Linköping Studies in Science and Technology. Dissertations No. 2324

Real-Time Certified MPC

Reliable Active-Set QP Solvers

Daniel Arnström

LINKÖPING UNIVERSITY

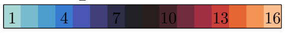
Linköping Studies in Science and Technology. Dissertations.
No. 2324

Real-Time Certified MPC

Reliable Active-Set QP Solvers

Daniel Arnström

LINKÖPING
UNIVERSITY

Cover illustration: Two-dimensional slice of the partition generated by the certification method in Paper A, applied to certify the computational complexity for the MPC of an AFTI-F16 aircraft. Each coordinate corresponds to a possible optimization problem that might need to be solved. Warmer colors correspond to the solver in Paper E requiring more iterations to produce a solution. Concretely, the number of iterations maps to a color as: 

. The code is available at `https://github.com/darnstrom/CertificationExamples`.

Linköping Studies in Science and Technology. Dissertations.
No. 2324

Real-Time Certified MPC: Reliable Active-Set QP Solvers

Daniel Arnström

daniel.arnstrom@liu.se
www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-8075-218-3 (print)
ISBN 978-91-8075-219-0 (PDF)
ISSN 0345-7524

Printed by LiU-Tryck, Linköping, Sweden 2023

Abstract

In Model Predictive Control (MPC), optimization problems are solved recurrently to produce control actions. When MPC is used in real time to control safety-critical systems, it is important to solve these optimization problems with guarantees on the worst-case execution time. In this thesis, we take aim at such worst-case guarantees through two complementary approaches:

(i) By developing methods that determine *exact* worst-case bounds on the computational complexity and execution time for deployed optimization solvers.

(ii) By developing efficient optimization solvers that are tailored for the given application and hardware at hand.

We focus on *linear* MPC, which means that the optimization problems in question are *quadratic programs* (QPs) that depend on parameters such as system states and reference signals. For solving such QPs, we consider *active-set methods*: a popular class of optimization algorithms used in real-time applications.

The first part of the thesis concerns complexity certification of well-established active-set methods. First, we propose a certification framework that determines the sequence of subproblems that a class of active-set algorithms needs to solve, for every possible QP instance that might arise from a given linear MPC problem (i.e., for every possible state and reference signal). By knowing these sequences, one can exactly bound the number of iterations and/or floating-point operations that are required to compute a solution. In a second contribution, we use this framework to determine the exact *worst-case execution time* (WCET) for linear MPC. This requires factors such as hardware and software implementation/compilation to be accounted for in the analysis. The framework is further extended in a third contribution by accounting for internal numerical errors in the solver that is certified. In a similar vein, a fourth contribution extends the framework to handle proximal-point iterations, which can be used to improve the numerical stability of QP solvers, furthering their reliability.

The second part of the thesis concerns efficient solvers for real-time MPC. We propose an efficient active-set solver that is contained in the above-mentioned complexity-certification framework. In addition to being real-time certifiable, we show that the solver is efficient, simple to implement, can easily be warm-started, and is numerically stable, all of which are important properties for a solver that is used in real-time MPC applications. As a final contribution, we use this solver to exemplify how the proposed complexity-certification framework developed in the first part can be used to tailor active-set solvers for a given linear MPC application. Specifically, we do this by constructing and certifying parameter-varying initializations of the solver.

Populärvetenskaplig sammanfattning

En populär reglerstrategi för att styra komplexa system är *modellprediktiv reglering* (MPC), där styrbeslut fattas genom att lösa matematiska optimeringsproblem. Detta kräver att datorprogram som kan lösa sådana problem, så kallade *optimeringslösare*, byggs in på systemen som ska styras. När MPC används för att styra snabba system kan tusentals optimeringsproblem behöva lösas varje sekund, vilket ställer höga krav på optimeringslösarna. Ett grundläggande krav är att optimeringsproblemen måste kunna lösas tillräckligt snabbt, annars blir de resulterande styrbesluten oförutsägbara. Att uppfylla detta krav är dock ofta utmanande i praktiken eftersom beräkningsresurser på inbyggda system är begränsade. Värstafallsgränser på tidsåtgång för optimeringslösare är således av stor vikt, speciellt när MPC används för att styra säkerhetskritiska system i realtid, där oförutsägbara styrbeslut kan få förödande, till och med livshotande, konsekvenser.

I den här avhandlingen presenteras ett ramverk som beräknar sådana värstafallsgränser. Specifikt undersöks *aktivmängdlösare*, en klass av optimeringslösare som ofta används för att lösa *kvadratiska optimeringsproblem*, vilket är problemen som måste lösas när MPC används för att styra system som kan modelleras med *linjära* modeller. Mer specifikt bestämmer ramverket vilken sekvens av delproblem (linjära ekvationssystem) som populära aktivmängdlösare kommer att behöva lösa, för alla möjliga kvadratiska optimeringsproblem som kan uppstå i en given MPC-applikation. Först bestäms *exakta* värstafallsgränser för antalet iterationer och/eller flyttalsberäkningar som aktivmängdlösare kräver. Ramverket används sedan för att ge exakta värstafallsgränser för *exekveringstid*. Slutligen utökas ramverket så att värstafallsgränserna tar hänsyn till numeriska problem som kan uppstå i aktivmängdlösare. Vi ger också exempel på hur ramverket kan användas för att skräddarsy aktivmängdlösare för en given MPC-applikation genom att konstruera och certifiera variabla starttillstånd.

På det hela taget kan ramverket som presenteras i avhandlingen användas för att erhålla garantier på maximal tidsåtgång för att beräkna ett styrbeslut med MPC, vilket, exempelvis, kan användas för att försäkra att hårdvaran som MPC-regulatorn är implementerad på inte överbelastas. Ramverket kan också användas för att göra aktivmängdlösare snabbare genom att skräddarsy dem till en specifik applikation och hårdvara, vilket möjliggör styrning av ännu snabbare system med MPC.

I avhandlingen presenteras också en ny aktivmängdlösare som ingår i det ovannämnda ramverket. Utöver garantier på värstafallsgränser är lösaren effektiv, enkel att implementera, och robust mot numeriska fel, vilket alla är viktiga egenskaper för en lösare som används för realtids-MPC.

Acknowledgments

First and foremost, I want to thank my supervisor Daniel Axehill for your guidance and support over the past five years. Your endless enthusiasm and kindness have kept me motivated throughout this journey. To my co-supervisor Anders Hansson, who was the lecturer in the first courses I took in automatic control and optimal control: thank you for inspiring me to take the first steps.

I also want to express my deepest appreciation to all colleagues at the Division of Automatic Control at Linköping University. Especially, I want to thank all fellow PhD students, past and present, for your camraderie and continuous injection of noise (the good kind!) over the past five years; without it, important modes in me would remain unexcited and important detours would remain unexplored. A special thanks goes to Anton Kullberg, Magnus Malmström, and Shamisa Shoja for their help in proofreading parts of this thesis. I also want to thank all senior staff of the group for sharing their wisdom about research and life. Special thanks to the head of division Martin Enqvist for cultivating a welcoming work environment and to Ninna Stensgård who is always ready to help with administrative tasks.

The research presented in this thesis was made possible by a grant from the Swedish Research Council (VR), which I am grateful for. I also want to thank Prof. Alberto Bemporad and Prof. David Broman for fruitful collaborations on research contained in this thesis.

Lots of progress on the research topics presented in this thesis was made during walks to and from work. In particular, many moments of insight came while passing by a herd of sheep populating a pasture close to campus during summers. For their contribution, I would have added them as co-authors to some of the papers, if only they would have told me their ORCIDs.

To my family: thank you for always supporting me in taking the road less traveled and for keeping me grounded.

Finally, I want to thank anyone who helped me become a better researcher and person during this five-year journey. The pages that follow contain mere by-products produced along the way.

Linköping, May 2023
Daniel Arnström

Contents

Notation

Notation	Meaning
$\mathbb{N}_m$	Set of positive integers up to m
$\mathbb{R}^{m\times n}$	Set of real $m \times n$ matrices
$\mathbb{S}^n_+$	Set of real $n \times n$ positive semi-definite matrices
$\mathbb{S}^n_{++}$	Set of real $n \times n$ positive definite matrices
$[M]_i$	The ith row of matrix M
$[M]_\mathcal{I}$	The rows of matrix $M \in \mathbb{R}^{m\times *}$ indexed by $\mathcal{I} \subseteq N_m$
$\{a_i\}_{i=1}^n$	A sequence of n elements $\left(\{a_i\}_{i=0}^n = \{a_1, a_2, \dots, a_n\}\right)$
$\mathbf{diag}(A, B, C)$	A block diagonal matrix with blocks A, B and C
$\mathbf{vec}(a, b, c)$	Vectors a, b and c stacked into a single vector

Abbreviations

Abbreviation	Meaning
EQP	Equality-constrained quadratic program
LP	Linear program(ming)
MPC	Model Predictive Control
mpLP	Multi-parameteric linear program(ming)
mpQP	Multi-parameteric quadratic program(ming)
QP	Quadratic program(ming)
WCET	Worst-case execution time

Part I

Background

1 Introduction

1.1 Background and motivation

At present, Model Predictive Control (MPC) is the go-to strategy for controlling complex systems [1]. It is also one of the control technologies that industry believes will have the most future impact [2]. Reasons for its success are (i) its interpretability; (ii) its ability to control nonlinear and multi-variable systems; (iii) that it can directly account for constraints on control actions and system states.

As its name suggests, MPC uses a model of the controlled system to predict its future states (given the applied control actions and the starting state). Through this model, the control problem can be posed as an optimization problem, in which a sequence of control actions should be selected to produce an "optimal" sequence of future states. To counteract model errors and external disturbances acting on the system, a new control action is recomputed every time the state of the system changes (visualized in Figure 1.1).

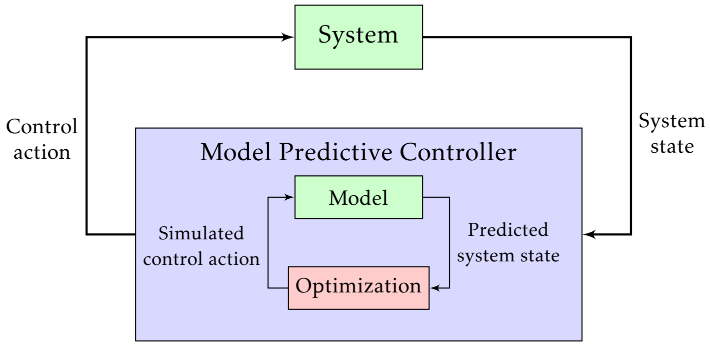


Figure 1.1: *Schematic overview of Model Predictive Control.*

Since an optimization problem has to be solved every time the system's state changes, MPC first saw success in the process industry [3], where the relatively slowly changing states of the plant gave enough time to solve the optimization problems with the software and hardware available at the time. Since then, improvements in software and hardware have enabled MPC to be applied to systems where the state is changing more rapidly, for example, in automotive [4; 5], aerospace [4; 6], and power systems [7; 8] applications.

As MPC is applied to control faster systems, the optimization problems have to be solved more frequently. At the same time, controllers are increasingly implemented on simpler hardware such as microcontrollers and FPGAs [9; 10]. These two trends taken together require the optimization problems to be solved faster with limited computational resources. To meet these increasingly challenging requirements, the solvers that are used to solve the optimization problems need to become more efficient and computationally frugal [11].

Further adding to the challenge, the complexity of solving the optimization problems might vary significantly between each time instance, since the optimization problems to be solved are dependent on the current state of the system, which changes over time. It is therefore nontrivial to guarantee that the computational resources at hand are sufficient to solve all possible optimization problems fast enough; yet, such guarantees are crucial to know *before* the MPC controller is deployed, especially if the controlled system is *safety critical* [12].

The ultimate objective with the research presented in this thesis is to provide such *a priori* guarantees; that is, to provide guarantees that the employed optimization solver is able to solve *all possible* optimization problems within the limited time frame for the application and hardware at hand. We approach this objective from two complementary directions:

(i) By developing certification methods that provide worst-case bounds on the computational complexity and execution time for a given solver.

(ii) By developing solvers that efficiently solve the optimization problems, allowing the limited resources to be used more economically.

The main focus of this thesis is on (i), where the optimization methods considered are *active-set methods* for *quadratic programming*. However, in addition to bounds on computational complexity, the proposed certification methods give precise information about the behavior of the optimization algorithm. These insights can therefore be used to tailor a solver for the specific set of problems at hand, which in turn can be used as a tool in developing more efficient solvers, i.e., to make strides in direction (ii).

Let us call MPC controllers that are based on solvers with the above-mentioned guarantees to be *real-time certified*. In short then, this thesis concerns tools that enable real-time certified MPC.

1.2 Thesis outline

The thesis is split into two parts: Part I gives background information about model predictive control (Chapter 2), quadratic programming (Chapter 3), and active-set methods (Chapter 3), and is based on the author's licentiate thesis [13]. Part I provides context to the publications that comprise Part II of the thesis, where the main contributions are presented. Below we summarize the publications included in Part II and the author's contribution to each. This introductory chapter then concludes with a literature review of related work.

Publications and the author's contribution

Paper A: A unifying complexity certification framework for active-set methods for convex quadratic programming

D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2758–2770, 2022.

Summary: We present a framework for determining the worst-case complexity of a family of well-established active-set methods. For every parameter of interest in a multi-parametric quadratic program, the method determines which sequence of subproblems active-set algorithms need to solve to find a solution. These sequences provide, in turn, worst-case bounds on the number of iterations and/or floating-point operations that are required to find a solution. Such bounds give worst-case complexity guarantees for linear MPC controllers, which is of importance when safety-critical systems are controlled in real time. The usefulness of the proposed method is illustrated on a set of multi-parametric quadratic programs originating from MPC problems. Specifically, we compute the exact worst-case number of iterations that primal and dual active-set algorithms require to compute a solution.

Background and the author's contribution: The overarching concept of tracking the working-set changes for the primal active-set method in [15] was conceived by DAx. The author of this thesis actualized and refined the idea considerably and did the majority of the work, including writing the manuscript, theoretical derivations and numerical experiments, resulting in the publication [16], which Paper A extends. The insight that the certification method developed in [16] could be extended to unify the results therein with the certification methods presented in [17] and [18] was conceived by the author of this thesis (inspired by discussions with DAx), resulting in Paper A. The majority of the work, including writing the manuscript, theoretical derivations and numerical experiments, was carried out by the author of this thesis. DAx reviewed the manuscript and helped refine the ideas through discussions.

Paper B: Exact worst-case execution-time analysis for implicit model predictive control

D. Arnström, D. Broman, and D. Axehill, "Exact worst-case execution-time analysis for implicit model predictive control," *arXiv preprint arXiv:2304.11576*, 2023, submitted.

Summary: We propose a method that determines the exact worst-case execution time (WCET) for implicit linear MPC. The method leverages the framework in Paper A to generate a finite set of "archetypal" optimization problems; we prove that these archetypal problems form an *execution-time equivalent cover* of all possible problems; that is, that they capture the execution time for solving any possible optimization problem that can be encountered online. Hence, by solving only these archetypal problems on the hardware on which the MPC is to be deployed, and by recording the execution times, we obtain the *exact* WCET. We validate the method on an MPC example where an inverted pendulum on a cart is stabilized. The experiments highlight the following advantages compared with classical WCET methods: (i) in contrast to classical static methods, our method gives the *exact* WCET; (ii) in contrast to classical measurement-based methods, our method guarantees a correct WCET estimate and requires fewer measurements on the hardware.

Background and the author's contribution: That the framework in Paper A generates more detailed information than just number of iterations and/or number of floating-point operations emerged through discussions between the author of this thesis and DAx. That it could, in fact, be used to determine the *exact* WCET was conceived by the author of this thesis, who wrote the manuscript, performed the experiments and made the theoretical derivations. DB provided expertise from the WCET field. Both DAx and DB reviewed the manuscript and helped refine the idea through discussions.

Paper C: Lift, partition, and project: Parametric complexity certification of active-set QP methods in the presence of numerical errors

D. Arnström and D. Axehill, "Lift, partition, and project: Parametric complexity certification of active-set QP methods in the presence of numerical errors," in *IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4381–4387.

Summary: A shortcoming with the framework in Paper A is that it does not account for numerical errors that might occur internally in the solvers that are certified. This might ultimately lead to optimistic complexity bounds if, for example, the solvers are implemented in single precision. In this paper we propose a general scheme that can be incorporated in the framework in Paper A (and in similar certification methods) to account for such numerical errors.

Background and the author's contribution: How numerical errors could be accounted for with the proposed lift-partition-project scheme was conceived by the author of this thesis, who also wrote the manuscript, performed the numerical experiments and made the theoretical derivations. DAx reviewed the manuscript.

