On Complexity Certification of Active-Set QP Methods with Applications to Linear MPC

Daniel Arnström
On Complexity Certification of Active-Set QP Methods with Applications to Linear MPC

Daniel Arnström
This is a Swedish Licentiate's Thesis.

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree. A Doctor's Degree comprises 240 ECTS credits (4 years of full-time studies). A Licentiate's degree comprises 120 ECTS credits, of which at least 60 ECTS credits constitute a Licentiate's thesis.
Abstract

In model predictive control (MPC) an optimization problem has to be solved at each time step, which in real-time applications makes it important to solve these efficiently and to have good upper bounds on worst-case solution time. Often for linear MPC problems, the optimization problem in question is a quadratic program (QP) that depends on parameters such as system states and reference signals. A popular class of methods for solving such QPs is active-set methods, where a sequence of linear systems of equations is solved.

The primary contribution of this thesis is a method which determines which sequence of subproblems a popular class of such active-set algorithms need 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, worst-case bounds on how many iterations, floating-point operations and, ultimately, the maximum solution time, these active-set algorithms require to compute a solution can be determined, which is of importance when, e.g, linear MPC is used in safety-critical applications.

After establishing this complexity certification method, its applicability is extended by showing how it can be used indirectly to certify the complexity of another, efficient, type of active-set QP algorithm which reformulates the QP as a nonnegative least-squares method.

Finally, the proposed complexity certification method is extended further to situations when enhancements to the active-set algorithms are used, namely, when they are terminated early (to save computations) and when outer proximal-point iterations are performed (to improve numerical stability).
Populärvetenskaplig sammanfattning

En populär reglerstrategi för att styra komplexa system är modellprediktiv reglering (MPC), som fattar styrbeslut genom att lösa ett matematiskt optimeringsproblem. Således behöver ett optimeringsproblem lösas varje gång ett nytt styrbeslut fattas, vilket för snabba system kan betyda att tusentals optimeringsproblem behöver lösas varje sekund. Dessa optimeringsproblem måste kunna lösas inom den ofta snäva tidsrymden mellan då två styrbeslut ska fattas för att kunna garantera ett säkert och förutsägbart beteende för det styrda systemet, vilket betyder att värstafallsgränser på tidsåtgång för att lösa möjliga optimeringsproblem är av stor vikt.

I den här avhandlingen presenteras metoder som kan användas för att beräkna sådana värstafallgränser. Huvudfokus är på aktivmängdalgoritmier, en klass av numeriska optimeringsmetoder som ofta används för att lösa en vanlig typ av optimeringsproblem (kvadratiska optimeringsproblem) som måste lösas när MPC används för att styra system som kan modelleras med linjära modeller. Mer specifikt presenteras metoder som bestämmer exakt vilken sekvens av delproblem (linjära ekvationssystem) som populära aktivmängdalgoritmier kommer att behöva lösas, för alla möjliga kvadratiska optimeringsproblem som kan uppstå när en viss MPC-regulator används. Först presenteras ett övergripande ramverk för att bestämma värstafallsgränser för aktivmängdalgoritmier. Detta ramverk utökas sedan för att ta hänsyn till vanliga praktiska tillägg som görs till aktivmängdalgoritmier.

Metoderna som presenteras i denna avhandling kan således användas för att erhålla garantier på maximal tidsåtgång som kommer att krävs för att beräkna ett styrbeslut när MPC används, vilket, exempelvis, kan användas för att försäkra att hårdvaran som MPC-regulatorn är implementerad på är tillräcklig och inte överbelastas.
Acknowledgments

First and foremost, I want to thank my supervisor Daniel Axehill for your guidance and support over the past years. Thank you for your endless enthusiasm and for sharing my penchant for compiling never-ending lists of research ideas. Moreover, I am grateful for my co-supervisor Anders Hansson who was the lecturer in the first courses I took in automatic control and optimal control. Thank you for sparking my interest.

I also want to express my appreciation to all colleagues at the Automatic Control group: To all fellow PhD students, thank you for your camaraderie. To all the senior staff, thank you for sharing your experience. Moreover, special thanks to the head of division Martin Enqvist for cultivating a welcoming work environment and for constantly striving to improve it, to Ninna Stensgård who is always ready to help with administrative tasks, and to Kristoffer Bergman, Robin Forsling, Anton Kullberg, Fredrik Ljungberg and Shamisa Shoja for their help in proofreading parts of this thesis.

I also want to acknowledge the financial support from the Swedish Research Council (VR) which made the research presented in this thesis possible.

Finally, even though words can not fully capture my gratitude, I want to thank my family for their unconditional support in all of my endeavours: To my mother, for your sisu. To my father, for your kindness. To my brother, for your companionship.

Linköping, February 2021
Daniel Arnström
I Background

1 Introduction
   1.1 Background and motivation ................................. 3
   1.2 Thesis outline ............................................ 5

2 Model Predictive Control ........................................ 9
   2.1 Preliminaries ............................................... 10
   2.2 Optimal control ............................................ 11
   2.3 Linear MPC .................................................. 13
      2.3.1 Multi-parametric Quadratic Programming ............. 15
      2.3.2 Extending the linear MPC formulation ............... 18

3 Active-set methods for convex Quadratic Programming ....... 21
   3.1 Quadratic Programming ..................................... 21
      3.1.1 Feasibility ........................................... 23
      3.1.2 Optimality ........................................... 23
   3.2 Active-set methods ......................................... 24
   3.3 A primal active-set algorithm ............................... 26
      3.3.1 Extensions to semi-definite Hessians ............... 29
   3.4 A dual active-set algorithm ................................ 33
   3.5 Practical concerns .......................................... 33

4 Concluding remarks ............................................... 37
   4.1 Summary of contributions .................................. 37
   4.2 Future work ................................................ 38

Bibliography ......................................................... 41
II Publications

A A unifying complexity certification framework for active-set methods for convex quadratic programming 49

1 Introduction ........................................... 52
2 Preliminaries ......................................... 53
  2.1 Equality constrained mpQP ....................... 53
  2.2 A primal active-set algorithm .................... 54
3 Properties of primal active-set algorithms ............. 57
  3.1 Addition of a constraint to $W$ ................. 57
  3.2 Removal of a constraint from $W$ ............... 58
  3.3 Parameter independence of singular search directions ... 59
4 Complexity Certification ................................ 60
  4.1 Overview ............................................ 60
  4.2 Removing constraints and checking for global optimality . 63
  4.3 Adding constraints and checking for local optimality .... 64
  4.4 Adding constraints when the EQP is unbounded ........ 67
5 Extensions and practical aspects ........................ 68
  5.1 Certifying number of floating-point operations ....... 68
  5.2 Special cases with simplified partition .............. 69
  5.3 Outer approximations of quadratic inequalities .......... 70
  5.4 Constraint selection rules .......................... 70
  5.5 Warm-starting ........................................ 71
6 Relationship to other methods ........................... 71
  6.1 Dual active-set methods for Quadratic Programming .... 71
  6.2 Active-set methods for Linear Programming .......... 72
  6.3 Explicitly solving mpQP ............................. 72
7 Numerical examples ...................................... 73
  7.1 Complexity certification ............................ 74
  7.2 Affine approximations of quadratic inequalities .......... 76
8 Conclusion and future work ................................ 76
Bibliography .............................................. 77

B Exact complexity certification of an early-terminating standard primal active-set method for quadratic programming 81

1 Introduction ........................................... 84
2 Preliminaries ......................................... 85
  2.1 A primal active-set algorithm ....................... 85
  2.2 Certification of the primal active-set algorithm ....... 86
3 Early termination of the primal active-set algorithm ....... 87
  3.1 Relaxing KKT-conditions ............................ 87
  3.2 Directly comparing objective function to value function . 88
  3.3 Direct method with an underestimator of $J^*$ ........ 92
4 Examples ............................................... 93
  4.1 Relaxing KKT-conditions ............................ 93
  4.2 Direct method ....................................... 94
Contents

5 Conclusion ........................................................................... 97
Bibliography ........................................................................... 97

C Exact complexity certification of a nonnegative least-squares method for quadratic programming .................................................. 99
1 Introduction ........................................................................... 102
2 Problem formulation .............................................................. 102
  2.1 Notation .............................................................................. 103
  2.2 Nonnegative least-squares method ........................................ 103
3 Properties of NNLS algorithm .................................................. 104
  3.1 Least-squares subproblems ................................................... 105
  3.2 Checking for global optimality and adding index to \( W \) ................. 107
  3.3 Updating iterate and removing component from \( W \) .................... 107
  3.4 Singular case ....................................................................... 109
4 Certification of NNLS algorithm ................................................ 111
5 Numerical example ............................................................... 113
6 Conclusion .............................................................................. 113
Bibliography ............................................................................ 114

D Complexity certification of proximal-point methods for numerically stable quadratic programming ............................................. 117
1 Introduction ........................................................................... 120
2 Problem formulation and notation ............................................. 121
  2.1 Notation .............................................................................. 121
3 The proximal-point method ..................................................... 121
  3.1 Evaluating the proximal operator ............................................ 123
4 Parametric behaviour ............................................................. 123
5 Complexity certification .......................................................... 126
  5.1 Certifying inner iterations ..................................................... 128
6 Numerical example ............................................................... 130
7 Conclusion .............................................................................. 131
Bibliography ............................................................................ 132
## Notation

### Some sets

| Notation | Meaning |
|----------|---------|
| $\mathbb{N}_m$ | Set of positive integers up to $m$ |
| $\mathbb{R}^{m \times n}$ | Set of real $m \times n$ matrices |
| $\mathcal{S}_n^+$ | Set of real $n \times n$ positive semi-definite matrices |
| $\mathcal{S}_n^{++}$ | Set of real $n \times n$ positive definite matrices |

### Operators

| Notation | Meaning |
|----------|---------|
| $[M]_i$ | The $i$th row of matrix $M$ |
| $[M]_{\mathcal{I}}$ | The rows of matrix $M \in \mathbb{R}^{m \times \ast}$ indexed by $\mathcal{I} \subseteq N_m$ |
| $\{a_i\}_{i=1}^n$ | A sequence of $n$ elements $\left(\{a_i\}_{i=0}^n = \{a_1, a_2, \ldots, a_n\}\right)$ |
| $\text{diag}(A, B, C)$ | A block diagonal matrix with blocks $A$, $B$ and $C$ |
| $\text{vec}(a, b, c)$ | Vectors $a$, $b$ and $c$ stacked into a single vector |

### Abbreviations

| Abbreviation | Meaning |
|--------------|---------|
| MPC | Model Predictive Control |
| QP | Quadratic Program(ming) |
| mpQP | Multi-parameteric Quadratic Program(ming) |
| EQP | Equality constrained quadratic program |
Part I

Background
1 Introduction

1.1 Background and motivation

Model Predictive Control (MPC) is an advanced control strategy which at present is the go-to strategy for controlling complex systems [41]. Some of the reasons for its success is that MPC easily handles the control of multi-variable systems and that constraints on control actions (for example voltage/force limitation of actuators) as well as constraints on system states (for example, speed, acceleration and/or jerk limits) can be taken into account when selecting a control action.

