbol{\Delta\lambda}^\dagger$ in place of $\boldsymbol{\lambda}^\dagger$, i.e. f^* is linear on the interval $[\,\boldsymbol{\lambda}\,,\,\boldsymbol{\lambda}^\dagger + \gamma\,\boldsymbol{\Delta\lambda}^\dagger\,]$ for all choices of $\gamma > 0$. Since the space of linear functions from $\mathbb{R}^{Nn_x}$ to $\mathbb{R}$ is closed, linearity also holds in the limit for $\gamma \to \infty$, which is the half-open interval $\{\boldsymbol{\lambda} + \gamma \cdot \boldsymbol{\Delta\lambda}^\dagger \mid \gamma \in [\,0\,,\infty\,)\}$. Since $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^\dagger)$ (which itself is a positive multiple of $\boldsymbol{\Delta\lambda}^\dagger$) we therefore have that $(\boldsymbol{\lambda}, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}))$ is itself a ridge, which is moreover parallel to $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$.

Since $f^*(\cdot)$ is specifically linear on $[\,\boldsymbol{\lambda}\,,\,\boldsymbol{\lambda}^\dagger\,]$, $(\boldsymbol{\lambda}, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}))$ differs from $(\boldsymbol{\lambda}^\dagger, \boldsymbol{\Delta\lambda}^\dagger)$ only by a shift that lies in the nullspace of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^\dagger)$ as we claimed above. □

Theorem 4.31 (Convergence to an infinite ray) *If* (PQP) *is infeasible, then exactly one of the two following statements is true:*

1. (QP_k) *is infeasible for at least one* $k \in \mathcal{S}$
2. *Algorithm 4.1 converges to an infinite ray if the regularization parameter* δ *is chosen sufficiently large*

Proof Let (PQP) be infeasible. Due to the special time coupling structure of (PQP) either there exists a minimal infeasible set (a selection of constraints of minimal size that cannot be fulfilled at the same time) that is contained in the set of local stage constraints of one (QP_k), or all minimal infeasible sets consist of local stage constraints (PQP3) of several stages $k_1 < k_2 < \ldots < k_n$ and the corresponding time coupling constraints between k_1 and k_n (a subset of Constraints (PQP2)). In the former case Statement 1 holds, and infeasibility is detected by the stage QP solver on first execution. In the latter case we have that the partial dual function $f^*(\boldsymbol{\lambda})$ exists and can be evaluated. In particular $f^*(\boldsymbol{\lambda})$ is unbounded by Lemma 4.23. In the remainder of the proof we show that in this case Algorithm 4.1 indeed converges to an infinite ray.

As we have seen in the proof to Theorem 4.20, the iterates $\boldsymbol{\lambda^i}$ of Algorithm 4.1 defined by Equations (4.4), (4.5), and (4.19) induce a monotonously increasing sequence $\{f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0}$. We claim that $\{f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0}$ does not converge if $f^*(\boldsymbol{\lambda})$ is unbounded.

Assume contrarily that $\{f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0}$ converges. Clearly $\boldsymbol{\mathcal{G}(\lambda)}$ does not vanish since $\boldsymbol{\mathcal{G}(\lambda)} = \mathbf{0}$ would imply the existence of a feasible solution of (PQP) by Remark 4.11. By Lemma 4.30 we further know the existence of a

$$0 < G_{\min} := \min_{\lambda\in\mathbb{R}^{Nn_x}} \|\boldsymbol{\mathcal{G}(\lambda)}\|. \tag{4.28}$$

If now $\{f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0}$ converges, clearly the updates $\alpha^i \boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$, have to vanish. Since there are only finitely many different $\boldsymbol{\tilde{\mathcal{M}}}(\cdot) \succ \mathbf{0}$ and $\boldsymbol{\mathcal{G}}(\cdot)$ is bounded away from 0 by Equation (4.28) this implies that

$$\alpha^i = \underset{0\le\alpha\le 1}{\arg\max}\, f^*(\boldsymbol{\lambda^i} + \alpha\boldsymbol{\Delta\lambda})$$

with $\boldsymbol{\Delta\lambda} = \boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$, cf. Equation (4.19), has to drop to 0. At each iteration i three cases for α^i could appear:

i) $\alpha^i = 1$.

ii) $\alpha^i = 0$. We have $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \neq \mathbf{0}$ and $\boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i}) \succ \mathbf{0}$, so $\boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$ is an ascent direction. By C^1-continuity of $f^*(\cdot)$ an ascent is possible and thus $\alpha^i \neq 0 \;\; \forall i \in \mathbb{N}_0$.

iii) $0 < \alpha^i < 1$. Then by maximality α^i fulfills $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i} + \alpha^i\boldsymbol{\Delta\lambda})^\top\boldsymbol{\Delta\lambda} = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i} + \alpha^i\boldsymbol{\Delta\lambda})^\top\boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) = 0$. Regard

$$\phi(\alpha) := \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i} + \alpha\,\boldsymbol{\Delta\lambda})^\top\boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$$

as a function in α. Clearly $\phi(0) = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\boldsymbol{\tilde{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$ is bounded away from 0, since there are only finitely many different $\boldsymbol{\tilde{\mathcal{M}}}(\cdot) \succ \mathbf{0}$ and $\boldsymbol{\mathcal{G}}(\cdot)$ is bounded away from $\mathbf{0}$ by Equation (4.28). Further $\phi(\alpha)$ is continuous and its derivative $\phi'(\alpha) = -\boldsymbol{\Delta\lambda}^\top\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i} + \alpha\,\boldsymbol{\Delta\lambda})\boldsymbol{\Delta\lambda}$ is bounded from below (for a fixed $\boldsymbol{\Delta\lambda}$), again since there are only finitely many different $\boldsymbol{\tilde{\mathcal{M}}}(\cdot)$; note that the ascent direction $\boldsymbol{\Delta\lambda}$ cannot grow to infinity for $i \to \infty$ since $f^*(\boldsymbol{\lambda})$ is concave and $f^*(\boldsymbol{\lambda^i})$ is increasing. Therefore α^i has to be bounded away from 0 by a problem-data specific constant that is independent from the iteration index i.

Since also α^i does not converge to 0 we have a contradiction, and our claim that $\{f^*(\boldsymbol{\lambda^i})\}_{i\in\mathbb{N}_0}$ diverges (even monotonously) holds true. In particular every fixed

function value $\bar{f}$ will be exceeded and for every suitably large $\bar{f}$ the iterates of Algorithm 4.1 will remain in $\bar{A}_{\cup}^{\inf}$ (as defined in Equation (4.23)) after a finite number of iterations.

Yet in the remainder of the proof we first show local and subsequently global attractiveness of a ridge in the sense of Theorem 4.28.

Let $A^{\dagger}$ denote the region that contains a ridge $(\boldsymbol{\lambda}^{\dagger}, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\dagger}))$. For two subsequent iterates both contained in $A^{\dagger}$ (characterized by a constant Newton Matrix $\boldsymbol{\mathcal{M}}$) we have the exact linear model

$$\begin{aligned}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i+1}}) &= \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}} + \alpha^i \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})) \\ &= \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) - \alpha^i \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}). \end{aligned} \tag{4.29}$$

We now claim that $\alpha^i = 1$ if $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i+1}}) \in \mathcal{A}^{\dagger}$, which is true by the following reasoning: by the argumentation in (ii) above, clearly $\alpha^i > 0$. If $\alpha^i < 1$, by optimality of the line search it would need to hold

$$\begin{aligned} & & \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}} + \alpha^i \boldsymbol{\Delta\lambda})^{\top} \boldsymbol{\Delta\lambda} = 0 \\ \Leftrightarrow & & \left(\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) - \alpha^i \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\right)^{\top} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) = 0 \\ \Leftrightarrow & & \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})^{\top} \left(\mathbf{I} - \alpha^i \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1}\right)^{\top} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) = 0. \end{aligned}$$

Since $\tilde{\boldsymbol{\mathcal{M}}}^{-1} \succ \mathbf{0}$ and $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) \neq \mathbf{0}$ by Equation (4.28), this could only be true if $\mathbf{I} - \alpha^i \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1}$ is semidefinite or indefinite. However, we have $\mathbf{I} - \alpha^i \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} = (\tilde{\boldsymbol{\mathcal{M}}} - \alpha^i \boldsymbol{\mathcal{M}}) \tilde{\boldsymbol{\mathcal{M}}}^{-1} = ((1 - \alpha^i) \boldsymbol{\mathcal{M}} + \delta \mathbf{I}) \tilde{\boldsymbol{\mathcal{M}}}^{-1} \succ \mathbf{0}$ for $\alpha^i < 1$ (recall that $\boldsymbol{\mathcal{M}}$ is singular on $A^{\dagger}$ by assumption, hence $\tilde{\boldsymbol{\mathcal{M}}} = \boldsymbol{\mathcal{M}} + \delta \mathbf{I}$), a contradiction, and thus $\alpha^i = 1$ holds.

Next, note that due to the cone-like quadratic shape of $A^{\dagger}$ (cf. Lemma 4.26) each subsequent iterate $\boldsymbol{\lambda}^{\boldsymbol{i+1}}$ is contained in $A^{\dagger}$ if $\boldsymbol{\lambda}^{\boldsymbol{i}} \in A^{\dagger}$. The linear expansion of the gradient, (4.29), becomes $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i+1}}) = \left(\mathbf{I} - \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1}\right) \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$. Clearly $\mathbf{I} - \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} = \mathbf{I} - \left(\tilde{\boldsymbol{\mathcal{M}}} - \delta \mathbf{I}\right) \tilde{\boldsymbol{\mathcal{M}}}^{-1} = \delta \tilde{\boldsymbol{\mathcal{M}}}^{-1}$ is positive definite. On the other hand $\boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} = \mathbf{I} - \delta \tilde{\boldsymbol{\mathcal{M}}}^{-1}$ is positive semidefinite, so $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i+1}})\| \leq \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\|$.

Whenever $\boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) \neq \mathbf{0}$ we have $\delta \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})^{\top} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{M}} \tilde{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) > 0$ for any $\delta > 0$. This can be easily verified, e.g., by using the fact that $\mathbf{0} \preceq \boldsymbol{\mathcal{M}} =: \boldsymbol{B}\boldsymbol{B}^{\top}$ has a (not necessarily full rank) symmetric factorization. Assuming

$\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \neq \mathbf{0}$ it then holds[3]

$$\begin{aligned}
& & \delta\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) > 0 \\
\Leftrightarrow \quad & & \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \left(\mathbf{I} - \boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\right)\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) > 0 \\
\Leftrightarrow \quad & \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) > 0 \\
\Rightarrow \quad & 2\cdot\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) > 0.
\end{aligned}$$

This implies that

$$\begin{aligned}
\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^{i+1}})\|^2 &= \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \left(\mathbf{I} - \boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\right)^\top \left(\mathbf{I} - \boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\right)\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \\
&= \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|^2 - 2\cdot\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \\
&\qquad + \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \\
&< \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|^2
\end{aligned}$$

for all iterates with $\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \neq \mathbf{0}$, and together with (4.29) we can conclude that the sequence of gradients converges to a limit $\boldsymbol{\mathcal{G}}^*$ that fulfills

$$\begin{aligned}
& \boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}^* = \mathbf{0} \\
\Leftrightarrow \quad & \left(\tilde{\boldsymbol{\mathcal{M}}} - \delta\,\mathbf{I}\right)\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}^* = \mathbf{0} \\
\Leftrightarrow \quad & \delta\,\tilde{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}^* = \boldsymbol{\mathcal{G}}^*\,,
\end{aligned}$$

the ridge property. Recall that by Lemma 4.30 the ridge is unique.

It remains to show that the iterates $\boldsymbol{\lambda^i}$ of Algorithm (4.1) also converge globally to $A^\dagger$. To see this, we consider the auxiliary function

$$\tilde{f}(\boldsymbol{\lambda}) := f^*(\boldsymbol{\lambda}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})^\top\boldsymbol{\lambda}.$$

This function is clearly piecewise quadratic and concave, just like f^*, and its second derivative equals the second derivative of f^*, while the first derivative of $\tilde{f}$ is given by $\frac{\partial\tilde{f}}{\partial\boldsymbol{\lambda}} = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})^\top - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})^\top$. It is furthermore bounded, since due to concavity we have for every $\boldsymbol{\lambda} \in \mathbb{R}^{Nn_x}$

$$\tilde{f}(\boldsymbol{\lambda}) \leq \tilde{f}(\boldsymbol{\lambda^\dagger}) + (\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger}))^\top(\boldsymbol{\lambda} - \boldsymbol{\lambda^\dagger}) = \tilde{f}(\boldsymbol{\lambda^\dagger}).$$

[3] We make use of the symmetry and positive semidefiniteness of $\boldsymbol{\mathcal{M}}\tilde{\boldsymbol{\mathcal{M}}}^{-1} = \mathbf{I} - \delta\,\tilde{\boldsymbol{\mathcal{M}}}^{-1}$.

The iterates $\boldsymbol{\lambda^i}$ of Algorithm (4.1) are increasing in $\tilde{f}$ iff the step directions $\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$ are always ascent directions, i.e., if

$$\alpha^i \left(\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) - \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})\right)^\top \tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \overset{!}{>} 0$$

holds. We have shown above that α^i is bounded away from 0 by an iteration-independent constant and therefore we equivalently have the condition

$$\delta\, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \overset{!}{>} \delta\, \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})^\top \tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}), \tag{4.30}$$

where δ is the regularization parameter in Algorithm 4.1. Since $\delta\,\tilde{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda^i})^{-1}$ is positive definite, each of the finitely many distinct values $\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}$ defines a scalar product through $\delta\,\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}$, that induces a norm on $\mathbb{R}^{Nn_x}$. Applying the Cauchy-Schwarz inequality to (4.30) we have that the iterates $\boldsymbol{\lambda^i}$ are increasing if

$$\begin{aligned} & \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|^2_{\delta\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}} \overset{!}{>} \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})\|_{\delta\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}} \cdot \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|_{\delta\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}} \\ \Leftrightarrow \quad & \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|_{\delta\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}} \overset{!}{>} \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})\|_{\delta\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1}}. \end{aligned} \tag{4.31}$$

While $\boldsymbol{\lambda^i}$ is not contained in a ridge, we have $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})\|_2 > \|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^\dagger})\|_2$ from Lemma 4.30; furthermore $\mathbf{I} - \delta\,\tilde{\boldsymbol{\mathcal{M}}}_{(\boldsymbol{j})}^{-1} = \mathbf{I} - \delta\,(\boldsymbol{\mathcal{M}}_{(\boldsymbol{j})} + \delta\,\mathbf{I})^{-1}$ vanishes for sufficiently large choices of δ, fulfilling (4.31) and thus showing that the iterates $\boldsymbol{\lambda^i}$ are also ascending in the auxiliary function $\tilde{f}$. Since $\tilde{f}$ is bounded, we can conclude that the region $A^\dagger$, containing a ridge, is eventually reached, thus concluding the proof of global convergence. □

Theorem 4.31 can algorithmically be used to detect infeasibility of (PQP). If any (QP_k) is infeasible, it will be detected by the stage QP solver (either qpOASES, or during the clipping operation) on the first execution and (PQP) is immediately known to be infeasible. Otherwise, in the case of infeasibility through the coupling constraints, we know by Theorem 4.31 that the iterates of Algorithm 4.1 will eventually converge to a ridge. Here, we could gradually increase the regularization parameter δ if the iterates remain in regions with singular Hessians. A check whether $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$ is in the nullspace of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i})$ (i.e., $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda^i})\,\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \approx \mathbf{0}$) every few (regularized) iterations without an active-set change, combined with a check whether any active-set change occurs at all in the Newton direction will eventually conclude infeasibility. Note that the check for active-set changes in the Newton direction can cheaply be performed both in qpOASES and the clipping QP solver by simply considering the signs in the

ratio test of the active and inactive constraints, cf. Section 4.3.6. Additionally, practical infeasibility can be concluded, when the objective function value exceeds a certain large threshold (caused by an explosion of the norm of the iterates $\boldsymbol{\lambda}^{\boldsymbol{i}}$).

Remark 4.32 In practice, the dual iterates $\boldsymbol{\lambda}^{\boldsymbol{i}}$ in Algorithm 4.1 were indeed always observed to grow very fast and reach a ridge quickly in infeasible problems due to the initially small regularization.

We note that the theoretical result of Theorem 4.31 is somewhat unsatisfactory, since it requires a modification of the regularization parameter δ (or even to only do gradient steps in regions with a singular Hessian). We initially aimed at proving Theorem 4.31 for generic regularized Newton steps, i.e., independent of δ. Despite some considerable effort, we could not come up with a formal argument that links Condition (4.31) with Lemma 4.30, since the ordering relations in the Euclidian norm on the one hand and the norm induced by $\delta\,\tilde{\boldsymbol{\mathcal{M}}}_{(j)}^{-1}$ on the other hand might be different. Still, we are confident that also general regularized Newton updates, independent of the choice of δ, are increasing in $\tilde{f}$ on a global scale (though not necessarily monotonously), and thus formulate the following conjecture.

Conjecture 4.33 *Theorem 4.31 holds for all choices of the regularization parameter $\delta > 0$.*

4.6 Concurrency in the Dual Newton Strategy

One important advantage of the Dual Newton Strategy is that, opposed to conventional active-set or interior-point methods, it is an easily parallelizable algorithm. Analyzing Algorithm 4.1 step by step in this respect, we observe that all stage QPs can be solved concurrently by $N+1$ threads in Step 2. Next, each block of the dual gradient $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda})$ only depends on the solution of two neighboring stage QPs (cf. Equation 4.7), and therefore the setup can be done concurrently by N threads (Step 3). Also in the setup of the symmetric Newton matrix $\boldsymbol{\mathcal{M}}$, Step 6, each diagonal block only depends on the solution of two adjacent stage QP solutions, while each off-diagonal block only depends on the solution of one stage QP; therefore the workload of the setup of $\boldsymbol{\mathcal{M}}$ can be distributed on N threads almost equally. During the line search procedure in Step 8 of Algorithm 4.1, the expensive steps in each iteration consist of solving all stage QPs for the new step size guess and computing the corresponding gradient, both of which can be spread over $N+1$ threads with almost equal workload.

Therefore, all steps of significant workload can already be run fully in parallel in each major iteration of Algorithm 4.1, except for the solution of the structured linear system, Step 7. In a sequential implementation, solving the system based on a structure-exploiting reverse Cholesky factorization as proposed in Section 4.3.5 seems most efficient. This algorithm, however, is high level completely serial and therefore cannot be expected to scale well with the number of available computing threads on modern and forthcoming multi-core CPU architectures. We therefore propose a parallelizable solution algorithm for the structured Newton system in Section 4.6.1, as an alternative to Algorithm 4.2. The overall cost of this algorithm is roughly twice as high in number of floating point operations (FLOPs) as Algorithm 4.2, but it comes at a parallel complexity of only $\mathcal{O}(\log N)$ on N threads; therefore, the overall time complexity of one iteration of the dual Newton strategy comes down to only $\mathcal{O}(\log N)$ FLOPs, compared to $\mathcal{O}(N^3)$ in active-set methods, and $\mathcal{O}(N)$ in state-of-the-art tailored interior-point (IP) methods[4].

4.6.1 A parallel solution algorithm for the Newton system

The core idea behind the algorithm proposed in this section is to exploit the block-tridiagonal structure of the Newton matrix $\boldsymbol{\mathcal{M}}$ in Equation (4.5) in a cyclic reduction strategy. We note that a similar and to some extent more general algorithm has been proposed independently in [Wri91] for the parallel solution of a linear system of similar structure, that originated from a related but slightly different problem. The work [Wri91] contains a deeper analysis of the applicability and extensions of the cyclic reduction strategy, as well as extensive numerical comparisons. In the following we only present the central algorithmic idea in our context and refer to [Wri91] for complementary reading. Note that in contrast to the setting of [Wri91], we have general applicability, since $\boldsymbol{\mathcal{M}}$ is positive definite and therefore the diagonal minors are invertible.

We consider the Newton system given by

$$\begin{bmatrix} \boldsymbol{D_1} & \boldsymbol{U_1} & & & & & \\ \boldsymbol{U_1^\top} & \boldsymbol{D_2} & \boldsymbol{U_2} & & & & \\ & \boldsymbol{U_2^\top} & \boldsymbol{D_3} & \boldsymbol{U_3} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \boldsymbol{U_{N-2}^\top} & \boldsymbol{D_{N-1}} & \boldsymbol{U_{N-1}} & \\ & & & & \boldsymbol{U_{N-1}^\top} & \boldsymbol{D_N} & \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\lambda_1} \\ \boldsymbol{\lambda_2} \\ \boldsymbol{\lambda_3} \\ \boldsymbol{\lambda_4} \\ \vdots \\ \boldsymbol{\lambda_{N-1}} \\ \boldsymbol{\lambda_N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g_1} \\ \boldsymbol{g_2} \\ \boldsymbol{g_3} \\ \vdots \\ \boldsymbol{g_{N-1}} \\ \boldsymbol{g_N} \end{bmatrix} \tag{4.32}$$

[4] We note that, in principle, a computational complexity of $\mathcal{O}(\log N)$ can also be achieved for IP methods using the same algorithmic idea as stated below, cf. Remark 4.35.

where $\boldsymbol{\lambda_i} \in \mathbb{R}^{n_x},\ i \in \mathcal{S}_0$ denote the block components of $\boldsymbol{\lambda}$ defined in Equation (4.2). We assume the Newton matrix $\boldsymbol{\mathcal{M}}$ strictly positive definite (otherwise we regularize). We can use the $2^{\text{nd}}, 4^{\text{th}}, \dots$ equation of (4.32) to eliminate $\boldsymbol{\lambda_2}, \boldsymbol{\lambda_4}, \dots$ concurrently from (4.32). This yields a reduced-size system of equations, which is again block-tridiagonal,

$$\begin{bmatrix} \bar{\boldsymbol{D}}_{\boldsymbol{1}} & \bar{\boldsymbol{U}}_{\boldsymbol{1}} & & & & \\ \bar{\boldsymbol{U}}_{\boldsymbol{1}}^{\top} & \bar{\boldsymbol{D}}_{\boldsymbol{3}} & \bar{\boldsymbol{U}}_{\boldsymbol{3}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & \bar{\boldsymbol{U}}_{\boldsymbol{N-1}} & \\ & & & \bar{\boldsymbol{U}}_{\boldsymbol{N-1}}^{\top} & \bar{\boldsymbol{D}}_{\boldsymbol{N}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\lambda_1} \\ \boldsymbol{\lambda_3} \\ \boldsymbol{\lambda_5} \\ \vdots \\ \boldsymbol{\lambda_{N-1}} \\ \boldsymbol{\lambda_N} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{g}}_{\boldsymbol{1}} \\ \bar{\boldsymbol{g}}_{\boldsymbol{3}} \\ \bar{\boldsymbol{g}}_{\boldsymbol{5}} \\ \vdots \\ \bar{\boldsymbol{g}}_{\boldsymbol{N-1}} \\ \bar{\boldsymbol{g}}_{\boldsymbol{N}} \end{bmatrix},$$

with block components given by

$$\bar{\boldsymbol{D}}_{\boldsymbol{i}} := \boldsymbol{D_i} - \boldsymbol{U}_{\boldsymbol{i-}}^{\top}\, \boldsymbol{D}_{\boldsymbol{i-}}^{-\boldsymbol{1}}\, \boldsymbol{U_{i-}} - \boldsymbol{U_i}\, \boldsymbol{D}_{\boldsymbol{i+}}^{-\boldsymbol{1}}\, \boldsymbol{U}_{\boldsymbol{i}}^{\top} \tag{4.33a}$$

$$\bar{\boldsymbol{U}}_{\boldsymbol{i}} := -\boldsymbol{U}_{\boldsymbol{i}}^{\top}\, \boldsymbol{D}_{\boldsymbol{i+}}^{-\boldsymbol{1}}\, \boldsymbol{U_{i+}} \tag{4.33b}$$

$$\bar{\boldsymbol{g}}_{\boldsymbol{i}} := \boldsymbol{g_i} - \boldsymbol{U}_{\boldsymbol{i-}}^{\top}\, \boldsymbol{D}_{\boldsymbol{i-}}^{-\boldsymbol{1}}\, \boldsymbol{g_{i-}} - \boldsymbol{U}_{\boldsymbol{i}}^{\top}\, \boldsymbol{D}_{\boldsymbol{i+}}^{-\boldsymbol{1}}\, \boldsymbol{g_{i+}}, \tag{4.33c}$$

where $i- = i-1$, and $i+ = i+1$. Applying this reduction step recursively, we obtain a system

$$\begin{bmatrix} \bar{\bar{\boldsymbol{D}}}_{\boldsymbol{1}} & \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{1}} \\ \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{1}}^{\top} & \bar{\bar{\boldsymbol{D}}}_{\boldsymbol{N}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\lambda_1} \\ \boldsymbol{\lambda_N} \end{bmatrix} = \begin{bmatrix} \bar{\bar{\boldsymbol{g}}}_{\boldsymbol{1}} \\ \bar{\bar{\boldsymbol{g}}}_{\boldsymbol{N}} \end{bmatrix}$$

after $\Theta(\log N)$ iterations, from which we can eliminate $\boldsymbol{\lambda_N}$, yielding

$$\left(\bar{\bar{\boldsymbol{D}}}_{\boldsymbol{1}} - \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{1}}\, \bar{\bar{\boldsymbol{D}}}_{\boldsymbol{N}}^{-\boldsymbol{1}}\, \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{1}}^{\top}\right) \boldsymbol{\lambda_1} = \bar{\bar{\boldsymbol{g}}}_{\boldsymbol{1}} - \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{1}}\, \bar{\bar{\boldsymbol{D}}}_{\boldsymbol{N}}^{-\boldsymbol{1}} \bar{\bar{\boldsymbol{g}}}_{\boldsymbol{N}}, \tag{4.34}$$

a dense system of size $n_x \times n_x$. This system can be efficiently solved by a direct Cholesky decomposition, followed by two backsolve steps.

With $\boldsymbol{\lambda_0}$ we can recover $\boldsymbol{\lambda_N}$ from

$$\boldsymbol{\lambda_N} = \bar{\bar{\boldsymbol{D}}}_{\boldsymbol{N}}^{-\boldsymbol{1}} \left(\bar{\bar{\boldsymbol{g}}}_{\boldsymbol{N}} - \bar{\bar{\boldsymbol{U}}}_{\boldsymbol{0}}^{\top}\, \boldsymbol{\lambda_0}\right), \tag{4.35}$$

and in general, we can recover $\boldsymbol{\lambda_i}$ from $\boldsymbol{\lambda_{i-}}$ and $\boldsymbol{\lambda_{i+}}$ by

$$\boldsymbol{\lambda_i} = \boldsymbol{D}_{\boldsymbol{i}}^{-\boldsymbol{1}} \left(\boldsymbol{g_i} - \boldsymbol{U}_{\boldsymbol{i-}}^{\top}\, \boldsymbol{\lambda_{i-}} - \boldsymbol{U_i}\, \boldsymbol{\lambda_{i+}}\right), \tag{4.36}$$

concurrently in reverse level order of the previous elimination procedure. Here $\boldsymbol{\lambda_{i-}}$ and $\boldsymbol{\lambda_{i+}}$ denote the $\boldsymbol{\lambda}$-blocks preceding and succeeding $\boldsymbol{\lambda_i}$ in the system of equations remaining in the reduction step that eliminated $\boldsymbol{\lambda_i}$.

The complete solution algorithm can be summarized as follows:

Algorithm 4.3: A parallel solution algorithm for Equation (4.5)

Input: Newton system given by $\boldsymbol{g^{(0)}} = \left[\boldsymbol{g_0}^\top, \ldots, \boldsymbol{g_N}^\top\right]^\top$,
$\boldsymbol{D^{(0)}} = \text{block diag}(\boldsymbol{D_0}, \ldots, \boldsymbol{D_N})$, $\boldsymbol{U^{(0)}} = \text{block diag}(\boldsymbol{U_0}, \ldots, \boldsymbol{U_{N-1}})$

Output: Solution $\boldsymbol{\lambda} = \left[\boldsymbol{\lambda_0}^\top, \ldots, \boldsymbol{\lambda_N}^\top\right]^\top$ to Equation (4.5)

$k_{\max} = \lceil \log_2(N-1) \rceil$

for $k = 1 : k_{\max}$ **do** `/* factor step */`

  for $i = 2^{k-1} : 2^k : N-1$ **do in parallel**

    $i- = i - 2^{k-1}$

    $i+ = \min(i + 2^k, N)$

    compute $\boldsymbol{D_i^{(k)}}, \boldsymbol{U_i^{(k)}}, \boldsymbol{g_i^{(k)}}$ from Eq. (4.33) with
    $\boldsymbol{D_\star} = \boldsymbol{D_\star^{(k-1)}}$, $\boldsymbol{U_\star} = \boldsymbol{U_\star^{(k-1)}}$, $\boldsymbol{g_\star} = \boldsymbol{g_\star^{(k-1)}}$ $\quad \forall \star \in \{i, i-, i+\}$

Compute $\boldsymbol{\lambda_0}$ from Eq. (4.34) with $\boldsymbol{\bar{U}_0} = \boldsymbol{U_0^{(k_{\max})}}$, $\boldsymbol{\bar{g}_0} = \boldsymbol{g_0^{(k_{\max})}}$, $\boldsymbol{\bar{D}_0} = \boldsymbol{D_0^{(k_{\max})}}$, $\boldsymbol{\bar{D}_N} = \boldsymbol{D_N^{(k_{\max})}}$ using a Cholesky decomposition

Compute $\boldsymbol{\lambda_N}$ from Eq. (4.35) with $\boldsymbol{\bar{D}_N} = \boldsymbol{D_N^{(k_{\max})}}$, $\boldsymbol{\bar{g}_N} = \boldsymbol{g_N^{(k_{\max})}}$, $\boldsymbol{\bar{U}_0} = \boldsymbol{U_0^{(k_{\max})}}$

for $k = k_{\max} : -1 : 1$ **do** `/* solve step */`

  for $i = 2^{k-1} : 2^k : N-1$ **do in parallel**

    recover $\boldsymbol{\lambda_i}$ using Eq. (4.36) with with
    $\boldsymbol{D_i} = \boldsymbol{D_i^{(k-1)}}$, $\boldsymbol{g_i} = \boldsymbol{g_i^{(k-1)}}$, $\boldsymbol{U_{i-}} = \boldsymbol{U_{i-}^{(k-1)}}$, $\boldsymbol{U_i} = \boldsymbol{U_i^{(k-1)}}$

Remark 4.34 Obviously the products of the block matrices $\boldsymbol{U_\star}$ and $\boldsymbol{D_\star^{-1}}$ in Equations (4.33) and (4.34) are most efficiently computed by a backsolve with a Cholesky factor of the diagonal blocks of the reduced size system, $\boldsymbol{D_\star}$. If System (4.32) is linearly dependent, i.e., if $\boldsymbol{\mathcal{M}}$ is rank deficient, the Cholesky factorization of one of these blocks will fail (otherwise, i.e., if all $\boldsymbol{D_\star}$ have full rank, (4.34-4.36) would constitute an linear injective mapping $\boldsymbol{\lambda} \to \boldsymbol{\mathcal{G}}$), and we can restart the factorization with a regularized $\boldsymbol{\mathcal{M}}$, analogously to Section 4.3.4.

Remark 4.35 We note that Algorithm 4.3 can also be employed in the factorization step of tailored interior-point methods, thus obtaining also $\mathcal{O}(\log N)$ parallel time complexity versions of this class of methods. This has been established in [Le14][5] for a structure-exploiting variant of Mehrotra's predictor-corrector scheme [Meh92].

[5] Do Duc Le developed these results in his Bachelor's thesis under the supervision of Janick Frasch.

4.6.2 A partial condensing approach

So far, we discussed the application of the dual Newton strategy directly to band-structured QPs arising from dynamic optimization. The main advantage of the dual Newton approach compared to the classical approach consisting of a condensing step followed by the solution of the dense problem by an active-set method lies in the fact that the specific Newton system considered here can be computed (i.e., set up) and solved at a computational complexity that is linear in horizon length in a sequential implementation, and even log-linear in a parallel implementation. For certain problems with long horizons and many states, however, the solution of the Newton system may become overly expensive, as the bandwidth of the fill-in of the dual Hessian matrix grows proportionally to the number of states and the factorization grows cubically in the bandwidth. The condensing/active-set approach would benefit overproportionally from the reduced problem size in this case if the number of controls is small.

For other problems that feature very long prediction horizons, the backsolve step in the solution of the Newton system may actually become the computational bottleneck, if the Hessian factorization can be warm-started efficiently due to active-set changes appearing only on the first stages.

An alternative in these cases may be to combine the dual Newton strategy with a condensing approach. In detail we propose to apply the condensing algorithms given in Section 3.2 to aggregate the stage problems (QP_k) by groups $\{N_0, \dots, N_1\}, \{N_1 + 1, \dots, N_2\}, \dots, \{N_m, \dots, N_{m+1}\}$, where $0 =: N_0 < N_1 < N_2 < \dots < N_{m+1} := N$. The resulting, aggregated stage problems $\overline{\mathrm{QP}}_k$, where $k \in \{0, \dots, m\}$, are then given by[6]

$$\begin{aligned} \min_{\bar{\boldsymbol{z}}_{\boldsymbol{k}}} \ & \frac{1}{2} \bar{\boldsymbol{z}}_{\boldsymbol{k}}{}^{\top} \bar{\boldsymbol{H}}_{\boldsymbol{k}} \bar{\boldsymbol{z}}_{\boldsymbol{k}} + \left(\bar{\boldsymbol{g}}_{\boldsymbol{k}}{}^{\top} + \begin{bmatrix} \bar{\boldsymbol{\lambda}}_{\boldsymbol{k}} \\ \bar{\boldsymbol{\lambda}}_{\boldsymbol{k+1}} \end{bmatrix}^{\top} \begin{bmatrix} -\bar{\boldsymbol{E}}_{\boldsymbol{k}} \\ \bar{\boldsymbol{C}}_{\boldsymbol{k}} \end{bmatrix} \right) \bar{\boldsymbol{z}}_{\boldsymbol{k}} \\ \text{s.t.} \quad & \bar{\boldsymbol{d}}^{\mathrm{l}}_{\boldsymbol{k}} \leq \bar{\boldsymbol{D}}_{\boldsymbol{k}} \, \bar{\boldsymbol{z}}_{\boldsymbol{k}} \ \leq \ \bar{\boldsymbol{d}}^{\mathrm{u}}_{\boldsymbol{k}}. \end{aligned} \tag{$\overline{\mathrm{QP}}_k$}$$

The stage variables of the aggregated problems ($\overline{\mathrm{QP}}_k$) are

$$\bar{\boldsymbol{z}}_{\boldsymbol{k}} := (\boldsymbol{x}_{\boldsymbol{N_k}}, \boldsymbol{u}_{\boldsymbol{N_k}}, \boldsymbol{u}_{\boldsymbol{N_k+1}}, \dots, \boldsymbol{u}_{\boldsymbol{N_{k+1}-1}}).$$

Only the multipliers coupling the aggregated problems remain, and we have $\bar{\boldsymbol{\lambda}}_{\boldsymbol{k}} := \boldsymbol{\lambda}_{\boldsymbol{N_k}}$ for $k \in \{0, \dots, m+1\}$. The condensed QP data is computed analogously to Section 3.2, where we have $\bar{\boldsymbol{H}}_{\boldsymbol{k}} \cong \begin{bmatrix} \bar{\boldsymbol{Q}}^{\boldsymbol{e}} & \bar{\boldsymbol{S}}^{\boldsymbol{e}\top} \\ \bar{\boldsymbol{S}}^{\boldsymbol{e}} & \bar{\boldsymbol{R}} \end{bmatrix}$, $\bar{\boldsymbol{g}}_{\boldsymbol{k}} \cong \begin{bmatrix} \bar{\boldsymbol{q}}^{\boldsymbol{e}} \\ \bar{\boldsymbol{r}} \end{bmatrix}$,

[6] We ignore the constant objective function term in this context.