Paper D: Complexity certification of proximal-point methods for numerically stable quadratic programming

D. Arnström, A. Bemporad, and D. Axehill, "Complexity certification of proximal-point methods for numerically stable quadratic programming," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1381–1386, 2021.

Summary: The numerical stability of any QP solver can be improved by performing proximal-point outer iterations, resulting in solving a sequence of better conditioned QPs. In this paper we present a method which, for a given multi-parametric quadratic program and any polyhedral set of parameters, determines which sequences of QPs have to be solved when using outer proximal-point iterations. By knowing these sequences, bounds on the worst-case complexity of the method can be obtained, which is of importance in real-time MPC. Moreover, we combine the proposed method with the framework in Paper A to obtain finer-grained complexity certificates of the proximal-point method, namely, the total number of inner iterations.

Background and the author's contribution: Through discussions during the collaboration on [22], AB familiarized the author of this thesis with the extensions made to [23] presented in [24], in which numerical stability was improved by performing proximal-point iterations. The idea of how the proximal-point iterations could be tracked parametrically was conceived by the author of this thesis, who made the majority of the work including writing the manuscript, numerical experiments and theoretical derivations. AB and DAx helped refine the idea and reviewed the manuscript.

Paper E: A dual active-set solver for embedded quadratic programming using recursive LDL^T updates

D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL^T updates," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4362–4369, 2022.

Summary: We present a dual active-set solver for quadratic programming that has properties suitable for use in embedded model predictive control applications. In particular, the solver is efficient, can easily be warm started, and is simple to code. Moreover, by using outer proximal-point iterations, as described in Paper D, ill-conditioned problems can be handled in a robust manner. Finally, since the solver is covered by the framework presented in Paper A, the exact worst-case computational complexity of the solver can be determined offline.

Background and the author's contribution: How the recursive LDL^T updates in [23] could be applied to, and extended to, dual QPs (and that this yields several favourable properties) was conceived by the author of this thesis. The author of this thesis coded the C-implementation of the solver, carried out the numerical experiments, made the theoretical derivations, and wrote the manuscript. AB and DAx reviewed the manuscript.

Paper F: Semi-explicit linear MPC using a warm-started active-set QP algorithm with exact complexity guarantees

D. Arnström and D. Axehill, "Semi-explicit linear MPC using a warm-started active-set QP algorithm with exact complexity guarantees," in *IEEE 60th Conference on Decision and Control (CDC)*, 2021, pp. 2557–2562.

Summary: We propose a semi-explicit approach for linear MPC in which a dual active-set quadratic programming algorithm is initialized through a pre-computed warm start. By using the framework from Paper A, we show how the computational complexity of the dual active-set algorithm can be determined offline for a given warm start. We also show how these complexity certificates can be used as quality measures when constructing warm starts, enabling the online complexity to be reduced further by iteratively refining the warm start. In addition to showing how the computational complexity of any pre-computed warm start can be determined, we propose a novel technique for generating warm starts with low overhead, both in terms of computations and memory.

Background and the author's contribution: The way of constructing suitable initializations for a given MPC problem was conceived by the author of this thesis, who also wrote the manuscript, performed the numerical experiments and made the theoretical derivations. DAx reviewed the manuscript.

Complete list of publications

For completeness, all publications that the author has contributed to as a PhD student are given below in chronological order (with respect to publication date). Publications included in Part II are marked with ⋆.

D. Arnström and D. Axehill, "Exact complexity certification of a standard primal active-set method for quadratic programming," in *IEEE 58th Conference on Decision and Control (CDC)*, Dec 2019, pp. 4317–4324.

D. Arnström and D. Axehill, "Exact complexity certification of a standard early-terminating primal active-set method for quadratic programming," in *Proceedings of the 2020 IFAC World Congress*, 2020.

D. Arnström, A. Bemporad, and D. Axehill, "Exact complexity certification of a nonnegative least-squares method for quadratic programming," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 1036–1041, 2020.

⋆ D. Arnström, A. Bemporad, and D. Axehill, "Complexity certification of proximal-point methods for numerically stable quadratic programming," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1381–1386, 2021.

D. Arnström, A. Bemporad, and D. Axehill, "A linear programming method based on proximal-point iterations with applications to multi-parametric programming," *IEEE Control Systems Letters*, vol. 6, pp. 2066–2071, 2021.

⋆ D. Arnström and D. Axehill, "Semi-explicit linear MPC using a warm-started active-set QP algorithm with exact complexity guarantees," in *IEEE 60th Conference on Decision and Control (CDC)*, 2021, pp. 2557–2562.

S. Shoja, D. Arnström, and D. Axehill, "Overall complexity certification of a standard branch and bound method for mixed-integer quadratic programming," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 4957–4964.

⋆ D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2758–2770, 2022.

⋆ D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL^T updates," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4362–4369, 2022.

⋆ D. Arnström and D. Axehill, "Lift, partition, and project: Parametric complexity certification of active-set QP methods in the presence of numerical errors," in *IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4381–4387.

S. Shoja, D. Arnström, and D. Axehill, "Exact complexity certification of a standard branch and bound method for mixed-integer linear programming," in *IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 6298–6305.

D. Arnström and D. Axehill, "BnB-DAQP: a mixed-integer QP solver for embedded applications," in *Proceedings of the 2023 IFAC World Congress*, 2023.

⋆ D. Arnström, D. Broman, and D. Axehill, "Exact worst-case execution-time analysis for implicit model predictive control," *arXiv preprint arXiv:2304.11576*, 2023, submitted.

1.3 Related work

Next, we review related work to the topics considered in this thesis. Specifically, we review work on real-time MPC (Section 1.3.1) and on complexity certification for optimization methods (Section 1.3.2).

1.3.1 Real-time MPC

As outlined in the introduction, MPC requires an optimization problem to be solved at each time step to produce a control action. There are several ways of solving such optimization problems; here we give a brief overview of some approaches. For textbooks on the topic, see [15; 32; 33]. Some additional tools for real-time MPC that are solver agnostic are to tailor (i) linear algebra routines [34]; (ii) the solver code [35; 36]; (iii) the degree of sparsity [37].

Linear MPC When the dynamics of the system to be controlled is linear (and time-invariant), the optimization problems that need to be solved in MPC can be cast as instances of a multi-parametric quadratic program of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f(\theta)^T x \\ \text{subject to} \quad & Ax \leq b(\theta), \end{aligned} \tag{1.1}$$

where the decision variable $x \in \mathbb{R}^n$ is related to the control actions, and the parameter $\theta \in \mathbb{R}^p$ is related to setpoints and the system state. How to go from a linear MPC problem to a problem of the form (1.1) is described in detail in Chapter 2. For a given linear MPC application, the positive semidefinite matrix $H \in \mathbb{S}^n_+$ and the constraint matrix $A \in \mathbb{R}^{m \times n}$ are *fixed*. Moreover, both the linear cost f and the constraint offset b are affine functions of θ. This enables a closed-form solution of (1.1), which is piecewise affine in θ over polyhedral regions [38]. Methods for computing the closed-form solution generally fall into two categories: geometrical [38–41] and combinatorial [42–45]; software packages that implement such methods include `MPT` [46], `POP` [47], and the `Hybrid Toolbox` [48].

The closed-form solution mentioned above is used in *explicit* MPC [49] by storing it as a lookup table and searching through this table online. A well-known drawback with explicit MPC is that the lookup table quickly becomes intractably large, restricting its use to small-scale problems. Techniques to reduce the complexity of the lookup table include clipping [50], polytopic approximation [51], and convex lifting [52].

An alternative approach to explicit MPC, sometimes called *implicit* MPC, is to solve the optimization problems from (1.1) online with embedded solvers. In particular, the value of θ is determined at each time step (by estimating the system state), which reduces (1.1) to a quadratic program (QP). By solving this QP, an optimal control action given the current state is achieved.

Implicit MPC has spawned great interest in the control community to develop high-performing QP solvers that can run on embedded hardware. Some of these

***Table 1.1:** Quadratic programming solvers for embedded optimization.*

Solver	Reference	Optimization method
OOQP	[71]	Interior point
qpOASES	[72]	Active set
FiOrdOS	[73]	Gradient projection
FORCES	[55]	Interior point
ECOS	[74]	Interior point
PQP	[75]	Projection-free gradient
GPAD	[62]	Gradient projection
QPNNLS	[23]	Active set
qpDUNES	[76]	Dual Newton
QRQP	[77]	Active set
BVLS	[78]	Active set
QPDAS	[79]	Active set
qpSWIFT	[80]	Interior point
OSQP	[63]	Operator splitting
HPIPM	[81]	Interior point
PRESAS	[82]	Active set
NASOQ	[83]	Active set
ASIPM	[84]	Active set + Interior point
FBstab	[85]	Proximal Fischer-Burmeister
DAQP	Paper E	Active set
QPALM	[65]	Augmented Lagrangian
ProxQP	[66]	Augmented Lagrangian
PIQP	[86]	Proximal interior point

solvers are listed in Table 1.1, ordered according to the publication date of its corresponding reference. The optimization methods that the QP solvers in Table 1.1 are based on include interior-point methods [53–55], active-set methods [56–58], gradient-projection methods [59–62], operator-splitting methods [63; 64], and augmented Lagrangian methods [65; 66]. Active-set methods are the main focus of this thesis and are therefore introduced more in-depth in Chapter 3. Two excellent introductions to interior-point methods are given in [67; 68]; a classical introduction to gradient methods is given in [69]. Introductions to both splitting and augmented Lagrangian methods are given in [70].

Nonlinear MPC When the dynamics of the controlled system is nonlinear, MPC requires a general nonlinear program (NLP) to be solved. While generic solvers such as `IPOPT` [87] can be used to solve these NLPs, methods that exploit the inherent structure of NLPs from MPC problems can significantly reduce the computation time. A popular approach is to linearize the dynamics at each iteration and then, as for the linear case described above, solve a quadratic program. This corresponds to the standard sequential quadratic programming (SQP) paradigm in nonlinear optimization [88]. An SQP-method that is adapted for real-time

MPC is that of real-time iterations (RTI) [89], where the linearization is made *before* a state is measured and where the resulting QPs are not solved completely to optimality [90]. Related to RTI, control policies from optimization solvers that do not run completely to optimality are analyzed in [91]. High-performant SQP and RTI routines are implemented in `acados` [92], which is a modular framework for fast embedded optimal control. In addition, `acados` provides specialized routines for the linearization of the dynamics. Other software for nonlinear MPC include `PolyMPC` [93], `GRAMMPC` [94], and `PANOC` [95].

1.3.2 Complexity certification

Generic complexity analysis Traditionally, the field of *computational complexity* deals with the *tractability* of problems [96]. As such, in contrast to the objective of this thesis, the exact number of computations a particular algorithm requires is not the primary focus; rather, the focus is on how the number of computations *scales* with the size of the problem, for example, if the number of computations scales polynomially or exponentially with the dimension of the problem.

A notorious result on the topic of tractability was given in [97] for the simplex method [98]. Therein, Klee and Minty proved that the worst-case number of iterations required by the simplex method to find a solution is exponential in the number of decision variables. This shows more broadly the intractability of *active-set methods*, since the simplex method can be interpreted as an active-set method [99]. Despite its dismal theoretical worst-case bounds, the simplex method has shown great practical performance ever since its emergence. This theory/practice-gap was partly explained through a “smoothed analysis” in [100], where it is shown that a small perturbation from the pathological Klee-Minty problems constructed in [97] results in polynomial complexity.

The exponential complexity of the simplex method, and more broadly that of active-set methods, spawned interest in alternative methods with polynomial worst-case complexity. A first breakthrough was made with the ellipsoid method [101]; unfortunately, it was not competitive with the simplex method in practice. Not long after, Karmakar famously proposed an interior-point method [102] with polynomial worst-case complexity *and* practical performance that was competitive with the simplex method. Still, albeit polynomial, the theoretical iteration bounds of interior-point methods are often several orders of magnitude away from the actual number of iterations performed in practice [103]. Informally, as mentioned in [67; 68], there often seems to be an inverse relationship between the tightness of complexity bounds of an interior point method, and how well the method performs in practice; for example, the popular predictor-corrector method by Mehrotra [104] has worse theoretical bounds than short-step interior-point methods, but the predictor-corrector method is faster in practice for most problems [67].

Classical iteration bounds for gradient methods, under different assumption on the objective function (e.g., smoothness, convexity and strong convexity), are given in [69]. The focus of such bounds is, again, on the rate of convergence, rather than the exact number of iterations or floating-point operations. More recently, direct computational methods for first-order methods have been proposed. In [105], the integral quadratic constraint (IQC) framework [106], well-known in the robust control community, is used to bound the number of iterations. A more general approach is proposed in [107; 108], where a semi-definite program (SDP) is solved to analyze the iteration complexity. This SDP framework is extended in [109] to allow for varying inner products, which is a step towards being able to apply it to interior-point methods.

MPC-specific complexity analysis The traditional computational complexity bounds cited above are often far from the complexity observed in practice. A reason for this is that the set of possible problems that need to be considered in a general setting is large. Hence, as is exemplified in [97], there often exist pathological problems that drive the upper complexity bound; however, these problems are seldom, as is exemplified in [100], encountered in practice.

A cornerstone of this thesis, and in the cited works below, is that the set of possible optimization problems for a given linear MPC application is more restrictive; specifically, they are of the form (1.1), which is a family of quadratic programs parameterized by θ. As is visualized in Figure 1.2, most traditional complexity analyses consider *all* Hessians $H \in \mathbb{S}_n^+$ and *all* constraint matrices $A \in \mathbb{R}^{m\times n}$, while both H and A are fixed for a given linear MPC application; that is, all variation in (1.1) comes from evaluating the affine functions $f(\theta)$ and $b(\theta)$ for different parameters $\theta \in \mathbb{R}^p$.

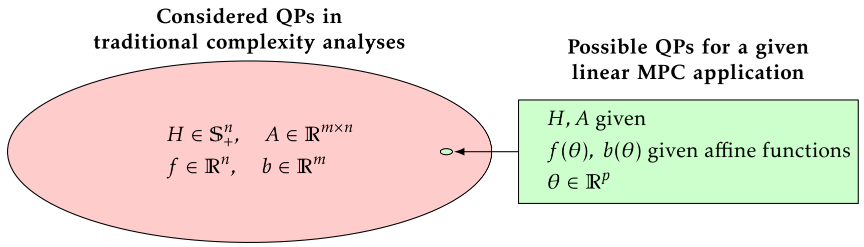

Considered QPs in traditional complexity analyses
$H \in \mathbb{S}_+^n$, $A \in \mathbb{R}^{m\times n}$
$f \in \mathbb{R}^n$, $b \in \mathbb{R}^m$
Possible QPs for a given linear MPC application
H, A given
$f(\theta)$, $b(\theta)$ given affine functions
$\theta \in \mathbb{R}^p$

Figure 1.2: *QPs of the form minimize $\frac{1}{2}x^T Hx + f^T x$ with respect to x, subject to $Ax \leq b$, that are: (i) often considered in traditional complexity analyses; (ii) possible for a given linear MPC application.*

In practice, a common way of exploiting the parametric structure described above is to sample the set of possible parameters θ. Given a set of samples, the corresponding QPs are solved with the algorithm to be certified and the number of required computations is recorded. A major drawback with such simulation-based approaches is that the worst-case performance can never be guaranteed, since the set of possible problems is continuous and that there, hence, always exist problems that have not been sampled.

More systematic approaches for exploiting the parametric structure have been considered for first-order methods. Two important terms in traditional iteration bounds for first-order methods (see e.g., [69]) are a Lipschitz constant and the norm of the difference of the starting iterate and the final iterate. In [62; 110–112], the particular structure of QPs in linear MPC is used to bound these two terms, which yields improved iteration bounds for accelerated gradient methods. For example, the iteration bounds reported in [110] is only about two to five times more conservative than the number of iterations observed in numerical experiments. This is far better than similar work on interior-point methods, where, for example, the bounds obtained in [103] are several order of magnitudes off from the actual number of iterations. Additional work on the complexity certification of first-order methods in the context of linear MPC is reported in [113] for an operator splitting method.

Several certification methods that use the parametric structure have also been proposed for active-set methods in the context of linear MPC. A large part of the contributions of this thesis are within this area. In [18], a method that determines the computational complexity of active-set methods for linear programming is proposed. The work in [17] proposes a certification method for the dual active-set method in [57]; this certification method has been applied in [114] to certify the computational complexity for linear MPC of synchronous motors. A complexity certification method for a standard primal active-set QP method (see, e.g., [15]) is proposed in [16]. The work in [115] certifies the computational complexity of the block-pivot active-set method proposed in [116]. In Paper A, we unify all the certification methods in [16–18] in a common framework. This framework also enables complexity certification of active-set methods that have not previously been certified.

