Structure-Exploiting Numerical Algorithms for Optimal Control

Isak Nielsen
Cover illustration: An illustration of the structure of the Karush-Kuhn-Tucker matrix for the unconstrained finite-time optimal control problems considered in this thesis. Thank you Jonas Linder for helping me with the design.
Tillägnas hela min underbara familj
Abstract

Numerical algorithms for efficiently solving optimal control problems are important for commonly used advanced control strategies, such as model predictive control (MPC), but can also be useful for advanced estimation techniques, such as moving horizon estimation (MHE). In MPC, the control input is computed by solving a constrained finite-time optimal control (CFTOC) problem on-line, and in MHE the estimated states are obtained by solving an optimization problem that often can be formulated as a CFTOC problem. Common types of optimization methods for solving CFTOC problems are interior-point (IP) methods, sequential quadratic programming (SQP) methods and active-set (AS) methods. In these types of methods, the main computational effort is often the computation of the second-order search directions. This boils down to solving a sequence of systems of equations that correspond to unconstrained finite-time optimal control (UFTOC) problems. Hence, high-performing second-order methods for CFTOC problems rely on efficient numerical algorithms for solving UFTOC problems. Developing such algorithms is one of the main focuses in this thesis. When the solution to a CFTOC problem is computed using an AS type method, the aforementioned system of equations is only changed by a low-rank modification between two AS iterations. In this thesis, it is shown how to exploit these structured modifications while still exploiting structure in the UFTOC problem using the Riccati recursion. Furthermore, direct (non-iterative) parallel algorithms for computing the search directions in IP, SQP and AS methods are proposed in the thesis. These algorithms exploit, and retain, the sparse structure of the UFTOC problem such that no dense system of equations needs to be solved serially as in many other algorithms. The proposed algorithms can be applied recursively to obtain logarithmic computational complexity growth in the prediction horizon length. For the case with linear MPC problems, an alternative approach to solving the CFTOC problem on-line is to use multiparametric quadratic programming (mp-QP), where the corresponding CFTOC problem can be solved explicitly off-line. This is referred to as explicit MPC. One of the main limitations with mp-QP is the amount of memory that is required to store the parametric solution. In this thesis, an algorithm for decreasing the required amount of memory is proposed. The aim is to make mp-QP and explicit MPC more useful in practical applications, such as embedded systems with limited memory resources. The proposed algorithm exploits the structure from the QP problem in the parametric solution in order to reduce the memory footprint of general mp-QP solutions, and in particular, of explicit MPC solutions. The algorithm can be used directly in mp-QP solvers, or as a post-processing step to an existing solution.
Numeriska algoritmer för att effektivt lösa optima styrningsproblem är en viktig komponent i avancerade regler- och estimeringsstrategier som exempelvis modell-prediktiv reglering (eng. model predictive control (MPC)) och glidande horisont estimering (eng. moving horizon estimation (MHE)). MPC är en reglerstrategi som kan användas för att styma system med flera styrsignaler och/eller utsignaler samt ta hänsyn till exempelvis begränsningar i styrdom. Den grundläggande principen för MPC och MHE är att styrsignalen och de estimerade variablerna kan beräknas genom att lösa ett optimalt styrningsproblem. Detta optimeringsproblem måste lösas inom en kort tidsram varje gång som en styrsignal ska beräknas eller som variabler ska estimeras, och således är det viktigt att det finns effektiva algoritmer för att lösa denna typ av problem. Två vanliga sådana är inrepunkts-metoder (eng. interior-point (IP)) och aktivmängd-metoder (eng. active-set (AS)), där optimeringsproblemet lösas genom att lösa ett antal enklare delproblem. Ett av huvudfokusen i denna avhandling är att beräkna lösningen till dessa delproblem på ett tidseffektivt sätt genom att utnyttja strukturen i delproblemen.


Linjära MPC problem kan också lösas genom att utnyttja teori från multiparametrisk kvadratisk programmering (eng. multiparametric quadratic programming (mp-QP)) där den optima lösningen beräknas i förhand och lagras i en tabell, vilket benämns explicit MPC. I detta fall behöver inte MPC problemet lösas varje gång en styrsignal beräknas, utan istället kan den förberäknade optimala styrsignalen släss upp. En nackdel med mp-QP är att det krävs mycket plats i minnet för att spara lösningen. I denna avhandling presenteras en strukturerutnyttjande algoritm som kan minska behovet av minne för att spara lösningen, vilket kan öka det praktiska användningsområdet för mp-QP och explicit MPC.
Acknowledgments

I would like to take the opportunity to express my deepest gratitude to my supervisor Dr. Daniel Axehill. Your guidance during my almost five years as a PhD student has been invaluable to me. Thank you for inspiring and motivating me in my research, and thank you for your excellent support and your never-ending enthusiasm. I am happy to have had you as my supervisor, but also to have you as a friend. Furthermore, I am very grateful for the valuable feedback and support I have received from my co-supervisor (former supervisor) Prof. Anders Hansson.

Also, thank you Prof. Svante Gunnarsson and Prof. Fredrik Gustafsson for inviting me to join the Automatic Control group as a PhD student. I would also like to thank Prof. Svante Gunnarsson for being an excellent head of the Division of Automatic Control, and Ninna Stensgård for helping me and others with administrative tasks, and for making everything run smoothly in the group.

This thesis has been significantly improved by constructive feedback from Dr. Johan Dahlin, Dr. Jonas Linder, Lic. Hanna Nyqvist, M.Sc. Oskar Ljungqvist, M.Sc. Kristoffer Bergman, M.Sc. Per Boström, and M.M.Sc. Linnea Ernebro. I am sincerely grateful that you spent some of your valuable time on proof-reading this thesis! Also, thank you Dr. Gustaf Hendeby and Dr. Henrik Tidefelt for providing the \LaTeX-class that has been used to write this thesis, and thank you Dr. Jonas Linder for helping me with the cover design and various \LaTeX issues.

During my time as a PhD student at the Division of Automatic Control I have had the pleasure of making a lot of new friends. Thank you Jonas, Johan D., and Hanna for being really good friends, and thank you Jonas for four fun years sharing office with you! Also, thank you Niclas E., Clas, Sina, André, Karl, Zoran, Daniel S., Emre, Tohid, Manon, Ylva, Patrik, Niklas W., Oskar, Martin L., Kamiar, Parinaz, Erik, Kristoffer, Per and Angela for being good friends and colleagues, and for sharing fun skiing and canoeing trips, for doing wine tasting and safari trips, for visiting Grand Canyon and speeding through Death Valley with an injured stranger in the car, and for much more. And thank you everyone else at the Automatic Control group for making this a stimulating and friendly working environment!

I would also like to thank all of my old and new friends outside the university for all the fun stuff we do together. It means a lot to me to have you as friends!

Financial support from the Swedish Research Council (VR) and CENIT (Center for Industrial Information Technology) are hereby gratefully acknowledged.

Finally, thank you Linnea for supporting me and cheering me up, for loving me and making me happy, and for letting me share all adventures in life with you! And to all of my family, which also includes Linnea’s family: Thank you for all your love and support, for always being there for me when I need it the most, and for being the amazing persons you are. I love you all!

Linköping, April Anno Domini 2017
Isak Nielsen
## Contents

### Notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background and Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Some methods for solving optimal control problems</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Related work</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Objectives</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Publications</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1 Publications included in the thesis</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2 Publications not included in the thesis</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Contributions</td>
<td>7</td>
</tr>
<tr>
<td>1.5 Outline of the Thesis</td>
<td>7</td>
</tr>
</tbody>
</table>

### 2 Optimization

<table>
<thead>
<tr>
<th>2 Optimization</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Basic Concepts</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Convex Optimization</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Lagrange Duality</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 Weak and strong duality</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Optimality Conditions</td>
<td>13</td>
</tr>
<tr>
<td>2.5 Quadratic Programming</td>
<td>14</td>
</tr>
<tr>
<td>2.6 Active-Set Methods for Quadratic Programming</td>
<td>14</td>
</tr>
<tr>
<td>2.6.1 Adding constraints to the working set</td>
<td>16</td>
</tr>
<tr>
<td>2.6.2 Removing constraints from the working set</td>
<td>17</td>
</tr>
<tr>
<td>2.6.3 Basic primal active-set method</td>
<td>18</td>
</tr>
<tr>
<td>2.7 Interior-Point Methods</td>
<td>19</td>
</tr>
<tr>
<td>2.7.1 Basic primal interior-point method</td>
<td>20</td>
</tr>
</tbody>
</table>

### 3 Optimal Control

<table>
<thead>
<tr>
<th>3 Optimal Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Constrained Finite-Time Optimal Control Problem</td>
<td>23</td>
</tr>
<tr>
<td>3.1.1 Model predictive control</td>
<td>24</td>
</tr>
<tr>
<td>3.1.2 Moving horizon estimation</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Solving Constrained Finite-Time Optimal Control Problems</td>
<td>29</td>
</tr>
</tbody>
</table>
3.2.1 Interior-point methods .............................................. 30
3.2.2 Active-set methods ........................................... 33
3.3 Computing the Newton Step .................................. 34
3.4 Solving the Karush-Kuhn-Tucker System using the Riccati Recursion ................. 37
  3.4.1 Derivation of the Riccati recursion ......................... 37
  3.4.2 Handling singular Karush-Kuhn-Tucker systems .......... 41
3.5 Computing the Eliminated Dual Variables ...................... 43
3.A Model Predictive Control Formulations .......................... 45
3.B Moving Horizon Estimation Formulations ..................... 46

4 Multiparametric Quadratic Programming .......................... 47
  4.1 Problem Definition .................................................. 47
  4.2 Characterizing the Solution for an Optimal Active Set ........... 49
  4.3 Characterizing the Critical Region .............................. 50
  4.4 Explicit Model Predictive Control ............................... 50
  4.A Explicit Model Predictive Control Formulations ............. 52

5 Low-Rank Modifications of the Riccati Factorization ............ 53
  5.1 Derivation of the Low-Rank Modifications ..................... 54
  5.1.1 Sequence of low-rank modifications ....................... 55
  5.1.2 Removing control input constraints from the working set .... 59
  5.1.3 Adding control input constraints to the working set ....... 61
  5.2 An Algorithm for Modifying the Riccati Factorization ........ 63
  5.3 Extension to more General Inequality Constraints ........... 65
  5.3.1 Primal and dual problem ..................................... 65
  5.3.2 Solving the primal problem by exploiting its dual problem ... 67
  5.4 Numerical Results ................................................. 71
  5.A Proof of Lemma 5.1 .............................................. 80
  5.B Derivation of the Dual Problem ................................... 81
  5.B.1 Constructing the dual problem ............................... 81
  5.B.2 Relation between the primal and dual variables ........... 85

6 Parallel Computations of Search Directions ........................ 89
  6.1 Computing the Search Direction via a Master Problem ........ 90
  6.2 Parametric Programming Approach .............................. 91
    6.2.1 Problem decomposition and variable elimination .......... 92
    6.2.2 Constructing the master problem ......................... 97
    6.2.3 Computing the solution ................................... 98
    6.2.4 Communication ............................................. 99
  6.3 Parallel Partial Condensing and Reduction Approach .......... 100
    6.3.1 Decomposition into subproblems ......................... 102
    6.3.2 Condensing a subproblem ................................. 103
    6.3.3 Reducing the control input dimension in a subproblem .... 107
    6.3.4 Constructing the master problem ......................... 114
    6.3.5 Solving the subproblems using the solution to the master problem ... 117
    6.3.6 Solving the subproblems using the cost-to-go function .... 119
6.3.7 Communication .................................................. 119
6.4 Computing the Search Direction in Parallel .......................... 120
6.4.1 Algorithms for parallel computation of the search directions ...... 124
6.5 Numerical Results .................................................. 127
6.A Definition of Problem Matrices ...................................... 133
6.A.1 Definition of the matrices in the UFTOC problem (6.34) ........ 133
6.A.2 Definition of the matrices used in Lemma 6.9 ................... 134
6.A.3 Definition of the matrices used in Lemma 6.12 ................... 134
6.B Proofs of theorems and lemmas in Section 6.2 ...................... 135
6.B.1 Proof of Lemma 6.4 ............................................ 135
6.B.2 Proof of Theorem 6.7 ........................................... 136
6.B.3 Proof of Lemma 6.6 ............................................ 138
6.C Proofs of lemmas in Section 6.3 .................................. 139
6.C.1 Proof of Lemma 6.9 ............................................ 139
6.C.2 Proof of Lemma 6.11 ........................................... 140
6.C.3 Proof of Lemma 6.12 ........................................... 142

7 Exploiting Low-Rank Structure in Multiparametric Quadratic Programming 145
7.1 Low-Rank Modifications of the Parametric Solution .................. 146
7.1.1 Adding one constraint to the optimal active set ................. 147
7.1.2 Removing one constraint from the optimal active set ............ 149
7.2 Memory Efficient Storage Tree .................................... 150
7.2.1 Computing the parametric solution and critical region .......... 152
7.2.2 Storing the parametric solution and the critical regions ......... 154
7.3 Experimental Evaluation ........................................... 156
7.3.1 Defining the problems ........................................ 156
7.3.2 Experimental results .......................................... 157

8 Conclusions and Future Work 161
8.1 Conclusions ...................................................... 161
8.2 Future Work ........................................................ 162

A Linear Algebra 165

Bibliography 171
## Notation

### Sets, spaces and subspaces

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{R}$</td>
<td>The set of real numbers</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>The set of integers</td>
</tr>
<tr>
<td>$\mathbb{Z}_+$</td>
<td>The set of positive integers</td>
</tr>
<tr>
<td>$\mathbb{Z}_{i,j}$</td>
<td>The set of integers ${z \in \mathbb{Z}\mid i \leq z \leq j}$</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>The $n$-dimensional Euclidean space</td>
</tr>
<tr>
<td>$\mathbb{R}^{n \times m}$</td>
<td>The space of real matrices with $n$ rows and $m$ columns</td>
</tr>
<tr>
<td>$\mathcal{S}_n^+$</td>
<td>The set of positive semidefinite matrices with $n$ columns</td>
</tr>
<tr>
<td>$\mathcal{S}_n^{++}$</td>
<td>The set of positive definite matrices with $n$ columns</td>
</tr>
<tr>
<td>$[a, b]$</td>
<td>The interval of real numbers $r \in \mathbb{R}$ such that $a \leq r \leq b$</td>
</tr>
<tr>
<td>$(a, b)$</td>
<td>The interval of real numbers $r \in \mathbb{R}$ such that $a &lt; r &lt; b$</td>
</tr>
<tr>
<td>$\mathcal{R}(A)$</td>
<td>The range space of a matrix $A$</td>
</tr>
<tr>
<td>$\mathcal{N}(A)$</td>
<td>The null space of a matrix $A$</td>
</tr>
<tr>
<td>$C^c$</td>
<td>The complement of the set $C$</td>
</tr>
<tr>
<td>$\mathcal{S}^\perp$</td>
<td>The orthogonal complement of a subspace $\mathcal{S}$</td>
</tr>
<tr>
<td>relint $C$</td>
<td>The relative interior of the set $C$</td>
</tr>
<tr>
<td>$\text{dom } f$</td>
<td>The domain of the function $f : \mathbb{R}^n \to \mathbb{R}^m$</td>
</tr>
</tbody>
</table>

### Symbols, operators and functions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \succeq 0$</td>
<td>The matrix $A$ is positive semidefinite</td>
</tr>
<tr>
<td>$A &gt; 0$</td>
<td>The matrix $A$ is positive definite</td>
</tr>
<tr>
<td>$x \succeq y$</td>
<td>$x_i \geq y_i$ holds for each pair of components $x_i$ and $y_i$</td>
</tr>
<tr>
<td>$x \succ y$</td>
<td>$x_i &gt; y_i$ holds for each pair of components $x_i$ and $y_i$</td>
</tr>
<tr>
<td>rank $A$</td>
<td>The rank of a matrix $A$</td>
</tr>
<tr>
<td>colrank $A$</td>
<td>The column rank of a matrix $A$</td>
</tr>
</tbody>
</table>
### Symbols, operators and functions, cont’d

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{blkdiag}(\cdot)$</td>
<td>A block diagonal matrix with the entries as the diagonal blocks</td>
</tr>
<tr>
<td>$\forall x$</td>
<td>For all $x$</td>
</tr>
<tr>
<td>$S_1 \oplus S_2$</td>
<td>The orthogonal sum of $S_1$ and $S_2$</td>
</tr>
<tr>
<td>$S_1 \subset S_2$</td>
<td>Denotes that $\forall x \in S_1 \implies x \in S_2$</td>
</tr>
<tr>
<td>$x \triangleq y$</td>
<td>Definition of the variable $x$ as $y$</td>
</tr>
<tr>
<td>$\nabla f$</td>
<td>The gradient of the function $f: \mathbb{R}^n \to \mathbb{R}$</td>
</tr>
<tr>
<td>$\nabla^2 f$</td>
<td>The Hessian of the function $f: \mathbb{R}^n \to \mathbb{R}$</td>
</tr>
<tr>
<td>$\nabla_x f$</td>
<td>The gradient of the function $f: \mathbb{R}^n \to \mathbb{R}$ with respect to the variable $x$</td>
</tr>
<tr>
<td>$\nabla^2_x f$</td>
<td>The Hessian of the function $f: \mathbb{R}^n \to \mathbb{R}$ with respect to the variable $x$</td>
</tr>
<tr>
<td>$I(I_n)$</td>
<td>The identity matrix (of size $n$)</td>
</tr>
<tr>
<td>$\mathbf{1}(\mathbf{1}_n)$</td>
<td>The vector (in $\mathbb{R}^n$) with all components equal to 1</td>
</tr>
<tr>
<td>$O(g(x))$</td>
<td>$f(x) = O(g(x)) \implies \frac{f(x)}{g(x)} \leq C$ when $x \to \infty$ for a constant $C &gt; 0$</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>Active-Set</td>
</tr>
<tr>
<td>CFTOC</td>
<td>Constrained Finite-Time Optimal Control</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DP</td>
<td>Dynamic Programming</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>IP</td>
<td>Interior-Point</td>
</tr>
<tr>
<td>LICQ</td>
<td>Linear Independence Constraint Qualification</td>
</tr>
<tr>
<td>KKT</td>
<td>Karush-Kuhn-Tucker</td>
</tr>
<tr>
<td>LQ</td>
<td>Linear-Quadratic</td>
</tr>
<tr>
<td>MHE</td>
<td>Moving Horizon Estimation</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>mp-QP</td>
<td>Multiparametric Quadratic Programming</td>
</tr>
<tr>
<td>NLP</td>
<td>Nonlinear Programming</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Programming</td>
</tr>
<tr>
<td>PWA</td>
<td>Piece-Wise Affine</td>
</tr>
<tr>
<td>SC</td>
<td>Strict Complementarity</td>
</tr>
<tr>
<td>SQP</td>
<td>Sequential Quadratic Programming</td>
</tr>
<tr>
<td>UFTOC</td>
<td>Unconstrained Finite-Time Optimal Control</td>
</tr>
</tbody>
</table>
Introduction

One of the main goals in automatic control is to regulate systems and processes in order to obtain a desired behaviour. Today, automatic control in various forms has become an important part of the modern society. It is used in a wide variety of applications, ranging from more traditional ones such as process industry and low-level control in flight and automotive applications, to more modern applications such as high-level planning and control of advanced autonomous vehicles. In all these types of applications, some kind of controller is used to obtain the desired behaviour of the system. Depending on the application and controlled system, the controller can be of different types and complexities. One framework for designing and synthesizing controllers for systems is optimal control, where the controller is constructed to obtain the desired behaviour in an optimal sense for some given performance criterion.

In this thesis, there are two main focuses: structure-exploiting numerical algorithms that can be used as important subroutines in second-order methods for solving optimal control problems, and structure-exploiting algorithms for reducing the memory that is required to store the solutions to multiparametric quadratic programming (mp-QP) problems. In this chapter, the background and motivation for the research leading to the contributions included in this thesis are presented together with related work in the research area. Furthermore, the objectives with the thesis, the publications that the thesis is based on and the main contributions in the thesis are listed. The chapter is concluded with an outline of the thesis.
1 Introduction

1.1 Background and Motivation

In optimal control theory, the fields of optimization and automatic control are combined into a framework for computing optimal control inputs to the system that is being controlled. Optimal control is widely applicable, and can be useful both in traditional control applications and in more recent applications such as motion planning for autonomous systems. The optimal control problem can be defined in continuous time or in discrete time, depending on the application. Discrete-time optimal control problems also arise when continuous-time optimal control problems are discretized (Jørgensen, 2004). In this thesis, discrete-time optimal control problems in the form of constrained finite-time optimal control (CFTOC) problems are considered. Hence, when the term “CFTOC problem” is used, it is assumed to be a discrete-time problem unless stated otherwise.

One variant of optimal control with great impact on industry is model predictive control (MPC). Essentially, in an MPC controller a CFTOC problem consisting of a dynamic model of the system, constraints on the states and control inputs, and an objective function, is solved at each iteration in the control loop. The dynamic model is used to predict the behaviour of the system, and the control input is computed by minimizing the objective function while satisfying the constraints. The possibility to easily handle multivariable dynamic systems, and constraints on states and control inputs has made MPC one of the most widely spread and commonly used advanced control strategies in industry (Maciejowski, 2002). Another application where CFTOC problems can be important is moving horizon estimation (MHE). In MHE, the state estimates are computed by solving an optimization problem, which often can be re-formulated into a CFTOC problem, in a receding horizon fashion similar as in MPC (Rao, 2000; Jørgensen, 2004).

Traditionally, MPC has been used mainly in petrochemical plants where the sample time of the controller is long enough to manage to compute a solution to the CFTOC problem in real-time. However, as hardware and algorithms for solving the CFTOC problems have developed, it has become possible to apply MPC and MHE to other processes that require much shorter sampling times or to more complex systems. Since the CFTOC problem is usually solved on-line, the applicability of MPC and MHE heavily relies on efficient optimization routines for solving CFTOC problems.

1.1.1 Some methods for solving optimal control problems

There exist many different methods to numerically solve optimal control problems, where some are surveyed in for example Polak (1973), Bertsekas (2000), Betts (2001), and Rao (2009). One common approach to solve continuous-time optimal control problems is to use direct methods, where the optimal control problem is transcribed into a nonlinear programming (NLP) problem (Bock and Plitt, 1984; Betts, 2001). The transcription can be done using for example multiple shooting methods (Barclay et al., 1998; Diehl et al., 2006). For CFTOC problems, the NLP problem can be directly formulated from the definition of the CFTOC problem (Friesz, 2010). Some examples of second-order methods that can be used for solving NLP problems are interior-point (IP) methods, and active-set (AS) methods.
such as sequential quadratic programming (SQP) methods. Which type of method that is preferred can be dependent on for example the type of system, which type of constraints that are used, and in which application the optimal control problem arises. For the details of different methods for solving NLP problems, see for example Barclay et al. (1998), Betts (2001), Boyd and Vandenberghe (2004), or Nocedal and Wright (2006).

In many cases, the main computational effort that is spent when solving CFTOC problems using IP, AS, and SQP methods boils down to computing second-order search directions. This is done by solving a sequence of Newton-system-like equations that correspond to unconstrained finite-time optimal control (UFTOC) problems (Rao et al., 1998; Barclay et al., 1998; Betts, 2001; Vandenberghe et al., 2002; Kirches et al., 2012). Hence, many commonly used second-order methods for solving CFTOC problems rely on highly efficient algorithms for computing the solutions to UFTOC problems.

Today there exist several popular software packages for solving NLP problems. Some of these are general purpose NLP solvers such as SNOPT (Gill et al., 2000, 2002), which is based on an SQP framework, or IPOPT (Wächter and Biegler, 2006) which is based on an IP framework. Others are developed primarily for optimal control problems, such as ACADO (Houska et al., 2011) which provides a framework for using a variety of algorithms for direct optimal control.

When considering MPC problems for linear time-invariant (LTI) systems, there is an alternative approach to solving the CFTOC problem on-line. It was shown in Bemporad et al. (2002) that the CFTOC problem can be formulated as an MP-QP problem, and that the solution to the CFTOC problem can be computed as a piecewise affine (PWA) function of the initial state. This is denoted explicit MPC, and the on-line work to compute the optimal control input is reduced to looking up the explicit solution of the CFTOC problem which was computed off-line.

### 1.1.2 Related work

Much effort in research has been spent on developing algorithms that efficiently solve UFTOC problems, which consequently can be used to computing the search directions in many popular second-order methods for optimal control. Examples of such approaches are to use algorithms that exploit the structure of the problem, to perform modifications of the factorizations that are involved when solving the UFTOC problems, or to use parallel computations in different ways. Examples of algorithms for efficiently solving UFTOC problems are presented in Jonson (1983), Rao et al. (1998), Hansson (2000), Vandenberghe et al. (2002), Jørgensen (2004), Åkerblad and Hansson (2004), Axehill and Hansson (2006), Axehill (2008), Axehill and Hansson (2008), Diehl et al. (2009), Wang and Boyd (2010), Axehill et al. (2010), Jerez et al. (2012), Axehill and Morari (2012), Domahidi et al. (2012), Frison and Jørgensen (2013a), Frison (2015), and Klintberg and Gros (2017).

One way of exploiting the structure from the UFTOC problem is to use the Riccati recursion to compute the solution. One early publication where the Riccati recursion is exploited in an AS method for optimal control is Jonson (1983). In
this reference, a Riccati recursion is used to factor one part of the Karush-Kuhn-Tucker (KKT) matrix. For the other part, standard low-rank modifications of factorizations are used on a dense system of equations of the size of the number of active inequality constraints. The computational complexity of this algorithm grows quadratically in the number of active inequality constraints. An alternative sparse non-Riccati factorization is used in Kirches et al. (2011), where the factorization is modified after changes in the AS iterations.

A parallel algorithm for solving UFTOC problems is partitioned dynamic programming (DP) for optimal control, which is presented in Wright (1991). The partitioned DP algorithm decomposes the UFTOC problem in time and constructs a master problem. To solve the master problem, another type of algorithm is applied recursively to obtain $O(\log N)$ computational complexity growth. In Soudbaksh and Annaswamy (2013) an extended parallel cyclic reduction algorithm is used to reduce the computations to smaller systems of equations that are solved in parallel. The computational complexity of this algorithm is also reported to be $O(\log N)$. In Zhu and Laird (2008), Laird et al. (2011), and Reuterswärd (2012) a time-splitting approach to split the prediction horizon into blocks is adopted. The subproblems in the blocks are connected through complicating variables and are solved in parallel using Schur complements. The complicating variables are computed via a consensus step, where a dense system of equations involving all complicating variables has to be solved serially. In Frison and Jørgensen (2013b) different numerical libraries for exploiting parallel linear algebra in the Riccati recursion are evaluated. Although parallelism for MPC and the corresponding CFTOC and UFTOC problems is a well developed research area, parallelism for MHE is a less explored field. However, a parallel algorithm that combines point-based search with a derivative-based Gauss-Newton method for solving MHE problems is proposed in Poloni et al. (2013).

Instead of solving the UFTOC problem in parallel, the CFTOC problem can itself be decomposed and solved in parallel. In Chang and Luh (1985) and Chang et al. (1990) a discrete-time optimal control problem is decomposed in time into several smaller subproblems, where an incentive coordination between the subproblems is obtained using an additive coordination term in the cost function of each subproblem. The optimal control problem is solved by iteratively solving a master problem and computing the optimal coordination terms. In Chang (1986) and Chang et al. (1989) a hierarchical decomposition approach for large-scale optimal control problems is presented, where the optimal control problem is decomposed in time but the coordination is achieved by choosing the initial and terminal states in each subproblem. The initial and terminal states form a master problem, and the optimal control problem is solved by iteratively solving the master problem and the subproblems to determine the optimal initial and terminal states. In primal decomposition, as described in Lasdon (1970) and Boyd et al. (2008), a convex optimization problem can be decomposed into several smaller subproblems that only share complicating variables and complicating constraints. The complicated variables are solved by iteratively solving the master problem and the subproblems.
In nonserial DP, methods that describe the connection between variables in the problem using tree representations are used, see for example Bertelè and Brioschi (1973), Moallemi (2007), and Shcherbina (2007). Nonserial DP shares the basic ideas with serial DP, see for example Bertsekas (2000), but can handle more general problem structures. In Khoshfetrat Pakazad (2015) and Khoshfetrat Pakazad et al. (2017) a message-passing algorithm for IP methods is presented. This algorithm extends the framework presented in Nielsen and Axehill (2014), where a direct (non-iterative) parallel algorithm for solving UFTOC problems using a parametric programming approach is presented. The message-passing algorithm utilizes tree structure representations similar to those in nonserial DP, and can be used to solve more general types of problems than the UFTOC problem considered in Nielsen and Axehill (2014).

In mp-QP and explicit MPC, one of the main limitations is the amount of memory that is required to store the parametric solution and the critical regions (Bemporad et al., 2002; Fuchs et al., 2010; Kvasnica et al., 2015). Hence, much effort in research has been spent on finding remedies to this limitation. One such algorithm is proposed in Borrelli et al. (2010), where explicit MPC and on-line MPC are combined. In that work, the main algorithm is similar to a standard AS method, such as the one presented in for example Nocedal and Wright (2006), but the search directions are computed off-line for all optimal active sets. In Kvasnica et al. (2012) a PWA function, which is only defined over the regions with non-saturated control inputs, is combined with a projection onto a non-convex set to reduce the memory footprint in explicit MPC. The method of implicitly enumerating all optimal active sets proposed in Gupta et al. (2011) and the semi-explicit approach in Borrelli et al. (2010) are combined in Kvasnica et al. (2015), where an algorithm which reduces the amount of memory that is required to store the solution is proposed.

### 1.2 Objectives

The objectives of this thesis are twofold:

- To improve the computational performance of methods for numerical optimal control by exploiting problem structure and parallel hardware in order to reduce the computation times for important time-consuming subroutines in popular second-order methods.

- To reduce the memory that is required to store the parametric solution and critical regions in mp-QP and explicit MPC, with the aim of making them more useful in practical applications where for example hardware with limited memory resources are used.

### 1.3 Publications

The thesis is based on both the publications listed in Section 1.3.1 and on new material. The new material is mainly introduced in Section 6.3.
1.3.1 Publications included in the thesis

The thesis is based on the following list of publications. The author of this thesis is the main contributor in all of these publications.


Parts of the material in these publications are included in this thesis with kind permissions from the copyright owners.

Furthermore, some of the material in this thesis has already been published in the author’s Licentiate’s thesis


1.3.2 Publications not included in the thesis

The following publications are authored or co-authored by the author of this thesis, but the results are not included in this thesis.


### 1.4 Contributions

The contributions in this thesis are structure-exploiting algorithms that can be implemented and used as important subroutines in second-order methods for solving common problem classes arising in optimal control, and an algorithm for reducing the memory footprint of mp-QP solutions. Specifically, the main contributions in this thesis are:

- Theory and algorithms for modifying the Riccati factorization when it is used to compute the search directions in an AS type solver applied to solve CFTOC problems. It is shown how to handle the case with a singular KKT matrix, which can be important in for example dual gradient projection solvers.

- Theory and algorithms for parallel computations of the search directions in IP and AS type methods applied to solve CFTOC problems. Different algorithms that exploit and maintain the problem structure are proposed, and they obtain logarithmic computational complexity growth in the prediction horizon length.

- Theory and algorithms for exploiting structure in the parametric solution between neighboring critical regions for an mp-QP problem to reduce the amount of memory that is required to store the solution.

### 1.5 Outline of the Thesis

In Chapter 2, an introduction to optimization theory and algorithms is presented. The chapter is intended to introduce the notation and concepts from optimization theory that are required to present the contributions in the thesis. In Chapter 3, optimal control theory is briefly surveyed and the type of problems that are considered in this thesis is introduced. Furthermore, in that chapter it is motivated why the contributions in this thesis are important for a wide range of applications. The mp-QP problem is introduced in Chapter 4, where also the basic notation and properties of the parametric solution are presented.