$\bar{\boldsymbol{D}}_{\boldsymbol{k}} \cong \begin{bmatrix} \bar{\boldsymbol{D}}^{\boldsymbol{e}} & \bar{\boldsymbol{D}} \end{bmatrix}$, $\bar{\boldsymbol{d}}^{\mathbf{l}}_{\boldsymbol{k}} \cong \bar{\boldsymbol{d}}$, and $\bar{\boldsymbol{d}}^{\mathbf{u}}_{\boldsymbol{k}} \cong \bar{\boldsymbol{d}}$ in the terminology of Section 3.2. The aggregated state transition matrix $\bar{\boldsymbol{C}}_{\boldsymbol{k}}$ is given by

$$\bar{\boldsymbol{C}}_{\boldsymbol{k}} := \begin{bmatrix} \bar{\boldsymbol{A}}^{\boldsymbol{k}}_{\boldsymbol{N_k}} & \bar{\boldsymbol{A}}^{\boldsymbol{k}}_{\boldsymbol{N_k+1}} \boldsymbol{B}_{\boldsymbol{N_k}} & \bar{\boldsymbol{A}}^{\boldsymbol{k}}_{\boldsymbol{N_k+2}} \boldsymbol{B}_{\boldsymbol{N_k+1}} & \cdots & \bar{\boldsymbol{A}}^{\boldsymbol{k}}_{\boldsymbol{N_{k+1}}} \boldsymbol{B}_{\boldsymbol{N_{k+1}-1}} \end{bmatrix},$$

with $\bar{\boldsymbol{A}}^{\boldsymbol{k}}_{\boldsymbol{j}} := \boldsymbol{A}_{\boldsymbol{N_{k+1}-1}} \boldsymbol{A}_{\boldsymbol{N_{k+1}-2}} \cdots \boldsymbol{A}_{\boldsymbol{j}}$, and we have $\bar{\boldsymbol{E}} := \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$ of consistent dimensionality.

The partially condensed block-structured QP can be set up offline, or, respectively, during the preparation phase. The condensing algorithms can be run in parallel for all $m+1$ aggregated stage problems. The Newton system to be solved in each iteration of the dual Newton strategy then is only of size $m\, n_x$ as opposed to $N\, n_x$ in the uncondensed problem. The bandwidth of the fill-in of the dual Hessian matrix identically remains $4\, n_x - 2$.

Applying partial condensing the the first $0 < N_1 \ll N$ stages in particular opens up the possibility to provide ultra-fast approximate feedback already after the first solution of the aggregated stage QP. The obtained solution $\bar{\boldsymbol{z}}_{\mathbf{0}}(\boldsymbol{\lambda})$ respects the stage and continuity constraints of the first N_1 stages, and can be interpreted as the optimal solution of a linear MPC controller on a shortened horizon of length N_1 with a linear terminal cost $\bar{\boldsymbol{\lambda}}_{\mathbf{1}}{}^{\top} \bar{\boldsymbol{C}}_{\mathbf{0}}\, \bar{\boldsymbol{z}}_{\mathbf{0}}$.

To which extent the solver performance benefits from partial condensing depends of course on the specific problem and on the computational architecture available; the choice of m therefore remains a tuning parameter which gradually shifts between a plain condensing/active-set method and the dual Newton strategy. The price to pay for the reduced size of the dual Newton system are more expensive stage problems due to increased dimensionality and possibly more active-set changes per dual Newton iteration on each stage, and, of course, the additional computational effort for the condensing operations in the preparation phase.

4.7 Real-Time Quadratic Programming

In the following, we present several complementary extensions for the dual Newton strategy that aim to improve its practical performance, particularly in online applications.

4.7.1 A preconditioned dual gradient method

Although no numerical results have been discussed up to this point, it will become clear in Chapter 6 that the dual Newton strategy performs rather well for a variety of problems. A particular noteworthy observation, which has to be anticipated here, is the fact that in most instances the number of major iterations in Algorithm 4.1, i.e., the number of nonsmooth Newton iterations, was significantly lower than the number of stage constraints which are active in the initialization and inactive in the solution or vice versa. This implies that an average dual Newton iteration, just as we hoped for when designing the method, jumps over several regions "towards" the solution, i.e. performs simultaneously *several* favorable active set changes, which make the current set of active stage constraints resemble more the set of active stage constraints in the solution[7]. This is remarkable, since a priori we cannot really expect the curvature information employed for the computation of our step to be accurate or even approximately accurate outside our current region due to the lack of C^2-continuity. In fact it is not hard to construct instances of a piecewise quadratic spline where — in its global phase — a nonsmooth Newton method does not converge any faster than a gradient method, i.e., requires hundreds or thousands of iterations, as opposed to tens, which we typically observe. Our expectation however is, that such worst-case instances normally do no not occur in practice[8].

The central point of the above discussion is to stress that the predictive powers of the Newton step outside the current region are limited. In particular, when the dual Newton strategy is coldstarted, we cannot expect the added or removed

[7]Recall that by Lemma 4.19 the dual Newton strategy is guaranteed to converge in one iteration once the correct set of active stage constraints is identified.

[8]Such motivations are actually quite common in active-set methods in general. The most well-known example arguably is the Simplex algorithm, which is very successfully applied in practice, despite the devastating worst-case complexity demonstrated in the famous paper by Klee and Minty [KM72]. Besides practical observations, we draw motivation from the success of two established classes of methods, active-set methods on the one hand, and dual decomposition gradient methods on the other hand. By Lemma 4.19 we are guaranteed an active set change in each iteration as long as the current region is not degenerate; we furthermore have an ascent in the dual function, so we can hope to perform at least as good as a simple active-set method. If, on the other hand, our current region is degenerate, such that we obtain a perturbed Newton step due to regularization, we can still expect to perform at least as good as a gradient method; since we can warmstart the dual Hessian factorizations, and in the specific case where we do not get a primal active set change in one of the stage constraints even with a full step we do not have to re-factorize at all, the per-iteration cost of the dual Newton strategy is not much more expensive than the iteration cost of a dual decomposition based gradient method. Furthermore, we can, in general, hope for a certain similarity of the Newton steps computed from neighboring regions (at least in the average case), as the Hessians characterizing neighboring regions only differ in a single direction, which get added to or removed from the Hessian due to the active set change.

curvature directions due to the active constraints in the (uninformed) initial guess to be particularly helpful (as long as they do not correspond to explicit or implicit equality constraints, of course). On the contrary, the dual Newton performance may even be impeded, if the region containing the initial guess is degenerate and therefore cannot be left quickly due to regularization perturbing the Newton direction. As an alternative to overcome these drawbacks, we propose to make use of a preconditioned dual gradient method whenever we cannot hope for significant additional information gain from employing regular Newton-type iterations — due to distance[9] to the solution, or due to inexactness of the Newton direction. Here, we of course have to ensure that exact dual Newton steps are tested on a regular basis (particularly in the very first iteration), to make sure we benefit from the guaranteed one-step terminal convergence due to Lemma 4.19. The key ingredient to this heuristic is the preconditioner, which we suggest to choose as the unconstrained[10] dual Hessian

$$\hat{\boldsymbol{\mathcal{M}}} := \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{C}}^{\top}, \tag{4.37}$$

i.e., the Hessian characterizing the region where no stage constraints are active. The resulting scaled gradient step then is computed as

$$\boldsymbol{\Delta\lambda} := \hat{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{i}). \tag{4.38}$$

Note that $\hat{\boldsymbol{\mathcal{M}}}$ inherits strict positive definiteness from $\boldsymbol{\mathcal{H}}$, as $\boldsymbol{\mathcal{C}}$ has full row rank by assumption.

The benefit of choosing the preconditioner $\hat{\boldsymbol{\mathcal{M}}}^{-1}$ over the correct dual Hessian is twofold. First of all, since $\hat{\boldsymbol{\mathcal{M}}}$ is fully determined at the time of the problem setup, a factorization can also be precomputed at this time and kept in the online context. Second, as we will show in the following, no globalization routine is required when using $\boldsymbol{\Delta\lambda}$ from (4.38), since the full step is guaranteed to be the optimal choice in (4.19) and in particular sufficient to ensure global convergence.

To establish our claim, we first show that $\hat{\boldsymbol{\mathcal{M}}}$ exhibits the most curvature among all dual Hessians. We use analogous notation to Lemma 4.21.

Lemma 4.36 *Let $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{n\times n}$ be positive definite, let $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{n\times l}$, where $l \leq n$, have full column rank, and let $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n\times m}$. Then, for $\boldsymbol{\mathcal{M}} := \boldsymbol{\mathcal{C}}\,\boldsymbol{\mathcal{Z}}\,(\boldsymbol{\mathcal{Z}}^{\top}\boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}})^{-1}\,\boldsymbol{\mathcal{Z}}^{\top}\,\boldsymbol{\mathcal{C}}^{\top}$ and $\hat{\boldsymbol{\mathcal{M}}} := \boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{C}}^{\top}$ we have*

$$\hat{\boldsymbol{\mathcal{M}}} \succeq \boldsymbol{\mathcal{M}}.$$

[9] Being far away from the solution can, for example, be assumed when we know from the context that the initial guess is bad or completely uninformed.

[10] W.l.o.g. and for notational convenience only, we assume that all stage equality constraints have been eliminated from the QP formulation. This does not affect the definiteness of $\hat{\boldsymbol{\mathcal{M}}}$ if LICQ holds in the QP solution, cf. Lemma 4.21.

Proof To verify our claim it is sufficient to show that

$$\boldsymbol{\mathcal{H}}^{-1} - \boldsymbol{\mathcal{Z}}\,(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}})^{-1}\,\boldsymbol{\mathcal{Z}}^\top \succeq \mathbf{0}. \tag{4.39}$$

To this end, we regard the auxiliary matrix

$$\boldsymbol{\mathcal{X}} := \begin{bmatrix} \boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}} & \boldsymbol{\mathcal{Z}}^\top \\ \boldsymbol{\mathcal{Z}} & \boldsymbol{\mathcal{H}}^{-1} \end{bmatrix}.$$

The Schur complement of $\boldsymbol{\mathcal{H}}^{-1}$ in $\boldsymbol{\mathcal{X}}$ is given by

$$\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}} - \boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}},$$

which is trivially positive semidefinite. Strict positive definiteness of $\boldsymbol{\mathcal{H}}^{-1}$ implies[11] that also

$$\boldsymbol{\mathcal{X}} \succeq \mathbf{0}$$

holds, which in turn can be equivalently characterized by positive semidefiniteness of

$$\boldsymbol{\mathcal{H}}^{-1} - \boldsymbol{\mathcal{Z}}\,(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}})^{-1}\,\boldsymbol{\mathcal{Z}}^\top,$$

the Schur complement of $\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\,\boldsymbol{\mathcal{Z}}$ in $\boldsymbol{\mathcal{X}}$. This implies (4.39) and therefore concludes our proof. □

Theorem 4.37 *Let* (PQP) *be feasible and let* $\boldsymbol{\lambda^0} \in \mathbb{R}^{Nn_x}$. *Then, the full-step preconditioned dual gradient method given by the update rule*

$$\boldsymbol{\lambda^{i+1}} := \boldsymbol{\lambda^i} + \boldsymbol{\Delta\lambda},$$

where $\boldsymbol{\Delta\lambda} := \hat{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})$, *converges globally to the unique maximum* $\boldsymbol{\lambda^*} \in \mathbb{R}^{Nn_x}$ *with* $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^*})\| = 0$.

Proof Clearly, $\boldsymbol{\Delta\lambda}$ is an ascent direction, since

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \boldsymbol{\Delta\lambda} = \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i})^\top \hat{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) > 0$$

for $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda^i}) \neq \mathbf{0}$. To being able to apply the proof of Theorem 4.20 it only remains to show that the full step is indeed the optimal step size for our specific choice of $\boldsymbol{\Delta\lambda}$, i.e., that the solution to Equation (4.19) fulfills

$$\alpha^* := \underset{0 \leq \alpha \leq 1}{\arg\max}\, f^*(\boldsymbol{\lambda^i} + \alpha \boldsymbol{\Delta\lambda^i}) = 1.$$

To see this, we expand the dual function $f^*(\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda})$ at a point $\boldsymbol{\lambda}$ in search direction $\alpha\boldsymbol{\Delta\lambda}$, exploiting its piecewise quadratic shape. W.l.o.g. we denote

[11] For further reading regarding the Schur complement and its applications we kindly refer to [Zha06, BV04].

the regions traversed by the straight path $(\boldsymbol{\lambda}, \boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda})$ for a fixed $0 < \alpha \leq 1$ ascendingly by $A^{(0)}, A^{(1)}, \ldots, A^{(m)}$ according to their visit order along the path, i.e., $\boldsymbol{\lambda} \in A^{(0)}$ and $\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda} \in A^{(m)}$. We denote the corresponding dual Hessians by $\boldsymbol{\mathcal{M}_0}, \boldsymbol{\mathcal{M}_1}, \ldots, \boldsymbol{\mathcal{M}_m}$. The regions $\{A^{(j)}\}_{j=0,\ldots,m}$ induce a natural partition of

$$\alpha = \sum_{j=0}^{m} \alpha_j,$$

where $\alpha_j \geq 0$ for all $j = 0, \ldots, m$. We further label the seams in search direction by

$$\boldsymbol{\lambda^{(j)}} := \boldsymbol{\lambda} + \sum_{l=0}^{j-1} \alpha_l \boldsymbol{\Delta\lambda}$$

for $j = 1, \ldots, m+1$, and consistently introduce $\boldsymbol{\lambda^{(0)}} = \boldsymbol{\lambda}$.

Since the dual function is a piecewise quadratic concave spline, we have

$$f^*(\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda}) = f^*(\boldsymbol{\lambda}) + \sum_{j=0}^{m} \left(\alpha_j \, \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(j)}}\big)^\top \boldsymbol{\Delta\lambda} - \frac{1}{2} \alpha_j^2 \, \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_j} \, \boldsymbol{\Delta\lambda} \right). \quad (4.40)$$

By C^1-continuity of f^*, we further have

$$\boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(j)}}\big) = \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big) - \sum_{l=0}^{j-1} \alpha_l \, \boldsymbol{\mathcal{M}_l} \, \boldsymbol{\Delta\lambda}. \quad (4.41)$$

Plugging this into (4.40) and regrouping the summands, we obtain

$$\begin{aligned} f^*(\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda}) &= f^*(\boldsymbol{\lambda}) + \sum_{j=0}^{m} \alpha_j \cdot \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)^\top \boldsymbol{\Delta\lambda} \\ &\quad - \sum_{j=0}^{m} \left(\alpha_j \sum_{l=0}^{j-1} \alpha_l \, \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_l} \, \boldsymbol{\Delta\lambda} + \frac{1}{2} \alpha_j^2 \, \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_j} \, \boldsymbol{\Delta\lambda} \right) \\ &= f^*(\boldsymbol{\lambda}) + \sum_{j=0}^{m} \alpha_j \cdot \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)^\top \boldsymbol{\Delta\lambda} \\ &\quad - \sum_{l=0}^{m} \left(\frac{1}{2} \alpha_l^2 + \alpha_l \sum_{j=l+1}^{m} \alpha_j \right) \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_l} \, \boldsymbol{\Delta\lambda}. \end{aligned}$$

W.l.o.g. we can assume that α_m is strictly positive by disregarding degenerate regions $A^{(m)}$ on the basis of C^1-continuity of f^*. Therefore, only α_m is sensitive

to an infinitesimal change in $\bar{\alpha}$, and we have

$$\frac{\partial f^*(\boldsymbol{\lambda} + \bar{\alpha}\boldsymbol{\Delta\lambda})}{\partial \alpha} = \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)^\top \boldsymbol{\Delta\lambda} - \alpha_m \, \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_m} \, \boldsymbol{\Delta\lambda} - \sum_{l=0}^{m-1} \alpha_l \, \boldsymbol{\Delta\lambda}^\top \boldsymbol{\mathcal{M}_l} \, \boldsymbol{\Delta\lambda}.$$

Applying Lemma 4.36, as well as the definitions of α and $\boldsymbol{\Delta\lambda}$, this allows us to conclude that

$$\begin{aligned}\frac{\partial f^*(\boldsymbol{\lambda} + \bar{\alpha}\boldsymbol{\Delta\lambda})}{\partial \alpha} &\geq \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)^\top \boldsymbol{\Delta\lambda} - \sum_{j=0}^{m} \alpha_j \, \boldsymbol{\Delta\lambda}^\top \hat{\boldsymbol{\mathcal{M}}} \, \boldsymbol{\Delta\lambda} \\ &= (1-\alpha) \cdot \underbrace{\boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)^\top \hat{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big)}_{>0} \\ &> 0\end{aligned}$$

holds[12] for all $\alpha < 1$. We can therefore in particular conclude that

$$\arg\max_{\alpha} f^*(\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda}) \geq 1$$

holds for all pairs $(\boldsymbol{\lambda}, \boldsymbol{\Delta\lambda})$, and by monotonicity of $f^*(\boldsymbol{\lambda} + \alpha\boldsymbol{\Delta\lambda})$, which is induced by concavity of f^*, the solution to (4.19) is given by

$$\alpha^* = \arg\max_{0 \leq \alpha \leq 1} f^*(\boldsymbol{\lambda^i} + \alpha\boldsymbol{\Delta\lambda^i}) = 1.$$

The proof of Theorem 4.20 is therefore applicable and we have global convergence of the full-step preconditioned dual gradient method. □

Remark 4.38 Preconditioned dual gradient steps also qualify as an alternative to the regularization approach presented in Section 4.3.4.

We can in particular establish a somewhat analogous result to Theorem 4.31 regarding the behavior of preconditioned gradient steps on infeasible problems.

Theorem 4.39 *If* (PQP) *is infeasible, then exactly one of the two following statements is true:*

1. (QP_k) *is infeasible for at least one* $k \in \mathcal{S}$

[12] Here, we made use of the fact that $\hat{\boldsymbol{\mathcal{M}}} \succ 0$ and that $\boldsymbol{\mathcal{G}}\big(\boldsymbol{\lambda^{(0)}}\big) \neq \mathbf{0}$.

2. *Algorithm 4.1 with the full-step update rule* $\boldsymbol{\lambda}^{\boldsymbol{i+1}} := \boldsymbol{\lambda}^{\boldsymbol{i}} + \boldsymbol{\Delta\lambda}$*, where* $\boldsymbol{\Delta\lambda} := \hat{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$*, converges to an infinite ray.*

Proof We follow the argumentation from the proof of Theorem 4.31. In particular it holds that if Statement 1 does not apply, the dual function f^* exists, can be evaluated, and grows to infinity (cf. Proof 4.31). By the lines of the proof of Theorem 4.37, the iterates $\boldsymbol{\lambda}^{\boldsymbol{i}}$ of Algorithm 4.1 with preconditioned gradient full-steps given by $\boldsymbol{\Delta\lambda} := \hat{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ induce a strictly monotonously inceasing sequence $\{f^*(\boldsymbol{\lambda}^{\boldsymbol{i}})\}_{i\in\mathbb{N}_0}$, since $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})^\top \hat{\boldsymbol{\mathcal{M}}}^{-1}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ does not vanish for non-vanishing gradients $\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ (cf. Lemma 4.30) as $\hat{\boldsymbol{\mathcal{M}}}$ is positive definite.

It remains to show that the iterates $\boldsymbol{\lambda}^{\boldsymbol{i}}$ diverge in a detectable fashion, by converging to an infinite ray. To this end, we regard the piecewise linear gradient expansion (4.41) using the preconditioned gradient full-step definition:

$$\begin{aligned}\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i+1}}) &= \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) - \sum_{j=0}^{m} \alpha_j \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \boldsymbol{\Delta\lambda} \\ &= \left(\mathbf{I} - \sum_{j=0}^{m} \alpha_j \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1}\right) \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\,. \end{aligned} \tag{4.42}$$

Since, by Lemma 4.36 it holds $\hat{\boldsymbol{\mathcal{M}}} \succeq \boldsymbol{\mathcal{M}}_{\boldsymbol{j}}$, we can establish that[13]:

$$\begin{aligned} & \mathbf{I} \succeq \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1} && \forall\ 0 \le j \le m \\ \Leftrightarrow \quad & \sum_{j=0}^{m} \alpha_j \mathbf{I} \succeq \sum_{j=0}^{m} \alpha_j \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1} \\ \Leftrightarrow \quad & \mathbf{0} \preceq \mathbf{I} - \sum_{j=0}^{m} \alpha_j \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1}\,. \end{aligned}$$

Here we used that by definition $\sum_{j=0}^{m} \alpha_j = 1$. Also, $\boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1}$ has all non-negative Eigenvalues for each j, and the gradient norm $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\|$ is therefore monotonously decreasing (but not necessarily strictly monotonously). On the other hand $\|\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}})\|$ is bounded from below by Lemma 4.30. Therefore the dual gradients converge in norm.

By (4.42) and the above characterization of $\mathbf{I} - \sum_{j=0}^{m} \alpha_j \boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1}$, this can only be satisfied if at the same time $\boldsymbol{\mathcal{M}}_{\boldsymbol{j}} \hat{\boldsymbol{\mathcal{M}}}^{-1} \boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}^{\boldsymbol{i}}) \to \mathbf{0}$ for any traversed region

[13] Note that $\alpha_j \ge 0$ holds.

$A^{(j)}$. In particular, by the update law (4.42) the gradients $\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ themselves converge to a fixed point $\mathcal{G}^*$, and consequently also for the dual steps it holds $\boldsymbol{\Delta\lambda}^{\boldsymbol{i}} \to \hat{\mathcal{M}}^{-1}\mathcal{G}^*$.

We now establish that the limit of the dual updates is an infinite ray. We have that asymptotically $\boldsymbol{\Delta\lambda}^{\boldsymbol{i}} = \hat{\mathcal{M}}^{-1}\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ lies in the nullspace of any traversed region (so it is a ray, i.e., a path without curvature in the dual function). Also, $\boldsymbol{\Delta\lambda}^{\boldsymbol{i}}$ is an ascent direction, as stated in the beginning of the proof.

It remains to show that the limit ray of the dual iterates is eventually contained in a single region, i.e., that we do not cycle. Following the path of the dual iterates only two possibilities exist. Either we traverse a region $A^{(j)}$ where the dual updates $\hat{\mathcal{M}}^{-1}\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ are not contained in the nullspace of $\boldsymbol{\mathcal{M}_j}$ anymore. Then, however, $\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i+1}})$ would shrink in norm[14] by (4.42), and as the gradients are convergent in norm we eventually cannot pass through such a region anymore. By the above argumentation we eventually end up on a ray (again), and the only other possibility is that we never leave this ray. Since there are only a finite number of regions, and due to convexity of each region a ray can never return to a region once it left it, we eventually have to remain in an unbounded region A^{inf} that contains the ray, which is therefore an infinite ray. □

Based on Theorem 4.39 we can guarantee that in practice the dual increments $\boldsymbol{\Delta\lambda}^{\boldsymbol{i}}$ will eventually lie in the nullspace of the dual Hessian at the current iterate $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ and point into a direction that does not exhibit any active set change and therefore allow us to algorithmically conclude infeasibility of the problem.

4.7.2 Stage constraint softening

Singularity of the dual Hessian matrix may obstruct fast convergence of the dual Newton method. Regardless of our initial LICQ assumption, this phenomenon occurs whenever the dual guess $\boldsymbol{\lambda}^{\boldsymbol{i}}$ forces a selection of active stage constraints (over several stages) that render the coupling constraints redundant[15], a situation that commonly appears "far away" from the optimal solution in the dual space.

Two possibilities discussed so far to treat singularity of the dual Hessian matrix are regularization (cf. Section 4.3.4) and fallback on the unconstrained dual

[14] $\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i+1}})$ would shrink significantly even, as $\boldsymbol{\mathcal{M}_j}$ can only take a finite number of distinct values and $\hat{\mathcal{M}}^{-1}\mathcal{G}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ is bounded in norm from below for an unbounded dual problem.

[15] Note that such a situation can only occur if state constraints are present. This can easily be verified be observing the special structure of the $\boldsymbol{E_k}$ in the definition of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^{\boldsymbol{i}})$ in Lemma 4.12.

Hessian matrix (cf. Section 4.7.1). A third option, that was originally proposed in [KKGD14] for a related method, is to relax state constraints by introducing ℓ^2-penalties on constraint violations instead. Accordingly, for $k \in \mathcal{S}$ the modified stage problems, which additionally feature slack variables $\boldsymbol{s_k} \in \mathbb{R}^{2n_d}$, read

$$\min_{\boldsymbol{z_k}, \boldsymbol{s_k}} \frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \left(\boldsymbol{g_k}^\top + \begin{bmatrix} \boldsymbol{\lambda_k} \\ \boldsymbol{\lambda_{k+1}} \end{bmatrix}^\top \begin{bmatrix} -\boldsymbol{E_k} \\ \boldsymbol{C_k} \end{bmatrix} \right) \boldsymbol{z_k} + \boldsymbol{\lambda_{k+1}}^\top \boldsymbol{c_k} + \frac{\gamma}{2} \|\boldsymbol{s_k}\|_2^2$$

$$\text{s.t.} \begin{bmatrix} \boldsymbol{D_k} \\ -\boldsymbol{D_k} \end{bmatrix} \boldsymbol{z_k} + \begin{bmatrix} -\overline{\boldsymbol{d}}_{\boldsymbol{k}} \\ \underline{\boldsymbol{d}}_{\boldsymbol{k}} \end{bmatrix} \leq \boldsymbol{s_k}. \tag{QP$_k^\gamma$}$$

Obviously (QP$_k^\gamma$) are a relaxation of (QP$_k$) with identical optimal solutions in the limit for $\gamma \to \infty$. We can define a dual function f^γ by summation of the stage problems, analogously to the definition of (DQP) in Section 4.2.1. It can easily be shown (see, e.g., [KKGD14] for details) that the Hessian of f^γ is given by

$$\boldsymbol{\mathcal{M}}^\gamma(\boldsymbol{\lambda}) = \boldsymbol{\mathcal{C}} \left(\boldsymbol{\mathcal{H}} + \gamma\, \boldsymbol{\mathcal{D}^{\mathcal{A}}}^\top \boldsymbol{\mathcal{D}^{\mathcal{A}}} \right)^{-1} \boldsymbol{\mathcal{C}}^\top,$$

where

$$\boldsymbol{\mathcal{D}^{\mathcal{A}}} := \begin{bmatrix} \boldsymbol{D_0^{\mathcal{A}}} & & & \\ & \boldsymbol{D_1^{\mathcal{A}}} & & \\ & & \ddots & \\ & & & \boldsymbol{D_N^{\mathcal{A}}} \end{bmatrix}$$

is the $(N+1)\, n_z$ by $\sum_{k=0}^{N} n_k^{\text{act}}$ block-diagonal matrix composed of the linear terms of the active stage constraints. Therefore $\boldsymbol{\mathcal{M}}^\gamma(\boldsymbol{\lambda})$ is non-singular for all $\boldsymbol{\lambda} \in \mathbb{R}^{N n_x}$ as $\boldsymbol{\mathcal{C}}$ is of full row rank by assumption. It is furthermore clear from the block-diagonal definition of $\boldsymbol{\mathcal{D}^{\mathcal{A}}}$, that the block-structure of the smoothened dual Hessian $\boldsymbol{\mathcal{M}}^\gamma(\boldsymbol{\lambda})$ is identical to the structure of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$.

Since (QP$_k^\gamma$) are a relaxation of (QP$_k$), we need to ensure that γ grows to infinity over the course of the dual Newton iterations when we apply the constraint relaxation strategy. In practice, a hybrid strategy, using softened stage constraints only initially when the guess of the dual solution is still far off, and eventually switching to exactly solved stage problems, may therefore turn out to be more effective. Alternatively, better-scaled regularization terms could be derived from $\boldsymbol{\mathcal{M}}^\gamma(\boldsymbol{\lambda})$ (in comparison to the default Levenberg-Marquardt-type regularization), carefully minding however the trade-off between computational complexity and quality of the regularization.

4.7.3 Warmstarting for repeated QP solution

A key advantage of the dual Newton strategy in the context of MPC and MHE are its warm-starting capabilities. While interior-point methods typically cannot be warmstarted efficiently, and the active set of a condensed QP can, even in the nominal case, change quite significantly from one sampling time to the next due to shifted state constraints, the optimization variables $\boldsymbol{\lambda}$ can be shifted alongside with the sampling time in Algorithm 4.1. It is notable that Newton's method guarantees a one-step convergence in this context if the shifted $\boldsymbol{\lambda}$-guess is in the correct quadratic region, i.e., if the optimal primal active set is consistent with the shifted one, *even if the QP data changes* (e.g., through re-linearization of the nonlinear problem in the RTI framework).

In detail, we suggest to perform the following simple shift (compare to Section 2.3.2) from the optimal dual vector of the QP at sampling time T_i, $\boldsymbol{\lambda^*}$, to an initial guess $\boldsymbol{\lambda^0}$ for the subsequent QP at sampling time T_{i+1}:

$$\begin{aligned} \boldsymbol{\lambda}_{\boldsymbol{k}}^{\mathbf{0}} &:= \boldsymbol{\lambda}_{\boldsymbol{k+1}}^{*} && \forall\, k = 1, \ldots, N-1 \\ \boldsymbol{\lambda}_{\boldsymbol{N}}^{\mathbf{0}} &:= \boldsymbol{\lambda}_{\boldsymbol{N}}^{*}\,. \end{aligned}$$

In the nominal case, this shift ensures that $\boldsymbol{\lambda^0}$ already lies in the correct quadratic region (and thus one-step convergence if the primal terminal stage variables $\boldsymbol{z_N}$ lie in a stable active set (e.g., given by a steady state).

We should stress here, that the possibility to warm-start the Cholesky-like factorizations from a previous factorization, as detailed in Section 4.3.5, is not only given between subsequent QP iterations, but also between subsequently solved QPs. While for regular SQP and the classical RTI scheme typically no benefits arise due to changing QP matrix data from re-linearization, the benefit may be significant in the case of LTI dynamics, as well as in the context of partial and/or inexact re-linearizations, i.e., when Mixed- or Multi-Level Iteration schemes are applied on the NLP level, cf. Section 2.4. In detail, this means that we are able to keep the complete dual Hessian factorization if only feasibility-improving or optimality-improving data updates were performed on the NLP level. Even if a full re-linearization was performed, e.g., on the first N_{frac} stages of a nonlinear MPC problem, we only need to restore the backwards Cholesky factorization for the first N_{frac} blocks. Particularly if only few or even only a single QP iteration is required due to accuracy of the prediction model, these deeply interconnected warm-starting capabilities may lead to a significant reduction of computation times.

In the spirit of parallel hierarchical QP data updates, it is even conceivable that a factorization of the dual Hessian is computed alongside a full re-linearization by

one (meta-)thread[16], while feedback is still provided using a factorized Hessian based on previous linearization information by another (meta-)thread.

4.7.4 Towards strict real-time guarantees

As the dual Newton strategy is an iterative solution method, the exact number of FLOPs required depends on the problem data. Within each dual Newton iteration however, the number of FLOPs required is known rather well (or can at least be overestimated rather tightly) if the clipping stage QP solver is applied; even if qpOASES is employed as stage QP solver (which itself is an iterative algorithm), its real-time option permits to specify limits on the available computation time.

This time limitation may result in a stage QP being solved inexactly, but the obtained primal-dual solution after the time limit has an interpretation as the optimal solution of a perturbed parametric QP, which lies on the straight homotopy path between the optimal QP of the previous call[17] and the desired QP to be solved in this call, cf. Section 1.5.4 (as well as [FBD08]).

In the context of a dual Newton iteration, where we assume that an optimal initialization of a stage problem $\mathrm{QP}_k(\boldsymbol{\lambda})$ is given, and we aim to solve the parametric stage problem $\mathrm{QP}_k(\boldsymbol{\lambda} + \boldsymbol{\Delta\lambda})$ for the full Newton step $\boldsymbol{\Delta\lambda}$, this perturbed QP has a favorable interpretation. It is simply the QP that corresponds to a curtailed Newton step $\bar{\alpha}\,\boldsymbol{\Delta\lambda}$, i.e., $\mathrm{QP}_k(\boldsymbol{\lambda} + \bar{\alpha}\,\boldsymbol{\Delta\lambda})$, and the obtained stage variables $\boldsymbol{z}^*(\boldsymbol{\lambda} + \bar{\alpha}\,\boldsymbol{\Delta\lambda})$ are the optimal solution of (QP_k) for the dual guess $\boldsymbol{\lambda} + \bar{\alpha}\,\boldsymbol{\Delta\lambda}$. The stepsize $\bar{\alpha}$ is moreover explicitly known from the homotopy method. We might even argue that a Newton step $\boldsymbol{\Delta\lambda}$, which causes so many active set changes in *a single* stage problem that the required computation time exceeds the (sensibly chosen) time limit, is likely to be undesirable anyway (recall that the extrapolation of the dual Hessian information outside the current region is formally merely a heuristic after all) and may therefore require an extensive line search — the cost of which is obviously reduced when limiting the Newton step size to $\bar{\alpha}$. On the other hand, a reasonably chosen time limitation for the stage QP solution will in general not interfere with the good terminal convergence properties of the dual Newton strategy, as the effort in a homotopy method like qpOASES grows with the number of active set changes.

[16] By using the term *meta-thread* we account for to the fact that linearization and QP solution may itself be parallelized on several sub-threads.

[17] Recall that qpOASES modifies the problem on a homotopy path from its initialization to the desired problem to be solved (keeping primal and dual optimality), as opposed to classical active-set methods, which typically only modify the primal and/or dual variables.

The only other iterative routine[18] within the main loop of Algorithm 4.1 is the line search procedure for globalization. Note, however, that in case of an exact line search an upper bound on the number of required iterations is given by the maximum possible number of active-set changes (which is determined by the number of stage constraints, or, more restrictively, by the number of permitted active-set changes, if a limitation is set in qpOASES). In case of a backtracking line-search, numerical reasons demand a minimum stepsize in a practical implementation, which in turn specifies the maximum possible number of line-search iterations.

Having analyzed the effect of limiting the computational costs *per dual Newton major iteration*, a naturally arising interest is to analyze the effect of setting a limit for *the number of iterations*. This aim motivates two important questions to be posed in this context:

- Is there a way to give a good a-priori estimate of, or bound on the number of dual Newton iterations required to solve a band-structured QP based on properties of the problem data?
- What are the implications of terminating suboptimally in the dual sense?

We will not be able to answer these questions in this thesis, but still indicate follow-up research directions.

A-priori estimates of the runtime of iterative optimization algorithms tend to be extremely conservative. The probably most well-known example of this is the extremely large worst-case runtime of the famous Simplex Algorithm for linear programming, which even grows exponentially in the problem size. This observation transfers to active-set methods for convex quadratic programming, cf. [NW06]. Even though runtime guarantees that grow only polynomially in the problem data can be given for some interior-point methods for convex quadratic programming (cf., e.g., [Wri97]), these guarantees often result in bounds which are orders of magnitude worse than the practically observed iteration numbers, and may therefore be too conservative to be used in practice.

More recently, first-order methods with rather tight a-priori bounds on the computational effort have been proposed [Nes04, RMJ11, KF11, BP12, PB14]. In particular the authors of [PB14] were able to present a method for linear model predictive control whose computational costs can be estimated well offline, although this requires the computationally expensive (but offline) solution of a mixed-integer linear programming problem. Starting from the preconditioned gradient steps discussed in Section 4.7.1, which provide a lower bound on the

[18] By iterative, in this context, we mean an algorithm whose costs depend not only on the problem dimensions, but also on the problem data.

progress made by the dual Newton method, as they underestimate the dual function, future research includes trying to transfer some of the guarantees obtained for fast-gradient methods to the dual Newton case.