2 Model Predictive Control

At the core of automatic control is the problem of selecting a control action, $u \in \mathbb{R}^{n_u}$, that makes some system state, $z \in \mathbb{R}^{n_z}$, take "desirable" values. An illustrative example is cruise control of a car, where the state z is the speed of the car, the control u is the throttle, and the "desirable" value of the state is the speed in which the driver wants to travel.

Often, the selection of the control action u is made by a *state-feedback law*, which generates a control action u given the current system state z. Formally, a state-feedback law is a mapping $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_u}$ that generates control actions as $u = g(z)$. An example of a simple, yet effective, feedback law is linear state feedback, where g is a linear mapping, i.e., $u = Lz$ for some gain matrix $L \in \mathbb{R}^{n_u \times n_z}$. Challenging control problems do, however, require more sophisticated feedback laws for sufficient performance.

In Model Predictive Control (MPC) [32], the mapping $g(z)$ is evaluated *implicitly* by solving an optimization problem, where this optimization problem depends on the current state z. More specifically, the optimization problem is formulated through a predictive model of the system; this model is then used to create a forecast of how z will change over time, and control actions that optimize this forecast are selected. An overview of MPC is given in Algorithm 1, where N denotes the length of the forecast, z_0 denotes the starting point of the forecast, and $\{u_k^*\}_{k=0}^{N-1}$ denotes the sequence of control actions that produce an "optimal" forecast. What Step 3 in Algorithm 1 entails is the main subject of this chapter.

Algorithm 1 Model predictive control

```
1: repeat
2:     Measure (or estimate) the current state $z$
3:     $\{u_k^*\}_{k=0}^{N-1} \leftarrow$ Formulate and solve (2.3) with $z_0$ set to $z$
4:     Apply the first control action $u_0^*$ to the system
```

In Section 2.2, we give a general formulation of the optimal control problems that are solved in Algorithm 1; these problems are then endowed with additional structure in Section 2.3, which leads to *linear* MPC: the main focus of this thesis. We then show that the optimal control problems solved in linear MPC can be seen as instances of a multi-parametric quadratic program (mpQP), which means that the optimization problems to be solved online are quadratic programs (QPs).

Before describing the optimal control problems, we give some introductory remarks on *prediction models* and *state estimation*; both are essential for MPC to work in practice, but are not considered in detail in this thesis.

2.1 Preliminaries

Prediction model As its name suggests, MPC is a control strategy that uses a *prediction model*. In the usual setting, this prediction model is a differential equation of the form

$$\dot{z} = f_c(z, u), \tag{2.1}$$

where $z \in \mathbb{R}^{n_z}$ is called the *state* of the system, $u \in \mathbb{R}^{n_u}$ is called the *control*, and the mapping $f_c : \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_z}$ is called the *dynamics* of the system.

However, to be able to use the prediction model in practice, a discrete-time model is required, since controllers are almost always[1] implemented on digital computers. Hence, we instead consider discrete-time dynamics of the system given in terms of a difference equation of the form

$$z_{k+1} = f_d(z_k, u_k), \tag{2.2}$$

where u_k and z_k denotes the control and state at time step k, respectively. Two common ways of deriving a discrete-time model f_d given a continuous-time model f_c, is through zero-order or first-order hold.

Approaches for determining a model of the form (2.2) include using physical laws, black-box models and everything in between (gray-box models); all of these, and more, are considered in the research field of *system identification* [118].

Even though determining a model that sufficiently captures the system's behavior is essential for MPC to be practical, we will in this thesis assume that such a model has already been determined, summarized in the following assumption.

Assumption A1 (Existence of prediction model). A discrete-time prediction model $z_{k+1} = f_d(z_k, u_k)$ of the system's dynamics is available.

State estimation State-feedback laws, and hence MPC, require information about the *entire* state z at the current time step (evident from Step 2 in Algorithm 1). In practice, however, the entire state is often not measured directly; instead, only parts of the state and/or quantities that are indirectly related to it are measured. Formally put, we measure a quantity y which is related to z by some mapping $y = h(z, u, w)$, where w is measurement noise that distorts the measurements. In

[1]See [117] for work on MPC with *analog* computers.

this situation, an estimate of z, denoted $\hat{z}$, is formed based on y and u. Mathematically, one forms a mapping $(y, u) \mapsto \hat{z}$ such that $\hat{z} \approx z$. The problem of constructing such a mapping is known as *state estimation* [119] and is, similar to system identification, an entire research field on its own. In this thesis we do not consider the state estimation problem; in other words, we assume that measurements of z, or at least an estimate of z, are available, summarized in the following assumption.

Assumption A2 (Availability of state). At each time step, either the state z is measured or an estimate of the state is available.

Remark 2.1. For convenience, we will only refer to the state z in the rest of the thesis, even though we might, practically speaking, mean the estimated state $\hat{z}$.

2.2 Optimal control

Assuming that Assumption A1 and A2 hold, i.e., that a prediction model f_d and the current state z are available, we turn our attention to the control problem considered in MPC:

> Given the state z, select an *inexpensive* and *admissible* sequence of control actions $\{u_k\}_{k=0}^{N-1}$ that generates a *desirable* sequence of states $\{z_k\}_{k=0}^{N}$, with $z_0 = z$. $(\star)$

The integer N in $(\star)$ is called the *prediction horizon* and determines how many time steps into the future we predict when selecting control actions. For convenience we will use the notation $\mathbf{u} \triangleq \{u_k\}_{k=0}^{N-1}$ and, similarly, $\mathbf{z} \triangleq \{z_k\}_{k=0}^{N}$ for the control and state sequences, respectively.

The fuzzy terms "inexpensive" and "desirable" used in $(\star)$ can be formalized through a so-called cost function $V(\mathbf{z}, \mathbf{u})$, which assigns a real value to a control sequence and its corresponding state sequence. A small value of V means that the sequence of control actions is "inexpensive" and that the generated states take "desirable" values. It is therefore up to the user to encode their notion of "inexpensive" and "desirable", as well as the trade-off between them, in V. The control problem can, hence, be stated as the *optimal* control problem of finding a state sequence $\mathbf{z}$ and control actions $\mathbf{u}$ which minimize V. We will in this thesis, as is often done in practice, consider V of the form $V(\mathbf{z}, \mathbf{u}) = V_f(z_N) + \sum_{k=0}^{N-1} \ell(z_k, u_k)$, where $V_f : \mathbb{R}^{n_z} \to \mathbb{R}$ is called the terminal cost and $\ell : \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is called the stage cost.

Importantly when minimizing V, the state sequence cannot be selected freely, since the state z_{k+1} is constrained by the dynamics $z_{k+1} = f_d(z_k, u_k)$. Often in practice, we cannot select $\mathbf{u}$ freely either, since the control actions are constrained to a set $u \in \mathcal{U}$, and since the states are constrained to a set $z \in \mathcal{Z}$. Such constraints arise from, e.g., actuator limits and speed limits, respectively. An additional constraint that the final predicted state z_N should reside in a set $\mathcal{Z}_f$, rather than $\mathcal{Z}$, is often used to guarantee that the resulting feedback law is stable [120].

In conclusion, the control problem in (★) can be formalized as the following *discrete-time optimal control problem*:

$$
\begin{aligned}
\underset{\mathbf{u},\mathbf{z}}{\text{minimize}} \quad & V_f(z_N) + \sum_{k=0}^{N-1} \ell(z_k, u_k) \\
\text{subject to} \quad & z_{k+1} = f_d(z_k, u_k), \quad \forall k = 0, \dots, N-1 \\
& (u_k, z_k) \in \mathcal{U} \times \mathcal{Z}, \quad \forall k = 0, \dots, N-1 \\
& z_N \in \mathcal{Z}_f \\
& z_0 = z,
\end{aligned}
\tag{2.3}
$$

where the minimizing $\mathbf{u}$ and $\mathbf{z}$ are denoted $\mathbf{u}^*$ and $\mathbf{z}^*$, respectively. Since the optimal control problem depends on the current state z through the initial constraint $z_0 = z$, both $\mathbf{u}^*$ and $\mathbf{z}^*$ are functions of z, i.e., $\mathbf{u}^* : \mathbb{R}^{n_z} \to \mathbb{R}^{n_u}$ and $\mathbf{z}^* : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$.

Remark 2.2. The simulation in (2.3) is always done from time index 0 to N since we have considered *time-invariant* dynamics, stage cost, and constraint sets. Hence, only *relative*, rather than *absolute*, time is of importance. The problem formulation can easily be extended to also handle the time-varying case (see, e.g., [32]), but this is not considered in this thesis.

Receding horizon

As mentioned in the introduction of this chapter and summarized by Algorithm 1, discrete optimal control problems of the form (2.3) are solved recurrently in MPC, with the initial state z_0 constrained to be the current state z. After solving such a problem, resulting in the optimal control sequence $\mathbf{u}^*(z) = \{u_i^*(z)\}_{i=0}^{N-1}$, only the *first* control action $u_0^*(z)$ is applied to the system. In other words, MPC implicitly defines the feedback law $u = u_0^*(z)$, and all $u_i^*(z)$ $i = 1, \dots, N-1$ are discarded. Discarding all control actions except $u_0^*(z)$ might at first seem drastic; there are, however, two main reasons for re-solving the optimal control problems in each time step:

The first reason is due to the finite horizon N. Since states beyond N time steps into the future are not accounted for in (2.3), consequences beyond the horizon are not directly taken into account. By re-solving the optimal control problems recurrently, we can take into account consequences beyond the nominal horizon. This is known as receding the horizon and because of this MPC is also known as a *receding horizon control* (RHC) strategy.

Remark 2.3. States beyond the horizon N can, however, be indirectly accounted for by selecting the terminal cost V_f and set $\mathcal{Z}_f$ with care; see, e.g., [120].

The second advantage of re-solving the optimal control problems in each time step is that the *predicted* states are just that: predictions. Even if we could let the horizon N tend to infinity, the optimized sequence of control actions cease to be optimal as soon as the predicted state trajectory deviates from the actual state trajectory; this always occurs in practice due to model errors and external disturbances. By re-solving the optimal control problem for the current state we can reassess our control actions, which robustifies the resulting control policy.

2.3 Linear MPC

The difficulty in solving (2.3) depends on the specific structure of the cost function, dynamics and constraints. A well-established structure that is commonly used in practice, [33], is based on the following assumptions:

Assumption A3 (Quadratic cost). The terminal and stage cost are *quadratic*. That is, $V_f(z) = \frac{1}{2}z^T Q_f z$ and $\ell(z,u) = \frac{1}{2}\left(z^T Q z + u^T R u\right)$ for some $Q_f, Q \in \mathbb{S}_+^{n_z}$ and $R \in \mathbb{S}_{++}^{n_u}$.

Assumption A4 (Linear dynamics). The system dynamics is *linear*. That is, $f_d(z,u) = Fz + Gu$ for $F \in \mathbb{R}^{n_z \times n_z}$ and $G \in \mathbb{R}^{n_z \times n_u}$.

Assumption A5 (Polyhedral constraints). The sets $\mathcal{Z} \times \mathcal{U}$ and $\mathcal{Z}_f$ are *polyhedral*. That is, $\mathcal{Z} \times \mathcal{U} = \{(z,u) : A_z z + A_u u \leq b\}$ for $A_z \in \mathbb{R}^{n_c \times n_z \cdot}$, $A_u \in \mathbb{R}^{n_c \times n_u}$ and $b \in \mathbb{R}^{n_c}$, and $\mathcal{Z}_f = \{x : A_f z \leq b_f\}$.

Problem (2.3) with Assumptions A3-A5 leads to the optimal control problem

$$\begin{aligned}
\underset{\mathbf{u},\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2} z_N^T Q_f z_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(z_k^T Q z_k + u_k^T R u_k\right) \\
\text{subject to} \quad & z_{k+1} = F z_k + G u_k, \quad k = 0,\dots,N-1 \\
& A_z z_k + A_u u_k \leq b, \quad k = 0,\dots,N-1 \\
& A_f z_N \leq b_f \\
& z_0 = z.
\end{aligned} \tag{2.4}$$

Remark 2.4. The control objective in (2.4) is to steer some or all states to the origin since $\frac{1}{2}z_N^T Q_f z_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(z_k^T Q z_k + u_k^T R u_k\right) \geq 0$, with equality for $z_i = 0$ and $u_i = 0$, $\forall i$ (following from $Q_f, Q \succeq 0$ and $R \succ 0$). The problem can, however, be modified to also be able to steer to another reference point than the origin, which is described in Section 2.3.2.

Algorithm 1 with optimization problems of the form (2.4) being solved in Step 3 is called *linear* MPC. To concretize the above-mentioned concepts, the following example illustrates how linear MPC can be used to stabilize an inverted pendulum on a cart.

Example 2.1: Linear MPC of an inverted pendulum

Consider the system illustrated in Figure 2.1 of an inverted pendulum on a cart. The control objective is to stabilize the system standing straight up ($\phi = 0$) with no displacement ($x = 0$). To do this, the cart's acceleration can be changed by applying a horizontal force F, which can maximally have a magnitude of 100 Newton. For this system we consider the state $z = (x, v, \phi, \omega)^T$, where v and ω are the velocity of the cart and the angular velocity of the pendulum, respectively, and the control $u = F/100$ (where the scaling is done for numerical reasons).

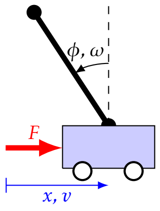


Figure 2.1: *Inverted pendulum*

A discrete-time model $z_{k+1} = Fz_k + Gu_k$, with sampling time $T_s = 0.02$ s, of the system dynamics is given by

$$F = \begin{pmatrix} 1 & 0.0181 & 0.0018 & 0.0000 \\ 0 & 0.8185 & 0.1783 & 0.0018 \\ 0 & -0.0038 & 1.0076 & 0.0201 \\ 0 & -0.3635 & 0.7500 & 1.0067 \end{pmatrix}, \quad G = \begin{pmatrix} 0.02 \\ 1.82 \\ 0.04 \\ 3.64 \end{pmatrix}. \tag{2.5}$$

This model is based on a nonlinear continuous-time model (derived through Newton's laws) that has been linearized around the origin and discretized using zero-order hold. Since the dynamics is linearized around the origin, we impose constraints on the angle ϕ to keep the states close to this linearization point, leading to the artificial constraint of $|\phi| \leq 0.5$.

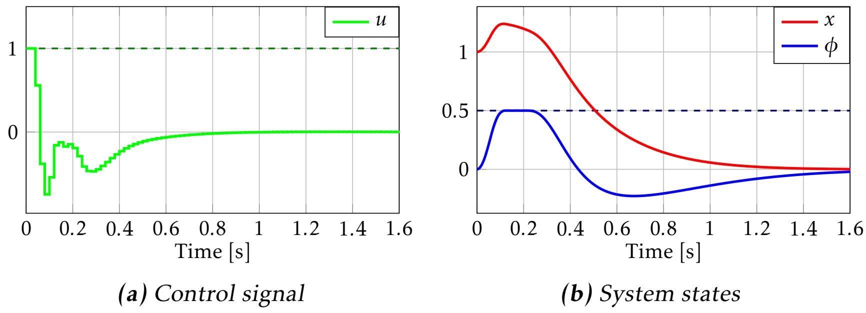
The constraints $|F| \leq 100$ and $|\phi| \leq 0.5$ can be cast in the form $A_z z + A_u u \leq b$ with

$$A_z = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_u = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}, \quad b = \begin{pmatrix} 0.5 \\ 0.5 \\ 1 \\ -1 \end{pmatrix}. \tag{2.6}$$

No specific terminal set is used: only $|\phi_N| \leq 0.5$ is imposed.

The horizon $N = 10$ is used and, since the objective is to drive x and ϕ (the first and third state) to zero, the weights in the stage cost is selected as $Q = \mathbf{diag}(10, 0, 1, 0)$ and $R = 1$. The final cost Q_f is set to the solution to the discrete-time algebraic Riccati equation that is solved to obtain the linear quadratic regulator (LQR) for an infinite horizon (see, e.g., [120] for details).

Figure 2.2 shows the resulting control actions and state trajectories when the system's starting state is $z_0 = (1, 0, 0, 0)^T$ (a displacement of 1 meter with the pendulum standing straight up at rest). The control law defined by Algorithm 1 is used, where the subproblems are of the form (2.4) with the above-mentioned data. The MPC controller generates control actions that steer both x and ϕ to zero, which was the goal, while satisfying the constraints imposed on u and ϕ. (Note that the linearized model also has, for simplicity, been used for the experiments).



***(a)** Control signal* ***(b)** System states*

***Figure 2.2:** Resulting control signal and state trajectory using MPC to stabilize the inverted pendulum. Control/state constraints are shown as dashed lines.*

2.3.1 Multi-parametric Quadratic Programming

A major advantage with endowing the optimal control problem (2.3) with the structure from Assumptions A3-A5 is that the resulting optimal control problem in (2.4) can be expressed as a multi-parametric quadratic program (mpQP), resulting in quadratic programs (QPs) being the optimization problems to solve. Solving problems on the classical problem formulation of a QP enables the use of the myriad of efficient quadratic programming methods and software available, as highlighted by Table 1.1 in Section 1.3.