As its name suggests, MPC uses a model of the plant’s behaviour to predict future states of the plant given the applied control actions and the current state. By using this model, the control problem is posed as an optimization problem, which, if solved, provides an optimal\textsuperscript{1} control action given the current state of the system.

Moreover, to counteract model errors and external disturbances acting on the system, a new control action is computed once the state of the system changes by solving another, similar, optimization problem. It is, hence, essential to be able to solve these optimization problems within a limited time frame (which gets narrower the faster the state of the system changes). A schematic overview of MPC is given in Figure 1.1.

Because of the limited time frame in which the optimization problem has to be solved, MPC first saw success in the process industry [48] where the relatively slowly changing states of the controlled plants gave enough time to solve the optimization problems with the software and hardware available at the time. With improvements in software and hardware since then, the application areas of MPC have expanded to include systems where the state is changing rapidly, for example, in automotive and aerospace applications [22; 23; 25]. As MPC is applied

\textsuperscript{1}What is meant by “optimal” can be specified based on specifications for the applications at hand.
to control faster systems, the optimization problems have to be solved more and more frequently (meaning that the time frame in which the problem has to be solved becomes narrower), which, together with the trend of implementing the controllers on embedded hardware [37; 43], means that the optimization problems have to be solved faster and with limited memory and computational resources. To meet these increasingly challenging requirements, the solvers which are used to solve the optimization problems need to become more and more efficient and frugal. Further adding to the challenge is that 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 crucial to be certain that the computational resources at hand are sufficient to solve the optimization problems within the limited time frame before the solver is employed.

The ultimate objective with the research presented in this thesis is to provide a priori guarantees that the employed solver will be able to solve all possible optimization problems that can be encountered online within the limited time frame for the application at hand. This objective can be approached from two different, complementary, directions:

(i) Developing solvers which efficiently solve the optimization problems, allowing the limited resources to be used more economically, which ultimately reduces the worst-case solution time.

(ii) Methods that, for a given solver, provide worst-case bounds on the computational complexity for solving all possible optimization problems that can be encountered online.

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

The thesis is split into two parts: Part I gives background information about model predictive control, quadratic programming and active-set methods, providing context to the publications which comprises Part II of the thesis, where the main contributions are presented. The remainder of this introductory chapter provides a summary of the publications included in Part II and the author's contribution to each.

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

Paper A \(^2\) is an edited version of

Daniel Arnström and Daniel Axehill. A unifying complexity certification framework for active-set methods for convex quadratic programming. arXiv preprint arXiv:2003.07605, 2020.

The paper is also a direct extension of the publication

Daniel Arnström and Daniel Axehill. Exact complexity certification of a standard primal active-set method for quadratic programming. In IEEE 58th Conference on Decision and Control, pages 4317–4324, Dec 2019.

Summary: In MPC an optimization problem has to be solved at each time step, which in real-time applications makes it important to solve these efficiently and to have good upper bounds on worst-case solution time. Often for linear MPC problems, the optimization problem in question is a quadratic program (QP) that depends on parameters such as system states and reference signals. A popular class of methods for solving such QPs is active-set methods, where a sequence of linear systems of equations are solved. We propose an algorithm for computing which sequence of subproblems an active-set algorithm will solve, for every parameter of interest. These sequences can be used to set worst-case bounds on how many iterations, floating-point operations and, ultimately, the maximum solution time, that the active-set algorithm requires to converge. The usefulness of the proposed method is illustrated on a set of QPs originating from MPC problems, by computing the exact worst-case number of iterations primal and dual active-set algorithms require to reach optimality.

Background and contribution: The overlying concept of tracking the working-set changes for the primal active-set method was conceived by Daniel Axehill. 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 [2], which Paper A is a direct extension of. Daniel Axehill helped out throughout the process and reviewed the manuscript. The insight that the certification method reported in

\(^2\) Under review for possible publication in IEEE Transactions on Automatic Control.
[2] could be extended to unify the results therein with the certification methods presented in [20] and [59] was conceived by the author of this thesis, 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 with the support of Daniel Axehill, who also reviewed the manuscript.

**Paper B: Exact complexity certification of an early-terminating standard primal active-set method for quadratic programming**

Paper B is an edited version of:

Daniel Arnström and Daniel 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.

**Summary:** In this paper we present a method to exactly certify the iteration complexity of a primal active-set algorithm for quadratic programs which is terminated early, given a specific multi-parametric quadratic program. The primal active-set algorithm’s real-time applicability is improved by early termination, increasing its computational efficiency, and by the proposed certification method, providing guarantees on worst-case behaviour. The certification method is illustrated on a multi-parametric quadratic program originating from model predictive control of an inverted pendulum, for which the relationship between allowed suboptimality and iterations needed by the primal active-set algorithm is presented.

**Background and contribution:** The idea of extending the complexity certification method presented in [2] to the case when the active-set algorithm is terminated early was conceived through joint discussion between both of the authors of the paper. The author of this thesis did the majority of the work including writing the manuscript, theoretical derivations and numerical experiments. Daniel Axehill helped throughout the process and reviewed the manuscript.

**Paper C: Exact complexity certification of a nonnegative least-squares method for quadratic programming**

Paper C\(^3\) is an edited version of:

Daniel Arnström, Alberto Bemporad, and Daniel Axehill. Exact complexity certification of a nonnegative least-squares method for quadratic programming. *IEEE Control Systems Letters*, 4(4):1036–1041, 2020.

**Summary:** In this paper we propose a method to exactly certify the complexity of the QP algorithm presented in [11] which is based on reformulating strictly convex quadratic programs to nonnegative least-squares problems. The exact complexity of the method is determined by proving the correspondence

\(^3\)The contents of this paper were also selected for presentation at the 59th IEEE Conference on Decision and Control, Jeju Island, Republic of Korea, 2020.
between the method and a standard primal active-set method for quadratic programming applied to the dual of the quadratic program to be solved. Once this correspondence has been established, the complexity certification framework established in Paper A can be used to certify the complexity of the nonnegative least-squares method. The usefulness of the proposed method is illustrated on a multi-parametric quadratic program originating from model predictive control of an inverted pendulum.

**Background and contribution:** The insight that the framework in Paper A could, indirectly, be applied to certify the complexity of QP algorithm in [11] was conceived by the author of this thesis, who also did the majority of the work including writing the manuscript, theoretical derivations and numerical experiments. Alberto Bemporad and Daniel Axehill helped in refining the idea and reviewed the manuscript.

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

Paper D is an edited version of:

Daniel Arnström, Alberto Bemporad, and Daniel Axehill. Complexity certification of proximal-point methods for numerically stable quadratic programming. *IEEE Control Systems Letters*, 5(4):1381–1386, 2021.

**Summary:** When solving a QP, one can improve the numerical stability of any QP solver 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 (mpQP) and any polyhedral set of parameters, determines which sequences of QPs will have to be solved when using outer proximal-point iterations. By knowing this sequence, bounds on the worst-case complexity of the method can be obtained, which is of importance in, for example, real-time MPC applications. Moreover, we combine the proposed method with previous work on complexity certification for active-set methods to obtain a more detailed certification of the proximal-point method’s complexity, namely the total number of inner iterations.

**Background and contribution:** Through discussions during the completion of Paper C, Alberto Bemporad familiarized the author of this thesis with the extensions made to [11] presented in [12], in which the 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. Alberto Bemporad and Daniel Axehill helped refining the idea and reviewed the manuscript.

---

4The contents of this paper were also selected for presentation at the 2021 American Control Conference, New Orleans, LA, USA, 2021.
At the core of automatic control is the problem of selecting a control action, $u \in \mathbb{R}^{n_u}$, which makes the system state, $z \in \mathbb{R}^{n_z}$, take "desirable" values. A widely-known and illustrative example of this is cruise control of a vehicle, where the state $z$ in this instance is the speed of the vehicle and the control $u$ is the throttle. In this application, a "desirable" value of the state is the speed in which the driver want to travel.

Often, the selection of the control action $u$ is made by a state feedback law, which given the current state $z$ of the system generates a control action $u$. Mathematically, a state feedback law is a mapping $g : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_u}$ which generates control actions as $u = g(z)$. An example of a simple, yet very popular and effective, feedback law is linear state feedback where $g$ is a linear function, i.e., $u = Lz$ for some gain matrix $L \in \mathbb{R}^{n_u \times n_z}$. For challenging control problems more sophisticated feedback laws are, however, usually needed for sufficient performance.

In Model Predictive Control (MPC) [50] the mapping $g(z)$ is evaluated implicitly by solving an optimization problem, where this optimization problem depends on the current state $z$, that is, different values of $z$ yield different optimization problems. An overview of MPC is given in Algorithm 1, where the details of what Step 3 entails is the main subject of this chapter.

**Algorithm 1 Model predictive control**

1: while true do
2: Measure (or estimate) current state $z$
3: $\{u^*_k\}_{k=0}^{N-1} \leftarrow$ Formulate and solve (2.3) with $z_0$ set to $z$
4: Use $u = u^*_0$ as the next control actions

In Section 2.2 a fairly general formulation of the optimal control problems that are solved in Algorithm 1 is introduced and these problems are 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 problems can be seen as instances of a multi-parametric quadratic program (mpQP), which means that the optimization problems to be solved on-line are quadratic programs (QPs). Before describing the optimal control problems we will, however, give some introductory remarks on the prediction model and the measurement of system states which are both necessary to formulate the optimal control problems introduced in the subsequent sections.

2.1 Preliminaries

As its name suggests, MPC is a control strategy which uses a prediction model. In the usual setting, this prediction model is a differential equation in the form

\[
\dot{z} = f_c(z, u),
\]

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 often called the dynamics of the system.

To be able to use the prediction model in practice, however, discrete-time, rather than continuous-time, models are necessary since controllers are usually implemented on computers, which inherently work in discrete time. Hence, we instead consider discrete-time dynamics of the system given in terms of a difference equation in the form

\[
z_{k+1} = f_d(z_k, u_k),
\]

where \( u_k \) and \( z_k \) denotes the control and state at time step \( k \), respectively.