The main contributions are included in chapters 5-7. Theory on how to modify the Riccati factorization when used in an AS type solver is presented in Chapter 5, and parallel algorithms for computations of second-order search directions are presented in Chapter 6. In Chapter 7, an algorithm for reducing the memory that is required to store the parametric solution to an mp-QP problem is presented. The conclusions of the thesis are presented in Chapter 8, where also some future work is listed. In Appendix A, useful results from linear algebra are included.
Mathematical optimization, or simply optimization, is a framework where the best possible solution to a problem is sought for. The best solution is defined by the corresponding optimization problem, which consists of an objective function and constraints that define the feasible set of the problem, i.e., which solutions that are allowed. The best solution is the element that is in the feasible set of the optimization problem, and either minimizes or maximizes the objective function, depending on the problem formulation. Depending on the optimization problem that is solved, it can exist a unique solution, several solutions or no solution.

Three examples of optimization problems are driving a car between two cities as fast (i), as cheap (ii) or by driving as short distance as possible (iii), while keeping the speed limits and driving on the road. The objective function in the first problem is the total time consumed when travelling between the two cities, and in the second problem it is the total cost for fuel, wear, etc. In the third problem, the objective function is the total distance travelled. The constraints in all three problems are typically the same, to keep the speed limits, to stay on the road and physical constraints such as the maximum power given by the car engine. These three objective functions represent three different optimization problems, all having (possibly) different optimal solutions. For example, the problem of driving as short distance as possible has a unique solution if there is only one road that gives the shortest distance. If there are several roads between the cities of equal minimal distance, then there exist several solutions where each is equally good. If there are no roads between the two cities the problem does not have any solution.

The purpose of this chapter is to introduce the basic concepts of optimization theory, and to give a brief survey of some common families of optimization methods that are used to compute the optimal solution to certain types of optimization problems. The chapter and the introduced notation are inspired by Boyd and Vandenberghe (2004), which is an extensive reference on convex optimization.
2.1 Basic Concepts

Consider an optimization problem given in the form

\[
\begin{align*}
\text{minimize} \quad & f_0(x) \\
\text{subject to} \quad & f_i(x) \leq 0, \ i \in \mathbb{Z}_{1,m} \\
& h_i(x) = 0, \ i \in \mathbb{Z}_{1,p},
\end{align*}
\]  

(2.1)

where \( x \in \mathbb{R}^n \) is the optimization variable, \( f_0 : \mathbb{R}^n \to \mathbb{R} \) is the objective function, \( f_i : \mathbb{R}^n \to \mathbb{R} \) for \( i \in \mathbb{Z}_{1,m} \) are the inequality constraint functions, and \( h_i : \mathbb{R}^n \to \mathbb{R} \) for \( i \in \mathbb{Z}_{1,p} \) are the equality constraint functions. When the objective function or constraint functions in (2.1) are not linear, the problem is referred to as a nonlinear program (NLP). The domain \( D \subset \mathbb{R}^n \) of the optimization problem (2.1) is the set of all points where the objective and constraint functions are defined, \( i.e., \)

\[
D \triangleq \bigcap_{i=0}^{m} \text{dom } f_i \cap \bigcap_{i=1}^{p} \text{dom } h_i.
\]  

(2.2)

A point \( x \in D \) is said to be feasible if it satisfies the inequality and equality constraints in (2.1). If there exists at least one feasible \( x \in D \) the optimization problem (2.1) is said to be feasible, and infeasible otherwise. The set of all feasible points is denoted the feasible set.

The optimal value \( p^* \) is the infimal value of the objective function evaluated at all feasible points and is defined as

\[
p^* \triangleq \inf \{ f_0(x) \mid f_i(x) \leq 0, \ i \in \mathbb{Z}_{1,m}, \ h_i(x) = 0, \ i \in \mathbb{Z}_{1,p} \},
\]  

(2.3)

where \( p^* = \infty \) if the optimization problem is infeasible, and \( p^* = -\infty \) if the optimization problem is unbounded from below. A feasible point \( x^* \in D \) with \( f_0(x^*) = p^* \) is called an optimal point or an optimal solution to (2.1).

An optimization problem

\[
\begin{align*}
\text{minimize} \quad & \tilde{f}_0(\tilde{x}) \\
\text{subject to} \quad & \tilde{f}_i(\tilde{x}) \leq 0, \ i \in \mathbb{Z}_{1,\tilde{m}} \\
& \tilde{h}_i(\tilde{x}) = 0, \ i \in \mathbb{Z}_{1,\tilde{p}},
\end{align*}
\]  

(2.4)

is said to be equivalent to (2.1) if \( x^* \) can be trivially computed from the solution \( \tilde{x}^* \), and vice versa. Hence, the solution to the original problem can be computed by solving an equivalent problem in a simpler form.

2.2 Convex Optimization

Convex optimization problems is a class of optimization problems that have some special properties which makes them useful in many different areas such as economics, mechanics, control engineering and circuit design (Boyd and Vandenberghe, 2004). Before the definition of a convex optimization problem can be presented,
some auxiliary definitions are required. Their properties are thoroughly discussed in standard literature such as Boyd and Vandenberghe (2004), and are repeated in definitions 2.1-2.3 for completeness.

**Definition 2.1 (Convex set).** A set $C$ is convex if for any two points $x, y \in C$ and any $\theta \in [0, 1]$ it holds that
\[
\theta x + (1 - \theta)y \in C. \tag{2.5}
\]

**Definition 2.2 (Convex function).** A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\text{dom } f$ is a convex set and if it for all points $x, y \in \text{dom } f$ and any $\theta \in [0, 1]$ holds that
\[
f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y), \tag{2.6}
\]
and strictly convex if strict inequality holds for $x \neq y$ and $\theta \in (0, 1)$.

**Definition 2.3 (Concave function).** A function $f : \mathbb{R}^n \to \mathbb{R}$ is concave if $-f$ is convex and strictly concave if $-f$ is strictly convex.

Now consider an optimization problem in the form (2.1) but where the equality constraint functions are affine, given by
\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq 0, \ i \in \mathbb{Z}_{1,m} \\
& \quad a_i^T x = b_i, \ i \in \mathbb{Z}_{1,p},
\end{align*} \tag{2.7}
\]
where the functions $f_i$ for $i \in \mathbb{Z}_{0,m}$ are all convex. Then (2.7) is referred to as a convex optimization problem (Boyd and Vandenberghe, 2004).

### 2.3 Lagrange Duality

In Lagrange duality the constraints in the standard problem (2.1), with non-empty domain $\mathcal{D}$, are taken into account by augmenting the objective function using weighted sums of the constraint functions (Boyd and Vandenberghe, 2004). The weights are called Lagrange multipliers and are here denoted $\lambda_i$ for the inequality constraints and $\nu_i$ for the equality constraints. The augmented objective function is denoted the Lagrangian, and is given by
\[
L(x, \lambda, \nu) \triangleq f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x). \tag{2.8}
\]

By minimizing the Lagrangian with respect to the variable $x$ over the domain $\mathcal{D}$, the Lagrange dual function
\[
g(\lambda, \nu) \triangleq \inf_{x \in \mathcal{D}} \left( f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right), \tag{2.9}
\]
is obtained. This is a concave function, and when $\lambda \geq 0$ it satisfies the relation
\[
g(\lambda, \nu) \leq p^*,
\]
and takes on the value $-\infty$ if the Lagrangian is unbounded in $x$ (Boyd and Vandenberghe, 2004).

**Remark 2.4.** The Lagrange dual function $g(\lambda, \nu)$ in (2.9) is concave even when the problem (2.1) is not convex (Boyd and Vandenberghe, 2004).

The Lagrange dual function defines a lower bound on the optimal value $p^*$ of (2.1) if $\lambda \geq 0$. Hence, maximizing (2.9) gives the best possible lower bound on $p^*$ that can be obtained from the Lagrange dual function. This lower bound can be found by solving the optimization problem
\[
\begin{align*}
\text{maximize} & \quad g(\lambda, \nu) \\
\text{subject to} & \quad \lambda \geq 0,
\end{align*}
\]
which is called the Lagrange dual problem, or simply the dual problem, of (2.1) and $\lambda$ and $\nu$ are called the dual variables. Since the objective function $g(\lambda, \nu)$ to be maximized is concave and the constraints in (2.11) are convex, the dual problem (2.11) is a convex optimization problem (Boyd and Vandenberghe, 2004). From Remark 2.4 it follows that it is a convex problem even when the primal problem (2.1) is not.

### 2.3.1 Weak and strong duality

Let the optimal value of the dual problem (2.11) be denoted $d^*$. Then, from the property (2.10) of the Lagrange dual function, it follows that
\[
d^* \leq p^*.
\]
The property (2.12) is referred to as weak duality and holds even if the primal problem (2.1) is not convex, or $d^*$ or $p^*$ are infinite. The duality gap is defined as the difference $p^* - d^*$ and is always non-negative.

Under certain conditions, the inequality (2.12) holds with equality, i.e., $d^* = p^*$, and the duality gap is zero. This property is referred to as strong duality, and it does not hold in general for an optimization problem. Conditions that certify that strong duality holds are called constraint qualifications. One such constraint qualification is given by Slater’s theorem, which states that strong duality holds for a convex optimization problem (2.7) given that Slater’s condition holds (Boyd and Vandenberghe, 2004). Slater’s condition states that there must exist a strictly feasible point for strong duality to hold. A weaker and refined version of Slater’s condition can be used to certify strong duality when some (or all) of the inequality constraint functions are affine. These conditions are repeated from Boyd and Vandenberghe (2004) in definitions 2.5 and 2.6.
2.4 Optimality Conditions

**Definition 2.5 (Slater’s condition).** There exists an \( x \in \text{relint } D \) such that
\[
f_i(x) < 0, \ i \in \mathbb{Z}_{1,m}, \quad a_i^T x = b_i, \ i \in \mathbb{Z}_{1,p}.
\] (2.13)

**Definition 2.6 (Slater’s refined condition).** There exists an \( x \in \text{relint } D \) such that
\[
f_i(x) \leq 0, \ i \in \mathbb{Z}_{1,k}, \quad f_i(x) < 0, \ i \in \mathbb{Z}_{k+1,m}, \quad a_i^T x = b_i, \ i \in \mathbb{Z}_{1,p},
\] (2.14)
where \( f_i \) for \( i \in \mathbb{Z}_{1,k} \) are all affine functions.

Here, \( \text{relint } D \) is the relative interior of the domain \( D \) (Boyd and Vandenberghe, 2004). Strong duality implies that if there exists a solution \( x^* \) to (2.7) with optimal value \( p^* \), then there exists dual variables \( \lambda^* \geq 0 \) and \( \nu^* \) such that \( g(\lambda^*, \nu^*) = d^* = p^* \) (Boyd and Vandenberghe, 2004).

### 2.4 Optimality Conditions

Consider an optimization problem given by (2.1) where the functions \( f_i \) for \( i \in \mathbb{Z}_{0,m} \) and \( h_i \) for \( i \in \mathbb{Z}_{1,p} \) are all differentiable and where strong duality holds. Then, the optimal primal and dual solutions to this optimization problem satisfy the KKT conditions (Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006).

**Definition 2.7 (Karush-Kuhn-Tucker conditions).** The conditions
\[
\nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) = 0, 
\] (2.15a)
\[
f_i(x) \leq 0, \ i \in \mathbb{Z}_{1,m}, \quad f_i(x) < 0, \ i \in \mathbb{Z}_{k+1,m},
\] (2.15b)
\[
h_i(x) = 0, \ i \in \mathbb{Z}_{1,p},
\] (2.15c)
\[
\lambda_i \geq 0, \ i \in \mathbb{Z}_{1,m},
\] (2.15d)
\[
\lambda_i f_i(x) = 0, \ i \in \mathbb{Z}_{1,m},
\] (2.15e)

are called the Karush-Kuhn-Tucker conditions.

The KKT conditions in Definition 2.7 are necessary for optimality of the solution to (2.1). If the problem is convex the conditions are also sufficient for optimality, which is summarized in Theorem 2.8.

**Theorem 2.8.** Consider a convex optimization problem in the form in (2.7) that satisfies Slater’s condition and has differentiable \( f_i \) for \( i \in \mathbb{Z}_{0,m} \) and \( h_i \) for \( i \in \mathbb{Z}_{1,p} \). Then the KKT conditions in Definition 2.7 are necessary and sufficient conditions for optimality, and any primal and dual pair \( \tilde{x} \) and \( (\tilde{\lambda}, \tilde{\nu}) \) that satisfies the KKT conditions (2.15) are primal and dual optimal.

**Proof:** See Section 5.5 in Boyd and Vandenberghe (2004).
2.5 Quadratic Programming

One important class of convex optimization problems are quadratic programs (QPs). Many problems can be directly formulated as QPs, but QP problems also arise as subproblems in methods for solving general constrained optimization problems (Nocedal and Wright, 2006). A QP problem is a convex optimization problem in the form (2.7) with affine constraint functions and a quadratic cost function, i.e.,

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T Q x + l^T x + c \\
\text{subject to} & \quad A_I x \preceq b_I \\
& \quad A_E x = b_E,
\end{align*}
\]

(2.16)

where \( x \in \mathbb{R}^n \), \( Q \in \mathbb{S}_+^n \), \( l \in \mathbb{R}^n \), \( c \in \mathbb{R} \), \( A_I \in \mathbb{R}^{m \times n} \), \( b_I \in \mathbb{R}^m \), \( A_E \in \mathbb{R}^{p \times n} \) and \( b_E \in \mathbb{R}^p \). Since the inequality constraints in (2.16) are affine in the optimization variable \( x \), it follows that Slater’s refined condition is satisfied, and hence Theorem 2.8 states that the KKT conditions in Definition 2.7 are necessary and sufficient for optimality. The KKT conditions (2.15) for a QP problem in the form in (2.16) reduce to

\[
\begin{align*}
Q x + l + A_I^T \lambda + A_E^T \nu &= 0, \quad (2.17a) \\
A_I x &\preceq b_I, \quad (2.17b) \\
A_E x &= b_E, \quad (2.17c) \\
\lambda &\succeq 0, \quad (2.17d) \\
\lambda_i (a_i^T x - b_i) &= 0, \quad i \in \mathbb{Z}_{1,m}, \quad (2.17e)
\end{align*}
\]

and hence any \( \hat{x} \) and \((\hat{\lambda}, \hat{\nu})\) that satisfy (2.17) are primal and dual global optimal solutions to (2.16). Here the notation \( a_i^T \) and \( b_i \) are used to denote the \( i \):th row or element of \( A_I \) and \( b_I \), respectively, \( i.e., \)

\[
\begin{bmatrix}
a_1^T \\
\vdots \\
a_m^T
\end{bmatrix} \triangleq A_I, \quad \begin{bmatrix}
b_1 \\
\vdots \\
b_m
\end{bmatrix} \triangleq b_I.
\]

(2.18)

The QP problem (2.16) can be solved using any method for convex optimization problems. As is mentioned in Chapter 1, the focus in this thesis is on numerical algorithms that can be used as subroutines in second-order optimization methods where the search directions are computed as Newton steps. Examples of such optimization methods are second-order active-set (AS) methods and interior-point (IP) methods, which will be outlined in the following sections.

2.6 Active-Set Methods for Quadratic Programming

Active-set methods is a class of methods that has been widely used since the 1970s (Nocedal and Wright, 2006). The basic idea in this type of methods is to find the set of equality constraints and inequality constraints in (2.16) that hold
2.6 Active-Set Methods for Quadratic Programming

with equality at the optimal solution. Here, this will be described following the presentation of the AS solver for QP problems presented in Nocedal and Wright (2006). To simplify notation, the set of all equality constraints and the set of all inequality constraints for a QP problem in the form (2.16) are introduced in Definition 2.9.

**Definition 2.9.** Let \( \mathcal{E} \) and \( \mathcal{I} \) be defined as the sets of indices of all the equality constraints and inequality constraints, respectively, in the problem (2.16).

The set of all indices of equality constraints and inequality constraints that hold with equality, or are active, is called the active set. The active set and its properties is an important concept in AS methods. The following definitions are inspired by Nocedal and Wright (2006).

**Definition 2.10 (Active set).** The active set at any feasible \( x \) in (2.16) is denoted \( \mathcal{A}(x) \) and consists of all indices of constraints in (2.16) that hold with equality at \( x \), i.e.,

\[
\mathcal{A}(x) \triangleq \mathcal{E} \cup \{ i \in \mathcal{I} \mid a_i^T x = b_i \}.
\]

(2.19)

With a slight abuse of notation, when a constraint is be said to be in the active set it should be interpreted as its corresponding index is in the active set.

**Definition 2.11 (Strongly and weakly active constraints).** A constraint \( i \in \mathcal{A}(x^*) \cap \mathcal{I} \) is weakly active if its dual variable \( \lambda_i^* = 0 \) for all \( \lambda^* \) satisfying (2.17). If \( \lambda_i^* > 0 \) for some \( \lambda^* \) satisfying (2.17), the constraint is called strongly active.

**Definition 2.12 (Strict Complementarity).** Given a local solution \( x^* \) and a vector \( \lambda^* \) that together satisfy (2.17). Then, the strict complementarity (SC) property holds if \( \lambda_i^* > 0 \) for each \( i \in \mathcal{A}(x^*) \cap \mathcal{I} \).

Note that from definitions 2.11 and 2.12 it follows that there are no weakly active constraints when \( \text{SC} \) holds. Next, the linear independence constraint qualification (LICQ) is defined. It is a property that is important for many optimization methods.

**Definition 2.13 (Linear independence constraint qualifications).** For the constraints with indices in the active set \( \mathcal{A}(x) \) the linear independence constraint qualification holds if the constraint gradients are linearly independent. When LICQ is violated it is referred to as primal degeneracy.

If the optimal active set \( \mathcal{A}(x^*) \) is known a priori, the optimal solution can be computed by solving the equality constrained QP problem

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} x^T Q x + l^T x + c \\
\text{subject to} \quad & a_i^T x = b_i, \ i \in \mathcal{A}(x^*).
\end{align*}
\]

(2.20)
However, the optimal active set is rarely known prior to the solution has been computed. The fundamental idea with an AS solver is to iteratively search for this optimal active set by adding and removing constraint indices from the so called working set. The working set at AS iteration $j$ is denoted $\mathcal{W}_j$ and consists of the indices of all equality constraints and the subset of the indices of the inequality constraints that are forced to hold with equality. Similarly let $\mathcal{W}^c_j$ be the complement of the working set $\mathcal{W}_j$ containing all the inequality constraints that are temporarily disregarded. It is required that LICQ holds for the constraints in the working set $\mathcal{W}_j$ (Nocedal and Wright, 2006).

Let $x_j$ denote the $j$:th iterate in the AS solver. The step to the next iterate $x_{j+1}$ is computed by solving an equality constrained QP where the constraints in the working set $\mathcal{W}_j$ are forced to hold with equality and the remaining constraints are temporarily disregarded, i.e.,

$$
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \hat{x}_{j+1}^T Q \hat{x}_{j+1} + l^T \hat{x}_{j+1} + c \\
\text{subject to} & \quad a_i^T \hat{x}_{j+1} = b_i, \ i \in \mathcal{W}_j.
\end{align*}
$$

(2.21)

When the solution $\hat{x}_{j+1}$ to (2.21) is computed, there are two different outcomes which require different actions to be taken. If $\hat{x}_{j+1} = x_j$, then this point minimizes the objective function in the current working set $\mathcal{W}_j$. Depending on the dual variables $\lambda^*_i$ for the inequality constraints in the working set, the iterate $\hat{x}_{j+1}$ is either an optimal point of the original problem, and at least one constraint has to be removed from the working set. The other outcome is when $\hat{x}_{j+1} \neq x_j$, and then a step from $x_j$ towards $\hat{x}_{j+1}$ along $p_j \equiv \hat{x}_{j+1} - x_j$ is taken. In this case, it is possible that a constraint is blocking the step form $x_j$ along $p_j$ and must be added to the working set. These different outcomes will be described below.

### 2.6.1 Adding constraints to the working set

For an iterate where $\hat{x}_{j+1} \neq x_j$, the new iterate is computed by taking a step along $p_j = \hat{x}_{j+1} - x_j$ such that

$$
\begin{align*}
x_{j+1} = x_j + \alpha_j p_j, \quad \alpha_j \in [0, 1].
\end{align*}
$$

(2.22)

If the point $x_j + p_j$ is feasible with respect to all constraints in (2.16), then $\alpha_j = 1$ is used to update the new iterate in (2.22) as $x_{j+1} = x_j + p_j = \hat{x}_{j+1}$. If $x_j + p_j$ is not feasible, then one or more constraints are violated when moving from $x_j$ to $x_j + p_j$ along $p_j$. These violated constraints are called blocking constraints, and the step-length parameter $\alpha_j$ is chosen such that all blocking constraints, and hence all constraints, remain satisfied.

All constraints $i \in \mathcal{W}_j$ are satisfied since they are included as equality constraints in (2.21). Hence, the blocking constraints must be among the constraints $i \in \mathcal{W}^c_j$ (i.e. $i \notin \mathcal{W}_j$) and $\alpha_j$ can be computed by determining which inequality constraint $i \in \mathcal{W}^c_j$ that is closest to $x_j$ along the direction given by $p_j$. Inserting the
point (2.22) into each separate constraint \(i \in W^c_j\) gives that the relation
\[
a_i^T (x_j + \alpha_j p_j) = a_i^T x_j + \alpha_j a_i^T p_j \leq b_i,
\]  
(2.23)
must hold for all \(i \in W^c_j\). Since \(x_j\) is a feasible point it follows that \(a_i^T x_j \leq b_i\) and, hence (2.23) is satisfied for all \(\alpha_j \geq 0\) whenever \(a_i^T p_j \leq 0\). This can be seen by writing (2.23) as
\[
a_i^T x_j \leq b_i \leq b_i - \alpha_j a_i^T p_j,
\]  
(2.24)
since \(a_i^T p_j \leq 0\) and \(\alpha_j \geq 0\). Hence constraint \(i\) cannot be a blocking constraint. If however \(a_i^T p_j > 0\), then (2.23) is satisfied only for certain choices of the step-length parameter \(\alpha_j\). The upper limit can be calculated as
\[
a_i^T (x_j + \alpha_j p_j) = a_i^T x_j + \alpha_j a_i^T p_j \leq b_i \iff \alpha_j \leq \frac{b_i - a_i^T x_j}{a_i^T p_j}.
\]  
(2.25)
To maximize the decrease of the objective function, \(\alpha_j\) is chosen as large as possible in the set \([0, 1]\) while retaining primal feasibility. Hence, the step-length parameter is chosen as
\[
\alpha_j \triangleq \min \left( 1, \min_{i \in W^c_j, a_i^T p_j > 0} \frac{b_i - a_i^T x_j}{a_i^T p_j} \right).
\]  
(2.26)
When there exist one or more blocking constraints, a new working set \(W_{j+1}\) is constructed by adding the blocking constraint that corresponds to the minimal step-length parameter in (2.26) to \(W_j\). Let the index of this constraint be \(i\). Then the new working set \(W_{j+1}\) is obtained as
\[
W_{j+1} = W_j \cup \{i\}.
\]  
(2.27)
To find the new step direction after modifying the working set a new equality constrained QP in the form (2.21) is solved for \(W_{j+1}\), and the procedure described above is repeated until a point \(\hat{x}_{j+1} = x_j\) that minimizes the current working set \(W_j\) for some iteration \(j\) is found. By defining \(\lambda_i \triangleq 0\) for all \(i \in W_j\) and noting that the point \(\hat{x}_{j+1}\) is a feasible point that minimizes the objective function in (2.21) it can be concluded that (2.17a), (2.17b), (2.17c) and (2.17e) in the KKT conditions are satisfied when \(\hat{x}_{j+1} = x_j\).

### 2.6.2 Removing constraints from the working set

Based on the discussion above, (2.17d) is the only KKT condition that is not guaranteed to be satisfied for the case when \(\hat{x}_{j+1} = x_j\). However, if also
\[
\lambda_i \geq 0, \ i \in W_j \cap I,
\]  
(2.28)
then also (2.17d) is satisfied, and the primal and dual point \(\hat{x}_{j+1}\) and \((\lambda, \nu)\) satisfies the KKT optimality conditions (2.17) for the QP problem (2.16). Hence, \(x^* = \hat{x}_{j+1}\) is an optimal solution to (2.16). If however \(\lambda_r < 0\) for some \(r \in W_j \cap I\), then the KKT condition (2.17d) is violated and the point \(\hat{x}_{j+1}\) cannot be optimal.

When \(\lambda_r < 0\) for some \(r \in W_j \cap I\), it may be possible to decrease the value of
the objective function by removing one of these constraints from the working set. Any of the constraints corresponding to a negative dual variable can be removed, but a common choice is to remove the most negative one. However, this does not guarantee that the decrease in the objective function is larger than if another constraint is removed (Nocedal and Wright, 2006). Let constraint \( r \) be removed from the working set. Then the working set \( W_j \) is modified as

\[
W_{j+1} = W_j \setminus \{r\}, \tag{2.29}
\]

and a new equality constrained QP in the form given by (2.21) with equality constraints given by the indices in \( W_{j+1} \) has to be solved.

### 2.6.3 Basic primal active-set method

The procedure which has been presented in this section describes the basic components of a traditional AS method, and is summarized in Algorithm 1. Inspiration for this algorithm is taken from Algorithm 16.3 in Nocedal and Wright (2006). In this algorithm, only one constraint is added or removed at each AS iteration. However, there exist methods that changes the working set by adding or removing several constraints at each iteration, see for instance Axehill (2008).

---

**Algorithm 1 Basic active-set solver for convex QPs (Nocedal and Wright, 2006)**

1. Compute a feasible starting point \( x_0 \)
2. Set \( W_0 \) to be a subset of the active constraints at \( x_0 \)
3. for \( j = 0,1,2,\ldots \) do
   4. Compute \( \hat{x}_{j+1} \) by solving (2.21)
   5. if \( \hat{x}_{j+1} = x_j \) then
      6. Compute the Lagrange multipliers \((\hat{\lambda}, \hat{\nu})\) that satisfy (2.17a)
      7. if \( \hat{\lambda}_i \geq 0 \) for all \( i \in W_j \cap I \) then
         8. \( x^* := x_j \)
         9. STOP
      10. else
         11. \( r := \arg\min_{i \in W_j \cap I} \hat{\lambda}_i \)
         12. Update iterate and working set: \( x_{j+1} := x_j, \ W_{j+1} := W_j \setminus \{r\} \)
   13. end if
   14. else
      15. Compute \( p_j := \hat{x}_{j+1} - x_j \) and step-length parameter \( \alpha_j \) from (2.26)
      16. Update iterate: \( x_{j+1} := x_j + \alpha_j p_j \)
      17. if \( \alpha_j < 1 \) (blocking constraints) then
         18. Set \( i \) to the blocking constraint giving \( \alpha_j \)
         19. Update working set: \( W_{j+1} := W_j \cup \{i\} \)
      20. else
         21. \( W_{j+1} := W_j \)
      22. end if
   23. end if
4. end for
At Line 1 in Algorithm 1, a feasible starting point \( x_0 \) for the solver needs to be found. Computing this feasible starting point for a general QP can be done using a so called phase I algorithm, and can take up to one-third to one-half of the total solution time (Goldfarb and Idnani, 1983). At Line 2 in Algorithm 1, the initial working set is a subset of the active constraints at \( x_0 \), and the iteration sequence to the solution is different for different choices of \( W_0 \) (Nocedal and Wright, 2006).

### 2.7 Interior-Point Methods

In this section a brief introduction of interior-point (IP) methods for convex optimization problems is given. Consider a convex optimization problem in the form (2.7) where the equality constraints are presented in matrix form, i.e.,

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq 0, \quad i \in \mathbb{Z}_{1,m} \\
& \quad Ax = b,
\end{align*}
\]

(2.30)

with twice continuously differentiable functions \( f_i \) for \( i \in \mathbb{Z}_{0,m} \) and \( A \in \mathbb{R}^{p \times n} \) with rank \( A = p < n \). Assume that the problem is solvable and strictly feasible. The problem (2.30) can be solved using an IP method, where a sequence of equality constrained convex optimization problems in the form

\[
\begin{align*}
\text{minimize} & \quad tf_0(x) + \phi(x) \\
\text{subject to} & \quad Ax = b,
\end{align*}
\]

(2.31)

are solved for increasing values of \( t > 0 \) (Boyd and Vandenberghe, 2004). Here, \( \phi(x) \) is the logarithmic barrier function given by

\[
\phi(x) \triangleq -\sum_{i=1}^{m} \log \left( -f_i(x) \right).
\]

(2.32)

Let \( x^*(t) \) be the optimal solution to (2.31) for a fixed \( t > 0 \). Then \( x^*(t) \) is called a central point and the set of points \( x^*(t) \) for \( t > 0 \) is called the central path associated with the problem (2.30). Every central point is strictly feasible and for each \( t \), the corresponding central point \( x^*(t) \) and the associated dual feasible pair \((\lambda^*(t), \nu^*(t))\) satisfy the perturbed KKT conditions

\[
\nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + A^T \nu = 0,
\]

(2.33a)

\[
f_i(x) \leq 0, \quad i \in \mathbb{Z}_{1,m},
\]

(2.33b)

\[
Ax = b, \quad i \in \mathbb{Z}_{1,p},
\]

(2.33c)

\[
\lambda_i \geq 0, \quad i \in \mathbb{Z}_{1,m},
\]

(2.33d)

\[
-\lambda_i f_i(x) = \frac{1}{t}, \quad i \in \mathbb{Z}_{1,m},
\]

(2.33e)
(Boyd and Vandenberghe, 2004). Furthermore, the duality gap associated with $x^*(t)$ and $(\lambda^*(t), \nu^*(t))$ is given by

$$f_0(x^*(t)) - g(\lambda^*(t), \nu^*(t)) = m/t,$$  \hfill (2.34)

which gives the relation

$$f_0(x^*(t)) - m/t = g(\lambda^*(t), \nu^*(t)) \leq p^* \iff f_0(x^*(t)) - p^* \leq m/t,$$  \hfill (2.35)

and hence $x^*(t)$ is no more than $m/t$-suboptimal (Boyd and Vandenberghe, 2004). Note that the perturbed KKT conditions (2.33) approach the KKT conditions (2.15) in Definition 2.7 as $t \to \infty$. The problem (2.31) can be solved using any method for linearly constrained convex optimization (Boyd and Vandenberghe, 2004). However, here Newton methods are used.

### 2.7.1 Basic primal interior-point method

In Algorithm 2, a simple basic primal IP method called the barrier method is introduced. This algorithm is based on Algorithm 11.1 in Boyd and Vandenberghe (2004), and it computes a sequence of central points for increasing $t$ until an $\epsilon$-suboptimal solution, i.e., a solution for $t > m/\epsilon$, is found. The computation of these central points is where most of the computational effort in IP methods is spent (Boyd and Vandenberghe, 2004).