Multi-parametric quadratic programs are of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + (f + f_\theta \theta)^T x \\ \text{subject to} \quad & Ax \leq b + W\theta \\ & Ex = d + D\theta, \end{aligned} \tag{2.7}$$

with the decision variable $x \in \mathbb{R}^n$ and the parameter $\theta \in \Theta_0 \subseteq \mathbb{R}^p$. The objective function is characterized by $H \in \mathbb{S}_+^n$, $f \in \mathbb{R}^n$, and $f_\theta \in \mathbb{R}^{n \times p}$. The inequality constraints are given by $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times p}$, and the equality constraints are given by $E \in \mathbb{R}^{e \times n}$, $d \in \mathbb{R}^e$ and $D \in \mathbb{R}^{e \times n}$. By changing the parameter θ, the linear term in the objective function and the right-hand-sides of the inequality and equality constraints are perturbed, resulting in different quadratic programs.

The main idea for putting the optimal control problem in (2.4) into the form of an mpQP in (2.7) is to view the initial state z_0 as the parameter ($\theta = z_0$) and to stack the control and states into vectors as

$$\mathbf{z} = \begin{pmatrix} z_0 \\ \vdots \\ z_N \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1}. \end{pmatrix}. \tag{2.8}$$

This results in $\mathbf{z} \in \mathbb{R}^{n_z(N+1)}$ and $\mathbf{u} \in \mathbb{R}^{n_u N}$. There are two common ways of expressing (2.4) as an mpQP: A sparse formulation and a condensed formulation.

Remark 2.5. (2.8) overloads the notation of $\mathbf{u}$ and $\mathbf{z}$ introduced in Section 2.2, but the intrinsic entities are the same: previously the entities were represented as sequences and, in this section, we represent the entities by stacking the elements of these sequences into vectors.

Sparse formulation

The most straightforward way of formulating the discrete optimal control problem in (2.4) as an mpQP is to view both $\mathbf{z}$ and $\mathbf{u}$ as optimization variables, leading to a *sparse* mpQP. First, the equality constraints in (2.4) for each time step k can, together with the initial constraint $z_0 = z = \theta$, be expressed as the following block-structured linear system of equations

$$\underbrace{\begin{pmatrix} I & 0 & 0 & \cdots & 0 \\ -F & I & 0 & \cdots & 0 \\ 0 & -F & I & \cdots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & \cdots & 0 & -F & I \end{pmatrix}}_{\triangleq \mathbf{E}_z} \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ -G & 0 & \cdots & 0 \\ 0 & -G & \ddots & 0 \\ \vdots & \ddots & \ddots & \\ 0 & 0 & & -G \end{pmatrix}}_{\triangleq \mathbf{E}_u} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} = \underbrace{\begin{pmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\triangleq \mathbf{D}} \theta. \tag{2.9}$$

Next, the stage costs $z_k^T Q z_k + u_k^T R u_k$ and terminal cost $z_N^T Q_f z_N$ can be combined, by introducing $\mathbf{Q} \triangleq \mathbf{diag}\left(Q, \dots, Q, Q_f\right)$ and $\mathbf{R} \triangleq \mathbf{diag}\left(R, \dots, R\right)$, into

$$V(\mathbf{z}, \mathbf{u}) = \frac{1}{2}\left(\mathbf{u}^T \mathbf{R} \mathbf{u} + \mathbf{z}^T \mathbf{Q} \mathbf{z}\right). \tag{2.10}$$

Finally, the inequality constraints $A_z z_k + A_u u_k \leq b$ for $k = 0, \dots, N-1$ and the terminal constraint $A_f z_N \leq b_f$ can be combined, by introducing block diagonal matrices $\mathbf{A}_u \triangleq \begin{pmatrix} \mathbf{diag}\left(A_u, \dots, A_u\right) \\ 0 \end{pmatrix}$ and $\mathbf{A}_z \triangleq \mathbf{diag}\left(A_z, \dots, A_z, A_f\right)$ and the vector $\mathbf{b} \triangleq \mathbf{vec}\left(b, \dots, b, b_f\right)$, into

$$\mathbf{A}_u \mathbf{u} + \mathbf{A}_z \mathbf{z} \leq \mathbf{b}. \tag{2.11}$$

Combining (2.9), (2.10) and (2.11) gives the multi-parametric quadratic program

$$\begin{aligned} \underset{\mathbf{u}, \mathbf{z}}{\text{minimize}} \quad & \frac{1}{2} \begin{pmatrix} \mathbf{u} \\ \mathbf{z} \end{pmatrix}^T \begin{pmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{Q} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{z} \end{pmatrix} \\ \text{subject to} \quad & \begin{pmatrix} \mathbf{E}_u & \mathbf{E}_z \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{z} \end{pmatrix} = \mathbf{D}\theta \\ & \begin{pmatrix} \mathbf{A}_u & \mathbf{A}_z \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{z} \end{pmatrix} \leq \mathbf{b}, \end{aligned} \tag{2.12}$$

which, with optimization variable $x = \begin{pmatrix} \mathbf{u} \\ \mathbf{z} \end{pmatrix}$, is of the form (2.7).

Condensed formulation

Considering $\mathbf{z}$ as an optimization variable, as is done in (2.12), is somewhat superfluous, since $\mathbf{z}$ is completely determined by the starting state z, the dynamics f_d, and the control actions $\mathbf{u}$. Hence, the problem can be *condensed* by using the equality constraints in (2.9) to eliminate $\mathbf{z}$, resulting in optimization over only $\mathbf{u}$. The most direct way of doing this is by inverting $\mathbf{E}_z$ in (2.9) to express $\mathbf{z}$ in terms of $\mathbf{u}$ and θ as

$$\mathbf{z} = \mathbf{E}_z^{-1}\mathbf{D}\theta - \mathbf{E}_z^{-1}\mathbf{E}_u\mathbf{u}, \tag{2.13}$$

where $\mathbf{E}_z^{-1}$ is guaranteed to exist since $\mathbf{E}_z$ is lower unit triangular. An alternative interpretation to the purely mathematical one of inverting $\mathbf{E}_z$ is to use the linear dynamics for forward simulation; this enables future states to be expressed in terms of only the control actions $\mathbf{u}$ and the starting state $z_0 = \theta$. That is, by using the dynamics $z_{k+1} = Fz_k + Gu_k$ iteratively the state at time step k can be expressed as

$$z_k = F^k z_0 + \sum_{i=1}^{k} F^{k-i} G u_i. \tag{2.14}$$

Stacking these equations for $k = 0, \ldots, N$, and using $z_0 = z = \theta$, leads to the linear system of equations

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} = \underbrace{\begin{pmatrix} I \\ F \\ F^2 \\ \vdots \\ F^N \end{pmatrix}}_{\triangleq \mathbf{F}} \theta + \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ G & 0 & \cdots & 0 \\ FG & G & \cdots & 0 \\ \vdots & & \ddots & \\ F^{N-1}G & F^{N-2}G & \cdots & G \end{pmatrix}}_{\triangleq \mathbf{G}} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix}, \tag{2.15}$$

more compactly written as

$$\mathbf{z} = \mathbf{F}\theta + \mathbf{G}\mathbf{u}. \tag{2.16}$$

Eliminating $\mathbf{z}$ in (2.12) using (2.16), and removing terms in the objective function that do not contain $\mathbf{u}$, gives the mpQP

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{u}^T\left(\mathbf{G}^T\mathbf{Q}\mathbf{G} + \mathbf{R}\right)\mathbf{u} + \theta^T\mathbf{F}^T\mathbf{Q}\mathbf{G}\mathbf{u} \\ \text{subject to} \quad & (\mathbf{A}_z\mathbf{G} + \mathbf{A}_u)\,\mathbf{u} \leq \mathbf{b} - \mathbf{A}_z\mathbf{F}\theta, \end{aligned} \tag{2.17}$$

which is of the form (2.7) with $x \triangleq \mathbf{u}$.

The condensed formulation does not contain any equality constraints and will for convenience be considered in the rest of the thesis. In other words, we consider mpQPs of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + (f + f_\theta\theta)^T x \\ \text{subject to} \quad & Ax \leq b + W\theta \end{aligned} \tag{2.18}$$

.

The subsequent ideas can, however, easily be extended to the case when equality constraints are used, but this requires some additional, obfuscating, notation. Some details on how equality constraints are handled in active-set methods are given in Paper E.

Remark 2.6. There is also a third approach, to which the sparse and condensed formulation are mere special cases. This approach is known as *partial condensing* [37], and only eliminates a *subset* of the elements of the stacked state vector $\mathbf{z}$ (in contrast to eliminating none or all elements of $\mathbf{z}$ which is done in the sparse and condensed formulation, respectively). The important point for this thesis is that the resulting optimization problems after partial condensing also can be represented as an mpQP of the form (2.7), i.e., all the subsequent ideas derived for mpQPs also apply to when partial condensing is used.

Remark 2.7. By picking another running cost than in Assumption A3, namely the 1- or ∞-norm, we can reformulate the MPC problem as a multi-parametric *linear* program (mpLP) rather than an mpQP (for details see, e.g., Chapter 9 in [33]). An mpLP can, however, be seen as a special case of an mpQP with $H = 0$.

2.3.2 Extending the linear MPC formulation

For practical purposes, the optimal control problems in (2.4) might need to be modified for certain applications. Here we describe how (2.4) can be extended to handle reference tracking and how the state constraints can be *softened* to ensure that a solution to the problem always exists. Importantly, both of these extensions still allow the resulting optimal control problem to be cast as an mpQP of the form (2.18), which is the basic starting point in the contributions described in Part II. Simply put, the exact origins of the mpQP is not important in Part II since all the results are derived for general mpQPs of the form (2.18).

Reference tracking

In the optimal control problem in (2.4), the objective is to steer all or some of the states to the origin (see Remark 2.4). In a more general setting, we might instead want to steer a linear combination of the states to a given value; that is, we want $Cz = r$ to hold, where $r \in \mathbb{R}^{n_r}$ is the desired reference value and $C \in \mathbb{R}^{n_r \times n_z}$ characterize the quantities to be controlled. In particular, the regulation problem considered in (2.4) is a special case with $r = 0$ and $C = I$.

When the reference value $r \neq 0$, a non-zero control action might be required to maintain the states at the desired reference. Hence, it becomes more reasonable to incur a cost on the change of u rather than its magnitude. In other words, we would like to incur a cost on $\Delta u_k \triangleq u_k - u_{k-1}$ rather than on u_k.

These extensions result in the modified objective function

$$V(\mathbf{z}, \mathbf{u}, r, u_{-1}) = \sum_{k=0}^{N-1} (Cz_k - r)^T Q (Cz_k - r) + \Delta u_k^T R \Delta u_k, \tag{2.19}$$

where now $Q \in \mathbb{S}_+^{n_r}$, in contrast to $Q \in \mathbb{S}_+^{n_z}$ from before. The optimal control problem (2.4) with the new objective function (2.19) can, using the ideas described in

Section 2.3.1, be cast as an mpQP. Consequently, the parameter θ contains, in addition to the state z, the reference value r and the previous control actions u_{-1}, i.e., $\theta = \mathbf{vec}(z, r, u_{-1})$. For additional details, see, e.g., Section 6.1 in [38].

Softening constraints

Since the constraints in (2.18) might be parameter dependent, some parameter values can lead to a QP that does not have a solution; that is, some parameters $\theta \in \mathbb{R}^p$ might result in $\{x : Ax \leq b + W\theta\} = \emptyset$. This occurs, for example, if the current state z violates state constraints that are imposed in (2.4). An approach to always ensure that the resulting QP has a solution, for any value of $\theta \in \mathbb{R}^p$, is to allow constraints to be violated, but to incur a cost for such violations. Constraints that are allowed to be violated are called *soft* constraints, while constraints that are always enforced are called *hard* constraints.

Usually in MPC, constraints on the control u are considered hard and constraints on the state z are considered soft. The reason for this is twofold: partly technical and partly pragmatic. The technical reason is that constraints on z are typically what cause the QPs to be infeasible. Softening these state constraints, hence, ensures that a solution exists. As for the pragmatic reason, constraints on u are often based on physical limitations of actuators, which are impossible to violate in practice, making them inherently hard. In contrast, state constraints often arise from desired system specifications that can be violated temporarily if needed; for example, cars are often capable of traveling significantly faster than enforced speed limits, but doing so would jeopardise safety.

There are several ways of softening constraints. A common approach, [121], is to add an optimization variable $\epsilon_s \in \mathbb{R}$ and to modify the constraints as

$$Ax \leq b + W\theta \rightarrow Ax \leq b + W\theta + S\epsilon_s, \tag{2.20}$$

where $S \in \mathbb{R}^m$ is a selection matrix with $[S]_i = 0$ if the ith constraints is a hard constraint and $[S]_i = 1$ if the ith constraint is a soft constraint. A large value on ϵ_s will relax the soft constraints, and by making it large enough, a feasible point will always be available for any value $\theta \in \mathbb{R}^p$. To incur a cost when soft constraints are violated, a term $\rho\epsilon_s^2$ is added to the objective function, where typically $\rho \gg 0$ to make sure that $\epsilon_s^* = 0$ if the unsoftened problem is feasible. In summary, the softened problem is given by

$$\begin{aligned} \underset{x,\epsilon_s}{\text{minimize}} \quad & \frac{1}{2}\begin{pmatrix} x \\ \epsilon_s \end{pmatrix}^T \begin{pmatrix} H & 0 \\ 0 & \rho \end{pmatrix} \begin{pmatrix} x \\ \epsilon_s \end{pmatrix} + \begin{pmatrix} (f + f_\theta\theta) \\ 0 \end{pmatrix}^T \begin{pmatrix} x \\ \epsilon_s \end{pmatrix} \\ \text{subject to} \quad & \begin{pmatrix} A & -S \end{pmatrix} \begin{pmatrix} x \\ \epsilon_s \end{pmatrix} \leq b + W\theta, \end{aligned} \tag{2.21}$$

which, importantly, is of the form (2.18).

3 Active-set methods for convex Quadratic Programming

In the previous chapter we derived how the optimization problems encountered in linear MPC can be cast as instances of a multi-parametric quadratic program (mpQP). Consequently, the optimization problems that need to be solved at each time step are quadratic programs (QPs). In this chapter we present in detail a popular class of algorithms for solving such QPs: active-set methods.

In Section 3.1 we give a brief overview of quadratic programming and in Section 3.2 we introduce active-set methods. A detailed description of a primal active-set algorithm is given Section 3.3, which is used to formulate a dual active-set algorithm in Section 3.4. Finally, Section 3.5 introduces practical aspects to consider when implementing active-set algorithms.

3.1 Quadratic Programming

If the parameter θ is fixed in the mpQP in (2.18), which in the context of linear MPC corresponds to measuring the state, the optimization problem becomes a quadratic program of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & J(x) \triangleq \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & [A]_i x \leq [b]_i, \quad \forall i \in \mathbb{N}_m \end{aligned} \tag{3.1}$$

where $[\,\cdot\,]_i$ denotes the ith row of a matrix. The objective function $J : \mathbb{R}^n \to \mathbb{R}$ consists of a quadratic term, defined by $H \in \mathbb{S}_+^n$, and a linear term, defined by $f \in \mathbb{R}^n$. The feasible set is a polyhedron, defined by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

The following example visualizes a two-dimensional QP to provide some geometrical intuition for (3.1) in the case when $H \succ 0$.

Example 3.1: Visualization of QP

Consider the quadratic program

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & 0.5x^2 + 2y^2 + xy + x - 14y \\ \text{subject to} \quad & -2x + y \leq 7, \\ & -x + 3y \leq 11, \end{aligned} \tag{3.2}$$

which is of the form (3.1) with $H = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix}$, $f = \begin{pmatrix} 1 \\ -14 \end{pmatrix}$, $A = \begin{pmatrix} -2 & 1 \\ -1 & 3 \end{pmatrix}$ and $b = \begin{pmatrix} 7 \\ 11 \end{pmatrix}$. Figure 3.1 illustrates the level sets of the objective function J, which are ellipses, together with the feasible set $Ax \leq b$. The minimum to (3.2) is obtained for $x^* = \big(-2, 3\big)$, which is the point where a level curve "just touches" the feasible set.