Concerning suboptimal termination of the dual Newton method, we have a certain analogy to the (unconverged) multiple shooting method, as a dual suboptimal solution (of a dynamic optimization QP) corresponds to a primal solution that respects all stage constraints but features a discontinuous (linearized) state trajectory. Still, since the stage constraints are respected by the obtained primal solution, the computed first control action is admissible (at least for the linearized setting) with respect to the input constraints and mixed control/input constraints.

However, since the state trajectory corresponding to an unconverged dual solution $\boldsymbol{\lambda}$ is discontinuous (and a continuous trajectory means convergence), we cannot really hope to be able to state strong guarantees (if at all) on the obtained primal solution regarding admissibility with respect to state constraints on later stages, or terminal constraints. Still, neglecting state constraints, one open idea for future research is to draw up contractivity arguments similarly to the ones seen for the RTI scheme in Section 2.3.3 (originally presented in [Die02]) to prove an analogy of convergence over moving horizons. It is important to note, however, that the proof concepts from [Die02] are fundamentally different from our setting. There, contraction was proven as a local property, whereas in the context of the dual Newton strategy we are interested in a global contraction property. To show the latter essentially requires to show a similarity between subsequent problems such that the progress (e.g., in terms of the dual function value) made in one problem overcompensates the set-back from the problem modification.

Practical tests on several benchmark problems from linear MPC indeed exhibited the desired behavior. A limitation of the number of dual Newton iterations permitted per MPC problem lead to suboptimal control actions at those sampling times where the dual Newton method was not fully converged yet, but optimality was regained during subsequent sampling times. In particular, the overall number of required dual Newton iterations (summed over all sampling times) was observed to not be significantly larger than for a fully converged algorithm, which indicates that it may be possible in some cases to "postpone" Newton iterations to later sampling times at the price of temporary suboptimality.

4.7.5 Code generation for linear algebra routines

The idea of automatically generating customized source code tailored to a specific problem instance was first reported in the domain of optimization in

[OK02]. Only in recent years, however, the use of such techniques has grown in popularity, to a large part triggered by [MB10], where significant performance increases in the solution of convex optimization problems were reported by means of an auto-generated interior-point method. The idea has been adopted in more specific contexts, such as nonlinear MPC [HFD11b], MHE [FKV+12], linear MPC with quadratic constraints [DZZ+12, KF13] or conic optimization [DCB13, CPDB13]. In some cases, speedups of up to two orders of magnitude have been observed compared to conventional implementations.

The improvement in efficiency achieved through code generation, or code export as it is sometimes synonymously called, typically is caused by several effects:

- Since all problem dimensions are known at the time the customized code is generated, the resulting code can work with statically allocated memory only, which permits higher efficiency in the memory management of the target computer system.
- Problem instance specific sparsity patterns may be detected at the time of the generation of the code and may be exploited, e.g., by avoiding unnecessary or redundant computations.
- Partial loop unrolling may permit certain compilers to vectorize operations for higher CPU and cache efficiency.
- The effect of certain optional components of the optimization code may be tested to a certain extent at the time of the generation and may be fixed in the resulting code, yielding a leaner code with fewer branches, that is potentially better optimizable by state-of-the-art compilers.

As for the dual Newton strategy, we have large amounts of the computational effort of each iteration being caused by linear algebra routines, such as setup and factorization of the dual Hessian matrix[19], which are well suited for customization and auto-generation. Applying code generation to tailor these routines to a specific problem instance is furthermore rather straightforward, as implementations of the required operations are typically rather simple. Still, similarly as in the cases reported above, customized routines may yield a significant practical performance improvement of the overall algorithm.

[19]In benchmark examples the share of the setup and factorization costs in the overall per-iteration costs were often observed to be more than 50%, and even up to 90% in some cases, even in a sequential implementation.

Chapter 5

Generalized Dual Newton Methods

This chapter extends the detailed investigation of dual Newton methods from Chapter 4 in the sense that we aim at generalizing the findings to other use cases than the block-structured quadratic programming problems at the core of this thesis. The goal of these generalizations is to benefit from the parallelizability and the distributed architecture of the dual Newton strategy also in more generic problem classes. Here, we specifically regard generic distributed convex QPs (without any specific assumption on the coupling structure) and partially separable (possibly non-convex) NLPs with a sparse coupling topology.

Acknowledgement The first part of this chapter is largely based on the paper "A Distributed Method for Convex Quadratic Programming Problems Arising in Optimal Control of Distributed Systems" by Attila Kozma, Janick Frasch, and Moritz Diehl [KFD13]. Attila Kozma is the main author of that publication and contributed the modified CG solver, the software implementation and most parts of the writing of the original paper. The analysis of the dual function was performed jointly by Janick Frasch and Attila Kozma. Janick Frasch also contributed the benchmark problem used in that paper (not part of this thesis). Moritz Diehl conceived of the general idea of applying a dual Newton method for distributed quadratic programming.

The second part of this chapter is an adaptation of the paper "An Augmented Lagrangian Based Algorithm for Distributed Non-Convex Optimization" by Boris Houska, Janick Frasch, and Moritz Diehl [HFD14]. Boris Houska is the

main author of that publication and contributed the original idea of using an augmented stage cost function to treat non-convex problems in a dual Newton framework. He also contributed the theoretical analysis to that paper. The details of the method were developed in joint discussions by Janick Frasch and Boris Houska. Janick Frasch also contributed the prototypic implementation of ALADIN and the numerical example. Moritz Diehl contributed several ideas and insights during joint discussions.

5.1 Distributed Quadratic Programming

Let us consider a partially separable QP of the form

$$\min_{\boldsymbol{z}} \sum_{k=0}^{N} \frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{m_k}^\top \boldsymbol{z_k} \tag{5.1a}$$

$$\text{s.t. } \boldsymbol{A_{(i,j)}}\, \boldsymbol{z_i} = \boldsymbol{B_{(i,j)}}\, \boldsymbol{z_j} \qquad \forall\ (i,j) \in \mathcal{E} \tag{5.1b}$$

$$\underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \leq \overline{\boldsymbol{d}}_{\boldsymbol{k}} \qquad \forall\ k \in \mathcal{S} \tag{5.1c}$$

in the following. Here, differing from the rest of this thesis, $\mathcal{S} = \{0, \dots, N\}$ does not denote a set of stages in the sense of a discretized dynamic optimization problem, but instead a set of nodes in a generic (directed, simple[1], finite) coupling topology graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where $\mathcal{E}$ is the edge set, which is of cardinality m. Analogously[2] to Section 4.2, we have $\boldsymbol{z_k} \in \mathbb{R}^{n_{z_k}}$, $\boldsymbol{0} \prec \boldsymbol{H_k} \in \mathbb{R}^{n_{z_k} \times n_{z_k}}$, $\boldsymbol{g_k} \in \mathbb{R}^{n_{z_k}}$, $\boldsymbol{D_k} \in \mathbb{R}^{n_{d_k} \times n_{z_k}}$, and $\underline{\boldsymbol{d}}_{\boldsymbol{k}}, \overline{\boldsymbol{d}}_{\boldsymbol{k}} \in \mathbb{R}^{n_{d_k}}$ for each $k \in \mathcal{S}$. We group the primal variables by $\boldsymbol{z} := (\boldsymbol{z_0}, \dots, \boldsymbol{z_N})$. The node coupling constraints are defined by $\boldsymbol{A_{(i,j)}} \in \mathbb{R}^{n_{(i,j)} \times n_{z_i}}$ and $\boldsymbol{B_{(i,j)}} \in \mathbb{R}^{n_{(i,j)} \times n_{z_j}}$ according to the system topology with $(i,j) \in \mathcal{E}$. We assume full row rank of all $\boldsymbol{A_{(i,j)}}$ and $\boldsymbol{B_{(i,j)}}$, as well as the existence of a (unique) solution that fulfills the LICQ.

Clearly, (5.1) is more generic than (PQP), as the coupling structure of (5.1) is arbitrary. Nevertheless, Algorithm 4.1 (in its generic form) is applicable to (5.1) as well, and we can observe that the convergence analysis from Section 4.4 does not make any specific assumptions on the coupling topology except for it to fulfill the LICQ. Therefore we can in particular conclude finite convergence of a general-purpose version of Algorithm 4.1 in this setting.

[1] We adopt the undirected notion of simpleness here, meaning that if $(i,j) \in \mathcal{E}$ then also $(j,i) \notin \mathcal{E}$ must hold for any two (distinct) $i, j \in \mathcal{S}$.

[2] Slightly more generic, we permit non-identical dimensionality of node variable vectors in this setting.

The generic coupling topology of (5.1) permits a large range of problems, from fully separable to completely connected. Arguably, the most interesting application case for the dual Newton strategy will be topologies that are connected (otherwise an independent solution of the subproblems is possible) but nonetheless exhibit a certain block-sparsity. QPs with such a topology (that is different from the linear topology described in Section 4.2) include subproblems from applying the so-called *Distributed Direct Multiple Shooting* (DDMS) method, cf. [SRKD11], as well as, more generically, subproblems from dynamic optimization with PDE constraints or distributed control, cf. [RM09].

Accordingly, the remainder of this chapter will be about working out the appropriate linear algebra for a structure-exploiting solution of (5.1) with less restrictive assumptions on the coupling topology by the dual Newton algorithm 4.1.

5.1.1 Dual decomposition

The partial Lagrangian function expressing (5.1a) and (5.1b) is given by

$$\begin{aligned}
\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) &:= \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{m_k}^\top \boldsymbol{z_k} \right) \\
&\qquad + \sum_{(i,j)\in\mathcal{E}} \boldsymbol{\lambda_{(i,j)}}^\top \left(\boldsymbol{A_{(i,j)}} \boldsymbol{z_i} - \boldsymbol{B_{(i,j)}} \boldsymbol{z_j} \right) \\
&= \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} \right. \\
&\qquad \left. + \sum_{j\in\mathrm{succ}(k)} \boldsymbol{\lambda_{(k,j)}}\, \boldsymbol{A_{(k,j)}}\, \boldsymbol{z_k} - \sum_{j\in\mathrm{pred}(k)} \boldsymbol{\lambda_{(j,k)}}\, \boldsymbol{B_{(j,k)}}\, \boldsymbol{z_k} \right) \\
&=: \sum_{k=0}^{N} \mathcal{L}_k(\boldsymbol{z_k}, \boldsymbol{\lambda}),
\end{aligned} \tag{5.2}$$

where $\boldsymbol{\lambda_{(i,j)}} \in \mathbb{R}^{n_{(i,j)}}$ for $(i,j) \in \mathcal{E}$ denote the dual variables associated with the coupling constraints (5.1b) and $\boldsymbol{\lambda} \in \mathbb{R}^{\nu}$ is an concatenation of these dual variables in arbitrary but fixed order, where $\nu := \sum_{(i,j)\in\mathcal{E}} n_{(i,j)}$. For future reference we introduce the indexing

$$\boldsymbol{\lambda} = \left(\boldsymbol{\lambda_{(i_1,j_1)}}, \boldsymbol{\lambda_{(i_2,j_2)}}, \ldots, \boldsymbol{\lambda_{(i_m,j_m)}} \right)$$

in accordance with the fixed concatenation order. We further have the predecessor and successor sets of each node defined by

$$\operatorname{pred}(k) := \{\, j \in \mathcal{S} \mid (j,k) \in \mathcal{E} \,\}$$

$$\operatorname{succ}(k) := \{\, j \in \mathcal{S} \mid (k,j) \in \mathcal{E} \,\},$$

as well as the neighborhood of incident edges $\mathcal{N}_k \subseteq \mathcal{E}$ for each node $k \in \mathcal{S}$ given by

$$\mathcal{N}_k := \{\, (k,j) \in \mathcal{E} \mid j \in \operatorname{succ}(k) \,\} \cup \{\, (j,k) \in \mathcal{E} \mid j \in \operatorname{pred}(k) \,\}.$$

Consequently, the dual problem to (5.1) reads

$$\max_{\boldsymbol{\lambda}} f^*(\boldsymbol{\lambda}) := \max_{\boldsymbol{\lambda}} \sum_{k=0}^{N} f_k^*(\boldsymbol{\lambda}) \tag{5.3}$$

with the separable dual function given through

$$f_k^*(\boldsymbol{\lambda}) \;:=\; \min_{\boldsymbol{z_k}} \mathcal{L}_k(\boldsymbol{z_k}, \boldsymbol{\lambda}) \tag{5.4a}$$

$$\text{s.t.} \quad \underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \;\leq\; \overline{\boldsymbol{d}}_{\boldsymbol{k}}. \tag{5.4b}$$

5.1.2 Derivative structure

From the definition of $\mathcal{L}_k(\boldsymbol{z_k}, \boldsymbol{\lambda})$ in (5.2) it is clear that for each $k \in \mathcal{S}$ we have that

$$\frac{\partial \boldsymbol{f_k^*}}{\partial \boldsymbol{\lambda_{(i,j)}}} = \mathbf{0}$$

for all multipliers of non-incident edges $(i,j) \notin \mathcal{N}_k$. The right hand side vector $\boldsymbol{\mathcal{G}(\lambda)} \in \mathbb{R}^{\nu}$ of the Newton system (4.5) for Problem (5.1) is therefore given by

$$\boldsymbol{\mathcal{G}}(\boldsymbol{\lambda}) \;=\; \frac{\partial \boldsymbol{f}^*}{\partial \boldsymbol{\lambda}}(\boldsymbol{\lambda})^{\top} \;=\; \begin{bmatrix} \frac{\partial \boldsymbol{f}_{i_1}^*}{\partial \boldsymbol{\lambda}_{(i_1,j_1)}}^{\top} + \frac{\partial \boldsymbol{f}_{j_1}^*}{\partial \boldsymbol{\lambda}_{(i_1,j_1)}}^{\top} \\ \frac{\partial \boldsymbol{f}_{i_2}^*}{\partial \boldsymbol{\lambda}_{(i_2,j_2)}}^{\top} + \frac{\partial \boldsymbol{f}_{j_2}^*}{\partial \boldsymbol{\lambda}_{(i_2,j_2)}}^{\top} \\ \vdots \\ \frac{\partial \boldsymbol{f}_{i_m}^*}{\partial \boldsymbol{\lambda}_{(i_m,j_m)}}^{\top} + \frac{\partial \boldsymbol{f}_{j_m}^*}{\partial \boldsymbol{\lambda}_{(i_m,j_m)}}^{\top} \end{bmatrix} (\boldsymbol{\lambda}), \tag{5.5}$$

using the enumeration indexing from concatenation.

Analogously to Section 4.3.3, we have

$$\frac{\partial f_k^*}{\partial \lambda_{(k,j)}} = {z_k^*}^\top A_{(k,j)}$$

for all $(k, j) \in \mathcal{N}_k$, and

$$\frac{\partial f_k^*}{\partial \lambda_{(j,k)}} = -{z_k^*}^\top B_{(j,k)}$$

for all $(j, k) \in \mathcal{N}_k$.

The sparsity pattern of the left-hand side Newton matrix $\mathcal{M} : \mathbb{R}^\nu \to \mathbb{R}^{\nu \times \nu}$ is induced by the (undirected) adjacency matrix corresponding to the coupling topology of (5.1). We have

$$\mathcal{M}(\lambda) := -\frac{\partial^2 f^*}{\partial \lambda^2}(\lambda) = \begin{bmatrix} M_{1,1} & \cdots & M_{1,m} \\ \vdots & \ddots & \vdots \\ M_{m,1} & \cdots & M_{m,m} \end{bmatrix} (\lambda), \tag{5.6}$$

where a block $M_{k,l}$ is nonzero only if the k-th and the l-th edge, $(i_k, j_k) \in \mathcal{E}$ and $(i_l, j_l) \in \mathcal{E}$, share at least one node, i.e. $i_k = i_l$, $j_k = i_l$, $i_k = j_l$, or $j_k = j_l$.

The nonzero Hessian blocks are, in analogy to Lemma 4.12, given by

$$M_{k,k} = A_{(i_k,j_k)} P^*_{i_k} {A_{(i_k,j_k)}}^\top + B_{(i_k,j_k)} P^*_{j_k} {B_{(i_k,j_k)}}^\top$$

on the diagonal, as well as by

$$M_{k,l} = A_{(i_k,j_k)} P^*_{i_k} {A_{(i_l,j_l)}}^\top$$

if $i_k = i_l$, by

$$M_{k,l} = -B_{(i_k,j_k)} P^*_{j_k} {A_{(i_l,j_l)}}^\top$$

if $j_k = i_l$, by

$$M_{k,l} = -A_{(i_k,j_k)} P^*_{i_k} {B_{(i_l,j_l)}}^\top$$

if $i_k = j_l$, and by

$$\boldsymbol{M_{k,l}} = \boldsymbol{B_{(i_k,j_k)}}\, \boldsymbol{P^*_{j_k}}\, \boldsymbol{B_{(i_l,j_l)}}^\top$$

if $j_k = j_l$. Here, analogously to Lemma 4.12, $\boldsymbol{P^*_q}$ denotes a projection of $\boldsymbol{H_q}$ onto the free variables for the current selection of active stage constraints in $\boldsymbol{z^*_q}(\boldsymbol{\lambda})$.

5.1.3 Iterative linear algebra

The arbitrary sparsity patterns of the the dual Hessian matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ of QP (5.1) render the solution of the Newton system by direct, factorization-based methods significantly more expensive than the solution of the block-tridiagonal system of (PQP) in Section 4.2, where efficient tailored factorizations could be applied, cf. Sections 4.3.4, 4.3.5, and 4.6.1. Manual variable reorderings in the dual Newton system can help to reduce these costs in certain cases; state-of-the-art solvers for sparse linear systems based on the ideas proposed, for example, in [DR83, Duf04, Duf06] can also be employed to exploit the sparsity of the coupling structure, but the cost per dual Newton iteration will, in general, still be significantly higher than in the block-banded case.

For the particular motivation of (5.1) arising in a distributed control problem, iterative linear algebra methods for the solution of the sparse Newton system offer certain benefits, as they do neither require to ever form (and store) the Hessian matrix or gradient explicitly in a central computing node, nor to compute a factorization or inverse of a matrix in the full size, which may be prohibitive for reasons of computation time, memory storage capacities, or communication limitations in distributed control applications. On this note, we sketch the employment of a tailored conjugate gradient (CG) method in a dual Newton strategy for distributed computing in the following. Additional details can be found in [KFD13]. For notational simplicity, we restate the Newton system (4.5) to be solved in Line 7 of Algorithm 4.1 as

$$\boldsymbol{\mathcal{M}}\, \boldsymbol{\Delta} = \boldsymbol{\mathcal{G}}. \tag{5.7}$$

Algorithm 5.1, which is based on [She94] describes the CG procedure with slight modifications to account for potential singularity of $\boldsymbol{\mathcal{M}}$ and to increase numerical stability.

Algorithm 5.1: Conjugate gradient method [KFD13]

Input : $\boldsymbol{\mathcal{M}}$, $\boldsymbol{\Delta^0}$, $\boldsymbol{\mathcal{G}}$, n_{maxCGIt}, ϵ_1, ϵ_2, ϵ_3

1 $\boldsymbol{r^0} := \boldsymbol{\mathcal{M}}\boldsymbol{\Delta^0} - \boldsymbol{\mathcal{G}}$, $\boldsymbol{p^0} := -\boldsymbol{r^0}$, $\tau^0 := \boldsymbol{r^0}^\top \boldsymbol{r^0}$

2 **for** $i = 0 : (n_{\text{maxCGIt}} - 1)$ **do**

3 $\quad \boldsymbol{s^i} := \boldsymbol{\mathcal{M}}\,\boldsymbol{p^i}$ // local comm

4 $\quad \begin{bmatrix} \delta^i \\ \mu^i \end{bmatrix} := \begin{bmatrix} \boldsymbol{p^i}^\top \boldsymbol{s^i} \\ \boldsymbol{r^i}^\top \boldsymbol{s^i} \end{bmatrix}$ // global sum

5 $\quad$ **if** $\delta^i < \epsilon_1$ **then**

6 $\quad\quad$ **if** $i = 0$ **then**

7 $\quad\quad\quad$ **return** $\boldsymbol{p^0}$

8 $\quad\quad$ **else**

9 $\quad\quad\quad$ **return** $\boldsymbol{\Delta^i}$

10 $\quad \alpha^i := \frac{\tau^i}{\delta^i}$ // local update

11 $\quad \boldsymbol{\Delta^{i+1}} := \boldsymbol{\Delta^i} + \alpha^i\,\boldsymbol{p^i}$ // local update

12 $\quad \boldsymbol{r^{i+1}} := \boldsymbol{r^i} + \alpha^i\,\boldsymbol{s^i}$ // local update

13 $\quad \tau^{i+1} := \boldsymbol{r^{i+1}}^\top \boldsymbol{r^{i+1}}$ // global sum

14 $\quad$ **if** $\tau^{i+1} < (\epsilon_3)^2$ **then**

15 $\quad\quad$ **return** $\boldsymbol{\Delta^{i+1}}$

16 $\quad$ **if** $\left|\frac{\tau^i + \alpha^i \mu^i}{\tau^i}\right| > \epsilon_2$ **then**

17 $\quad\quad \beta^{i+1} := 0$ // local update

18 $\quad$ **else**

19 $\quad\quad \beta^{i+1} := \frac{\tau^{i+1}}{\tau^i}$ // local update

20 $\quad \boldsymbol{p^{i+1}} := -\boldsymbol{r^{i+1}} + \beta^{i+1}\,\boldsymbol{p^i}$ // local update

21 **return** $\boldsymbol{\Delta^{i+1}}$

In detail, we safeguard the CG algorithm against $\boldsymbol{p^i}$ being in the nullspace of $\boldsymbol{\mathcal{M}}$ in Lines 5 through 9; if $\boldsymbol{p^i}$ is too close to a potential singular direction of $\boldsymbol{\mathcal{M}}$, we simply return the best known approximate solution[3] to (5.7).

In exact arithmetic, it is well known that at most ν CG iterations are required to solve the Newton system 5.7 of size ν exactly, assuming that $\boldsymbol{\mathcal{M}} \succ 0$. In floating-point arithmetic, round-off errors may accumulate and eventually lead to non-orthogonal residuals, i.e., $\left|\boldsymbol{r^{i+1}}^\top \boldsymbol{r^i}\right| \gg 0$. This situation is safeguarded in Line 16 by simply resetting the conjugacy on detection of non-orthogonality of the residuals.

[3] If already $\boldsymbol{p^0}$ is a singular direction of $\boldsymbol{\mathcal{M}}$, the steepest descent step $\boldsymbol{p^0}$ is the best known approximation to $\boldsymbol{\Delta}$.

If the norm of the current residual is sufficiently small (Line 14), or if the iteration limit is hit, the current approximate solution is returned.

5.1.4 Discussion

We have seen in the analysis of the dual Newton strategy in Section 4.6 that all steps of Algorithm 4.1 qualify for a distributed, concurrent execution with the exception of the solution of the Newton system and the line search. If we employ Algorithm 5.1 for the solution of (5.7), the vast majority of operations in each CG step are local operations, as indicated by the comments in Algorithm 5.1. In particular, the computation of the residual update s^i in Line 3 requires only local communication of each distributed node with its neighbors according to the coupling topology of the underlying problem. Only the computation of the scalar products in Lines 4 and 13 requires a so-called *global summation* step, cf. [LvdG93], which is however limited to the negotiation and distribution of a single number for each of the three scalar products. If the well-known Armijo line search procedure is employed for globalization, the global communication requirement in this step is also limited to the negotiation and exchange of a single number, the dual function value for the current candidate stepsize, in each line search iteration, cf. [KFD13].

As mentioned before, the main use case of the CG-based dual Newton strategy are problems where a centralized Newton system solution is impossible or undesired. Therefore, naturally the competitors for this variant of the method come from the area of distributed control algorithms, where (dual-decomposition based) first-order methods have proven to be quite successful, cf. [KCD13] and the references therein. It is noteworthy in this context, that the computational cost of each Newton-CG iteration, i.e., each iteration of Algorithm 5.1, is comparable to the per-iteration cost of dual-decomposition-based first-order methods. Based on this assumption, the paper [KFD13] indicated slight superiority of the CG-based dual Newton strategy over well-established dual-decomposition-based first-order methods on the there considered benchmark problem.

5.2 Distributed Nonlinear Programming

Going beyond quadratic programming as discussed in Chapter 4 and Section 5.1, in the following we address the issue of how the conception of the dual Newton strategy can be transferred to general (convex and non-convex) nonlinear programming. This has implications for distributed nonlinear optimization in

general, but qualifies in particular for alternatives to the SQP-based algorithms for the solution of problems in the form of (MS-NLP), which are a very original and essential basis in finite-dimensional dynamic optimization.

Particularly for their warmstarting capabilities in the context of online optimization, we focused on SQP-type methods in this thesis. While the linearization step for problems of the kind (MS-NLP) is straightforward to parallelize, we developed a parallelizable QP solver with rather good warmstarting capabilities in Chapter 4. In the attempt to avoid unnecessary data collection and re-distribution (which may jeopardize the parallel performance of the overall NLP algorithm) the question of how to integrate further the distributed linearization and the distributed stage subproblem solution in the dual Newton strategy, arises naturally. We propose an algorithmic concept in the following which can be seen as a dual Newton strategy with stage NLP subproblems in place of the QP subproblem in the original method from Chapter 4. The vision for the application in dynamic optimization in a parallel fashion is to have each stage assigned to a specific computational node, which maintains all data related to this stage. Nonlinear local problems are solved on each stage's node individually and concurrently for all nodes, while only the negotiation of the dual "prices" corresponding to the coupling constraints requires communication and interaction in form of data exchange between the stages' nodes.

We only sketch the central ideas leading to this extension in the following, and refer to [HFD14] for theoretical background and a rigorous convergence proof.

5.2.1 A separable NLP formulation

Let us consider the partially separable NLP

$$\min_{\boldsymbol{z}} \sum_{k=0}^{N} \ell_k(\boldsymbol{z_k}) \tag{5.8a}$$

$$\text{s.t.} \sum_{k=0}^{N} \boldsymbol{\mathcal{C}_k}\, \boldsymbol{z_k} = \boldsymbol{c} \tag{5.8b}$$

$$\boldsymbol{d_k}(\boldsymbol{z_k}) \leq \boldsymbol{0} \qquad \forall\, k \in \mathcal{S} \tag{5.8c}$$

for $\boldsymbol{z} := (\boldsymbol{z_0}, \ldots, \boldsymbol{z_N}) \in \mathbb{R}^{(N+1)n_z}$ with nonlinear (and possibly non-convex) separable objective functions $\ell_k : \mathbb{R}^{n_z} \to \mathbb{R}$, nonlinear (possibly non-convex) decoupled node (stage) constraint functions $\boldsymbol{d_k} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_d}$ for all $k \in \mathcal{S}$ and a linear coupling structure given by a contribution matrix $\boldsymbol{\mathcal{C}_k} \in \mathbb{R}^{Nn_x \times n_z}$ for

each node (stage) $k \in \mathcal{S}$ to the consensus term $\boldsymbol{c} \in \mathbb{R}^{Nn_x}$. Even though only a linear coupling between the nodes (stages) is permitted here, problems in the form of (MS-NLP) can be implemented by replacing stage nonlinear coupling constraints of the form

$$\boldsymbol{E_{k+1}}\, \boldsymbol{z_{k+1}} = \boldsymbol{q}(\boldsymbol{z_k})$$

by linear ones

$$\boldsymbol{E_{k+1}}\, \boldsymbol{z_{k+1}} = \boldsymbol{s_k},$$

introducing the auxiliary variables $\boldsymbol{s_k} \in \mathbb{R}^{n_x}$, which are specified by separable stages constraints

$$\boldsymbol{s_k} = \boldsymbol{q}(\boldsymbol{z_k})$$

for all $k \in \mathcal{S}_N$.

Analogously to Section 4.2.1, we have the dual problem to (5.8) given by

$$\max_{\boldsymbol{\lambda}} \quad f^*(\boldsymbol{\lambda}) \;=\; -\boldsymbol{\lambda}^\top \boldsymbol{c} + \sum_{k=0}^{N} f_k^*(\boldsymbol{\lambda}),$$

where

$$f_k^*(\boldsymbol{\lambda}) \;:=\; \min_{\boldsymbol{z_k}} \; \ell_k(\boldsymbol{z_k}) + \boldsymbol{\lambda}^\top \boldsymbol{C_k}\, \boldsymbol{z_k} \tag{5.9a}$$

$$\text{s.t.} \;\; \boldsymbol{d_k}(\boldsymbol{z_k}) \leq \boldsymbol{0}\,. \tag{5.9b}$$

The dual function $f^*(\boldsymbol{\lambda})$ is concave and twice continuously differentiable whenever the minimizer $\boldsymbol{z_k^*}(\boldsymbol{\lambda})$ of $f_k^*(\boldsymbol{\lambda})$ is a regular KKT point, and can be evaluated separably. We can build up a local quadratic model of the dual function at a fixed $\boldsymbol{\lambda}$ and minimize it by computing the Newton step increment in the dual variables, $\boldsymbol{\Delta\lambda}$, from

$$(\,\boldsymbol{\mathcal{M}} + \delta\, \mathbf{I}\,)\; \boldsymbol{\Delta\lambda} \;=\; \boldsymbol{\mathcal{G}}, \tag{5.10}$$

for $\delta \geq 0$ (depending on whether we require/desire regularization or not), where

$$\boldsymbol{\mathcal{M}} := \boldsymbol{\mathcal{C}}\, \boldsymbol{\mathcal{Z}^*} \left(\boldsymbol{\mathcal{Z}^*}^\top \boldsymbol{\mathcal{H}^*}\, \boldsymbol{\mathcal{Z}^*} \right)^{-1} \boldsymbol{\mathcal{Z}^*}^\top \boldsymbol{\mathcal{C}}^\top,$$

and

$$\boldsymbol{\mathcal{G}} := \sum_{k=0}^{N} \boldsymbol{C_k}\, \boldsymbol{z_k^*}(\boldsymbol{\lambda}) - \boldsymbol{c},$$

cf. Section 4.3.3. Here,

$$\boldsymbol{\mathcal{H}^*} = \text{block diag}(\boldsymbol{H_0^*}, \ldots, \boldsymbol{H_N^*}) \in \mathbb{R}^{(N+1)n_z \times (N+1)n_z}$$

is a positive definite approximation of the Hessian of the Lagrangian of the dual function, i.e.,

$$\boldsymbol{H_k^*} \approx \boldsymbol{\nabla^2} \left(\boldsymbol{\ell_k} \left(\boldsymbol{z_k^*(\lambda)} \right) + \boldsymbol{\mu_k}^\top \boldsymbol{d_k} \left(\boldsymbol{z_k^*(\lambda)} \right) \right),$$

and

$$\boldsymbol{\mathcal{C}} := \begin{bmatrix} \boldsymbol{\mathcal{C}_0} & \cdots & \boldsymbol{\mathcal{C}_N} \end{bmatrix}.$$

Further,

$$\boldsymbol{\mathcal{Z}^*} = \operatorname{block\,diag}(\boldsymbol{Z_0^*}, \ldots, \boldsymbol{Z_N^*}) \in \mathbb{R}^{(N+1)n_z \times ((N+1)n_z - m)}$$

for $m \leq (N+1)n_d$ is a basis matrix for the nullspace of

$$\boldsymbol{\mathcal{D}^*} = \operatorname{block\,diag}(\boldsymbol{D_0^*}, \ldots, \boldsymbol{D_N^*}) \in \mathbb{R}^{m \times (N+1)n_z},$$

which is the matrix consisting of the gradients of all active stage constraints $\frac{\boldsymbol{\partial}}{\boldsymbol{\partial z}} \boldsymbol{(d_k)_i(z_k^*(\lambda))}$ for all $1 \leq i \leq n_d$ with $(d_k)_i(\boldsymbol{z_k^*(\lambda)}) = 0$.

5.2.2 Stage-problem augmentation

Unfortunately, however, we cannot expect convergence of performing dual Newton iterations in a naïve way whenever (5.8) is not strictly convex or its minimizer is not a regular KKT point, cf. [HFD14]. Instead, the idea of our framework is to handle non-convexities by an augmented-Lagrangian approach that is inspired by the Alternating Direction Method of Multipliers (ADMM)[4]. To this end, we require the step in the primal variables associated with the current dual Newton step $\boldsymbol{\Delta\lambda}$. We make use of a rather well-known technical Lemma (to be found in similar forms, for example, in [NW06]):

Lemma 5.1 *Let $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{n \times n}$ be positive definite, and let $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{m \times n}$ with $m \leq n$ be of full row rank, while $\boldsymbol{g} \in \mathbb{R}^n$. Then the unique solution of*

$$\begin{bmatrix} \boldsymbol{\mathcal{H}} & \boldsymbol{\mathcal{D}}^\top \\ \boldsymbol{\mathcal{D}} & \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta z} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{m} \\ \boldsymbol{0} \end{bmatrix}$$

with $\boldsymbol{\Delta z} \in \mathbb{R}^n$ and $\boldsymbol{\mu} \in \mathbb{R}^m$ is given by

$$\boldsymbol{\Delta z} = -\boldsymbol{\mathcal{Z}} \left(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{Z}} \right)^{-1} \boldsymbol{\mathcal{Z}}^\top \boldsymbol{m},$$

where $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{n \times m}$ is a basis matrix for the nullspace of $\boldsymbol{\mathcal{D}}$.

[4] In addition to the seminal works of [GM75, GM76], we refer to [BPCP11] for an in-depth analysis of ADMM.

Proof Substituting $\boldsymbol{\Delta z} = \boldsymbol{\mathcal{Y}}\, \boldsymbol{p_y} + \boldsymbol{\mathcal{Z}}\, \boldsymbol{p_z}$, where $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n \times (n-m)}$ is arbitrary such that $\begin{bmatrix} \boldsymbol{\mathcal{Y}} & \boldsymbol{\mathcal{Z}} \end{bmatrix}$ is nonsingular, the result can be obtained by a straightforward analytical solution of the linear system exploiting the nullspace basis property of $\boldsymbol{\mathcal{Z}}$. Details can be found, for example, in [NW06]. □

Lemma 5.1 implies that within each region of the dual space, for which $\boldsymbol{\mathcal{D}}$ is fixed, the predicted change in the primal variables corresponding to a dual step $\boldsymbol{\Delta\lambda}$ is given by[5]

$$\boldsymbol{\Delta z} = \boldsymbol{\mathcal{Z}} \left(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{Z}} \right)^{-1} \boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{C}}^\top \boldsymbol{\Delta\lambda}. \tag{5.11}$$

Herein, the prediction assumes the current selection of stage constraints fixed and disregards all inactive stage constraints.

It is important to observe that $\boldsymbol{\mathcal{Z}} \left(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{Z}} \right)^{-1} \boldsymbol{\mathcal{Z}}^\top$ and in particular $\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{Z}}$ are block-diagonal and can therefore be computed (inverted/factorized) concurrently on each stage. The prediction of the primal increment on each stage is then given by

$$\boldsymbol{\Delta z_k} = \boldsymbol{Z_k} \left(\boldsymbol{Z_k}^\top \boldsymbol{H_k} \boldsymbol{Z_k} \right)^{-1} \boldsymbol{Z_k}^\top \boldsymbol{\mathcal{C}_k}^\top \boldsymbol{\Delta\lambda}, \tag{5.12}$$

where the sparsity structure of $\boldsymbol{\mathcal{C}_k}$ depends on the coupling structure of the original problem (e.g., only two blocks of $\boldsymbol{\mathcal{C}_k}$ are nonzero if (5.8) is derived from a dynamic optimization problem).