#### Algorithm 2 The Barrier Method (Boyd and Vandenberghe, 2004)

1: Initialize a strictly feasible $x$, $t := t^{(0)} > 0$, $\mu > 1$, tolerance $\epsilon > 0$
2: loop
3: Centering step: Compute $x^*(t)$ by solving (2.31), starting at $x$
4: Update: $x := x^*(t)$
5: if Stopping criterion: $m/t < \epsilon$ then
6: STOP
7: else
8: Increase $t$: $t := \mu t$
9: end if
10: end loop

The centering step at Line 3 in Algorithm 2 consists of solving a problem in the form (2.31), which can be done by computing the solution to the perturbed KKT system (2.33) (Boyd and Vandenberghe, 2004). For a primal IP method, this perturbed KKT system is solved by eliminating the dual variable $\lambda_i$ from (2.33). The elimination is done using (2.33e) to compute $\lambda_i$ as

$$\lambda_i = \frac{1}{-tf_i(x)},$$  \hfill (2.36)
which inserted in (2.33a) gives the modified KKT system

\[ \nabla f_0(x) + \sum_{i=1}^{m} \frac{1}{-t f_i(x)} \nabla f_i(x) + A^T \nu = 0, \tag{2.37a} \]

\[ Ax = b, \tag{2.37b} \]

which are also the KKT conditions for (2.31). These can be solved using Newton’s method starting at a feasible \( x \). See for example Boyd and Vandenberghe (2004) or Nocedal and Wright (2006) for a description of Newton’s method. In Boyd and Vandenberghe (2004) it is shown that the Newton steps that are computed when solving the centering problem (2.31) can be interpreted as Newton steps for solving the modified KKT system (2.37) by scaling \( \nu \). The Newton step \( \Delta x_{nt} \) and the corresponding dual variable \( \nu_{nt} \) for the centering problem (2.31) are given by the solution to the linear system of equations

\[
\begin{bmatrix}
  t \nabla^2 f_0(x) + \nabla^2 \phi(x) \\
  A \\
  0
\end{bmatrix}
\begin{bmatrix}
  \Delta x_{nt} \\
  \nu_{nt}
\end{bmatrix} =
\begin{bmatrix}
  t \nabla f_0(x) + \nabla \phi(x) \\
  0
\end{bmatrix},
\tag{2.38}
\]

where \( \nu = (1/t)\nu_{nt} \). Hence, it follows that solving the centering step (2.31) requires the solution to a sequence of linear systems of equations in the form (2.38). Solving these systems of equations fast is crucial to the performance of an IP method. The KKT matrix in (2.38) must be non-singular for \( \Delta x_{nt} \) to be a proper Newton step (Boyd and Vandenberghe, 2004). In this reference, some conditions that certify non-singularity of the KKT matrix are presented.

The system of equations (2.38) that defines a Newton step to the centering problem (2.31) can be interpreted as the optimality conditions for the equality constrained QP problem that is obtained by inserting the second-order Taylor expansions of \( f_0(x) \) and \( \phi(x) \) around \( x \) into (2.31). The Taylor expansions of these functions are given by

\[ f_0(x + \Delta x) \approx f_0(x) + \nabla f_0(x)^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f_0(x) \Delta x, \tag{2.39a} \]

\[ \phi(x + \Delta x) \approx \phi(x) + \nabla \phi(x)^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \phi(x) \Delta x, \tag{2.39b} \]

and inserting these into the centering problem (2.31) results in

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \Delta x^T \left( t \nabla^2 f_0(x) + \nabla^2 \phi(x) \right) \Delta x + \left( t \nabla f_0(x)^T + \nabla \phi(x)^T \right) \Delta x + t f_0(x) + \phi(x) \\
\text{subject to} & \quad A \Delta x = 0.
\end{align*}
\tag{2.40}
\]

The equality constraints in (2.40) are obtained by inserting \( x + \Delta x \) into the equality constraints in the centering problem (2.31), i.e.,

\[ A (x + \Delta x) = Ax + A \Delta x = b \iff A \Delta x = 0. \tag{2.41} \]

It is clear that the KKT optimality conditions for (2.40) is given by the system of equations in (2.38). The conclusion is that the centering problem (2.31) can
be solved by computing the solution to a sequence of equality constrained QP problems in the form (2.40), which is equivalent to solving a sequence of systems of equations in the form in (2.38).

Here, a basic primal IP method is used to illustrate the basic steps and computations that are required in an IP method. However, similar linear algebra is also used in primal-dual IP methods (Rao et al., 1998; Boyd and Vandenberghe, 2004). Hence, algorithms for efficiently solving systems of equations in the form (2.38) are important also for more general types of IP methods than the barrier method.
The very early history of optimal control dates back to the geometric shape problems “the catenary” and the “brachistochrone” studied in the 17th century, and the following foundation of calculus of variations by Euler and Lagrange in the 18th century. It gained in popularity in aerospace applications during the 1960’s due to the development of more powerful computers (Sargent, 2000). In brief, the optimal control problem for dynamical systems consists of computing an optimal control input such that an objective function is minimized, while being subject to initial and possibly terminal constraints on the states, and inequality constraints on the states and control inputs (Speyer and Jacobson, 2010). The optimal control problem can be defined in continuous time or in discrete time, depending on the application. Discrete-time optimal control problems are also important since they arise from the discretized versions of continuous-time optimal control problems (Betts, 2001; Rao, 2009). For detailed descriptions of optimal control theory, see for example Polak (1973), Zhou et al. (1996), Bertsekas (2000), Betts (2001), or Speyer and Jacobson (2010).

As mentioned in Chapter 1, there exist many different methods to solve optimal control problems. In this thesis, the focus is spent on numerical algorithms for increasing the performance in second-order methods for discrete-time optimal control problems in the form of constrained finite-time optimal control (CFTOC) problems. This type of problems is an important component in MPC (Maciejowski, 2002), but can also be useful for example in motion planning problems for dynamical systems (LaValle, 2006; Ljungqvist et al., 2017; Bergman and Axehill, 2017) and MHE problems (Rao, 2000; Jørgensen, 2004; Haverbeke et al., 2009).
3.1 Constrained Finite-Time Optimal Control Problem

Here, CFTOC problems with quadratic objective functions and affine equality and inequality constraints are considered. This type of problem is not only important in itself, but CFTOC problems in this form also arise as important subproblems in optimization solvers such as SQP methods applied to CFTOC problems for nonlinear dynamical systems with general nonlinear constraints (Barclay et al., 1998; Betts, 2001; Jørgensen, 2004; Friesz, 2010). The CFTOC problem considered here is given by the QP problem

\[
\begin{align*}
\text{minimize} & \quad x, u \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t & u_t \end{bmatrix}^T \begin{bmatrix} \tilde{Q}_{x,t} & \tilde{Q}_{xu,t} \\ \tilde{Q}_{xu,t} & \tilde{Q}_{u,t} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \right) + \begin{bmatrix} l_x, t \\ l_{u,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \bar{c}_t + \tilde{c}_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = \tilde{A}_t x_t + \tilde{B}_t u_t + \tilde{a}_t, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad \tilde{H}_{x,t} x_t + \tilde{H}_{u,t} u_t + \tilde{h}_t \leq 0, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad \tilde{H}_{x,N} x_N + \tilde{h}_N \leq 0,
\end{align*}
\]

(3.1)

where \( x_t \in \mathbb{R}^{n_x} \) are the states, \( u_t \in \mathbb{R}^{n_u,t} \) are the control inputs, and \( \tilde{H}_{x,t} \in \mathbb{R}^{n_c,t \times n_x}, \tilde{H}_{u,t} \in \mathbb{R}^{n_c,t \times n_u,t} \) and \( \tilde{h}_t \in \mathbb{R}^{n_c,t} \) describe the \( n_c,t \) inequality constraints.

In this thesis, variables in sans-serif such as

\[
\begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix}, \quad \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix},
\]

(3.2)

denote vectors consisting of stacked vector components. The equality constraints in the CFTOC problem (3.1) are the discrete-time dynamics constraints of the controlled system.

Assumption 3.1.

\[
\begin{bmatrix} \tilde{Q}_{x,t} & \tilde{Q}_{xu,t} \\ \tilde{Q}_{xu,t} & \tilde{Q}_{u,t} \end{bmatrix} \in \mathbb{S}^{n_x+n_u,t}_{++}, \quad t \in \mathbb{Z}_{0,N-1}, \quad \tilde{Q}_{x,N} \in \mathbb{S}^{n_x}_+.
\]

(3.3)

Assumption 3.2.

\[
\tilde{Q}_{u,t} \in \mathbb{S}^{n_u,t}_{++}, \quad t \in \mathbb{Z}_{0,N-1}.
\]

(3.4)

If the objective function in the CFTOC problem (3.1) satisfies assumptions 3.1 and 3.2, it is a convex QP problem in the form (2.16). Furthermore, if the optimization problem is feasible it has a unique solution due to Assumption 3.2 (Maciejowski, 2002; Jerez et al., 2012). In the rest of this thesis, assumptions 3.1 and 3.2 hold unless stated otherwise.
Since CFTOC problems in the form (3.1) arise as subproblems in for example SQP and IP solvers applied to optimal control problems for nonlinear systems, the CFTOC problem (3.1) can be interpreted as either an optimal control problem for a system with affine dynamics, or as a subproblem in a solver for a more general optimal control problem. However, from a computational point of view the origin of the CFTOC problem (3.1) is not important since the same methods can be used to compute a solution to it.

The CFTOC problem (3.1) can be cast in a more compact form using similar notation as in (2.16), giving

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} l_x^T \\ l_u^T \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \bar{c} \\
\text{subject to} & \quad \bar{A}x + \bar{B}u + \bar{a} = 0 \\
& \quad \bar{H}_x x + \bar{H}_u u + \bar{h} \preceq 0,
\end{align*}
\]

where \( \bar{Q}_{x}, \bar{Q}_{ux}, \bar{Q}_{u}, \bar{l}_x, \bar{l}_u, \bar{c}, \bar{A}, \bar{B}, \bar{a}, \bar{H}_x, \bar{H}_u \) and \( \bar{h} \) are defined in Appendix 3.A. Note that \( \bar{A} \) is an invertible matrix by definition since it is a lower triangular matrix with all diagonal elements equal to \(-1\). Furthermore, note that the matrices in (3.5) are time varying and that the initial state \( \bar{x} \) is included in \( \bar{a} \). However, the explicit dependence on time for the matrices are omitted for brevity.

The CFTOC problem as stated in (3.1) or (3.5) is given in the non-condensed (or sparse) form, where the states \( x \) are included in the optimization problem. It is however possible to use the dynamics constraints in (3.5) to eliminate the state variables as

\[
x = -\bar{A}^{-1}\bar{B}u - \bar{A}^{-1}\bar{a},
\]

to obtain a strictly convex QP problem with only the control inputs \( u \) as optimization variables (Maciejowski, 2002). This will be referred to as the condensed formulation, and the condensed CFTOC problem is given by

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} u^T Q u + l^T u + c \\
\text{subject to} & \quad A_T u + b_T \preceq 0.
\end{align*}
\]

Here, the following definitions are used

\[
\begin{align*}
Q & \triangleq \bar{Q}_{u} - \bar{Q}_{xu} \bar{A}^{-1} \bar{B} - \bar{B}^T \bar{A}^{-T} \bar{Q}_{xu} + \bar{B}^T \bar{A}^{-T} \bar{Q}_{x} \bar{A}^{-1} \bar{B} \in S_{++}^m, \\
l & \triangleq \bar{l}_u + \bar{B}^T \bar{A}^{-T} \bar{Q}_{x} \bar{A}^{-1} \bar{a} - \bar{Q}_{xu} \bar{A}^{-1} \bar{a} - \bar{B}^T \bar{A}^{-T} \bar{l}_x \in \mathbb{R}^m, \\
c & \triangleq \bar{c} - \bar{l}_x^T \bar{A}^{-T} \bar{a} + \frac{1}{2} \bar{a}^T \bar{A}^{-T} \bar{Q}_{x} \bar{A}^{-1} \bar{a} \in \mathbb{R}, \\
A_T & \triangleq \bar{H}_u - \bar{H}_x \bar{A}^{-1} \bar{B} \in \mathbb{R}^p \times m, \quad b_T \triangleq \bar{h} - \bar{H}_x \bar{A}^{-1} \bar{a} \in \mathbb{R}^p,
\end{align*}
\]

with \( m \triangleq \sum_{t=0}^{N-1} n_{u,t} \) and \( p \triangleq \sum_{t=0}^{N} n_{c,t} \). Note that \( Q \) is positive definite, which follows from Assumption 3.2. See for example Maciejowski (2002) for the details.

Depending on the problem, the condensed or the sparse formulation can be more beneficial from a computational point of view. In the condensed formulation, the
Hessian matrix in the resulting QP problem (3.7) is dense and hence the solution can be computed in cubic computational complexity in the horizon $N$ (Boyd and Vandenberghe, 2004). The sparse formulation of the CFTOC problem can be solved using structure-exploiting algorithms with linear computational complexity growth in $N$, which can result in significant speed-ups (Rao et al., 1998). In recent years, methods for combining the benefits with both formulations have been presented. In Jerez et al. (2012) an approach to obtain a block-banded Hessian where the size of the bands are independent of $N$ is presented, which gives a linear computational complexity growth in $N$. In Axehill (2014, 2015) partial condensing is introduced as a way of combining the sparse and the condensed formulations to choose the best condensation strategy, which often is in between the traditional sparse and the condensed formulations. This has later been studied further in Kouzoupis et al. (2015), Frison (2015), Frison et al. (2016), Kouzoupis et al. (2016), and Quiroynen (2017). How to exploit the structure associated with the CFTOC problem even for the dense formulation has been investigated in Axehill and Morari (2012) and Frison and Jørgensen (2013). In this thesis, structure-exploiting algorithms for the sparse formulation are considered.

### 3.1.1 Model predictive control

Model predictive control is a control strategy where the control input that is applied to the controlled plant is computed by solving a CFTOC problem at each iteration in the control loop. It has become one of the most widely used advanced control strategies in industry today, and some important reasons for its success are that it can handle multivariable systems and constraints on control inputs and state variables in a structured way (Maciejowski, 2002). Depending on for example which application, controlled system and problem formulation that is used, the MPC problem and the corresponding CFTOC problem can be of various types. Some common types are linear MPC, nonlinear MPC and hybrid MPC. For linear MPC, the CFTOC problem is often in the form (3.1). However, CFTOC problems in the form (3.1) are also important when solving more general MPC problems such as nonlinear MPC.

The solution to the CFTOC problem can be computed on-line or off-line. One technique for solving the optimization problem off-line is explicit MPC, where the state-space is divided into polyhedral regions and an affine state feedback is computed off-line for each region, see for example Bemporad et al. (2002). Given the current state during run-time, the correct feedback can be applied by determining which region the current state belongs to. However, the complexity and memory requirements of computing and storing the regions and control laws might be intractable already for small-sized problems (Grancharova et al., 2003). An presentation of explicit MPC will be given in Chapter 4, where also multiparametric programming (mp-QP) problems are surveyed.

In MPC, a dynamic model of the controlled system is used to predict the behaviour over a prediction horizon, and the control input is optimized over this prediction horizon to minimize a performance criterion. The prediction is performed over a prediction horizon $N$ steps into the future, and longer prediction horizons gene-
rally generates more computationally demanding optimization problems but better control performance. Only the first control input is applied to the system that is controlled. The MPC controller operates using a receding horizon strategy, meaning that the control signal at time $t_0$ is computed using the predictions up to time $t_0 + N$, and the control input at time $t_1$ is computed using the predictions up to time $t_1 + N$, etc. This is illustrated in Figure 3.1, where the control input at time $t = 20$ is about to be computed. Here, a prediction horizon of $N = 20$ is used, and the predicted control input and output variable are marked in red. The blue dashed line symbolizes the predicted output. The real output might differ from the predicted one due to for example model error and/or disturbances. At each step in the MPC control loop the measured (or estimated) current state is used as initial value, and hence the MPC controller operates in closed loop (Maciejowski, 2002). The basic steps in an MPC control loop are outlined in Algorithm 3.

![MPC - receding horizon](image)

**Figure 3.1**: Here, an illustration of the receding horizon strategy in the MPC control loop is presented. The control input at time $t = 20$ is computed for an MPC problem with $N = 20$. The predicted output and control inputs are marked in red. The blue dashed line indicates that the predicted output and the real output might differ from each other.

**Algorithm 3**  Basic MPC operation

1: **repeat**
2: Measure (or estimate) current state at time $t$
3: Solve a CFTOC problem over the prediction horizon $t$ to $t + N$ to compute the control inputs
4: Apply the first computed control input at time $t$
5: **until** Break by user
3.1.2 Moving horizon estimation

Although MHE problems are not optimal control problems per se, they have a similar problem structure as the MPC problem (Rao, 2000; Jørgensen, 2004). In MHE, the state-estimate is obtained as the solution to a highly structured optimization problem solved on-line in a receding horizon fashion. In the same spirit as MPC adds the possibility for optimal control under constraints, MHE adds the possibility for optimal estimation under constraints. It has been shown in Rao (2000), Jørgensen (2004), and Haverbeke et al. (2009) that problem structure can be exploited also for this application. Furthermore, the optimization problem that is solved on-line in MHE can be shown to have a similar structure to the one in so-called smoothing (Rao, 2000; Kailath et al., 2000). In smoothing, measurements are available along the entire time window of estimation, which means that non-causal estimation can be performed. From a computational point of view, MHE can be interpreted as repeatedly solving smoothing problems in a receding horizon fashion where only the last state estimate is actually returned as an estimate. This is analogous to that only the first computed control input is applied in MPC. Depending on the type of system and problem formulation, the MHE problem can be of different types and MHE can be applied to linear, nonlinear or hybrid systems.

It is shown in Jørgensen (2004) that the optimization problem arising in MHE can be written as a CFTOC problem. Hence, methods for solving CFTOC problems in the form (3.1) can also be useful when solving the optimization problem arising in MHE and smoothing applications.

Here, the optimization problem arising in MHE problems for affine systems with affine inequality constraints is repeated from Jørgensen (2004), but using slightly different notation. This optimization problem is given by the inequality constrained QP problem

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} (\bar{x}_0 - \hat{x}_0)^T \tilde{P}_0^{-1} (\bar{x}_0 - \hat{x}_0) + \\
& \quad \frac{1}{2} \sum_{k=0}^{N_{\text{mhe}}} \begin{bmatrix} \bar{w}_k - \bar{\tilde{w}}_k \\ \bar{v}_k - \bar{\tilde{v}}_k \end{bmatrix}^T \begin{bmatrix} Q_{w,k} & \hat{Q}_{wv,k} \\ \hat{Q}_{wv,k}^T & Q_{v,k} \end{bmatrix}^{-1} \begin{bmatrix} \bar{w}_k - \bar{\tilde{w}}_k \\ \bar{v}_k - \bar{\tilde{v}}_k \end{bmatrix} \\
\text{subject to} & \quad \bar{x}_{k+1} = \tilde{A}_k \bar{x}_k + \tilde{B}_k \bar{\tilde{w}}_k + \bar{\tilde{a}}_k, \; k \in \mathbb{Z}_{0,N_{\text{mhe}}} \\
& \quad \bar{y}_k = \tilde{C}_k \bar{x}_k + \bar{\tilde{v}}_k + \bar{\tilde{d}}_k, \; k \in \mathbb{Z}_{0,N_{\text{mhe}}} \\
& \quad \bar{H}_{x,k} \bar{x}_k + \bar{H}_{w,k} \bar{\tilde{w}}_k + \bar{\tilde{h}}_k \succeq 0, \; k \in \mathbb{Z}_{0,N_{\text{mhe}}}.
\end{align*}$$

(3.9)

Here $N_{\text{mhe}}$ is the estimation horizon, $\bar{x}_k \in \mathbb{R}^{n_x}$ is the state, $\bar{\tilde{w}}_k \in \mathbb{R}^{n_w}$ is the process noise, $\bar{\tilde{v}}_k \in \mathbb{R}^{n_y}$ is the measurement noise and $\bar{y}_k \in \mathbb{R}^{n_y}$ is the measured output (Jørgensen, 2004). The initial state and the initial covariance matrix are $\hat{x}_0$ and $\tilde{P}_0$, respectively, and $\bar{\tilde{w}}_k$ and $\bar{\tilde{v}}_k$ are the nominal values for $\bar{w}_k$ and $\bar{v}_k$, respectively. A stochastic interpretation of this problem is found in for example Robertson et al. (1996) and Kailath et al. (2000). However, here the problem (3.9) is considered to be a deterministic optimization problem, which is sufficient from a computational point of view. It is shown in Jørgensen (2004) how (3.9) can be
written as a CFTOC problem by eliminating $\tilde{v}_k$ using the measurement equation $\tilde{y}_k = \tilde{C}_k \tilde{x}_k + \tilde{v}_k + \tilde{d}_k$ and by introducing new variables. To re-formulate the optimization problem (3.9), introduce the notation $N \triangleq N^{\text{mhe}} + 2$, and $x_t$ for $t \in \mathbb{Z}_{0,N}$ and $w_t$ for $t \in \mathbb{Z}_{0,N-1}$ as

$$x_0 \triangleq \tilde{x}_0, \quad x_t \triangleq \tilde{x}_{t-1}, \quad t \in \mathbb{Z}_{1,N}, \quad w_0 \triangleq \tilde{x}_0 - \tilde{x}_0, \quad w_t \triangleq \tilde{w}_{t-1}, \quad t \in \mathbb{Z}_{1,N-1}. \quad (3.10)$$

Furthermore, introduce $\tilde{Q}_{x,t}$, $\tilde{Q}_{xw,t}$, $\tilde{l}_{x,t}$, $\tilde{l}_{xw,t}$, $\tilde{c}_t$, $\tilde{A}_t$, $\tilde{B}_t$, $\tilde{h}_t$, $\tilde{H}_{x,t}$, $\tilde{H}_{w,t}$ and $\tilde{h}_N$ as in Appendix 3.B. Then, the MHE problem (3.9) can be equivalently written

$$\text{minimize}_{x,w} \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t \end{bmatrix}^T \begin{bmatrix} \tilde{Q}_{x,t} & \tilde{Q}_{xw,t} \\ \tilde{Q}_{xw,t}^T & \tilde{Q}_{w,t} \end{bmatrix} \begin{bmatrix} x_t \\ w_t \end{bmatrix} + \begin{bmatrix} \tilde{l}_{x,t} \\ \tilde{l}_{w,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ w_t \end{bmatrix} + \tilde{c}_t \right) + \frac{1}{2} \begin{bmatrix} x_N \end{bmatrix}^T \tilde{Q}_{x,N} x_N + \tilde{l}_{x,N}^T x_N + \tilde{c}_N$$

subject to $\quad x_0 = \tilde{x}_0$

$$\quad x_{t+1} = \tilde{A}_t x_t + \tilde{B}_t w_t + \tilde{a}_t, \quad t \in \mathbb{Z}_{0,N-1}$$

$$\quad H_{x,t} x_t + \tilde{H}_{w,t} w_t + \tilde{h}_t \preceq 0, \quad t \in \mathbb{Z}_{0,N-1}$$

$$\quad \tilde{H}_{x,N} x_N + \tilde{h}_N \preceq 0.$$  \hspace{1cm} (3.11)

The optimization problem in (3.11) is in the same form as the CFTOC problem in (3.1), and hence the MHE problem (3.9) can be solved by instead computing the solution to a CFTOC problem in the form (3.1).

### 3.2 Solving Constrained Finite-Time Optimal Control Problems

When assumptions 3.1 and 3.2 hold, the CFTOC problem (3.1) is a convex QP problem with a unique solution (Maciejowski, 2002). Hence, any method for solving this type of problem can be applied to (3.1) to find the optimal solution. In many cases, the effort spent when solving the CFTOC problem using second-order methods boils down to solving Newton-system-like equations that correspond to UFTOC problems. See the related work in Section 1.1.2 for some examples where the structure of the CFTOC problem have been exploited in different ways to improve the performance of second-order methods for CFTOC problems.

In this section, IP and AS methods that compute the search directions using the Newton method are presented. The purpose with this section is to show that the search directions for such methods can be computed by solving UFTOC problems. The section should not be considered a survey on QP methods for solving CFTOC problems, but it should rather be seen as a motivation for emphasizing the significance of the contributions presented in chapters 5-7.
3.2.1 Interior-point methods

The CFTOC problem (3.1) can be solved using IP methods that exploit the special problem structure. Examples of how the structure can be exploited are found in for example Wright (1993), Rao et al. (1998), Hansson (2000), Axehill et al. (2007), and Domahidi et al. (2012). In Section 2.7 a basic IP method to solve a convex optimization problem was introduced, and here it will be shown that the main computational effort when using IP methods to solve the CFTOC problem (3.1) is spent when solving a sequence of UFTOC problems.

In Section 2.7 it is argued that, when using an IP method for solving a convex optimization problem, a sequence of centering problems in the form (2.31) are solved. Furthermore, each centering problem is solved by computing a sequence of Newton steps corresponding to equality constrained QPs in the form (2.40). For the CFTOC problem (3.1), each of these equality constrained QPs can be obtained by using second-order Taylor expansions of the non-linear functions in the centering problem (2.31) around a feasible point \((\bar{x}, \bar{u})\). By using the compact formulation of the CFTOC problem as in (3.5), the Hessian and gradient of the objective function \(f_0\) are given by

\[
s\nabla^2 f_0(x, u) = s \begin{bmatrix} \hat{Q}_x & \hat{Q}_{xu} \\ \hat{Q}_{ux} & \hat{Q}_u \end{bmatrix}, \quad s\nabla f_0(x, u) = s \begin{bmatrix} \hat{Q}_x \\ \hat{Q}_{xu} \\ \hat{Q}_{ux} \\ \hat{Q}_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + s \begin{bmatrix} \hat{l}_x \\ \hat{l}_u \end{bmatrix},
\]

where “s” is used here instead of “t” in (2.31) and (2.38) to avoid confusion with the time index \(t\) in (3.1). To compute the gradient and Hessian of the logarithmic barrier function in (2.32), \(\nabla \phi(\bar{x}, \bar{u})\) and \(\nabla^2 \phi(\bar{x}, \bar{u})\), it is noted from (2.32) and the definition of the inequality constraints in (3.1) that

\[
\phi(x, u) = \sum_{t=0}^{N-1} \phi_t(x_t, u_t) + \phi_N(x_N),
\]

\[
\phi_t(x, u) \triangleq - \sum_{i=1}^{n_{c,t}} \log \left( - \tilde{H}_{x,i,t} x - \tilde{H}_{u,i,t} u - \tilde{h}_{i,t} \right),
\]

\[
\phi_N(x) \triangleq - \sum_{i=1}^{n_{c,N}} \log \left( - \tilde{H}_{x,i,N} x - \tilde{h}_{i,N} \right),
\]

and hence \(\nabla \phi(x, u)\) and \(\nabla^2 \phi(x, u)\) will have the structure

\[
\nabla \phi(x, u) = \begin{bmatrix} \nabla_x \phi_0(x_0, u_0) \\ \vdots \\ \nabla_x \phi_N(x_N) \\ \nabla_u \phi_0(x_0, u_0) \\ \vdots \\ \nabla_u \phi_{N-1}(x_{N-1}, u_{N-1}) \end{bmatrix}, \quad \nabla^2 \phi(x, u) = \begin{bmatrix} \nabla^2_x \phi(x, u) & \nabla^2_{xy} \phi(x, u) \\ \nabla^2_{ux} \phi(x, u) & \nabla^2_u \phi(x, u) \end{bmatrix}.
\]

(3.14)
From the definition of $\phi(x, u)$ it follows that the blocks in the Hessian $\nabla^2 \phi(x, u)$ in (3.14) are block diagonal, and that they are defined as

$$
\nabla^2 \phi(x, u) \triangleq \begin{bmatrix}
\nabla^2_x \phi_0(x_0, u_0) & \cdots & \nabla^2_x \phi_{N}(x_N) \\
\vdots & \ddots & \vdots \\
\nabla^2_u \phi_0(x_0, u_0) & \cdots & \nabla^2_u \phi_{N-1}(x_{N-1}, u_{N-1}) \\
\vdots & \ddots & \vdots \\
\nabla^2_{xu} \phi_0(x_0, u_0) & \cdots & \nabla^2_{xu} \phi_{N-1}(x_{N-1}, u_{N-1}) \\
0 & \cdots & 0
\end{bmatrix},
$$

(3.15a)

$$
\nabla^2 u \phi(x, u) \triangleq \begin{bmatrix}
\nabla^2 x_0 \phi_0(x_0, u_0) & \cdots & \nabla^2 x_0 \phi_{N}(x_N) \\
\vdots & \ddots & \vdots \\
\nabla^2 x_{u0} \phi_0(x_0, u_0) & \cdots & \nabla^2 x_{u0} \phi_{N-1}(x_{N-1}, u_{N-1}) \\
0 & \cdots & 0
\end{bmatrix},
$$

(3.15b)

$$
\nabla^2_{xu} \phi(x, u) \triangleq \begin{bmatrix}
\nabla^2 x_0 \phi_0(x_0, u_0) & \cdots & \nabla^2 x_0 \phi_{N-1}(x_{N-1}, u_{N-1}) \\
0 & \cdots & 0
\end{bmatrix},
$$

(3.15c)

$$
\nabla^2 u \phi(x, u) \triangleq (\nabla^2_{xu} \phi(x, u))^T.
$$

(3.15d)

The notation $\nabla^2 \phi_t(\bar{x}_t, \bar{u}_t)$ is used to denote the Hessian of $\phi_t(x, u)$ with respect to $x$ evaluated at the point $x = \bar{x}_t$ and $u = \bar{u}_t$, and analogously for $\nabla^2 \phi_t(\bar{x}_t, \bar{u}_t)$ and $\nabla^2_{xu} \phi_t(\bar{x}_t, \bar{u}_t)$. The gradient and Hessian $\nabla \phi(\bar{x}, \bar{u})$ and $\nabla^2 \phi(\bar{x}, \bar{u})$ are computed by inserting $\bar{x}$ and $\bar{u}$ into (3.14).