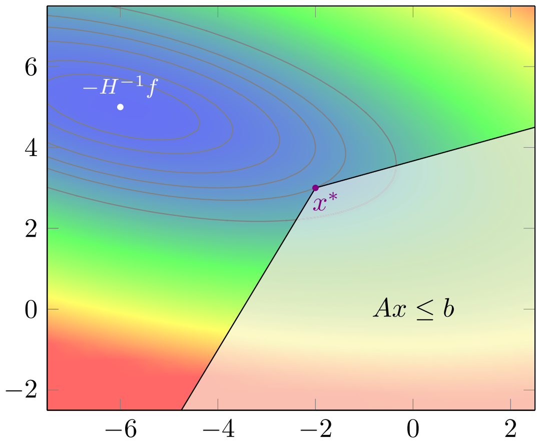


Figure 3.1: *Example quadratic program. Warmer colors correspond to higher objective function values. The gray ellipses correspond to some level curves of the objective function (which increase outward from $x = -H^{-1}f$). The white opaque area corresponds to the feasible set.*

Remark 3.1. The shape of the level curves of (3.1), which are ellipses when $H \succ 0$, only depends on H and *not* f; changing f only translates the unconstrained minimum $-H^{-1}f$, which is the center of these ellipses. Similarly, the normals of the constraining half-planes only depend on A and *not* b; changing b only offsets these half-planes. Hence, since the parameter θ in (2.18) neither affects H nor A, different values of θ only translate the unconstrained optimum and offset the half-planes: the shape of the level curves and the normal of the half-planes remain the same. This structure is exploited in explicit MPC [49] and in the framework presented in Paper A.

3.1.1 Feasibility

When the QP in (3.1) does not have a solution it is said to be *infeasible*. Infeasibility can arise in two different cases. The first case occurs when there does not exist any point that satisfies the constraint $Ax \leq b$, i.e., if the feasible set is empty: $\{x \in \mathbb{R}^n : Ax \leq b\} = \emptyset$. In this case the problem is said to be *primal infeasible* or sometimes (carelessly) just *infeasible*. The processes of finding a point that satisfies $Ax \leq b$ is often called *phase-1*.

The second case of infeasibility can only occur if H is singular. Then there might exist primal feasible points that make the objective function arbitrarily small ($J \to -\infty$). Hence, a minimum does not exist. In this case the QP is said to be *dual infeasible* or *unbounded*. A simple example of an unbounded QP is the problem of minimizing $x^2 + y$ subject to $x \leq 0$ and $y \leq 0$, which can be made arbitrarily small by keeping x fix and decreasing y. (Note that if the constraint would be $y \geq 0$ instead of $y \leq 0$ the resulting QP would be bounded, highlighting that unboundedness is dependent on both the objective function *and* the feasible set).

3.1.2 Optimality

A solution to (3.1), denoted x^*, satisfies the following optimality conditions, known as the *KKT conditions* [122]:

$$Hx^* + A^T\lambda = -f \tag{3.3a}$$
$$Ax^* \leq b \tag{3.3b}$$
$$\lambda \geq 0 \tag{3.3c}$$
$$([b]_i - [A]_i x^*)[\lambda]_i = 0, \quad \forall i \in \mathbb{N}_m, \tag{3.3d}$$

for some dual variable $\lambda \in \mathbb{R}^m$. Condition (3.3a) is called the *stationarity* condition, (3.3b) is called the *primal feasibility* condition, (3.3c) is called the *dual feasibility* condition, and (3.3d) is called the *complementary slackness* condition. These conditions encode the following necessary properties for an optimal point x^*:

- x^* satisfies the constraints in (3.1); ensured by (3.3b).
- The gradient at x^* is perpendicular to the feasible set; ensured by (3.3a), (3.3c) and (3.3d).

For a more detailed description of these conditions, see, e.g., Chapter 12 in [15].

In general the KKT conditions are only necessary conditions for optimality, but for convex problems, which for QPs is equivalent to $H \succeq 0$, the conditions are also sufficient [123]. Furthermore, (3.3a) has a unique solution x^* if $H \succ 0$, while there might be multiple solutions if $H \succeq 0$.

Finally, note that the conditions (3.3a), (3.3b), and (3.3c) are all linear while (3.3d) is nonlinear. The complementary slackness condition (3.3d) is, hence, what makes quadratic programming nontrivial. In active-set methods, soon to be introduced, the complementary slackness condition is enforced to hold throughout all iterations, while the rest of the conditions are gradually ensured to hold.

3.2 Active-set methods

Most of the difficulty in solving the QP in (3.1) stems from the inequality constraints $Ax \leq b$. If, instead, the QP only contained *equality* constraints, the problem could be solved by solving *one* set of linear equations. Concretely, the minimizer x^* of the equality constrained QP (EQP)

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & Ex = d, \end{aligned} \tag{3.4}$$

is a solution to the linear equation system

$$\begin{pmatrix} H & E^T \\ E & 0 \end{pmatrix}\begin{pmatrix} x^* \\ \lambda \end{pmatrix} = \begin{pmatrix} -f \\ d \end{pmatrix}, \tag{3.5}$$

often called a *KKT system*. In other words, stationarity (3.3a) and primal feasibility (3.3b) are the only necessary conditions for optimality when only equality constraints are present, whereas dual feasibility (3.3c) and complementary slackness (3.3d) also become necessary for optimality once inequality constraints are present.

The straightforwardness of solving EQPs is what is exploited in active-set methods. An important insight, that forms the foundation for active-set methods, is that only the constraints that *hold with equality* at x^* are relevant for finding an optimizer, motivating the following definitions.

Definition 3.1 (Active constraint). An inequality constraint $a^T x \leq c$ is *active* at a point $\tilde{x} \in \mathbb{R}^n$ if it holds with equality at $\tilde{x}$, i.e., if $a^T \tilde{x} = c$.

Definition 3.2 (Active set). The *active set* at a point $x \in \mathbb{R}^n$, denoted $\mathcal{A}(x)$, to (3.1) is the set of all inequality constraints that are active at x; that is, the set $\mathcal{A}(x) \triangleq \{i \in \mathbb{N}_m : [A]_i x = [b]_i\}$.

The following lemma formalizes the importance of the active set at x^*; intuitively, it states that removing constraints that are inactive at x^* from the problem formulation does not change the solution x^*.

Lemma 3.1 (Sufficiency of active set). *Let x^* be a solution to* (3.1) *and let $\mathcal{A}^* \triangleq \mathcal{A}(x^*)$. Then x^* is also the solution to the EQP*

$$\begin{aligned} \underset{x}{\textit{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \textit{subject to} \quad & [A]_i x = [b]_i, \quad \forall i \in \mathcal{A}^* \end{aligned} \tag{3.6}$$

Proof: From the complementary slackness condition (3.3d) we have that $[\lambda]_i = 0,\ \forall i \in \mathbb{N}_m \setminus \mathcal{A}^*$. This inserted into the stationarity condition gives

$$Hx^* + [A]^T_{\mathcal{A}^*}[\lambda]_{\mathcal{A}^*} = -f. \tag{3.7}$$

Furthermore, the definition of $\mathcal{A}^*$ imposes the equality constraint

$$[A]_{\mathcal{A}^*}x^* = [b]_{\mathcal{A}^*}. \tag{3.8}$$

Taken together, (3.7) and (3.8) form the KKT system

$$\begin{pmatrix} H & [A]_{\mathcal{A}^*}^T \\ [A]_{\mathcal{A}^*} & 0 \end{pmatrix} \begin{pmatrix} x^* \\ [\lambda]_{\mathcal{A}^*} \end{pmatrix} = \begin{pmatrix} -f \\ [b]_{\mathcal{A}^*} \end{pmatrix},$$

which coincides with the KKT system for (3.6). □

The key takeaway from Lemma 3.1 is that if $\mathcal{A}^*$ would be known, solving (3.1) simplifies to solving a single system of linear equations. This motivates the main objective of active-set methods: identifying $\mathcal{A}^*$. This identification is done iteratively by updating a so-called *working set*, denoted $\mathcal{W}$, which can be seen as an estimate of $\mathcal{A}^*$. Updates to $\mathcal{W}$ are done by adding/removing constraints to/from it. Such additions/removals are determined by solving an EQP, defined by the current working set $\mathcal{W}$. In other words, the task of solving the QP is split into solving a sequence of EQPs (system of linear equations), where each EQP is defined by the current working set. A prototypical active-set algorithm for quadratic programming, which summarizes the discussion above, is given in Algorithm 2.

Algorithm 2 (Prototypical active-set method for solving QP (3.1))

```
1: repeat
2:     $(x, \lambda) \leftarrow$ Solve KKT system defined by $\mathcal{W}$
3:     if $(x, \lambda)$ is primal and dual feasible then
4:         return $(x^*, \lambda^*, \mathcal{A}^*) \leftarrow (x, \lambda, \mathcal{W})$        ▷ Optimal solution found
5:     else
6:         Modify $\mathcal{W}$ based on primal and/or dual violation of $x$ and $\lambda$.
```

Different approaches for modifying the working set $\mathcal{W}$ at Step 6 lead to different types of active-set methods:

- **Primal** methods work with a primal iterate x and ensure that the iterate is primal feasible throughout all iterations. *Primal feasibility is ensured and dual feasibility is sought after* [15; 56; 98].
- **Dual** methods work with a dual iterate λ and ensure that the iterate is dual feasible throughout all iterations. *Dual feasibility is ensured and primal feasibility is sought after* [57; 124; 125].
- **Primal-dual** methods work with a primal-dual pair (x, λ). Neither primal nor dual feasibility of the iterates are ensured before termination [116; 126].
- **Parametric** methods are specialized for mpQPs and start with the optimal solution given a nominal parameter θ_0. The working set is then updated by using a homotopy [127] for the mpQP to obtain a solution for the current parameter θ [72; 128].

Commonly, primal and dual methods change $\mathcal{W}$ one element at a time, by either removing or adding an index to it. These are the methods considered in this thesis. For a comprehensive survey of the theoretical details underlying these types of active-set methods see, e.g., [129]. We will now introduce such a primal active-set algorithm.

3.3 A primal active-set algorithm

Before getting into the details of the steps performed in a primal active-set algorithm, we relate its workings in terms of the KKT conditions (3.3). In a primal active-set algorithm, primal feasibility (3.3b) of an iterate $x \in \mathbb{R}^n$ is maintained throughout all iterations. Simultaneously, the complementary slackness condition (3.3d) is enforced through a working set $\mathcal{W} \subseteq \mathbb{N}_m$. In particular, this is done by imposing that all equality constraints contained in $\mathcal{W}$ should hold with equality, i.e., $[A]_i x = [b]_i$, $\forall i \in \mathcal{W}$; implicitly, $\mathcal{W}$ also forces the dual variables of the constraints in its complement, denoted $\bar{\mathcal{W}} \triangleq \mathbb{N}_m \setminus \mathcal{W}$, to be fixed at zero, i.e., $[\lambda]_i = 0$, $\forall i \in \bar{\mathcal{W}}$. To summarize, both primal feasibility (3.3b) and complementary slackness (3.3d) are enforced throughout all iterations, which leave dual feasibility (3.3c) and stationarity (3.3a) to be sought after in each iteration.

The primal active-set algorithm that we consider here is given in Algorithm 3, and is described in detail below. In the algorithm, a subscript k denotes a variable's value at iteration k (e.g., x_k and $\mathcal{W}_k$ denote x and $\mathcal{W}$ at iteration k).

Remark 3.2. As is shown in [130], many active-set methods in the literature are mathematically equivalent, in the sense that they produce the same sequence of iterates before reaching the solution; Algorithm 3 belongs to this family. As such, Algorithm 3 is mathematically equivalent to the active-set algorithms presented in, for example, [15; 56; 98]. The differences between these active-set algorithms are numerical, for example, how systems of linear equations are solved. This equivalency makes the certification framework presented in Paper A in Part II particularly powerful.

Algorithm 3 (A primal active-set algorithm for solving (3.1) when $H \succ 0$)

```
Input: $x_0, \mathcal{W}_0, k = 0$
Output: $x^*, \lambda^*$
1:  repeat
2:      Compute $x_k^*$ by solving (3.9)
3:      if $Ax_k^* \leq b$ then                                          ▷ $x_k^*$ is primal feasible
4:          Compute $\lambda_k$ by solving (3.10)
5:          if $\lambda_k \geq 0$ then return $x^* \leftarrow x_k^*,\ \lambda^* \leftarrow \lambda_k$      ▷ Optimal solution found
6:          else  $l \leftarrow \operatorname{argmin}_{i \in \mathcal{W}_k} [\lambda_k]_i$
7:              $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{l\};\quad x_{k+1} \leftarrow x_k^*$
8:      else  $l \leftarrow \operatorname{argmin}_{i \in \bar{\mathcal{W}}_k : [A]_i x_k^* > [b]_i} \frac{[b]_i - [A]_i x_k}{[A]_i (x_k^* - x_k)}$     ▷ $x_k^*$ is **not** primal feasible
9:          $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{l\};\quad x_{k+1} \leftarrow x_k + \alpha_k^l (x_k^* - x_k)$
10:     $k \leftarrow k + 1$
```

Since primal feasibility should be maintained throughout all iterations, Algorithm 3 starts with an iterate x_0 that is primal feasible, i.e., $Ax_0 \leq b$, and a working set $\mathcal{W}_0$ that contains a subset of the active constraints at x_0, i.e, $\mathcal{W}_0 \subseteq \mathcal{A}(x_0)$. In each iteration of the algorithm, a constraint will either be added to or removed from the working set $\mathcal{W}_k$, unless the current iterate x_k is equal to the optimal solution x^*, in which case the algorithm terminates. We will now explain/motivate each step in the algorithm in detail.

(Step 2): An iteration always starts by computing the solution x_k^* to an EQP defined by the current working-set $\mathcal{W}_k$ of the form

$$\begin{aligned} x_k^* \triangleq \underset{x}{\text{argmin}} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & [A]_i x = [b]_i, \quad \forall i \in \mathcal{W}_k. \end{aligned} \tag{3.9}$$

The solution x_k^* to (3.9) is called a *constrained stationary point* (CSP), since it satisfies the stationarity condition (3.3a) and is "constrained" to be on the manifold defined by $\mathcal{W}_k$. We will for the time being consider the strictly convex case ($H \succ 0$), which ensures that x_k^* exists and is unique. In Section 3.3.1 we extend the algorithm to also handle the case when H is positive *semi*definite.

(Step 3): The aim in an iteration is, while maintaining primal feasibility, to decrease the objective function value by moving from the current iterate x_k to x_k^* along the line segment $L_k \triangleq \{x \in \mathbb{R}^n : x = x_k + \alpha(x_k^* - x_k), \alpha \in [0, 1]\}$. The largest possible decrease would be achieved by setting $x_{k+1} = x_k^*$, which is only possible if x_k^* is primal feasible, i.e., if $Ax_k^* \leq b$.

(Step 4): If x_k^* is primal feasible we, hence, set $x_{k+1} = x_k^*$, which means that the stationarity condition (3.3a) is satisfied. Since primal feasibility (3.3b) and complementary slackness (3.3d) are ensured to hold throughout all iterations, the only KKT condition remaining to be satisfied is dual feasibility (3.3c). To check dual feasibility, the dual variables are computed by solving

$$[A]_{\mathcal{W}_k}^T [\lambda_k]_{\mathcal{W}_k} = -Hx_k^* - f. \tag{3.10}$$

(Step 5): If $\lambda_k \geq 0$, dual feasibility is satisfied, i.e., all KKT-conditions are satisfied. Hence, the algorithm terminates with the global optimizer x_k^*.

(Step 6 & 7): Otherwise, if $\lambda_k \ngeq 0$, a constraint corresponding to a negative component of λ_k is removed from $\mathcal{W}$. There are different policies for selecting which negative component to remove, known as *selection rules*. The most common selection rule, which is used in Algorithm 3, is Dantzig's selection rule that selects the *most negative* component of λ_k, i.e.,

$$l = \min_{i \in \mathcal{W}_k} [\lambda_k]_i. \tag{3.11}$$

Selecting any negative component does, however, ensure the convergence of the algorithm and some alternative rules are surveyed in [131]. To summarize, when x_k^* is primal feasible but not dual feasible, the working set and the iterate is updated as

$$\mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{l\}, \quad x_{k+1} = x_k^*. \tag{3.12}$$

(Step 8 & 9): If x_k^* is *not* primal feasible ($Ax_k^* \not\leq b$), it follows from the convexity of $Ax \leq b$ that at least one constraint in $\bar{\mathcal{W}}_k$ becomes violated while moving from x_k to x_k^* along the line segment L_k. To retain primal feasibility, a violated constraint will be added to the working set. This constraint is determined by moving from x_k to x_k^* along L_k until primal feasibility is lost, i.e., until the *first* constraint in $\bar{\mathcal{W}}_k$ becomes active. Let α_k^i be the step length taken along L_k which activates constraint i, explicitly given by

$$[A]_i(x_k + \alpha_k^i(x_k^* - x_k)) = [b]_i \Leftrightarrow \alpha_k^i = \frac{[b]_i - [A]_i x_k}{[A]_i(x_k^* - x_k)}. \tag{3.13}$$

Hence, the index of the first violated constraint that becomes active, denoted l, can be determined by

$$l = \underset{i \in \bar{\mathcal{W}}_k : [A]_i x_k^* > [b]_i}{\operatorname{argmin}} \alpha_k^i, \tag{3.14}$$

and the working set and iterate are updated as

$$\mathcal{W}_{k+1} = \mathcal{W}_k \cup \{l\}, \quad x_{k+1} = x_k + \alpha_k^l(x_k^* - x_k). \tag{3.15}$$

(Step 10) After $\mathcal{W}$ has been updated, the above-mentioned steps are repeated until a dual feasible iterate (i.e., the global solution) has been found.