We claim that $\boldsymbol{\Delta z}$, when computed from (5.11), where $\boldsymbol{\Delta\lambda}$ is the solution to the (unregularized) dual Newton system (5.10), is the step that closes the consensus gap in the current quadratic/equality-constrained model (based on the primal guess $\boldsymbol{z} \in \mathbb{R}^{(N+1)n_z}$) in the sense that it is the solution to

$$\begin{aligned} \min_{\boldsymbol{\Delta z}} \quad & \frac{1}{2} \boldsymbol{\Delta z}^\top \boldsymbol{\mathcal{H}}\, \boldsymbol{\Delta z} + \boldsymbol{g}^\top \boldsymbol{\Delta z} \\ \text{s.t.} \quad & \boldsymbol{\mathcal{C}} \left(\boldsymbol{z} + \boldsymbol{\Delta z} \right) = \boldsymbol{c} \\ & \boldsymbol{\mathcal{D}}\, \boldsymbol{\Delta z} = \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{g} = \left(\frac{\partial}{\partial \boldsymbol{z_0}} \boldsymbol{\ell_0}(\boldsymbol{z_0}), \ldots, \frac{\partial}{\partial \boldsymbol{z_N}} \boldsymbol{\ell_N}(\boldsymbol{z_N}) \right)$. To see this, let us regard the dual reformulation of the above QP, which is

$$\begin{aligned} \max_{\boldsymbol{\lambda}^+} \min_{\boldsymbol{\Delta z}} \quad & \frac{1}{2} \boldsymbol{\Delta z}^\top \boldsymbol{\mathcal{H}}\, \boldsymbol{\Delta z} + \boldsymbol{g}^\top \boldsymbol{\Delta z} + \boldsymbol{\lambda^+}^\top \left(\boldsymbol{\mathcal{C}} \left(\boldsymbol{z} + \boldsymbol{\Delta z} \right) - \boldsymbol{c} \right) \\ \text{s.t.} \quad & \boldsymbol{\mathcal{D}}\, \boldsymbol{\Delta z} = \boldsymbol{0}. \end{aligned}$$

[5] Here and in the following, we omit the implicit dependency of $\boldsymbol{\mathcal{Z}}$ and $\boldsymbol{\mathcal{H}}$ on $\boldsymbol{\lambda}$ for notational convenience, and define $\boldsymbol{\mathcal{Z}} \equiv \boldsymbol{\mathcal{Z}}^*$ and $\boldsymbol{\mathcal{H}} \equiv \boldsymbol{\mathcal{H}}^*$ to this end.

Assuming that $\boldsymbol{z} = (\boldsymbol{z_0}, \ldots, \boldsymbol{z_N})$ is such that each $\boldsymbol{z_k}$ is a KKT point of (5.9) for $k \in \mathcal{S}$, we can conclude from the stationarity requirement of the stage Lagrangians that

$$\boldsymbol{g} + \boldsymbol{\mathcal{C}}^\top \boldsymbol{\lambda} + \boldsymbol{\mathcal{D}}^\top \boldsymbol{\mu} = \boldsymbol{0}.$$

Multiplying this by a $\boldsymbol{\Delta z}^\top$ from the nullspace of $\boldsymbol{\mathcal{D}}$ from the left, we can reformulate the above QP once more to obtain

$$\begin{aligned} \max_{\boldsymbol{\lambda}^+} \min_{\boldsymbol{\Delta z}} \; & \frac{1}{2} \boldsymbol{\Delta z}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\Delta z} + \left(\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}\right)^\top \boldsymbol{\mathcal{C}} \boldsymbol{\Delta z} + {\boldsymbol{\lambda}^+}^\top \left(\boldsymbol{\mathcal{C}} \boldsymbol{z} - \boldsymbol{c}\right) \\ \text{s.t.} \; & \boldsymbol{\mathcal{D}} \boldsymbol{\Delta z} = \boldsymbol{0}, \end{aligned}$$

which, by applying Lemma 5.1 to the inner minimization problem, is solved by $\boldsymbol{\Delta z}^* = -\boldsymbol{\mathcal{Z}} \left(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{Z}}\right)^{-1} \boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{C}}^\top \left(\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}\right)$ to

$$\max_{\boldsymbol{\lambda}^+} \; -\frac{1}{2} \left(\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}\right)^\top \boldsymbol{\mathcal{M}} \left(\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}\right) + {\boldsymbol{\lambda}^+}^\top \left(\boldsymbol{\mathcal{C}} \boldsymbol{z} - \boldsymbol{c}\right),$$

which is merely a reformulation of (5.10) to an optimization problem[6], i.e., we have the equivalence $\left(\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}\right) = \boldsymbol{\Delta \lambda}$ of both solutions and our claim regarding $\boldsymbol{\Delta z}$ holds.

We now make use of the fact that $\boldsymbol{\mathcal{C}} \left(\boldsymbol{z} + \boldsymbol{\Delta z}\right) = \boldsymbol{c}$ to obtain the ADMM-inspired decoupled augmented Lagrangian stage problem formulations. We propose to solve

$$\min_{\boldsymbol{z_k^i}} \; \ell_k(\boldsymbol{z_k^i}) + {\boldsymbol{\lambda^i}}^\top \boldsymbol{\mathcal{C}_k} \boldsymbol{z_k^i} + \frac{\rho}{2} \left\| \boldsymbol{z_k^i} - \boldsymbol{y_k^i} \right\|^2_{\boldsymbol{\Sigma_k}} \tag{5.13a}$$

$$\text{s.t.} \; \boldsymbol{d_k}(\boldsymbol{z_k^i}) \leq \boldsymbol{0} \tag{5.13b}$$

for all $k \in \mathcal{S}$ in each iteration i of a dual Newton procedure, where the anchor $\boldsymbol{y^i} = \left(\boldsymbol{y_0^i}, \ldots, \boldsymbol{y_N^i}\right)$ is the current primal iterate. This way, we obtain an improved primal guess $\boldsymbol{z^i} = (\boldsymbol{z_0^i}, \ldots, \boldsymbol{z_N^i})$ for the setup of the dual Newton

[6] Observe that we can add the constant term $-\boldsymbol{\lambda}^\top \left(\boldsymbol{\mathcal{C}} \boldsymbol{z} - \boldsymbol{c}\right)$ without changing the solution of the maximization problem.

problem. If we assume that $\boldsymbol{y^i} = \boldsymbol{z^{i-1}} + \boldsymbol{\Delta z^{i-1}}$ and $\boldsymbol{\Sigma_k} = \boldsymbol{\mathcal{C}_k}^\top \boldsymbol{\mathcal{C}_k}$ we have that

$$\begin{aligned}\sum_{k=0}^{N} \|\boldsymbol{z_k^i} - \boldsymbol{y_k^i}\|_{\boldsymbol{\Sigma_k}}^2 &= \sum_{k=0}^{N} \|\boldsymbol{\mathcal{C}_k}\,(\boldsymbol{z_k^i} - \boldsymbol{y_k^i})\|_2^2 \\ &\geq \left\| \sum_{k=0}^{N} \boldsymbol{\mathcal{C}_k}\,(\boldsymbol{z_k^i} - \boldsymbol{y_k^i}) \right\|_2^2 \\ &= \|\boldsymbol{\mathcal{C}}\,(\boldsymbol{z^i} - \boldsymbol{y^i})\|_2^2 \\ &= \|\boldsymbol{\mathcal{C}}\,\boldsymbol{z^i} - \boldsymbol{c}\|_2^2\end{aligned}$$

(the classical augmented Lagrangian penalty due to [Hes69, Pow69]), which motivates our choice of the stage problem augmentation term[7]. Note however that in general the ADMM choice $\boldsymbol{\Sigma_k} = \boldsymbol{\mathcal{C}_k}^\top \boldsymbol{\mathcal{C}_k}$ is not sufficient to obtain global convergence of our modified dual Newton scheme, but we require positive definite matrices $\boldsymbol{\Sigma_k}$ and a sufficiently large scaling parameter $\rho > 0$ to tackle nonconvexities.

5.2.3 A nonlinear dual Newton strategy

Summarizing the above observations and propositions, we have the following scheme as given by Algorithm 5.2. A sufficiently large positive definite penalty term to compensate the duality gap is defined for each stage problem first. The precise choice of $\boldsymbol{\Sigma_k^i} \succ \boldsymbol{0}$ does not influence the convergence guarantee, but the ADMM-inspired scaling $\boldsymbol{\Sigma_k^i} \approx \boldsymbol{\mathcal{C}_k}^\top \boldsymbol{\mathcal{C}_k}$ (possibly using regularization) seems reasonable. Based on the (local) stage problem solutions, a dual Newton problem is set up, analogously to Algorithm 4.1. Again, the exact choice of $\boldsymbol{H_k} \succ \boldsymbol{0}$ has no impact on the convergence guarantees of the algorithm, but may very well determine the speed of convergence, just as in an SQP method, cf. [HFD14]. After the dual consensus step $\boldsymbol{\Delta\lambda^i}$ as well as the associated step in the primal variables $\boldsymbol{\Delta z^i}$ is computed — exploiting the sparsity patterns defined by the coupling structure of (5.8) — a sensible step size selection is required for the primal and dual updates to ensure global convergence.

[7] Observe in particular that the minima of the proposed, separable penalty terms and the classical augmented Lagrangian penalty coincide and are attained in any primal solution that satisfies (5.8b).

Algorithm 5.2: Augmented Lagrangian based Alternating Direction Inexact Newton Method (ALADIN) [HFD14]

Input: Initial guess $(\boldsymbol{y^0}, \boldsymbol{\lambda^0})$, termination criteria n_{maxIt}, ϵ

Output: Approximate optimal solution $(\boldsymbol{y^*}, \boldsymbol{\lambda^*}, \boldsymbol{\mu^*})$

for $i = 0 : (n_{\text{maxIt}} - 1)$ **do**
- Choose $\mathbf{0} \prec \boldsymbol{\Sigma_k^i} \approx \boldsymbol{\mathcal{C}_k}^\top \boldsymbol{\mathcal{C}_k}$, $k \in \mathcal{S}$ and $\rho > 0$ sufficiently large
- Obtain $(\boldsymbol{z^i}, \boldsymbol{\mu^i})$ by solving (5.13) for all $k \in \mathcal{S}$
- Set up dual gradient $\boldsymbol{\mathcal{G}^i} := \boldsymbol{\mathcal{C}}\, \boldsymbol{z^i} - \boldsymbol{c}$
- **if** $\left\|\boldsymbol{\mathcal{G}^i}\right\| \leq \epsilon$ and $\rho \left\|\boldsymbol{\Sigma_k}(\boldsymbol{z_k} - \boldsymbol{y_k})\right\| \leq \epsilon$ **then**
  - **return** $(\boldsymbol{y^i}, \boldsymbol{\lambda^i}, \boldsymbol{\mu^i})$
- Choose $\mathbf{0} \prec \boldsymbol{H_k} \approx \boldsymbol{\nabla^2}(\boldsymbol{\ell_k}(\boldsymbol{z_k}) + \boldsymbol{\mu_k}^\top \boldsymbol{d_k}(\boldsymbol{z_k}))$, $k \in \mathcal{S}$
- Set up Newton matrix $\boldsymbol{\mathcal{M}^i}$
- Compute $\boldsymbol{\Delta\lambda^i}$ from (5.10) for $\gamma \geq 0$
- Compute $\boldsymbol{\Delta z^i}$ from (5.12)
- Compute suitable step sizes $\alpha_1, \alpha_2, \alpha_3 \in [0, 1]$
- Update primal iterate by $\boldsymbol{y^{i+1}} := \boldsymbol{y^i} + \alpha_1 (\boldsymbol{z^i} - \boldsymbol{y^i}) + \alpha_2 \boldsymbol{\Delta z^i}$
- Update dual iterate by $\boldsymbol{\lambda^{i+1}} := \boldsymbol{\lambda^i} + \alpha_3 \boldsymbol{\Delta\lambda^i}$

To this end we require both a primal and a dual merit function. We suggest to first employ the ℓ^1-penalty function

$$\Phi(\boldsymbol{z}) := \sum_{k=0}^{N} \ell_k(\boldsymbol{z_k}) + \bar{\boldsymbol{\lambda}} \left\|\boldsymbol{\mathcal{C}z} - \boldsymbol{c}\right\|_1 + \bar{\kappa} \sum_{k,i} \max\{\, 0,\, (d_k)_i(\boldsymbol{z_k}) \,\}$$

for this purpose, which is rather well-known from SQP methods. Here, $0 < \bar{\lambda} < \infty$ and $0 < \bar{\kappa} < \infty$ are assumed to be sufficiently large constants such that Φ is an exact penalty function for (5.8). If the combined stage NLP and dual Newton full step decreases $\Phi(\cdot)$ sufficiently, we choose $\alpha_1 = \alpha_2 = \alpha_3 = 1$; if not, we check whether the pure NLP step given by $\alpha_1 = 1$ and $\alpha_2 = \alpha_3 = 0$ guarantees a sufficient progress in the primal merit function $\Phi(\cdot)$[8].

Unfortunately, situations may occur where neither $\alpha_1 = \alpha_2 = \alpha_3 = 1$, nor $\alpha_1 = 1$ and $\alpha_2 = \alpha_3 = 0$ lead to sufficient progress in the primal merit function. This may, for example, be the case if the current selection of active stage constraints is too restrictive (e.g., due to a bad initial guess of $\boldsymbol{\lambda}$ that is far

[8]The outlined procedure is only one possible option. For example, we could also employ a line search strategy in α_1, or a field search in both primal step directions to increase the chances of finding a suitable step candidate.

off the optimal dual vector) and $\boldsymbol{\mathcal{C}}^\top \boldsymbol{\Delta\lambda}$ lies in the nullspace of the reduced dual Hessian $\boldsymbol{\mathcal{Z}} \left(\boldsymbol{\mathcal{Z}}^\top \boldsymbol{\mathcal{H}}\, \boldsymbol{\mathcal{Z}} \right)^{-1} \boldsymbol{\mathcal{Z}}^\top$, giving us $\boldsymbol{\Delta z} = \mathbf{0}$. Such a situation may, in principle, occur whenever $\boldsymbol{\mathcal{M}}$ is singular. If, additionally, the computed augmented NLP step $(\boldsymbol{z} - \boldsymbol{y})$ is too perturbed (e.g., again due to a bad initial guess of $\boldsymbol{\lambda}$) such that it is not a strict descent direction of $\Phi(\cdot)$, no progress is possible in the primal variables and we set $\alpha_1 = \alpha_2 = 0$. Instead, we regard the augmented dual function

$$V_\rho(\bar{\boldsymbol{y}}, \boldsymbol{\lambda}) := -\boldsymbol{\lambda}^\top \boldsymbol{c} + \sum_{k=0}^{N} \left\{ \min_{\boldsymbol{z_k}} \ell_k(\boldsymbol{z_k}) + \boldsymbol{\lambda}^\top \boldsymbol{\mathcal{C}_k}\, \boldsymbol{z_k} + \frac{\rho}{2} \left\| \boldsymbol{z_k} - \bar{\boldsymbol{y}}_{\boldsymbol{k}} \right\| \right\}$$

$$\text{s.t. } \boldsymbol{d_k}(\boldsymbol{z_k}) \leq \mathbf{0},$$

which we aim to maximize over all $\boldsymbol{\lambda} \in \left[\boldsymbol{\lambda^i}, \boldsymbol{\lambda^i} + \alpha_3\, \boldsymbol{\Delta\lambda^i} \right]$ via a line search, analogously to Algorithm 4.1.

We do not include a formal convergence proof here, but refer the reader to [HFD14], where convergence of Algorithm 5.2 is established even in a slightly more generic formulation. The proof is inspired by the convergence proofs of SQP methods; in particular, local superlinear and even quadratic convergence can be shown when sufficiently good Hessian approximations or exact Hessian matrices are employed.

5.2.4 Numerical case study

We indicate the potential of the proposed algorithmic concept by a brief case study. To this end, a prototypic Matlab implementation of ALADIN was realized. We consider a discretized variant of the two state, one input nonconvex optimal control problem from [CA98] given by

$$\min_{\boldsymbol{x},\boldsymbol{u}} \frac{1}{2} \sum_{k=0}^{N-1} (x_{k,0}^2 + x_{k,1}^2 + u_k^2) + \frac{1}{2} \begin{bmatrix} x_{N,0} \\ x_{N,1} \end{bmatrix}^\top \boldsymbol{P} \begin{bmatrix} x_{N,0} \\ x_{N,1} \end{bmatrix} \tag{5.14a}$$

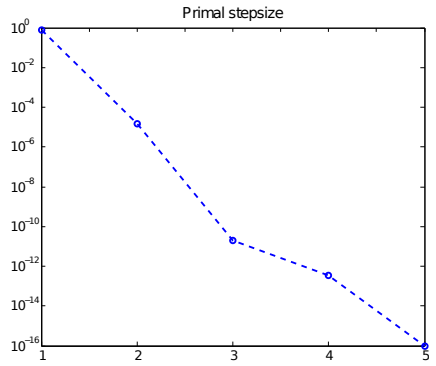
$$\text{s.t. } x_{k+1,0} = x_{k,0} + x_{k,1} + 0.2\, u_k\, (4 + x_{k,0}) \qquad \forall\, k \in \mathcal{S}_N \tag{5.14b}$$

$$x_{k+1,1} = x_{k,1} + x_{k,0} + 0.8\, u_k\, (1 - x_{k,1}) \qquad \forall\, k \in \mathcal{S}_N \tag{5.14c}$$

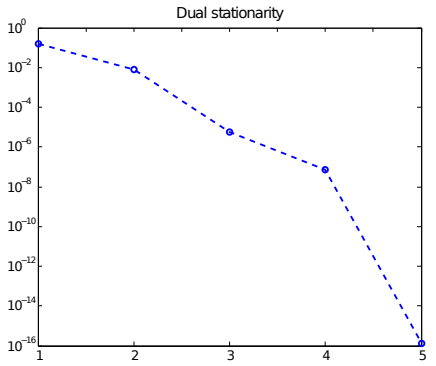
$$-2 \leq u_k \leq 2, \qquad \forall\, k \in \mathcal{S}_N \tag{5.14d}$$

$$x_{0,0} = 0.2 \tag{5.14e}$$

$$x_{0,1} = 0.1 \tag{5.14f}$$



(a) Pimal NLP stepsize.



(b) Dual stationarity.

Figure 5.1: Convergence of ALADIN on the Chen-Allgöwer example.

where the terminal cost is defined by $\boldsymbol{P} := \begin{bmatrix} 16.5926 & 11.5926 \\ 11.5926 & 16.5926 \end{bmatrix}$.

Using the reformulation detailed at the beginning of Section 5.2.1, we can cast (5.14) into the form of (5.8). The resulting NLP has 5 variables per stage, and we use a horizon length of $N = 20$.

For this demonstration, we computed a regularized dual Newton step in each iteration, using $\gamma = 10^{-6}$, and chose $\rho = 10$. We use exact Hessian matrices in the dual problem, but regularize, if necessary, to ensure positive definiteness. The scaling matrices $\Sigma_k,\ k \in \mathcal{S}$ are set to the Hessians of the (non-augmented) stage NLP Lagrangians at the previous NLP solution.

Figure 5.1 shows the convergence behavior of Algorithm 5.2 in terms of the two termination criteria specified in Algorithm 5.2 on the just detailed problem. We can, in particular, see the local superlinear (quadratic) convergence behavior in the dual stationarity, where the number of correct digits doubles from iteration 4 to 5.

Part III

Software and Applications

Chapter 6

The Open-Source Solver qpDUNES

In this chapter, we present a software implementation of the dual Newton strategy from Section 4 that was developed as part of this thesis and is freely and openly available for download at [QPD14] for unrestricted use. Numerical case studies in comparison with state-of-the-art structure-exploiting convex QP solution approaches are carried out on several benchmark problems from linear and nonlinear MPC (the latter one in combination with the also open-source ACADO Code Generation Tool [ACA13, HFD11b]) to demonstrate the effectiveness as well as the efficiency of the dual Newton strategy.

Acknowledgement The software package qpDUNES has been designed and implemented by Janick Frasch, and the open-source project is currently maintained by Janick Frasch. Joachim Ferreau advised with the initial software design. Joachim Ferreau and Milan Vukov contributed small code pieces, mainly in the area of datatypes and interfaces. qpDUNES is essentially self-sufficient, but employs a copy of the open-source software package qpOASES [QPO14] (developed mainly by Joachim Ferreau, Andreas Potschka, and Christian Kirches) internally, which is up to small modifications identical with version 3.0 beta of this software. The numerical results presented in this chapter have originally been published in the papers "A Parallel Quadratic Programming Method for Dynamic Optimization Problems" by Janick Frasch, Sebastian Sager, and Moritz Diehl [FSD13], and "A New Quadratic Programming Strategy for Efficient Sparsity Exploitation in SQP-based Nonlinear MPC and MHE" by Janick Frasch, Milan Vukov, Hans Joachim Ferreau, and Moritz Diehl [FVFD13].

Janick Frasch is the main author of both publications, and contributed the open-source software design and implementation, the numerical benchmarking, as well as most parts of the writing. Milan Vukov and Janick Frasch jointly developed the ACADO/qpDUNES interface. Milan Vukov also contributed the ACADO implementation of the nonlinear hanging chain example. Joachim Ferreau served for fruitful discussions regarding the initial software design of qpDUNES, and contributed the ABB compressor benchmark example.

6.1 Open-source Software Implementation

The core aspects of the dual Newton strategy, as introduced in Chapter 4, have been implemented in a software package that goes under the name qpDUNES. A version of the code is available for download at [QPD14], and can be used freely under the statutes of the *GNU Lesser General Public License (LGPL)* [GNU11], version 3. This essentially means that the packages qpDUNES can be used, modified, adapted and redistributed freely as desired so by the user, without affecting the license under which other user software, with which it might possibly be combined, is used or distributed.

6.1.1 Design decisions

The software qpDUNES is written in the C programming language according to the C90 standard to enlarge compatibility with both standard desktop computers under various operation systems as well as various embedded hardware platforms. The code is entirely self-contained with the exception of minor functionality being drawn from the `cmath` library, which, however, is typically considered to be very standard functionality. Certain optional and not performance-critical functionality, such as execution time measurements for example, requires linking against additional libraries which may not be available on all operating systems.

The variant of qpDUNES that is designed to solve QP problems with general affine stage constraints and not necessarily diagonal Hessian matrices (i.e., problems where the clipping operator (4.6) cannot be employed for the stage subproblem solution) underlyingly makes use of the dense parametric active-set QP solver qpOASES [QPO14], version 3.0 beta, which is identically released under the LGPL license and implemented in C++.

Memory allocation is performed in qpDUNES only on a global scale at the time of the problem setup. No additional memory is being used at runtime, which permits predictability of the memory consumption as well as reusing of memory

blocks for increased memory efficiency. This design decision further facilitates the implementation of a static memory variant of qpDUNES drastically, as only the (in the current, generic implementation adaptive) allocation and deallocation routines have to be replaced by their static counterparts, which match the dimensions of the desired problem to be solved.

The QP problem data, as well as the auxiliary data workspace, are stored in separate memory blocks for each stage subproblem. On a global scale, only data which is clearly centralized, such as the dual Newton system and the current dual $\boldsymbol{\lambda}$ iterate, is stored. This is done to improve parallel efficiency on computing architectures with a physically separated memory layout. Additionally, this way shifting of the problem data for warmstarting is possible without the need for any significant data transfer operations. Furthermore, different subproblem solvers for different stage problems can easily be employed (e.g., for problems with diagonal Hessians and box constraints on all stages except for the last, which may feature a dense Hessian and general affine constraints due to the problem featuring an LQR terminal cost or a complex terminal set constraint).

In its generic implementation, qpDUNES makes use of a dedicated linear algebra module that handles all linear operations in a symbolic way (i.e., when, for example, a multiplication of the state transition matrix $\boldsymbol{A}$ with a vector $\boldsymbol{x}$ is required, a specific routine name `multiplyAx` from the linear algebra module is called) and resolves them to generic informationless linear algebra operations. This way, the linear algebra module is easily replaceable by a code-generated one that exploits the specific structures of each symbolic operation based on problem-specific knowledge of (for example) the sparsity patterns of the corresponding operation. Application of such code generation techniques has lead to significant performance increases in related areas like interior-point solvers [MB09, DZZ+12] and nonlinear MPC [HFD11b].

6.1.2 Structure exploitation

The generic linear algebra implementation of qpDUNES distinguishes three classes of Hessian matrix sparsity patterns: identity Hessians, diagonal Hessians, and generic dense Hessians. Only dense Hessians require $n_z^2 + 1$ memory units[1] to store an $n_z \times n_z$ matrix, while diagonal Hessians only store the diagonal entries in a consecutive order, requiring only $n_z + 1$ memory units; identity Hessians only require an indicator flag intentifying them in their type. All internal symbolic linear algebra routines exploit these specific structures where possible in their implementation.

[1] The additional memory unit is required for the Hessian type indicator flag.

The fact that the dual Hessian matrix is symmetric and in particular block tri-diagonal is also exploited in the memory layout. Instead of using a generic $Nn_x \times Nn_x$ matrix representation, we store the dual Hessian matrix

$$\begin{bmatrix} \boldsymbol{W_1} & \boldsymbol{U_1} & & \\ \boldsymbol{U_1}^\top & \boldsymbol{W_2} & \ddots & \\ & \ddots & \ddots & \boldsymbol{U_{N-1}} \\ & & \boldsymbol{U_{N-1}}^\top & \boldsymbol{W_N} \end{bmatrix}$$

by

$$\begin{bmatrix} & \boldsymbol{W_1} \\ \boldsymbol{U_1}^\top & \boldsymbol{W_2} \\ \vdots & \vdots \\ \boldsymbol{U_{N-1}}^\top & \boldsymbol{W_N} \end{bmatrix},$$

using a row-major format, thus only requiring $(2N-1)\,n_x$ memory units. Analogously, the symmetric factor of the dual Hessian is saved in a more compact representation. As pointed out in Sections 4.3.4 and 4.3.5, the employed factorization routines make use of this explicitly known block-sparsity pattern.

In addition, we distinguish between general affine constraints and box constraints in qpDUNES in order to benefit from the simplified structures of the latter.

6.1.3 Interfacing and problem setup

The QP solver qpDUNES can be employed in several ways in the solution of dynamic optimization problems. Both, a C/C++ and a Matlab interface exist for setting up and solving QP problems in the form of

$$\begin{aligned} \min_{\boldsymbol{z}} \; & \sum_{k=0}^{N} \left(\frac{1}{2} \boldsymbol{z_k}^\top \boldsymbol{H_k}\, \boldsymbol{z_k} + \boldsymbol{g_k}^\top \boldsymbol{z_k} \right) \\ \text{s.t. } & \boldsymbol{E_{k+1}}\, \boldsymbol{z_{k+1}} = \boldsymbol{C_k}\, \boldsymbol{z_k} + \boldsymbol{c_k} && \forall\, k \in \mathcal{S}_N \\ & \underline{\boldsymbol{d}}_{\boldsymbol{k}} \leq \boldsymbol{D_k}\, \boldsymbol{z_k} \leq \overline{\boldsymbol{d}}_{\boldsymbol{k}} && \forall\, k \in \mathcal{S} \\ & \underline{\boldsymbol{z}}_{\boldsymbol{k}} \leq \boldsymbol{z_k} \leq \overline{\boldsymbol{z}}_{\boldsymbol{k}} && \forall\, k \in \mathcal{S}. \end{aligned}$$

Here, we require the distinction of box constraints and affine constraints in order to be able to benefit from the simpler structures of box constraints.

In addition to this, linear MPC problems can be set up more conveniently in the common MPC syntax by providing transition matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ alongside state

and control references $\boldsymbol{x}_{\mathbf{ref}}$ and $\boldsymbol{u}_{\mathbf{ref}}$ with the corresponding tracking weights $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{S}$, and $\boldsymbol{P}$, and bounds (cf. Section 2.1.2); corresponding variants for LTV problems are also available, both from C/C++ and Matlab.

In both, the QP and the linear MPC interface, an elementary sparsity detection is performed at the time of the problem setup to be able to treat identity, diagonal, and dense primal quadratic cost terms individually. Depending on whether additionally affine constraints are present or not, the appropriate stage QP solver (the clipping operation (4.6) or qpOASES) is selected automatically. Both, cold- and warmstarted solution of sequences of QPs is possible from C/C++ and Matlab.

For nonlinear MPC and MHE, qpDUNES has been coupled to the ACADO Code Generation Tool [HFD11b]. The ACADO Code Generation Tool is part of the ACADO Toolkit (available at [ACA13]), which is an open source software for the modeling, simulation and control of nonlinear dynamic processes. It allows to export a lean RTI scheme based on a Gauss-Newton Hessian approximation, which is tailored to a specific problem's requirements in the spirit of code generation. For the evaluation of the nonlinear dynamic system, both constant stepsize explicit and implicit Runge-Kutta ingrators are available, cf. [QVD12]. In earlier versions of the ACADO Code Generation Tool the solution of the structured quadratic subproblems was only possible by a condensing approach (in the form of [AFVD13]) followed by a solution of the dense QP using qpOASES [FBD08]. In recent versions [FVFD13], the interface to qpDUNES permits to bypass the condensing step and to solve structured QPs directly in a sparsity-exploiting fashion. The ACADO Code Generation tool has already been successfully applied in a variety of applications, e.g., [HFD11b, GZD12, GAGD12, VLH+12, FKV+12, FKSD12, FGZ+13, ZFD13, ZGD13, GVD13, GQD12].

6.2 Numerical Performance in Linear MPC

In the following, we analyze the numerical performance of qpDUNES in comparison with state-of-the-art solvers for convex quadratic programs. The focus here is on methods that exploit the structures prevalent in the uncondensed QPs arising from linear MPC, where (for high-accuracy solution) interior-point methods can be seen as the incumbent class of methods.

6.2.1 Double integrator

The first benchmark example is motivated by the energy optimal control of a cart on a rail in the context of a Badminton robot [WSPS12]. The dynamics of the system boil down to a simple double integrator with two states, position and velocity (in this order), and acceleration as control input. The optimization problem obtained after discretization is a convex QP in the form of Problem (PQP) with

$$\boldsymbol{H_k} = \begin{bmatrix} \sigma & & \\ & \sigma & \\ & & 1 \end{bmatrix}, \qquad \boldsymbol{g_k} = \mathbf{0}\ ,$$

$$\boldsymbol{C_k} = \begin{bmatrix} 1 & 0.01 & 0 \\ 0 & 1 & 0.01 \end{bmatrix}, \qquad \boldsymbol{c_k} = \mathbf{0}\ ,$$

$$\boldsymbol{D_k} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, \quad \underline{\boldsymbol{d}}_{\boldsymbol{k}} = (-1.9, -3, -30)\ , \quad \text{and} \quad \underline{\boldsymbol{d}}_{\boldsymbol{k}} = (1.9, 3, 30)$$

for all stages $k \in \mathcal{S}$, and $k \in \mathcal{S}_0$, respectively. Additionally we have the initial value constraint fixing

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{z_0} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

and two arrival constraints demanding the cart to arrive at position 0 at a certain index $\bar{k}$ and staying there for at least 10 ms (one time discretization step), giving the robot arm time to hit the shuttlecock. A small regularization term $\sigma > 0$ in this formulation ensures positive definiteness of the $\boldsymbol{Q_k}$.

This benchmark problem is particularly interesting, as it directly shows the limitations of the dual Newton strategy. Purely energy-minimal operation of the badminton robot would correspond to $\sigma = 0$, resulting in non-strictly convex QPs. To be able to treat this problem with the original dual Newton strategy, regularization is always required. Despite this factual drawback, we still believe that the dual Newton strategy is well suited, since in almost all practical applications regularization is beneficial and often even leads to more desirable properties of the obtained solutions.

Additionally to the small regularization parameter (leading to a rather badly conditioned dual function), we choose arrival times close to infeasibility, ensuring that many state constraints become active in the solution. As discussed in section 4.3.4, scenarios with many active state constraints tentatively are particularly challenging for the dual Newton strategy. Also note in this context that the unconstrained optimum (favoring no action) lies far outside the feasible region.

We compare the average computation time over the first 20 MPC iterations for horizon lengths of $N = 50, 100, 150, 200$, where the cart is forced to arrive at the desired position at $\bar{k} = 50$. We add random noise to the simulated system dynamics, and choose $\sigma = 10^{-4}$. If the MPC problem was rendered infeasible by the (precomputed) noise vector, we discard this instance and generate a new noise vector.

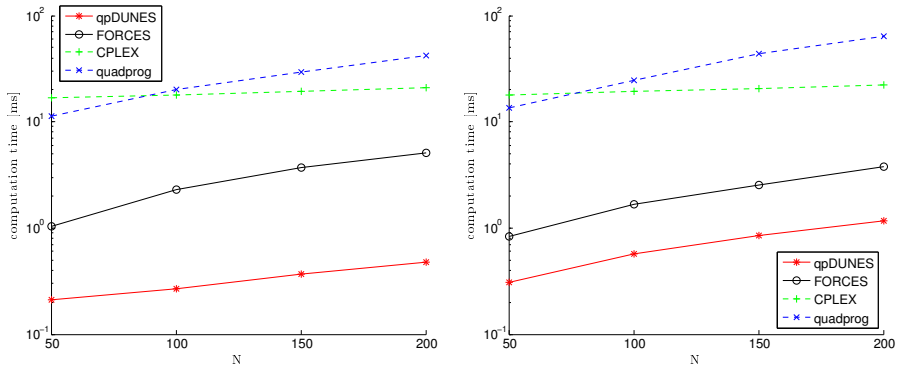
We report computation times on a standard desktop PC featuring a 3.4 GHz Intel i7 CPU under a Ubuntu 13.04 Linux for our method, qpDUNES, in comparison against FORCES [DZZ+12], a very recent and highly efficient structure exploiting interior-point method that uses automatic code generation to create a custom solver tailored to the dynamics of a MPC problem. We run FORCES with default settings, as we did not observe any significant performance improvement in other configurations. It was indicated in [VDF+13] that FORCES will outperform even very efficient active-set methods on prediction horizons of the considered length. It was indicated in [DZZ+12] that FORCES also outperforms other tailored interior-point methods rigorously.

For completeness we include a comparison against the popular solvers CPLEX 12.5.1 [IBM09], and Matlab's quadprog, see[Mat06]. CPLEX implements efficient general purpose QP solvers. Both, an active-set method and an interior-point method is available. We chose to use CPLEX in automatic mode (with parallelization switched off), as fixing CPLEX to either method lead to a performance decrease rather than to an increase. Matlab's quadprog was also run in default configuration, using its interior-point solver. We note that the active-set method was observed to perform orders of magnitude worse than the interior-point method and is therefore not included in the comparison.

All benchmarks were run from Matlab R2013a, as FORCES is currently only provided via a mex-interface. CPLEX and quadprog were called as precompiled libraries for Linux through their default Matlab interface. FORCES was downloaded through its Matlab interface as a custom solver tailored to the model dynamics. Both FORCES and qpDUNES were compiled to a `.mex` file using gcc 4.7.3 with standard code optimization options.

We perform the MPC simulation 1000 times, with different random noise vectors, and report averaged computation times in milliseconds in Figure 6.1(a). We observe that both customized solvers, qpDUNES and FORCES, perform observably well. Still, qpDUNES performs yet a factor of $4.9 - 10.6$ better on this benchmark problem than the runner-up FORCES.

To get a glimpse on worst-case computation times (comparisons that tentatively favor interior-point methods), we additionally report solution times of a single QP without any prior knowledge about the solution (i.e., cold started). We



(a) MPC benchmark.

(b) Cold started benchmark.

Figure 6.1: Computation times of the double integrator benchmark.

chose horizon lengths of $N = 50, 100, 150, 200$ and forced the cart to arrive at the desired position at $\bar{k} = 45$ (almost the minimum time possible).