The equality constraints in the problem (2.40) can be formed by inserting the point $\bar{x} + \Delta x$ and $\bar{u} + \Delta u$ into the dynamics equations in (3.5), i.e.,

$$
\begin{align*}
\bar{A}(\bar{x} + \Delta x) + \bar{B}(\bar{u} + \Delta u) + \bar{a} &= 0 \iff \\
\bar{A}\Delta x + \bar{B}\Delta u &= 0.
\end{align*}
$$

(3.16)

The last step holds since $\bar{x}$ and $\bar{u}$ are primal feasible and hence satisfy

$$
\bar{A}\Delta x + \bar{B}\Delta u = 0.
$$

(3.17)

By using the expressions for the gradients and Hessians in (3.12) and (3.14), the objective function of the approximated optimization problem (2.40) can be written

$$
\begin{align*}
\frac{1}{2} \begin{bmatrix}
\Delta x \\
\Delta u
\end{bmatrix}^T \begin{pmatrix}
\bar{Q}_x & \bar{Q}_{xu} \\
\bar{Q}_{xu} & \bar{Q}_u
\end{pmatrix} + \begin{pmatrix}
\nabla^2_x \phi(\bar{x}, \bar{u}) & \nabla^2_{xu} \phi(\bar{x}, \bar{u}) \\
\nabla^2_{xu} \phi(\bar{x}, \bar{u}) & \nabla^2_u \phi(\bar{x}, \bar{u})
\end{pmatrix} \begin{pmatrix}
\Delta x \\
\Delta u
\end{pmatrix} + \\
\begin{pmatrix}
\nabla x \phi(\bar{x}, \bar{u}) \\
\nabla u \phi(\bar{x}, \bar{u})
\end{pmatrix}^T \begin{pmatrix}
\Delta x \\
\Delta u
\end{pmatrix} + sf_0(\bar{x}, \bar{u}) + \phi(\bar{x}, \bar{u}).
\end{align*}
$$

(3.18)

If the structure in the matrices in (3.18) is considered, it is clear that the quadratic part of the objective function is sparse with the same sparsity pattern as for the quadratic part in the CFTOC problem (3.5). Also the linear term has similar
structure as in (3.5). Now define
\[
Q_{x,t} \triangleq s\bar{Q}_{x,t} + \nabla^2_x \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19a)
\[
Q_{x,N} \triangleq s\bar{Q}_{x,N} + \nabla^2_x \phi_N (\bar{x}_N),
\] (3.19b)
\[
Q_{u,t} \triangleq s\bar{Q}_{u,t} + \nabla^2_u \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19c)
\[
Q_{xu,t} \triangleq s\bar{Q}_{xu,t} + \nabla^2_{xu} \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19d)
\[
l_{x,t} \triangleq s\tilde{Q}_{x,t} + s\bar{Q}_{xu,t} \bar{u}_t + s\tilde{l}_{x,t} + \nabla_x \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19e)
\[
l_{x,N} \triangleq s\tilde{Q}_{x,N} \bar{x}_N + s\tilde{l}_{x,N} + \nabla_x \phi_N (\bar{x}_N),
\] (3.19f)
\[
l_{u,t} \triangleq s\tilde{Q}_{u,t} \bar{u}_t + s\bar{Q}_{xu,t} \bar{x}_t + s\tilde{l}_{u,t} + \nabla_u \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19g)
\[
c_t \triangleq s \left[ \begin{array}{c} \bar{x}_t \\ \bar{u}_t \end{array} \right]^T \left[ \begin{array}{cc} \tilde{Q}_{x,t} & \tilde{Q}_{xu,t} \\ \tilde{Q}_{xu,t}^T & \tilde{Q}_{u,t} \end{array} \right] \left[ \begin{array}{c} \bar{x}_t \\ \bar{u}_t \end{array} \right] + s \left[ \begin{array}{c} \tilde{l}_{x,t} \\ \tilde{l}_{u,t} \end{array} \right]^T \left[ \begin{array}{c} \bar{x}_t \\ \bar{u}_t \end{array} \right] + s\tilde{c}_t + \phi_t (\bar{x}_t, \bar{u}_t), \ t \in \mathbb{Z}_{0,N-1},
\] (3.19h)
\[
c_N \triangleq s \left( \bar{x}_N^T \tilde{Q}_{x,N} \bar{x}_N + \bar{l}_{x,N}^T \bar{x}_N + \tilde{c}_N \right) + \phi_N (\bar{x}_N),
\] (3.19i)
and re-write the dynamics equations (3.16) as the equivalent equations
\[
\Delta x_0 = 0,
\] (3.20a)
\[
\Delta x_{t+1} = \tilde{A}_t \Delta x_t + \tilde{B}_t \Delta u_t, \ t \in \mathbb{Z}_{0,N-1}.
\] (3.20b)

Then, it can be concluded that when solving the CFTOC problem (3.1), the centering step (2.31) can be solved by computing a sequence of Newton steps corresponding to the solutions to UFTOC problems in the form

\[
\text{minimize}_{x,u} \sum_{t=0}^{N-1} \frac{1}{2} \left[ \begin{array}{c} \Delta x_t \\ \Delta u_t \end{array} \right]^T \left[ \begin{array}{cc} Q_{x,t} & Q_{xu,t} \\ Q_{xu,t}^T & Q_{u,t} \end{array} \right] \left[ \begin{array}{c} \Delta x_t \\ \Delta u_t \end{array} \right] + \left[ \begin{array}{c} l_{x,t} \\ l_{u,t} \end{array} \right]^T \left[ \begin{array}{c} \Delta x_t \\ \Delta u_t \end{array} \right] + c_t
\]
\[
+ \frac{1}{2} \Delta x_N^T Q_{x,N} \Delta x_N + l_{x,N}^T \Delta x_N + c_N
\]
subject to
\[
\Delta x_0 = 0 \\
\Delta x_{t+1} = \tilde{A}_t \Delta x_t + \tilde{B}_t \Delta u_t, \ t \in \mathbb{Z}_{0,N-1}.
\] (3.21)

In this section the centering step and the corresponding Newton steps for a basic barrier method was presented. However, systems of equations of the same structure may also arise when using primal-dual IP methods (Rao et al., 1998; Boyd and Vandenberghe, 2004; Axehill et al., 2007). Hence, the main computational effort in many IP solvers that are applied to solve CFTOC problems is spent on solving sequences of Newton steps in the form of UFTOC problems as in (3.21).
3.2 Solving Constrained Finite-Time Optimal Control Problems

3.2.2 Active-set methods

In this section it will be shown how the search directions that are required by an AS solver, like for example the one in Algorithm 1, can be computed when it is applied to solve a CFTOC problem. Here, CFTOC problems with upper and lower bounds on the control inputs will be considered. Hence, the CFTOC problem is given in form

\[
\begin{align*}
\text{minimize} \quad & 
\frac{1}{2} x_N^T \tilde{Q} x_N + \tilde{r}_N T \tilde{r}_N + \tilde{c}_N \\
\text{subject to} \quad & \begin{cases} \end{cases}
\end{align*}
\]

(3.22)

Let assumptions 3.1 and 3.2 still hold for (3.22) to guarantee a unique solution.

Let the working set \( W_j \) and its complement \( W^c_j \) at AS iteration \( j \) be introduced as in Section 2.6. In the CFTOC problem in (3.22) only control input box constraints are used, and hence the working set describes the set of control inputs that are temporarily fixed to a constant value (either the maximum or minimum limit). Adding a constraint to the working set, i.e., forcing a constraint to hold with equality, thus corresponds to removing that control input as an optimization variable from the optimization problem (3.22). Similarly, by disregarding an inequality constraint, the corresponding control input becomes unconstrained and is introduced as an optimization variable in the optimization problem. This can be formalized by introducing \( w_t \) as the free part of the control inputs and \( v_t \) as the fixed part, i.e.,

\[
w_t \triangleq u_t(W^c_j), \quad v_t \triangleq u_t(W_j), \quad u_t = \Pi \begin{bmatrix} w_t \\ v_t \end{bmatrix},
\]

(3.23)

where \( \Pi \) is a permutation matrix satisfying \( \Pi^T \Pi = I \). Here the notation \( u_t(W_j) \) is used to denote the elements in \( u_t \) that are affected by constraints with indices in \( W_j \). Using this notation, \( \tilde{B}_t, \tilde{Q}_{x,u,t}, \tilde{Q}_{xu,t} \) and \( \tilde{l}_{u,t} \) can be partitioned into parts corresponding to \( w_t \) and \( v_t \), respectively, giving

\[
\begin{align*}
\tilde{B}_t &= \begin{bmatrix} \tilde{B}_{w,t} & \tilde{B}_{v,t} \end{bmatrix} \Pi^T, \\
\tilde{Q}_{x,u,t} &= \begin{bmatrix} \tilde{Q}_{xw,t} & \tilde{Q}_{xv,t} \end{bmatrix} \Pi^T, \\
\tilde{Q}_{xu,t} &= \Pi \begin{bmatrix} \tilde{Q}_{xw,t} & \tilde{Q}_{xv,t} \end{bmatrix} \Pi^T, \\
\tilde{l}_{u,t} &= \begin{bmatrix} \tilde{l}_{w,t} \\ \tilde{l}_{v,t} \end{bmatrix} \Pi^T.
\end{align*}
\]

(3.24a)

The variables \( v_t \) can be eliminated from the optimization problem by inserting the fixed values given by the corresponding inequality constraints in the working set \( W_j \). Inserting this partitioning of \( u_t \) and \( \tilde{B}_t \) into the dynamics constraints in (3.22) gives
\[ x_{t+1} = \tilde{A}_t x_t + \tilde{B}_t u_t + \tilde{a}_t = \hat{A}_t x_t + \hat{B}_{w,t} w_t + \hat{B}_{v,t} v_t + \hat{a}_t = \tilde{A}_t x_t + \tilde{B}_{w,t} w_t + \tilde{a}_{v,t}, \] (3.25)

where \( \tilde{a}_{v,t} \) is defined as
\[ \tilde{a}_{v,t} \triangleq \tilde{B}_{v,t} v_t + \tilde{a}_t. \] (3.26)

Similarly, by using the partitioning of the control input and the matrices in the objective function in the CFTOC problem (3.22) gives the expression
\[
\sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t \\ w_t \end{bmatrix}^T \begin{bmatrix} \tilde{Q}_{x,t} & \tilde{Q}_{x,w,t} \\ \tilde{Q}_{x,w,t}^T & \tilde{Q}_{w,t} \end{bmatrix} \begin{bmatrix} x_t \\ w_t \end{bmatrix} + \begin{bmatrix} l_{x,t} \\ l_{w,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ w_t \end{bmatrix} + c_t \right) + \frac{1}{2} x_N^T \tilde{Q}_{x,N} x_N + l_{x,N}^T x_N + \tilde{c}_N,
\] (3.27)

where \( l_{x,t}, l_{w,t} \) and \( c_t \) are defined as
\[ l_{x,t} \triangleq \tilde{l}_{x,t} + \tilde{Q}_{x,v_t} v_t, \] (3.28a)
\[ l_{w,t} \triangleq \tilde{l}_{w,t} + \tilde{Q}_{w,v_t} v_t, \] (3.28b)
\[ c_t \triangleq \tilde{c}_t + \frac{1}{2} v_t^T \tilde{Q}_{v,t} v_t + \tilde{l}_{v,t}^T v_t. \] (3.28c)

Hence, by using this partition of the control input and the matrices, the UFTOC problem that is solved at AS iteration \( j \) to compute the Newton step can be written in the form
\[
\begin{align*}
\text{minimize} & \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t \\ w_t \end{bmatrix}^T \begin{bmatrix} \tilde{Q}_{x,t} & \tilde{Q}_{x,w,t} \\ \tilde{Q}_{x,w,t}^T & \tilde{Q}_{w,t} \end{bmatrix} \begin{bmatrix} x_t \\ w_t \end{bmatrix} + \begin{bmatrix} l_{x,t} \\ l_{w,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ w_t \end{bmatrix} + c_t \right) + \\
& \quad \frac{1}{2} x_N^T \tilde{Q}_{x,N} x_N + l_{x,N}^T x_N + \tilde{c}_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = \tilde{A}_t x_t + \tilde{B}_{w,t} w_t + \tilde{a}_{v,t}, \quad t \in \mathbb{Z}_{0,N-1}.
\end{align*}
\] (3.29)

### 3.3 Computing the Newton Step

As was indicated in the previous section, the computation of the Newton step is an important part when using both AS and IP methods to solve a CFTOC problem. Both types of methods require a sequence of Newton steps to be computed in order to solve the CFTOC problem, and this is where the most computational effort is used (Rao et al., 1998; Vandenbergh et al., 2002; Nocedal and Wright, 2006; Axehill et al., 2007). Hence, efficient numerical algorithms that compute the solution to the corresponding UFTOC problems are crucial to obtain high performance in many of these types of methods.
The equality constrained QP problems (3.21) and (3.29) are obtained when computing the search directions in IP and AS solvers when solving CFTOC problems. Both these optimization problems have the same structure as a UFTOC problem and hence, from a computational point of view, they can be interpreted as actually being UFTOC problems in the form

$$\begin{align*}
\min_{x,u} & \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_{x,t} & Q_{xu,t} \\ Q_{xu,t}^T & Q_{u,t} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} l_{x,t} \\ l_{u,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ u_t \end{bmatrix} + c_t \right) + \\
& \quad \frac{1}{2} x_N^T Q_{x,N} x_N + l_{x,N}^T x_N + c_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = A_t x_t + B_t u_t + a_t, \ t \in \mathbb{Z}_{0,N-1},
\end{align*}$$

(3.30)

where $x_t \in \mathbb{R}^{n_x}$ are the states, $u_t \in \mathbb{R}^{n_u,t}$ are the control inputs, and the equality constraints corresponds to the dynamics constraints of an affine dynamical system.

**Assumption 3.3.** $Q_t = \begin{bmatrix} Q_{x,t} & Q_{xu,t} \\ Q_{xu,t}^T & Q_{u,t} \end{bmatrix} \in \mathbb{S}^{n_x + n_u,t}_+, \ t \in \mathbb{Z}_{0,N-1}, \ Q_{x,N} \in \mathbb{S}^{n_x}_+.$

**Assumption 3.4.** $Q_{u,t} \in \mathbb{S}^{n_u,t}_+, \ t \in \mathbb{Z}_{0,N-1}.$

When assumptions 3.3 and 3.4 hold, the UFTOC problem has a unique optimal solution (Åström and Wittenmark, 1984; Maciejowski, 2002). Note that LICQ always holds for the UFTOC problem in (3.30).

If numerical differences are neglected, the number of problems in the sequence of UFTOC problems that are solved to compute the solution to a CFTOC problem is independent of how these UFTOC problems are solved. Hence, the overall relative performance gain in terms of computation time for solving the entire sequence of problems in order to solve the CFTOC problem is roughly the same as the relative performance gain obtained when solving a single UFTOC problem. However, note that in practice numerical errors in the solution of the UFTOC problems might result in that different numbers of UFTOC problems are required to be solved in order to compute the solution to the CFTOC problem. Investigating this is however outside the scope of this thesis.

**Definition 3.5.** Consider a constraint $a_i^T x - b_i = 0$ or $f_i(x) \leq 0$ in a convex optimization problem in the form in (2.7). Then the notation

$$\gamma_i \leftrightarrow a_i^T x - b_i = 0 \text{ or } \gamma_i \leftrightarrow f_i(x) \leq 0,$$

(3.31)

associates $\gamma_i$ to be the dual variable corresponding to constraint $i$.

In Definition 3.5 the notation $\leftrightarrow$ is introduced to indicate the connection between an equality or inequality constraint and the corresponding dual variable in a convex optimization problem. By extending the notation in Definition 3.5 to vector valued constraints, and introducing $\lambda_0, \ldots, \lambda_N \in \mathbb{R}^{n_x}$ as the dual variables associated
with the equality constraints in (3.30) gives

\[
\begin{align*}
\lambda_0 &\leftrightarrow -x_0 + \bar{x} = 0, \quad (3.32a) \\
\lambda_{t+1} &\leftrightarrow -x_{t+1} + A_t x_t + B_t u_t + a_t = 0, \quad t \in \mathbb{Z}_{0,N-1}. \quad (3.32b)
\end{align*}
\]

Note that the dual variables \(\lambda_t \in \mathbb{R}^{n_x}\) are vector valued, whereas in Definition 3.5 \(\gamma_i\) is a scalar. Furthermore, by also defining

\[
\lambda \triangleq \begin{bmatrix} \lambda_0 \\ \vdots \\ \lambda_N \end{bmatrix},
\]

the KKT conditions (2.15) for the UFTOC problem (3.30) are given by the linear system of equations

\[
\begin{bmatrix}
Q_x & Q_{xu} & A^T \\
Q_{xu}^T & Q_u & B^T \\
A & B & 0
\end{bmatrix}
\begin{bmatrix} x \\ u \\ \lambda \end{bmatrix} =
\begin{bmatrix} -l_x \\ -l_u \\ -a \end{bmatrix},
\]

where the highly structured \(Q_x, Q_{xu}, Q_u, l_x, l_u, A, B\) and \(a\) are defined as for problem (3.5) in Appendix 3.A. By re-arranging the variables and the equations in (3.34), the KKT system can be written in an equivalent, almost block diagonal, form given by

\[
\begin{bmatrix}
-I & -I & -I & \cdots & -I \\
-I & Q_{x,0} & Q_{xu,0} & A_0^T & 0 \\
-Q_{xu,0}^T & Q_{u,0} & B_0 & 0 & 0 \\
0 & 0 & A_0 & B_0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
-I & -I & Q_{x,N-1} & Q_{xu,N-1} & A_{N-1}^T \\
-Q_{xu,N-1}^T & Q_{u,N-1} & B_{N-1}^T & Q_{u,N-1} & B_{N-1} \\
A_{N-1} & B_{N-1} & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix} \lambda_0 \\ x_0 \\ u_0 \\ \lambda_1 \\ x_1 \\ \vdots \\ u_{N-1} \\ \lambda_N \\ x_N \end{bmatrix} =
\begin{bmatrix} -\bar{x} \\ -l_{x,0} \\ -l_{u,0} \\ -a_0 \\ -l_{x,1} \\ \vdots \\ -l_{x,N-1} \\ -l_{u,N-1} \\ -a_{N-1} \\ -l_{x,N} \end{bmatrix}.
\]

According to Theorem 2.8, a solution to the system of equations (3.35) is an optimal primal and dual solution to the UFTOC problem (3.30). Hence, solving the UFTOC problem (3.30), i.e., computing a Newton step in an IP or AS solver, corresponds to solving the linear system of equations in (3.35).

When assumptions 3.3 and 3.4 hold, the KKT matrix is non-singular (Åström and Wittenmark, 1984; Vandenberghe et al., 2002) and hence there exists a unique solution to (3.35). There exists many methods to solve the linear systems of equations in (3.35). The naive way to solve (3.35) would be to compute the inverse of the KKT matrix, but since the inverse is not explicitly needed, some factorization method is used in practise. Good references on numerical algorithms for solving linear systems of equations are for example Golub and Van Loan (1996) and Stewart (1998). Using a factorization method such as the LDL factorization,
that does not exploit the special structure in the KKT matrix, to compute the solution to (3.35) requires $O\left(N^3 (n_x + n_u)^3\right)$ floating-point operations. When a sequence of UFTOC problems in the form in (3.30) are solved, the solution to the KKT systems in the form in (3.35) can be solved by modifying the factorization instead of re-computing it from scratch, which requires $O\left(N^2 (n_x + n_u)^2\right)$ floating-point operations when no structure is exploited. However, the KKT matrix in (3.35) has a special almost block-diagonal structure, and some popular algorithms that exploit the sparse structure require only $O\left(N (n_x + n_u)^3\right)$ floating-point operations to solve (3.35). One such algorithm that can be used to solve (3.35) efficiently is the Riccati recursion (Jonson, 1983; Rao et al., 1998; Vandenberghe et al., 2002; Axehill, 2008).

3.4 Solving the Karush-Kuhn-Tucker System using the Riccati Recursion

The Riccati recursion factors the KKT matrix, and the factorization is followed by backward and forward substitutions for solving the system of equations. In this section the derivation of the Riccati recursion is presented, and it is also presented how to handle the case with a singular KKT matrix.

3.4.1 Derivation of the Riccati recursion

The derivation of the Riccati recursion that will be presented here is similar to what is presented in Rao et al. (1998) and Axehill (2005), but is here given for a UFTOC problem in the form in (3.30). The details of the derivation of the Riccati recursion will be needed when deriving the algorithms in chapters 5 and 6.

By inspecting the system of equations (3.35) it can be seen that the KKT system consists of $N$ blocks of equations in the form

$$
\begin{bmatrix}
-I & Q_{x,t} & Q_{xu,t} \\
0 & Q_{xu,t}^T & Q_{u,t} \\
0 & A_t & B_t
\end{bmatrix}
\begin{bmatrix}
\lambda_t \\
x_t \\
u_t
\end{bmatrix} +
\begin{bmatrix}
A_t^T & 0 \\
B_t^T & 0 \\
0 & -I
\end{bmatrix}
\begin{bmatrix}
\lambda_{t+1} \\
x_{t+1} \\
u_{t+1}
\end{bmatrix} =
\begin{bmatrix}
-l_{x,t} \\
l_{u,t} \\
-a_t
\end{bmatrix},
$$

$t \in \mathbb{Z}_{0,N-1}$, (3.36)

and the two boundary equations

$$
x_0 = \bar{x},
$$

$$
-\lambda_N + Q_{x,N} x_N = -l_{x,N}.
$$

An induction argument will now be used to show that there exist a matrix $P_t \in S_n^{nx}$ and a vector $\Psi_t \in \mathbb{R}^{nx}$ for $t \in \mathbb{Z}_{0,N}$ such that

$$
P_t x_t - \lambda_t = \Psi_t,
$$

(3.38)

holds. Furthermore, it will be shown how the property in (3.38) can be exploited to solve the system of equations (3.35).

First, note that the choice $P_N = Q_{x,N}$ and $\Psi_N = -l_{x,N}$ in (3.38) satisfies the
boundary equation (3.37b). Now assume that (3.38) holds for some arbitrary \( t+1 \) with \( P_{t+1} \in S_{++}^{n_{x}} \) and \( t \in \mathbb{Z}_{0,N-1} \). Inserting \( \lambda_{t+1} = P_{t+1} x_{t+1} - \Psi_{t+1} \) into (3.36) gives the equivalent system of equations

\[
\begin{bmatrix}
-I & Q_{x,t} & Q_{xu,t} \\
0 & Q_{xu,t} & Q_{u,t} \\
0 & A_{t} & B_{t}
\end{bmatrix}
\begin{bmatrix}
\lambda_{t} \\
x_{t} \\
u_{t}
\end{bmatrix}
+ \begin{bmatrix}
A_{t}^{T} P_{t+1} \\
B_{t}^{T} P_{t+1} \\
-I
\end{bmatrix}
\begin{bmatrix}
x_{t+1} \\
-\Psi_{t+1}
\end{bmatrix}
= \begin{bmatrix}
l_{x,t} \\
l_{u,t}
\end{bmatrix}.
\] (3.39)

By eliminating \( x_{t+1} \) from (3.39) using the dynamics constraints in the last block equation in (3.39), this system of equations can equivalently be written as

\[
\begin{bmatrix}
-I & Q_{x,t} + A_{t}^{T} P_{t+1} A_{t} & Q_{xu,t} + A_{t}^{T} P_{t+1} B_{t} \\
0 & Q_{xu,t} + B_{t}^{T} P_{t+1} A_{t} & Q_{u,t} + B_{t}^{T} P_{t+1} B_{t}
\end{bmatrix}
\begin{bmatrix}
\lambda_{t} \\
x_{t} \\
u_{t}
\end{bmatrix}
+ \begin{bmatrix}
A_{t}^{T} \\
B_{t}^{T}
\end{bmatrix}
(P_{t+1} a_{t} - \Psi_{t+1})
= \begin{bmatrix}
l_{x,t} \\
l_{u,t}
\end{bmatrix}.
\] (3.40)

Now, introduce the auxiliary variables \( F_{t+1}, H_{t+1} \) and \( G_{t+1} \) as

\[
F_{t+1} \triangleq Q_{x,t} + A_{t}^{T} P_{t+1} A_{t} \in S_{++}^{n_{x}}, \quad (3.41a)
\]

\[
H_{t+1} \triangleq Q_{xu,t} + A_{t}^{T} P_{t+1} B_{t} \in \mathbb{R}^{n_{x} \times n_{u,t}}, \quad (3.41b)
\]

\[
G_{t+1} \triangleq Q_{u,t} + B_{t}^{T} P_{t+1} B_{t} \in S_{++}^{n_{u,t}}. \quad (3.41c)
\]

Then, the block equations in (3.40) can equivalently be written as

\[
\begin{bmatrix}
-I & F_{t+1} & H_{t+1} \\
0 & H_{t+1} & G_{t+1}
\end{bmatrix}
\begin{bmatrix}
\lambda_{t} \\
x_{t} \\
u_{t}
\end{bmatrix}
+ \begin{bmatrix}
A_{t}^{T} \\
B_{t}^{T}
\end{bmatrix}
(P_{t+1} a_{t} - \Psi_{t+1})
= \begin{bmatrix}
l_{x,t} \\
l_{u,t}
\end{bmatrix}.
\] (3.42)

Furthermore, implicitly define the auxiliary variables \( K_{t+1} \) and \( k_{t+1} \) by using the systems of equations

\[
G_{t+1} K_{t+1} \triangleq -H_{t+1}^{T}, \quad (3.43a)
\]

\[
G_{t+1} k_{t+1} \triangleq -(l_{u,t} + B_{t}^{T} P_{t+1} a_{t} - B_{t}^{T} \Psi_{t+1}). \quad (3.43b)
\]

Then, by multiplying (3.42) with the invertible matrix

\[
\begin{bmatrix}
I & K_{t+1}^{T} \\
0 & I
\end{bmatrix},
\] (3.44)

from the left and using (3.43a) gives the equivalent system of equations

\[
\begin{bmatrix}
-I & F_{t+1} + K_{t+1}^{T} H_{t+1} & 0 \\
0 & H_{t+1} & G_{t+1}
\end{bmatrix}
\begin{bmatrix}
\lambda_{t} \\
x_{t} \\
u_{t}
\end{bmatrix}
+ \begin{bmatrix}
(A_{t} + B_{t} K_{t+1})^{T} \\
B_{t}^{T}
\end{bmatrix}
(P_{t+1} a_{t} - \Psi_{t+1})
= \begin{bmatrix}
l_{x,t} - K_{t+1}^{T} l_{u,t} \\
l_{u,t}
\end{bmatrix}.
\] (3.45)
From the implicit definitions of $K_{t+1}$ and $k_{t+1}$ in (3.43) it follows that the feedback
\[ u_t = K_{t+1}x_t + k_{t+1}, \]  
(3.46)
satisfies the lower block equations in (3.45). Furthermore, by using (3.43a) it follows that the first block equation in (3.45) can be written as
\[ -\lambda_t + (F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1})x_t + (A_t + B_t K_{t+1})^T (P_{t+1} a_t - \Psi_{t+1}) = -l_{x,t} - K_{t+1}^T l_{u,t} \iff -\lambda_t + (F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1})x_t = A_t^T \Psi_{t+1} - A_{t+1}^T P_{t+1} a_t - B_{t+1}^T \Psi_{t+1}. \]  
(3.47)

By identifying the terms in (3.47) with the ones in (3.38), it follows that (3.47) can be written
\[ -\lambda_t + P_t x_t = \Psi_t, \]  
(3.48)
with
\[ P_t \triangleq F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1}, \]  
(3.49a)
\[ \Psi_t \triangleq A_t^T \Psi_{t+1} - A_{t+1}^T P_{t+1} a_t - l_{x,t} - K_{t+1}^T (l_{u,t} + B_t^T P_{t+1} a_t - B_{t+1}^T \Psi_{t+1}). \]  
(3.49b)

By using the definitions of $K_{t+1}$ and $k_{t+1}$ from (3.43) and $G_{t+1} \in S_{++}^{n_u,t}$, it follows that (3.49b) can equivalently be written
\[ \Psi_t = A_t^T \Psi_{t+1} - A_{t+1}^T P_{t+1} a_t - l_{x,t} - H_{t+1} k_{t+1}. \]  
(3.50)

Furthermore, define the auxiliary matrix
\[ M_{t+1} \triangleq \begin{bmatrix} F_{t+1} & H_{t+1} \\ H_{t+1}^T & G_{t+1} \end{bmatrix} = \begin{bmatrix} Q_{x,t} & Q_{x,t}^T \\ Q_{x,t} & Q_{u,t} \end{bmatrix} + \begin{bmatrix} A_t^T & \Psi_{t+1} \\ B_t^T \end{bmatrix} = S_{++}^{n_x+n_u,t}, \]  
(3.51)
which is positive semidefinite by construction. Then it follows from the definition of $K_{t+1}$ in (3.43a) and $G_{t+1} \in S_{++}^{n_u,t}$ that $P_t$ can be equivalently computed as
\[ P_t = F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1} = F_{t+1} - H_{t+1} G_{t+1}^{-1} H_{t+1}^T = M_{t+1} / G_{t+1} \in S_{++}^{n_x}. \]  
(3.52)
The last equality follows from the definition of the Schur complement in Definition A.6, and $P_t \in S_{++}^{n_x}$ follows from Theorem A.10, both given in Appendix A.

Hence, if there exist a $P_{t+1} \in S_{++}^{n_x}$ and a $\Psi_{t+1} \in \mathbb{R}^{n_x}$ such $P_{t+1} x_{t+1} - \lambda_{t+1} = \Psi_{t+1}$ for some $t \in \mathbb{Z}_{0,N-1}$, then it also exist a $P_t \in S_{++}^{n_x}$ and a $\Psi_t \in \mathbb{R}^{n_x}$ such that $P_t x_t - \lambda_t = \Psi_t$. This holds for an arbitrary $t \in \mathbb{Z}_{0,N-1}$. 
Remark 3.6. In the derivation of $P_t$ in (3.49a), the non-singularity of $G_{t+1} \in S^{nu,t}_{+}$ is not exploited. Hence, the definition of $P_t$ in (3.49a) holds even when $G_{t+1} \in S^{nu,t}_{+}$.

Note that the matrix $P_t$ in (3.52) is unique, which can be derived from the uniqueness of $F_{t+1}$, $H_{t+1}$ and $G_{t+1}$. Furthermore, it follows from the definitions of $F_{t+1}$, $H_{t+1}$ and $G_{t+1}$ in (3.41) that (3.52) is the discrete-time Riccati equation (Rao et al., 1998; Bertsekas, 2000)

$$P_t = Q_{x,t} + A_t^T P_{t+1} A_t - (Q_{xu,t} + A_t^T P_{t+1} B_t)(Q_{u,t} + B_t^T P_{t+1} B_t)^{-1}(Q_{xu,t} + B_t^T P_{t+1} A_t).$$

(3.53)

Furthermore, by denoting the cost-to-go function (Bertsekas, 2000) at time $t$ for the uftoc problem (3.30) as $J(x_t)$, it follows that

$$J(x_t) \triangleq x_t^T P_t x_t - \Psi_t^T x_t + \tilde{c}_t,$$

(3.54)

where $\tilde{c}_t \in \mathbb{R}$ can be computed as

$$\tilde{c}_N \triangleq c_N,$$

(3.55a)

$$\tilde{c}_t \triangleq \tilde{c}_{t+1} + c_t + \frac{1}{2} a_t^T P_{t+1} a_t - \Psi_{t+1}^T a_t - \frac{1}{2} k_{t+1}^T G_{t+1} k_{t+1}, \quad t \in \mathbb{Z}_{0,N-1}.$$  

(3.55b)

See Bertsekas (2000) for the details for the definition of the cost-to-go function.

From the definitions of $P_t$ and $\Psi_t$ in (3.49), and of $K_{t+1}$ and $k_{t+1}$ in (3.43), it follows that when these have been computed, the solution to the KKT system (3.35) is given by

$$x_0 = \bar{x},$$

(3.56a)

$$u_t = k_{t+1} + K_{t+1} x_t, \quad t \in \mathbb{Z}_{0,N-1},$$

(3.56b)

$$x_{t+1} = A_t x_t + B_t u_t + a_t, \quad t \in \mathbb{Z}_{0,N-1},$$

(3.56c)

$$\lambda_t = P_t x_t - \Psi_t, \quad t \in \mathbb{Z}_{0,N}.$$  

(3.56d)

Here (3.56b) follows directly from (3.46). This is summarized in algorithms 4, 5 and 6. The computation of $P_t$, $H_{t+1}$, $F_{t+1}$, $G_{t+1}$ and $K_{t+1}$ can be interpreted as a factorization of the KKT matrix in (3.35) (Rao et al., 1998; Vandenberghe et al., 2002) and is summarized in Algorithm 4. This factorization will be referred to as the Riccati factorization in this thesis. Algorithm 5 is a backward recursion where the variables $k_{t+1}$ and $\Psi_t$ are computed. Finally, to compute the solution to the uftoc problem, the forward recursion in Algorithm 6 is used. At Line 6 in Algorithm 4, $G_{t+1}$ is factored using a suitable factorization such as the Cholesky factorization. The factorization of $G_{t+1}$ can be used to solve the system of equations at Line 7 in Algorithm 4 and at Line 3 in Algorithm 5. Hence, the matrix $G_{t+1}$ is never explicitly inverted and the factorization is reused during several operations. See for example Golub and Van Loan (1996) or Stewart (1998) for more details on different factorization techniques. In Algorithm 5, the backward recursion to compute the $\tilde{c}_t$ in the cost-to-go function in (3.54) is included.
3.4 Solving the Karush-Kuhn-Tucker System using the Riccati Recursion

### Algorithm 4 Riccati factorization

1. \( P_N := Q_{x,N} \)
2. \( \text{for } t = N - 1, \ldots, 0 \text{ do} \)
3. \( F_{t+1} := Q_{x,t} + A_t^T P_{t+1} A_t \)
4. \( G_{t+1} := Q_{u,t} + B_t^T P_{t+1} B_t \)
5. \( H_{t+1} := Q_{xu,t} + A_t^T P_{t+1} B_t \)
6. \( \text{Compute and store a factorization of } G_{t+1} \)
7. \( \text{Compute a solution } K_{t+1} \text{ to} \)
8. \( G_{t+1} K_{t+1} = -H_{t+1}^T \)
9. \( P_t := F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1} \)
10. \( \text{end for} \)

### Algorithm 5 Backward recursion

1. \( \Psi_N := -l_{x,N}, \bar{c}_N := c_N \)
2. \( \text{for } t = N - 1, \ldots, 0 \text{ do} \)
3. \( \text{Compute a solution } k_{t+1} \text{ to} \)
4. \( G_{t+1} k_{t+1} = B_t^T \Psi_{t+1} - l_{u,t} - B_t^T P_{t+1} a_t \)
5. \( \Psi_t := A_t^T \Psi_{t+1} - H_{t+1}^T k_{t+1} - l_{x,t} - A_t^T P_{t+1} a_t \)
6. \( \bar{c}_t := \bar{c}_{t+1} + c_t + \frac{1}{2} a_t^T P_{t+1} a_t - \Psi_{t+1} a_t - \frac{1}{2} k_{t+1}^T G_{t+1} k_{t+1} \)
7. \( \text{end for} \)

#### 3.4.2 Handling singular Karush-Kuhn-Tucker systems

Some methods for solving CFTOC problems require the factorization of the KKT matrix of a UFTOC problem in the form in (3.30), but where Assumption 3.4 does not hold. Examples of such methods are dual AS methods and dual gradient projection methods as the one presented in Axehill (2008) and Axehill and Hansson (2008). In such problems, the matrices \( G_{t+1} \) in the Riccati factorization might be singular and hence the matrices \( K_{t+1} \) might be non-unique. However, \( P_t \) is still unique even in the case of a singular KKT matrix (Axehill, 2008).