There are many ways of deriving a discrete-time model \( f_d \) given a continuous-time model \( f_c \), for example through zero-order or first-order hold. Some alternatives for obtaining a model from scratch is to use physical laws, black-box models or everything in between (gray-box models). The problem of deriving a model for the system is considered in the area of system identification [40]. Even though having a model which sufficiently captures the real system’s behaviour 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.

\textbf{Assumption A1 (Linear dynamics).} A discrete-time prediction model \( z_{k+1} = f_d(z_k, u_k) \) of the plant’s dynamics is available.

Since MPC defines a state feedback law, information about the entire state \( z \) at the current time step is, in addition to a prediction model, necessary. In practice, however, the entire state is often not measured directly; 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 which distorts the measurements. In this situation, an estimate of \( z \), denoted \( \hat{z} \), is formed based on \( y \) and \( u \). Mathematically, a mapping \( (y, u) \mapsto \hat{z} \) such that \( \hat{z} \approx z \) is constructed. The problem of constructing such a mapping is known as state estimation [53] and is, akin to system identification, an entire research field on its own and will not be considered in detail in this thesis.
In other words, we assume that measurements of \( z \), or at least an estimate of \( z \), is 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 it 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

Now, assuming that a prediction model \( f_d \) and a measurement/estimate of 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 \( \{u_k\}^{N-1}_{k=0} \) which generates a desirable state sequence \( \{z_k\}^{N}_{k=0} \), \((\star)\) with \( z_0 = z \),

where \( N \) is called the prediction horizon and determines how many time steps into the future we predict when selecting a control action. Ideally, we want to let \( N \to \infty \) to take all future states into account, but a larger \( N \) also increases the number of control actions that have to be computed, leading to a trade-off between foresight and computational complexity when selecting \( N \). For convenience we will use the notation \( u \triangleq \{u_k\}^{N-1}_{k=0} \) and, similarly, \( z \triangleq \{z_k\}^{N}_{k=0} \) for the control and state sequences, respectively.

The fuzzy terms "inexpensive" and "desirable" used in control problem \((\star)\) above can be formalized through a so-called cost function \( V(z, u) \), which assigns a real value to a control and a 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 \( z \) and control actions \( u \) which minimizes \( V \). We will in this thesis, as is often done in practice, consider \( V \) in the form \( V(z, u) = V_f(z_N) + \sum^{N-1}_{k=0} \ell(z_k, u_k) \), 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 \( u \) freely either since the control actions are constrained to a set \( u \in U \) and the states to a set \( z \in Z \). Such constraints appear, e.g., actuator limits and speed limits, respectively. An additional constraint that the final predicted state \( z_N \) should reside in a set \( Z_f \), rather than \( Z \), is often used to guarantee that the resulting feedback law is stable [42].

The control problem in \((\star)\) can, hence, be formalized as the following discrete
optimal control problem:

\[
\begin{align*}
\text{minimize} & \quad V_f(z_N) + \sum_{k=0}^{N-1} \ell(z_k, u_k) \\
\text{subject to} & \quad z_{k+1} = f_d(z_k, u_k), \quad \forall k = 0, \ldots, N - 1 \\
& \quad (u_k, z_k) \in U \times Z, \quad \forall k = 0, \ldots, N - 1 \\
& \quad z_N \in Z_f \\
& \quad z_0 = z,
\end{align*}
\tag{2.3}
\]

where the values of \(u\) and \(z\) that minimize (2.3) are denoted \(u^*\) and \(z^*\), respectively. Since the optimal control problem depends on the current state \(z\), through the initial constraint \(z_0 = z\), its solution will be a function of \(z\), i.e., \(u^*(z)\) and \(z^*(z)\).

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, which means that only relative, rather than absolute, time has to be considered. The problem formulation can, however, easily be extended to also handle the time-varying case, but this is not considered in this thesis.

Receding horizon

In MPC, as was mentioned in the introduction of this chapter and summarized by Algorithm 1, discrete optimal control problems in the form (2.3) are solved recurrently, and the initial state \(z_0\) is constrained to be the current state \(z\). After solving an optimal control problem, resulting in the sequence \(u^*(z) = \{u^*_i(z)\}_{i=0}^{N-1}\), only the first control action \(u^*_0(z)\) is applied to the plant. In other words, MPC results in the feedback law \(u = u^*_0(z)\), which means that all \(u^*_i(z) i = 1, \ldots, N - 1\) are discarded. At first, it might seem drastic to discard all other control actions except \(u^*_0(z)\), but there are two main reasons that motivates 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 the optimization, ensuing consequences beyond the horizon are not directly handled. By re-solving the optimal control problems in later time steps, 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 \(Z_f\) with care, see, e.g., [42].

The second advantage of re-solving the optimal control problems in each time step is that the predicted states when solving the optimal control problem are just that: predictions. Even if we would be able to let the horizon \(N \to \infty\), the optimized sequence of control actions cease to be optimal as soon as the predicted state trajectory deviates from the actual state trajectory, which always occurs in practice due to model errors and external disturbances. By regularly re-solving the optimal control problem for the current state we can reassess our control actions and such feedback robustifies the selection of control actions.
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 \[18\] is based on the following assumptions:

**Assumption A3 (Quadratic cost).** The terminal and stage cost are quadratic, i.e., \(V_f(z) = \frac{1}{2}z^T Q_f z\) and \(\ell(z,u) = \frac{1}{2} (z^T Q z + u^T R u)\) for some \(Q_f, Q \in S^n_{+}^{z}\) and \(R \in S^n_{+}^{u}\).

**Assumption A4 (Linear dynamics).** The system dynamics is given by \(f_d(z,u) = F z + G u\) 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 set \(Z \times U\) and \(Z_f\) are polyhedral sets, i.e., \(Z \times U = \{(z,u) : A_z z + A_u u \leq b\}\) for \(A_z \in \mathbb{R}^{n_c \times n_z}\), \(A_u \in \mathbb{R}^{n_c \times n_u}\) and \(b \in \mathbb{R}^{n_c}\), and \(Z_f = \{x : A_f z \leq b_f\}\).

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

\[
\begin{align*}
\text{minimize} & \quad z_N^T Q_f z_N + \frac{1}{2} \sum_{k=0}^{N-1} (z_k^T Q z_k + u_k^T R u_k) \\
\text{subject to} & \quad z_{k+1} = F z_k + G u_k, \quad k = 0, \ldots, N-1 \\
& \quad A_z z_k + A_u u_k \leq b, \quad k = 0, \ldots, N-1 \\
& \quad A_f z_N \leq b_f \\
& \quad z_0 = z.
\end{align*}
\]

(2.4)

**Remark 2.4.** The control objective in (2.4) is to steer some or all states to the origin since \(\frac{1}{2}z_N^T Q_f z_N + \frac{1}{2} \sum_{k=0}^{N-1} (z_k^T Q z_k + u_k^T R u_k) \geq 0\), 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 in 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 \(\omega\) 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).
A discrete-time model \( z_{k+1} = Fz_k + Gu_k \), with sampling time \( T_s = 0.02 \text{ s} \), of the system dynamics is given by

\[
F = \begin{bmatrix}
1 & 0.0181 & 0.0018 & 0.0000 \\
0 & 0.8185 & 0.1783 & 0.0018 \\
0 & -0.0038 & 1.0076 & 0.0201 \\
0 & -0.3635 & 0.7500 & 1.0067
\end{bmatrix}, \quad G = \begin{bmatrix}
0.02 \\
1.82 \\
0.04 \\
3.64
\end{bmatrix},
\]

which 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 \( \phi \) 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

\[
A_z = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}, \quad A_u = \begin{bmatrix}
0 \\
0 \\
1 \\
1
\end{bmatrix}, \quad b = \begin{bmatrix}
0.5 \\
0.5 \\
1 \\
1
\end{bmatrix}.
\]

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 \( \phi \) (the first and third state) to zero, the weights in the stage cost is selected as \( Q = \text{diag}(10, 0, 1, 0) \) and \( R = 1 \). The final cost \( Q_f \) is set to the solution to the discrete-time algebraic Riccati equation which is solved to obtain the linear quadratic regulator (LQR) for an infinite horizon (see, e.g., [42] 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) and the control law defined by Algorithm 1 is used (where the subproblems are in the form (2.4) with the above-mentioned data). The MPC controller generates control actions which steer both \( x \) and \( \phi \) to zero, which was the goal, while adhering to the constraints imposed on \( u \) and \( \phi \). (Note that the linearized model also has, for simplicity, been used for the experiments).

**Figure 2.1: Inverted pendulum**
2.3 Linear MPC

2.3.1 Multi-parametric Quadratic Programming

A major advantage of 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, including gradient projection methods [8; 46; 51], operator splitting methods [54], interior-point methods [30; 56; 58], and, the QP methods which are the main focus of this thesis, active-set methods [11; 26; 27; 33; 39].

An mpQP has the form

\[
\text{minimize} \quad \frac{1}{2} x^T H x + (f + f_\theta \theta)^T x \\
\text{subject to} \quad Ax \leq b + W \theta \\
E x = d + V \theta, \tag{2.7}
\]

where the iterate \( x \in \mathbb{R}^n \) and the parameter \( \theta \in \Theta_0 \subseteq \mathbb{R}^p \). The objective function is given 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, finally, the equality constraints are given by \( E \in \mathbb{R}^{e \times n} \), \( d \in \mathbb{R}^e \) and \( V \in \mathbb{R}^{e \times n} \). By changing the parameter \( \theta \) 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 casting the optimal control problem in (2.4) into an mpQP in the form (2.7) is to view the current state \( z \) as the parameter (\( \theta = z \)) and to stack the control and states into vectors as

\[
z = \begin{pmatrix} z_0 \\ \vdots \\ z_N \end{pmatrix}, \quad u = \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix}, \tag{2.8}
\]
resulting in $z \in \mathbb{R}^{n_z(N+1)}$ and $u \in \mathbb{R}^{n_u N}$. 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 $u$ and $z$ 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 $z$ and $u$ as optimization variables, leading to a *sparse* mpQP. First, the equality constraints in (2.4) for each time step $k$ can, together with $z_0 = z = \theta$, be expressed as the following block-structured linear system of equations

$$
\begin{pmatrix}
I & 0 & 0 & \ldots & 0 \\
-F & I & 0 & \ldots & 0 \\
0 & -F & I & \ldots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & -F & I \\
\end{pmatrix}
\begin{pmatrix}
z_0 \\
z_1 \\
z_2 \\
\vdots \\
z_N \\
\end{pmatrix}
= 
\begin{pmatrix}
0 & 0 & \ldots & 0 \\
-G & 0 & \ldots & 0 \\
0 & -G & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots \\
0 & \ldots & 0 & -G \\
\end{pmatrix}
\begin{pmatrix}
u_0 \\
u_1 \\
u_2 \\
\vdots \\
u_{N-1} \\
\end{pmatrix}
= 
\begin{pmatrix}
I \\
0 \\
0 \\
\vdots \\
0 \\
\end{pmatrix}
\theta 
\tag{2.9}
$$