Even though qpDUNES is not tailored for a single QP solution, it was observed that qpDUNES outperforms the other considered solvers by a factor of $2.7 - 3.3$ even on this benchmark scenario, cf. Figure 6.1(b). The numbers reported are in milliseconds, and averaged over 1000 identical (cold started) runs.

In a third comparison, we analyze the effect of different primal regularization parameters in the dual Newton strategy. For a horizon length of $N = 50$, and an arrival index of $\bar{k} = 45$, we compare values for σ between 10^{-2} and 10^{-8} (possibly causing very ill-conditioned dual functions) in Figure 6.2. Again, computation times are reported in milliseconds, and averaged over 1000 identical (cold started) runs. As expected, we observe that the choice of σ barely has any influence on the performance of the interior-point methods. The dual Newton strategy in contrast is sensitive to the choice of σ, and there is a tendency to higher computational demand for smaller primal regularization parameters (i.e., a more ill-conditioned dual problem), as anticipated. Still, for all considered regularization parameters, qpDUNES outperformed its competitors.

6.2.2 Chain of masses

The second benchmark example we consider here is taken from [WB10]. MPC is used to control a chain of six oscillating masses that are connected by springs to each other and to a wall on each side. We use the same parameters as stated

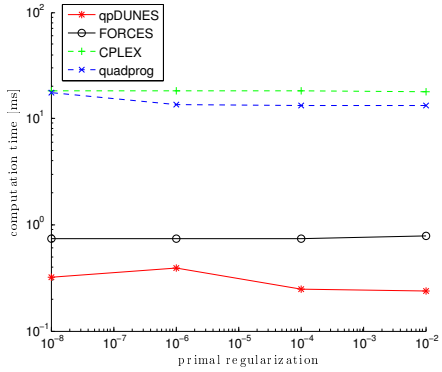
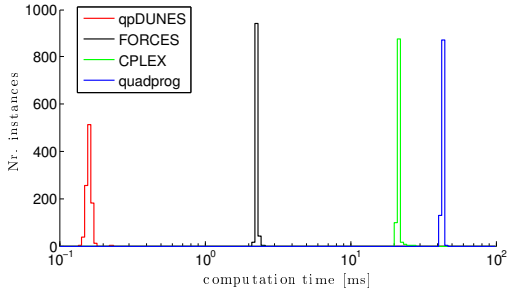


Figure 6.2: Computation times of one cold started QP solution for different primal regularization parameters.



(a) Computation time histogram. Peaks from left to right: qpDUNES, FORCES, CPLEX, quadprog.

Solver	Computation time
qpDUNES	0.16 ms
FORCES	2.27 ms
CPLEX	21.41 ms
quadprog	42.83 ms

(b) Average computation times of one QP solution.

Figure 6.3: Chain of masses benchmark problem.

in [WB10], and thus end up with an MPC problem of 12 states, 3 control inputs and a prediction horizon of 30 intervals. We simulate the MPC problem on 100 time steps.

We computed 1000 random noise vectors that perturb the positions of the masses. We again compare the computation times obtained from qpDUNES with those obtained from the solvers FORCES, CPLEX, and quadprog.

Table 6.3(b) shows average computation times for one QP solution, and Figure 6.3(a) shows a histogram of the average iteration times over the different instances. It can be seen that due to the efficient warmstarting, qpDUNES is at least one order of magnitude faster than the other considered solvers, even

in presence of perturbations. We note that qpDUNES was rarely observed to exceeded 4 iterations, while FORCES needed 6 to 9 iterations in most cases. Since the factorization in qpDUNES can be warmstarted, and the structure of the band-matrix is similar to the one occurring in interior-point methods, each iteration in the dual Newton strategy is roughly at most as expensive as an iteration of an interior-point method. It should be noted that in this benchmark problem significantly less constraints become active, a fact from which qpDUNES benefits overproportionally.

6.2.3 Hanging chain

To comment on the weaknesses of the dual Newton strategy, we consider a third problem, which has already been used for benchmarking in several publications, see [WBD06, FBD08, VDF+13]. The problem again features a chain of masses — however in three-dimensional space this time. The chain assumes its steady state very close (0.01 m) to a wall, cf. Figure 6.4. One end of the chain is fixed at the origin, while the other one is free and can be controlled by its velocities in x, y, and z direction. Note that, analogously to [VDF+13], we placed the wall closer to the equilibrium position than it has been considered in the original setting from [WBD06]. This means that potentially a large amount of state constraints becomes active in the solution, and thus we yield a more challenging problem, particularly for the dual Newton strategy. As in [FBD08] we perform linear MPC based on a linearization in the steady state, trying to stabilize the problem quickly at its equilibrium. For the detailed model equations we refer to [WBD06].

We consider a chain of 5 masses, which results in a system of 33 states (the free masses' positions and velocities) and 3 controls (the last masses' velocities), on a varying horizon length between 30 and 60 intervals. Noise is added on the velocities of each mass in each simulation step (adding noise directly on the positions might result in an infeasible problem very easily due to the closeness between the equilibrium and the constraining wall). We explicitly note that systems of such a ratio between state dimensionality and horizon length are not the targeted application domain for the dual Newton strategy. Still, if we consider average computation times over 50 MPC iterations, we observe that the dual Newton strategy performs reasonably well, cf. Figure 6.5(a). Our solver qpDUNES performs a factor of 3.4 to 5.6 faster in this comparison than FORCES, the closest competitor. It is interesting to observe that the computation time is more or less constant over all considered horizon lengths. This can be explained by the efficient warmstarting of the factorization in the MPC context, cf. Section 4.3.5. Moreover, a longer horizon will cause that the equlibrium is reached (first at the end of the prediction horizon) at an earlier

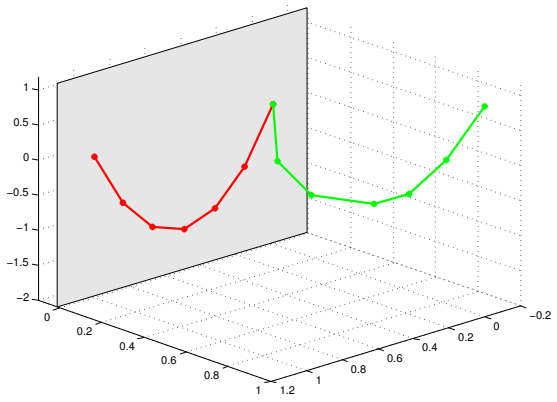


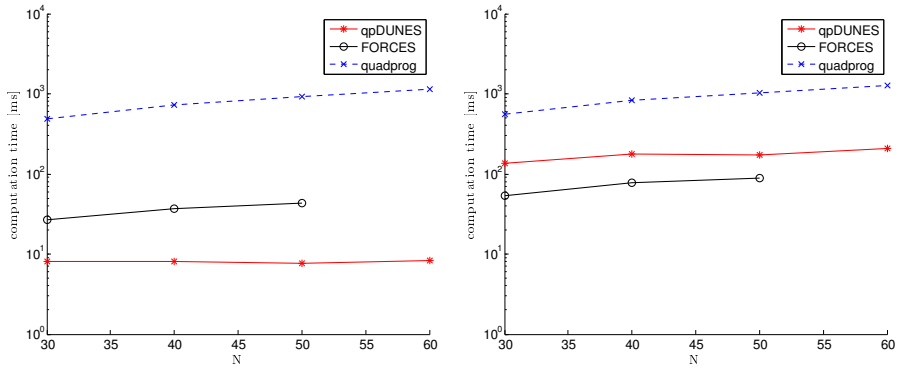
Figure 6.4: Hanging chain scenario. The chain in steady state is depicted in red, while the initial state is drawn in green.

point in the simulation, thus making the multiplier shift more effective from this point on.

In this comparison we did not include CPLEX, as its Barrier method experienced numerical problems in almost all considered noise scenarios and therefore terminated early and suboptimally. We were also not able to obtain FORCES code for a problem of 60 or more intervals length with this number of states; note that this is not a numerical limitation of the solver, but rather a technical limitation of the FORCES download server, which generates the code specifically for each problem instance.

Despite all these shortcomings of the comparison, we believe that this very challenging problem yields some important insights when we consider maximum computation times in Figure 6.5(b). Here we observe that FORCES outperforms qpDUNES by a factor of 2.5 to 2 on the considered horizon lengths. Due to the fact that the objective is driving the optimization variables to points with many active state constraints (about 70 active constraints in the solution for the $N = 50$ setting), qpDUNES takes many iterations that require regularization of the dual Hessian matrix. Overall, qpDUNES needed about twice as many iterations in the maximum as FORCES; additionally, many line search iterations were required when the dual Hessian needed to be regularized.

Nonetheless, this comparison also yields some positive news for the dual Newton strategy. We observed that the high computation times of qpDUNES were exclusively observed in the first MPC iteration, when qpDUNES was cold-started, and the initial condition is far away from the equilibrium. When



(a) Average computation times. (b) Maximum computation times.

Figure 6.5: The hanging chain benchmark problem.

excluding this first iteration from the comparison, qpDUNES outperformed FORCES again by a factor of 3 to 6.1 also in maximum computation times for one MPC iteration.

6.3 Numerical Performance in Nonlinear MPC

We provide a comparison of a condensing/qpOASES based QP solution strategy with qpDUNES within the ACADO Code Generation tool in the following. Both, condensing and qpOASES are implemented in a highly efficient manner within the ACADO Code Generation tool, as indicated in a wide range of applications, cf., e.g., [FOL+07, FBD08, HFD11b, FHGD11a, FHGD11b, FKP+13, VDF+13, FJ13, AFVD13]. All simulations were performed on a 3.4GHz Intel i7 based desktop computer, running the 64-bit version of Ubuntu Linux 13.04. All codes are compiled with Clang 3.2.1, using the flag `-O3` and execution times are measured with the Linux function `clock_gettime()`.

6.3.1 A nonlinear hanging chain of masses

The first class of benchmark problems deals with stabilizing the same strongly deflected chain of M masses connected by springs as in Section 6.2.3, and in [WBD06, FBD08, VDF+13]. Again, one end of the chain is attached to a fixed

point, while the velocity of the other end can be controlled. Here, however, we consider the full nonlinear dynamics from [WBD06].

Just as before, each mass is described by its position and velocity coordinates, resulting in a total of $n_x = 6\,M + 3$ states and $n_u = 3$ control inputs. We chose a sampling time of $T_s = 200\,\text{ms}$ and varying prediction horizon lengths $N \in \{10, \ldots, 100\}$. For integration of the nonlinear system dynamics an implicit Gauss-Legendre integrator of order four was used within the ACADO Code Generation tool (cf. [QVD12] for details), with two integration steps per discretization interval.

The ACADO Code Generation tool employs the RTI scheme, where each iteration is split into a preparation and a feedback phase. We consider the timings of these two phases separately in the following. In the condensing-based approach, the preparation phase consists of the linearization of the NLP and the condensing routine that yields the reduced-size QP, both of which are of significant computational effort. The time spent in the feedback phase is dominated by the solution of the condensed QP by qpOASES. In the sparse (qpDUNES-based) approach, no condensing routine is needed, so the preparation phase is dominated only by the effort for the linearization of the NLP. Almost all time of the feedback phase is then spent in the solution of the sparse QP.

We present average computation times for one RTI in Figure 6.6 and highlight the time spent in the feedback phase. Note that the preparation phase always has to be shorter for the sparse QP strategy in comparison with the condensing-based approach. It can be seen that for a moderate number of states n_x, the sparse approach is already competitive on short horizon lengths, while it clearly outperforms the condensing-based approach both in terms of feedback time and in terms of total iteration time on longer horizon lengths due to its lower per-iteration computational complexity. The test case for $M = 5$ shows the benefits of the condensing-based approach for systems with many states that lie in a drastically reduced size of the problem that remains to be solved in the feedback phase; still, regarding the overall iteration times, the sparse approach performs reasonably well, even though it is not targeted for this class of problems.

A similar pattern can be observed when regarding maximum computation times (over all simulation steps) in Figure 6.7. For a small to medium number of states, the sparse approach dominates already on relatively short horizon lengths, both in the feedback phase and (thus even more) in the total iteration time. Obviously the computational efforts for integration and condensing are problem-data independent and thus the relative gap between both approaches decreases a bit for long horizons in the consideration of maximum computation times, as observed in the test case of $M = 3$. For a rather large number of states

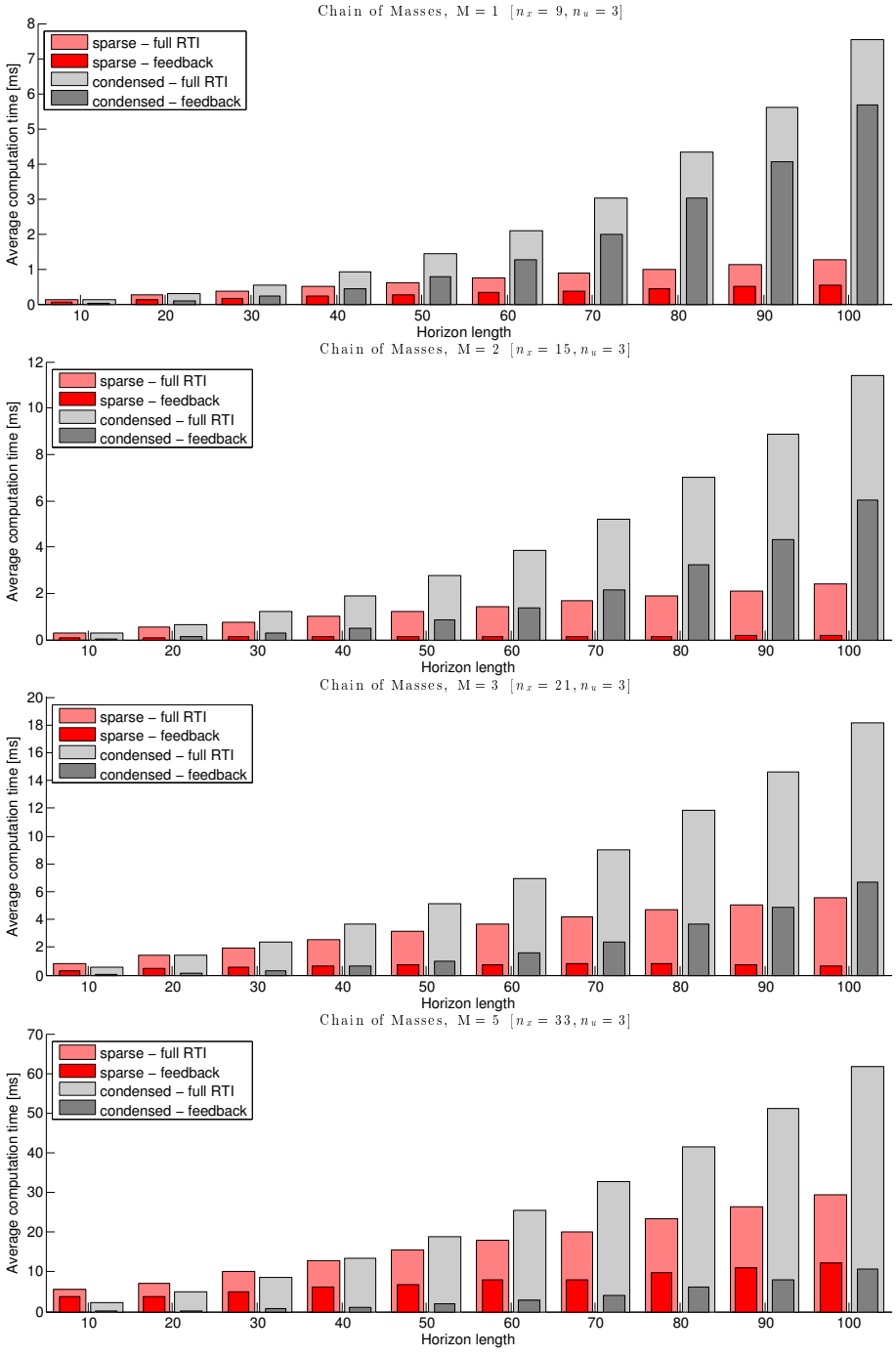


Figure 6.6: Average computation time benchmark for four chain-of-masses test cases.

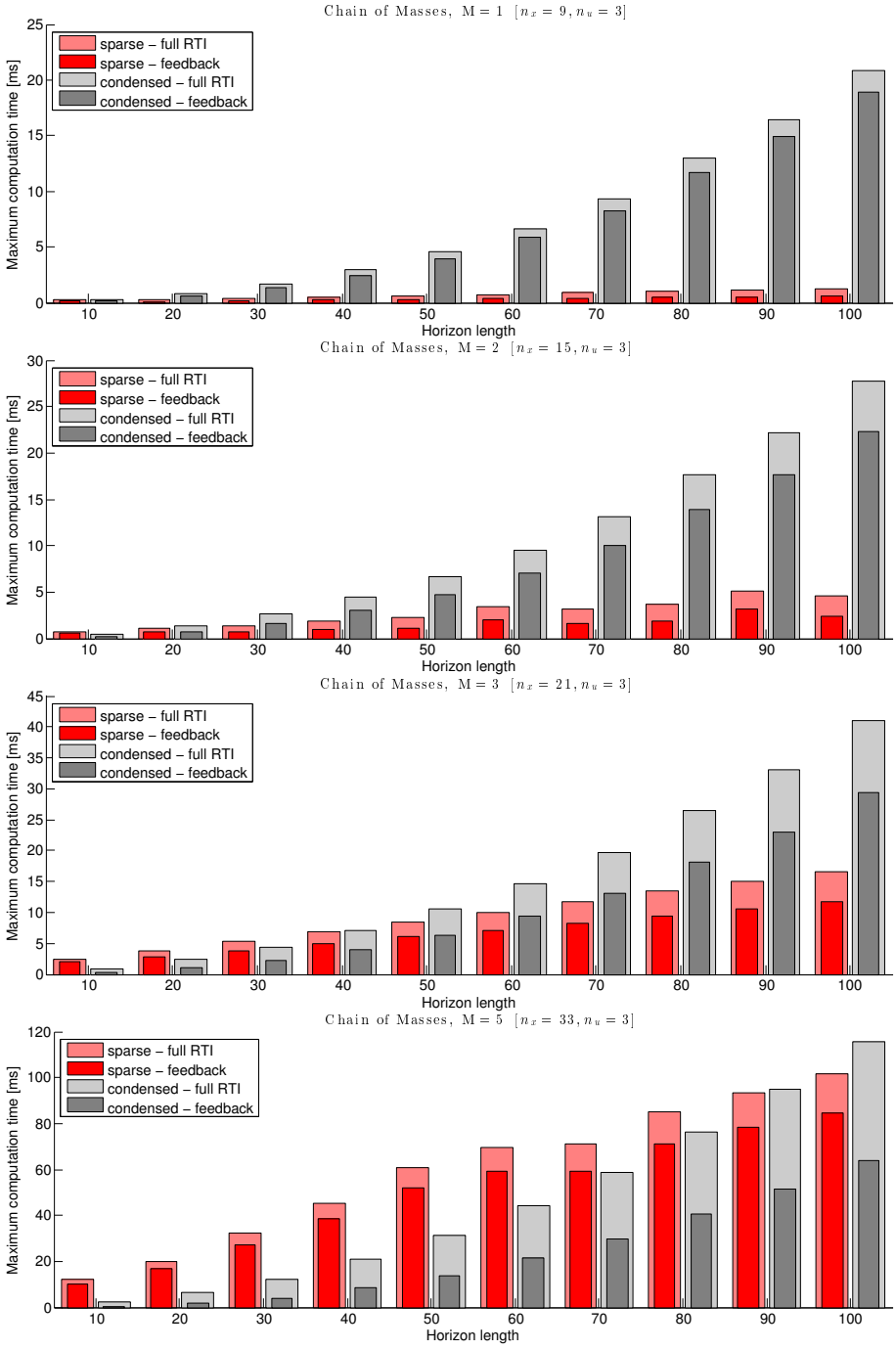


Figure 6.7: Maximum computation time benchmark for four chain-of-masses test cases.

(large at least for fast scale applications), particularly in the test cases $M = 5$, the feedback time (i.e., the pure QP solution time) of the condensed approach dominates the QP solution time of the sparse approach for all considered horizon lengths. This effect is even more visible in the maximum computation time plot, since, again, the time spent in condensing plays less of a roll here.

Particularly in the test case $M = 5$ it could be observed that the sparse approach suffers more than the condensed approach from the bad conditioning of the problems, resulting from the objective function pressing strongly against the constraints. Improving the stability of factorization and smarter regularization approaches are subject of ongoing research. In general it should be noted that the relatively large gap between average and maximum computation times of the dual Newton method can partly be explained by the wrong initalization of the QP solver after the chain is deflected. Excluding the first iteration from consideration already leads to significantly shorter maximum computation times (e.g., about 20 ms for the sparse QP solution in the $M = 5$ case on a horizon length of 100 steps). The other part of the explanation obviously is the active-set nature of the dual Newton strategy.

6.3.2 Anti-surge control of a gas compressor

While the first nonlinear benchmark problem is purely academic (but well scalable), we use a real-world motivated second benchmark problem. Nonlinear MPC is used to prevent the occurence of surge in centrifugal compressors (see [CPMB12] for details). Centrifugal compressors are widely used in gas extraction plants or gas pipelines to extract and transport natural gas from the source to the consumer. As compressing is an energy-intensive process, it is important to operate compressors efficiently in order to save resources. This means to operate them at working points that are close to surge, an instable system state that can cause severe damage to the compressor and piping system. We describe the compressor by a nonlinear ODE model similar to the one presented in [CPMB12]. It comprises $n_x = 6$ differential states and $n_u = 2$ control inputs: the opening of the recycle valve as well as the torque of the compressor's drive (which are both subject to physical limitations). Our nonlinear MPC aims at tracking a given operating point in the event of a simulated sudden closure of the compressor's outlet valve.

The anti-surge controller is running at a sampling time of 25 ms on a prediction horizon of length N as given in Figure 6.8. Again, the sparse approach performs competitively on all considered horizon lengths, tieing for horizon lengths around $N = 30$. For longer prediction horizons significant computational savings can be achieved from applying the sparse approach. Note in particular

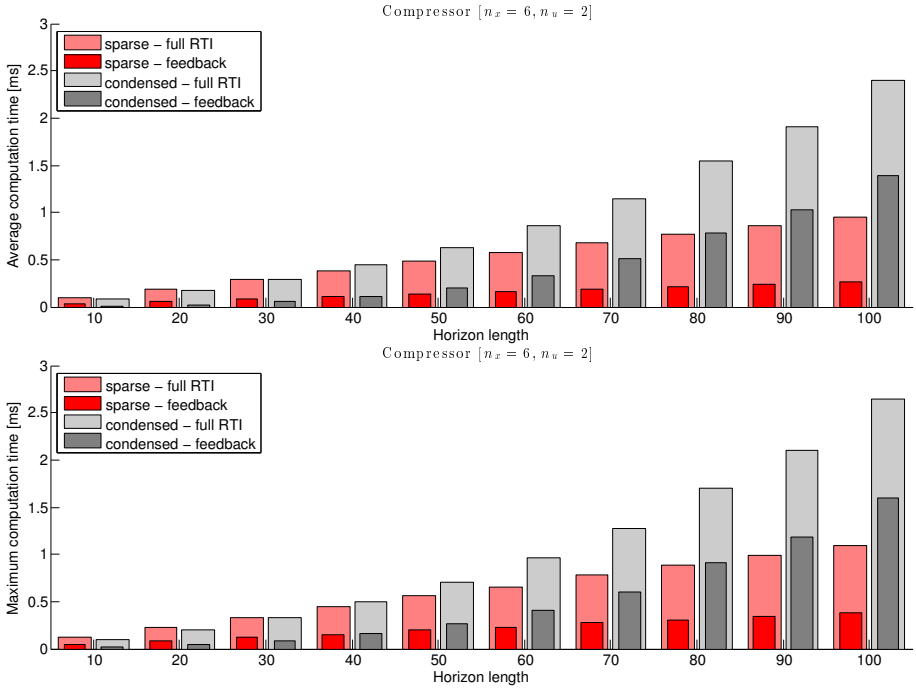


Figure 6.8: Average and maximum computation time benchmark for the compressor test case.

the approximately linear growth of QP solution times of the sparse approach (in the horizon length) in comparison with the strikingly superlinear growth of the QP solution times of the condensed approach. Due to the practical relevance, we also include maximum computation times in the lower part of Figure 6.8. Since both, the sparse and the condensed approach, rely on active-set based QP solution methods, the absolute computation times increase. The problem data independent effort for integration and condensing causes the relative performance gap between the sparse and the condensed approach on long horizons to diminish slightly, while the absolute gap even increases.

6.4 Discussion

The above numerical results indicate that the dual Newton strategy in general, and qpDUNES in particular may present advantages over established methods

and software for the solution of structured QPs in dynamic optimization.

We note that the utilized implementation of qpDUNES is essentially based on the details given in Section 4.3. Additional, possibly significant speedup can be expected from extensions of qpDUNES, e.g., by the factorization proposed in Section 4.6.1, a warmstarted forward factorization for MHE problems[2], initial preconditioned gradient iterations, and code generation. Furthermore, using preconditioned gradient iterations and a rigorous infeasibility detection, the reliability of qpDUNES can be enhanced.

An deep and efficient interconnection of qpDUNES with other algorithmic concepts for dynamic optimization NLPs, such as a Multi-Level Iteration scheme, as well as an efficient implementation of the ALADIN concept from Section 5.2 are favorable directions for future research and development. In this context, it is particularly interesting to investigate whether the RTI approach to NLP solving can be transferred to the ALADIN framework, e.g., by performing only one consensus step (after re-solving the first stage NLP for initial value embedding).

[2]Conversely to MPC, we expect active-set changes to tentatively happen rather on the last stages in MHE, which motivates the usage of a forward in time factorization order.

Chapter 7

Autonomous Driving

Model predictive control (MPC) is an established approach for assisting drivers of ground vehicles with the operation of certain lower-level components, such as engine control or path following in difficult handling conditions, cf., for example, [FOL+07, FBA+07, FBA+08] and the references therein. In this section we address how, beyond that, nonlinear MPC can be used as a *single*, holistic control scheme for the autonomous operation of ground vehicles in challenging situations. To this end, we derive a detailed model of the vehicle dynamics that goes significantly beyond those models used in automotive (online[1]) control tasks in earlier publications, e.g., [Ger05, FBA+07, SKB08, KSBS10, GLB+10, KFKS11, GGL+12]. It covers important nonlinear dynamics that play a major role when the vehicle is operated close to the limits of its handling capability and can therefore become safety-critical. Since we propose a single-level controller, limited maneuverability or infeasible path planning due to model oversimplification can be avoided[2].

We derive a representation of the vehicle dynamics in spatial (i.e., track-progress dependent) differential equations to allow for a state-independent representation of road boundaries as well as of obstacles along the track, which is inherited unalteredly in the linearized MPC problem (i.e., we have the same constraint representation in linear and in nonlinear MPC) to be solved

[1] For offline simulation of vehicle dynamics (without real-time requirements) a variety of high-fidelity tools and models exists, e.g., the open-source VDrift simulator [Ven10].

[2] This is a commonly observed effect from the mismatch between the different models in multi-layer controller designs, cf. the observations in [FBA+08, GLB+10, GGF+12] for example.

at every sampling time[3]. Above all, we stress the importance of the use of structure-exploiting, efficient numerical methods[4] to render these challenging problems computationally feasible in real-time. We demonstrate the practical applicability of the spatial vehicle dynamics representation for obstacle avoidance by experimental results from a Jaguar X-Type passenger car and show simulation results demonstrating the superiority of tailored numerical methods. Finally, we extend our considerations from safe road-following to a time-optimal driving prospect.

Acknowledgement This chapter is largely based on the following publications:

- "Spatial Predictive Control for Agile Semi-Autonomous Ground Vehicles" by Yiqi Gao, Andrew Gray, Janick Frasch, Theresa Lin, Eric Tseng, Karl Hedrick, and Francesco Borrelli [GGF+12]
- "An Auto-Generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles" by Janick Frasch, Andrew Gray, Mario Zanon, Joachim Ferreau, Sebastian Sager, Francesco Borrelli, and Moritz Diehl [FGZ+13]
- "Nonlinear Moving Horizon Estimation for Combined State and Friction Coefficient Estimation in Autonomous Driving" by Mario Zanon, Janick Frasch, and Moritz Diehl [ZFD13]
- "Model Predictive Control of Autonomous Vehicles" by Mario Zanon, Janick Frasch, Milan Vukov, and Moritz Diehl [ZFV+14]
- "Towards Time-Optimal Race Car Driving using Nonlinear MPC in Real-Time" by Robin Verschueren, Stijn De Bruyne, Mario Zanon, Janick Frasch, and Moritz Diehl [VBZ+14].

The papers [FGZ+13], [ZFD13], and [ZFV+14] propose ideas around autonomous driving, particularly with the goal of driver assistance in safety-critical situations, based on detailed first-principles vehicle models that are implementable in real-time and tested in simulation. Janick Frasch is the main author of [FGZ+13], and introduced core modelling ideas and coordinated the research activities. Mario Zanon and Janick Frasch jointly developed the

[3]Here, we anticipate the solution of the nonlinear MPC problem by the RTI scheme from Section 2.3.1.

[4]Here, we use the ACADO Code Generation implementation (cf. Section 6.3) of the RTI scheme with the condensing algorithm from Section 3.2 on the software side, as most results in this section were obtained before the completion of qpDUNES. Additionally, the horizon length to number of states ratio of the obstacle-avoidance scenarios falls rather into the métier of condensing based approaches.

extended vehicle dynamics model, which we will present in Section 7.1. The spatial transformation for the consideration of road constraints and obstacles (which was additionally also used in [GGF+12]) was derived in its current form jointly by Yiqi Gao, Andrew Gray, and Janick Frasch. The simulation results from [FGZ+13], [ZFD13], and [ZFV+14] were obtained from an ACADO Code Generation MPC/MHE design that was initiated by Janick Frasch and extensively augmented by Mario Zanon.

The papers [GGF+12] and [VBZ+14] feature experimental data, in [GGF+12] from a full-size Jaguar X-Type passenger car at Ford Research Laboratories in Dearborn, MI, USA, and in [VBZ+14] from a downscaled model car setup at Siemens LMS (now Siemens Industry Software), Leuven, Belgium. Yiqi Gao is the main author of [GGF+12], and conducted the on-site experiments with Andrew Gray and Eric Tseng. The used MPC formulation was derived jointly by Yiqi Gao, Andrew Gray, and Janick Frasch. The publication [VBZ+14] is based on results of Robin Verschueren's master thesis, which he conducted under the joint supervision by Stijn De Bruyne, Mario Zanon, and Janick Frasch. The one-level tracking MPC formulation, as well as the time-optimal MPC formulation from [VBZ+14] was developed and implemented by Robin Verschueren based on ideas derived jointly by Mario Zanon and Janick Frasch. Stijn De Bruyne coordinated the hardware setup at Siemens LMS.

Joachim Ferreau and Milan Vukov contributed to [FGZ+13] and, respectively, [ZFV+14] with support and smaller algorithmic modifications around the ACADO Code Generation tool. Theresa Lin, Karl Hedrick, Francesco Borrelli, Sebastian Sager, and Moritz Diehl contributed to [GGF+12], and, respectively, [FGZ+13], [ZFD13], [ZFV+14], and [VBZ+14] through viable feedback and supervision of the activities.

7.1 A Comprehensive Vehicle Model

We investigate several model components in the following that influence the dynamics of road vehicles. The basis is formed by a 6 degrees of freedom (DoF) vehicle model, which can be found in similar forms in the pertinent literature, e.g., [KN05]. Note that we consider the force contributions at each of the four wheels individually, in contrast to several other publications (e.g., [Ger05, SKB08, KSBS10, Keh10, KFKS11, GGF+12]), which reduced the model to a bicycle model. This extension allows us to account for effects such as load transfer, which may become critical in the limit operation range of the vehicle. On the downside, the increased model complexity requires a higher computational effort for the evaluation of the dynamic system, as well

as, possibly, additional parameters that need to be identified on a real-world vehicle.

We restrict our considerations to a car with front steering and rear wheel drive in the following. Extensions to other actuation configurations can be realized in a straightforward manner. The control inputs are the steering rate $\dot{\delta}$, the accelerating engine torque T^{a}, and four braking torques $T^{\mathrm{b}}_{\mathrm{fl}}$, $T^{\mathrm{b}}_{\mathrm{fr}}$, $T^{\mathrm{b}}_{\mathrm{rl}}$, $T^{\mathrm{b}}_{\mathrm{rr}}$. Throughout this paper we use subscripts $\mathrm{fl}, \mathrm{fr}, \mathrm{rl}, \mathrm{rr}$ to denote quantities corresponding to the front left, front right, rear left, and rear right wheel, respectively. For clarity of the notation we define $\mathcal{F} := \{\mathrm{f}, \mathrm{r}\}$ and $\mathcal{S} := \{\mathrm{l}, \mathrm{r}\}$ and use $\mathcal{F} \times \mathcal{S} = \{\mathrm{fl}, \mathrm{fr}, \mathrm{rl}, \mathrm{rr}\}$.

7.1.1 Chassis dynamics

The vehicle chassis is modeled as a rigid body, described by its global position at the vehicle's center of gravity (CoG) in the X-Y plane, its global orientation, and the corresponding velocities in a local x-y-z frame. The z-direction is pointing upwards. Initially, we disregard roll, pitch and heave (vertical displacement) motions of the car. The chassis dynamic equations consequently read

$$m\,\dot{v^x} = m\,v^y\,\dot{\psi} + F^x_{\mathrm{fr}} + F^x_{\mathrm{fl}} + F^x_{\mathrm{rr}} + F^x_{\mathrm{rl}} + F_{\mathrm{D}}\,, \tag{7.1a}$$

$$m\,\dot{v^y} = -m\,v^x\,\dot{\psi} + F^y_{\mathrm{fr}} + F^y_{\mathrm{fl}} + F^y_{\mathrm{rr}} + F^y_{\mathrm{rl}}\,, \tag{7.1b}$$

$$\begin{aligned} I^\psi\,\ddot{\psi} = {} & a\,(F^y_{\mathrm{fl}} + F^y_{\mathrm{fr}}) - b\,(F^y_{\mathrm{rl}} + F^y_{\mathrm{rr}}) \\ & + c\,(F^x_{\mathrm{fr}} - F^x_{\mathrm{fl}} + F^x_{\mathrm{rr}} - F^x_{\mathrm{rl}})\,, \end{aligned} \tag{7.1c}$$

$$\dot{X} = v^x \cos\psi - v^y \sin\psi\,, \tag{7.1d}$$

$$\dot{Y} = v^x \sin\psi + v^y \cos\psi\,, \tag{7.1e}$$

where m denotes the mass and I^ψ the yaw moment of inertia of the car. The distances of the tires from the vehicle's CoG are characterized by a, b and c, cf. Figure 7.1. The vehicle's yaw angle ψ is obtained by direct integration of $\dot{\psi}$. The drag force due to air resistance is denoted by F_{D}, while $F^x_{\diamond\star}$ and $F^y_{\diamond\star}$, $\diamond\star \in \mathcal{F} \times \mathcal{S}$ denote the components of the tire contact forces.