If \( G_{t+1} \) is singular, then there might not exist any solution to the system of equations at Line 7 in Algorithm 4, i.e.,

\[
G_{t+1} K_{t+1} = -H_{t+1}^T, \tag{3.57}
\]

or if it is solvable, the solution is not unique. From Theorem A.10 and Lemma A.4 it follows that the system of equations in (3.57) is actually solvable since

\[
M_{t+1} = \begin{bmatrix}
F_{t+1} & H_{t+1} \\
H_{t+1}^T & G_{t+1}
\end{bmatrix} \in S_+^{n_x + n_{u,t}}, \tag{3.58}
\]

giving \( \mathcal{R}(H_{t+1}^T) \subset \mathcal{R}(G_{t+1}) \). Hence, the Riccati factorization exists even in the case of a singular KKT matrix, but \( K_{t+1} \) is not unique. Furthermore, by using the basic property (A.1a) of the pseudo-inverse of \( G_{t+1} \), i.e.,

\[
G_{t+1} = G_{t+1} G_{t+1}^\dagger G_{t+1}, \tag{3.59}
\]
Algorithm 6 Forward recursion

1: \( x_0 := \bar{x} \)
2: for \( t = 0, \ldots, N - 1 \) do
3: \( u_t := k_{t+1} + K_{t+1} x_t \)
4: \( x_{t+1} := A_t x_t + B_t u_t + a_t \)
5: \( \lambda_t := P_t x_t - \Psi_t \)
6: end for
7: \( \lambda_N := P_N x_N - \Psi_N \)

together with (3.43a), (3.49a) and Remark 3.6, it follows that \( P_t \) can be computed as in (3.52) but using the pseudo-inverse, giving

\[
P_t = F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1} = F_{t+1} - K_{t+1}^T G_{t+1} \dagger \ \alpha_{t+1} - H_{t+1} G_{t+1}^\dagger H_{t+1}^T. \tag{3.60}
\]

The expression in (3.60) can be recognized as the generalized Schur complement of \( G_{t+1} \) in \( M_{t+1} \) given by

\[
P_t = M_{t+1}/\dagger G_{t+1} \in S_{++}^n, \tag{3.61}
\]

which is introduced in Definition A.5. Here, positive semidefiniteness of \( P_t \) follows from Theorem A.10. Furthermore, since \( F_{t+1}, H_{t+1} \) and \( G_{t+1}^\dagger \) are unique, it follows that also \( P_t \) is unique. However, when \( G_{t+1} \) is singular \( K_{t+1} \) is non-unique.

Even though the Riccati factorization always exists for a singular KKT matrix, it is not certain that a solution exists to the linear systems of equations (3.35), or the solution is not unique. Since \( K_{t+1} \) can always be computed, the existence of a solution to the KKT system is determined by the existence of a solution to (3.43b), corresponding to Line 3 in Algorithm 5. If no solution \( k_{t+1} \) exists to the system of equations in (3.43b), then the KKT system is not solvable. If the KKT system is not solvable, then it should be reported to the solver applied to solve the CFTOC problem that the search direction can not be computed by solving the UFTOC problem. However, the Riccati factorization might still be useful as is shown in Axehill (2008). It is then the task of the specific solver to handle this case at the level above the numerical linear algebra.

If on the other hand (3.43b) is solvable, then it follows from Lemma A.4 that

\[
\kappa_t \triangleq \left( l_{u,t} + B_t^T P_{t+1} a_t - B_t^T \Psi_{t+1} \right) \in \mathbb{R}(G_{t+1}). \tag{3.62}
\]

From Penrose (1955) it follows that the solutions to (3.43) are not unique when \( G_{t+1} \) is singular, and that all solutions to (3.43a) and (3.43b) are given by

\[
K_{t+1} \triangleq -G_{t+1}^\dagger H_{t+1}^T + \left( I - G_{t+1}^\dagger G_{t+1} \right) Z_{t+1}, \tag{3.63a}
\]

\[
k_{t+1} \triangleq -G_{t+1}^\dagger \kappa_t + \left( I - G_{t+1}^\dagger G_{t+1} \right) z_{t+1}. \tag{3.63b}
\]

Here \( Z_{t+1} \in \mathbb{R}^{n_{u,t} \times n_x} \) and \( z_{t+1} \in \mathbb{R}^{n_{u,t}} \) denote an arbitrary matrix and vector,
respectively, and $\kappa_t$ is defined in (3.62). Note that $\left(I - G_{t+1}^\dagger G_{t+1}\right)$ is the orthogonal projection onto $\mathcal{N}(G_{t+1})$ (Golub and Van Loan, 1996). Hence, from (3.56b) and the properties of orthogonal projections it follows that $u_t$ can be computed as

$$u_t = -G_{t+1}^\dagger H_{t+1}^T x_t - G_{t+1}^\dagger \kappa_t + u_t^N, \quad u_t^N \in \mathcal{N}(G_{t+1}).$$

(3.64)

Furthermore, from (3.63) and $\mathcal{R}(H_{t+1}^T) \subset \mathcal{R}(G_{t+1})$, together with the definition of $\kappa_t$ in (3.62) and the assumption that the $\kkt$ system is solvable, it follows that

$$K_{t+1}^T \kappa_t = -H_{t+1} G_{t+1}^\dagger \kappa_t = H_{t+1} k_{t+1}.$$  

(3.65)

Here $\kappa_t \in \mathcal{R}(G_{t+1}) \iff \left(I - G_{t+1} G_{t+1}^\dagger\right) \kappa_t = 0$, and $\left(I - G_{t+1} G_{t+1}^\dagger\right) H_{t+1}^T = 0$ have been used. From (3.65) it follows that for the case when $G_{t+1}$ is singular and the KKT system (3.35) is solvable, it is still possible to compute $\Psi_t$ as in (3.50).

In the case of non-unique solutions to the KKT system, the solver that is applied to solve the CFTOC problem must decide on which solution to choose. Conceptually, this choice is made at the level above the numerical linear algebra.

### 3.5 Computing the Eliminated Dual Variables

In Section 3.2.2 it was shown how the Newton step can be computed when using an AS method to solve the CFTOC problem (3.22). The UFTOC problem that is solved as a subproblem in the AS method is obtained by eliminating the variables $v_t$ that are constrained to fixed values by the inequality constraints in the working set. In Section 2.6 it was shown that the dual variables corresponding to the inequality constraints are important in an AS solver, and in this section it will be shown how to compute the dual variables associated with the inequality constraints when computing the Newton step as in Section 3.2.2.

The inequality box constraints $u_{\min,t} \leq u_t \leq u_{\max,t}$ in (3.22) can equivalently be written

$$\begin{bmatrix} I \\ -I \end{bmatrix} u_t \preceq \begin{bmatrix} u_{\max,t} \\ -u_{\min,t} \end{bmatrix}.$$ 

(3.66)

Let the dual variables $\mu_t \in \mathbb{R}^{2n_u,t}$ associated with these inequality constraints be introduced as

$$\mu_t \leftrightarrow \begin{bmatrix} I \\ -I \end{bmatrix} u_t - \begin{bmatrix} u_{\max,t} \\ -u_{\min,t} \end{bmatrix} \preceq 0.$$  

(3.67)

Now, using a similar partition and notation as in Section 3.2.2, let $\mu_{v,t}$ be the dual variables corresponding to the constraints in the working set $\mathcal{W}_j$ and let $\mu_{w,t}$ be the dual variables corresponding to the constraints that are temporarily disregarded at AS iteration $j$. The dual variables $\mu_{w,t}$ are defined to be zero, $i.e., \mu_{w,t} \equiv 0$ (Nocedal and Wright, 2006). In order to compute $\mu_{v,t}$, the KKT system for the equality constrained QP problem (3.29) before eliminating the fixed $v_t$ is studied. Let $\tilde{v}_t$ denote the fixed values of $v_t$, $i.e., v_t = \tilde{v}_t$. 
Algorithm 7 Forward recursion - eliminated dual variables

1: for $t = 0, \ldots, N - 1$ do
2: \[ \mu_{w,t} := 0 \]
3: \[ \mu_{v,t} := l_{v,t} + Q_{xv,t}^T x_t + Q_{wv,t}^T w_t + Q_{v,t} v_t + B_{v,t}^T \lambda_{t+1} \]
4: end for

Using the partitioning of the control input as in Section 3.2.2 and the definition of $\mu_t$ and $\bar{v}_t$ in this section, it can be seen that a single block of equations in the KKT system for the equality constrained QP problem before eliminating $v_t$ has the structure

\[
\begin{bmatrix}
\vdots \\
-I & Q_{x,t} & Q_{xw,t} & Q_{xv,t} & 0 & A_t^T & 0 & \cdots \\
0 & Q_{xw,t}^T & Q_{w,t} & Q_{wv,t} & 0 & B_{w,t}^T & 0 \\
0 & Q_{xv,t}^T & Q_{v,t}^T & -I & B_{v,t}^T & 0 \\
0 & 0 & 0 & -I & 0 & 0 & 0 \\
0 & A_t & B_{w,t} & B_{v,t} & 0 & 0 & -I \\
\end{bmatrix}
\begin{bmatrix}
\lambda_t \\
x_t \\
w_t \\
v_t \\
\mu_{v,t} \\
\lambda_{t+1} \\
x_{t+1} \\
\end{bmatrix}
= 
\begin{bmatrix}
\vdots \\
-l_{x,t} \\
-l_{w,t} \\
-l_{v,t} \\
-\bar{v}_t \\
-a_t \\
-l_{x,t+1} \\
\end{bmatrix}
\]

(3.68)

The fourth block row in (3.68) was used in Section 3.2.2 to eliminate the fixed part of the control input. From the third block row in (3.68) it follows that $\mu_{v,t}$ can be computed from

\[- \mu_{v,t} + Q_{xv,t}^T x_t + Q_{wv,t}^T w_t + Q_{v,t} v_t + B_{v,t}^T \lambda_{t+1} = -l_{v,t}, \]

(3.69)

which is summarized in Algorithm 7. Note that by eliminating $v_t$ and $\mu_{v,t}$ from (3.68), the KKT system (3.68) reduces to the KKT system for the UFTOC problem (3.29).
Appendix

In this appendix the matrices, vectors and constants used in the compact formulations (3.5) and (3.34) of the CFTOC problem and KKT system, respectively, are defined.

3.A Model Predictive Control Formulations

The matrices and vectors of stacked elements in the problem (3.5) are defined as

\[
\tilde{Q}_x \triangleq \begin{bmatrix} \tilde{Q}_{x,0} & & \\ & \ddots & \\ & & \tilde{Q}_{x,N} \end{bmatrix}, \quad \tilde{Q}_{xu} \triangleq \begin{bmatrix} \tilde{Q}_{xu,0} & & \\ & \ddots & \\ & & \tilde{Q}_{xu,N-1} \end{bmatrix}, \quad \tilde{Q}_u \triangleq \begin{bmatrix} \tilde{Q}_{u,0} & & \\ & \ddots & \\ & & \tilde{Q}_{u,N-1} \end{bmatrix},
\]

\[ \tilde{A} \triangleq \begin{bmatrix} -I & & \\ \tilde{A}_0 & -I & \\ & \ddots & \ddots \\ & & \tilde{A}_{N-1} & -I \end{bmatrix}, \quad \tilde{B} \triangleq \begin{bmatrix} 0 & & \\ \tilde{B}_0 & \ddots & \\ & \ddots & \ddots \\ & & \tilde{B}_{N-1} \end{bmatrix}, \quad \tilde{a} \triangleq \begin{bmatrix} \tilde{x} \\ \tilde{a}_0 \\ \vdots \\ \tilde{a}_{N-1} \end{bmatrix}, \quad \text{(3A.70a)}
\]

\[
\tilde{H}_x \triangleq \begin{bmatrix} \tilde{H}_{x,0} & & \\ & \ddots & \\ & & \tilde{H}_{x,N} \end{bmatrix}, \quad \tilde{H}_{xu} \triangleq \begin{bmatrix} \tilde{H}_{xu,0} & & \\ & \ddots & \\ & & \tilde{H}_{xu,N-1} \end{bmatrix}, \quad \tilde{h} \triangleq \begin{bmatrix} \tilde{h}_0 \\ \vdots \\ \tilde{h}_{N-1} \end{bmatrix}, \quad \text{(3A.70b)}
\]

\[
\tilde{l}_x \triangleq \begin{bmatrix} \tilde{l}_{x,0} & & \\ & \ddots & \\ & & \tilde{l}_{x,N} \end{bmatrix}, \quad \tilde{l}_{xu} \triangleq \begin{bmatrix} \tilde{l}_{xu,0} & & \\ & \ddots & \\ & & \tilde{l}_{xu,N-1} \end{bmatrix}, \quad \tilde{x} \triangleq \begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix}, \quad \tilde{u} \triangleq \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}, \quad \tilde{c} \triangleq \sum_{t=0}^{N} \tilde{c}_t. \quad \text{(3A.70d)}
\]

For the problem (3.30) the matrices and vectors in (3.34) are defined analogously.
3.B Moving Horizon Estimation Formulations

The matrices, vectors and scalars in the CFTOC problem (3.11) that correspond to the MHE optimization problem (3.9) are defined in this section. For the end points \( t = 0 \) and \( t = N \) the definitions are given by

\[
\begin{align*}
\tilde{Q}_{x,0} & \triangleq 0, \quad \tilde{Q}_{xw,0} \triangleq 0, \quad \tilde{Q}_{w,0} \triangleq \tilde{P}_0^{-1}, \quad \tilde{l}_{x,0} \triangleq 0, \\
\tilde{l}_{w,0} & \triangleq 0, \quad \tilde{c}_0 \triangleq 0, \quad \tilde{Q}_{x,N} \triangleq 0, \quad \tilde{l}_{x,N} \triangleq 0, \quad \tilde{c}_N \triangleq 0,
\end{align*}
\]

(3B.71a)

\[
\begin{align*}
\tilde{A}_0 & \triangleq I, \quad \tilde{B}_0 = I, \quad \tilde{a}_0 \triangleq 0,
\end{align*}
\]

(3B.71b)

For \( t + 1 \in \mathbb{Z}_{1,N-1} \) the definitions are given by

\[
\begin{align*}
\begin{bmatrix} \tilde{Q}_{x,t+1} & \tilde{Q}_{xw,t+1} \\ \tilde{Q}_{xw,t+1}^T & \tilde{Q}_{w,t+1} \end{bmatrix} & \triangleq \begin{bmatrix} \bar{C}_t^T V_t \bar{C}_t & -\bar{C}_t^T S_t^T \\ -S_t \bar{C}_t & W_t \end{bmatrix}, \\
\tilde{l}_{x,t+1} & \triangleq \bar{C}_t^T \left( S_t^T \tilde{w}_t - V_t \bar{y}_t \right), \quad \tilde{l}_{w,t+1} \triangleq S_t \bar{y}_t - W_t \tilde{w}_t,
\end{align*}
\]

(3B.72a)

\[
\begin{align*}
\tilde{c}_{t+1} & \triangleq \frac{1}{2} \bar{y}_t^T V_t \bar{y}_t - \bar{w}_t^T S_t \bar{y}_t + \frac{1}{2} \bar{w}_t^T W_t \bar{w}_t, \\
\tilde{H}_x,t+1 & \triangleq \bar{H}_x,t, \quad \tilde{H}_{w,t+1} \triangleq \bar{H}_{w,t}, \quad \tilde{h}_{t+1} \triangleq \bar{h}_t,
\end{align*}
\]

(3B.72b)

where \( V_t, W_t, S_t \) and \( \bar{y}_t \) are defined as

\[
\begin{align*}
\begin{bmatrix} W_t & V_t \\ S_t^T & \bar{V}_t \end{bmatrix} & \triangleq \begin{bmatrix} \bar{Q}_{w,t} & \bar{Q}_{w,v,t} \\ \bar{Q}_{w,v,t} & \bar{Q}_{v,t} \end{bmatrix}^{-1}, \quad t \in \mathbb{Z}_{0,N-2}, \\
\bar{y}_t & \triangleq \bar{y}_t - \bar{d}_t - \bar{v}_t, \quad t \in \mathbb{Z}_{0,N-2}.
\end{align*}
\]

(3B.73a)

(3B.73b)
In parametric programming, the optimization problem depends on a parameter which can be thought of as input data to the optimization problem (Bank et al., 1982). When the optimization problem is dependent on several parameters it is referred to as multiparametric programming. One class of such problems that has proved to be useful in control is mp-QP problems. See for example Bank et al. (1982) for a survey on parametric programming. In this chapter, the mp-QP problem is introduced and the solution to the mp-QP problem is presented.

The purpose of this chapter is to introduce the notation and basic properties of the mp-QP problem and its solution to form a foundation for Chapter 7, where the structure of the parametric solution will be exploited. Solving the mp-QP problem is outside the scope of this thesis. For the details on popular methods for solving mp-QP problems, see for example Bemporad et al. (2002), Tøndel et al. (2003c), Spjøtvold et al. (2006), and Gupta et al. (2011).

### 4.1 Problem Definition

Consider an optimization problem in the form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}u^T Hu + \theta^T Fu \\
\text{subject to} & \quad Gu \preceq b + E\theta \\
& \quad \theta \in \Theta,
\end{align*}
\tag{4.1}
\]

where \( u \in \mathbb{R}^m \) is the optimization variable, \( \theta \in \mathbb{R}^n \) is the parameter, the cost function is determined by \( H \in \mathbb{S}_+^m \) and \( F \in \mathbb{R}^{n \times m} \), the inequality constraints
are given by $G \in \mathbb{R}^{p \times m}$, $b \in \mathbb{R}^p$ and $E \in \mathbb{R}^{p \times n}$, and $\Theta$ is a polyhedral set. The problem (4.1) is an mp-QP problem with parameter $\theta$ (Bemporad et al., 2002). By introducing the change of variables

$$z \triangleq u + H^{-1}F^T \theta,$$ (4.2)

the mp-QP problem (4.1) can be transformed into the equivalent mp-QP problem

$$\min_z \quad \frac{1}{2} z^T H z$$
subject to
$$G z \preceq b + S \theta$$
$$\theta \in \Theta,$$ (4.3)

where $S \triangleq E + GH^{-1}F^T \in \mathbb{R}^{p \times n}$. For any choice of parameter $\theta \in \Theta$, the problem (4.3) is a strictly convex QP problem, and the necessary and sufficient optimality conditions are given by the KKT conditions introduced in Definition 2.7. For the mp-QP problem in (4.3) these are given by

$$Hz + G^T \lambda = 0,$$ (4.4a)
$$G z \preceq b + S \theta,$$ (4.4b)
$$\lambda \succeq 0,$$ (4.4c)
$$\lambda_k (G_k z - b_k - S_k \theta) = 0, \quad k \in \mathbb{Z}_{1:p}.$$ (4.4d)

Here $\lambda \leftrightarrow G z - b - S \theta \preceq 0$ are the Lagrange multipliers associated with the inequality constraints (Nocedal and Wright, 2006). It is shown in Bemporad et al. (2002) that the solution to the mp-QP problem (4.3) is given by the pwa function

$$z^*(\theta) = k_i + K_i \theta \quad \text{if} \quad \theta \in \mathcal{P}_i, \quad i \in \mathbb{Z}_{1,R},$$ (4.5)

where $k_i \in \mathbb{R}^m$, $K_i \in \mathbb{R}^{m \times n}$ define the parametric solution in the polyhedral critical region $\mathcal{P}_i$ for $i \in \mathbb{Z}_{1,R}$. It is shown in Tøndel et al. (2003c) that each parameter $\theta \in \mathcal{P}_i$ and solution $z^*(\theta)$ correspond to one or more optimal active sets. Hence, it follows that also each critical region $\mathcal{P}_i$ corresponds to one or more optimal active sets (Tøndel et al., 2003c). When LICQ and SC hold for, the optimal active set is unique (Gupta et al., 2011).

The pwa function (4.5) and the $R$ critical regions are computed by solving the mp-QP problem (4.3) parametrically, basically by considering all possible combinations of optimal active constraints. In Bemporad et al. (2002) and Tøndel et al. (2003c) the critical regions and the corresponding optimal active sets are determined by geometric approaches for exploring the parameter space. In Bemporad et al. (2002) the parameter space is searched by introducing artificial cuts, which might introduce unnecessary regions. The algorithm in Tøndel et al. (2003c) exploits the relation between neighboring critical regions and the optimal active sets, and relies on the assumption that the facet-to-facet property holds. This algorithm is reported to avoid unnecessary partitioning. In Spjøtvold et al. (2006) the mp-QP problem is solved using a combination of the approaches in Tøndel et al. (2003c) and Bemporad et al. (2002), where the requirement for the facet-to-facet property to hold is relaxed. In Gupta et al. (2011) the mp-QP problem is solved using an
4.2 Characterizing the Solution for an Optimal Active Set

In Chapter 7, the relation between the critical region and the optimal active set will be exploited. Hence, although this thesis does not consider the problem of actually solving the mp-QP problem, it is important to present the structure of the solution. In this section it will be shown how the parametric solution relates to the optimal active set, and in Section 4.3 it will be shown how a critical region relates to the corresponding optimal active set.

For any feasible choice of the parameter $\theta \in \Theta$, the set of indices of constraints that hold with equality at the optimum is called the optimal active set. Let the set of all optimal active sets be denoted $A^*$, and let the elements in $A^*$ be denoted $A_i$ for $i \in \mathbb{Z}_{1,R}$. Let $N_i$ be the set of inactive constraints, satisfying $A_i \cup N_i = \mathbb{Z}_{1,p}$ and $A_i \cap N_i = \emptyset$. Let $G_{A_i}$ and $G_{N_i}$ be the matrices consisting of the rows in $G$ that are indexed by $A_i$ and $N_i$, respectively, and let the same hold for matrices and vectors $S_{A_i}$, $S_{N_i}$, $b_{A_i}$, and $b_{N_i}$.

For simplicity, assume that the SC property which is introduced in Definition 2.12 holds. This implies that no constraints in the optimal active set are weakly active. How to compute the solution when SC does not hold is presented in for example Tøndel et al. (2003c), but for the purpose of this thesis it is sufficient to present the case with no weakly active constraints. Furthermore, let $G_{A_i}$ have full row rank, meaning that LICQ holds for $A_i$ (Tøndel et al., 2003c). According to Definition 2.13, violation of LICQ is referred to as primal degeneracy and how to handle this can be seen in for example Bemporad et al. (2002). If LICQ is violated but the matrix $[G_{A_i} - b_{A_i} - S_{A_i}]$ has full row rank, it results in a non-full dimensional critical region that in general is a facet between full dimensional regions (Bemporad et al., 2002; Tøndel et al., 2003c). How to handle the case when the matrix does not have full row rank is presented in for example Tøndel et al. (2003a). When LICQ and SC hold, an optimal active set $A_i$ uniquely corresponds to a critical region $P_i$ (Tøndel et al., 2003c; Gupta et al., 2011).

For any parameter $\theta \in P_i$, where $P_i$ corresponds to $A_i$, the solution to the mp-QP problem (4.3) can be computed by parametrically solving an equality constrained mp-QP problem where the constraints in the active set $A_i$ are forced to hold with equality, and the rest of the inequality constraints are disregarded. This equality
constrained mp-QP problem is given by
\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} z^T H z \\
\text{subject to} \quad & G_A z = b_A + S_A \theta.
\end{align*}
\] (4.6)
The parametric solution to (4.6) is an affine function in the parameter \( \theta \) (Bemporad et al., 2002), which for a parameter \( \theta \in \Theta \) is given by
\[
\begin{align*}
\lambda^{\star}(\theta) &= \sigma_i + \Gamma_i \theta, \quad (4.7a) \\
z^{\star}(\theta) &= k_i + K_i \theta \triangleq -H^{-1}G_{A_i}^T (\sigma_i + \Gamma_i \theta), \quad (4.7b)
\end{align*}
\] where \( \sigma_i \in \mathbb{R}^{|A_i|} \) and \( \Gamma_i \in \mathbb{R}^{|A_i| \times n} \) are defined as
\[
\begin{align*}
\sigma_i &\triangleq - (G_{A_i} H^{-1} G_{A_i}^T)^{-1} b_A, \\
\Gamma_i &\triangleq - (G_{A_i} H^{-1} G_{A_i}^T)^{-1} S_A.
\end{align*}
\] (4.8)
See for example Bemporad et al. (2002), Tøndel et al. (2003c), Spjøtvold et al. (2006), and Nocedal and Wright (2006) for further details on how to solve equality constrained problems in the form (4.6).

### 4.3 Characterizing the Critical Region

The critical region \( \mathcal{P}_i \) is the set of parameters for which the active set \( A_i \) is optimal, that is, all parameters \( \theta \in \Theta \) such that the primal and dual feasibility constraints given by (4.4b) and (4.4c), respectively, are satisfied. By inserting the parametric solution (4.7) in (4.4b) and (4.4c), it follows that the \( \mathcal{P}_i \) is defined as the set
\[
\mathcal{P}_i \triangleq \left\{ \theta \in \Theta \mid G_{N_i} (k_i + K_i \theta) \preceq b_{N_i} + S_{N_i} \theta, \quad \sigma_i + \Gamma_i \theta \succeq 0 \right\}. \quad (4.9)
\]
The primal and dual feasibility conditions are given by the first and second inequalities in (4.9), respectively. \( \mathcal{P}_i \) is a polyhedron with \( p \) hyperplanes (Bemporad et al., 2002). All hyperplanes in (4.9) which are redundant can be removed to obtain a description of \( \mathcal{P}_i \) consisting of the minimal number of describing hyperplanes (Bemporad et al., 2002; Tøndel et al., 2003c). Let \( \mathcal{H}_i^p \subset N_i \) and \( \mathcal{H}_i^d \subset A_i \) be the indices of the describing hyperplanes that are related to the primal (superscript “p”) and dual (superscript “d”) feasibility conditions in (4.9), respectively. Then the critical region \( \mathcal{P}_i \) in (4.9) is equivalent to
\[
\mathcal{P}_i = \left\{ \theta \in \Theta \mid G_{\mathcal{H}_i^p} (k_i + K_i \theta) \preceq b_{\mathcal{H}_i^p} + S_{\mathcal{H}_i^p} \theta, \quad \sigma_i,_{\mathcal{H}_i^d} + \Gamma_i,_{\mathcal{H}_i^d} \theta \succeq 0 \right\}, \quad (4.10)
\] which describes the same set as \( \mathcal{P}_i \) in (4.9), but using only \( |\mathcal{H}_i^p| + |\mathcal{H}_i^d| \leq p \) hyperplanes. Here, the use of a double index as in for example \( \Gamma_i,_{\mathcal{H}_i^d} \) is used to denote the rows in \( \Gamma_i \) indexed by \( \mathcal{H}_i^d \).

### 4.4 Explicit Model Predictive Control

In control, mp-QP problems have become useful since it was shown in Bemporad et al. (2002) that the CFTOC problem used in linear MPC for an LTI system can be
cast as an mp-QP problem with the initial state as parameter. Solving this mp-QP problem prior to the on-line execution is referred to as explicit MPC, and the solution is explicitly given as a function of the initial state. The on-line computational effort consists of evaluating the pwa function for a given parameter (Bemporad et al., 2002), which allows for a division free implementation of the control law that can be computed within an a priori known worst case execution time (Kvasnica et al., 2015). However, the time that is consumed to evaluate the pwa function is one of the main limitations with mp-QP and explicit MPC (Bemporad et al., 2002; Tøndel et al., 2003b). Hence, much focus has been spent on developing algorithms for efficient evaluation of the PWA function. In Tøndel et al. (2003b), the authors propose an algorithm based on a binary search tree, which provides evaluation times that are logarithmic in the number of regions. In Herceg et al. (2013b) a graph traversal algorithm is used to evaluate pwa functions and the graph is constructed while solving the mp-QP problem. In Fuchs et al. (2010) the point location problem is solved using linear decision functions, and significantly better performance in terms of computational time at a small cost of increased memory has been reported.

In linear MPC for LTI systems, the input is computed by solving a CFTOC problem where all matrices are time invariant. The problem considered in Bemporad et al. (2002) and many other works on explicit MPC is a CFTOC problem in the form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \sum_{t=0}^{N-1} \left( x_t^T \tilde{Q} x_t + u_t^T \tilde{Q} u_t \right) + \frac{1}{2} x_N^T \tilde{P} x_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = \tilde{A} x_t + \tilde{B} u_t, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad \tilde{H}_x x_t + \tilde{H}_u u_t + \tilde{h} \preceq 0, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad \tilde{H}_x x_N + \tilde{h} \preceq 0,
\end{align*}
\]

(4.11)

where assumptions 3.1 and 3.2 hold. Similarly to what is shown in Bemporad et al. (2002), the CFTOC problem (4.11) can equivalently be written in the form of an mp-QP problem in the form in (4.1) by defining the matrices

\[
\begin{align*}
H & \triangleq \tilde{Q}_u + \tilde{B}^T \tilde{Q}_x \tilde{B} \in \mathbb{S}_{++}^{N_{u}}; \quad F \triangleq \tilde{A}^T \tilde{Q}_x \tilde{B}, \\
G & \triangleq \tilde{H}_x \tilde{B} + \tilde{H}_u, \quad b \triangleq -\tilde{h}, \quad E \triangleq -\tilde{H}_x \tilde{A}, \quad \theta \triangleq \bar{x},
\end{align*}
\]

(4.12a)

where $\tilde{Q}_x$, $\tilde{Q}_u$, $\tilde{A}$, $\tilde{B}$, $\tilde{H}_x$, $\tilde{H}_u$, and $\bar{h}$ are defined as in (4A.13) in Appendix 4.A. Note the similarities between the mp-QP problem and the condensed CFTOC formulation (3.7). The conceptual difference between them is that in the mp-QP problem the initial state is considered as a parameter and it is solved for all possible values of the initial state, whereas the condensed CFTOC problem in (3.7) is solved for a given value of the initial state. By re-writing the mp-QP problem (4.1) into (4.3) and solving it parametrically, the optimal solution to the CFTOC problem is given by $u^*(\theta) = z^*(\theta) - H^{-1}F^T \theta$ where $z^*(\theta)$ is given by the PWA function (4.5) and the change of variables is defined as in (4.2) (Bemporad et al., 2002). Only the first control input $u^*_0(\theta)$ is applied in the MPC control loop.
Appendix

4.A Explicit Model Predictive Control Formulations

The matrices in the compact formulation of the cftoc problem (4.11) are defined by

\[
\tilde{A} \triangleq \begin{bmatrix} I \\ \tilde{A} \\ \vdots \\ \tilde{A}^N \end{bmatrix}, \quad \tilde{B} \triangleq \begin{bmatrix} 0 \\ \tilde{B} \\ \tilde{A} \tilde{B} \\ \vdots \\ \tilde{A}^{N-1} \tilde{B} \quad \tilde{A}^{N-2} \tilde{B} \quad \vdots \quad \tilde{B} \end{bmatrix}, \quad (4A.13a)
\]

\[
\tilde{Q}_x \triangleq \begin{bmatrix} \tilde{Q}_x \\ \vdots \\ \tilde{Q}_x \end{bmatrix}, \quad \tilde{Q}_u \triangleq \begin{bmatrix} \tilde{Q}_u \\ \vdots \\ \tilde{Q}_u \end{bmatrix}, \quad (4A.13b)
\]

\[
\tilde{h}_x \triangleq \begin{bmatrix} \tilde{H}_x \\ \vdots \\ \tilde{H}_x \end{bmatrix}, \quad \tilde{h}_u \triangleq \begin{bmatrix} \tilde{H}_u \\ \vdots \\ \tilde{H}_u \end{bmatrix}, \quad \tilde{h} \triangleq \begin{bmatrix} \tilde{h} \\ \vdots \\ \tilde{h} \end{bmatrix}. \quad (4A.13c)
\]
Low-Rank Modifications of the Riccati Factorization

When computing the search directions in AS methods it is often crucial to modify the factorization of the KKT matrix instead of re-factorizing it between AS iterations to obtain high performance (Nocedal and Wright, 2006). Since this has traditionally not been considered possible when using the Riccati factorization to factor the KKT matrix, it has sometimes been argued that this factorization is not suitable for AS methods. However, in Nielsen et al. (2013a) a method for performing low-rank modifications of the Riccati factorization by exploiting the structured changes between AS iterations when solving CFTOC problems in the form (3.22) was introduced, showing that this is indeed possible. In this chapter, the results from Nielsen et al. (2013a) and the extension of this result in Nielsen (2015) and Nielsen and Axehill (2017) are presented.