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 $Q \triangleq \text{diag}(Q, \ldots, Q, Q_f)$ and $R \triangleq \text{diag}(R, \ldots, R)$, into

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

Finally, the inequality constraints $A_z z_k + A_u u_k \leq b$ for $k = 0, \ldots, N - 1$ and the terminal constraint $A_f z_N \leq b_f$ can be combined, by introducing block diagonal matrices $A_u \triangleq \text{diag}(A_u, \ldots, A_u, 0)$ and $A_z \triangleq \text{diag}(A_z, \ldots, A_z, A_f)$ and the vector $b \triangleq \text{vec}(b, \ldots, b, b_f)$, into

$$
A_u u + A_z z \leq b. 
\tag{2.11}
$$

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

$$
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{pmatrix} u & z \end{pmatrix}^T \begin{pmatrix} R & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} u \\ z \end{pmatrix} \\
\text{subject to} & \quad \begin{pmatrix} E_u & E_z \end{pmatrix} \begin{pmatrix} u \\ z \end{pmatrix} = D \theta \\
& \quad \begin{pmatrix} A_u & A_z \end{pmatrix} \begin{pmatrix} u \\ z \end{pmatrix} \leq b,
\end{align*}
\tag{2.12}
$$

which with optimization variable $x = \begin{pmatrix} u \\ z \end{pmatrix}$ is in the form (2.7).
Condensed formulation

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

$$z = E_z^{-1} \mathbf{D} \theta - E_z^{-1} E_u u,$$

(2.13)

where $E_z^{-1}$ is guaranteed to exist since $E_z$ is lower unit triangular. An alternative interpretation to the purely mathematical one of inverting $E_z$, which is also preferable numerically, is to use the linear dynamics for forward simulation to express future states in terms of only the control actions $u$ and the starting state $z_0 = \theta$. That is, by using the dynamics $z_{k+1} = F z_k + G u_k$ iteratively the state at time step $k$ can be expressed as

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

(2.14)

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

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

(2.15)

which can be compactly written as

$$z = F \theta + G u.$$

(2.16)

Eliminating $z$ in (2.12) using (2.16), and removing terms in the objective function which do not contain $u$, gives the mpQP

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} u^T \left( G^T Q G + R \right) u + 2 \theta^T F Q G u \\
\text{subject to} & \quad (A_z G + A_u) u \leq b - A_z F \theta,
\end{align*}$$

(2.17)

which is in the form (2.7) with $x = u$.

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

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T H x + (f + f_0 \theta)^T x \\
\text{subject to} & \quad A x \leq b + W \theta
\end{align*}$$

(2.18)
However, the subsequent ideas can easily be extended to the case when equality constraints are used but requires some additional, obfuscating, notation.

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 [7], and only eliminates a subset of the elements of the stacked state vector \( z \) (in contrast to eliminating none or all elements of \( z \) 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 in the form (2.7), i.e, all of 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 \( \infty \) -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 [18]). 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 to be viable 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 these extensions still allow the resulting optimal control problem to be cast as an mpQP, 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 of the results are derived for a general mpQP in 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, however, want to steer a linear combination of our states to a given value, i.e., want \( Cz = r \) to hold, where \( r \in \mathbb{R}^n_r \) is the desired reference value and \( C \in \mathbb{R}^{n_r \times n_z} \) defines 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 \( r \) is not 0 a non-zero control action might be required to maintain the states at the desired reference which, hence, make it more reasonable to incur a cost on the change of \( u \) rather than the 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

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

where now \( Q \in S^{n_r}_+ \), in contrast to \( Q \in S^{n_z}_+ \) 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 where the parameter \( \theta \) now, in addition to the
2.3 Linear MPC

state $z$, also contains the reference value $r$ and the previous control actions $u_{-1}$, i.e., $\theta = \text{vec}(z, r, u_{-1})$. For additional details, see, e.g., Section 6.1 in [15].

**Softening constraints**

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

Usually in MPC, the 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, generally, what cause the QPs to be infeasible. Softening these state constraints, hence, ensure that a solution exists. The pragmatic reason for considering constraint on $u$ as hard and constraints on $z$ as soft is that constraints on $u$ are, often, imposed based on physical limitations of actuators, which cannot be violated in practice (even if you tried), motivating them to be seen as hard constraints. In contrast, state constraints often arise from desired system specifications which can, if necessary, temporarily be violated (for example constraints on the velocity due to safe driving), motivating them to be regarded as soft constraints.

There are several ways of softening constraints but a commonly used approach [60] is to add a single optimization variable $\epsilon_s \in \mathbb{R}$ to minimize over and modify the constraints as

$$Ax \leq b + W\theta \rightarrow Ax \leq b + W\theta + S\epsilon_s,$$

where $S \in \mathbb{R}^m$ is a selection matrix with $[S]_i = 0$ if the $i$th constraints is a hard constraint and $[S]_i = 1$ if the $i$th constraint is a soft constraint. A large value on $\epsilon_s$ will, hence, relax the soft constraints and by setting its value 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. The softened problem is, hence, given by

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

(2.21)

which is on the same form as (2.18).
Active-set methods for convex Quadratic Programming

In the previous chapter we derived how the optimal control problems encountered in linear MPC can be cast as instances of an mpQP, resulting in QPs being solved in each time step when linear MPC is used in practice. In this chapter we present in detail a class of QP algorithms which we consider in this thesis for solving these QPs, namely, 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 also formulate a dual active-set algorithm in Section 3.4. Finally, Section 3.5 introduces some important concepts to keep in mind when an active-set algorithm is used in practice.

3.1 Quadratic Programming

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

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

(3.1)

where $[\cdot]_i$ denotes the $i$th row of a matrix and $x \in \mathbb{R}^n$. 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 geometric intuition for (3.1) in the case when $H > 0$. 

21
Example 3.1: Visualization of QP

Consider the quadratic program

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

(3.2)

which is in 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 $x^* = (-2, 3)$, which is the point in which the level curves "just touches" the feasible set.

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

Remark 3.1. The shape of the level curves of (3.1), which are ellipses when $H > 0$, 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 depends on $A$ and not $b$; Changing $b$ only offsets these half-planes. Since the parameter $\theta$ in (2.18) neither affects $H$ nor $A$, different values of $\theta$ only translates the unconstrained optimum and off-sets the half-planes, but the shape of the level curves and the normal of the half-planes remain the same, which is the reason for the structured parameter dependence of the solutions to (2.18) that is exploited in explicit MPC [13] and in the papers presented in Part II.
3.1 Quadratic Programming

3.1.1 Feasibility

When the QP in (3.1) does not have a solution it is said to be infeasible, and infeasibility can arise in two different ways. The first case occurs when there does not exist any point which 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 which satisfies \( Ax \leq b \) is often called phase-1.

The second case of infeasibility, which can occur if \( H \) is singular, occurs when there exist primal feasible points which make the objective function arbitrarily small, i.e., a finite 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, also known as the KKT conditions [38]

\[
\begin{align*}
    Hx^* + A^T \lambda &= -f \\
    Ax^* &\leq b \\
    \lambda &\geq 0 \\
    ([b]_i - [A]_i x^*)[\lambda]_i &= 0, \quad \forall i \in \mathbb{N}_m,
\end{align*}
\]

for some \( \lambda \in \mathbb{R}^m \), which is called a dual variable/multiplier. 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, finally, (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 [45].

In general the KKT conditions are only necessary conditions for optimality, but for convex problems, which means in particular for a QPs with \( H \succeq 0 \), the conditions are also sufficient [19]. Furthermore, (3.3a) has a unique solution \( x^* \) if \( H > 0 \) 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 is, hence, the condition which makes quadratic programming somewhat challenging. In active-set methods, soon to be introduce, the complementary slackness condition is ensured to hold throughout all iterations, by using a so-called working set, 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 the QP, instead, only contained equality constraints (known as an equality constrained QP (EQP)) the problem could be solved by solving one set of linear equations. Concretely, the minimizer $x^*$ of the EQP

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

is a solution to the system of linear equations

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

which is also known as a KKT system. (In other words, stationarity and primal feasibility are the only necessary conditions for optimality when only equality constraints are present, while dual feasibility and complementary slackness also become necessary for optimality once inequality constraints are present.)

The straightforwardness of solving EQPs is what is exploited in active-set methods and an important insight, which forms the foundation for active-set methods, is that only the inequality constraints which hold with equality at $x^*$ are, in fact, 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 $x$, i.e., the set $\mathcal{A}(x) \doteq \{i \in \mathbb{N}_m : [A]_i x = [b]_i\}$.

The following lemma formalizes the importance of the active set at $x^*$, which, intuitively, 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 $\mathcal{A}^* \doteq \mathcal{A}(x^*)$. Then $x^*$ is also the solution to the EQP

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

**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

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

(3.7)

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

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

(3.8)
Taken together, (3.7) and (3.8) form the KKT system
\[
\begin{bmatrix}
H & [A^T]_{A^*} \\
[A]_{A^*} & 0 \\
\end{bmatrix}
\begin{bmatrix}
x^* \\
[\lambda]_{A^*} \\
\end{bmatrix}
= \begin{bmatrix}
-f \\
[b]_{A^*} \\
\end{bmatrix},
\]
which coincides with the KKT system for (3.6).

The key takeaway from Lemma 3.1 is that if \( A^* \) 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 \( A^* \). This identification is done by iteratively updating a so-called working set, denoted \( \mathcal{W} \), which can be seen as an estimate of \( A^* \). Updates to \( \mathcal{W} \) are done by adding/removing constraints to/from it, and which constraints are added/removed are determined by solving an EQP that is defined by the current working set \( \mathcal{W} \). In other words, the QP is solved by solving a sequence of EQPs (system of linear equations), where each EQP in the sequence is determined by the current working set.

A prototypical formulation of an active-set algorithm for quadratic programming is given in Algorithm 2.

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

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

Different approaches for modifying \( \mathcal{W} \) leads to different types of 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* [21; 27; 45].

- **Dual** methods work with a dual iterate \( \lambda \) and ensure that the iterate is dual feasible throughout all iterations. *Dual feasibility is ensured and primal feasibility is sought after* [9; 33; 47].

- **Primal-dual** methods work with a primal-dual pair \((x, \lambda)\). Neither primal nor dual feasibility of the iterates are ensured before termination [36; 39].

- **Parametric** methods are specialized for mpQPs and starts with the optimal solution given a nominal parameter \( \theta_0 \) and update the working set by using a homotopy [1] for the mpQP to obtain a solution for the current parameter \( \theta \) [17; 26].