An example of Algorithm 3 applied to a strictly convex QP is given below.

Example 3.2: Primal active-set algorithm in action

Let us use Algorithm 3 to solve the QP in Example 3.1. We select the starting iterate $x_0 = (0, 0)$ (which is primal feasible) and the starting working set $\mathcal{W}_0 = \emptyset$.
Iteration 0: Solving (3.9) with $\mathcal{W} = \emptyset$ results in $x_0^* = -H^{-1}f = (-6, 5)$, i.e., the unconstrained minimum. Since $Ax_0^* \not\leq b$, x_0^* is not primal feasible (also evident from Figure 3.2) a constraint will be added to $\mathcal{W}$. By computing the step length until activation for each constraint in $\bar{\mathcal{W}}_0$ (constraint 1 and 2) we get $\alpha_0^1 = 0.41$ and $\alpha_0^2 = 0.52$. Since $\alpha_0^1 < \alpha_0^2$, constraint 1 is added to the working set and the iterate is updated as $x_1 = x_0 + 0.41(x_0^* - x_0)$. The initial iteration ends with $x_1 = (-2.47, 2.06)$ and $\mathcal{W}_1 = \{1\}$.
Iteration 1: Solving (3.9) with $\mathcal{W}_1 = \{1\}$ results in $x_1^* = (-1.71, 3.57)$, which is primal infeasible since it violates the remaining constraint not in $\mathcal{W}_1$ (constraint 2). The second constraint is, hence, added to the working set and, since the step length to activation of constraint 2 is $\alpha_1^2 = 0.62$, the iterate is updated as $x_2 = x_1 + 0.62(x_1^* - x_1)$. The iteration ends with $x_2 = (-2, 3)$ and $\mathcal{W}_2 = \{1, 2\}$.
Iteration 2: Solving (3.9) with $\mathcal{W}_2 = 1, 2$ trivially gives $x_2^* = x_2$ since the constrained set is a single point, which also means that $Ax_2^* = b \leq b$, i.e., x_2^* is primal feasible. Hence, either a constraint will be removed from $\mathcal{W}$ or the global optimum has been reached. Computing λ by (3.10) gives $\lambda_2 = (0.4, 1.2) \geq 0$, which is dual feasible, resulting in the algorithm terminating with the global optimum.

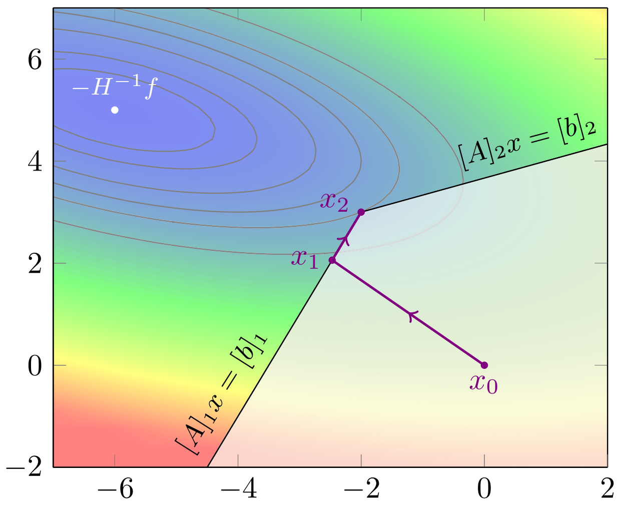


***Figure 3.2:** Active-set iterations*

The path taken before reaching optimality is illustrated in Figure 3.2. In conclusion we get $x^* = x_2^* = (-2, 3)$ and $\mathcal{A}^* = \mathcal{W}_2 = \{1, 2\}$. The working-set sequence to reach optimality was $\emptyset \rightarrow \{1\} \rightarrow \{1, 2\}$.

3.3.1 Extensions to semidefinite Hessians

Two problems that might arise when H is allowed to be singular is that the subproblems in (3.9) do: (i) not have a unique solution, (ii) are unbounded. Both uniqueness and unboundedness can be determined by properties of the so-called *reduced Hessian*.

Definition 3.3 (Reduced Hessian). The *reduced Hessian* given $\mathcal{W}_k$ is defined as $Z_k^T H Z_k$, where the matrix $Z_k \in \mathbb{R}^{n \times |\mathcal{W}_k|}$ is a full-rank matrix with columns spanning the null space of $[A]_{\mathcal{W}_k}$.

An interpretation of the reduced Hessian is as the Hessian for the resulting quadratic form if x is restricted to the subspace $\{x \in \mathbb{R}^n : [A]_i x = [b]_i, \forall i \in \mathcal{W}_k\}$.

Now, the uniqueness/existence of a solution to the EQP subproblem (3.9) can be determined by checking whether the reduced Hessian is nonsingular, formalized by the following lemma (for more details, see, e.g., Section 16.1 in [15]).

Lemma 3.2 (Existence of unique solution). *If the reduced Hessian $Z_k^T H Z_k \succ 0$ (i.e., is nonsingular) there exists a unique solution to the EQP subproblem* (3.9).

Proof: The main idea behind the proof is to transform the EQP in (3.9) to an unconstrained quadratic program. Let $\xi \in \mathbb{R}^n$ be any point that satisfies $[A]_{\mathcal{W}_k}\xi = [b]_{\mathcal{W}_k}$ (i.e., is a particular solution). Then any point in $x \in \mathbb{R}^n$ that satisfies the equality constraints $[A]_{\mathcal{W}_k}x = [b]_{\mathcal{W}_k}$ can be decomposed into $x = \xi + Z_k\tilde{x}$, where $\tilde{x} \in \mathbb{R}^{|\mathcal{W}_k|}$ (recall that Z_k is a full rank matrix with column vectors that span the null space of $[A]_{\mathcal{W}_k}$). Therefore, the objective function of subproblem (3.9) can be written as

$$\begin{aligned}\frac{1}{2}x^T Hx + f^T x &= \frac{1}{2}(\xi + Z_k\tilde{x})^T H(\xi + Z_k\tilde{x}) + f^T(\xi + Z_k\tilde{x})\\ &= \frac{1}{2}\tilde{x}^T Z_k^T HZ_k\tilde{x} + (f^T + \xi^T H)Z_k\tilde{x} + c\end{aligned}$$

with $c \triangleq \frac{1}{2}\xi^T H\xi + f^T\xi$. Solving the EQP in (3.9) is, hence, equivalent to solving the unconstrained problem

$$\underset{\tilde{x}}{\text{minimize}} \quad \frac{1}{2}\tilde{x}^T Z_k^T HZ_k\tilde{x} + (f^T + \xi^T H)Z_k^T\tilde{x} + c \tag{3.16}$$

which has a unique solution iff $Z_k^T HZ_k$ is nonsingular, i.e., if $Z_k^T HZ_k \succ 0$ (since Z_k has full rank and $H \succeq 0$). □

A consequence of Lemma 3.2 is that if the reduced Hessian remains nonsingular, the same type of iterations considered in Algorithm 3 can be performed without any modification. Problems only arise when the reduced Hessian becomes singular. Next we show that singularity cannot occur after a constraint is added to $\mathcal{W}$.

Lemma 3.3 (Reduced Hessian after an addition to $\mathcal{W}$). *If $Z_kHZ_k \succ 0$ and constraint l is added in iteration k of Algorithm 3, then $Z_{k+1}HZ_{k+1} \succ 0$.*

Proof: If constraint l is added in iteration k, the submatrix $[A]_{\mathcal{W}_{k+1}}$ will have one additional row compared with $[A]_{\mathcal{W}_k}$. Since the added row will be linearly independent to the rows in $[A]_{\mathcal{W}_k}$ (see, e.g., Section 16.5 in [15]), there exist null space bases that are related as $Z_k = \begin{pmatrix} Z_{k+1} & \tilde{z}\end{pmatrix}$ for some $\tilde{z} \in \mathbb{R}^n$ (and both Z_k and Z_{k+1} have full rank). Hence, the reduced Hessians are related as

$$Z_k^T HZ_k = \begin{pmatrix} Z_{k+1} & \tilde{z}\end{pmatrix}^T H\begin{pmatrix} Z_{k+1} & \tilde{z}\end{pmatrix} = \begin{pmatrix} Z_{k+1}^T HZ_{k+1} & Z_{k+1}H\tilde{z}\\ \tilde{z}^T HZ_{k+1} & \tilde{z}^T H\tilde{z}\end{pmatrix} \succ 0, \tag{3.17}$$

which implies, from Sylvester's criterion, that $Z_{k+1}^T HZ_{k+1} \succ 0$. □

Lemma 3.3 implies that the reduced Hessian only becomes singular after a constraint is removed from $\mathcal{W}$. In that case, the EQP in (3.9) does not have a unique solution and the iterate in the active-set algorithm has to be updated by other means. The following lemma gives guidance in how the iterate should be updated when the reduced Hessian becomes singular.

Lemma 3.4 (Unbounded search direction). *Let x_k satisfy $[A]_{\mathcal{W}_k} x_k = [b]_{\mathcal{W}_k}$ and $\tilde{p} \in \mathbb{R}^n$ be a solution to*

$$H\tilde{p} = 0, \quad [A]_{\mathcal{W}_k}\tilde{p} = 0, \quad f^T\tilde{p} < 0. \tag{3.18}$$

Then

(i) $[A]_{\mathcal{W}_k}(x_k + \alpha\tilde{p}) = [b]_{\mathcal{W}_k}, \forall\alpha \in \mathbb{R}$

(ii) $J(x_k + \alpha\tilde{p}) \to -\infty$ *as* $\alpha \to \infty$.

Proof: (i) Directly follows from $[A]_{\mathcal{W}_k}\tilde{p} = 0$ and $[A]_{\mathcal{W}_k} x_k = [b]_{\mathcal{W}_k}$. (ii) Evaluating the objective function J for any point $x_k + \alpha\tilde{p}$ gives

$$\begin{aligned} J(x_k + \alpha\tilde{p}) &= \frac{1}{2}(x_k + \alpha\tilde{p})H(x + \alpha\tilde{p}) + f^T(x_k + \alpha\tilde{p}) \\ &= \frac{1}{2}x_k^T H x_k + \alpha x_k^T H\tilde{p} + \frac{\alpha^2}{2}\tilde{p}^T H\tilde{p} + f^T x_k + \alpha f^T\tilde{p} \\ &= J(x_k) + \alpha x_k^T H\tilde{p} + \frac{\alpha^2}{2}\tilde{p}^T H\tilde{p} + \alpha f^T\tilde{p} \\ &= J(x_k) + \alpha f^T\tilde{p}, \end{aligned} \tag{3.19}$$

where $H\tilde{p} = 0$ has been used in the last equality. Now, since $f^T\tilde{p} < 0$ it follows that $\alpha f^T\tilde{p} \to -\infty$ when $\alpha \to \infty$ and, hence, $J(x_k + \alpha\tilde{p}) \to -\infty$ when $\alpha \to \infty$. □

Lemma 3.4 implies that if there exists a $\tilde{p}$ that satisfies (3.18), moving along the ray $x_k + \alpha\tilde{p}$, $\alpha > 0$ can decrease the objective function by an arbitrary amount while staying on the manifold defined by the working set $\mathcal{W}_k$. The following lemma ensures that there always exists a solution to (3.18) when the removal of a constraint in Algorithm 3 result in a singular reduced Hessian.

Lemma 3.5 (Existence of singular descent direction). *If the reduced Hessian becomes singular after the removal of a constraint l in iteration $k-1$ of Algorithm 3, there exists $\tilde{p} \in \mathbb{R}^n$ that solves* (3.18).

Proof: Let Z_k, again, be a matrix with columns that are a basis to the null space of $[A]_{\mathcal{W}_k}$. Then, since the reduced Hessian is singular, there exists p such that $Z_k^T H Z_k p = 0$. Because Z_k has full rank, it follows that $HZ_k p = 0$. Now let $\tilde{p} \triangleq Z_k p$. It then directly follows that $H\tilde{p} = 0$ and $[A]_{\mathcal{W}_k}\tilde{p} = 0$. What remains is to prove that $\tilde{p}$ also satisfies $f^T\tilde{p} < 0$. Since a constraint was removed in iteration $k-1$ the stationarity condition (3.3a) held for x^*_{k-1} and λ_{k-1}, i.e.,

$$Hx^*_{k-1} + [A]^T_{\mathcal{W}_k}[\lambda_{k-1}]_{\mathcal{W}_k} + [A]_l^T[\lambda_{k-1}]_l = -f, \tag{3.20}$$

where we have used that $[A]^T_{\mathcal{W}_{k-1}}[\lambda_{k-1}]_{\mathcal{W}_{k-1}} = [A]^T_{\mathcal{W}_k}[\lambda_{k-1}]_{\mathcal{W}_k} + [A]_l^T[\lambda_{k-1}]_l$ since $\mathcal{W}_{k-1} = \mathcal{W}_k \cup \{l\}$. Transposing (3.20) and multiplying with $\tilde{p}$ from the right then gives

$$\begin{aligned} f^T\tilde{p} &= -(Hx^*_{k-1} + [A]^T_{\mathcal{W}_k}[\lambda_{k-1}]_{\mathcal{W}_k} + [A]_l^T[\lambda_{k-1}]_l)^T\tilde{p} \\ &= -\left((x^*_{k-1})^T H\tilde{p} + [\lambda_{k-1}]^T_{\mathcal{W}_k}[A]_{\mathcal{W}_k}\tilde{p} + [\lambda_{k-1}]_l[A]_l\tilde{p}\right) \\ &= -[\lambda_{k-1}]_l[A]_l\tilde{p}, \end{aligned} \tag{3.21}$$

where $H\tilde{p} = 0$ and $[A]_{\mathcal{W}_k}\tilde{p} = 0$ have been used in the last equality. Now, since constraint l was removed in iteration $k-1$ we have that $[\lambda_{k-1}]_l < 0$. Moreover, $[A]_l\tilde{p} \neq 0$ since otherwise the reduced Hessian at iteration $k-1$ would be singular. Taken together, $-[\lambda_{k-1}]_l[A]_l\tilde{p} \neq 0$. Now all of the above arguments also hold for $-\tilde{p}$ which means that $f^T\tilde{p} < 0$ or $f^T(-\tilde{p}) < 0$. In conclusion, we can assume w.l.o.g. that $\tilde{p}$ was selected such that $f^T\tilde{p} < 0$ (otherwise we could just flip its sign). $\square$

To summarize the implications of Lemma 3.4 and 3.5: If, in Algorithm 3, the reduced Hessian becomes singular after the removal of a constraint, there always exists a direction $\tilde{p} \in \mathbb{R}^n$ that can decrease the objective function by an arbitrary amount, while staying primal feasible.

When deciding the step length α to move in the direction $\tilde{p}$, two different scenarios can occur. If there exists a constraint in $\bar{\mathcal{W}}_k$ that becomes active while moving along the ray $x_k + \alpha\tilde{p}$, $\alpha > 0$, the *first* constraint in $\bar{\mathcal{W}}_k$ that becomes active is added to $\mathcal{W}_k$ (similar to Steps 8 & 9 in Algorithm 3). Otherwise, if no constraint in $\bar{\mathcal{W}}_k$ becomes active while moving along the ray, it follows from Lemma 3.4 that the objective function can be made arbitrarily small while maintaining primal feasibility, resulting in an unbounded problem. Concretely, no inactive constraint will become activated if $[A]_{\bar{\mathcal{W}}_k}\tilde{p} \geq 0$.

The above-mentioned amendments to Algorithm 3 result in Algorithm 4.