In order to develop a general framework that can be used in different types of AS solvers, the modifications of the Riccati factorization are derived for UFTOC problems where the KKT matrix can be singular, and for general modifications of the working set where several constraints are simultaneously added (or removed) at different time indices between AS iterations. Both these features can be important when using for example dual projection AS solvers like the one in Axehill (2008). Furthermore, it is shown how CFTOC problems with general inequality constraints on the states and control inputs can be handled while keeping the ability to modify the Riccati factorization between AS iterations.

For now, however, consider a CFTOC problem in the form (3.22) where Assumption 3.2 does not hold, i.e., a problem where the inequality constraints are given by upper and lower bounds on the control inputs and that might have a singular quadratic weight on the control inputs in the objective function. Note that this is a more general problem description than the one considered in Section 3.2.2,
and problems of this more general type can be useful in for example dual projection as solvers like the one in Axehill (2008). As was shown in Section 3.2.2 the computation of the search directions in an as method applied to this cftoc problem corresponds to solving uftoc problems in the form (3.29). This chapter will focus on how to solve this uftoc problem and how to exploit the structure in the problem after changing the working set in an as solver. The partitioning of the control input $u_t$ in (3.22) is done as in (3.23). The uftoc problem (3.29) is repeated here for convenience

$$\begin{align*}
\text{minimize} & \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} x_t^T Q_{x,t} x_t + \frac{1}{2} w_t^T Q_{w,t} w_t + l_{x,t}^T x_t + l_{w,t}^T w_t + c_t \right) + \\
& \quad \frac{1}{2} x_N^T Q_{x,N} x_N + l_{x,N}^T x_N + c_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = A_t x_t + B_{w,t} w_t + a_{v,t}, \quad t \in \mathbb{Z}_{0,N-1},
\end{align*}$$

(5.1)

where the problem matrices, vectors and constants are defined as in Section 3.2.2. Note that $Q_{w,t} \in \mathbb{S}_{++}^n$ since Assumption 3.2 is neglected.

### 5.1 Derivation of the Low-Rank Modifications

Here it will be shown how to modify the Riccati factorization in Algorithm 1 between as iterations when solving a cftoc problem as in (3.22). Since $Q_{w,t} \in \mathbb{S}_{++}^n$, the KKT matrix for the uftoc problem that is solved to compute the search direction might be singular. This results in that some $G_{t+1}$ in Algorithm 4 might be singular. To be able to handle cases where the KKT matrix is singular, the modifications of the Riccati factorization will be derived using mathematical tools such as generalized Schur complements (gscs), the quotient formula for gscs and the Moore-Penrose pseudo-inverse. For a detailed description of these, see for example Albert (1969), Carlson et al. (1974), and Zhang (2005).

Between as iterations, the working set can be modified in different ways depending on the solver. The most common types of modifications are either adding one or several constraints to the working set, or removing one or several constraints from the working set. The traditional as method in Algorithm 1 only modifies the working set with one constraint at each iteration. However, as type methods such as the projection method in Axehill (2008) modify the working set with several constraints in each iteration. This often results in that fewer search directions need to be computed in total to solve the CFTOC problem, which in turn increases the performance of the solver in terms of computation time. Here, it will be described how to handle both these types of modifications.

To derive the algorithm for how to modify the Riccati factorization, the details from Section 3.4 will be exploited. In Section 3.4.2 it was shown that when the
KKT matrix is singular, \( P_t \) can be computed as

\[
P_t = F_{t+1} - K_{t+1}^T G_{t+1} K_{t+1} = M_{t+1}^{-1} G_{t+1}.
\]

The gsc operator /\(^\dagger\) is introduced in Definition A.5 in Appendix A, and \( M_{t+1} \) is defined in (3.51), i.e.,

\[
M_{t+1} = \begin{bmatrix} F_{t+1} & H_{t+1} \\ H_{t+1}^T & G_{t+1} \end{bmatrix} \in S_{+}^{n_{x+n_{w,t}}},
\]

(5.3)

Note that when \( G_{t+1} \in S_{++}^{n_{w,t}} \), the gsc is equivalent to the Schur complement in Definition A.6 in Appendix A. Hence, as is shown in Section 3.4.2, in that case \( P_t \) can be computed as

\[
P_t = M_{t+1}^{-1} G_{t+1} = F_{t+1} - H_{t+1} G_{t+1}^{-1} H_{t+1}^T.
\]

(5.4)

Furthermore, the quotient formula for gscs will be important in the derivation of the algorithm for performing the low-rank modifications of the Riccati factorization, and is given in Lemma 5.1.

**Lemma 5.1 (Quotient formula for generalized Schur complements).** Let the positive semi-definite matrices \( \mathcal{M} \geq 0 \) and \( \mathcal{M} \geq 0 \) be partitioned as

\[
\mathcal{M} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{12}^T & M_{22} & M_{23} \\ M_{13}^T & M_{23}^T & M_{33} \end{bmatrix}, \quad \mathcal{M} = \begin{bmatrix} M_{22} & M_{23} \\ M_{23}^T & M_{33} \end{bmatrix}.
\]

(5.5)

Then

\[
\left( M /\(^\dagger\) M_{33} \right) /\(^\dagger\) \left( \mathcal{M} /\(^\dagger\) M_{33} \right) = M /\(^\dagger\) \mathcal{M} = M_{11} - M_{12} M_{22}^{-1} M_{12}^T - \left( M_{13} - M_{12} M_{22}^{-1} M_{12}^T \right) \left( M_{33} - M_{23} M_{22}^{-1} M_{23}^T \right) /\(^\dagger\),
\]

(5.6)

with

\[
\mathcal{R} \left( M_{13}^T - M_{23}^T M_{22}^{-1} M_{12}^T \right) \subset \mathcal{R} \left( M_{33} - M_{23} M_{22}^{-1} M_{23} \right),
\]

(5.7a)

\[
M_{33} - M_{23} M_{22}^{-1} M_{23} \geq 0.
\]

(5.7b)

**Proof:** The proof of Lemma 5.1 is given in Appendix 5.A.

In the remainder of this chapter, a tilde will be used to indicate a matrix that has been modified. Hence, the modified version of some matrix \( X \) is denoted \( \tilde{X} \).

### 5.1.1 Sequence of low-rank modifications

In this section it will be shown how a modification of \( P_{t+1} \) for some \( t \) is propagated backwards in the Riccati factorization in Algorithm 4.
Modifications for the case of a downdate of $P_{t+1}$

Assume that the modified $P_{t+1}$ for some $t \in \mathbb{Z}_{0,N-1}$, denoted by $\hat{P}_{t+1}$, is a downdate of $P_{t+1}$ of size $\hat{k}$, given by

$$\hat{P}_{t+1} = P_{t+1} - V_{t+1}^- C_{t+1}^- \dagger V_{t+1}^- \quad \in \mathcal{S}_+^{n_x},$$  \hspace{1cm} (5.8)

with

$$C_{t+1}^- \in \mathcal{S}_+^{\hat{k}} \quad \text{and} \quad V_{t+1}^- \in \mathbb{R}^{n_x \times \hat{k}}, \quad \mathcal{R} \left( V_{t+1}^- \right) \subset \mathcal{R} \left( C_{t+1}^- \right).$$  \hspace{1cm} (5.9)

Here, a superscript "\dagger" is used to indicate a matrix that is related to a downdate. It will later be shown using Lemma 5.1 that the assumption (5.9) for the downdate is made without any loss of generality for the type of modifications considered in this chapter. Furthermore, the downdate in (5.8) is considered to be of low rank if $\hat{k} < n_x$.

It will now be shown how this downdate of $P_{t+1}$ affects the matrices in the Riccati factorization for the preceding time-steps $\tau \in \mathbb{Z}_{0,t}$. By replacing $P_{t+1}$ at lines 3-5 in Algorithm 4 with $\hat{P}_{t+1}$ from (5.8), straightforward calculations give

$$\hat{F}_{t+1} = Q_{x,t} + A_t^T \hat{P}_{t+1} A_t = Q_{x,t} + A_t^T P_{t+1} A_t - A_t^T V_{t+1}^- C_{t+1}^- \dagger V_{t+1}^- A_t,$$

where $F_{t+1}$ is given by (5.3),  \hspace{1cm} (5.10a)

$$\hat{G}_{t+1} = G_{t+1} - B_{w,t} V_{t+1}^- \dagger V_{t+1}^- B_{w,t},$$

$$\hat{H}_{t+1} = H_{t+1} - A_t^T V_{t+1}^- C_{t+1}^- \dagger V_{t+1}^- B_{w,t}.$$  \hspace{1cm} (5.10b, 5.10c)

The equations in (5.10) can be written in matrix form as

$$\hat{M}_{t+1} = \begin{bmatrix} \hat{F}_{t+1} & \hat{H}_{t+1} \\ \hat{H}_{t+1}^T & \hat{G}_{t+1} \end{bmatrix} = \begin{bmatrix} F_{t+1} & H_{t+1} \\ H_{t+1}^T & G_{t+1} \end{bmatrix} - \begin{bmatrix} A_t^T V_{t+1}^- \\ B_{w,t}^T V_{t+1}^- \end{bmatrix} C_{t+1}^- \dagger \begin{bmatrix} V_{t+1}^- A_t \\ V_{t+1}^- B_{w,t} \end{bmatrix} \in \mathcal{S}_{++}^{n_x + n_{w,t}}.$$  \hspace{1cm} (5.11)

Since $\hat{P}_{t+1} \in \mathcal{S}_+^{n_x}$ and $\hat{M}_{t+1}$ can alternatively be defined as in (5.3), the positive semidefiniteness of $\hat{M}_{t+1}$ follows by construction. Define the auxiliary matrix

$$\hat{M}_{t+1} \triangleq \begin{bmatrix} F_{t+1} & H_{t+1} & A_t^T V_{t+1}^- \\ H_{t+1}^T & G_{t+1} & B_{w,t}^T V_{t+1}^- \end{bmatrix},$$  \hspace{1cm} (5.12)

and let $\hat{M}_{t+1}$ be the second diagonal block of $\hat{M}_{t+1}$. Note that

$$\hat{G}_{t+1} = \hat{M}_{t+1} \dagger C_{t+1}^-.$$  \hspace{1cm} (5.13)

From Theorem A.10 and Lemma A.4 it follows that $\hat{M}_{t+1} \succeq 0$ since

$$C_{t+1}^- \succeq 0, \quad \mathcal{R} \left( V_{t+1}^- \right) \subset \mathcal{R} \left( C_{t+1}^- \right).$$  \hspace{1cm} (5.14)
by assumption, and that $\tilde{M}_{t+1}$ in (5.11) can be written as
\[ \tilde{M}_{t+1} = \tilde{M}_{t+1}^\dagger C_{t+1}^- \succeq 0. \] (5.15)
Hence also $\tilde{M}_{t+1} \succeq 0$ holds, and hence it follows that the quotient formula for
GSCs in Lemma 5.1 can be used to compute
\[ \tilde{P}_t = \tilde{M}_{t+1}^\dagger \tilde{G}_{t+1} = \tilde{M}_{t+1}^\dagger \tilde{M}_{t+1}. \] (5.16)
Here the first equality in (5.6) has been used. Furthermore, partition the matrix
\[ \hat{M}_{t+1} \text{ in (5.12)} \]
into the following blocks
\[ M_{11} = F_{t+1}, \quad M_{12} = H_{t+1}, \quad M_{13} = A_t^T V_{t+1}^-; \]
\[ M_{22} = G_{t+1}, \quad M_{23} = B_{w,t}^T V_{t+1}^-; \quad M_{33} = C_{t+1}^-, \]
\[ M_{21} = M_{12}^T, \quad M_{31} = M_{13}^T, \quad M_{32} = M_{23}^T. \] (5.17c)
Then, the modified version of $P_t$ is computed using the second equality in (5.6) in
Lemma 5.1 as
\[ \tilde{P}_t = P_t - V_t^- C_t^- V_t^- T \in S_{+}^{n_x}, \] (5.18)
where
\[ V_t^- \triangleq \left( A_t^T - H_{t+1} G_{t+1}^T B_{w,t}^T \right) V_{t+1}^- \in \mathbb{R}^{n_x \times \tilde{k}}, \] (5.19a)
\[ C_t^- \triangleq C_{t+1}^- - V_{t+1}^- T B_{w,t} G_{t+1}^T B_{w,t} V_{t+1}^- \in S_{+}^{\tilde{k}}, \] (5.19b)
\[ \mathcal{R} \left( V_t^- T \right) \subset \mathcal{R} \left( C_t^- \right). \] (5.19c)
Note that the modification in (5.19) at time $t$ satisfies the assumption (5.9) that
was made for the downdate at time $t+1$.

**Modifications for the case of an update of $P_{t+1}$**

Now assume instead that the modified $P_{t+1}$, denoted as $\tilde{P}_{t+1}$, is an update of $P_{t+1}$
of size $\tilde{k}$ in the form
\[ \tilde{P}_{t+1} = P_{t+1} + V_{t+1}^+ C_{t+1}^+ V_{t+1}^T \in S_{+}^{n_x}, \] (5.20)
with
\[ C_{t+1}^+ \in S_{+}^{\tilde{k}}, \quad V_{t+1}^+ \in \mathbb{R}^{n_x \times \tilde{k}}, \quad \mathcal{R} \left( V_{t+1}^+ T \right) \subset \mathcal{R} \left( C_{t+1}^+ \right). \] (5.21)
Here a superscript “+” is used to denote a matrix that is related to an update.
Similarly as for a downdate, it will be shown that the assumption in (5.21) are
without loss of generality. By re-arranging the terms, (5.20) can be equivalently
written as
\[ P_{t+1} = \tilde{P}_{t+1} - V_{t+1}^+ C_{t+1}^+ V_{t+1}^T \in S_{+}^{n_x}. \] (5.22)
Then, using similar calculations as for a downdate, it can be shown that an update
of $P_{t+1}$ results in an update of $P_t$, given by
\[ P_t = \tilde{P}_t - V_t^+ C_t^+ V_t^T \in S_{+}^{n_x} \iff \tilde{P}_t = P_t + V_t^+ C_t^+ V_t^T \in S_{+}^{n_x}, \] (5.23)
with
\[ V^+_t \triangleq \left( A^T_t - \tilde{H}_{t+1} \tilde{G}_{t+1}^T B_{w,t}^T \right) V^+_t + 1 \in \mathbb{R}^{n_x \times \tilde{k}}, \]  
(5.24a)
\[ C^+_t \triangleq C^+_{t+1} - V^+_t B_{w,t} G_{t+1}^T B_{w,t}^T V^+_t + 1 \in \mathbb{S}^+, \]  
(5.24b)
\[ \mathcal{R} \left( V^+_t T \right) \subset \mathcal{R} \left( C^+_{t+1} \right). \]  
(5.24c)

Here \( \tilde{G}_{t+1} \) and \( \tilde{H}_{t+1} \) are defined analogously as in (5.10), giving
\[ \tilde{F}_{t+1} = F_{t+1} + A^T_t V^+_t C^+_{t+1}^T V^+_t T A_t, \]  
(5.25a)
\[ \tilde{G}_{t+1} = G_{t+1} + B_{w,t}^T V^+_t C^+_{t+1}^T V^+_t T B_{w,t}, \]  
(5.25b)
\[ \tilde{H}_{t+1} = H_{t+1} + A^T_t V^+_t C^+_{t+1}^T V^+_t T B_{w,t}. \]  
(5.25c)

Furthermore, note that the modified matrices \( \tilde{H}_{t+1} \) and \( \tilde{G}_{t+1} \) are used in (5.24), and that the modification at time \( t \) (5.24) satisfies the assumption (5.21) that was made for the modification at time \( t + 1 \).

**Propagation of low-rank modifications**

From the derivations of how a downdate or an update of \( P_{t+1} \) affect the matrices in the Riccati factorization, it follows that a modification of \( P_{t+1} \) of size \( \tilde{k} \) results in a modification of the same type of \( P_t \) of (also) size \( \tilde{k} \), which is stated in the following theorem:

**Theorem 5.2.** Consider a modification of size \( \tilde{k} \) of \( P_{t_m} \in \mathbb{S}^{n_x}_+ \) at a single time instant \( t_m \in \mathbb{Z}_{1,N} \) in either of the forms
\[ \begin{cases} \tilde{P}_{t_m} = P_{t_m} - V_{t_m}^- C_{t_m}^{-1} V_{t_m}^- T \in \mathbb{S}^{n_x}_+ \text{ (downdate)} \\ \tilde{P}_{t_m} = P_{t_m} + V_{t_m}^+ C_{t_m}^+ V_{t_m}^+ T \in \mathbb{S}^{n_x}_+ \text{ (update)} \end{cases} \]  
(5.26)

where \( C_{t_m}^-, C_{t_m}^+ \in \mathbb{S}^{\tilde{k}}_+, V_{t_m}^-, V_{t_m}^+ \in \mathbb{R}^{n_x \times \tilde{k}} \), and \( \mathcal{R} \left( V_{t_m}^- T \right) \subset \mathcal{R} \left( C_{t_m}^- \right), \mathcal{R} \left( V_{t_m}^+ T \right) \subset \mathcal{R} \left( C_{t_m}^+ \right) \), respectively.

Then it holds for all \( t \in \mathbb{Z}_{0,t_m-1} \) that \( P_t \in \mathbb{S}^{n_x}_+ \) is modified as
\[ \begin{cases} \tilde{P}_t = P_t - V_t^- C_t^{-1} V_t^- T \in \mathbb{S}^{n_x}_+ \text{ (downdate)} \\ \tilde{P}_t = P_t + V_t^+ C_t^+ V_t^+ T \in \mathbb{S}^{n_x}_+ \text{ (update)} \end{cases} \]  
(5.27)

with \( C_t^-, C_t^+ \in \mathbb{S}^{\tilde{k}}_+, V_t^-, V_t^+ \in \mathbb{R}^{n_x \times \tilde{k}} \), and \( \mathcal{R} \left( V_t^- T \right) \subset \mathcal{R} \left( C_t^- \right), \mathcal{R} \left( V_t^+ T \right) \subset \mathcal{R} \left( C_t^+ \right) \), respectively.

**Proof:** From the derivations of (5.18) and (5.23) it follows that a modification of size \( \tilde{k} \) of \( P_{t+1} \) at an arbitrary \( t \in \mathbb{Z}_{0,N-1} \) in either of the forms (5.8) or (5.20) results in a similar modification of size \( \tilde{k} \) of \( P_t \). Since \( P_{t_m} \) is modified as in (5.26), the proof follows by induction. \( \square \)
To compute the modified version of $K_{t+1}$ defined in (3.43a), the linear system of equations

$$
\tilde{G}_{t+1} \tilde{K}_{t+1} = -\tilde{H}_{t+1}^T, \quad (5.28)
$$
can be solved. Since $\tilde{M}_{t+1} \succeq 0$ it follows from Theorem A.10 that $\tilde{G}_{t+1} \tilde{G}_{t+1}^T \tilde{H}_{t+1}^T = \tilde{H}_{t+1}^T$ holds. Hence, from Lemma A.4 it follows that the system of equations is solvable even when $\tilde{G}_{t+1} \in S_{+}^{n_{w,t}}$ is singular.

For the common case when $\tilde{G}_{t+1} \in S_{++}^{n_{w,t}}$, it is possible to use the Sherman-Morrison-Woodbury formula for efficient computations. For the details the reader is referred to for example Golub and Van Loan (1996).

**Remark 5.3.** The factorization of $\tilde{G}_{t+1}$ can be computed by modifying the factorization of $G_{t+1}$ whenever it is possible. Hence, in these cases a solution to (5.28) can be computed without having to re-factorize $\tilde{G}_{t+1}$, which requires less computations than re-solving (5.28) from scratch. See for example Golub and Van Loan (1996) and Stewart (1998) for examples of different factorization and modification algorithms.

### 5.1.2 Removing control input constraints from the working set

In this section it will be shown how to modify the Riccati factorization when removing constraints from the working set at time index $t$, when a modification of $P_{t+1}$ as in (5.8) of size $\tilde{k}$ is potentially already present. Let $k$ control input constraints that are affecting the control input at time $t$ be removed from the working set at AS iteration $j$. This means that these $k$ inequality constraints that previously were forced to hold are temporarily disregarded, which affects the UFTOC problem (5.1) in the same way as adding $k$ new control inputs. Assume without loss of generality that the new control inputs are appended at the end of $w_t$. Then the matrices $B_{w,t}$, $Q_{w,t}$ and $Q_{xw,t}$ in (5.1) are modified as

$$
\tilde{B}_{w,t} \triangleq \begin{bmatrix} B_{w,t} & b \end{bmatrix}, \quad \tilde{Q}_{w,t} \triangleq \begin{bmatrix} Q_{w,t} & q_w \\ q_w^T & q_w^T \end{bmatrix}, \quad \tilde{Q}_{xw,t} \triangleq \begin{bmatrix} Q_{xw,t} & q_{xw} \end{bmatrix}, \quad (5.29)
$$
giving $\tilde{B}_{w,t} \in \mathbb{R}^{n_x \times (n_{w,t} + k)}$, $\tilde{Q}_{w,t} \in S_{+}^{n_{w,t} + k}$ and $\tilde{Q}_{xw,t} \in \mathbb{R}^{n_x \times (n_{w,t} + k)}$. From (3.41) it then follows that $\tilde{G}_{t+1} \in S_{+}^{n_{w,t} + k}$ and $\tilde{H}_{t+1} \in \mathbb{R}^{n_x \times (n_{w,t} + k)}$ are given by

$$
\tilde{G}_{t+1} = \begin{bmatrix} G_{t+1} & g \\ g^T & g^0 \end{bmatrix} - \begin{bmatrix} B_{w,t}^T \\ b^T \end{bmatrix} V_{t+1}^ - C_{t+1}^ - V_{t+1}^ T \begin{bmatrix} B_{w,t} & b \end{bmatrix}, \quad (5.30a)
$$

$$
\tilde{H}_{t+1} = \begin{bmatrix} H_{t+1} \\ h \end{bmatrix} - A_t^T V_{t+1}^ - C_{t+1}^ - V_{t+1}^ T \begin{bmatrix} B_{w,t} & b \end{bmatrix}, \quad (5.30b)
$$

where

$$
\begin{bmatrix} h \\ g \\ g^0 \end{bmatrix} \triangleq \begin{bmatrix} q_{xw} \\ q_w \\ q_w^0 \end{bmatrix} + \begin{bmatrix} A_t^T \\ B_{w,t}^T \end{bmatrix} P_{t+1} b. \quad (5.31)
$$

In analogy with Section 5.1.1, $\tilde{P}_t$ can be computed as $\tilde{P}_t = \tilde{M}_{t+1} / \tilde{G}_{t+1}$, where $\tilde{M}_{t+1}$ is computed as in (5.11) but instead using $\tilde{G}_{t+1}$ and $\tilde{H}_{t+1}$ from (5.30). Define
the auxiliary matrix $\hat{M}_{t+1}$

$$
\hat{M}_{t+1} \triangleq \begin{bmatrix}
F_{t+1} & H_{t+1} & h & A_t^T V_{t+1}^{-1} \\
H_{t+1}^T & G_{t+1} & g & B_{w,t}^T V_{t+1}^{-1} \\
V_{t+1}^{-1} A_t & V_{t+1}^{-1} B_{w,t} & V_{t+1}^{-1} b & C_{t+1}
\end{bmatrix},
$$

(5.32)

and let $\hat{M}_{t+1}$ be the second diagonal block. The matrix in (5.32) is defined similarly as in (5.12), but where the effect of the added control inputs are included as well. By using this auxiliary matrix and the definition of the gsc, it is follows that

$$
\hat{M}_{t+1} = \hat{M}_{t+1}/C_{t+1}^-, 
$$

(5.33)

and

$$
\hat{G}_{t+1} = \hat{M}_{t+1}/C_{t+1}^-.
$$

(5.34)

Hence, the quotient formula for gscs in Lemma 5.1 can be used as in Section 5.1.1 to compute $\hat{P}_t$ as

$$
\hat{P}_t = \hat{M}_{t+1}/C_{t+1}^-.
$$

(5.35)

Furthermore, by partitioning $\hat{M}_{t+1}$ as in Lemma 5.1 with

$$
M_{11} = F_{t+1}, \quad M_{12} = H_{t+1}, \quad M_{13} = [h \quad A_t^T V_{t+1}^--],
$$

(5.36a)

$$
M_{22} = G_{t+1}, \quad M_{23} = [g \quad B_{w,t}^T V_{t+1}^-], \quad M_{33} = \begin{bmatrix}
g^0 & b^T V_{t+1}^- \\
V_{t+1}^- T b & C_{t+1}^-
\end{bmatrix},
$$

(5.36b)

$$
M_{21} = M_{12}^T, \quad M_{31} = M_{13}^T, \quad M_{32} = M_{23}^T.
$$

(5.36c)

Using the second equality in Lemma 5.1, it follows that $\hat{P}_t$ can be computed as

$$
\hat{P}_t = F_{t+1} - H_{t+1} G_{t+1}^T V_{t+1}^- C_{t+1}^- V_{t+1}^- T_{\hat{P}_t} \in S_{++}^{n_x},
$$

(5.37)

where

$$
V_{t}^- \triangleq \begin{bmatrix}
h - H_{t+1} G_{t+1}^T g & (A_t^T - H_{t+1} G_{t+1}^T B_{w,t}^T) V_{t+1}^-
\end{bmatrix},
$$

(5.38a)

$$
C_{t}^- \triangleq \begin{bmatrix}
g^0 & b^T V_{t+1}^- \\
V_{t+1}^- T b & C_{t+1}^-\end{bmatrix} - \begin{bmatrix}
g^T \\
V_{t+1}^- T B_{w,t}^T
\end{bmatrix} G_{t+1}^T [g \quad B_{w,t}^T V_{t+1}^-],
$$

(5.38b)

$$
C_{t}^- \in S_{++}^{k+k}, \quad R(V_{t}^- T) \subset R(C_{t}^-).
$$

(5.38c)

Note that the downdate (5.38) at time $t$ satisfies the assumption (5.9) that was made for the downdate at time $t+1$.

Hence, removing $k$ control input constraints at time $t$ from the working set when a downdate in the form (5.8) of $P_{t+1}$ of size $\hat{k}$ is already present results in a downdate of $P_t$ in the same form as (5.8) but of increased size $\hat{k} + k$. The modified version $\hat{K}_{t+1} \in R^{(n_{w,t}+k) \times n_x}$ can be computed by solving (5.28) but using $\hat{G}_{t+1}$ and $\hat{H}_{t+1}$ from (5.30) instead of (5.10).
When removing $k$ constraints from the working set, also $k$ components in $v_t$ in (3.23) are removed and are instead included in $\tilde{w}_t$. Hence, the corresponding matrices and vectors $B_{w,t}, a_{w,t}, Q_{xw,t}, Q_{wv,t}$, $l_{w,t}$ and $l_{v,t}$ in (3.24a), (3.24b) and (3.25) are affected accordingly (i.e. the corresponding $k$ columns $b$ in $B_{w,t}$ are instead included in $\tilde{B}_{w,t}$ etc.). These changes are only affecting the right-hand-side of the system of equations in the KKT system in (3.35), and hence they are not affecting the matrices in the factorization computed in Algorithm 4.

**Remark 5.4.** Note that if $k + k$ is close to, or larger than, $n_x$ it might be better to re-compute the Riccati factorization for the remaining time-steps. This trade-off can be investigated off-line by modifying the factorization for different sizes of modifications and determine the fastest alternative, but the details are left as future work.

**Remark 5.5.** If there is no modification of $P_{t+1}$, then $C_t^- = g^0 - g^T G_{t+1}^T g \in S^+_k$ and $V_t^- = h - H_{t+1} G_{t+1}^T g \in \mathbb{R}^{n_x \times k}$, with $\mathcal{R} \left( V_t^{-T} \right) \subset \mathcal{R} \left( C_t^- \right)$.

**Remark 5.6.** For the common case when $\tilde{G}_{t+1} \in S_{++}^{n_{w,t+k}}$, low-rank modifications can be exploited by using the Sherman-Morrison-Woodbury formula for efficient computations of $K_{t+1}$. Furthermore, the factorization of $\tilde{G}_{t+1}$ should be modified instead of re-computed as is mentioned in Remark 5.3.