Model (7.1) allows to account for load transfers originating from the tire-street interaction forces only under the assumption of a rigid suspension as done in [FGZ+13, ZFD13]. This assumption can be dropped by additionally introducing dynamic equations for the pitch and roll movement of the car, denoted by p

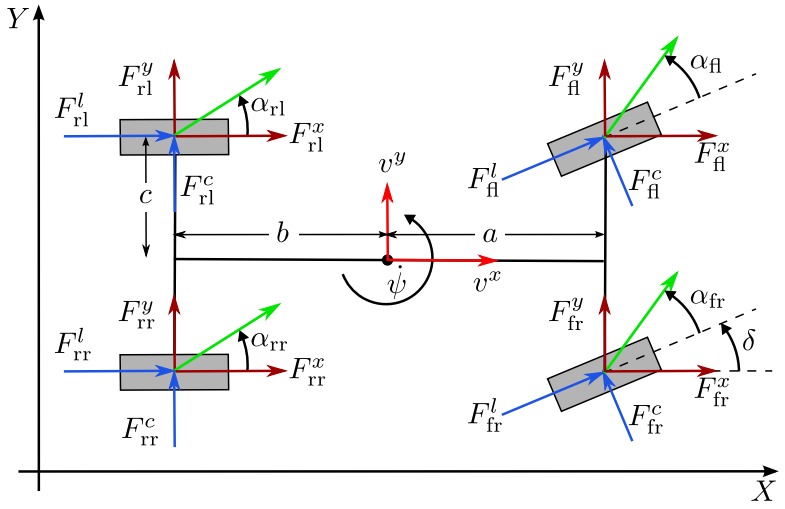


Figure 7.1: Tire forces and slip angles of the 4-wheel vehicle model in inertial coordinates. The tires' directions of movement are indicated by green vectors.

and r:

$$I^p \ddot{p} = T^p_{\mathrm{s}} + T^p_{\mathrm{lt}}\,, \tag{7.2a}$$

$$I^r \ddot{r} = T^r_{\mathrm{s}} + T^r_{\mathrm{lt}}\,. \tag{7.2b}$$

Here, T^p_{s} and T^r_{s} denote the suspension torques counteracting the load transfer torques T^p_{lt} and T^r_{lt}, while I^p and I^r denote the pitch and roll moments of inertia of the chassis. We do not assume the availability of detailed road profile information (e.g., about potholes) here and consequently neglect the heave movement of the vehicle.

Due to the front steering assumption, the longitudinal contact forces $F^{\mathrm{l}}_{\diamond\star}$ and the cornering (lateral) contact forces $F^{\mathrm{c}}_{\diamond\star}$ of each tire $\diamond\star \in \mathcal{F} \times \mathcal{S}$ enter by

$$F^x_{\mathrm{r}\star} = F^{\mathrm{l}}_{\mathrm{r}\star} \qquad \forall \star \in \mathcal{S}$$

$$F^y_{\mathrm{r}\star} = F^{\mathrm{c}}_{\mathrm{r}\star} \qquad \forall \star \in \mathcal{S}$$

in the rear, and

$$F^x_{\mathrm{f}\star} = F^{\mathrm{l}}_{\mathrm{f}\star}\cos\delta - F^{\mathrm{c}}_{\mathrm{f}\star}\sin\delta \qquad \forall\, \star \in \mathcal{S}$$

$$F^y_{\mathrm{f}\star} = F^{\mathrm{l}}_{\mathrm{f}\star}\sin\delta - F^{\mathrm{c}}_{\mathrm{f}\star}\cos\delta \qquad \forall\, \star \in \mathcal{S}$$

in the front. The steering angle δ is obtained from the input $\dot{\delta}$ by integration.

7.1.2 Pacejka tire forces

The tire contact forces $F^{\mathrm{l}}_{\diamond\star}$ and $F^{\mathrm{c}}_{\diamond\star}$ are computed from a combined longitudinal and lateral Pacejka-type tire slip model, cf. [Pac06]. For each tire $\diamond\star \in \mathcal{F}\times\mathcal{S}$, the tire model is given in abstract form by

$$(F^{\mathrm{l}}_{\diamond\star}, F^{\mathrm{c}}_{\diamond\star}) = f_{\mathrm{P}}(\alpha_{\diamond\star}, \kappa_{\diamond\star}, \mu, F^z_{\diamond\star})\ .$$

The inputs to this model are the side slip angles $\alpha_{\diamond\star}$, the slip ratio $\kappa_{\diamond\star}$, and the normal load $F^z_{\diamond\star}$. The influence of road friction is modeled by a parameter $\mu \in [0,1]$ that also enters the tire model. The precise semi-empirical model equations of Pacejka's Magic Formula can be found, for example, in [Pac06, KFKS11, GZF12]. Note that we neglected the aligning moments here, as they typically have little influence in high velocity driving.

In the Pacejka model, the side slip angle is defined as the angle between the wheel's orientation and the actual direction of movement (cf. Figure 7.1) and can be obtained from the chassis velocity and the steering angle by simple geometrical relations.

The slip ratio is defined as $\kappa_{\diamond\star} = \frac{\omega_{\diamond\star}R_{\mathrm{e}} - v_{\diamond\star}}{v_{\diamond\star}}$ and represents the longitudinal slip during acceleration and deceleration. Here, $v_{\diamond\star}$ denotes the wheel speed with respect to the ground (which can be computed explicitly from the chassis velocities), while R_{e} denotes the effective tire rolling radius (cf. [Pac06]). The rotational speed of the wheel is denoted by $\omega_{\diamond\star}$. Its computation is detailed in Section 7.1.3.

We will also elaborate on how to obtain $F^z_{\diamond\star}$ in Sections 7.1.5 and 7.1.6.

7.1.3 Wheel dynamics

The four wheels are modeled as independent bodies with only spinning inertia. The wheels' rotational velocities $\omega_{\diamond\star}$ are computed from a first order model in the accelerating torques $T^{\mathrm{a}}_{\diamond\star}$ and the braking torques $T^{\mathrm{b}}_{\diamond\star}$, taking the wheel

moment of inertia I^{w} (which is here assumed to be identical for all wheels) into account. Note that the consideration of wheel dynamics enhances the predictive quality of the model but also renders the underlying ODE system stiff, making its solution computationally significantly more challenging. The wheel dynamic equations are given by

$$\dot{\omega}_{\diamond\star} = \frac{1}{I^{\mathrm{w}}}(T^{\mathrm{a}}_{\diamond\star} + T^{\mathrm{b}}_{\diamond\star} - R_{\mathrm{e}}F^{\mathrm{l}}_{\diamond\star}) \qquad \forall\ \diamond\star \in \mathcal{F}\times\mathcal{S}\,, \tag{7.3}$$

where again R_{e} denotes the effective tire rolling radius. We assume individual wheel braking (i.e., $T^{\mathrm{b}}_{\diamond\star}$ are inputs for all $\diamond\star \in \mathcal{F}\times\mathcal{S}$) and include a differential model for the total acceleration torque T^{a}, yielding, for a rear-wheel driven car,

$$T^{\mathrm{a}}_{\mathrm{f}\star} = 0, \qquad \forall\ \star \in \mathcal{S}$$

$$T^{\mathrm{a}}_{\mathrm{r}\star} = T^{\mathrm{a}}\left(1 - \frac{\omega_{\mathrm{r}\star}}{\omega_{\mathrm{rl}} + \omega_{\mathrm{rr}}}\right) \qquad \forall\ \star \in \mathcal{S}\,.$$

We can either assume T^{a} to directly be an input, i.e., assume its realization through a lower-level engine controller, or include an engine model, as detailed in Section 7.1.4.

7.1.4 Engine model

In some contexts, like time-optimal driving for example, the simplification that the acceleration torque T^{a} is available (almost) instantaneously by realization through a lower-level engine controller may be too strong to achieve good control performance. A remedy to this may be to approximate the acceleration torque set-point change behavior by a first-order model, or to include a detailed engine model.

The latter has been achieved in [Keh10, KFKS11] for example, where a detailed engine model has been extracted from the VDrift simulation environment [Ven10] and implemented in an optimal control formulation. The proposed model can essentially be summarized by the form

$$T^{a} = q(\Upsilon)\cdot(T^{\mathrm{cb}} + T^{\mathrm{fr}}), \tag{7.4}$$

where $q(\Upsilon)$ is the transmission ratio between the engine and the driving axle, which depends on the integer choice of the gear[5] $\Upsilon \in \mathbb{N}$, and where T^{cb} and T^{fr} denote the (positive) torque due to combustions and the (negative) torque due to friction.

[5]Even though, technically speaking, Υ is a control, i.e., we would have $\Upsilon : \mathcal{T} \to \mathbb{N}$, we neglect the time dependency here for notational convenience.

For a continuous accelerator $\Phi \in [0,1]$, the combustion torque is given by

$$T^{\mathrm{cb}} = \Phi \cdot F^{\mathrm{tq}}(\omega^{\mathrm{e}}), \tag{7.5}$$

where ω^{e} denotes the rotational engine speed and $F^{\mathrm{tq}}(\omega^{\mathrm{e}})$ is an interpolated torque curve from a measurement grid. The engine friction torque is given by

$$T^{\mathrm{fr}} = (1-\Phi) \cdot C_1 \cdot \omega^{\mathrm{e}}, \tag{7.6}$$

where $C_1 < 0$ summarizes constant coefficients that enter in the friction term. Under the assumption that the transmission is engaged, we can express the rotational engine speed directly in terms of gear choice (transmission ratio) and the driving wheels' rotational velocity:

$$\omega^{\mathrm{e}} = q(\Upsilon) \cdot \frac{1}{2}(\omega_{\mathrm{rl}} + \omega_{\mathrm{rr}}).$$

One way to treat the integer gear decision in the optimal control problem is to use a *partial outer convexification* reformulation of all dynamic equations of the problem where the gear-dependent transmission ration $q(\Upsilon)$ enters. This has, for example, been performed in [SKB08, KSBS10, Keh10, KFKS11]. The essential idea of this approach is to introduce binary multipliers (controls) $\eta_i,\ i \in \mathcal{G} := \{1,\dots,n_{\mathrm{gears}}\}$ for each possible integer control action (i.e., gear choice) $\Upsilon = i$ (for any $i \in \mathcal{G}$) in order to reformulate the (abstract) dynamic system with state $\boldsymbol{\xi}$, continuous controls $\boldsymbol{u}$, and integer control Υ,

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t), \Upsilon(t)),$$

to

$$\dot{\boldsymbol{\xi}}(t) = \sum_{i \in \mathcal{G}} \eta_i(t) \cdot \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t), i),$$

$$1 = \sum_{i \in \mathcal{G}} \eta_i(t)$$

for $t \in \mathcal{T}$. This reformulation allows to relax integral gear choices to continuous inputs $\eta_i(t) \in [0,1]$, and to recover integrality by an appropriate rounding scheme where necessary[6,7]. Further details on this approach can be found in [SKB08, SRB09, KSBS10].

[6] Often, the relaxed solution computed for the outer convexification reformulation is already of bang-bang type and therefore integer feasible, cf. [KSBS10].

[7] If engine speed constraints need to be incorporated, a more elaborated approach is necessary, see [JKS13] for a survey of possible problem formulations and methods.

While the just stated approach is, in some sense, the natural way to treat the combinatorial nature of the gear choice, it comes at the price of (possibly significantly) higher expenditure for the evaluation and derivative generation of the dynamic system, which canonically increases linearly in the number of possible integer events (i.e., gear choices). Furthermore, the number of continuous control variables of the relaxed problem also grows (affinely) with the number of permitted integer choices.

As an alternative to this approach, we can exploit the fixed correspondence[8] between rotational engine speed and the driving axle's rotational speed to (offline) precompute automatic shifting points in dependency of the driving axle's rotational speed (which is a state), i.e., take out the integer decision. This approach works rather well in time-optimal driving, for example, since the maximum positive (acceleration) torque T^a can be scaled continuously for any (state-dependent) value ω^{e} between[9] 0 and $q(\Upsilon) \cdot F^{\mathrm{tq}}(\omega^{\mathrm{e}})$ by the continuous accelerator Φ. We can therefore determine Υ explicitly in dependency of the driving axle's rotational speed $\frac{1}{2}(\omega_{\mathrm{rl}} + \omega_{\mathrm{rr}})$ such that the *disposable* accelerating torque $q(\Upsilon) \cdot F^{\mathrm{tq}}(\omega^{\mathrm{e}})$ is maximal. Since the torque curve is typically defined through an interpolation of a measurement grid, we can simply rescale the measurement grid by each gear's transmission ratio and use an appropriate interpolation of the maximum measurement values at each axle speed point to obtain the torque curve of the automatic gearshift.

Introducing this automatic gear shift model permits to realize any desired accelerating torque value without the need for treating integer variables in the online control problem. On the other hand, however, the choice of shifting points by maximum available acceleration torque may not always permit to allocate to maximum physically disposable negative engine torque (engine braking); still, this effect is typically not impeding the control performance in time-optimal driving, as in modern passenger cars the braking systems are usually (owing to antiblock controllers) well overdesigned and tire-road interaction forces constitute the performance-limiting bottleneck rather than the available braking torque. Still, in the view of other control objectives, such as energy-optimal driving (particularly of hybrid electric vehicles which use regenerative breaking), an automatic gearshift reformulation may not always be possible.

[8] We can assume the correspondence between rotational engine speed and the driving axle's rotational speed to be fixed whenever the transmission is engaged.

[9] Recall that $C_1 < 0$ and $\omega^{\mathrm{e}} > 0$ for forward driving.

7.1.5 Vertical forces: a load transfer model

One way to obtain the vertical loads $F^z_{\diamond\star}$ acting on each wheel $\diamond\star \in \mathcal{F} \times \mathcal{S}$ is to assume a rigid suspension[10]. Then, the $F^z_{\diamond\star}$ are computed as the equilibrium of the main body with respect to the vertical forces and the pitch and roll torques. The pitch and roll torques of the chassis, T^p_{lt} and T^r_{lt}, are induced by the forces acting on the wheels and consequently are, for a height h between the vehicle's CoG and its projection onto the wheelbase, given by

$$T^p_{\mathrm{lt}} = -(F^x_{\mathrm{fl}} + F^x_{\mathrm{fr}} + F^x_{\mathrm{rl}} + F^x_{\mathrm{rr}})\, h\,, \tag{7.7a}$$

$$T^r_{\mathrm{lt}} = (F^y_{\mathrm{fl}} + F^y_{\mathrm{fr}} + F^y_{\mathrm{rl}} + F^y_{\mathrm{rr}})\, h\,. \tag{7.7b}$$

The equilibrium longitudinal and lateral load transfer components acting on the wheels, $\tilde{\Delta}^{F^z}_{\mathrm{lon}}$ and $\tilde{\Delta}^{F^z}_{\mathrm{lat}}$, are then given by

$$2\,\tilde{\Delta}^{F^z}_{\mathrm{lon}} = \frac{T^p_{\mathrm{lt}}}{a+b}\,,$$

$$2\,\tilde{\Delta}^{F^z}_{\mathrm{lat}} = \frac{T^r_{\mathrm{lt}}}{2\,c}\,.$$

Note that this formulation of the load transfer implies an algebraic loop through the tire model, which can be relaxed by introducing first order models with a time constant τ_{LT} for the load transfer states $\Delta^{F^z}_{\mathrm{lon}}$ and $\Delta^{F^z}_{\mathrm{lat}}$,

$$\dot{\Delta}^{F^z}_{\bullet} = \frac{1}{\tau_{\mathrm{LT}}}(\tilde{\Delta}^{F^z}_{\bullet} - \Delta^{F^z}_{\bullet}) \qquad \forall\, \bullet \in \{\mathrm{lon}, \mathrm{lat}\}\,. \tag{7.8}$$

Denoting the rest normal loads by $\bar{F}^z_{\diamond\star}$, the vertical forces $F^z_{\diamond\star}$ are then given by

$$F^z_{\mathrm{fl}} := \bar{F}^z_{\mathrm{fl}} + \Delta^{F^z}_{\mathrm{lon}} - \Delta^{F^z}_{\mathrm{lat}}\,,$$

$$F^z_{\mathrm{fr}} := \bar{F}^z_{\mathrm{fr}} + \Delta^{F^z}_{\mathrm{lon}} + \Delta^{F^z}_{\mathrm{lat}}\,,$$

$$F^z_{\mathrm{rl}} := \bar{F}^z_{\mathrm{rl}} - \Delta^{F^z}_{\mathrm{lon}} - \Delta^{F^z}_{\mathrm{lat}}\,,$$

$$F^z_{\mathrm{rr}} := \bar{F}^z_{\mathrm{rr}} - \Delta^{F^z}_{\mathrm{lon}} + \Delta^{F^z}_{\mathrm{lat}}\,.$$

7.1.6 Vertical forces: a suspension model

For application cases in which the assumption of a rigid suspension is too restrictive, we develop a spring/suspension model in the following.

[10] For passenger cars, assuming a rigid suspension may be an oversimplification, but for model cars, this may actually lead to an accurate description of the vehicle behavior.

The elastic and damping forces of each suspension $\diamond\star \in \mathcal{F} \times \mathcal{S}$ read

$$F^{\mathrm{el}}_{\diamond\star} := -k_\diamond \, \Delta^z_{\diamond\star} \,,$$

$$F^{\mathrm{d}}_{\diamond\star} := -D_\diamond \, \dot{\Delta}^z_{\diamond\star} \,,$$

where $k_\diamond$ and $D_\diamond$ for $\diamond \in \mathcal{F}$ are the elastic and damping constants of the suspension springs, and $\Delta^z_{\diamond\star}$ is the vertical displacement of the chassis at the corresponding suspension. The vertical displacement is caused by the roll and pitch angle configuration of the chassis, r and p, which introduces a chassis rotation through

$$\boldsymbol{R} := \boldsymbol{R_y}(p)\, \boldsymbol{R_x}(r) := \begin{bmatrix} \cos p & 0 & \sin p \\ 0 & 1 & 0 \\ -\sin p & 0 & \cos p \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{bmatrix} .$$

Consequently, the vertical displacement of the chassis at each suspension is given by

$$\Delta^z_{\mathrm{fl}} := -a \, \sin p + c \, \cos p \, \sin r \,,$$

$$\Delta^z_{\mathrm{fr}} := -a \, \sin p - c \, \cos p \, \sin r \,,$$

$$\Delta^z_{\mathrm{rl}} := \quad b \, \sin p + c \, \cos p \, \sin r \,,$$

$$\Delta^z_{\mathrm{rr}} := \quad b \, \sin p - c \, \cos p \, \sin r \,,$$

and changes at the rates of

$$\dot{\Delta}^z_{\mathrm{fl}} := -a \, \dot{p} \, \cos p - c \, \dot{p} \, \sin p \, \sin r + c \, \dot{r} \, \cos p \, \cos r \,,$$

$$\dot{\Delta}^z_{\mathrm{fr}} := -a \, \dot{p} \, \cos p + c \, \dot{p} \, \sin p \, \sin r - c \, \dot{r} \, \cos p \, \cos r \,,$$

$$\dot{\Delta}^z_{\mathrm{rl}} := \quad b \, \dot{p} \, \cos p - c \, \dot{p} \, \sin p \, \sin r + c \, \dot{r} \, \cos p \, \cos r \,,$$

$$\dot{\Delta}^z_{\mathrm{rr}} := \quad b \, \dot{p} \, \cos p + c \, \dot{p} \, \sin p \, \sin r - c \, \dot{r} \, \cos p \, \cos r \,.$$

The suspension forces in combination with the rest normal loads $\bar{F}^z_{\diamond\star}$ then define the vertical forces $F^z_{\diamond\star}$ by

$$F^z_{\diamond\star} := \bar{F}^z_{\diamond\star} + F^{\mathrm{el}}_{\diamond\star} + F^{\mathrm{d}}_{\diamond\star} \qquad\qquad \forall \;\; \diamond\star \in \mathcal{F} \times \mathcal{S} \,.$$

At the same time, the suspension forces introduce a roll torque given by

$$\begin{aligned} T^r_{\mathrm{s}} &:= \left(F^{\mathrm{el}}_{\mathrm{fl}} + F^{\mathrm{el}}_{\mathrm{rl}}\right) \, c - \left(F^{\mathrm{el}}_{\mathrm{fr}} + F^{\mathrm{el}}_{\mathrm{rr}}\right) \, c + \left(F^{\mathrm{d}}_{\mathrm{fl}} + F^{\mathrm{d}}_{\mathrm{rl}}\right) \, c - \left(F^{\mathrm{d}}_{\mathrm{fr}} + F^{\mathrm{d}}_{\mathrm{rr}}\right) \, c \\ &= -2(k_{\mathrm{f}} + k_{\mathrm{r}}) \, c^2 \, \cos p \, \sin r - 2 \, (D_{\mathrm{f}} + D_{\mathrm{r}}) \, c^2 \, \left(\dot{p} \, \sin p \, \sin r - \dot{r} \cos p \cos r\right) , \end{aligned}$$

and a pitch torque given by

$$\begin{aligned} T_{\mathrm{s}}^{p} &:= -\left(F_{\mathrm{fl}}^{\mathrm{el}} + F_{\mathrm{fr}}^{\mathrm{el}}\right) a + \left(F_{\mathrm{rl}}^{\mathrm{el}} + F_{\mathrm{rr}}^{\mathrm{el}}\right) b - \left(F_{\mathrm{fl}}^{\mathrm{d}} + F_{\mathrm{fr}}^{\mathrm{d}}\right) a + \left(F_{\mathrm{rl}}^{\mathrm{d}} + F_{\mathrm{rr}}^{\mathrm{d}}\right) b \\ &= -2\left(k_{\mathrm{f}}\, a^2 + k_{\mathrm{r}}\, b^2\right) \sin p - 2\left(D_{\mathrm{f}}\, a^2 + D_{\mathrm{r}}\, b^2\right) \dot{p} \cos p\,, \end{aligned}$$

which influence the chassis's roll and pitch angle configuration, r and p, through Equations (7.2). Note that for the actual implementation, we make use of a small angle approximation in the suspension, i.e., we choose

$$T_{\mathrm{s}}^{r} \approx -2\left(\left(k_{\mathrm{f}} + k_{\mathrm{r}}\right) c^2 \sin r + \left(D_{\mathrm{f}} + D_{\mathrm{r}}\right) c^2\, \dot{r}\right),$$

and

$$T_{\mathrm{s}}^{p} \approx -2\left(\left(k_{\mathrm{f}}\, a^2 + k_{\mathrm{r}}\, b^2\right) \sin p + \left(D_{\mathrm{f}}\, a^2 + D_{\mathrm{r}}\, b^2\right) \dot{p}\right).$$

The counteracting torques due to the chassis load transfer are given by Equations (7.7).

7.2 Spatial Reformulation of Vehicle Dynamics

We propose a model transformation from time-dependent vehicle dynamics to track-dependent (spatial) dynamics. This allows a natural formulation of road bounds and obstacles by simple box constraints even under varying vehicle speed. Similar ideas have been developed in the area of robotics before, see, e.g., [PJ87].

7.2.1 Coordinate system transformation

We project the X-Y coordinates of the vehicle on a curve σ which we take to be the center-line of a road. The ODE model is then stated with respect to the independent variable s, the parametrization of σ by its arc-length. The states X, Y, and ψ are replaced by

$$e^{y} := (Y - Y^{\sigma}) \cos \psi^{\sigma} - (X - X^{\sigma}) \sin \psi^{\sigma}$$

and

$$e^{\psi} := \psi - \psi^{\sigma},$$

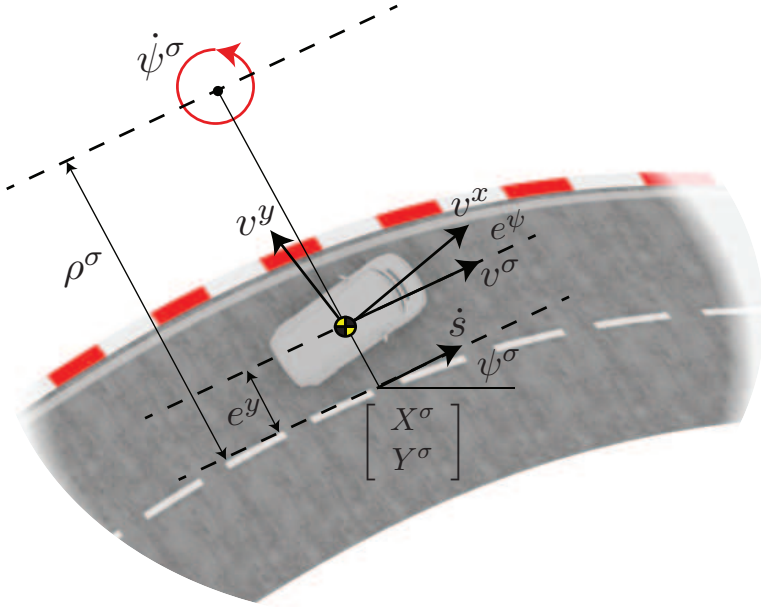


Figure 7.2: The curvilinear coordinate system. The dynamics are derived about a curve defining the center-line of a track. The coordinate s is defined as the arc-length along the track. The relative spatial coordinates e^y and e^ψ are shown. Figure taken from [FGZ+13].

where (X^σ, Y^σ) and ψ^σ denote the position and orientation of the current reference point on the path given by s. Figure 7.2 details the states in the new curvilinear coordinate system.

The spatial dynamics of the state vector ξ in relation to the time dependent dynamics are

$$\xi' := \frac{d\xi}{ds} = \frac{d\xi}{dt}\,\frac{dt}{ds}\,.$$

If $\dot{s} \neq 0$ is assumed at any time, i.e., if the vehicle is always traveling (w.l.o.g.) forward along the reference curve with a non-vanishing speed, we have $\frac{dt}{ds} = \frac{1}{\dot{s}}$ by the inverse function theorem. It therefore holds

$$\xi' = \frac{1}{\dot{s}}\,\dot{\xi}\,,$$

where $\dot{\xi}$ is defined by the time-dependent system dynamics of the vehicle. For the computation of $\dot{s}$ we observe in Figure 7.2 that

$$v^\sigma = (\rho^\sigma - e^y)\,\dot{\psi}^\sigma$$

and

$$v^\sigma = v^x\,\cos(e^\psi) - v^y\,\sin(e^\psi)$$

holds, where $\dot{\psi}^\sigma$ is the rate of change of the path orientation ψ^σ and ρ^σ is the radius of local curvature of σ. The vehicle's velocity along σ, $\dot{s} = \frac{ds}{dt}$, is then given by

$$\begin{aligned}\dot{s} &= \rho^\sigma\,\dot{\psi}^\sigma \\ &= \frac{\rho^\sigma}{\rho^\sigma - e^y}\,(v^x\,\cos(e^\psi) - v^y\,\sin(e^\psi)) \\ &= \frac{1}{1 - \frac{e^y}{\rho^\sigma}}\,(v^x\cos(e^\psi) - v^y\sin(e^\psi))\,.\end{aligned}$$

Note that $e^y < \rho^\sigma$ always needs to be fulfilled in order to guarantee uniqueness of the projection onto the centerline. This essentially means that the car needs to drive sufficiently close to the reference when the road curvature is high. Also note that ρ^σ only depends on the parametrization s through $\rho^\sigma(s) = \left(\frac{d^2}{ds^2}\sigma(s)\right)^{-1}$ but is independent of system state and input.

If necessary, time information may be recovered by integrating $\frac{dt}{ds}$ along σ:

$$t(s) = \int_{s_0}^{s} \frac{1}{\dot{s}(\tau)}\,\mathrm{d}\tau\,.$$

Inertial coordinates may be recovered by a transformation from the spatial coordinates to the global coordinates:

$$\begin{aligned}X &= X^\sigma - e^y\,\sin(\psi^\sigma) \\ Y &= Y^\sigma + e^y\,\cos(\psi^\sigma) \\ \psi &= \psi^\sigma + e^\psi\,.\end{aligned}$$

7.2.2 Model summary

We list the full system of states ξ and control inputs ν (in spatial coordinates) in the following. For the basic chassis and wheel dynamics we have:

State	Unit	Description
v^x	m/s	Longitudinal velocity of vehicle
v^y	m/s	Lateral velocity of vehicle
$\dot{\psi}$	rad/s	Vehicle's yaw rate
e^ψ	rad	Yaw angle relative to path
e^y	m	Deviation from center-line
δ	rad	Steering angle
ω_{fl}	rad/s	Rotational velocity front left wheel
ω_{fr}	rad/s	Rotational velocity front right wheel
ω_{rl}	rad/s	Rotational velocity rear left wheel
ω_{rr}	rad/s	Rotational velocity rear right wheel

If we choose to assume a rigid suspension, we additionally get the states:

State	Unit	Description
Δ^z_{lon}	N	Vertical load transfer in longitudinal direction
Δ^z_{lat}	N	Vertical load transfer in lateral direction

If, on the other hand, we include the suspension model from Section 7.1.6, we have the additional states:

State	Unit	Description
r	rad	Chassis roll angle
$\dot{r}$	rad/s	Roll angular velocity of the chassis
p	rad	Chassis pitch angle
$\dot{p}$	rad/s	Pitch angular velocity of the chassis

The inputs to the vehicle model read

Control	Unit	Description
$\dot{\delta}$	rad	Front steering rate
T^{a}	N	Engine torque
$T^{\mathrm{b}}_{\mathrm{fl}}$	N	Front left brake torque
$T^{\mathrm{b}}_{\mathrm{fr}}$	N	Front right brake torque
$T^{\mathrm{b}}_{\mathrm{rl}}$	N	Rear left brake torque
$T^{\mathrm{b}}_{\mathrm{rr}}$	N	Rear right brake torque

if we choose to omit the engine model, and

Control	Unit	Description
$\dot{\delta}$	rad	Front steering rate
Φ	-	Normalized accelerator
$T_{\mathrm{fl}}^{\mathrm{b}}$	N	Front left brake torque
$T_{\mathrm{fr}}^{\mathrm{b}}$	N	Front right brake torque
$T_{\mathrm{rl}}^{\mathrm{b}}$	N	Rear left brake torque
$T_{\mathrm{rr}}^{\mathrm{b}}$	N	Rear right brake torque
η_i	-	(Binary) gear multiplier for each gear $i \in \mathcal{G}$

if we include the engine model. Note that the η_i controls drop out if we choose to implement an automatic gear shift.

7.3 Agile Collision Avoidance

In a first scenario, we are interested in employing the spatial vehicle model for obstacle avoidance of a road vehicle. Therein, we assume that the decision in navigating to the left or right of an obstacle is made by a higher-level obstacle recognition algorithm, beyond the scope of this thesis. Then, obstacle and road boundary constraints are modeled as simple bounds on the state vector in the spatial dynamics formulation.

The control problem task is to track the road centerline (given by $e_y = 0$ in the spatial coordinate system) and a reference velocity of 10 m/s in icy driving conditions ($\mu \approx 0.3$) while respecting the road bounds and several obstacles that render the track centerline infeasible. The infinite dimensional optimization problem to be solved on a receding (spatial) prediction horizon $[s_0, s_f]$ can then by summarized as

$$\min_{\boldsymbol{\xi}(\cdot),\boldsymbol{\nu}(\cdot)} \quad \int_{s_0}^{s_\mathrm{f}} \|\boldsymbol{\xi}(\tau) - \boldsymbol{\xi}_{\mathbf{ref}}(\tau)\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{\nu}(\tau)\|_{\boldsymbol{R}}^2 \,\mathrm{d}\tau$$

$$+\|\boldsymbol{\xi}(s_\mathrm{f}) - \boldsymbol{\xi}_{\mathbf{ref}}(s_\mathrm{f})\|_{\boldsymbol{P}_{\mathbf{LQR}}}^2 \tag{7.9a}$$

$$\text{s.t.} \quad \boldsymbol{\xi}'(s) = \boldsymbol{f}\left(s, \boldsymbol{\xi}(s), \boldsymbol{\nu}(s)\right) \qquad \forall\ s \in [s_0, s_f] \tag{7.9b}$$

$$e^y(s) \in [e_L^y(s), e_U^y(s)] \qquad \forall\ s \in [s_0, s_f] \tag{7.9c}$$

$$\boldsymbol{\nu}(s) \in [-1, 1] \times [0, 1] \times [-1, 0]^4 \qquad \forall\ s \in [s_0, s_f] \tag{7.9d}$$

$$\boldsymbol{\xi}(s_0) = \boldsymbol{\xi_0}\,. \tag{7.9e}$$

Here, we denote the state by $\boldsymbol{\xi} : \mathbb{R} \to \mathbb{R}^{n_x}$ and the control input by $\boldsymbol{\nu} : \mathbb{R} \to \mathbb{R}^{n_u}$. No engine model is assumed here, i.e., the accelerating torque T^{a} serves directly as an input, and all inputs are scaled for numerical performance. Here, $\|\cdot\|_{\{\boldsymbol{Q},\boldsymbol{R},\boldsymbol{P}_{\mathbf{LQR}}\}}$ denotes the Euclidean norm with weighting matrices $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{P}_{\mathbf{LQR}}$, respectively, while $\xi_0 \in \mathbb{R}^{n_x}$ is the current state measurement and $\xi_{\mathrm{ref}} : \mathbb{R} \to \mathbb{R}^{n_x}$ denotes the parametric reference vector. The terminal weighting matrix $\boldsymbol{P}_{\mathbf{LQR}}$ is obtained as the solution of the Riccati equation, computed with the chosen weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ (cf. Section 2.1.2). Lower and upper road bounds, taking obstacles into account, are denoted by $e_L^y(\cdot)$ and $e_U^y(\cdot)$, respectively.

As the proposed control scheme is tracking the centerline reference, it will try to circumnavigate obstacles as tightly as possible, causing the obstacle (track) bounds to become active. In the presence of perturbations, this may lead to infeasibility of the optimization problems close to the obstacle. A reliable control problem formulation may therefore need to relax the obstacle constraints (including a safety margin) using a slack variable reformulation to avoid infeasible MPC subproblems. A combined ℓ^1 and ℓ^2 penalty can therein be used to obtain a non-vanishing gradient even on small constraint violation[11], while having a fast-growing cost for larger violations.

7.3.1 Experimental validation

A reduced version of the spatial model derived in Sections 7.1 and 7.2 has been validated experimentally at a Ford vehicle test center equipped with icy and snowy handling tracks in Michigan, USA, cf. [GGF+12]. The used model neglects the wheel dynamics, the engine model, as well as both load transfer and suspension. The vehicle model parameters from a Jaguar X-Type passenger car were used, and can be found at [GZF12]. The road-friction coefficient on the test track was assumed with $\mu = 0.3$.

The test setup features a two-level controller setup with a spatial path planner that runs at a sampling rate of 5 Hz and a lower-level path follower that runs at a sampling rate of 20 Hz and uses move blocking to reduce the computational demand. Both controllers were run in a dSPACE Autobox system, equipped with a DS1005 processor board and a DS2210 I/O board. The infinite dimension optimization problem has been discretized using Bock's multiple shooting method. Since wheel dynamics were neglected, the obtained IVPs are non-stiff,

[11] This formulation is aimed at avoiding a violation of the (now relaxed) path constraints whenever possible. If fact, if the ℓ^1 penalty term is chosen sufficiently large, this can be guaranteed.

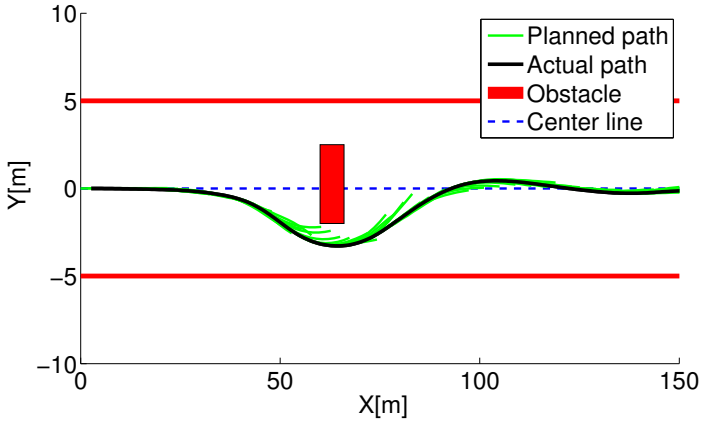


Figure 7.3: Experimental results (planned and realized path) of the test vehicle avoiding a single obstacle. Figure taken from [GGF+12].

and an (approximate) solution could be obtained by an explicit discretization of stepsize 1. Computations on the real-time system were carried out using a variant of NPSOL [GMSW01] for the NLP solution. Further details on the setup can be found in [GGF+12].