Algorithm 4 (A primal active-set algorithm for solving (3.1) when $H \succeq 0$)

```
Input: $x_0$, $\mathcal{W}_0$ (such that $Z_0^T H Z_0 \succ 0$), $k = 0$
Output: $x^*$, $\lambda^*$
1: repeat
2:    Compute $x_k^*$ by solving (3.9)
3:    if (3.9) is unbounded then SINGULARITERATION
4:    else
5:       if $Ax_k^* \leq b$ then                                   ▷ $x_k^*$ is primal feasible
6:          Compute $\lambda_k$ by solving (3.10)
7:          if $\lambda_k \geq 0$ then return $x^* \leftarrow x_k^*$, $\lambda^* \leftarrow \lambda_k$        ▷ Optimal solution found
8:          else  $l \leftarrow \underset{i \in \mathcal{W}_k}{\operatorname{argmin}}\, [\lambda_k]_i$
9:                $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{l\}$;  $x_{k+1} \leftarrow x_k^*$
10:      else  $l \leftarrow \underset{i \in \bar{\mathcal{W}}_k : [A]_i x_k^* > [b]_i}{\operatorname{argmin}} \frac{[b]_i - [A]_i x_k}{[A]_i(x_k^* - x_k)}$          ▷ $x_k^*$ is not primal feasible
11:            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{l\}$;  $x_{k+1} \leftarrow x_k + \alpha_k^l(x_k^* - x_k)$
12:   $k \leftarrow k + 1$
13: procedure SINGULARITERATION
14:   Compute $\tilde{p}$ from (3.18)
15:   if $[A]_{\bar{\mathcal{W}}_k}\tilde{p} \geq 0$ then break unbounded
16:   else  $l \leftarrow \underset{i \in \bar{\mathcal{W}}_k : [A]_i \tilde{p} < 0}{\operatorname{argmin}} \frac{[b]_i - [A]_i x_k}{[A]_i \tilde{p}}$
17:         $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{l\}$;  $x_{k+1} \leftarrow x_k + \frac{[b]_l - [A]_l x_k}{[A]_l \tilde{p}} \tilde{p}$
```

3.4 From primal to dual active-set algorithms

When $H \succ 0$, one can instead of solving the QP in (3.1) solve its so-called *dual*

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \frac{1}{2}\lambda^T A^T H^{-1} A \lambda + (b - AH^{-1}f)^T \lambda \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned} \tag{3.22}$$

which is also a QP. It can be shown, see [132], that the solution λ^* to (3.22) satisfies the same KKT conditions as (3.1). After solving (3.22), the primal solution x^* to (3.1) can hence be retrieved from λ^* through the stationarity condition (3.3a) as

$$x^* = -H^{-1}(A^T \lambda^* + f). \tag{3.23}$$

The main idea behind *dual* active-set QP methods is to work with the dual QP in (3.22) instead of the primal QP in (3.1). Generally, the Hessian of the dual, $A^T H^{-1} A$, is positive semidefinite, making the extensions presented in Section 3.3.1 necessary. Hence, applying Algorithm 4 to the QP in (3.22) directly defines a dual active-set algorithm. This is concretized in Paper E.

One major advantage of working with the dual QP is that the constraints are simple nonnegativity constraints $\lambda \geq 0$, which, for example, makes it trivial to find a feasible starting iterate. Moreover, the computation of step lengths to constraints in $\bar{\mathcal{W}}_k$, performed in Step 10, simplifies to $\alpha_k^i = \frac{[\lambda_k]_i}{[\lambda_k^* - \lambda_k]_i}$ since $\lambda \geq 0 \Leftrightarrow A\lambda \leq b$ with $A = -I$ and $b = 0$.

In dual active-set algorithms, similar to primal active-set algorithms, a working set $\mathcal{W}$ ensures that the complementary slackness condition (3.3d) is satisfied throughout all iterations. Moreover, a point in which the stationarity condition (3.3a) is satisfied (a solution to a KKT system) is pursued in each iteration. The main difference between primal and dual active-set algorithms, in the context of the KKT conditions (3.3), is that dual feasibility is maintained throughout all iterations while primal feasibility is sought after. Hence, all iterates except the last one violate some constraint in (3.1).

3.5 Practical concerns

This chapter concludes with some important practical concerns for the active-set algorithms introduced in this chapter.

Warm starts

The QPs that are solved in two adjacent time steps in linear MPC are often very similar, since changing the parameter in (2.18) only perturbs f and b, while H and A remain constant. Moreover, a small perturbation of θ leads to small perturbations to f and b. Hence, if θ (i.e., the state) only changes slightly between two time steps, the solutions to the corresponding QPs are close. In such situations, the solution from the previously solved QP can be used to initialize the active-set algorithm when solving the next QP. This is known as *warm-starting* the solver and often reduces the computational effort. Conversely, initializing the solver without any prior knowledge is known as *cold-starting* the solver. Even though warm starts usually improve the average computational complexity for active-set algorithms, some care has to be taken since the worst-case computational complexity can be exacerbated [133].

An important advantage of active-set methods compared with, for example, interior-point methods are that they can easily be warm started; the optimal iterate and working set from a previously solved QP can be used directly in selecting the starting iterate and working set for another, similar, QP. In the context of general mpQPs, warm-starting *primal* active-set algorithms can be challenging since the perturbation of b (caused by the change in θ) might yield a solution to the previous problem infeasible for the new QP. In the context of linear MPC, however, it is often possible to obtain a primal feasible iterate even if b is perturbed. This follows from the constraints usually consisting of box constraints of the form $l \leq x \leq u$ (where l and u do *not* depend on θ) and more general constraints of the form $Ax \leq b$ (where b depends on θ) that are *softened* (see Section 2.3.2). Therefore, selecting any starting point x_0 between l and u (which is trivial) and initializing the slack ϵ_s for the soft constraints to a sufficiently high value to counteract the parametric perturbation of b leads to a primal feasible starting iterate.

In contrast to a primal active-set algorithm, warm-starting the dual active-set algorithm described in Section 3.4 after b and f have been perturbed is always trivial, since any nonnegative $\lambda \geq 0$ suffices (i.e., the constraints to the dual problem are not perturbed by a change in θ).

Early termination

An advantage of primal over dual active-set algorithms is that they retain primal feasibility throughout all iterations. Hence, a primal active-set algorithm can be terminated early and still provide an iterate that satisfies the constraints. The solutions after such early terminations will be suboptimal but can be "good enough" for the application at hand. This is often the case in MPC where a suboptimal solution that is close enough to the optimum might yield sufficient control performance and the resulting feedback law might still be stable [134]. Hence, using a primal active-set algorithm that is terminated early for linear MPC can reduce the computational complexity significantly, while still producing an acceptable control law.

In contrast, a dual active-set algorithm produces primal infeasible iterates all the way up until an optimum is found. Some constraints will, therefore, always be violated if the dual active-set algorithm is terminated early. Primal feasibility of the solution is crucial in the context of MPC since the constraints (especially the hard constraints) often encode physical limitations, such as actuator limits, which are impossible, or discouraged, to be exceeded in practice. Early-terminating dual active-set algorithms are therefore not viable in linear MPC.

There are, however, MPC applications where early-terminating dual active-set algorithms are very useful, namely, in MPC of hybrid systems [135], where some states take discrete values. In these applications, the optimization problems that are solved in each time step are mixed-integer quadratic programs (MIQPs), which are usually solved through branch-and-bound, where a sequence of QPs are solved. In branch-and-bound, lower bounds on optimal objective function values can be used to reduce the number of QPs that need to be solved, which greatly decreases the computational complexity [136]. Such lower bounds are readily available in dual active-set algorithms since they monotonically increase the objective function value. Hence, if the objective function value exceeds a certain limit (which corresponds to the objective function value of a known feasible solution) the branch-and-bound method can immediately abort the solution process of this QP and prune the corresponding branch.

Numerical stability

Round-off errors can often lead to cycling of active-set algorithms [137], especially if round-off errors affect the checks for primal feasibility and dual feasibility which are done in Steps 5 and 7 in Algorithm 4, respectively. One way of ensuring that the algorithm terminates in finite time, despite round-off errors, is to modify the algorithm by incorporating anti-cycling schemes, e.g., the ones presented in [137; 138].

Another approach to reduce the likelihood of cycling, which does not require any modification to the optimization algorithm, is to regularize the problem which is solved (since ill-conditioned problems exacerbate round-off errors). In practice, ill-conditioned Hessians are often regularized by adding a positive diagonal matrix, i.e., $H \to H + \epsilon I$ for some $\epsilon > 0$. A drawback of such regularizations is, however, that the solution also is perturbed. An alternative way of regularizing the QP, which does not perturb the solution, is to perform so called *proximal-point* iterations, by iteratively solving QPs of the form

$$\begin{aligned} x_{k+1} = \underset{x}{\operatorname{argmin}} \quad & \frac{1}{2} x^T (H + \epsilon I) x + (f - \epsilon x_k)^T x \\ \text{subj. to} \quad & Ax \leq b. \end{aligned} \tag{3.24}$$

It can then be shown (see, e.g., Theorem 10.28 in [139]) that $x_k \to x^*$ as $k \to \infty$ if such iterations are performed. Moreover, a larger ϵ leads to a better conditioned Hessian, but also to more iterations before convergence (again, see, e.g., Theorem 10.28 in [139] for the convergence rate).

Remark 3.3. Outer proximal-point iterations can be used to improve the numerical stability of *any* QP method. However, active-set methods are especially suited to be used in conjunction with proximal-point iterations because of their warm-starting capabilities.

Solving KKT systems

Most of the computational load in Algorithm 4 is in solving the KKT systems in Step 2. Efficiently solving these KKT systems is, hence, essential for the active-set algorithm to be practically viable. Solving them from scratch in each iteration would lead to a large computational cost. Fortunately, the systems can be solved very efficiently by leveraging that only a single element is added/removed from $\mathcal{W}$ at a time, which makes the KKT system between two iterations similar. For example, the KKT system to be solved in iteration k is

$$\begin{pmatrix} H & [A]_{\mathcal{W}_k}^T \\ [A]_{\mathcal{W}_k} & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -f \\ [b]_{\mathcal{W}_k} \end{pmatrix}. \tag{3.25}$$

If we assume that constraint m is added, the next KKT system to solve is

$$\begin{pmatrix} H & [A]_{\mathcal{W}_k}^T & [A]_m^T \\ [A]_{\mathcal{W}_k} & 0 & 0 \\ [A]_m & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \lambda^+ \end{pmatrix} = \begin{pmatrix} -f \\ [b]_{\mathcal{W}_k} \\ [b]_m \end{pmatrix}. \tag{3.26}$$

That is, only one extra equation and one extra optimization variable $\lambda^+ \in \mathbb{R}$ have been added. Hence, instead of solving the KKT system from scratch, a factorization of the matrix in the left-hand side (the *KKT matrix*) can be computed and updated through low-rank updates. Some standard ways of factorizing the matrix are through LDL^T or QR factorizations (see, e.g., [140]). Moreover, in the context of linear MPC with a sparse formulation, the KKT matrix can be factorized using the Riccati factorization [53]. For efficient low-rank updates of the LDL^T and QR factorization see, e.g., [141]; for low-rank updates of the Riccati factorization see [142].

In the complexity certification framework in Paper A, we abstract away the numerical method that is used for solving the KKT systems. If the working set sequence is known, which is what is determined in Paper A, we know *exactly* which sequences of KKT systems need to be solved. Hence, given any particular way of factorizing the KKT matrix we can determine how many floating-point operations are needed to solve the sequences of KKT systems.

In the dual solver proposed in Paper E, we solve KKT systems by first eliminating the equality constraints imposed by the working set, and then we factorize the reduced system with an LDL^T factorization. This factorization is then updated with rank-1 updates.

4 Contributions and outlook

We conclude Part I by summarizing the main contributions presented in Part II, both broadly and in detail; furthermore, we give some future research directions to extend/improve the results therein.

4.1 Summary of contributions

Broadly, the contributions in Part II improve the *reliability* and *applicability* of active-set methods in real-time MPC applications. This is done through:

(i) A **certification framework** for active-set methods that determines their worst-case number of iterations, floating-point operations, and execution time (Paper A & B); this framework is further extended in Paper C & D to account for numerical aspects.

(ii) An **efficient active-set solver** (Paper E) that is covered by the certification framework. In Paper F, we exemplify how the solver can be tailored for the specific application at hand by using the certification framework.

The certification framework improves the reliability of active-set methods in MPC applications, since it enables *a priori* guarantees that the available hardware is sufficient for the application at hand. This is particularly important for MPC of hard real-time systems, where hard deadlines need to be met.

The proposed solver improves the applicability of active-set methods in real-time MPC applications, since it efficiently solves the type of problems that commonly arise in such applications. Moreover, the solver is easy to implement, is numerically stable, and can easily be warm started, all of which are favourable properties for real-time MPC solvers. On top of this, it is covered by the proposed certification framework, which enables hard real-time guarantees.

Concretely, the main contributions of each paper in Part II are listed below.

Paper A

(A1) For a given multi-parametric (positive *semi*-definite) quadratic program, we propose a method that determines *exactly* which sequence of linear system of equations a primal active-set method needs to solve before termination, for *any* parameter of interest. This, in turn, enables *exact* bounds on the required number of iterations and/or floating-point operations.

(A2) We unify the complexity certification methods for active-set methods proposed in [16–18] in a single framework.

(A3) By considering several mpQPs originating from MPC problems, we show how the proposed framework can be used to, for example, compare the problem-specific performance of primal and dual active-set methods.

Paper B

(B1) We introduce the concept of *execution-time equivalent covers* for programs with parametrized inputs.

(B2) We show how the framework in Paper A can be used to generate execution-time equivalent covers; specifically, this applies to programs that realize active-set methods covered by the framework in Paper A (e.g., the ones in [15; 56; 57] and Paper E).

(B3) We propose a method that determines the *exact* worst-case execution time (WCET) of programs that realize any active-set method covered by the framework in Paper A.

Paper C

(C1) We extend the framework in Paper A to be able to account for numerical errors. The extension builds on a three-step approach consisting of: (i) *lifting* the parameter space to include numerical errors; (ii) *partitioning* the parameter space based on the solver's behavior; (iii) *projecting* down the new regions onto the nominal parameter space.

(C2) We give an abstract representation of the certification methods covered by the framework in Paper A (e.g. [16–18]) and similar methods (e.g., [115]), in terms of parameter-dependent finite automatons.

Paper D

(D1) We propose a method that, for a given multi-parametric quadratic program, determines which sequence of regularized QPs needs to be solved when proximal-point iterations are performed, for *any* parameter of interest.

(D2) We combine the proposed certification method with the framework in Paper A to be able to give bounds on the total number of inner iterations or the total number of floating-point operations when proximal-point iterations are used.

Paper E

(E1) We show how favourable properties of the QP solver in [23] can be retained without transforming QPs into nonnegative least-squares problems. This leads to several improvements: (i) direct resusability of matrix factors for warm starts; (ii) improved numerical stability; (iii) improved efficiency by reducing intermediate computations stemming from the nonnegative least-square reformulation.

(E2) Based on (E1), we propose an efficient and simple dual active-set QP solver that is based on recursive low-rank updates to an LDL^T factorization. The solver is covered by the framework proposed in Paper A, which is important from a real-time perspective.

(E3) We provide an open-source, library-free, and high-performing C implementation (available at `https://github.com/darnstrom/daqp`) of the proposed solver. This implementation is shown to be capable of outperforming state-of-the-art solvers on small/medium-sized QPs arising in real-time MPC applications.

Paper F

(F1) We show how the framework in Paper A can be used to certify the complexity of parametric warm starts for active-set methods. This can be seen as extending the ideas presented in [18] from multi-parametric *linear* programs to multi-parametric *quadratic* programs.

(F2) We propose a method for generating parametric warm starts that does not require any regions to be explicitly stored. Instead, the regions are implicitly stored in the problem data of the multi-parametric quadratic program. As a result, the overhead of the proposed semi-explicit scheme is minor, both in terms of computations and memory.

4.2 Future work

In addition to the future research directions suggested in the publications in Part II, we give three general research directions to extend/improve the work presented in this thesis.

Linear parameter-varying systems The complexity certification framework in Paper A, along with its subsequent extensions, operates on given multi-parametric quadratic programs. In particular, it requires the Hessian and constraint matrix to be *parameter independent*. An interesting future research direction is to try to extend the framework to allow for the Hessian and constraint matrix to be parameter dependent. Specifically, this would allow the complexity certification of MPC of linear parameter-varying (LPV) systems, and more broadly to nonlinear MPC. To expect that the *exact* analysis from Paper A directly carries over is naive, but fairly tight complexity bounds might still be possible. One way forward is to apply the lift-partition-project scheme proposed in Paper C, with the perturbation originating from parameter-independent approximations of the Hessian and constraint matrix instead of numerical errors.

Optimizing the optimizer To "feedback" the certificates produced by the complexity certification methods proposed in this thesis to tailor solvers is another interesting research direction. Although this was partly done in Paper F, it only scratched the surface of possible optimizations. For example, custom selection rules for adding/removing constraints that are *optimal* for a particular application could be developed with the aid of the certification framework. Moreover, the solver can be tweaked in ways that improve the performance for a particular problem that might generally not even guarantee convergence. Paper B introduces even more possibilities for optimization, including hardware selection, compiler options, and explicit trade-offs between required memory and computations.