The most commonly used primal and dual methods tend to 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 and we will now introduce such a primal active-set algorithm. For a comprehensive survey of the theoretical details underlying these type of active-set methods see, e.g., [57].

3.3 A primal active-set algorithm

Before getting into the details of the steps performed in a primal active-set algorithm, we start out by relating 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 while, simultaneously, the complementary slackness condition (3.3d) is enforced through a working set $W \subseteq \mathbb{N}_m$. The working set $W$ enforces complementary slackness by imposing that all equality constraints contained in it should hold with equality, i.e., $[A][x] = [b]$, $\forall i \in W$, while the dual multipliers for the constraints not in $W$ are, implicitly, fixed to be zero, i.e., $[A][x] = 0$, $\forall i \in \bar{W}$ (where $\bar{W}$ denotes the complement of $W$, i.e., $\bar{W} \equiv \mathbb{N}_m \setminus W$).

Since both primal feasibility and complementary slackness are enforced throughout all iterations, dual feasibility (3.3c) and stationarity (3.3a) are sought after in each iteration.

The primal active-set algorithm considered herein 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 $W_k$ denote $x$ and $W$ at iteration $k$).

\begin{algorithm}
\caption{(A primal active-set algorithm for solving (3.1) when $H > 0$)}
\begin{itemize}
  \item [\textbf{Input:}] $x_0, W_0, k = 0$
  \item [\textbf{Output:}] $x^*, \lambda^*$
  \begin{itemize}
    \item [1:] \textbf{while} true \textbf{do}
    \item [2:] Compute $x^*_k$ by solving (3.9)
    \item [3:] \textbf{if} $Ax^*_k \leq b$ \textbf{then} $\rightarrow x^*_k$ is primal feasible
    \item [4:] Compute $\lambda_k$ by solving (3.10)
    \item [5:] \textbf{if} $\lambda_k \geq 0$ \textbf{then return} $x^* \leftarrow x^*_k$, $\lambda^* \leftarrow \lambda_k$ $\rightarrow$ Optimal solution found
    \item [6:] \textbf{else} $l \leftarrow \arg\min_{i \in W_k} [\lambda_k]_i$
    \item [7:] $W_{k+1} \leftarrow W_k \setminus \{l\}$; $x_{k+1} \leftarrow x^*_k$
    \item [8:] \textbf{else} $l \leftarrow \arg\min_{i \in \bar{W}_k, [A][x] > [b]} [A][x^*_k - x_k]$
    \item [9:] $W_{k+1} \leftarrow W_k \cup \{l\}$; $x_{k+1} \leftarrow x_k + \alpha_l^k(x^*_k - x_k)$ $\rightarrow x^*_k$ is not primal feasible
  \end{itemize}
  \item [10:] $k \leftarrow k + 1$
\end{itemize}
\end{algorithm}
Since primal feasibility should be maintained throughout all iterations, Algorithm 3 starts with an iterate $x_0$ which is primal feasible, i.e., $Ax_0 \leq b$, and a working set $W_0$ which contains a subset of the active constraints at $x_0$, i.e., $W_0 \subseteq A(x_0)$. In each iteration of the algorithm, a constraint will either be added to or removed from the working set $W_k$, 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.

(Step 2): An iteration always starts by computing the solution $x_k^*$ to the EQP defined by the current working-set $W_k$

$$x_k^* \triangleq \text{argmin}_x \frac{1}{2} x^T H x + f^T x$$
subject to $[A]_i x = [b]_i, \; \forall i \in W_k$. \hfill (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 $W_k$.

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 is dual feasibility (3.3c), which can be checked by investigating the dual variables $\lambda$ for the constraints in $W$. To check dual feasibility, the dual variables are, hence, computed by solving

$$[A]_i^T W_k [\lambda]_{W_k} = -Hx_k^* - f.$$ \hfill (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 $\lambda_k \geq 0$, a constraint corresponding to a negative component of $\lambda_k$ is removed from $W$. There are different policies for selecting which negative component to remove, known as selection rules, but the most commonly used selection rule is Dantzig’s selection rule which selects the constraint corresponding to the most negative component of $\lambda_k$, i.e.,

$$l = \min_{i \in W_k} [\lambda_k]_i.$$ \hfill (3.11)

Selecting any negative component does, however, ensure the convergence of the algorithm and some alternative rules are surveyed in [55]. To summarize, when
Active-set methods for convex Quadratic Programming

3

$x^*_k$ is primal feasible but not dual feasible, the working set and the iterate is updated as

$$W_{k+1} = W_k \setminus \{l\}, \quad x_{k+1} = x^*_k. \quad (3.12)$$

**Step 8 & 9:** If $x^*_k$ is not primal feasible ($Ax^*_k \not\in b$), it follows from the convexity of $Ax \leq b$ that at least one constraint in $W_k$ becomes violated while moving from $x_k$ to $x^*_k$ along $L_k$. To retain primal feasibility, a violated constraint will be added to the working set and 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 $W_k$ becomes active. Let $a^i_k$ be the step length taken along $L_k$ which activates constraint $i$, explicitly given by

$$[A]_i(x_k + a^i_k(x^*_k - x_k)) = [b]_i \Leftrightarrow a^i_k = \frac{[b]_i - [A]_i x_k}{[A]_i(x^*_k - x_k)}. \quad (3.13)$$

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

$$l = \arg\min_{i \in W_k : [A]_i x^*_k > [b]_i} a^i_k, \quad (3.14)$$

and the working set and iterate are updated as

$$W_{k+1} = W_k \cup \{l\}, \quad x_{k+1} = x_k + a^l_k(x^*_k - x_k). \quad (3.15)$$

**Step 10** After $W$ 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 solve a strictly convex QP is given below.

---

**Example 3.2: Primal active-set algorithm in action**

Consider again the QP in Example 3.1 which will now be solved using Algorithm 3. We select the starting iterate $x_0 = (0, 0)$ (which is primal feasible) and the starting working set $W_0 = \emptyset$.

**Iteration 0:** Solving (3.9) with $W = \emptyset$ results in $x^*_0 = -H^{-1}f = (-6, 5)$, i.e., the unconstrained minimum. Since $Ax^*_0 \not\in b$, $x^*_0$ is not primal feasible (also evident from Figure 3.2) and a constraint will be added to $W$. By computing the step length until activation for each constraint in $W_0$ (constraint 1 and 2) we get $a^1_0 = 0.41$ and $a^2_0 = 0.52$. Since $a^1_0 < a^2_0$, 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 $W_1 = \{1\}$.

**Iteration 1:** Solving (3.9) with $W_1$ results in $x^*_1 = (-1.71, 3.57)$ which is primal infeasible since it violates the remaining constraint not in $W_1$ (constraint 2). The second constraint is, hence, added to the working set and, since the step length to activation of constraint 2 is $a^2_1 = 0.62$, the iterate is updated as $x_2 = x_1 + 0.62(x^*_1 - x_1)$ and constraint 2 is added to the working set. The iteration ends with $x_2 = (-2, 3)$ and $W_2 = \{1, 2\}$.

**Iteration 2:** Solving (3.9) with $W_2$, 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 $\lambda$ 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.

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

### 3.3.1 Extensions to semi-definite Hessians

The main problem that arises when $H$ is allowed to be singular is that the subproblems in (3.9) might not have a unique solution; the subproblems might even be unbounded. Both uniqueness and unboundedness can be determined by the 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 [45]).

**Lemma 3.2 (Existence of unique solution).** If the reduced Hessian $Z_k^T H Z_k > 0$ (i.e., is nonsingular) there exists a unique solution to the EQP subproblem (3.9).
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 which satisfies \([A]_{W_k}\xi = [b]_{W_k}\) (i.e., is a particular solution). Then any point in \( x \in \mathbb{R}^n \) which satisfies the equality constraints \([A]_{W_k}x = [b]_{W_k}\) can be decomposed into \( x = \xi + Z_k\tilde{x} \), where \( \tilde{x} \in \mathbb{R}^{|W_k|} \) (recall that \( Z_k \) is a full rank matrix with column vectors that span the null space of \([A]_{W_k}\)). Therefore, the objective function of subproblem (3.9) can be written as

\[
\frac{1}{2} x^T H x + f^T x = \frac{1}{2} (\xi + Z_k\tilde{x})^T H (\xi + Z_k\tilde{x}) + f^T (\xi + Z_k\tilde{x})
\]

\[
= \frac{1}{2} \tilde{x}^T Z_k^T H Z_k \tilde{x} + (f^T + \xi^T H)Z_k \tilde{x} + c
\]

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

\[
\min_{\tilde{x}} \quad \frac{1}{2} \tilde{x}^T Z_k^T H Z_k \tilde{x} + (f^T + \xi^T H)Z_k \tilde{x} + c \tag{3.16}
\]

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

A consequence of Lemma 3.2 is that while 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, which can only happen after the removal of a constraint from \( W \) (if the initial reduced Hessian is nonsingular), as is shown by the following lemma.

**Lemma 3.3 (Reduced Hessian after an addition to \( W \)).** If \( Z_k H Z_k > 0 \) and constraint \( l \) is added in iteration \( k \) of Algorithm 3, \( Z_{k+1} H Z_{k+1} > 0 \).

**Proof:** If constraint \( l \) is added in iteration \( k \), \([A]_{W_{k+1}}\) will have one additional row compared with \([A]_{W_k}\). Since the added row will be linearly independent to the rows in \([A]_{W_k}\) (see, e.g., Section 16.5 in [45]), there exist null space bases which are related as \( Z_k = (Z_{k+1} \quad \tilde{z}) \) 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

\[
Z_k^T H Z_k = (Z_{k+1} \quad \tilde{z})^T H (Z_{k+1} \quad \tilde{z}) = \left( \begin{array}{cc} Z_{k+1}^T H Z_{k+1} & Z_{k+1}^T H \tilde{z} \\ \tilde{z}^T H Z_{k+1} & \tilde{z}^T H \tilde{z} \end{array} \right) > 0, \tag{3.17}
\]

which implies, from Sylvester’s criterion, that \( Z_{k+1}^T H Z_{k+1} > 0 \).
Lemma 3.4 (Unbounded search direction). Let \( x_k \) satisfy \([A]_{W_k} x_k = [b]_{W_k}\) and \( \bar{p} \in \mathbb{R}^n\) be a solution to
\[
H \bar{p} = 0, \quad [A]_{W_k} \bar{p} = 0, \quad f^T \bar{p} < 0.
\]
Then
(i) \([A]_{W_k} (x_k + \alpha \bar{p}) = [b]_{W_k}, \forall \alpha \in \mathbb{R}\)
(ii) \(J(x_k + \alpha \bar{p}) \rightarrow -\infty\) as \(\alpha \rightarrow \infty\).