Experimental results are shown in Figures 7.3 and 7.4. It can be observed, that both, in the single obstacle scenario and in the two obstacle scenario, a collision is well avoided. It can, however, also be seen that occasionally a drastic mismatch between the (green) planned trajectory and the (black) realized trajectory is present. Besides vehicle parameter uncertainties, this can be explained with the use of an oversimplified model, the two-level controller setup, and uncertainty in the road-friction coefficient.

7.3.2 Simulation results

In an attempt to overcome these issues, we address the use of a higher-fidelity one-level controller setup in the following. We put particular emphasis on efficient numerical methods to render the accompanying computational challenge feasible.

To this end, we employ the RTI scheme (cf. Section 2.3.1) implementation of the ACADO Code Generation tool, which has already been introduced in Section 6.3. The ACADO Code Generation tool makes use of symbolic differentiation for generating plain C-code for all function and derivative evaluations. All problem

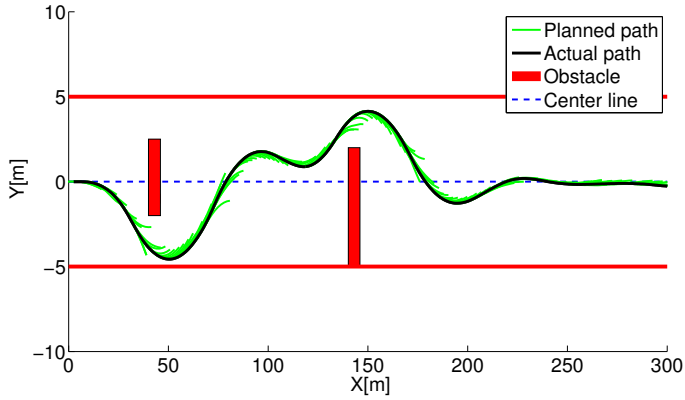


Figure 7.4: Experimental results (planned and realized path) of the test vehicle avoiding two obstacles. Figure taken from [GGF+12].

dimensions are hard coded based on static memory allocation. Moreover, inner loops of linear algebra operations are partially unrolled for increased performance without a significant increase in memory consumption. We make use of a tailored constant stepsize Gauss-Legendre integration method of order 2 (as introduced in [Qui12]) for the solution of the ODE system and its associated variational differential equations. We use a condensing/qpOASES-based QP solution strategy for solution of the MPC problem on a prediction horizon of length 20.

The considered scenarios resemble the ones from the experimental results from [GGF+12]. In particular, we simulate a slippery (e.g., snow-covered or icy) road surface by using a friction coefficient of $\mu = 0.3$. We also use the same set of parameters of a Jaguar X-Type reported in [GGF+12, GZF12]. Obstacles of each 6 m length are positioned at $s = 43$ m and $s = 123$ m (cf. Figure 7.5) on a 200 m long straight track. The first obstacle has a width of 2 m and needs to be avoided on the left, while the second obstacle of width 0.8 m needs to be avoided on the right.

The vehicle is traveling at 10 m/s, tracking the initial speed and the road reference while avoiding the obstacles. The total length of the prediction horizon is 20 m, with a discretization interval of 1 m. We initially employ the detailed vehicle model from Section 7.1 with wheel dynamics, load transfer (assuming a rigid suspension) and direct engine torque actuation.

	Feedback time	Full Iteration
Setup [GGF+12] + $\mathcal{M}_6$ average	156.9 ms	156.9 ms
ACADO + $\mathcal{M}_6$ average	0.06 ms	5.03 ms
ACADO + $\mathcal{M}_{12}$ average	3.1 ms	27.1 ms
ACADO + $\mathcal{M}_{12}$ maximum	6.5 ms	33.4 ms

Table 7.1: Computation times of one MPC step.

We chose diagonal weighting matrices

$$Q = \mathrm{diag}(1, 1, 10, 10, 10, 10^{-8}, 10^{-8}, 0.1, 0.1, 0.1, 0.1, 1, 1)$$

and

$$R = \mathrm{diag}(10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 1),$$

where states and controls are arranged in the order from Section 7.2.2. Integration of the dynamic system and the corresponding variational differential equations is based on a grid of 40 integrator steps in total. Initially, we assume full state observation, as well as knowledge of the road friction coefficient μ. Computational results were obtained on a PC featuring an Intel i7 mobile CPU at 2.7 GHz under Ubuntu 12.04. The generated C code has been compiled in a MEX function and all simulations were run in Matlab R2011b.

Figures 7.5 and 7.6 display the most important states and the control inputs for the considered scenario. Dashed vertical lines indicate when the obstacle becomes visible to the controller. The proposed control scheme avoids both obstacles and regains its initial speed after passing the second obstacle.

Table 7.1 shows average and worst-case computation times from this scenario. We denote the detailed model including wheel dynamics and load transfer by $\mathcal{M}_{12}$. Note that the obtained worst-case computation times are still real-time feasible for a 50 ms actuation system as it was used in [GGF+12], while the feedback delay is even one order of magnitude faster. For comparison, we also include average computation times obtained using the model used for path planning in [GGF+12], here denoted by $\mathcal{M}_6$. Using the proposed NMPC scheme these are significantly faster as the model is non-stiff and less complex. We also include computation times using the path planner in the setup from [GGF+12] to give the reader a rough idea of the potential performance gain by using tailored solution algorithms, even though an exact comparison is virtually

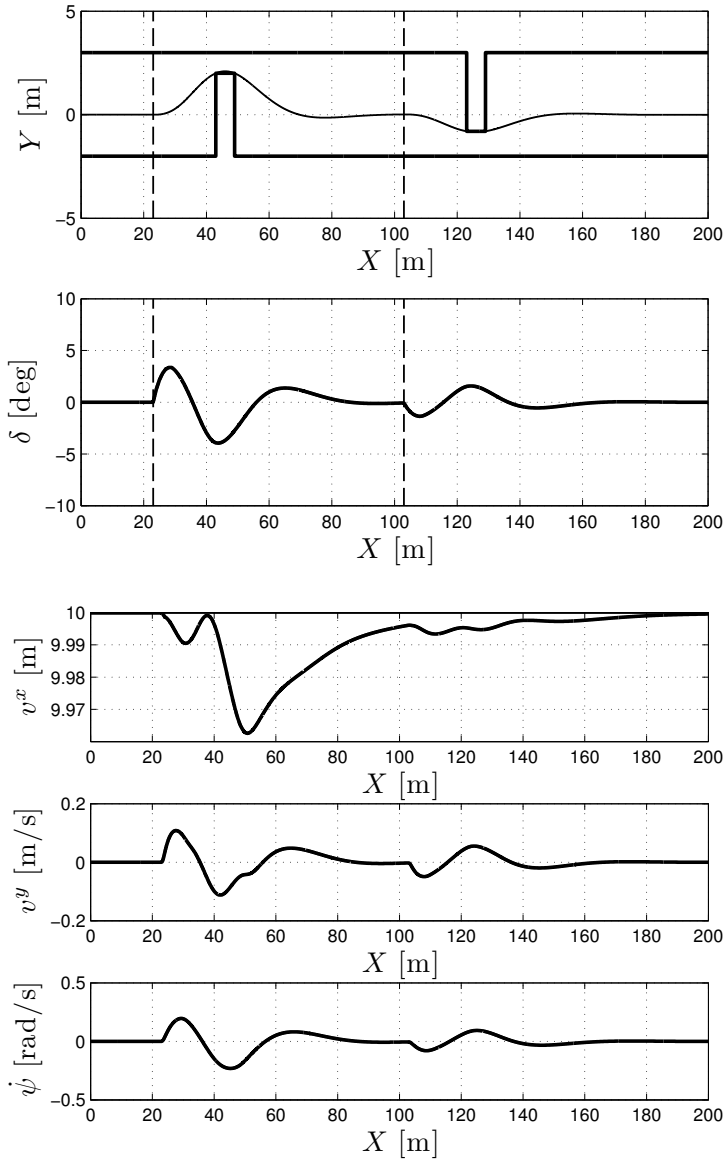


Figure 7.5: State trajectories from a simulated obstacle avoidance scenario using the high fidelity model from Sections 7.1 and 7.2.

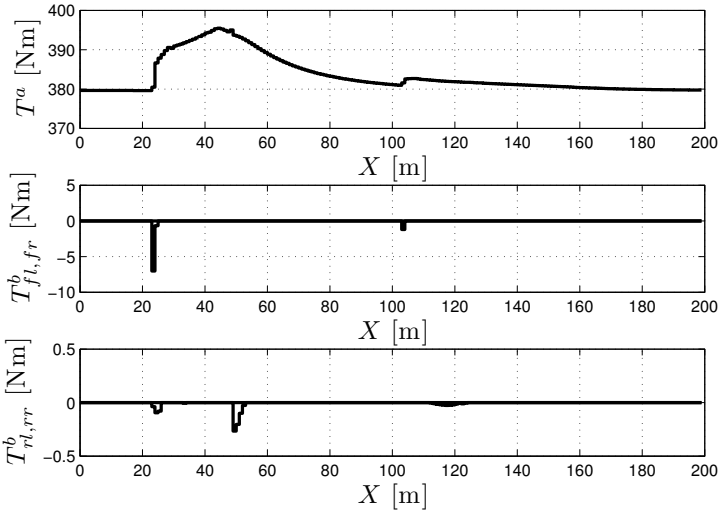


Figure 7.6: Control inputs from a simulated obstacle avoidance scenario using the high fidelity model from Sections 7.1 and 7.2.

Sensor	Measurements	Standard deviation σ
IMU	Linear acceleration	10^{-2} m/s^2
IMU	Angular velocity	0.1 rad/s
GPS	Position	10^{-2} m
Force sensor	Vertical forces	$5 \cdot 10^2$ N
Encoder	Wheel rotational velocity	10^{-3} rad/s
Encoder	Steering angle	10^{-3} rad

Table 7.2: Standard deviations of available measurements. Taken from [ZFV+14].

impossible, as the proposed approaches differ in central algorithmic components like the used integration method and the number of SQP iterations performed per MPC step.

Based on the vehicle model from Section 7.1, we can also derive a corresponding MHE scheme for real-time state and friction coefficient estimation, cf. [ZFD13, ZFV+14]. We assume to this end that measurements from an inertial measurement unit (IMU), a GPS, force sensors on the suspensions, and encoders on the wheels and the steering wheel are available. Here, we assume that all sensor measurements are uncorrelated. The corresponding sensor uncertainty,

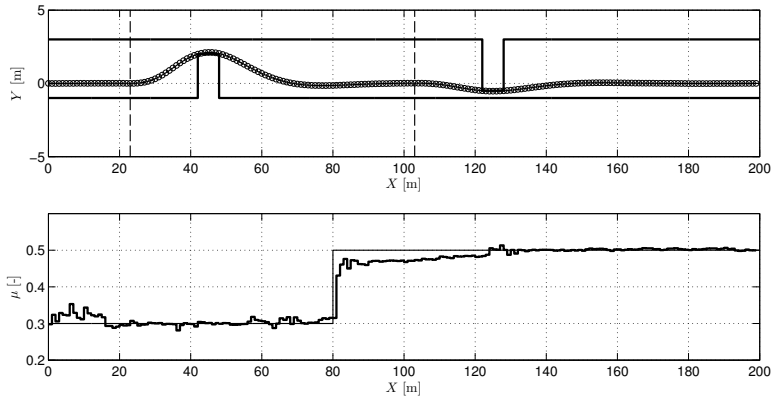


Figure 7.7: Vehicle position (estimated drawn by circles, actual drawn solid) and road friction coefficient (estimated drawn thick, actual drawn thin) for an obstacle avoidance scenario. Taken from [ZFV+14].

which we use for MHE tuning (cf. Section 2.2.3), can be read from Table 7.2. EKF updates to the arrival cost were used, cf. Section 2.2.2.

As for estimation of the road friction coefficient μ, we need to permit sudden changes due to changing ground conditions. One way to do so is to introduce a first order model for the friction coefficient, thus making it time varying. We propose to penalize $\dot{\mu}$ by a Huber penalty (cf. Section 2.2.1), which filters out small amounts of noise due to its "local" ℓ^2 characteristic, but allows for a fast detection of jumps in μ due to its "global" ℓ^1 characteristic.

In the following, we show simulation results for an obstacle avoidance scenario with noisy and incomplete state observation. The estimation horizon of the MHE scheme has been chosen as 10 m, divided into $N = 10$ stages of uniform duration length 1 m. The road friction coefficient is unknown to the vehicle and is taken to be $\mu = 0.3$ on the first part of the track. After 80 m, the friction coefficient increases to $\mu = 0.5$. The vehicle model is identical to the previous setting, but now employs the suspension model from Section 7.1.6.

Figure 7.7 shows that the proposed control/estimation scheme is able to safely avoid a collision also in the presence of incomplete state observation. The proposed Huber penalty permits to detect the jump in the road friction coefficient fast and reliably. Comparisons against an ℓ^2-penalty formulation confirmed the superiority of this approach in the considered scenario.

7.4 Time Optimal Driving

To tackle the problem of the time-optimal driving, we propose to aim at minimizing the time required for the vehicle to reach the end of the (fixed-length) spatial prediction horizon,

$$T = \int_{t_0}^{t_f} 1 \, \mathrm{d}t = \int_{s_0}^{s_f} \frac{1}{\dot{s}(\tau)} \, \mathrm{d}\tau \ . \tag{7.10}$$

Under the assumption that the prediction horizon is sufficiently long, we can expect this (finite-horizon) objective to approximately coincide with our goal of driving time-optimally.

For an efficient implementation, we further use an objective formulation based on the following observation.

Observation 7.1 Let all track data (i.e., road bounds, obstacles, ...) be fixed. Let $\mathcal{U}$ be the class of admissible control functions that satisfy the vehicle actuation constraints, and let $\mathcal{X} := \{\xi : \mathbb{R} \to \mathbb{R}^{n_x} \mid \xi'(s) = f(s, \xi(s), \nu(s)),\ e^y(s) \in [e_L^y(s), e_U^y(s)],\ \nu(s) \in \mathcal{U}\}$ be the set of all admissible trajectories. If $T^* := \arg\min_{\mathcal{X}} T$ exists, then for any $0 < T_{\mathrm{ref}} < T^*$, the global optimum of the optimization problem

$$\begin{aligned} \min_{\xi(\cdot),\nu(\cdot),T} \quad & \|T - T_{\mathrm{ref}}\|^2 \\ \text{s.t.} \quad & \xi'(s) = f(s, \xi(s), \nu(s)) \\ & e^y(s) \in [e_L^y(s), e_U^y(s)] \\ & \nu(s) \in \mathcal{U} \\ & \xi(0) = \xi_0, \end{aligned} \tag{7.11}$$

$(\hat{\xi}, \hat{u})$, satisfies $\hat{T} = T^*$. ⌟

By utilizing a sufficiently small (i.e., infeasible) "target time" T_{ref} we can therefore have an approximate time-optimal MPC formulation in least-squares form.

We briefly present experimental validation of the applicability of the proposed approach in the following. The tests were performed on a model race car setup at Siemens LMS using the ACADO Code Generation tool, and originally reported in [Ver14, VBZ+14]. Note that therein an oversimplified, slip-free vehicle model has been used due to the lack of vehicle parameters for the model car, featuring

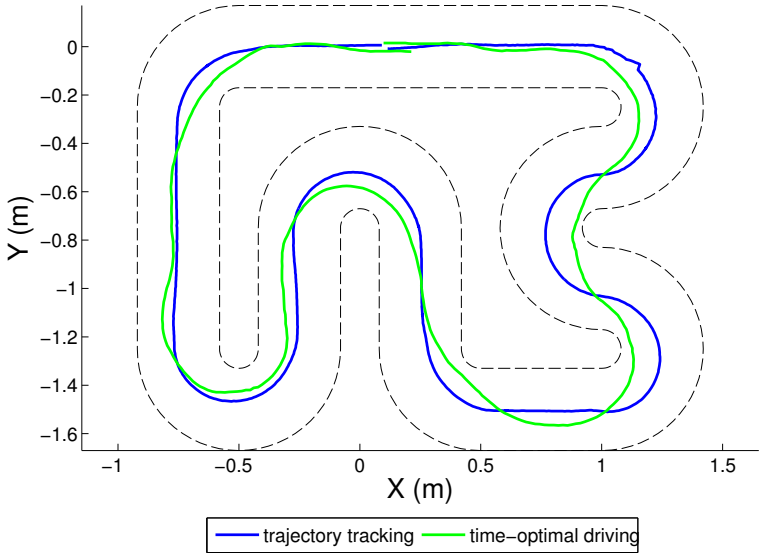


Figure 7.8: Performance comparison of trajectory tracking and time-optimal driving on the experimental setup. Artifacts in the trajectories are a result of noise in the vision system. Figure taken from [VBZ+14].

only position, orientation and velocity predictions; estimates for these states are obtained from an overhead camera-based infrared sensing system through a Kalman filter, cf. [Ver14].

We compare the time-optimal MPC formulation with an MPC formulation tracking the centerline at a reference speed of $v_{\text{ref}} = 1.0\,\text{m/s}$ in Figure 7.8. This reference speed was experimentally verified to be the largest speed at which the tracking performance was still satisfactory. For higher reference velocities, e.g., $v_{\text{ref}} = 1.2\,\text{m/s}$, the model-plant mismatch arising from the slip-free model caused the vehicle to deviate significantly from its spatial reference curve and resulted in unsafe driving behavior (touching track boundaries), cf. [VBZ+14]. In the time-optimal MPC formulation we chose a target time of $T_{\text{ref}} = 0.24\,\text{s}$ (which certainly is infeasible, since the model cars' velocity is limited to $4.0\,\text{m/s}$) for a prediction horizon of length $1.0\,\text{m}$. For numerical reasons we additionally regularized all other states and controls.

The lap time of the tracking scheme is 9.11 s, while the one of the time-optimal approach is 8.21 s. Both times refer to periodic trajectories, which are obtained after an initial transient of several rounds. The observed trajectories indeed exhibit typical time-optimal driving behavior, such as cutting corners, and

therefore indicate suitability of the proposed time-optimal MPC formulation.

Directions of future research include the use of higher-detail dynamic models for the model car setup, as well as a more thorough theoretical investigation of the Gauss-Newton tracking approach for time optimal driving and the use of qpDUNES to render longer prediction horizons real-time feasible.

Chapter 8

Conclusions

In this thesis, we have developed a variety of structure-exploiting concepts for dynamic optimization problems. They result in more efficient algorithms that can tackle problems on long prediction or estimation horizons, both on current and on future-generation computational architectures.

For linear problems (e.g., linear MPC, or linear regression with constraints) we have seen the dual Newton strategy from Chapter 4 to be an effective tool with the potential to outperform competing state-of-the-art solvers. We have analyzed several theoretical aspects that underline the observed performance, like the guaranteed active-set changes in each iteration or the warmstarting capabilities of the inverted factorization. Moreover, we have designed the algorithm in a largely concurrent fashion, which even triggered developments in other classes of methods, like interior-point QP solvers (cf. [Le14]) or NLP solvers (cf. [HFD14]). The software implementation of the basic dual Newton algorithm, qpDUNES, has already shown to be highly competitive in a variety of applications and benchmark problems and is freely available for download and unrestricted usage at [QPD14]. For band-structured linear-quadratic problems of shorter horizons, we have presented an improved, more efficient condensing algorithm that already made its way into the open-source ACADO Code Generation tool [ACA13].

For nonlinear problems we have presented a flexible algorithmic concept based on the Real-Time Iteration scheme in Section 2.4 that allows to gradually tune a controller between linear MPC and nonlinear MPC. Partial re-linearizations permit to treat problems of long prediction horizons in a computationally efficient manner, using accurate linearization information on proximate stages and less frequently updated information on farther stages. The warmstarted Newton system factorization within the dual Newton strategy has the potential

to benefit from this less frequently changing information also on the linear MPC/QP level. We have further seen an extension of the dual decomposition idea from QP to NLP in Section 5.2 in form of the ALADIN algorithm. This concept has the potential to fuse decomposition approaches from the solution of convex problems in dynamic optimization with the inherently bi-level approach of SQP from the treatment of nonconvex, nonlinear dynamic optimization problems, and may lead to a new kind of algorithm design in nonlinear MPC and MHE.

In the area of MPC applications, we have demonstrated that the usage of high-fidelity predictive models is computationally feasible when appropriate numerical tools are used that exploit the very much present structure of the problem, reducing the computational demand by orders of magnitude and thus advancing the applicability frontier of optimization-based control and estimation schemes.

In addition, this thesis raised several open questions and ideas for directions of future research. Among the short-term goals are the integration of some of the extensions of the dual Newton strategy developed in this thesis into the open-source software qpDUNES, such as preconditioned gradient steps or a more holistic integration of the stage problem solvers for even higher efficiency. Furthermore, in the medium term, extended research on the nonconvex dual Newton concept is desirable. In the author's opinion, the deeper integration of linearization routines and subproblem solution algorithms *within each stage* has the potential to lead to highly efficient algorithms for nonlinear band-structured optimization problems, such as nonlinear MPC or MHE. In a parallel computation architecture, each computational node could then be assigned one or several problem stages, keeping all stage problem data locally. The stage subproblem setup and solution can be performed entirely concurrently and without the need for a centralized coordination, thus largely reducing latencies from synchronization or data exchange. Only few data blocks need to be exchanged during the solution of the dual consensus problem when an algorithm in the style of the cyclic reduction scheme from Algorithm 4.3 is employed. This way, the algorithm design could even bridge the gap to distributed control and optimization, thus potentially enlarging the range of treatable problems vastly.

The idea of hierarchical QP data updates from Section 2.4 could even be carried one step further in the context of economic MPC, by introducing an additional layer above a tracking MPC scheme with exact and inexact linearization information, which provides updated references based on ultimate economic objectives to the tracking MPC scheme. Derivative information obtained during the economic optimization (for example) could be reused in the tracking problem for efficiency. This way, both, the desire for fast feedback (e.g., to counteract disturbances), and the desire for optimal operation with respect

to an intrinsic objective, can be realized within a single controller. Furthermore, a desirable goal for the hierarchical QP data update schemes would be to derive automatic criteria, based on which the re-computation of linearization information could be triggered, and to integrate MPC and MHE in an attempt to share common data (such as derivative information on stages that move from the prediction horizon into the estimation horizon, for example).

In the area of autonomous driving, open research directions include an in-depth analysis of implications of the application of algorithms like the generalized Gauss-Newton method to the problem of time-optimal driving. In particular, it is not clear yet how convergence to local minima (which are obviously undesirable in time-optimal driving) can be avoided rigorously in the used algorithmic setting. Above that, extended real-world experiments on a full-size vehicle test setup could help to identify further needs regarding the implementability of nonlinear MPC and MHE algorithms for driver assistant systems, and eventually for fully autonomous driving.

Appendix A

Mathematical Notation

This chapter gives a brief summary of the notation used throughout this thesis.

By $\mathbb{R}$ we denote the space of real numbers, by $\mathbb{R}_+$ the positive real numbers, by $\mathbb{R}_{0+}$ the non-negative real numbers, and by $\mathbb{N}$ the space of natural numbers including 0. A bar on a set generally denotes a closure, e.g., $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$.

We typically use calligraphic letters to denote specific sets, e.g., $\mathcal{S} := \{1, \ldots, N\}$, where $N \in \mathbb{N}$. We use subindices to exclude elements, i.e., we have $\mathcal{S}_N := \mathcal{S} \backslash \{N\}$ or $\mathcal{S}_1 := \mathcal{S} \backslash \{1\}$, for example.

A superscript $n \in \mathbb{N}$ attached to a set, as in $\mathbb{R}^n$, denotes the set of n-dimensional column vectors over this basis. Analogously $\mathbb{R}^{n \times m}$ denotes the set of $n \times m$ matrices. Generally, a in lowercase regular math font denotes a number (e.g., $a \in \mathbb{R}$), whereas $\boldsymbol{a}$ in lowercase boldface math font refers to a higher-dimensional vector (e.g., $\boldsymbol{a} \in \mathbb{R}^n$) and $\boldsymbol{A}$ in uppercase boldface math font denotes higher-dimensional matrices (e.g., $\boldsymbol{A} \in \mathbb{R}^{n \times n}$). Positive semidefiniteness of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A} \succeq \boldsymbol{0}$, and strict positive definiteness by $\boldsymbol{A} \succ \boldsymbol{0}$. A vector-valued relation symbol, e.g., $\boldsymbol{a} \geq \boldsymbol{0}$ is meant component-wise if not stated otherwise. A vector or matrix $\boldsymbol{a}^\top$ or $\boldsymbol{A}^\top$ denotes the respective transposed of $\boldsymbol{a}$ and $\boldsymbol{A}$. For improved readability, we occasionally use the sloppy form $(\boldsymbol{a_1}, \ldots, \boldsymbol{a_N}) := \left[\boldsymbol{a_1}^\top \quad \cdots \quad \boldsymbol{a_N}^\top \right]^\top$ to concatenate vectors (or to define a vector). The $\mathrm{diag}(\cdots)$ notation is used to define diagonal matrices, i.e.,

$$\mathrm{diag}(a_1, \ldots, a_n) := \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix},$$

and the canonical extension to block-diagonal matrices is given by

$$\operatorname{block\,diag}(\boldsymbol{A_1}, \ldots, \boldsymbol{A_N}) := \begin{bmatrix} \boldsymbol{A_1} & & \\ & \ddots & \\ & & \boldsymbol{A_N} \end{bmatrix}.$$

On occasions, we use calligraphic capital letters to denote block matrices, e.g., $\mathcal{H} := \begin{bmatrix} H_{1,1} & H_{1,2} \\ H_{2,1} & H_{2,2} \end{bmatrix}$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ and a set $\mathcal{A} \subseteq \{1, \ldots, m\}$, we use the notation $[\boldsymbol{A}_{\cdot,\boldsymbol{i}}]_{i \in \mathcal{A}}$ to denote the lumped matrix of the column vectors corresponding to the index set $\mathcal{A}$.

If not mentioned otherwise in the respective context, $\|\boldsymbol{a}\|$ denotes an appropriate norm of the respective vector space $\boldsymbol{a}$ lives in. By $\|\boldsymbol{a}\|_p$ we denote the ℓ^p norm (i.e., $\|\boldsymbol{a}\|_p := \left(\sum_{i=1}^N a_i^p\right)^{1/p}$), where $p \in \bar{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$, and by $\|\boldsymbol{a}\|_{\boldsymbol{A}}$, where $\boldsymbol{A} \succeq 0$ symmetric, we denote the norm induced by $\boldsymbol{A}$ via $\|\boldsymbol{a}\|_{\boldsymbol{A}} := \sqrt{\boldsymbol{a}^\top \boldsymbol{A} \boldsymbol{a}}$.

For a (block) matrix $\boldsymbol{\mathcal{X}} := \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B}^\top \\ \boldsymbol{B} & \boldsymbol{D} \end{bmatrix}$ we define the Schur complement of $\boldsymbol{D}$ in $\boldsymbol{\mathcal{X}}$ by $\boldsymbol{S} := \boldsymbol{A} - \boldsymbol{B}^\top \boldsymbol{D}^{-1} \boldsymbol{B}$, and the Schur complement of $\boldsymbol{A}$ in $\boldsymbol{\mathcal{X}}$ by $\boldsymbol{S'} := \boldsymbol{D} - \boldsymbol{B}\, \boldsymbol{A}^{-1} \boldsymbol{B}^\top$.

We use a semicolon to separate conditional arguments of a function $f : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{X}_3$ from the relevant principal arguments in the respective context, i.e., $f(a; b)$. This may be relevant in combination with derivatives, for example, when we are only interested in a derivative of f with respect to a. Furthermore, for functions $f : \mathcal{X}_1 \to \mathcal{X}_2$ and $g : \mathcal{X}_3 \to \mathcal{X}_2$ we occasionally use $f \circ g := f(g(\cdot))$ to express the concatenated application. In general, vector (or matrix) valued functions are denoted by boldface letters, e.g., $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^m$.

The derivative of an m-dimensional function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} := \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix},$$

or through the nabla-operator $\boldsymbol{\nabla_x f} := \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^\top$. When clear from the context, we also omit the subscript x on occasions.

Regarding the asymptotic growth of functions, we make use of the Landau symbols; in particular we use $f \in \mathcal{O}(g)$ to indicate that f does not grow substantially faster than g, i.e., $\limsup_{x \to \infty} \left| \frac{f(x)}{g(x)} \right| < \infty$.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex (on a subset $\mathcal{X} \subseteq \mathbb{R}^n$), if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and all $\alpha \in [0, 1]$ it holds:

$$f(\alpha\, \boldsymbol{x} + (1 - \alpha)\, \boldsymbol{y}) \leq \alpha\, f(\boldsymbol{x}) + (1 - \alpha)\, f(\boldsymbol{y}).$$

We call f strictly convex (on $\mathcal{X}$) if the inequality is strict. An alternative characterization of (strict) convexity for differentiable functions is given by

$$\left(\boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{y})\right)^\top (\boldsymbol{x} - \boldsymbol{y}) \left\{\begin{matrix}\geq\\>\end{matrix}\right\} 0 \qquad \forall\ \boldsymbol{x}, \boldsymbol{y}\ \in \mathcal{X}.$$

A function f is called (strictly) concave, if $-f$ is (strictly) convex.

We call a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ Lipschitz continuous, if there is a constant $L < \infty$ such that

$$\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\| \leq L \cdot \|\boldsymbol{x} - \boldsymbol{y}\|$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

By the linearization of a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ (in a point $\bar{\boldsymbol{x}}$), we usually refer to the first-order Taylor expansion of $\boldsymbol{f}$, given by $\bar{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{f}(\bar{\boldsymbol{x}}) + \boldsymbol{\nabla} \boldsymbol{f}(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}})$. We adopt a liberal understanding of linearization, and also refer to a QP approximation of an NLP as linearization.

For $\boldsymbol{m} \in \mathbb{R}^n$, we use $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ to denote the n-dimensional normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$.

Bibliography

[AÅD12] J. Andersson, J. Åkesson, and M. Diehl. CasADi – A symbolic package for automatic differentiation and optimal control. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, Lecture Notes in Computational Science and Engineering, Berlin, 2012. Springer. pages 85

[ABK+09] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Fast Nonlinear Model Predictive Control with an Application in Automotive Engineering. In L. Magni, D.M. Raimondo, and F. Allgöwer, editors, *Lecture Notes in Control and Information Sciences*, volume 384, pages 471–480. Springer Verlag Berlin Heidelberg, 2009. pages 86

[ACA13] ACADO Toolkit, 2009–2013. [Online; accessed 23-February-2009]. pages 189, 193, 233

[AD10a] J. Albersmeyer and M. Diehl. The Lifted Newton Method and its Application in Optimization. *SIAM Journal on Optimization*, 20(3):1655–1684, 2010. pages 21

[AD10b] J. Albersmeyer and M. Diehl. The Lifted Newton Method and its Application in Optimization. *SIAM Journal on Optimization*, 20(3):1655–1684, 2010. pages 98

[AFVD13] Joel A. E. Andersson, Janick V. Frasch, Milan Vukov, and Moritz Diehl. A Condensing Algorithm for Nonlinear MPC with a Quadratic Runtime in Horizon Length. *Automatica*, 2013. Submitted. pages 95, 98, 193, 200

[Alb10a] J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic*

systems. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2010. pages 22, 24, 59

[Alb10b] Jan Albersmeyer. *Adjoint-based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, Ruprecht-Karls-Universitität Heidelberg, 2010. pages 41, 42, 96, 97

[AM12] Daniel Axehill and Manfred Morari. An alternative use of the riccati recursion for efficient optimization. *Systems & Control Letters*, 61(1):37–40, 2012. pages 98, 116

[And13] Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013. pages 19, 42, 95, 97, 98

[ARA11] R. Amrit, J. Rawlings, and D. Angeli. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control*, 35:178–186, 2011. pages 56

[BBM03] F. Borrelli, A. Bemporad, and M. Morari. A Geometric Algorithm for Multi-Parametric Linear Programming. *Journal of Optimization Theory and Applications*, 118(3):515–540, 2003. pages 64

[BBM14] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control*. (in preparation), 2014. pages 64

[BDKS07] H.G. Bock, M. Diehl, E.A. Kostina, and J.P. Schlöder. Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*, pages 3–22. SIAM, 2007. pages 5, 74, 77, 79, 80, 81, 82, 86

[Bei12] D. Beigel. *Efficient goal-oriented global error estimation for BDF-type methods using discrete adjoints*. PhD thesis, Universität Heidelberg, 2012. pages 24

[Bel57] R. Bellman. *Dynamic programming*. Princeton University Press, 1957. pages 19

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 and 2. Athena Scientific, Belmont, MA, 1995. pages 52

[Ber99] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999. pages 122

[Ber07] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3rd edition, 2007. pages 20

[Bes96] M.J. Best. *Applied Mathematics and Parallel Computing*, chapter An Algorithm for the Solution of the Parametric Quadratic Programming Problem, pages 57–76. Physica-Verlag, Heidelberg, 1996. pages 48, 124

[BGFB94] S.P. Boyd, L.E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 14. SIAM, 1994. pages 43

[BH75] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Wiley, New York, 1975. pages 20

[Bie10] Lorenz T. Biegler. *Nonlinear Programming*. MOS-SIAM Series on Optimization. SIAM, 2010. pages 19, 29, 37, 95

[BKS07] H.G. Bock, E. Kostina, and J.P. Schlöder. Numerical methods for parameter estimation in nonlinear differential algebraic equations. *GAMM Mitteilungen*, 30/2:376–408, 2007. pages 39

[BM99] A. Bemporad and M. Morari. Robust model predictive control: A survey. In A. Garulli, A. Tesi, and A. Vicino, editors, *Robustness in Identification and Control*, number 245 in Lecture Notes in Control and Information Sciences, pages 207–226. Springer-Verlag, 1999. pages 43

[BMDP02] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002. pages 48, 64

[BNS95] L. T. Biegler, J. Nocedal, and C. Schmid. A reduced Hessian method for large-scale constrained optimization. *SIAM Journal of Optimization*, 5:314–347, 1995. pages 98

[Boc78] H.G. Bock. Numerical Solution of Nonlinear Multipoint Boundary Value Problems with Applications to Optimal Control. *Zeitschrift für Angewandte Mathematik und Mechanik*, 58:407, 1978. pages 20

[Boc81] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K.H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of

Springer Series in Chemical Physics, pages 102–125. Springer, Heidelberg, 1981. pages 97

[Boc83] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983. pages 39

[Boc87] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1987. pages 22, 37, 38, 39

[Bor03] F. Borrelli. *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Computer Science*. Springer, 2003. pages 64

[BP84] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 242–247. Pergamon Press, 1984. pages 21, 22, 94, 98

[BP12] Alberto Bemporad and Panagiotis Patrinos. Simple and certifiable quadratic programming algorithms for embedded linear model predictive control. In *4th IFAC Nonlinear Model Predictive Control Conference.*, pages 14–20, 2012. pages 117, 165

[BPCP11] S. Boyd, N. Parikh, E. Chu, and B. Peleato. Distributed optimization and statistics via alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. pages 179

[Bro70] C.G. Broyden. The convergence of a class of double rank minimization algorithms, part I and II. *J. Inst. Maths. Applns.*, 6:76–90 and 222–231, 1970. pages 37

[BRT97] A.B. Berkelaar, K. Roos, and T. Terkaly. *Recent Advances in Sensitivity Analysis and Parametric Programming*, chapter 6: The Optimal Set and Optimal Partition Approach to Linear and Quadratic Programming. Kluwer Publishers, Dordrecht, 1997. pages 120

[BT89] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989. pages 125

[BTEGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009. pages 52

[Bul71] R. Bulirsch. Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Technical report, Carl-Cranz-Gesellschaft, Oberpfaffenhofen, 1971. pages 22