High-performance computing The method in Paper A is highly suitable for parallelization. Hence, it has the potential for being applied to larger MPC problems than the ones considered in this thesis. A straightforward way to parallelize the algorithm is to use domain decomposition; that is, by splitting the parameter space into regions and then apply the certification method for each region on separate workers. A more sophisticated way is to dynamically allocate regions to workers as the parameter space is made finer by the partitioning described in Paper A.

Bibliography

[1] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[2] T. Samad, M. Bauer, S. Bortoff, S. Di Cairano, L. Fagiano, P. F. Odgaard, R. R. Rhinehart, R. Sánchez-Peña, A. Serbezov, F. Ankersen, P. Goupil, B. Grosman, M. Heertjes, I. Mareels, and R. Sosseh, "Industry engagement with control research: Perspective and messages," *Annual Reviews in Control*, vol. 49, pp. 1–14, 2020.

[3] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] S. Di Cairano and I. V. Kolmanovsky, "Real-time optimization and model predictive control for aerospace and automotive applications," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 2392–2409.

[5] L. Del Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky, *Automotive model predictive control: models, methods and applications*. Springer, 2010, vol. 402.

[6] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.

[7] J. Rodriguez, C. Garcia, A. Mora, F. Flores-Bahamonde, P. Acuna, M. Novak, Y. Zhang, L. Tarisciotti, S. A. Davari, Z. Zhang *et al.*, "Latest advances of model predictive control in electrical drives—Part I: Basic concepts and advanced strategies," *IEEE Transactions on Power Electronics*, vol. 37, no. 4, pp. 3927–3942, 2021.

[8] J. Rodriguez, C. Garcia, A. Mora, S. A. Davari, J. Rodas, D. F. Valencia, M. Elmorshedy, F. Wang, K. Zuo, L. Tarisciotti *et al.*, "Latest advances of model predictive control in electrical drives—Part II: Applications and benchmarking with classical control methods," *IEEE Transactions on Power Electronics*, vol. 37, no. 5, pp. 5047–5061, 2021.

[9] I. McInerney, G. A. Constantinides, and E. C. Kerrigan, "A survey of the implementation of linear model predictive control on FPGAs," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 381–387, 2018.

[10] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 2014.

[11] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 194–13 209, 2017.

[12] T. A. Johansen, "Toward dependable embedded model predictive control," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1208–1219, 2017.

[13] D. Arnström, "On complexity certification of active-set QP methods with applications to linear MPC," Licentiate Thesis, Linköping University Electronic Press, 2021.

[14] D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2758–2770, 2022.

[15] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[16] D. Arnström and D. Axehill, "Exact complexity certification of a standard primal active-set method for quadratic programming," in *IEEE 58th Conference on Decision and Control (CDC)*, Dec 2019, pp. 4317–4324.

[17] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transactions on Automatic Control*, vol. 62, pp. 6094–6109, 2017.

[18] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1524–1534, 07 2011.

[19] D. Arnström, D. Broman, and D. Axehill, "Exact worst-case execution-time analysis for implicit model predictive control," *arXiv preprint arXiv:2304.11576*, 2023, submitted.

[20] D. Arnström and D. Axehill, "Lift, partition, and project: Parametric complexity certification of active-set QP methods in the presence of numerical errors," in *IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4381–4387.

[21] D. Arnström, A. Bemporad, and D. Axehill, "Complexity certification of proximal-point methods for numerically stable quadratic programming," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1381–1386, 2021.

[22] D. Arnström, A. Bemporad, and D. Axehill, "Exact complexity certification of a nonnegative least-squares method for quadratic programming," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 1036–1041, 2020.

[23] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1111–1116, 2015.

[24] A. Bemporad, "A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 525–531, 2017.

[25] D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL^T updates," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4362–4369, 2022.

[26] D. Arnström and D. Axehill, "Semi-explicit linear MPC using a warm-started active-set QP algorithm with exact complexity guarantees," in *IEEE 60th Conference on Decision and Control (CDC)*, 2021, pp. 2557–2562.

[27] D. Arnström and D. Axehill, "Exact complexity certification of a standard early-terminating primal active-set method for quadratic programming," in *Proceedings of the 2020 IFAC World Congress*, 2020.

[28] D. Arnström, A. Bemporad, and D. Axehill, "A linear programming method based on proximal-point iterations with applications to multi-parametric programming," *IEEE Control Systems Letters*, vol. 6, pp. 2066–2071, 2021.

[29] S. Shoja, D. Arnström, and D. Axehill, "Overall complexity certification of a standard branch and bound method for mixed-integer quadratic programming," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 4957–4964.

[30] S. Shoja, D. Arnström, and D. Axehill, "Exact complexity certification of a standard branch and bound method for mixed-integer linear programming," in *IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 6298–6305.

[31] D. Arnström and D. Axehill, "BnB-DAQP: a mixed-integer QP solver for embedded applications," in *Proceedings of the 2023 IFAC World Congress*, 2023.

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

[33] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems.* Cambridge University Press, 2017.

[34] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO: Basic linear algebra subroutines for embedded optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 44, no. 4, pp. 1–30, 2018.

[35] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6548–6554, 2020.

[36] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, pp. 1–27, 2012.

[37] D. Axehill, "Controlling the level of sparsity in MPC," *Systems & Control Letters*, vol. 76, pp. 1–7, 2015.

[38] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[39] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multiparametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.

[40] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari, "Computation of the constrained infinite time linear quadratic regulator," *Automatica*, vol. 40, no. 4, pp. 701–708, 2004.

[41] C. N. Jones and M. Morrari, "Multiparametric linear complementarity problems," in *IEEE 45th Conference on Decision and Control (CDC).* IEEE, 2006, pp. 5687–5692.

[42] A. Gupta, S. Bhartiya, and P. Nataraj, "A novel approach to multiparametric quadratic programming," *Automatica*, vol. 47, no. 9, pp. 2112–2117, 2011.

[43] R. Oberdieck, N. A. Diangelakis, and E. N. Pistikopoulos, "Explicit model predictive control: A connected-graph approach," *Automatica*, vol. 76, pp. 103–112, 2017.

[44] P. Ahmadi-Moshkenani, T. A. Johansen, and S. Olaru, "Combinatorial approach toward multiparametric quadratic programming based on characterizing adjacent critical regions," *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3221–3231, 2018.

[45] M. Herceg, C. N. Jones, M. Kvasnica, and M. Morari, "Enumeration-based approach to solving parametric linear complementarity problems," *Automatica*, vol. 62, pp. 243–248, 2015.

[46] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 502–510.

[47] R. Oberdieck, N. A. Diangelakis, M. M. Papathanasiou, I. Nascu, and E. N. Pistikopoulos, "Pop–parametric optimization toolbox," *Industrial & Engineering Chemistry Research*, vol. 55, no. 33, pp. 8979–8991, 2016.

[48] A. Bemporad, "Hybrid Toolbox - User's Guide," 2004, http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox.

[49] A. Bemporad, "Explicit model predictive control," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2019, pp. 1–7.

[50] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit MPC," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1878–1883, 2011.

[51] C. N. Jones and M. Morari, "Polytopic approximation of explicit model predictive controllers," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2542–2553, 2010.

[52] N. A. Nguyen, M. Gulan, S. Olaru, and P. Rodriguez-Ayerbe, "Convex lifting: Theory and control applications," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1243–1258, 2018.

[53] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[54] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2010.

[55] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE conference on decision and control (CDC)*. IEEE, 2012, pp. 668–674.

[56] R. Fletcher, "A general quadratic programming algorithm," *IMA Journal of Applied Mathematics*, vol. 7, no. 1, pp. 76–91, 1971.

[57] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, pp. 1–33, 9 1983.

[58] 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: IFAC-Affiliated Journal*, vol. 18, no. 8, pp. 816–830, 2008.

[59] D. Axehill and A. Hansson, "A dual gradient projection quadratic programming algorithm tailored for model predictive control," in *IEEE 47th Conference on Decision and Control (CDC)*. IEEE, 2008, pp. 3057–3064.

[60] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *IEEE 48th Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference*. IEEE, 2009, pp. 7387–7393.

[61] P. Giselsson, "Improved fast dual gradient methods for embedded model predictive control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2303–2309, 2014.

[62] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.

[63] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, pp. 1–36, 2020.

[64] Y. Pu, M. N. Zeilinger, and C. N. Jones, "Fast alternating minimization algorithm for model predictive control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 980–11 986, 2014, 19th IFAC World Congress.

[65] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: A proximal augmented lagrangian method for nonconvex quadratic programs," *Mathematical Programming Computation*, vol. 14, no. 3, pp. 497–541, 2022.

[66] A. T. J. C. Antoine Bambade, Sarah El-Kazdadi, "PROX-QP: Yet another quadratic programming solver for robotics and beyond," in *RSS 2022 - Robotics: Science and Systems*, New York, United States, 2022. [Online]. Available: https://hal.inria.fr/hal-03683733

[67] J. Renegar, *A Mathematical View of Interior-Point Methods in Convex Optimization*. Society for Industrial and Applied Mathematics, 2001. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718812

[68] S. J. Wright, *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

[69] Y. Nesterov, *Introductory lectures on convex optimization : a basic course.*, ser. Applied optimization: 87. Kluwer Acad. Publ., 2004.

[70] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[71] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 1, pp. 58–81, 2003.

[72] 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*, vol. 6, no. 4, pp. 327–363, 2014.

[73] F. Ullmann, "FiOrdOs: A Matlab toolbox for C-code generation for first order methods," Ph.D. dissertation, ETH Zürich (Master's thesis), 2011.

[74] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 3071–3076.

[75] S. D. Cairano, M. Brand, and S. A. Bortoff, "Projection-free parallel quadratic programming for linear model predictive control," *International Journal of Control*, vol. 86, no. 8, pp. 1367–1385, 2013.

[76] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 289–329, 2015.

[77] J. A. Andersson and J. B. Rawlings, "Sensitivity analysis for nonlinear programming in CasADi," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 331–336, 2018, 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[78] N. Saraf and A. Bemporad, "A bounded-variable least-squares solver based on stable QR updates," *IEEE Transactions on Automatic Control*, 2019.

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

[80] A. G. Pandala, Y. Ding, and H. W. Park, "qpSWIFT: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.

[81] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

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

[83] K. Cheshmi, D. M. Kaufman, S. Kamil, and M. M. Dehnavi, "NASOQ: numerically accurate sparsity-oriented QP solver," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 96–1, 2020.

[84] J. Frey, S. Di Cairano, and R. Quirynen, "Active-set based inexact interior point qp solver for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6522–6528, 2020, 21st IFAC World Congress.

[85] D. Liao-McPherson and I. Kolmanovsky, "FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming," *Automatica*, vol. 113, p. 108801, 2020.

[86] R. Schwan, Y. Jiang, D. Kuhn, and C. N. Jones, "PIQP: A proximal interior-point quadratic programming solver," *arXiv preprint arXiv:2304.00290*, 2023.

[87] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[88] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[89] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

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

[91] D. Liao-McPherson, M. M. Nicotra, and I. Kolmanovsky, "Time-distributed optimization for real-time model predictive control: Stability, robustness, and constraint satisfaction," *Automatica*, vol. 117, p. 108973, 2020.

[92] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[93] P. Listov and C. Jones, "PolyMPC: An efficient and extensible tool for real-time nonlinear model predictive tracking and path following for fast mechatronic systems," *Optimal Control Applications and Methods*, vol. 41, no. 2, pp. 709–727, 2020.

[94] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented lagrangian approach (GRAMPC)," *Optimization and Engineering*, vol. 20, pp. 769–809, 2019.

[95] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE 56th Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[96] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation.* Springer Science & Business Media, 1998.

[97] V. Klee and G. J. Minty, "How good is the simplex algorithm," *Inequalities*, vol. 3, no. 3, pp. 159–175, 1972.

[98] G. B. Dantzig, *Linear programming and extensions.* Princeton University Press, 1963.

[99] P. E. Gill and E. Wong, "Methods for convex and general quadratic programming," *Mathematical Programming Computation*, vol. 7, no. 1, pp. 71–112, 2015.

[100] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[101] L. G. Khachiyan, "A polynomial algorithm in linear programming," in *Doklady Akademii Nauk*, vol. 244, no. 5. Russian Academy of Sciences, 1979, pp. 1093–1096.

[102] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984, pp. 302–311.

[103] L. McGovern and E. Feron, "Requirements and hard computational bounds for real-time optimization in safety-critical control systems," vol. 3, 1998, pp. 3366–3371 vol.3.

[104] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[105] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.

[106] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[107] Y. Drori and M. Teboulle, "Performance of first-order methods for smooth convex minimization: a novel approach," *Mathematical Programming*, vol. 145, no. 1, pp. 451–482, 2014.

[108] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Smooth strongly convex interpolation and exact worst-case performance of first-order methods," *Mathematical Programming*, vol. 161, no. 1, pp. 307–345, 2017.

[109] E. De Klerk, F. Glineur, and A. B. Taylor, "Worst-case convergence analysis of inexact gradient and Newton methods through semidefinite programming performance estimation," *SIAM Journal on Optimization*, vol. 30, no. 3, pp. 2053–2082, 2020.

[110] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[111] 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 *IEEE 50th Conference on Decision and Control (CDC)*. IEEE, 2011, pp. 5223–5229.

[112] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *IEEE 51st Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 3165–3170.

[113] Y. Pu, M. N. Zeilinger, and C. N. Jones, "Complexity certification of the fast alternating minimization algorithm for linear MPC," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 888–893, 2017.

[114] G. Cimini, D. Bernardini, S. Levijoki, and A. Bemporad, "Embedded model predictive control with certified real-time optimization for synchronous motors," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 893–900, 2021.

[115] G. Cimini and A. Bemporad, "Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs," *Automatica*, vol. 100, pp. 29–37, 2019.

[116] K. Kunisch and F. Rendl, "An infeasible active set method for quadratic problems with simple bounds," *SIAM Journal on Optimization*, vol. 14, pp. 35–52, 01 2003.

[117] S. Vichik and F. Borrelli, "Solving linear and quadratic programs with an analog circuit," *Computers & Chemical Engineering*, vol. 70, pp. 160–171, 2014.

[118] L. Ljung, "System identification," *Wiley encyclopedia of electrical and electronics engineering*, pp. 1–19, 1999.

[119] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

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

[121] A. Zheng and M. Morari, "Stability of model predictive control with mixed constraints," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1818–1823, 1995.

[122] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Traces and emergence of nonlinear programming*. Springer, 2014, pp. 247–258.

[123] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[124] M. J. D. Powell, "On the quadratic programming algorithm of Goldfarb and Idnani," in *Mathematical Programming Essays in Honor of George B. Dantzig Part II.* Springer, 1985, pp. 46–61.

[125] R. A. Bartlett and L. T. Biegler, "QPSchur: a dual, active-set, schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, no. 1, pp. 5–32, 2006.

[126] M. Hintermüller, K. Ito, and K. Kunisch, "The primal-dual active set strategy as a semismooth Newton method," *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 865–888, 2002.

[127] E. L. Allgower and K. Georg, *Numerical continuation methods: an introduction.* Springer Science & Business Media, 2012, vol. 13.

[128] M. J. Best, "An algorithm for the solution of the parametric quadratic programming problem," in *Applied Mathematics and Parallel Computing.* Springer, 1996, pp. 57–76.

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

[130] M. J. Best, "Equivalence of some quadratic programming algorithms," *Mathematical Programming*, vol. 30, no. 1, p. 71, 1984.

[131] T. Terlaky and S. Zhang, "Pivot rules for linear programming: a survey on recent theoretical developments," *Annals of Operations Research*, vol. 46, no. 1, pp. 203–233, 1993.

[132] W. S. Dorn, "Duality in quadratic programming," *Quarterly of Applied Mathematics*, vol. 18, no. 2, pp. 155–162, 1960.

[133] M. Herceg, C. Jones, and M. Morari, "Dominant speed factors of active set methods for fast MPC," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 608–627, 2015.

[134] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.

[135] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[136] R. Fletcher and S. Leyffer, "Numerical experience with lower bounds for MIQP branch-and-bound," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 604–616, 1998.

[137] R. Fletcher, "Resolving degeneracy in quadratic programming," *Annals of Operations Research*, vol. 46, no. 2, pp. 307–334, 1993.

[138] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "A practical anti-cycling procedure for linearly constrained optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 437–474, 1989.

[139] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[140] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, vol. 3.

[141] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," *Mathematics of computation*, vol. 28, no. 126, pp. 505–535, 1974.

[142] I. Nielsen and D. Axehill, "Low-rank modifications of Riccati factorizations for model predictive control," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 872–879, 2017.

Part II

Publications

Publications

PhD Dissertations
Division of Automatic Control
Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

FACULTY OF SCIENCE AND ENGINEERING

Linköping Studies in Science and Technology. Dissertations No. 2324
Department of Electrical Engineering

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se

LINKÖPING
UNIVERSITY