Proof: (i) Directly follows from \([A]_{W_k} \bar{p} = 0\) and \([A]_{W_k} x_k = [b]_{W_k}\). (ii) Evaluating the objective function \(J\) for the point \(x_k + \alpha \bar{p}\) gives
\[
J(x_k + \alpha \bar{p}) = \frac{1}{2}(x_k + \alpha \bar{p})H(x + \alpha \bar{p}) + f^T(x_k + \alpha \bar{p})
= \frac{1}{2}x_k^T H x_k + \alpha x_k^T H \bar{p} + \frac{\alpha^2}{2} \bar{p}^T H \bar{p} + f^T x_k + \alpha f^T \bar{p}
= J(x_k) + \alpha x_k^T H \bar{p} + \frac{\alpha^2}{2} \bar{p}^T H \bar{p} + \alpha f^T \bar{p}
= J(x_k) + \alpha f^T \bar{p},
\]
where \(H \bar{p} = 0\) has been used in the last equality. Now since \(f^T \bar{p} < 0\) we have that \(\alpha f^T \bar{p} \rightarrow -\infty\) when \(\alpha \rightarrow \infty\) and, hence, \(J(x_k + \alpha \bar{p}) \rightarrow -\infty\) when \(\alpha \rightarrow \infty\).

Lemma 3.4 implies that if there exists a \(\bar{p}\) which satisfies (3.18), moving along the ray \(x_k + \alpha \bar{p}, \alpha > 0\) can decrease the objective function by an arbitrary amount while staying on the manifold defined by the working set \(W_k\). The following lemma ensures that when the reduced Hessian becomes singular after a constraint is removed from \(W\), there always exists a solution to (3.18).

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 \(\bar{p} \in \mathbb{R}^n\) which solves (3.18).

Proof: Let \(Z_k\) be matrix with columns that are a basis to the null space of \([A]_{W_k}\). Then, since the reduced Hessian is singular, there exists \(p\) such that \(Z_k^T H Z_k p = 0\), which, because \(Z_k\) has full rank, implies that \(H Z_k p = 0\). Now let \(\bar{p} \doteq Z_k p\). It then directly follows that \(H \bar{p} = 0\) and \([A]_{W_k} \bar{p} = 0\). What remains to prove is that \(\bar{p}\) also satisfies \(f^T \bar{p} < 0\). Since a constraint was removed in iteration \(k - 1\) the stationarity condition (3.3a) held for \(x^*_{k-1}\) and \(\lambda_{k-1}\), i.e.,
\[
H x^*_{k-1} + [A]_{W_k}^T [\lambda_{k-1}]_{W_k} + [A]_{I}^T [\lambda_{k-1}]_{I} = -f,
\]
where we have used that \([A]_{W_{k-1}}^T [\lambda_{k-1}]_{W_{k-1}} = [A]_{W_k}^T [\lambda_{k-1}]_{W_k} + [A]_{I}^T [\lambda_{k-1}]_{I}\) since \(W_{k-1} = W_k \cup \{l\}\). Transposing (3.20) and multiplying with \(\bar{p}\) from the right then gives
\[
f^T \bar{p} = -(H x^*_{k-1} + [A]_{W_k}^T [\lambda_{k-1}]_{W_k} + [A]_{I}^T [\lambda_{k-1}]_{I})^T \bar{p}
= -((x^*_{k-1})^T H \bar{p} + [\lambda_{k-1}]_{W_k}^T [A]_{W_k} \bar{p} + [\lambda_{k-1}]_{I} [A]_{I} \bar{p})
= -[\lambda_{k-1}]_{I} [A]_{I} \bar{p},
\]
where \( H\tilde{p} = 0 \) and \([A\]_{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, \([A]\tilde{p} \neq 0\) since \([A]_l^T \perp \text{Null}(\{A\}_{W_k})\). Taken together, \([-\lambda_{k-1}]_l[A]\tilde{p} \neq 0\). 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).

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 \(\tilde{p} \in \mathbb{R}^n\) which can decrease the objective function by an arbitrary amount while staying on the manifold defined by the current working set.

When deciding the step length \(\alpha\) to move in the direction \(\tilde{p}\), two different scenarios can occur. If there exists a constraint in \(\hat{W}_k\) which becomes active while moving along the ray \(x_k + \alpha\tilde{p}, \alpha > 0\), the first constraint in \(\hat{W}_k\) which becomes active is added to \(W_k\) (similar to Steps 8 & 9 in Algorithm 3). Otherwise, if no constraint in \(\hat{W}_k\) 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 \([A]_{W_k}\tilde{p} \geq 0\).

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

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

**Input:** \(x_0, \mathcal{W}_0\) (such that \(Z_0^THZ_0 > 0\), \(k = 0\)

**Output:** \(x^*, \lambda^*\)

1: while true do
2: Compute \(x^*_k\) by solving (3.9) \(\triangleright x^*_k\) is primal feasible
3: if (3.9) is unbounded then SINGULARITERATION \(\triangleright\) Optimal solution found
4: else
5: if \(Ax^*_k \leq b\) then \(l \leftarrow \text{argmin}_{i \in \hat{W}_k} [\lambda_k]_i\) \(\triangleright x^*_k\) is not primal feasible
6: Compute \(\lambda_k\) by solving (3.10)
7: if \(\lambda_k \geq 0\) then return \(x^* \leftarrow x^*_k, \lambda^* \leftarrow \lambda_k\)
8: else \(l \leftarrow \text{argmin}_{i \in \hat{W}_k:\lambda_k > 0} \frac{[b]_i - [A]_i x_k}{[A]_i (x^*_k - x_k)}\)
9: \(\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{l\}; \quad x_{k+1} \leftarrow x^*_k\)
10: else \(l \leftarrow \text{argmin}_{i \in \hat{W}_k:[A]_i x_k^* > [b]_i} \frac{[b]_i - [A]_i x_k}{[A]_i (x^*_k - x_k)}\)
11: \(\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{l\}; \quad x_{k+1} \leftarrow x_k + \alpha^*_k(x^*_k - x_k)\)
12: \(k \leftarrow k + 1\)

13: procedure SINGULARITERATION
14: Compute \(\tilde{p}\) from (3.18)
15: if \([A]_{\hat{W}_k}\tilde{p} \geq 0\) then break unbounded
16: else \(l \leftarrow \text{argmin}_{i \in \hat{W}_k:[A]_i \tilde{p} < 0} \frac{[b]_i - [A]_i x_k}{[A]_i \tilde{p}}\)
17: \(\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{l\}; \quad x_{k+1} \leftarrow x_k + \frac{[b]_i - [A]_i x_k}{[A]_i \tilde{p}} \tilde{p}\)
3.4 A dual active-set algorithm

When $H > 0$, one can instead of solving the QP in (3.1) solve its so-called dual, which is also a QP given as

$$ \begin{align*}
\min_{\lambda} & \quad \frac{1}{2} A^T H^{-1} A \lambda + (b - AH^{-1} f)^T \lambda \\
\text{subject to} & \quad \lambda \geq 0.
\end{align*} \tag{3.22} $$

It can be shown, see [24], that the solution $\lambda^*$ to (3.22) satisfies the same KKT conditions as (3.1). After solving (3.22), the primal solution $x^*$ to (3.1) can therefore be retrieved from $\lambda^*$ 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 QP in (3.1). Generally, the Hessian of the dual, $A^T H^{-1} A$, is positive semi-definite, 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.

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 when using Algorithm 4. Moreover, the computation of step lengths to constraints in $\bar{W}_k$, performed in Step 10, simplifies to $\alpha^i_k = \frac{[\lambda^i_k]}{[\lambda^i_k - \lambda^0_k]}$ 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 $W$ 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, which means that all iterates (except the last one which is the optimal solution) produced by a dual active-set algorithm violate some constraints 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 $b$ (and a small change in $\theta$ results in a small perturbation to $f$ and $b$) while $H$ and $A$ remain constant. Hence, if the state (which is related to $\theta$) 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 to reduce computations, known as warm starting the solver. 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 sometimes be exacerbated [35].

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 sometimes be challenging since the perturbation of $b$ (caused by the change in $\theta$) might yield a solution to the previous problem infeasible for the new QP. In the context of linear MPC, however, it is often easy to obtain a primal feasible iterate even if $b$ is perturbed since the constraints usually consist of box constraints in the form $l \leq x \leq u$ (where $l$ and $u$ do not depend on $\theta$) and more general constraints in the form $Ax \leq b$ (where $b$ depends on $\theta$). Since the constraints $Ax \leq b$ are usually softened (see Section 2.3.2), selecting any starting point $x_0$ between $l$ and $u$ (which is trivial) and initializing the slack $\epsilon_s$ for the soft constraints to a high enough 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 $\theta$).

**Early termination**

An advantage of primal over dual active-set algorithms is that they retain primal feasibility throughout all iterations while monotonically decreasing the objective function. Hence, a primal active-set algorithm can be terminated early and still provide an iterate which satisfies the constraints. The solutions after such early terminations will be suboptimal but can sometimes 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 [52]. 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 to be used in linear MPC.

There are, however, MPC applications where early-terminating dual active-set algorithms are very useful, namely, in MPC of hybrid systems (where some of the states take discrete values) [14]. In these applications, the optimization problems that have to be 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 to be solved. In branch-and-bound, lower bounds on the optimal objective function value can be used to reduce the number of QPs that need to be solved, which greatly decreases the computational complexity [29]. 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 [28], 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 [28; 32].

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., $H \rightarrow H + \epsilon I$ for some $\epsilon > 0$. A drawback of such regularizations is, however, that the solution to the QP is also 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 in the form

$$x_{k+1} = \arg\min_x \frac{1}{2} x^T (H + \epsilon I) x + (f - \epsilon x_k)^T x \quad \text{subj. to } Ax \leq b.$$  

(3.24)

It can then be shown (see, e.g., Theorem 10.28 in [10]) that $x_k \rightarrow x^*$ as $k \rightarrow \infty$ if such iterations are performed. Moreover, a larger $\epsilon$ leads to a better conditioned Hessian, but also to more iterations before convergence (again, see, e.g., Theorem 10.28 in [10] 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 and solving them from scratch in each iteration would lead to a large computational cost. However, these systems can be solved very inexpensively since only a single element is added/removed from $W$ at a time, making the KKT system between two adjacent iterations very similar. For example, in iteration $k$ the KKT system to be solved has the form

$$\begin{pmatrix} H & [A]^T_W \\ [A]_W & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -f \\ [b]_W \end{pmatrix},$$

and if we assume that constraint $m$ is added in this iteration the next KKT system to solve is in the form

$$\begin{pmatrix} H & [A]^T_W & [A]^T_m \\ [A]_W & 0 & 0 \\ [A]_m & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \lambda^+ \end{pmatrix} = \begin{pmatrix} -f \\ [b]_W \\ [b]_m \end{pmatrix}.$$  \hspace{1cm} (3.26)

That is to say, 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 (known as the KKT matrix) can be computed and updated through low-rank updates when constraint are added/removed to/from the working set $W$. Some standard ways of factorizing the matrix is through the $LDL^T$ factorization or a $QR$ factorization (see, e.g., [34]). Moreover, in the context of linear MPC when a sparse formulation is used, the KKT matrix can be factorized using the Riccati factorization [49]. For efficient low-rank updates of the $LDL^T$ and $QR$ factorization see, e.g., [31] and for low-rank updates of the Riccati factorization see [44].