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004. pages 30, 43, 61, 157

[CA98] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998. pages 184

[CB07] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2nd edition, 2007. pages 54

[CPDB13] Eric Chu, Neal Parikh, Alexander Domahidi, and Stephen Boyd. Code generation for embedded second-order cone programming. In *Control Conference (ECC), 2013 European*, pages 1547–1552. IEEE, 2013. pages 167

[CPMB12] A. Cortinovis, D. Pareschi, M. Mercangoez, and T. Besselmann. Model predictive anti-surge control of centrifugal compressors with variable-speed drives. In *Proceedings of the 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, Trondheim, Norway*, pages 251–256, 2012. pages 204

[CWTB00] A. M. Cervantes, A. Wächter, R. Tutuncu, and L. T. Biegler. A reduced space interior point strategy for optimization of differential algebraic systems. *Computers and Chemical Engineering*, 24:39–51, 2000. pages 98

[Dan67] John M. Danskin. *The theory of Max-Min and its application to weapons allocation problems*. Springer-Verlag, Berlin, New York, 1967. pages 126

[DBS+02] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002. pages 5, 64, 65

[DCB13] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *Control Conference (ECC), 2013 European*, pages 3071–3076. IEEE, 2013. pages 167

[DF06] Y.-H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to low and upper bounds. *Mathematical Programming*, 106(3):403–421, 2006. pages 117

[DFA+05] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, 2005. pages 69, 73, 74

[DFA07] M. Diehl, R. Findeisen, and F. Allgöwer. A Stabilizing Real-time Implementation of Nonlinear Model Predictive Control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*, pages 23–52. SIAM, 2007. pages 65, 69, 73, 74

[DFH09] M. Diehl, H. J. Ferreau, and N. Haverbeke. *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, chapter Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pages 391–417. Springer, 2009. pages 64, 69, 76, 98

[Die02] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002. Download also at: http://www.ub.uni-heidelberg.de/archiv/1659/. pages 14, 19, 22, 37, 39, 65, 67, 69, 70, 71, 72, 73, 166

[DLS01] M. Diehl, D.B. Leineweber, and A.A.S. Schäfer. MUSCOD-II Users' Manual. IWR-Preprint 2001-25, Universität Heidelberg, 2001. pages 24

[Dom13] A. Domahidi. *Methods and Tools for Embedded Optimization and Control*. PhD thesis, ETH Zürich, 2013. pages 44, 117

[DR83] I.S. Duff and J.K. Reid. The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations. *ACM Transactions on Mathematical Software*, 9(3):302–325, 1983. pages 174

[Duf04] I.S. Duff. MA57 — a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2):118–144, 2004. pages 174

[Duf06] I.S. Duff. Sparse system solution and the HSL library. Technical Report RAL-TR-2006-014, Rutherford Appleton Laboratory, September 2006. pages 174

[DZZ+12] A. Domahidi, A. Zgraggen, M.N. Zeilinger, M. Morari, and C.N. Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *IEEE Conference on Decision and Control (CDC)*, pages 668 – 674, Maui, HI, USA, December 2012. pages 43, 117, 167, 191, 195

[Eve63] Hugh Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963. pages 122

[FAC+05] C.A. Floudas, I.G. Akrotirianakis, S. Caratzoulas, C.A. Meyer, and J. Kallrath. Global optimization in the 21st century: Advances and challenges. *Computers and Chemical Engineering*, 29(6):1185–1202, 2005. pages 27

[FBA+07] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat. Predictive Active Steering Control for Autonomous Vehicle Systems. *Control Systems Technology, IEEE Transactions on*, 15(3):566–580, 2007. pages 207

[FBA+08] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat. Low complexity MPC schemes for integrated vehicle dynamics control problems. *9th International Symposium on Advanced Vehicle Control*, 2008. pages 207

[FBD08] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008. pages 48, 85, 116, 117, 124, 127, 136, 164, 193, 198, 200

[Fer06] H.J. Ferreau. An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control. Master's thesis, University of Heidelberg, 2006. pages 48, 116

[Fer11] H.J. Ferreau. *Model Predictive Control Algorithms for Applications with Millisecond Timescales*. PhD thesis, K.U. Leuven, 2011. pages 45, 48, 50

[FGZ+13] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An Auto-generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles. In *Proceedings of the European Control Conference*, 2013. pages 64, 193, 208, 209, 210, 219

[FHGD11a] Hans Joachim Ferreau, Boris Houska, Kurt Geebelen, and Moritz Diehl. Real-time control of a kite-model using an auto-generated nonlinear mpc algorithm. In *Proceedings of the 18th IFAC World Congress*, 2011. pages 200

[FHGD11b] H.J. Ferreau, B. Houska, K. Geebelen, and M. Diehl. Real-time control of a kite-carousel using an auto-generated nonlinear MPC algorithm. In *Proceedings of the IFAC World Congress*, 2011. pages 24, 64, 200

[Fia83] A.V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming.* Academic Press, New York, 1983. pages 61, 120, 121

[Fin97] P.K. Findeisen. Moving horizon estimation of discrete time systems. Master's thesis, University of Wisconsin-Madison, 1997. pages 62

[FJ13] G. Frison and J. Jorgensen. A Fast Condensing Method for Solution of Linear-Quadratic Control Problems. In *Proceedings of the 52nd IEEE Conference on Decision and Control*, 2013. pages 98, 116, 200

[FKD12] H.J. Ferreau, A. Kozma, and M. Diehl. A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, Noordwijkerhout, The Netherlands*, 2012. pages 117, 120, 121, 122, 125, 127

[FKP+13] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2013. (under review). pages 85, 117, 124, 127, 200

[FKSD12] J. V. Frasch, T. Kraus, W. Saeys, and M. Diehl. Moving Horizon Observation for Autonomous Operation of Agricultural Vehicles. In *Proceedings of the European Control Conference (ECC)*, 2012. pages 193

[FKV+12] H.J. Ferreau, T. Kraus, M. Vukov, W. Saeys, and M. Diehl. High-speed moving horizon estimation based on automatic code generation. In *Proceedings of the 51th IEEE Conference on Decision and Control (CDC 2012)*, 2012. pages 24, 64, 167, 193

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *Computer J.*, 13:317–322, 1970. pages 37

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, 2nd edition, 1987. pages 36, 37, 39, 45, 47, 121

[FOL+07] H. J. Ferreau, P. Ortner, P. Langthaler, L. del Re, and M. Diehl. Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control*, 31(2):293–301, 2007. pages 200, 207

[FPEN09] G.F. Franklin, J.D. Powell, and A. Emami-Naeini. *Feedback control of dynamic systems*. Prentice Hall, 6 edition, 2009. pages 53

[Fre79] Jens Frehse. Existence of Optimal Controls I. *Operations Research Verfahren*, 31:213–225, 1979. pages 21

[FSD13] J. V. Frasch, S. Sager, and M. Diehl. A Parallel Quadratic Programming Method for Dynamic Optimization Problems. *Mathematical Programming Computations*, 2013. submitted. pages 115, 189

[FVFD13] J. V. Frasch, M. Vukov, H.J. Ferreau, and M. Diehl. A new Quadratic Programming Strategy for Efficient Sparsity Exploitation in SQP-based Nonlinear MPC and MHE. In *Proceedings of the 19th IFAC World Congress*, 2013. accepted for publication. pages 189, 193

[FWSB12] J.V. Frasch, L. Wirsching, S. Sager, and H.G. Bock. Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control*, 2012. pages 51, 75, 82, 86, 95, 98, 110

[GAGD12] S. Gros, H. Ahmad, K. Geebelen, and M. Diehl. In-flight estimation of the aerodynamic roll damping and trim angle for a tethered aircraft based on multiple-shooting. In *System Identification Conference*, 2012. pages 193

[GD13] S. Gros and M. Diehl. NMPC based on Huber Penalty Functions to Handle Large Deviations of Quadrature States. In *American Control Conference*, 2013. pages 61

[GDK+13] Pontus Giselsson, Minh Dang Doan, Tamas Keviczky, Bart De Schutter, and Anders Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829 – 833, 2013. pages 122

[Ger05] M. Gerdts. Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with

gear shift. *Optimal Control Applications and Methods*, 26:1–18, 2005. pages 86, 207, 209

[GGF+12] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J.K. Hedrick, and F. Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012. pages 207, 208, 209, 223, 224, 225, 226

[GGL+12] A. Gray, Y. Gao, T. Lin, J.K. Hedrick, E. Tseng, and F. Borrelli. Predictive control for agile semi-autonomous ground vehicles using motion primitves. *Proceedings American Control Conference*, 2012. pages 207

[GK08] M. Gerdts and M. Kunkel. A nonsmooth Newton's method for discretized optimal control problems with state and control constraints. *Journal of Industrial and Management Optimization*, 4:247–270, 2008. pages 137

[GLB+10] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME 2010 Dynamic Systems and Control Conference*, pages 265–272. American Society of Mechanical Engineers, 2010. pages 207

[GM75] R. Glowinski and A. Marrocco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires. *Rev. Française Automat. Informat. Recherche Opérationnelle, RAIRO Analyse Numérique*, 9(R-2):41–76, 1975. pages 179

[GM76] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17 – 40, 1976. pages 179

[GM78] P.E. Gill and W. Murray. Numerically Stable Methods for Quadratic Programming. *Mathematical Programming*, 14:349–372, 1978. pages 30

[GMSW84] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints. *ACM Transactions on Mathematical Software*, 10(3):282–298, 1984. pages 48

[GMSW01] P. Gill, W. Murray, M. Saunders, and M. Wright. User's guide for NPSOL 5.0: A fortran package for nonlinear programming. Technical report sol 86-6, Stanford University, 2001. pages 224

[GMW81] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press, London, 1981. pages 128

[GNU11] GNU Lesser General Public License. http://www.gnu.org/copyleft/lesser.html, 2007–2011. pages 5, 190

[Gol70] D. Goldfarb. A family of variable metric methods derived by variational means. *Maths. Comp.*, 17:739–764, 1970. pages 37

[GP11] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Springer, London, 2011. pages 53, 54, 55

[GQD12] S. Gros, R. Quirynen, and M. Diehl. Aircraft Control Based on Fast Nonlinear MPC & Multiple-shooting. In *Conference on Decision and Control*, 2012. pages 193

[GR10] P. Giselsson and A. Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7272–7277, 2010. pages 122

[Gri00] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000. pages 42

[Grü12] Lars Grüne. NMPC Without Terminal Constraints. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control 2012*, 2012. pages 53

[Grü13] L. Grüne. Economic receding horizon control without terminal constraints. *Automatica*, 49:725–734, 2013. pages 4

[GVD13] S. Gros, M. Vukov, and M. Diehl. A Real-time MHE and NMPC Scheme for the Control of Multi-Mega Watts Wind Turbines. In *Conference on Decision and Control*, 2013. pages 193

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives*. SIAM, 2 edition, 2008. pages 42

[GZD12] S. Gros, M. Zanon, and M. Diehl. Orbit Control for a Power Generating Airfoil Based on Nonlinear MPC. In *American Control Conference*, 2012. (submitted). pages 193

[GZF12] A. Gray, M. Zanon, and J. Frasch. Parameters for a Jaguar X-Type. http://www.mathopt.de/RESEARCH/obstacleAvoidance.php, 2012. pages 212, 223, 225

[HBG+05] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, September 2005. pages 85

[Hes69] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. pages 182

[HF11] B. Houska and H.J. Ferreau. ACADO Toolkit User's Manual. http://www.acadotoolkit.org, 2009–2011. pages 24

[HFD11a] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. pages 24

[HFD11b] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011. pages 24, 64, 167, 189, 191, 193, 200

[HFD14] B. Houska, J. V. Frasch, and M. Diehl. An augmented lagrangian based algorithm for distributed non-convex optimization. *SIAM Journal on Optimization (under review)*, 2014. pages 169, 177, 179, 182, 183, 184, 233

[HNW96] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 1996. pages 96

[Hou07] B. Houska. Robustness and Stability Optimization of Open-Loop Controlled Power Generating Kites. Master's thesis, University of Heidelberg, 2007. pages 19

[Hou11] B. Houska. *Robust Optimization of Dynamic Systems*. PhD thesis, Katholieke Universiteit Leuven, 2011. (ISBN: 978-94-6018-394-2). pages 19, 52

[HS14] T. Huschto and S. Sager. Solving Stochastic Optimal Control Problems by a Wiener Chaos Approach. *Vietnam Journal of Mathematics*, 42(1):83–113, 2014. pages 14

[Hub81] P.J. Huber. *Robust Statistics*. Wiley Series in Probability and Mathematical Statistics. Wiley, New York, 1981. pages 60

[IBM09] IBM Corp. *IBM ILOG CPLEX V12.1, User's Manual for CPLEX*, 2009. pages 195

[JKS13] M. Jung, C. Kirches, and S. Sager. On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization – Festschrift for Martin Grötschel*, pages 387–417. Springer Berlin Heidelberg, 2013. pages 214

[KBM96] M. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, November 1996. pages 43

[KBSS11] C. Kirches, H.G. Bock, J.P. Schlöder, and S. Sager. Block Structured Quadratic Programming for the Direct Multiple Shooting Method for Optimal Control. *Optimization Methods and Software*, 26(2):239–257, April 2011. pages 116

[KCD13] A. Kozma, C. Conte, and M. Diehl. Benchmarking large scale distributed convex quadratic programming algorithms. *Optimization Methods & Software*, 2013. pages 45, 176

[KCM05] A.S. Kapadia, W. Chan, and L.A. Moyé. *Mathematical Statistics With Applications*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, 2005. pages 60

[KDK+11a] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & Chemical Engineering*, 35(1):71–83, 2011. pages 62, 105

[KDK+11b] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & Chemical Engineering*, 35(1):71–83, 2011. pages 62, 63

[Keh10] F. Kehrle. Optimal Control of Vehicles in Driving Simulators. Diploma thesis, Ruprecht–Karls–Unversität Heidelberg, March 2010. pages 209, 213, 214

[KF11] Markus Kögel and Rolf Findeisen. A fast gradient method for embedded linear predictive control. In *Proceedings of the 18th IFAC world congress*, pages 1362–1367, 2011. pages 165

[KF13] Markus Kogel and Rolf Findeisen. On efficient predictive control of linear systems subject to quadratic constraints using condensed, structure-exploiting interior point methods. In *Control Conference (ECC), 2013 European*, pages 27–34. IEEE, 2013. pages 167

[KFD13] A. Kozma, J. V. Frasch, and M. Diehl. A Distributed Method for Convex Quadratic Programming Problems Arising in Optimal Control of Distributed Systems. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, 2013. pages 169, 174, 175, 176

[KFKS11] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager. Optimal control of formula 1 race cars in a VDrift based virtual environment. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proceedings of the 18th IFAC World Congress*, pages 11907–11912, 2011. pages 207, 209, 212, 213, 214

[Kir11] C. Kirches. *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*. Advances in Numerical Mathematics. Springer Vieweg, Wiesbaden, July 2011. pages 14, 21, 22, 25, 26, 35, 65, 67, 116, 117

[KKGD14] A. Kozma, E. Klintberg, S. Gros, and M. Diehl. An improved distributed dual Newton-CG method for convex quadratic programming problems. In *American Control Conference*, 2014. (submitted for publication). pages 162

[KKWB08] Tom Kraus, Peter Kühl, Leonard Wirsching, and Hans Georg Bock. State and Parameter Estimation on Moving Horizons with Application to the Tennessee Eastman Benchmark Process. *Journal of Robotics and Autonomous Systems*, 2008. pages 62

[KM72] V. Klee and G.J. Minty. How Good is the Simplex Algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972. pages 155

[KN05] U. Kiencke and L. Nielsen. *Automotive Control Systems*. Springer, 2005. pages 209

[Koz13] A. Kozma. *Distributed Optimization Methods for Large Scale Optimal Control*. Phd thesis, Arenberg Doctoral School, KU Leuven, 2013. pages 45

[KPBS13] C. Kirches, A. Potschka, H.G. Bock, and S. Sager. A Parametric Active Set Method for a Subclass of Quadratic Programs with Vanishing Constraints. *Pacific Journal of Optimization*, 9(2):275–299, 2013. pages 48

[Kra07] T. Kraus. Real-Time State And Parameter Estimation for NMPC-Based Feedback Control With Application To The Tennessee Eastman Benchmark Process. Master's thesis, University of Heidelberg, 2007. pages 62

[KSBS10] C. Kirches, S. Sager, H.G. Bock, and J.P. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 31(2):137–153, March/April 2010. pages 86, 207, 209, 214

[KWBS12] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons. *Journal of Process Control*, 22(3):540–550, 2012. pages 98, 111

[KWSB10] C. Kirches, L. Wirsching, S. Sager, and H.G. Bock. Efficient Numerics for Nonlinear Model Predictive Control. In M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels, editors, *Recent Advances in Optimization and its Applications in Engineering*, pages 339–359. Springer, 2010. ISBN 978-3-6421-2597-3. pages 77

[LBBS03] D.B. Leineweber, I. Bauer, H.G. Bock, and J.P. Schlöder. An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization. Part I: Theoretical Aspects. *Computers and Chemical Engineering*, 27:157–166, 2003. pages 22

[LBS+03] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II). *Computers and Chemical Engineering*, 27:157–174, 2003. pages 22, 98

[Le14] Do Duc Le. Parallelisierung von Innere-Punkte-Verfahren mittels Cyclic Reduction. Bachelor's thesis, OvGU Magdeburg, 2014. pages 117, 152, 233

[Lei95] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Master's thesis, Universität Heidelberg, 1995. pages 22, 24, 85, 96, 98

[Lei99] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999. pages 14, 19, 22, 24, 37, 85, 94, 96, 98

[Loc01] A. Locatelli. *Optimal control – an introduction*. Birkhäuser, Basel Boston Berlin, 2001. pages 20

[Löf03a] J. Löfberg. Approximations of closed-loop minimax MPC. In Proceedings of the 42nd IEEE Conference on Decision and Control, pp 1438–1442, 2003. pages 43

[Löf03b] J. Löfberg. *Minmax approaches to robust model predictive control*. PhD thesis, Linköpings Universitet, 2003. pages 43

[LS97] W. Li and J. Swetits. A new algorithm for solving strictly convex quadratic programs. *SIAM Journal of Optimization*, 7(3):595–619, 1997. pages 117, 122, 128, 137

[Lue79] D.G. Luenberger. *Introduction to Dynamic Systems*. Wiley, 1979. pages 11, 16

[LvdG93] J. G. Lewis and R. A. van de Geijn. Distributed memory matrix-vector multiplication and conjugate gradient algorithms. In *Supercomputing '93. Proceedings*, pages 484 – 492, nov. 1993. pages 176

[Mat06] The MathWorks. http://www.mathworks.com/, 2006. pages 195

[MB09] J. Mattingley and S. Boyd. *Convex Optimization in Signal Processing and Communications*, chapter Automatic Code Generation for Real-Time Convex Optimization. Cambridge University Press, 2009. pages 117, 191

[MB10] Jacob Mattingley and Stephen Boyd. Real-time convex optimization in signal processing. *IEEE Signal Processing Magazine*, 27(3):50–62, May 2010. pages 167

[McG00] L.K. McGovern. *Computational analysis of real-time convex optimization for control systems*. PhD thesis, Massachusetts Institute of Technology, 2000. pages 43

[Meh92] S. Mehrotra. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization*, 2(4):575–601, 1992. pages 152

[MG95] PE Moraal and JW Grizzle. Observer design for nonlinear systems with discrete-time measurements. *Automatic Control, IEEE Transactions on*, 40(3):395–404, 1995. pages 61

[Mif77] Robert Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977. pages 122

[MM95] H. Michalska and D. Q. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40(6):995–1006, 1995. pages 62

[MP96] Timothy Maly and Linda R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20(1–2):57–79, February 1996. pages 97

[MRL93] K.R. Muske, J.B. Rawlings, and J.H. Lee. Receding Horizon Recursive State Estimation. In *Proc. Amer. Contr. Conf.*, pages 900–904, San Francisco, 1993. pages 62

[MRZ62] David D. Morrison, James D. Riley, and John F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Communications of the ACM*, 5(12):613–614, 1962. pages 22

[MSA03] J.R.R.A. Martins, P. Sturdza, and J.J. Alonso. The Complex-Step Derivative Approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, September 2003. pages 41

[Mus95] K.R. Muske. *Linear Model Predictive Control of Chemical Processes*. PhD thesis, University of Texas, Austin, 1995. pages 62

[NAW98] J.C. Newman, W.K. Anderson, and D.L. Whitfield. Multidisciplinary Sensitivity Derivatives Using Complex Variables. Technical Report MSSU-COE-ERC-98-08, Mississippi State University, July 1998. pages 41

[NDS08] I. Necoara, D. Doan, and J.A.K. Suykens. Application of the proximal center decomposition method to distributed model predictive control. In *47th IEEE Conference on Decision and Control*, pages 2900 –2905, dec. 2008. pages 122

[Nes04] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004. pages 40, 45, 165

[NS08] I. Necoara and J.A.K. Suykens. Applications of a smoothing technique to decomposition in convex optimization. *IEEE Trans. Automatic control*, 53(11):2674–2679, 2008. pages 122

[NS09] I. Necoara and J.A.K. Suykens. Interior-point Lagrangian decomposition method for separable convex optimization. *J. Optim. Theory and Appl.*, 143(3):567–588, 2009. pages 122

[NW00] J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag, 2000. pages 127, 128

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006. pages 26, 28, 29, 30, 31, 33, 34, 35, 36, 37, 40, 44, 45, 46, 47, 48, 165, 179, 180

[OK02] T. Ohtsuka and A. Kodama. Automatic code generation system for nonlinear receding horizon control. *Transactions of the Society of Instrument and Control Engineers*, 38(7):617–623, 2002. pages 167

[OR70] J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, New York, 1970. pages 36

[Osb69] M.R. Osborne. On shooting methods for boundary value problems. *Journal of Mathematical Analysis and Applications*, 27:417–433, 1969. pages 20, 22

[Pac06] H. B. Pacejka. *Tyre and Vehicle Dynamics*. Elsevier, 2006. pages 212

[PB14] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *Automatic Control, IEEE Transactions on*, 59(1):18–33, Jan 2014. pages 40, 165

[PBGM62] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962. pages 20

[PBS09] A. Potschka, H.G. Bock, and J.P. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24(2):237–252, 2009. pages 24

[PJ87] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *Robotics and Automation, IEEE Journal of*, 3(2):115 –123, april 1987. pages 218

[PKBS10] A. Potschka, C. Kirches, H.G. Bock, and J.P. Schlöder. Reliable solution of convex quadratic programs with parametric active set methods. Technical report, Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, GERMANY, November 2010. pages 48

[PLCS06] L. Petzold, S. Li, Y. Cao, and R. Serban. Sensitivity analysis of differential-algebraic equations and partial differential equations. *Computers and Chemical Engineering*, 30:1553–1559, 2006. pages 24

[Pot06] Andreas Potschka. Handling Path Constraints in a Direct Multiple Shooting Method for Optimal Control Problems. Diplomarbeit, Ruprecht-Karls-Universität Heidelberg, 2006. pages 24

[Pot11a] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Universität Heidelberg, 2011. pages 21, 26, 38

[Pot11b] Andreas Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, University of Heidelberg, 2011. pages 14, 85

[Pow69] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, 1969. pages 182

[PRW07] G. Pannocchia, J.B. Rawlings, and S.J. Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43:852–860, 2007. pages 64

[QGD13] R. Quirynen, S. Gros, and M. Diehl. Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. In *Proceedings of the European Control Conference*, 2013. pages 24, 59

[QPD14] qpDUNES Website. http://mathopt.de/qpdunes, 2012–2014. pages 189, 190, 233

[QPO14] qpOASES. http://www.qpOASES.org, 2007–2014. pages 189, 190

[QS93] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58:353–367, 1993. pages 122, 137

[Qui12] R. Quirynen. Automatic code generation of Implicit Runge-Kutta integrators with continuous output for fast embedded optimization. Master's thesis, KU Leuven, 2012. pages 97, 225

[QVD12] R. Quirynen, M. Vukov, and M. Diehl. Auto Generation of Implicit Integrators for Embedded NMPC with Microsecond Sampling Times. In Mircea Lazar and Frank Allgöwer, editors, *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference*, 2012. pages 24, 193, 201

[Ran09] Anders Rantzer. Dynamic dual decomposition for distributed control. In *Proceedings of the 2009 American Control Conference*, pages 884–888, 2009. pages 122

[Ric12] S. Richter. *Computational complexity certification of gradient methods for real-time model predictive control.* PhD thesis, ETH Zürich, 2012. pages 40

[RJM09] S. Richter, C.N. Jones, and M. Morari. Real-time input-constrained MPC using fast gradient methods. In *Proceedings of the IEEE Conference on Decision and Control, Shanghai, China*, 2009. pages 117

[RL95] D.G. Robertson and J.H. Lee. A least squares formulation for state estimation. *Journal of Process Control*, 5(4):291–299, 1995. pages 62

[RLR96] D.G. Robertson, J.H. Lee, and J.B. Rawlings. A moving horizon-based approach for least-squares state estimation. *AIChE Journal*, 42:2209–2224, 1996. pages 62, 63

[RM09] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design.* Nob Hill, 2009. pages 15, 16, 17, 18, 53, 54, 55, 57, 58, 59, 62, 171

[RMJ11] S. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained MPC based on Lagrange Relaxation and the fast gradient method. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5223 –5229, 2011. pages 122, 165

[Rob74] S.M. Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming*, 7:1–16, 1974. pages 37

[Rob96] D. Robertson. *Development and Statistical Interpretation of Tools for Nonlinear Estimation.* PhD thesis, Auburn University, 1996. pages 62, 63

[RRL01] C.V. Rao, J.B. Rawlings, and J.H. Lee. Constrained linear state estimation - a moving horizon approach. *Automatica*, 37(2):1619–1628, 2001. pages 62

[RWR98] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998. pages 117

[Sag05] S. Sager. *Numerical methods for mixed–integer optimal control problems.* Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. Available at http://sager1.de/sebastian/downloads/Sager2005.pdf. pages 14, 19, 21, 22, 116, 117

[SB08] J. Stoer and R. Bulirsch. *Introduction to numerical analysis.* Number 12 in Texts in applied mathematics. Springer, 2008. pages 22, 96

[SBS98] V.H. Schulz, H.G. Bock, and M.C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs. *SIAM Journal on Scientific Computing*, 19:440–467, 1998. pages 95

[Sha70] D.F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Maths. Comp.*, 24:647–656, 1970. pages 37

[She94] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994. pages 174

[SJ11] Dan Sui and Tor A Johansen. Moving horizon observer with regularisation for detectable systems without persistence of excitation. *International Journal of Control*, 84(6):1041–1054, 2011. pages 61

[SKB08] S. Sager, C. Kirches, and H.G. Bock. Fast solution of periodic optimal control problems in automobile test-driving with gear shifts. In *Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico*, pages 1563–1568, 2008. ISBN: 978-1-4244-3124-3. pages 207, 209, 214

[Son98] E.D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems.* Number 6 in Textbooks in Applied Mathematics. Springer-Verlag, New York, second edition edition, 1998. pages 12, 15, 16, 18

[SRB09] S. Sager, G. Reinelt, and H.G. Bock. Direct Methods With Maximal Lower Bound for Mixed-Integer Optimal Control Problems. *Mathematical Programming*, 118(1):109–149, 2009. pages 214

[SRKD11] C. Savorgnan, C. Romani, A. Kozma, and M. Diehl. Multiple shooting for distributed systems with applications in hydro electricity production. *Journal of Process Control*, 21:738–745, 2011. pages 171

[Ste94] R.F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994. pages 58

[TD12] Q. Tran-Dinh. *Sequential Convex Programming and Decomposition Approaches for Nonlinear Optimization*. Phd thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, November 2012. pages 35, 45

[TDNSD12] Q. Tran-Dinh, I. Necoara, C. Savorgnan, and M. Diehl. An Inexact Perturbed Path-Following Method for Lagrangian Decomposition in Large-Scale Separable Convex Optimization. *SIAM J. Optimization*, (under revision), 2012. pages 122

[VBZ+14] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards Time-Optimal Race Car Driving using Nonlinear MPC in Real-Time. In *Proceedings of the 53rd Conference on Decision and Control (CDC)*, 2014. pages 208, 209, 230, 231

[VDF+13] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl. Auto-generated Algorithms for Nonlinear Model Predicitive Control on Long and on Short Horizons. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, 2013. pages 98, 195, 198, 200

[Ven10] Joe Venzon. VDrift Website. http://vdrift.net, March 2010. pages 207, 213

[Ver14] R. Verschueren. Design and Implementation of a Time-Optimal Controller for Model Race Cars. Master's thesis, KU Leuven, 2014. pages 230, 231

[VLH+12] M. Vukov, W. Van Loock, B. Houska, H.J. Ferreau, J. Swevers, and M. Diehl. Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In *The 2012 American Control Conference, Montreal, Canada.*, 2012. pages 24, 64, 193

[WB10] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010. pages 196, 197

[WBD06] L. Wirsching, H. G. Bock, and M. Diehl. Fast NMPC of a chain of masses connected by springs. In *Proceedings of the IEEE International Conference on Control Applications, Munich*, pages 591–596, 2006. pages 198, 200, 201

[Wri91] S. Wright. Partitioned dynamic programming for optimal control. *SIAM Journal on Optimization*, 1(4):620–642, 1991. pages 150

[Wri97] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997. pages 31, 44, 165

[WSPS12] X. Wang, J. Stoev, G. Pinte, and J. Swevers. Energy optimal point-to-point motion using Model Predictive Control. In *Proceedings ASME 2012 5th Annual Dynamic Systems and Control Conference*, 2012. pages 194

[Zaf90] E. Zafiriou. Robust model predictive Control of processes with hard constraints. *Computers & Chemical Engineering*, 14(4–5):359–371, 1990. pages 48, 120, 121

[ZB09] V. M. Zavala and L.T. Biegler. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45:86–93, 2009. pages 64

[ZFD13] M. Zanon, J. Frasch, and M. Diehl. Nonlinear Moving Horizon Estimation for Combined State and Friction Coefficient Estimation in Autonomous Driving. In *Proceedings of the European Control Conference*, 2013. pages 193, 208, 209, 210, 228

[ZFV+14] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. Model Predictive Control of Autonomous Vehicles. In *Proceedings of the Workshop on Optimization and Optimal Control of Automotive Systems*. 2014. pages 208, 209, 228, 229

[ZGD13] M. Zanon, S. Gros, and M. Diehl. Rotational Start-up of Tethered Airplanes Based on Nonlinear MPC and MHE. In *Proceedings of the European Control Conference*, 2013. pages 193

[ZGD14] M. Zanon, S. Gros, and M. Diehl. Indefinite Linear MPC and Approximated Economic MPC for Nonlinear Systems. *Journal of Process Control*, 24:1273–1281, 2014. pages 56

[Zha06] Fuzhen Zhang. *The Schur complement and its applications*, volume 4 of *Numerical Methods and Algorithms*. Springer, 2006. pages 157

[ZJRM09] M.N. Zeilinger, C.N. Jones, D.M. Raimondo, and M. Morari. Real-time mpc-stability through robust mpc design. In *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference (CDC/CCC)*, pages 3980–3986, 2009. pages 43

Curriculum Vitae

Janick Frasch
born September 11, 1986 in Bamberg, Germany.

2013 – 2014	PhD studies in Mathematics, University of Magdeburg, Germany
2012 – 2013	PhD studies in Electrical Engineering, KU Leuven, Belgium
2010 – 2012	PhD studies in Mathematics, University of Heidelberg, Germany
2011	PhD studies in Mechanical Engineering, University of California, Berkeley, USA
2009 – 2010	Visiting Researcher, NEC Innovation Laboratories, Nara, Japan
2005 – 2009	Undergraduate studies in Mathematics and Computer Science, TU Kaiserslautern, Germany (Dipl.-Math.)
2008	Undergraduate studies in Mathematics, National University of Singapore
1996 – 2005	Abitur degree (general university entrance qualification), Clavius-Gymnasium Bamberg, Germany

List of Publications

Journal Papers:

1. J. V. Frasch, S. Sager, M. Diehl. *A Parallel Quadratic Programming Method for Dynamic Optimization Problems.* Mathematical Programming Computations, 201x. (In review)

2. B. Houska, J. V. Frasch, M. Diehl. *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization.* SIAM Journal on Optimization, 201x. (In review)

3. J. A. E. Andersson, J. V. Frasch, M. Vukov, M. Diehl. *A Condensing Algorithm for Nonlinear MPC with a Quadratic Runtime in Horizon Length.* Automatica, 201x. (In review)

Conference Proceedings:

1. R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, M. Diehl. *Towards Time-Optimal Race Car Driving using Nonlinear MPC in Real-Time.* Proceedings of the 53rd Conference on Decision and Control (CDC), Los Angeles, USA, 2014. (Accepted for publication)

2. J. V. Frasch, M. Vukov, H. J. Ferreau, M. Diehl. *A New Quadratic Programming Strategy for Efficient Sparsity Exploitation in SQP-Based Nonlinear MPC and MHE.* Proceedings of the 19th IFAC World Congress, Cape Town, South Africa, 2014.

3. A. Kozma, J. V. Frasch, M. Diehl. *A Distributed Method for Convex Quadratic Programming Problems Arising in Optimal Control of Distributed Systems.* Proceedings of the 52nd Conference on Decision and Control, Florence, Italy, 2013.

4. J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, M. Diehl. *An Auto-Generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles.* Proceedings of the 12th European Control Conference, Zurich, Switzerland, 2013.

5. J. V. Frasch, T. Kraus, W. Saeys, M. Diehl. *Moving Horizon Observation for Autonomous Operation of Agricultural Vehicles.* Proceedings of the 12th European Control Conference, Zurich, Switzerland, 2013.

6. M. Zanon, J. V. Frasch, M. Diehl. *Nonlinear Moving Horizon Estimation for Combined State and Friction Coefficient Estimation in Autonomous Driving.* Proceedings of the 12th European Control Conference, Zurich, Switzerland, 2013.

7. J. V. Frasch, L. Wirsching, S. Sager, H.G. Bock. *Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control.* Proceedings of the 4th IFAC Conference on Nonlinear Model Predictive Control, Noordwijkerhout, The Netherlands, 2012.

8. Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, F. Borrelli. *Spatial Predictive Control for Agile Semi-Autonomous Ground Vehicles.* Proceedings of the 11th International Symposium on Advanced Vehicle Control, Seoul, Korea, 2012.

9. F. Kehrle, J. V. Frasch, C. Kirches, S. Sager. *Optimal Control of Formula 1 Race Cars in a VDrift based Virtual Environment.* Proceedings of the 18th IFAC World Congress, Milan, Italy, 2011.

Book chapter:

1. M. Zanon, J. V. Frasch, M. Vukov, S. Sager, M. Diehl. *Model Predictive Control of Autonomous Vehicles.* In: Optimization and Optimal Control of Automotive Systems. Springer, 2014.

Declaration of Honor

I hereby declare that I produced this thesis without prohibited assistance and that all sources of information that were used in producing this thesis, including my own publications, have been clearly marked and referenced.

In particular I have not willfully:

- Fabricated data or ignored or removed undesired results.
- Misused statistical methods with the aim of drawing other conclusions than those warranted by the available data.
- Plagiarized data or publications or presented them in a distorted way.

I know that violations of copyright may lead to injunction and damage claims from the author or prosecution by the law enforcement authorities.

This work has not previously been submitted as a doctoral thesis in the same or a similar form in Germany or in any other country. It hast not previously been published as a whole.

Magdeburg, September 30, 2014

Janick Frasch

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS CENTER
Kasteelpark Arenberg 10, bus 2446
B-3001 Heverlee
email@esat.kuleuven.be
http://www.esat.kuleuven.be