### 5.1.3 Adding control input constraints to the working set

When control input constraints are added to the working set, the modifications of the Riccati factorization can be derived in a similar way as when constraints are removed. To do this, assume that $P_{t+1}$ is potentially already updated as in (5.20), and that $k$ control input constraints that are affecting the control input at time $t$ are added to the working set at AS iteration $j$. Adding constraints to the working set corresponds to removing these control inputs from the problem and treating them as constants. Assume, without loss of generality, that the $k$ control inputs are removed from the $k$ last entries of $w_t$. Then $\tilde{B}_{w,t}$ is obtained as the first $n_{w,t} - k$ columns of $B_t$, etc. Hence, the modified matrices $\tilde{B}_{w,t}$, $\tilde{Q}_{w,t}$ and $\tilde{Q}_{xw,t}$ in (5.1) are obtained from

$$B_{w,t} = [\tilde{B}_{w,t} \ b], \quad Q_{w,t} = \begin{bmatrix} \tilde{Q}_{w,t} & q_w \end{bmatrix}, \quad Q_{xw,t} = [\tilde{Q}_{xw,t} \ q_{xw}] . \quad (5.39)$$

By using these implicit definitions of $\tilde{B}_{w,t} \in \mathbb{R}^{n_x \times (n_{w,t}-k)}$, $\tilde{Q}_{w,t} \in S_{++}^{n_{w,t}-k}$ and $\tilde{Q}_{xw,t} \in \mathbb{R}^{n_x \times (n_{w,t}-k)}$ in the definitions of $F_{t+1}$, $G_{t+1}$ and $H_{t+1}$, it follows that $\tilde{F}_{t+1}$, $\tilde{G}_{t+1}$, $\tilde{H}_{t+1}$ are given by

$$\tilde{F}_{t+1} = F_{t+1} + A^T V_{t+1}^+ C_{t+1}^T \tilde{V}_{t+1}^+ , \quad (5.40a)$$

$$\begin{bmatrix} \tilde{G}_{t+1} & \tilde{g}^T \strut \\
\tilde{g} & g^0 \end{bmatrix} = G_{t+1} + \begin{bmatrix} \tilde{B}_{w,t} V_{t+1}^+ & \strut \\
\tilde{b}^T V_{t+1}^+ \end{bmatrix} C_{t+1}^T \begin{bmatrix} V_{t+1}^T \tilde{B}_{w,t} & V_{t+1}^T b \end{bmatrix} , \quad (5.40b)$$

$$\begin{bmatrix} \tilde{H}_{t+1} & \tilde{b} \strut \\
\tilde{b} & h \end{bmatrix} = H_{t+1} + A^T V_{t+1}^+ C_{t+1}^T \begin{bmatrix} V_{t+1}^T \tilde{B}_{w,t} & V_{t+1}^T b \end{bmatrix} \quad (5.40c)$$
Re-arranging the terms in (5.40) gives the equivalent equations

\[ F_{t+1} = \tilde{F}_{t+1} - A_t^T V_{t+1}^+ C_{t+1}^+ V_{t+1}^+ T, \]

\[ G_{t+1} = \begin{bmatrix} \tilde{G}_{t+1}^T \\ \tilde{g}^T \\ \tilde{g}^0 \end{bmatrix} - \begin{bmatrix} B_{w,t}^T V_{t+1}^+ \\ b_{w,t}^T V_{t+1}^+ \end{bmatrix} C_{t+1}^+ \begin{bmatrix} V_{t+1}^T \tilde{B}_{w,t} V_{t+1}^+ T b \\ V_{t+1}^+ T b \end{bmatrix}, \]

\[ H_{t+1} = \begin{bmatrix} \tilde{H}_{t+1} \\ \tilde{h} - A_t^T V_{t+1}^+ C_{t+1}^+ \end{bmatrix} \begin{bmatrix} V_{t+1}^T \tilde{B}_{w,t} V_{t+1}^+ T b \\ V_{t+1}^+ T b \end{bmatrix}, \]

which are in a similar form as in (5.10). Then, the derivations in Section 5.1.2 apply also here and \( \tilde{M}_{t+1} \) and \( \hat{M}_{t+1} \) can be defined analogously as in (5.32), but using the matrices in (5.41), giving

\[
\begin{bmatrix}
\tilde{F}_{t+1} & \tilde{H}_{t+1} & \tilde{h} & A_t^T V_{t+1}^- \\
\tilde{G}_{t+1}^T & \tilde{g}^T & \tilde{g}^0 & B_{w,t}^T V_{t+1}^- \\
V_{t+1}^T A_t & V_{t+1}^T B_{w,t} & V_{t+1}^T b & C_{t+1}^- \\
\end{bmatrix} \geq 0. \tag{5.42}
\]

Positive semidefiniteness of \( \hat{M}_{t+1} \) can be shown as in Section 5.1.2. By using Lemma 5.1 for \( \tilde{M}_{t+1} \) in (5.42), it follows that

\[ P_t = \hat{M}_{t+1}^{1/2} \tilde{M}_{t+1} = \tilde{F}_{t+1} - \tilde{H}_{t+1} \tilde{G}_{t+1}^\dagger \tilde{H}_{t+1}^T - V_t^+ C_t^+ V_t^+ T \in \mathbb{S}^{n_x}_+, \]

and \( \hat{P}_t \) can be computed as

\[ \hat{P}_t = P_t + V_t^+ C_t^+ V_t^+ T \in \mathbb{S}^{n_x}_+. \]

Here \( V_t^+ \) and \( C_t^+ \) are defined as

\[ V_t^+ \triangleq \tilde{h} - \tilde{H}_{t+1} \tilde{G}_{t+1}^\dagger \tilde{g} \begin{bmatrix} A_t^T - \tilde{H}_{t+1} \tilde{G}_{t+1}^\dagger \tilde{B}_{w,t}^T \end{bmatrix} V_{t+1}^+, \]

\[ C_t^+ \triangleq \begin{bmatrix} \tilde{g}^0 \\ b_{w,t}^T V_{t+1}^+ \\ C_{t+1}^+ \end{bmatrix} - \begin{bmatrix} \tilde{g}^T \\ V_{t+1}^T \tilde{B}_{w,t} \end{bmatrix} \tilde{G}_{t+1}^\dagger \begin{bmatrix} \tilde{g} \\ \tilde{B}_{w,t} V_{t+1}^+ \end{bmatrix}, \]

\[ C_t^+ \in \mathbb{S}^{k+k}_+, \quad \mathcal{R} \left( \begin{bmatrix} V_t^+ \\ T \end{bmatrix} \right) \subset \mathcal{R} \left( C_t^+ \right). \]

Note the tildes in the right hand side of the definitions in (5.45), and that the update (5.45) at time \( t \) satisfies the assumption (5.21) that was made for the update at time \( t + 1 \). Hence, adding \( k \) control input constraints at time \( t \) to the working set when an update in the form (5.20) of size \( \tilde{k} \) already present results in an update of \( P_t \) in the same form as (5.20) but of increased dimension \( \tilde{k} + k \). The modified \( \tilde{K}_{t+1} \) can be computed by solving (5.28), but using the modified matrices in (5.41). Note that remarks 5.4 and 5.6 apply here as well.

**Remark 5.7.** If there is no modification of \( P_{t+1} \), then \( C_t^+ \triangleq g^0 - g^T \tilde{G}_{t+1}^\dagger g \in \mathbb{S}^k_+ \) and \( V_t^+ \triangleq h - \tilde{H}_{t+1} \tilde{G}_{t+1}^\dagger g \in \mathbb{R}^{n_x \times k} \), with \( \mathcal{R} \left( V_t^+ T \right) \subset \mathcal{R} \left( C_t^+ \right) \).
When adding \( k \) constraints to the working set, also \( k \) components in \( v_t \) in (3.23) are added and they are removed from \( w_t \). Hence, \( B_{v,t}, a_{v,t}, Q_{xv,t}, Q_{v,t}, Q_{uv,t}, l_{w,t} \) and \( l_{v,t} \) in (3.24a), (3.24b) and (3.25) are also affected accordingly (i.e., the corresponding \( k \) columns \( b \) in \( B_{w,t} \) are included in \( \tilde{B}_{v,t} \) etc.). These changes are only affecting the right-hand-side of the system of equations in the KKT system in (3.35), and hence they are not affecting the matrices in the factorization computed in Algorithm 4.

### 5.2 An Algorithm for Modifying the Riccati Factorization

In the previous sections, it was shown how the Riccati factorization is modified when constraints are added to, or removed from, the working set at an AS iteration. When constraints on \( u_t \) for a single time instance \( t \) are removed from the working set, then the factorization is modified as in Section 5.1.2 at time \( t \) followed by downdates as presented in Section 5.1.1 for time \( \tau \in \mathbb{Z}_{0,t-1} \). For the case of adding constraints at time \( t \) to the working set, the modification at time \( t \) is done as in Section 5.1.3, followed by updates as presented in Section 5.1.1 of the factorization for \( \tau \in \mathbb{Z}_{0,t-1} \). When either adding or removing several constraints at different time indices, the modifications can be handled in two different ways: either as several separate modifications which are independently performed after each other, or as modifications that are gradually increasing in size. The first approach is the naive way to do it. The second approach is described in sections 5.1.2 and 5.1.3, where it is shown how to increase the size of the modification when \( P_t \) are computed in (5.37) and (5.44), respectively. If the solver both adds and removes constraints in the same AS iteration, then several separate modifications of the Riccati factorization are required. According to Remark 5.4 it can sometimes be better to recompute parts of the Riccati factorization instead of modifying it.

The procedure for modifying the Riccati factorization using the theory presented in this chapter is summarized in Algorithm 8, which begins with modifying the matrices in the Riccati factorization according to Section 5.1.2 or Section 5.1.3 depending on whether constraints are removed or added to the working set, respectively. Let \( t_m \) be the largest time index where \( W_j \) is modified. Then, since \( P_{t_m+1} \) is not modified, Remark 5.5 or Remark 5.7, respectively, applies for the modification at time \( t_m \). For \( t < t_m \) the matrices in the Riccati factorization are modified as in Section 5.1.1 if the working set is not changed at time \( t \), as in Section 5.1.2 if constraints are removed from the working set at time \( t \), or as in Section 5.1.3 if constraints are added to the working set at time \( t \). Note that it is only possible to either add constraints to the working set or remove constraints from the working set at the same AS iteration using Algorithm 8. Furthermore, the Riccati factorization is only modified for \( t \leq t_m \leq N \). This is exploited also in Axehill (2008) in order to avoid re-computing the full Riccati factorization between AS iterations. To improve performance even further, the factorization of \( G_{t+1} \) can be modified to avoid re-computing it whenever that is possible as mentioned in Remark 5.3. Techniques for modifying factorizations are found in for example Golub and Van Loan (1996), Stewart (1998), and Nocedal and Wright (2006).
When using an AS method to solve CFTOC problems with $Q_{u,t} \in S_{++}^{n_{u,t}}$, and where $k$ constraints are removed at time $t_m$ in AS iteration $j$ and Cholesky factorizations of $G_{t+1}$ are used, the computational complexity for re-computing the Riccati factorization is approximately $O\left(N \left((n_w + k)^3 + n_w^2 + n_w^2(n_x + n_w^2(n_w + k))\right)\right)$. When the Riccati factorization is instead modified, the computational complexity reduces to approximately $O\left(t_m \left(n_w n_x + n_w^2 + k n_w n_x + kn_x^2\right)\right)$. If the solution to (5.28) is computed using the Sherman-Morrison-Woodbury formula, the complexity is further reduced to approximately $O\left(t_m \left(n_w^2 + kn_x^2 + kn_w n_x + kn_x^2\right)\right)$. Hence, by modifying the Riccati factorization using the theory presented in this chapter, it can be possible to reduce the computational complexity growth in the prediction horizon length and the problem dimensions to $O\left(t_m\right)$, $O\left(n_w^2\right)$ and $O\left(n_w^2\right)$, respectively. However, the exact expression for the complexity depends on, for example, the choice of factorization and modification techniques for $G_{t+1}$ in algorithms 4 and 8.

**Algorithm 8** Modification of the Riccati factorization

1. Set $t_m$ as the largest $t$ for which $W_j$ is modified
2. if Constraints are removed at time $t_m$ then
   3. Modify the factorization as in Section 5.1.2 using Remark 5.5
4. else if Constraints are added at time $t_m$ then
5. Modify the factorization as in Section 5.1.3 using Remark 5.7
6. end if
7. for $t = t_m - 1, \ldots, 0$ do
8. if No modification of $W_j$ at time $t$ then
   9. Modify factorization as in Section 5.1.1
10. else if Constraints are removed from $W_j$ at time $t$ then
11. Modify factorization as in Section 5.1.2
12. else if Constraints are added to $W_j$ at time $t$ then
13. Modify factorization as in Section 5.1.3
14. end if
15. end for

To compute the solution to the modified UFTOC problem when using the modified Riccati factorization, the recursions in algorithms 5-7 need to be re-computed. Since the factorization is only modified for $t \in \mathbb{Z}_{0,t_m}$, the backward recursion in Algorithm 5 only needs to be re-computed for $t \leq t_m$ using the modified matrices $\tilde{B}_{w,t}$, $\tilde{G}_t$, $\tilde{H}_t$ and $\tilde{P}_t$. The forward recursions in algorithms 6 and 7 need to be computed for the full horizon $N$.

**Computing the low-rank modifications in parallel**

When several constraints are added to or removed from the working set at AS iteration $j$, the modifications of the Riccati factorization can be handled by performing several independent modifications after each other as was mentioned in the introduction to this section. Hence, if several computational units are available, the modifications can be performed in parallel if they are synchronized properly. The details of how to perform the modifications in parallel are left as future work.
However, to illustrate the concept, consider a case when one constraint is added at
time $t_1$ and one at time $t_2 > t_1$. The Riccati factorization can be modified serially
by first computing the modification for $t_1$ completely, followed by the modification
for $t_2$. However, if two computational units are available it is possible to let unit 1
perform the modification for the constraint that was added at time $t_1$, and to let
unit 2 perform the modification for the constraint added at time $t_2$. Let $t^i$ denote
the time index in the Riccati factorization that is currently being modified by
unit $i$. Then, by enforcing the synchronization $t^1 < t^2$ it follows that the modified
Riccati factorization is identical when performing the modifications in parallel or
in serial. This example can easily be extended to the case when constraints are
added at the same time index.

### 5.3 Extension to more General Inequality Constraints

The CFTOC problem that is the core of many MPC problems in industry is often of
a more general type than (3.22) and also includes constraints on the states and the
possibility to control only certain states or a combination of states (Maciejowski,
2002). In this section, it is shown how to exploit low-rank modifications of the
Riccati factorization in order to improve performance when solving more general
problems than (3.22). The purpose of the section is not to introduce complete
new AS solvers that can be used to solve the more general CFTOC problems, but
rather to present how the theory in this chapter can be useful also when solving
more general problems than (3.22).

#### 5.3.1 Primal and dual problem

Here, a primal CFTOC problem of a more general type than (3.1) and a correspon-
ding dual problem is presented. This problem is considered here since it can be
shown that for a certain choice of dual variables, the dual problem is also in the
form of a CFTOC problem, see Axehill (2005) and Appendix 5.B. This property
will be exploited in this section. A superscript “$p$” is used to denote variables in
the primal problem, and a superscript “$d$” is used to denote variables in the dual
problem.

**Primal problem**

Consider a CFTOC problem with states $x^p_t \in \mathbb{R}^{n_x}$, controlled variables $z^p_t \in \mathbb{R}^{n_z,t}$
and control inputs $u^p_t \in \mathbb{R}^{n_u,t}$, and with inequality constraints that involve both
states and control inputs. This CFTOC problem is in a general form that covers
many linear MPC applications, and is given by the following optimization problem

\[
\begin{align*}
\text{minimize} \quad & \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} z^p_t & u^p_t \end{bmatrix}^T \begin{bmatrix} Q^p_{z,t} & Q^p_{zu,t} \\ Q^p_{zu,t} & Q^p_{u,t} \end{bmatrix} \begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix} + \begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix}^T \begin{bmatrix} c^p_t \\ c^p_t \end{bmatrix} \right) + \\
& \frac{1}{2} z^p_N^T Q^p_{z,N} z^p_N + l^p_{z,N}^T z^p_N + c^p_N \\
\text{subject to} \quad & x^p_0 = \bar{x} \\
& x^p_{t+1} = A^p_t x^p_t + B^p_t u^p_t + a^p_t, \quad t \in \mathbb{Z}_{0,N-1} \\
& z^p_t = M^p_t x^p_t, \quad t \in \mathbb{Z}_{0,N} \\
& H^p_{x,t} x^p_t + H^p_{u,t} u^p_t + h^p_t \preceq 0, \quad t \in \mathbb{Z}_{0,N-1} \\
& H^p_{x,N} x^p_N + h^p_N \preceq 0,
\end{align*}
\]

(5.46)

where \(H^p_{x,t} \in \mathbb{R}_{c,t}^{n_x \times n_x}, H^p_{u,t} \in \mathbb{R}_{c,t}^{n_x \times n_u,t}\) and \(h^p_t \in \mathbb{R}_{c,t}^{n_x,t}\). Here, \(n_{c,t}\) is the number of inequality constraints at time \(t\). Note that the controlled output \(z^p\), rather than the state \(x^p\), is penalized in the objective function. Furthermore, let Assumption 5.8 hold for the problem in (5.46).

**Assumption 5.8.**

\[
\begin{bmatrix} Q^p_{z,t} \\ Q^p_{zu,t} \\ Q^p_{u,t} \end{bmatrix} \in \mathbb{S}_{++}^{n_z,t+n_u,t}, \quad t \in \mathbb{Z}_{0,N-1}, \quad Q^p_{z,N} \in \mathbb{S}_{++}^{n_z,N}.
\]

(5.47)

The CFTOC problem (5.46) can be solved using for example a primal AS solver, but the theory presented in this chapter is not directly applicable in that case due to the more complex inequality constraints than the ones used in (3.22). Hence, exploiting low-rank modifications of the Riccati factorization instead of re-computing it cannot be done to increase performance. However, the primal problem in (5.46) can also be solved using a dual AS solver as in Axehill and Hansson (2006), or using a dual gradient projection method as in Axehill (2008). In these types of solvers it is sometimes possible to exploit low-rank modifications of the Riccati factorization to speed up computations.

**Dual problem**

In Axehill (2005), an optimization problem which is equivalent to the (Lagrange) dual problem of a slightly less general problem than (5.46) is introduced. The primal problem in that work does not have any linear terms in the objective function and the cross-terms between \(z^p_t\) and \(u^p_t\) are zero, and it is shown that also the dual problem can be interpreted as a CFTOC problem using a suitable variable transformation. This dual problem is used in the solvers in Axehill and Hansson (2006), Axehill (2008), and Axehill and Hansson (2008).

In this thesis, a dual problem to (5.46) is derived using analogous calculations as in Axehill (2005). The details of the derivation are presented in Appendix 5.B. The dual problem can be interpreted as a CFTOC problem with states \(x^d_{\tau} \in \mathbb{R}^{n_x}\), control
inputs $u^d \in \mathbb{R}^{n_z,N−t+n_c,N−\tau}$, prediction horizon $N^d = N + 1$, and no quadratic cost on the final state, given in the form

$$\begin{align*}
\text{minimize} & \quad \sum_{\tau=0}^{N^d-1} \left( \frac{1}{2} \begin{bmatrix} x_{\tau}^d \\ u_{\tau}^d \end{bmatrix}^T Q_{x,\tau}^d \begin{bmatrix} x_{\tau}^d \\ u_{\tau}^d \end{bmatrix} + \begin{bmatrix} l_{x,\tau}^d \\ l_{u,\tau}^d \end{bmatrix}^T \begin{bmatrix} x_{\tau}^d \\ u_{\tau}^d \end{bmatrix} + c_{\tau}^d \right) + \\
& \quad \left( \begin{bmatrix} l_x^d \\ l_d^d \end{bmatrix} \right)^T x_{N^d}^d \\
\text{subject to} & \quad x_0^d = 0 \\
& \quad x_{\tau+1}^d = A^d_{\tau} x_{\tau}^d + B^d_{\tau} u_{\tau}^d, \quad \tau \in \mathbb{Z}_{0,N^d-1} \\
& \quad \begin{bmatrix} 0 \\ -I_{n_c,N-\tau} \end{bmatrix} u_{\tau}^d \preceq 0, \quad \tau \in \mathbb{Z}_{0,N^d-1},
\end{align*}$$

(5.48)

where $\tau = N - t$ and

$$\begin{bmatrix} Q_{x,\tau}^d \\ Q_{z,u,\tau}^d \end{bmatrix} \in \mathbb{S}_+^{n_x+n_z,N−t+n_c,N−\tau}, \quad Q_{x,N^d}^d \in \mathbb{S}_+^{n_x}.
$$

(5.49)

The state variables, control inputs and all matrices and vectors in the dual CFTOC problem are defined in Appendix 5.B. The fact that $Q_{u,\tau}^d$ is not necessarily positive definite is the main reason for considering the singular case in this chapter.

The solution to the primal CFTOC problem (5.46) can be derived from the solution to the dual CFTOC problem (5.48). The relations between the solutions to the primal and dual problems are given by (5B.80) and (5B.84) in Appendix 5.B, but are repeated here for convenience:

$$\begin{align*}
x^p_t &= -\lambda_{N+1-t}, \quad t \in \mathbb{Z}_{0,N}, \\
u^p_t &= -\left( \bar{Q}_{zu,t} \right)^T l_{z,t}^p - \bar{Q}_{u,t} l_{u,t}^p - \bar{Q}_{u,t} \left( B_t^p \right)^T x^d_{N-t} + \\
&\quad \left( \bar{Q}_{zu,t} \right)^T - \bar{Q}_{u,t} \left( H_{u,t}^p \right)^T \right) u^d_{N-t}, \quad t \in \mathbb{Z}_{0,N-1},
\end{align*}$$

(5.50a, 5.50b)

where $\lambda_{\tau}$ are defined in (5B.81a) as the dual variables corresponding to the equality constraints in the dual problem, and $\bar{Q}_{zu,t}$ and $\bar{Q}_{u,t}$ are defined in (5B.72).

### 5.3.2 Solving the primal problem by exploiting its dual problem

Here, two approaches for exploiting the theory for modifying the Riccati factorization even in the case with more general inequality constraints than what is used in (3.22) are presented.

#### Solving the dual inequality constrained problem

The primal problem (5.46) can be solved using for example a dual as type solver as proposed in Axehill and Hansson (2006), or a dual gradient projection method as in Axehill (2008) and Axehill and Hansson (2008). In these types of methods, the primal problem is solved by constructing the corresponding dual problem and applying primal methods to solve that dual problem. The solver iterates until dual optimality is obtained, while maintaining dual feasibility. Since Slater’s condition
in Definition 2.6 is satisfied for (5.46), strong duality holds (except for a change of sign to obtain a minimization problem in (5.48)). Hence, if the dual problem is bounded from below, the dual optimal point for (5.48) corresponds to a primal feasible and optimal point for (5.46) (Boyd and Vandenberghe, 2004).

When using the previously mentioned solvers, the dual CFTOC problem is solved by iteratively changing the working set to find the optimal active set as in a primal AS method. Furthermore, the dual CFTOC problem (5.48) is in the same form as the CFTOC problem (3.22). Hence, it follows that (5.48) is solved by computing a sequence of search directions corresponding to the solutions to UFTOC problems in the form in (5.1). Since these solvers exploit the structure in (5.1) by using a Riccati recursion to compute the search directions, the theory presented in this chapter can readily be applied to perform low-rank modifications of the Riccati factorization. An outline of the main steps in a dual AS type solver such as the one in Axehill and Hansson (2006) is given in Algorithm 9.

Algorithm 9 Outline of dual AS type solver

1: Construct the dual problem (5.48) from the primal problem (5.46)
2: Find a dual feasible point (the origin is a trivial choice for a dual AS solver and no phase I algorithm is needed) and an initial working set
3: repeat
4: Compute the dual search direction by solving the problem in (5.1)
5: Perform a step in the dual variables $x^d$ and $u^d$, and update the working set if needed
6: until Dual optimal point $x^{d*}$ and $u^{d*}$ is found
7: Compute the primal optimal point from (5.50)

The primal method that is used to solve the dual problem must handle the positive semidefinite case (Goldfarb and Idnani, 1983). In Axehill and Hansson (2006), this is done by imposing extra assumptions on the inequality constraints in (5.46).

Computing the search directions in the dual

When a dual solver is used to solve (5.46) primal feasibility is obtained only at the optimum (Goldfarb and Idnani, 1983; Bartlett and Biegler, 2006). In for example a real-time MPC control loop, this might be problematic due to real-time computation time constraints. If the solver that is applied to solve the CFTOC problem is forced to terminate before an optimum is found, the control input that is returned by the solver is not necessarily primal feasible. Hence, either a control input which might lead to violation of constraints is used as input to the plant, or the control input has to be computed in some alternative way. An approach to address this problem, and still be able to exploit low-rank modifications of the Riccati factorization, is to use a primal solver which maintains primal feasibility but that computes the primal search directions by solving dual UFTOC problems for the fixed working sets. The main idea with such a method will described for a primal CFTOC problem in the form (5.46), and a sketch of the main steps is given in Algorithm 10 at the end of this section.
Let the inequality constraint matrices be decomposed as

\[
\begin{bmatrix}
H_{x,1,t}^P \\
\vdots \\
H_{x,n_c,t,t}^P
\end{bmatrix}
\triangleq H_{x,t}^P,
\begin{bmatrix}
H_{u,1,t}^P \\
\vdots \\
H_{u,n_c,t,t}^P
\end{bmatrix}
\triangleq H_{u,t}^P,
\begin{bmatrix}
h_{1,t}^P \\
\vdots \\
h_{n_c,t,t}^P
\end{bmatrix}
\triangleq h_t^P,
\]

and let the notation \((i, t) \in W_j\) denote that the inequality constraint

\[H_{x,i,t}^P x_t + H_{u,i,t}^P u_t + h_{i,t}^P \leq 0,\]

is in the working set \(W_j\) and is hence forced to hold with equality.

From Section 2.6 it follows that when a primal \(\text{AS}\) method like for example the one given in Algorithm 1 is applied to solve the \(c\text{ftoc}\) problem (5.46), an equality constrained optimization problem in the form

\[
\begin{align*}
\text{minimize} & \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} [z_t^P x_t^P + Q_{z,t}^P T Q_{z,u,t}^P] [z_t^P u_t^P] + \frac{1}{2} z_N^P T Q_z^P z_N^P + c_t^P \right) + \\
\text{subject to} & \quad x_0^P = \bar{x} \\
& \quad x_{t+1}^P = A_t x_t^P + B_t u_t^P + a_t^P, \quad t \in Z_0, N-1 \\
& \quad z_t^P = M_t^P x_t^P, \quad t \in Z_0, N \\
& \quad H_{x,i,t}^P x_t^P + H_{u,i,t}^P u_t^P + h_{i,t}^P = 0, \quad \forall (i, t) \in W_j \\
& \quad H_{x,i,N}^P x_N^P + h_{i,N}^P = 0, \quad \forall (i, N) \in W_j,
\end{align*}
\]

is solved at each \(\text{AS}\) iteration \(j\). In principle, it is possible to solve (5.53) by solving its dual problem and then computing the corresponding primal solution from the solution to this dual problem. The dual problem to (5.53) is constructed from the problem in (5.48) by letting some of the inequality constraints in (5.48) hold with equality, and temporarily disregarding the rest. How to do this is shown next.

The primal constraints, the dual control inputs and the working set \(W_j\) are related via the dual variables \(\gamma_t\) for the primal constraints, which are defined in (5B.65d) and (5B.65e). The relations are given by the following two cases:

- For all primal inequality constraints \((i, t) \in W_j\) that are forced to hold with equality in the primal problem in (5.53), the corresponding dual variable \(\gamma_{i,t}\) is not constrained since the complementarity slackness condition (2.17e) holds for all choices of \(\gamma_{i,t}\). Hence, by using the definition of \(u_{d,\tau}^T\) in (5B.76c) in Appendix 5.B and the relation \(\tau = N - t\), it follows that \(u_{n_z,t+i,N-t}^d\) is an unconstrained optimization variable in the dual problem in (5.54).

- For all primal inequality constraints \((i, t) \in W_j^c\) that are temporarily disregarding, \(i.e.,\) are not in the working set, the corresponding dual variable is defined as \(\gamma_{i,t} = 0\) (Nocedal and Wright, 2006). Then, from the definition of \(u_{d,\tau}^T\), it follows that \(u_{n_z,t+i,N-t}^d = 0\).
To summarize: if the primal constraint \((i, t)\) is in the working set then the corresponding dual control input is unconstrained in the dual problem, and if the primal constraint \((i, t)\) is not in the working set then the corresponding dual control input is constrained to zero. Then, the dual problem of the primal equality constrained problem in (5.53) is given by

\[
\begin{aligned}
\text{minimize} & \quad x^d, u^d \\
\text{subject to} & \quad x^d_0 = 0 \\
& \quad x^d_{\tau + 1} = A^d_{\tau} x^d_{\tau} + B^d_{\tau} u^d_{\tau}, \quad \tau \in \mathbb{Z}_{0,N^d-1} \\
& \quad -u^d_{n_z,N-\tau+i,\tau} = 0, \quad \forall (i, N - \tau) \in W^c_j.
\end{aligned}
\]  

(5.54)

The primal solver requests a solution to (5.53) to compute the primal search direction. The update of the primal variables in the primal AS solver does not depend on how the search direction is computed, except for numerical differences. Hence, the logics involved in performing the update step and possibly adding or removing constraints from the working set are the same regardless of how the search direction is computed. As a consequence, instead of solving (5.53) to compute the primal search direction, it can be computed by solving (5.54). The solution to (5.53) can then easily be recovered from the solution to (5.54) using (5.50). Hence, the primal variables in the AS solver can be updated using the primal search direction that is computed from the dual UFTOC problem while also maintaining primal feasibility.

By eliminating the constrained variables \(u^d_{n_z,N-\tau+i,\tau}\) from the dual problem (5.54), it can be interpreted as a UFTOC problem in the same form as (5.1). Furthermore, the dual problem (5.54) is constructed such that adding a primal constraint to the working set \(W_j\) corresponds to removing a dual constraint, and vice versa. Hence, the dual problem is changed as in Section 3.2.2 when primal inequality constraints are added or removed from the working set. As a result, low-rank modification of the Riccati factorization as presented in this chapter can be exploited when computing the solution to the dual UFTOC problem. A primal AS solver that computes the primal search direction by solving a dual equality constrained problem is outlined in Algorithm 10. Note that Algorithm 10 is not a detailed description of an AS solver, but it is included to highlight how to exploit the theory presented in this chapter for problems with more general constraints than in (3.22).

In Algorithm 10 the dual CFTOC problem (5.48) is constructed only once, and in the MPC application for the case with LTI systems this can be done once offline. By inspecting the structure of the dual UFTOC problem that is solved at each AS iteration, it can be seen that only the equality constraints \(u^d_{n_z,N-\tau+i,\tau} = 0\) are changed between different iterations. Hence, only computationally cheap operations are needed at Line 4 in Algorithm 10.
Algorithm 10 Outline of primal AS solver that computes the search direction by solving a dual UFTOC problem

1: Construct the dual problem (5.48) from the primal problem (5.46)
2: Find a primal feasible point (for example by using a Phase I approach) and an initial working set
3: repeat
   4: Form the dual UFTOC problem (5.54)
   5: Solve the problem (5.54)
   6: Compute the variables $x^p_{j+1}$ and $u^p_{j+1}$ using (5.50). If needed, compute also the dual variables $\alpha_{j+1}, \beta_{j+1}, \gamma_{j+1}$ for the primal problem from (5B.76)
   7: Perform the update step in the primal variables $x^p$ and $u^p$ and change the working set as in for example Algorithm 1 if necessary
4: until Primal optimal point $x^p*$ and $u^p*$ is found

5.4 Numerical Results

In this section, the proposed algorithm for solving the UFTOC problem (5.1) by modifying the Riccati factorization between AS iterations is compared to re-computing the standard Riccati recursion. All computation times that are presented are obtained from computing the full solution of the UFTOC problems, including the Riccati factorization and the backward and forward recursions given in algorithms 5 and 6, respectively. A proof-of-concept implementation is made in MATLAB, where most of the main operations such as Cholesky factorizations have been implemented in m-code to get a fair comparison of the computational times. In the implementation used here, the gaxpy Cholesky in Golub and Van Loan (1996) and the Cholesky modifications from Stewart (1998) are used. The m-code is converted into C-code using MATLAB’s code generation framework, and the generated C-code is used to produce the numerical results presented in this section. As always, to get a completely fair comparison of the algorithms, fully optimized implementations in a compiled language should be used. However, this is outside the scope of this thesis and is left as future work.

The numerical results were produced by executing the C-code on an Intel Xeon W3565 @3.2 GHz processor running Linux (version 2.6.32-504.12.2.el6.x86_64) and MATLAB (version 9.1.0.441655 (R2016b)). The default settings have been used for the code generation in MATLAB, with the exception that the compilation flag ‘-O3’ has been used to optimize the code for speed.

The algorithms are compared by solving random UFTOC problems in the form (5.1) for stable LTI systems. The dimensions $n_x$ and $n_w$ of the problem are chosen as 10 different values that are logarithmically spaced in the interval $\mathbb{Z}_{10,200}$, giving the set of dimensions as $\{10, 14, 20, 28, 38, 53, 74, 103, 143, 200\}$. The computation times are averaged over 20 different problems of the same dimensions. In Figure 5.1 the computation times are normalized with respect to the maximum computation time 0.32 seconds for the standard Riccati recursion. In this figure, problems with $N =$
have been solved after removing a constraint at either $t_m = 0$ (modifying one step of the factorization) or $t_m = N - 1$ (modifying the full factorization), which are the best and worst cases for the modifying algorithm, respectively. In Figure 5.2 a similar plot is presented, but for problems with $N = 100$. Here, the normalization is with respect to the maximum computation time 3.25 seconds for the standard Riccati recursion. In the figures 5.3 and 5.5, the normalized computation times for fixed $n_x = 10$ and $n_w = 10$, respectively, are seen. In the figures 5.4 and 5.6, zoomed-in versions of these plots are presented. These four plots are re-scaled slices of the surface plots in Figure 5.1. Furthermore, in Figure 5.7 the performance gains for different $N$ and $t_m \in \{1, \frac{N}{4}, \frac{3N}{4}, N\}$ are investigated by plotting the ratio between the computation times when modifying the Riccati factorization and re-computing it. The ratios are presented for problems of dimension $n_x = n_w = 10$, $n_x = n_w = 103$ and $n_x = n_w = 200$, respectively.