In Part II 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 sequence of KKT systems that will be solved for any QP instance that can arise from a given mpQP. Hence, given any particular way of factorizing the KKT matrix we can determine how many flops are needed to solve the sequence of KKT systems.
In this chapter, the first part of this thesis is concluded by summarizing the main contributions presented in Part II and by giving some future research directions to extend/improve the results therein.

### 4.1 Summary of contributions

In Paper A we present a unifying complexity certification framework for a class of active-set QP methods (which includes Algorithm 4 that was presented in Chapter 3). This framework can determine exactly which working-set sequences are generated for any possible QP instance the can arise for a given mpQP. Knowing these working-set sequences means that exactly which subproblems (linear systems of equations) that have to be solved to generate a solution can be determined, which can, moreover, be used to provide guarantees on worst-case upper bounds on iterations, floating-point operations and, ultimately, solution time, needed by the active-set algorithm. By using the proposed certification method, these complexity bounds can, hence, be determined before the active-set algorithm is used online in, e.g., linear MPC, which can be used to ascertain that the available hardware is sufficient.

In Paper B we extend the certification method presented in Paper A to also handle early-terminating active-set algorithms. Given a user-specified tolerance on suboptimality (both absolute and relative), the resulting working-set changes needed to yield an iterate within the tolerance is determined. This extension can, for example, be used to analyze the trade-off between suboptimality and worst-case computational complexity of the considered active-set algorithm.

In Paper C we show that the computational complexity of an active-set algorithm (QPNNLS [11]) which does not directly fall into the framework presented in Paper A can, nevertheless, be certified indirectly with this framework. This is
Concluding remarks

done by proving that the working-set sequence produced by QPNNLS will be the same as for the algorithm considered in Paper A applied to the dual of the QP to be solved. The contribution of Paper C is, hence, twofold: proving the equivalence between two QP methods and, from this equivalence, making it possible to certify the complexity of QPNNLS.

Finally, in Paper D we present a complexity certification method which determines the worst-case number of outer proximal-point iterations that will be performed when solving any QP instance that can arise for a given mpQP. Furthermore, we also show how the certification of outer iterations can be combined with the framework presented in Paper A to determine the worst-case total number of inner iterations and/or floating-point operations if an active-set method covered by this framework is used as the inner solver.

4.2 Future work

There are multiple directions in which the work presented in Part II can be extended and applied. Here we mention four prospective directions.

High-performance computing

As is discussed in Paper A, the proposed certification method shares the rapid complexity growth that explicit MPC suffers from. An advantage of the certification method, however, is that all of this complexity only has to be dealt with offline, while the complexity has to be dealt with offline and online for explicit MPC since the critical regions have to be stored and looked up online. Since the complexity is solely in offline computations for the proposed certification method, using high-performance computing (HPC) to further increase the problem sizes which can be certified within a reasonable time is an interesting research direction, especially since the method presented in Paper A is well-suited for parallelization.

Optimizing the optimizer

There are numerous degrees of freedom when an active-set algorithm is used. For example, the method (factorization) used for solving the systems of linear equations in each iteration, the selection rule that is used when updating the working set and whether a primal or dual algorithm is used. The best choices among these options are problem dependent and an interesting research direction is, hence, to use the framework proposed in Paper A to survey which particular choices give the best worst-case performance when QP instances from a particular mpQP are solved.

Analyzing warm starts

One of the favourable properties of active-set algorithms are that they can easily be warm started. Since the method in Paper A considers any starting working
set and allows the starting iterate to be affine in the parameter, the certification method therein can be used to certify the complexity of semi-explicit warm starts for mpQPs (similar to what is done in [59] for mpLPs).

**Developing a refined active-set algorithm**

The active-set algorithm in [11] (QPNNLS) has proven to be very efficient in solving quadratic programs which arise in real-time MPC. Some of the insights in Paper C can be used to avoid superfluous computations in QPNNLS which stem from reformulating the QP as a nonnegative least-squares problem. This might result in an active-set QP method which is even more efficient than QPNNLS.
[1] Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012.

[2] Daniel Arnström and Daniel Axehill. Exact complexity certification of a standard primal active-set method for quadratic programming. In *IEEE 58th Conference on Decision and Control*, pages 4317–4324, Dec 2019.

[3] Daniel Arnström and Daniel 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.

[4] Daniel Arnström and Daniel Axehill. A unifying complexity certification framework for active-set methods for convex quadratic programming. *arXiv preprint arXiv:2003.07605*, 2020.

[5] Daniel Arnström, Alberto Bemporad, and Daniel Axehill. Exact complexity certification of a nonnegative least-squares method for quadratic programming. *IEEE Control Systems Letters*, 4(4):1036–1041, 2020.

[6] Daniel Arnström, Alberto Bemporad, and Daniel Axehill. Complexity certification of proximal-point methods for numerically stable quadratic programming. *IEEE Control Systems Letters*, 5(4):1381–1386, 2021.

[7] Daniel Axehill. Controlling the level of sparsity in MPC. *Systems & Control Letters*, 76:1–7, 2015.

[8] Daniel Axehill and Anders Hansson. A dual gradient projection quadratic programming algorithm tailored for model predictive control. In *2008 47th IEEE Conference on Decision and Control*, pages 3057–3064, 12 2008.

[9] Roscoe A Bartlett and Lorenz T Biegler. QPSchur: a dual, active-set, schur-complement method for large-scale and structured convex quadratic programming. *Optimization and Engineering*, 7(1):5–32, 2006.

[10] Amir Beck. *First-order methods in optimization*. SIAM, 2017.
[11] Alberto Bemporad. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, 61(4):1111–1116, 2015.

[12] Alberto Bemporad. A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. *IEEE Transactions on Automatic Control*, 63(2):525–531, 2017.

[13] Alberto Bemporad. Explicit model predictive control. In John Baillieul and Tariq Samad, editors, *Encyclopedia of Systems and Control*, pages 1–7. Springer London, London, 2019.

[14] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[15] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[16] Michael J Best. Equivalence of some quadratic programming algorithms. *Mathematical Programming*, 30(1):71, 1984.

[17] Michael J Best. An algorithm for the solution of the parametric quadratic programming problem. In *Applied Mathematics and Parallel Computing*, pages 57–76. Springer, 1996.

[18] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[19] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[20] Gionata Cimini and Alberto Bemporad. Exact complexity certification of active-set methods for quadratic programming. *IEEE Transactions on Automatic Control*, 62:6094–6109, 2017.

[21] George B Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.

[22] Luigi Del Re, Frank Allgöwer, Luigi Glielmo, Carlos Guardiola, and Ilya Kolmanovsky. *Automotive model predictive control: models, methods and applications*, volume 402. Springer, 2010.

[23] Stefano Di Cairano and Ilya V Kolmanovsky. Real-time optimization and model predictive control for aerospace and automotive applications. In *2018 Annual American Control Conference (ACC)*, pages 2392–2409. IEEE, 2018.

[24] William S Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162, 1960.
[25] Utku Eren, Anna Prach, Başaran Bahadır Koçer, Saša V Raković, Erdal Kayacan, and Behçet Açıkmeşe. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.

[26] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[27] Roger Fletcher. A general quadratic programming algorithm. *IMA Journal of Applied Mathematics*, 7(1):76–91, 1971.

[28] Roger Fletcher. Resolving degeneracy in quadratic programming. *Annals of Operations Research*, 46(2):307–334, 1993.

[29] Roger Fletcher and Sven Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization*, 8(2):604–616, 1998.

[30] Gianluca Frison and Moritz Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. *arXiv preprint arXiv:2003.02547*, 2020.

[31] Philip E Gill, Gene H Golub, Walter Murray, and Michael A Saunders. Methods for modifying matrix factorizations. *Mathematics of computation*, 28(126):505–535, 1974.

[32] Philip E Gill, Walter Murray, Michael A Saunders, and Margaret H Wright. A practical anti-cycling procedure for linearly constrained optimization. *Mathematical Programming*, 45(1):437–474, 1989.

[33] Donald Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[34] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

[35] Martin Herceg, CN Jones, and M Morari. Dominant speed factors of active set methods for fast MPC. *Optimal Control Applications and Methods*, 36(5):608–627, 2015.

[36] Michael Hintermüller, Kazufumi Ito, and Karl Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2002.

[37] Juan L Jerez, Paul J Goulart, Stefan Richter, George A Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, 2014.
[38] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In Traces and emergence of nonlinear programming, pages 247–258. Springer, 2014.

[39] Karl. Kunisch and Franz Rendl. An infeasible active set method for quadratic problems with simple bounds. SIAM Journal on Optimization, 14:35–52, 01 2003.

[40] Lennart Ljung. System identification. Wiley encyclopedia of electrical and electronics engineering, pages 1–19, 1999.

[41] David Q Mayne. Model predictive control: Recent developments and future promise. Automatica, 50(12):2967–2986, 2014.

[42] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. Automatica, 36(6):789–814, 2000.

[43] Ian McInerney, George A Constantinides, and Eric C Kerrigan. A survey of the implementation of linear model predictive control on FPGAs. IFAC-PapersOnLine, 51(20):381–387, 2018.

[44] Isak Nielsen and Daniel Axehill. Low-rank modifications of Riccati factorizations for model predictive control. IEEE Transactions on Automatic Control, 63(3):872–879, 2017.

[45] Jorge Nocedal and Stephen Wright. Numerical Optimization. Springer Science & Business Media, 2006.

[46] Panagiotis Patrinos and Alberto Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. IEEE Transactions on Automatic Control, 59:18–33, 01 2014.

[47] Michael James David Powell. On the quadratic programming algorithm of goldfarb and idnani. In Mathematical Programming Essays in Honor of George B. Dantzig Part II, pages 46–61. Springer, 1985.

[48] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. Control Engineering Practice, 11(7):733–764, 2003. ISSN 0967-0661.

[49] Christopher V Rao, Stephen J Wright, and James B Rawlings. Application of interior-point methods to model predictive control. Journal of Optimization Theory and Applications, 99(3):723–757, 1998.

[50] James B Rawlings, David Q Mayne, and Moritz Diehl. Model predictive control: theory, computation, and design, volume 2. Nob Hill Publishing Madison, WI, 2017.

[51] Stefan Richter, Colin N Jones, and Manfred Morari. Real-time input-constrained MPC using fast gradient methods. In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, pages 7387–7393. IEEE, 2009.
[52] Pierre OM Scokaert, David Q Mayne, and James B Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

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

[54] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, pages 1–36, 2020.

[55] Tamás Terlaky and Shuzhong Zhang. Pivot rules for linear programming: a survey on recent theoretical developments. *Annals of Operations Research*, 46(1):203–233, 1993.

[56] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, 18(2):267–278, 2010.

[57] Elizabeth Lai Sum Wong. *Active-set methods for quadratic programming*. PhD thesis, UC San Diego, 2011.

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

[59] Melanie N Zeilinger, Colin N Jones, and Manfred Morari. Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Transactions on Automatic Control*, 56:1524–1534, 07 2011.

[60] Alex Zheng and Manfred Morari. Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, 40(10):1818–1823, 1995.
Part II

Publications
The papers associated with this thesis have been removed for copyright reasons. For more details about these see:

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-173716
Licentiate Theses
Division of Automatic Control
Linköping University

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.
B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.
A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.
G. Malmberg: A Study of Adaptive Control Missiles. Thesis No. 76, 1986.
S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.
M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.
K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.
A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.
P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.
K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.
I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.
F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.
H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.
S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.
K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.
J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.
P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.
H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.
T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.
T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.
P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.
J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.
J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.
A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.
M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.
K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.
J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.

A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.

M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.

U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

M. Larsson: On Modeling and Diagnosis of Discrete Event Dynamic systems. Thesis No. 648, 1997.

N. Bergman: Bayesian Inference in Terrain Navigation. Thesis No. 649, 1997.

V. Einarsson: On Verification of Switched Systems Using Abstractions. Thesis No. 705, 1998.

J. Blom, F. Gunnarsson: Power Control in Cellular Radio Systems. Thesis No. 706, 1998.

P. Spångéus: Hybrid Control using LP and LMI methods – Some Applications. Thesis No. 724, 1998.

M. Norrlöf: On Analysis and Implementation of Iterative Learning Control. Thesis No. 727, 1998.

A. Hagenblad: Aspects of the Identification of Wiener Models. Thesis No. 793, 1999.

F. Tjärnström: Quality Estimation of Approximate Models. Thesis No. 810, 2000.

C. Carlsson: Vehicle Size and Orientation Estimation Using Geometric Fitting. Thesis No. 840, 2000.

J. Löfberg: Linear Model Predictive Control: Stability and Robustness. Thesis No. 866, 2001.

O. Härkegård: Flight Control Design Using Backstepping. Thesis No. 875, 2001.

J. Elbornsson: Equalization of Distortion in A/D Converters. Thesis No. 883, 2001.

J. Roll: Robust Verification and Identification of Piecewise Affine Systems. Thesis No. 899, 2001.

I. Lind: Regressor Selection in System Identification using ANOVA. Thesis No. 921, 2001.

R. Karlsson: Simulation Based Methods for Target Tracking. Thesis No. 930, 2002.

P.-J. Nordlund: Sequential Monte Carlo Filters and Integrated Navigation. Thesis No. 945, 2002.

M. Östring: Identification, Diagnosis, and Control of a Flexible Robot Arm. Thesis No. 948, 2002.

C. Olsson: Active Engine Vibration Isolation using Feedback Control. Thesis No. 968, 2002.

J. Jansson: Tracking and Decision Making for Automotive Collision Avoidance. Thesis No. 965, 2002.

N. Persson: Event Based Sampling with Application to Spectral Estimation. Thesis No. 981, 2002.

D. Lindgren: Subspace Selection Techniques for Classification Problems. Thesis No. 995, 2002.

E. Geijer Lundin: Uplink Load in CDMA Cellular Systems. Thesis No. 1045, 2003.

M. Enqvist: Some Results on Linear Models of Nonlinear Systems. Thesis No. 1046, 2003.

T. Schön: On Computational Methods for Nonlinear Estimation. Thesis No. 1047, 2003.

F. Gunnarsson: On Modeling and Control of Network Queue Dynamics. Thesis No. 1048, 2003.

S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.
M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

A. Eidehall: An Automotive Lane Guidance System. Thesis No. 1122, 2004.

E. Wernholt: On Multivariable and Nonlinear Identification of Industrial Robots. Thesis No. 1131, 2004.

J. Gillberg: Methods for Frequency Domain Estimation of Continuous-Time Models. Thesis No. 1133, 2004.

G. Hendeby: Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems. Thesis No. 1199, 2005.

D. Axehill: Applications of Integer Quadratic Programming in Control and Communication. Thesis No. 1218, 2005.

J. Sjöberg: Some Results On Optimal Control for Nonlinear Descriptor Systems. Thesis No. 1227, 2006.

D. Törnqvist: Statistical Fault Detection with Applications to IMU Disturbances. Thesis No. 1258, 2006.

H. Tidefelt: Structural algorithms and perturbations in differential-algebraic equations. Thesis No. 1318, 2007.

S. Moberg: On Modeling and Control of Flexible Manipulators. Thesis No. 1336, 2007.

J. Wallén: On Kinematic Modelling and Iterative Learning Control of Industrial Robots. Thesis No. 1343, 2008.

J. Harju Johansson: A Structure Utilizing Inexact Primal-Dual Interior-Point Method for Analysis of Linear Differential Inclusions. Thesis No. 1367, 2008.

J. D. Hol: Pose Estimation and Calibration Algorithms for Vision and Inertial Sensors. Thesis No. 1370, 2008.

H. Ohlsson: Regression on Manifolds with Implications for System Identification. Thesis No. 1382, 2008.

D. Ankelhed: On low order controller synthesis using rational constraints. Thesis No. 1398, 2009.

P. Skoglar: Planning Methods for Aerial Exploration and Ground Target Tracking. Thesis No. 1420, 2009.

C. Lundquist: Automotive Sensor Fusion for Situation Awareness. Thesis No. 1422, 2009.

C. Lyzell: Initialization Methods for System Identification. Thesis No. 1426, 2009.

R. Falkeborn: Structure exploitation in semidefinite programming for control. Thesis No. 1430, 2010.

D. Petersson: Nonlinear Optimization Approaches to $H_2$-Norm Based LPV Modelling and Control. Thesis No. 1453, 2010.

Z. Sjanić: Navigation and SAR Auto-focusing in a Sensor Fusion Framework. Thesis No. 1464, 2011.

K. Granström: Loop detection and extended target tracking using laser data. Thesis No. 1465, 2011.

J. Callmer: Topics in Localization and Mapping. Thesis No. 1489, 2011.

F. Lindsten: Rao-Blackwellised particle methods for inference and identification. Thesis No. 1480, 2011.

M. Skoglund: Visual Inertial Navigation and Calibration. Thesis No. 1500, 2011.

S. Khoshfetrat Pakazad: Topics in Robustness Analysis. Thesis No. 1512, 2011.

P. Axelsson: On Sensor Fusion Applied to Industrial Manipulators. Thesis No. 1511, 2011.

A. Carvalho Bittencourt: On Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1516, 2012.

P. Rosander: Averaging level control in the presence of frequent inlet flow upsets. Thesis No. 1527, 2012.
N. Wahlström: Localization using Magnetometers and Light Sensors. Thesis No. 1581, 2013.
R. Larsson: System Identification of Flight Mechanical Characteristics. Thesis No. 1599, 2013.
Y. Jung: Estimation of Inverse Models Applied to Power Amplifier Predistortion. Thesis No. 1605, 2013.
M. Syldatk: On Calibration of Ground Sensor Networks. Thesis No. 1611, 2013.
M. Roth: Kalman Filters for Nonlinear Systems and Heavy-Tailed Noise. Thesis No. 1613, 2013.
D. Simon: Model Predictive Control in Flight Control Design — Stability and Reference Tracking. Thesis No. 1642, 2014.
J. Dahlin: Sequential Monte Carlo for inference in nonlinear state space models. Thesis No. 1652, 2014.
M. Kok: Probabilistic modeling for positioning applications using inertial sensors. Thesis No. 1656, 2014.
J. Linder: Graybox Modelling of Ships Using Indirect Input Measurements. Thesis No. 1681, 2014.
G. Mathai: Direction of Arrival Estimation of Wideband Acoustic Wavefields in a Passive Sensing Environment. Thesis No. 1721, 2015.
I. Nielsen: On Structure Exploiting Numerical Algorithms for Model Predictive Control. Thesis No. 1727, 2015.
C. Veiback: Tracking of Animals Using Airborne Cameras. Thesis No. 1761, 2016.
N. Evestedt: Sampling Based Motion Planning for Heavy Duty Autonomous Vehicles. Thesis No. 1762, 2016.
H. Nyqvist: On Pose Estimation in Room-Scaled Environments. Thesis No. 1765, 2016.
Y. Zhao: Position Estimation in Uncertain Radio Environments and Trajectory Learning. Thesis No. 1772, 2017.
P. Kasebzadeh: Parameter Estimation for Mobile Positioning Applications. Thesis No. 1786, 2017.
K. Radnosrati: On Timing-Based Localization in Cellular Radio Networks. Thesis No. 1808, 2018.
G. Lindmark: Methods and Algorithms for Control Input Placement in Complex Networks. Thesis No. 1814, 2018.
M. Lindfors: Frequency Tracking for Speed Estimation. Thesis No. 1815, 2018.
D. Ho: Some results on closed-loop identification of quadcopters. Thesis No. 1826, 2018.
O. Ljungqvist: On motion planning and control for truck and trailer systems. Thesis No. 1832, 2019.
P. Boström-Rost: On Informative Path Planning for Tracking and Surveillance. Thesis No. 1838, 2019.
K. Bergman: On Motion Planning Using Numerical Optimal Control. Thesis No. 1843, 2019.
M. Klingspor: Low-rank optimization in system identification. Thesis No. 1855, 2019.
A. Bergström: Timing-Based Localization using Multipath Information. Thesis No. 1867, 2019.
F. Ljungberg: Estimation of Nonlinear Greybox Models for Marine Applications. Thesis No. 1880, 2020.
E. Hedberg: Control, Models and Industrial Manipulators. Thesis No. 1894, 2020.
R. Forsling: Decentralized Estimation Using Conservative Information Extraction. Thesis No. 1897, 2020.