The numerical results presented in the figures clearly show that modifying the Riccati factorization instead of re-computing it can significantly reduce the computation time for solving the UFTOC problem (5.1). The proposed algorithm is especially efficient for large problem sizes and/or when only a small part of the factorization is modified. However, as indicated in Figure 5.4 and Figure 5.6, it is also possible to decrease the computation time even for smaller problem dimensions than what is seen in figures 5.3 and 5.5. By comparing Figure 5.1 and Figure 5.2 it is seen that the performance gains are similar even though there is a factor 10 between the prediction horizons, and hence the advantages of the proposed algorithm is not only for long prediction horizons. In Figure 5.7 it can, for example, be seen that for the worst case (when the full factorization needs to be modified), the computation time for solving a UFTOC problem with $n_x = 10$ and $n_w = 10$ using the proposed modification algorithm requires only approximately 70% of the computation time that is required when re-computing the factorization. The decrease in computation time is even larger for problems of larger dimensions or when only a part of the Riccati factorization is modified. The $O(t_m)$ computational complexity growth for the modifying algorithm, which is presented in Section 5.2, is numerically verified in Figure 5.7, where it is also shown that the performance is comparable for all $N \in \{10, 20, 40, 60, 80, 100\}$.

The accuracies of the numerical solutions have been measured as the Euclidean norm of the KKT residual for the UFTOC problem (5.1). For a problem with $N = 100$ and $n_x = n_w = 200$, the maximum residual norm is in the order $10^{-10}$ for both the standard Riccati recursion and the proposed algorithm. Hence, practical experience from the numerical results indicate that the modification algorithm does not loose in numerical accuracy compared to the standard Riccati recursion.
Figure 5.1: Normalized computation time for solving a UFTOC problem with $N = 10$ after one constraint is removed from the working set. The Riccati factorization is computed using both the standard (Algorithm 4) and modifying (Algorithm 8) algorithms, and the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 0.32 seconds for the standard Riccati recursion.
Figure 5.2: Normalized computation time for solving a UFTOC problem with $N = 100$ after one constraint is removed from the working set. The Riccati factorization is computed using both the standard (Algorithm 4) and modifying (Algorithm 8) algorithms, and the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 3.25 seconds for the standard Riccati recursion.
Figure 5.3: A comparison of computation times for problems with $n_x = 10$ and $N = 10$ as a function of $n_w$. Here the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 0.11 seconds for the standard Riccati recursion.
Figure 5.4: A comparison of computation times for problems with $n_x = 10$, $N = 10$ and $n_w \in \{10, 14, 20, 28\}$. Here the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 0.001 seconds for the standard Riccati recursion. Note that this is a scaled and zoomed-in version of the plot presented in Figure 5.3.
Figure 5.5: A comparison of computation times for problems with $n_w=10$ and $N=10$ as a function of $n_x$. Here the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 0.04 seconds for the standard Riccati recursion.
Figure 5.6: A comparison of computation times for problems with $n_w = 10$, $N = 10$ and $n_x \in \{10, 14, 20, 28\}$. Here the cases where only one step (best case) and all steps (worst case) are modified are shown. The normalization is with respect to the maximum computation time 0.001 seconds for the standard Riccati recursion. Note that this is a scaled and zoomed-in version of the plot presented in Figure 5.5.
Figure 5.7: The ratio between the computation time for solving a UFTOC problem when computing the Riccati factorization using Algorithm 8 and using Algorithm 4, respectively. Here, the ratios for UFTOC problems with $n_x = n_w = 200$, $n_x = n_w = 103$ and $n_x = n_w = 10$ are presented.
Appendix

In this appendix, the proof of the quotient formula for GSCs in Lemma 5.1 is presented. Furthermore, the derivation of the dual CFTOC problem (5.48) and the definition of the dual variables are presented here.

5.A Proof of Lemma 5.1

Proof: Consider the positive semidefinite matrices \( \mathcal{M} \succeq 0 \) and \( \tilde{\mathcal{M}} \succeq 0 \) which are partitioned as
\[
\mathcal{M} = \begin{bmatrix} M_{11} & \tilde{\mathcal{M}} \\ \tilde{\mathcal{M}}^T & M \end{bmatrix}, \quad \tilde{\mathcal{M}} = \begin{bmatrix} M_{22} & M_{23} \\ M_{23}^T & M_{33} \end{bmatrix}, \quad \tilde{\mathcal{M}} = [M_{12} \ M_{13}] . \tag{5A.55}
\]
From Theorem A.10 on page 168 in Appendix A and \( \tilde{\mathcal{M}} \succeq 0 \) it follows that
\[
\tilde{\mathcal{M}} / \tilde{\mathcal{M}}^\dagger M_{22} = M_{33} - M_{23}^T \tilde{\mathcal{M}}^\dagger M_{23} \succeq 0 . \tag{5A.56}
\]
Furthermore, since \( \mathcal{M} \succeq 0 \) it follows from Theorem A.10 and Lemma A.4 that
\[
\mathcal{R} \left( \tilde{\mathcal{M}}^T \right) \subset \mathcal{R} \left( \tilde{\mathcal{M}} \right) \text{ (Lancaster and Tismenetsky, 1985), and from Theorem A.14 on page 168 it follows that that}
\]
\[
\mathcal{R} \left( M_{13}^T - M_{23}^TM_{22}^\dagger M_{12}^T \right) \subset \mathcal{R} \left( M_{33} - M_{23}^TM_{22}^\dagger M_{23} \right) . \tag{5A.57}
\]
By defining the permutation matrix
\[
\Pi \triangleq \begin{bmatrix} 0 & 0 & I \\ 0 & I & 0 \\ I & 0 & 0 \end{bmatrix} , \tag{5A.58}
\]
the matrix \( A \) used in Theorem 4 in Carlson et al. (1974) can be written
\[
A = \Pi \mathcal{M} \Pi^T . \tag{5A.59}
\]
Hence, the relation
\[
\left( \mathcal{M} / \mathcal{M}_{33} \right)^\dagger \left( \tilde{\mathcal{M}} / \mathcal{M}_{33} \right) = \mathcal{M} / \tilde{\mathcal{M}} , \tag{5A.60}
\]
follows directly from Theorem 4 in Carlson et al. (1974).

Furthermore, since $\mathcal{R}(\hat{\mathcal{M}}^T) \subset \mathcal{R}(\tilde{\mathcal{M}})$ there exists a matrix $P$ such that

$$\tilde{\mathcal{M}}P = \hat{\mathcal{M}}^T,$$  \hspace{1cm} (5A.61)

and the matrix $\mathcal{M}$ can thus be written

$$\mathcal{M} = \begin{bmatrix} M_{11} & P^T \tilde{\mathcal{M}} \\ \tilde{\mathcal{M}}P & \hat{\mathcal{M}} \end{bmatrix}.$$  \hspace{1cm} (5A.62)

By using this representation and Theorem A.10 on page 168, the gsc of $\tilde{\mathcal{M}}$ in $\mathcal{M}$ can be written

$$\mathcal{M}^\dagger \tilde{\mathcal{M}} = M_{11} - P^T \tilde{\mathcal{M}}^\dagger \tilde{\mathcal{M}} = M_{11} - P^T \tilde{\mathcal{M}} \succeq 0.$$  \hspace{1cm} (5A.63)

Here, the property (A.1a) on page 165 of the pseudo inverse has been used.

From Theorem A.14 on page 168, a solution to (5A.61) can be computed. By inserting this solution into (5A.63) and using standard matrix algebra gives

$$\mathcal{M}^\dagger \tilde{\mathcal{M}} = M_{11} - P^T \tilde{\mathcal{M}} \tilde{\mathcal{M}}^\dagger \tilde{\mathcal{M}}^T \left( M_{13} - M_{12} M_{22} M_{23} \right) \left( M_{33} - M_{23} M_{12} M_{23} \right)^\dagger \left( M_{13}^T - M_{23}^T M_{12}^T M_{23} \right),$$  \hspace{1cm} (5A.64)

which concludes the proof.

5.B Derivation of the Dual Problem

The derivation of the dual CFTOC problem (5.48) will be similar to the derivation of the dual problem in Axehill (2005). However, although the derivations are essentially a repetition of those in Axehill (2005), the derivation for the slightly more general CFTOC problem (5.46) is presented in this section for completeness.

5.B.1 Constructing the dual problem

Consider a primal CFTOC problem given in the form in (5.46) where Assumption 5.8 holds. Introduce the dual variables $\alpha_t$, $\beta_t$ and $\gamma_t$ that are associated with the constraints in the primal CFTOC problem (5.46) as

$$\alpha_0 \leftrightarrow -x^p_0 + \bar{x} = 0,$$  \hspace{1cm} (5B.65a)

$$\alpha_{t+1} \leftrightarrow -x^p_{t+1} + A^p_t x^p_t + B^p_t u^p_t + a^p_t = 0, \; t \in \mathbb{Z}_{0,N-1},$$  \hspace{1cm} (5B.65b)

$$\beta_t \leftrightarrow -z^p_t + M^p_t x^p_t = 0, \; t \in \mathbb{Z}_{0,N},$$  \hspace{1cm} (5B.65c)

$$\gamma_t \leftrightarrow H^p_{x,t} x^p_t + H^p_{u,t} u^p_t + h^p_t \leq 0, \; t \in \mathbb{Z}_{0,N-1},$$  \hspace{1cm} (5B.65d)

$$\gamma_N \leftrightarrow H^p_{x,N} x^p_N + h^p_N \leq 0,$$  \hspace{1cm} (5B.65e)

where the use of the notation $\leftrightarrow$ is introduced in Definition 3.5.

The first step to derive the dual problem is to form the Lagrangian of the primal
The CFTOC problem using the definitions of the dual variables in (5B.65). By using the definition of the Lagrangian in (2.8), it follows that it is given by

\[ L_P(x^p, u^p, z^p, \alpha, \beta, \gamma) = \sum_{t=0}^{N-1} \left( \frac{1}{2} \left[ z_t^p \right]^T \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \end{bmatrix} \begin{bmatrix} z_t^p \\ u_t^p \end{bmatrix} + \begin{bmatrix} z_t^p \\ u_t^p \end{bmatrix}^T \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \end{bmatrix} \right) + \frac{1}{2} z_N^p T Q_{z,N}^p z_N^p + l_N^p T z_N^p + c_N^p + \sum_{t=0}^{N-1} \alpha_{t+1}^T \left( -x_{t+1}^p + A_t^p x_t^p + B_t^p u_t^p + a_t^p \right) + \\alpha_0^T (-x_0^p + \bar{x}) + \sum_{t=0}^N \beta_t^T (-z_t^p + M_t^p x_t^p) + \sum_{t=0}^{N-1} \gamma_t^T \left( H_{x,t}^p x_t^p + H_{u,t}^p u_t^p + h_t^p \right) + \\gamma_N^T \left( H_{x,N}^p x_N^p + h_N^p \right), \tag{5B.66} \]

with the corresponding Lagrange dual function

\[ g(\alpha, \beta, \gamma) = \inf_{x^p, u^p, z^p} L_P(x^p, u^p, z^p, \alpha, \beta, \gamma). \tag{5B.67} \]

To simplify the minimization over the primal variables \( x^p, u^p, z^p \) in (5B.67), the terms in the Lagrangian (5B.66) are re-ordered, resulting in

\[ L_P(x^p, u^p, z^p, \alpha, \beta, \gamma) = \sum_{t=0}^{N-1} \left( \frac{1}{2} \left[ z_t^p \right]^T \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \end{bmatrix} \begin{bmatrix} z_t^p \\ u_t^p \end{bmatrix} + \begin{bmatrix} z_t^p \\ u_t^p \end{bmatrix}^T \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \end{bmatrix} \right) + \\frac{1}{2} z_N^p T Q_{z,N}^p z_N^p + l_N^p T z_N^p + c_N^p + \sum_{t=0}^{N-1} \left( \alpha_{t+1}^T B_t^p + \gamma_t^T H_{u,t}^p \right) u_t^p - \sum_{t=0}^N \beta_t^T z_t^p + \\sum_{t=0}^{N-1} \left( -\alpha_t^T + \alpha_{t+1}^T A_t^p + \beta_t^T M_t^p + \gamma_t^T H_{x,t}^p \right) x_t^p + \left( -\alpha_N^T + \beta_N^T M_N^p + \gamma_N^T H_{x,N}^p \right) x_N^p + \\alpha_0^T \bar{x} + \sum_{t=0}^N \alpha_{t+1}^T a_t^p + \sum_{t=0}^{N-1} \gamma_t^T h_t^p + \gamma_N^T h_N^p. \tag{5B.68} \]

When Assumption 5.8 holds, it follows that (5B.68) is strictly convex in the primal variables \( u^p \) and \( z^p \), and linear in the primal variable \( x^p \). Hence, if (5B.68) is minimized with respect to all primal variables, the optimal value is not bounded below unless the coefficient in front of \( x^p \) is zero. As a result, the explicit equality constraints that exclude all the values of the dual variables \( \alpha_t, \beta_t, \gamma_t \) such that \( g(\alpha, \beta, \gamma) = -\infty \) can be formulated as

\[ -\alpha_t + \left( A_t^p \right)^T \alpha_{t+1} + \left( M_t^p \right)^T \beta_t + \left( H_{x,t}^p \right)^T \gamma_t = 0, \ t \in \mathbb{Z}_{0,N-1}, \tag{5B.69a} \]

\[ -\alpha_N + \left( M_N^p \right)^T \beta_N + \left( H_{x,N}^p \right)^T \gamma_N = 0. \tag{5B.69b} \]

These can be identified as the transpose of the coefficients in front of \( x_t^p \) and \( x_N^p \).
in (5B.68), respectively. By inserting (5B.69a) and (5B.69b) into (5B.68), and re-arranging some terms, the Lagrangian can be written

\[
L_P(u^p, z^p, \alpha, \beta, \gamma) = \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix}^T \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \\ Q_{z,u,t}^p & Q_{u,u,t}^p \end{bmatrix} \begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix} + l^p_{z,t} - \beta_t \right) + \left( \begin{bmatrix} p^p_{u,t} + (B_t^p)^T \alpha_{t+1} + (H_{u,u,t}^p)^T \gamma_t \end{bmatrix}^T \begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix} + c^p_t + (a^p_t)^T \alpha_{t+1} + (h^p_t)^T \gamma_t \right) + \frac{1}{2} z^p_N Q_{z,N}^{-1} z^p_N + c^p_N + (h^p_N)^T \gamma_N + \bar{x}^T \alpha_0,
\]

(5B.70)

which is a strictly convex quadratic function in the primal variables \( z^p \) and \( u^p \). From basic optimization results it follows that the minimum of (5B.70) with respect to \( z^p \) and \( u^p \) is obtained at

\[
\begin{bmatrix} z^p_t \\ u^p_t \end{bmatrix} = - \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \\ \bar{Q}_{z,u,t}^p & \bar{Q}_{u,u,t}^p \end{bmatrix}^{-1} \begin{bmatrix} l^p_{z,t} - \beta_t \\ \bar{Q}_{u,u,t}^p \alpha_{t+1} + (H_{u,u,t}^p)^T \gamma_t \end{bmatrix},
\]

(5B.71a)

\[
z^p_N = - (Q_{z,N}^p)^{-1} (l^p_{z,N} - \beta_N).
\]

(5B.71b)

Inserting this solution into the Lagrangian (5B.70) and introducing the notation

\[
\begin{bmatrix} \bar{Q}_{z,t} \\ \bar{Q}_{u,u,t} \end{bmatrix} \triangleq \begin{bmatrix} Q_{z,t}^p & Q_{z,u,t}^p \\ \bar{Q}_{z,u,t}^p & \bar{Q}_{u,u,t}^p \end{bmatrix}^{-1},
\]

(5B.72)

to simplify notation gives (after standard, but tedious, matrix algebra and re-arrangements) the Lagrange dual function \( g(\alpha, \beta, \gamma) \) as

\[
- g(\alpha, \beta, \gamma) = \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} \alpha_{t+1} \\ \beta_t \end{bmatrix}^T \begin{bmatrix} B_t^p \bar{Q}_{u,u,t}^p (B_t^p)^T & -B_t^p (\bar{Q}_{z,u,t}^p)^T \\ -\bar{Q}_{z,u,t}^p (B_t^p)^T & \bar{Q}_{z,t} \end{bmatrix} \begin{bmatrix} \alpha_{t+1} \\ \beta_t \end{bmatrix} + \begin{bmatrix} l^p_{z,t} - \beta_t \\ \bar{Q}_{u,u,t}^p \alpha_{t+1} + (H_{u,u,t}^p)^T \gamma_t \end{bmatrix} \right) + \frac{1}{2} \begin{bmatrix} \beta_N \end{bmatrix}^T \begin{bmatrix} (Q_{z,N}^p)^{-1} \beta_N \\ \gamma_N \end{bmatrix} + \frac{1}{2} l^p_{z,N} (Q_{z,N}^p)^{-1} l^p_{z,N} - \bar{x}^T \alpha_0,
\]

(5B.73)
which is a concave function. Note the “−” sign in the left hand side. Hence, by explicitly including the constraints (5B.69) for which the Lagrangian is bounded from below in the primal variables, the dual problem (2.11) corresponding to (5.46) can be written as

\[
\begin{align*}
\text{maximize} & \quad g(\alpha, \beta, \gamma) \\
\text{subject to} & \quad -\alpha_t + (A_{\tau}^T)^T \alpha_{t+1} + (M_t^p)^T \beta_t + \left(H_{x,t}^p\right)^T \gamma_t = 0, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad -\alpha_N + (M_N^p)^T \beta_N + \left(H_{x,N}^p\right)^T \gamma_N = 0 \\
& \quad \gamma_t \succeq 0, \quad t \in \mathbb{Z}_{0,N},
\end{align*}
\]

or equivalently

\[
\begin{align*}
\text{minimize} & \quad -g(\alpha, \beta, \gamma) \\
\text{subject to} & \quad -\alpha_t + (A_{\tau}^T)^T \alpha_{t+1} + (M_t^p)^T \beta_t + \left(H_{x,t}^p\right)^T \gamma_t = 0, \quad t \in \mathbb{Z}_{0,N-1} \\
& \quad -\alpha_N + (M_N^p)^T \beta_N + \left(H_{x,N}^p\right)^T \gamma_N = 0 \\
& \quad \gamma_t \succeq 0, \quad t \in \mathbb{Z}_{0,N}.
\end{align*}
\]

(5B.74)

Here (5B.75) is formed by using that maximizing the concave function \(g(\alpha, \beta, \gamma)\) subject to the constraints in (5B.74) corresponds to minimizing the convex function \(-g(\alpha, \beta, \gamma)\) subject to the same constraints (Boyd and Vandenberghe, 2004). It will now be shown that the dual problem (5B.75) can be interpreted as a CFTOC problem in the form in (3.1) by introducing new variables \(x^d_{\tau} \in \mathbb{R}^{n_x}\) and \(u^d_{\tau} \in \mathbb{R}^{n_{z,N-\tau} + n_{c,N-\tau}}\). The dimensions will be clear from the definitions. Now define \(\tau \triangleq N - t\) and \(N^d \triangleq N + 1\), and let \(x^d_{\tau}\) and \(u^d_{\tau}\) be defined as

\[
\begin{align*}
x^d_0 & \triangleq 0, \\
x^d_{\tau} & \triangleq \alpha_{N+1-\tau}, \quad \tau \in \mathbb{Z}_{1,N+1}, \\
u^d_{\tau} & \triangleq \begin{bmatrix} \beta_{N-\tau} \\ \gamma_{N-\tau} \end{bmatrix}, \quad \tau \in \mathbb{Z}_{0,N}.
\end{align*}
\]

(5B.76)

Furthermore, define the following matrices, vectors and constants (for \(\tau \in \mathbb{Z}_{1,N}\) unless stated otherwise)

\[
\begin{align*}
A_0^d & \triangleq 0, \quad Q_{x,0}^d \triangleq 0, \quad Q_{x,0}^d \triangleq 0, \quad l_{x,0}^d \triangleq 0, \\
A_{\tau}^d & \triangleq (A_{N-\tau}^p)^T, \\
B_{\tau}^d & \triangleq \begin{bmatrix} (M_{N-\tau}^p)^T \\ \left(H_{x,N-\tau}^p\right)^T \end{bmatrix}, \quad \tau \in \mathbb{Z}_{0,N}, \\
Q_{x,\tau}^d & \triangleq B_{N-\tau}^p \bar{Q}_{u,N-\tau} (B_{N-\tau}^p)^T, \\
\end{align*}
\]

(5B.77)
\[ Q^d_{u,0} \triangleq \begin{bmatrix} (Q_{x,N}^p)^{-1} & 0 \\ 0 & 0 \end{bmatrix}, \]  
\[ Q^d_{u,\tau} \triangleq \begin{bmatrix} \bar{Q}_{x,N-\tau} & \bar{Q}_{zu,N-\tau} \\ -H_{u,N-\tau} (Q_{zu,N-\tau}^p)^T & H_{u,N-\tau}^p \end{bmatrix}, \]
\[ Q^d_{xu,\tau} \triangleq \begin{bmatrix} -B_{N-\tau} (Q_{zu,N-\tau}^p)^T & B_{N-\tau}^p \bar{Q}_{u,N-\tau} \\ -H_{u,N-\tau} (Q_{zu,N-\tau}^p)^T & H_{u,N-\tau}^p \end{bmatrix}, \]
\[ l^d_{x,\tau} \triangleq -a_{N-\tau}^p + B_{N-\tau}^p (Q_{zu,N-\tau}^p)^T l^p_{x,N-\tau} + B_{N-\tau}^p \bar{Q}_{u,N-\tau} l^p_{u,N-\tau}, \]
\[ l^d_{u,0} \triangleq \begin{bmatrix} -\left( Q_{x,N}^p \right)^{-1} & 0 \\ 0 & -h_{u,N}^p \end{bmatrix}, \]
\[ l^d_{u,\tau} \triangleq \begin{bmatrix} -\bar{Q}_{x,N-\tau} l^p_{x,N-\tau} - (Q_{zu,N-\tau})^T l^p_{u,N-\tau} \\ -h_{u,N}^p + H_{u,N}^p (Q_{zu,N-\tau})^T l^p_{u,N-\tau} + H_{u,N}^p \bar{Q}_{u,N-\tau} l^p_{u,N-\tau} \end{bmatrix}, \]
\[ l^d_{x,Nd} \triangleq -\bar{x}, \]
\[ c^d_0 \triangleq \frac{1}{2} l^p_{x,N} T \left( Q_{x,N}^p \right)^{-1} l^p_{x,N} - c_N^p, \]
\[ c^d_{\tau} \triangleq \frac{1}{2} \begin{bmatrix} l^p_{x,N-\tau} \\ l^p_{u,N-\tau} \end{bmatrix}^T \begin{bmatrix} \bar{Q}_{x,N-\tau} & \bar{Q}_{zu,N-\tau} \\ Q_{zu,N-\tau} & Q_{u,N-\tau} \end{bmatrix} \begin{bmatrix} l^p_{x,N-\tau} \\ l^p_{u,N-\tau} \end{bmatrix} - c_{N-\tau}^p. \]

Then, the dual problem (5B.75) can equivalently be written

\[
\begin{aligned}
\text{minimize}_{x^d, u^d} & \quad \sum_{\tau=0}^{N_d-1} \left( \frac{1}{2} \begin{bmatrix} x^d_{\tau,0} \\ u^d_{\tau,0} \end{bmatrix}^T \begin{bmatrix} Q^d_{x,\tau} & Q^d_{xu,\tau} \\ (Q^d_{xu,\tau})^T & Q^d_{u,\tau} \end{bmatrix} \begin{bmatrix} x^d_{\tau,0} \\ u^d_{\tau,0} \end{bmatrix} \right) + l^d_{x,Nd}^T x^d_{N_d} \\
\text{subject to} & \quad x^d_0 = 0 \\
& \quad x^d_{\tau+1} = A^d_{\tau} x^d_{\tau} + B^d_{\tau} u^d_{\tau}, \quad \tau \in \mathbb{Z}_{0,N_d-1} \\
& \quad [0 \quad -I_{n_c,N-\tau}] u^d_{\tau} \leq 0, \quad \tau \in \mathbb{Z}_{0,N_d-1}.
\end{aligned}
\]

Using these definitions, the dual problem (5B.78) can be interpreted as a CFTOC problem in the form in (3.1).

### 5.8.2 Relation between the primal and dual variables

The relation between the primal variables \( x^p_i \) and \( u^p_i \), and the solution to the dual problem (5B.78) is investigated in this section. The relation can be used to compute the primal solution given the dual solution. The primal control inputs
can be computed using (5B.71a) giving
\[ u^p_t = - (Q_{zu,t})^T l^p_{z,t} + (Q_{zu,t})^T \beta_t - \bar{Q}_{u,t}^p l^p_{u,t} - \bar{Q}_{u,t} (B^p_t)^T \alpha_{t+1} - \bar{Q}_{u,t} \left( H^p_{u,t} \right)^T \gamma_t, \] (5B.79)
which, by using the definitions of \( x^d_t \) and \( u^d_t \) in (5B.76), is equivalent to
\[ u^p_t = - (Q_{zu,t})^T l^p_{z,t} - \bar{Q}_{u,t}^p l^p_{u,t} - \bar{Q}_{u,t} (B^p_t)^T x^d_{N-t} + \left[ (Q_{zu,t})^T - \bar{Q}_{u,t} \left( H^p_{u,t} \right)^T \right] u^d_{N-t}. \] (5B.80)
Hence, given the dual state variables \( x^d_t \) and dual control inputs \( u^d_t \), the primal control input is computed using (5B.80).

To compute the primal state variables \( x^p_t \), the optimality conditions (2.17a) for the dual problem (5B.78) is used. Let the dual variables to the dual problem be defined as
\[ \lambda_{t+1} \leftrightarrow -x^d_{t+1} + A^d_t x^d_t + B^d_t u^d_t = 0, \quad \tau \in Z_0,N^d-1, \] (5B.81a)
\[ \mu_{\tau} \leftrightarrow [0 - I_{n_c,N-\tau}] u^d_{\tau} \leq 0, \quad \tau \in Z_0,N^d-1. \] (5B.81b)

By using the definitions of \( Q^d_{x,\tau} \), \( Q^d_{zu,\tau} \), \( l^d_{x,\tau} \) and \( A^d_t \) in (5B.77d), (5B.77g), (5B.77h) and (5B.77b), respectively, the optimality condition (2.17a) gives (for \( \tau \in Z_0,N^d-1 \))
\[ Q^d_{x,\tau} x^d_{\tau} + Q^d_{zu,\tau} u^d_{\tau} + l^d_{x,\tau} + \left( A^d_t \right)^T \lambda_{t+1} - \lambda_{\tau} = 0 \iff \lambda_{\tau} = A^p_t \lambda_{t+1} + B^p_t \bar{Q}_{u,t} (B^p_t)^T x^d_t - B^p_t \left[ (Q_{zu,t})^T - \bar{Q}_{u,t} \left( H^p_{u,t} \right)^T \right] u^d_{t} + (5B.82) \]
\[ B^p_t (\bar{Q}_{zu,t})^T l^p_{z,t} + B^p_t \bar{Q}_{u,t} l^p_{u,t} - a^p_t = A^p_t \lambda_{t+1} - B^p_t u^p_t - a^p_t. \]
Here (5B.80) is used in the second equality to simplify the expression. For \( \tau = N^d \) the optimality condition is
\[ l^d_{x,N^d} - \lambda_{N^d} = 0 \iff \lambda_{N^d} = l^d_{x,N^d} = -\bar{x}, \] (5B.83)
where the definition of \( l^d_{x,N^d} \) in (5B.77k) has been used. Hence, from (5B.82) and (5B.83) it follows that if the primal variables \( x^p_t \) are computed as
\[ x^p_t = -\lambda_{N+1-t}, \] (5B.84)
they will satisfy the dynamics equations in the primal problem (5.46). By inserting this choice of the primal variables into (5B.82) and (5B.83) gives
\[ x^p_{t+1} = A^p_t x^p_t + B^p_t u^p_t + a^p_t, \quad t \in Z_0,N-1, \] (5B.85)
and
\[ x^p_0 = \bar{x}. \] (5B.86)
It is trivial to verify that (5B.85) and (5B.86) are the dynamics constraints and the initial condition, respectively, in the primal CFTOC problem (5.46).
Furthermore, using the definitions of $Q^d_{xu,\tau}$, $Q^d_{u,\tau}$, $l^d_{u,\tau}$ and $B^P_t$ in (5B.77g), (5B.77f), (5B.77j) and (5B.77c), respectively, in the optimality condition (2.17a) for the dual problem (5B.78) also gives

$$
\begin{align*}
\left( Q^d_{xu,\tau} \right)^T x^d_{\tau} + Q^d_{u,\tau} u^d_{\tau} + l^d_{u,\tau} + \left[ 0 \right] \mu_{\tau} + \left( B^d_{\tau} \right)^T \lambda_{\tau+1} &= 0 \\
\left[ -\bar{Q}_{zu,t} (B^P_t)^T \right] x^d_{\tau} + \left[ \bar{Q}_{z,t} - \bar{Q}_{zu,t} \left( H^P_{u,t} \right)^T \right] u^d_{\tau} + \\
\left[ -H^P_{u,t} \bar{Q}_{zu,t} \right] x^d_{\tau} + H^P_{u,t} \bar{Q}_{u,t} \left( H^P_{u,t} \right)^T u^d_{\tau} + \\
\left[ 0 \right] \mu_{\tau} + \left[ M^P_t H^P_{x,t} \right] \lambda_{\tau+1} &= 0.
\end{align*}
$$

(5B.87)

The lower block-equations in (5B.87) gives

$$
\begin{align*}
H^P_{u,t} \bar{Q}_{u,t} (B^P_t)^T x^d_{\tau} + H^P_{u,t} \left[ -\left( \bar{Q}_{zu,t} \right)^T \bar{Q}_{u,t} \left( H^P_{u,t} \right)^T \right] u^d_{\tau} + H^P_{u,t} \bar{Q}_{zu,t} l^P_{z,t} + \\
H^P_{u,t} \bar{Q}_{u,t} l^P_{z,t} + h^P_t + H^P_{x,t} \lambda_{\tau+1} - \mu_{\tau} &= 0 \\
H^P_{u,t} \left[ \frac{-\bar{Q}_{zu,t}}{H^P_{u,t} \bar{Q}_{u,t}} \right] u^d_{\tau} + \bar{Q}_{u,t} l^P_{z,t} + \bar{Q}_{u,t} l^P_{u,t} + \bar{Q}_{u,t} (B^P_t)^T x^d_{\tau} &= \mu_{\tau}.
\end{align*}
$$

(5B.88)

By using the expressions for $x^P_t$ and $u^P_t$ in (5B.84) and (5B.80), respectively, together with $\tau = N - t$ in (5B.88) the following relation is obtained

$$
- H^P_{x,t} x^P_t - H^P_{u,t} u^P_t - h_t = \mu_{N-t} \iff H^P_{x,t} x^P_t + H^P_{u,t} u^P_t + h_t = -\mu_{N-t}.
$$

(5B.89)

Hence, the dual variables $\mu_{\tau}$ corresponding to the inequality constraints in the dual problem (5B.78) can be interpreted as the slack variables for the inequality constraints in the primal problem (5.46).
Parallel Computations of Search Directions

Two different ways to exploit parallelism when solving CFTOC problems are either
to parallelize the solver that is applied to solve the CFTOC problem, or to paralle-
lize the computations of the search directions in the solver. Examples where the
parallelism is applied directly to the CFTOC problem can be found in for exam-
ple Chang and Luh (1985), Chang et al. (1990), Chang et al. (1989), and Boyd
et al. (2008). In this thesis, the focus is not on solving the CFTOC problem itself
in parallel, but rather to solve the UFTOC problems which correspond to comput-
ing the second-order search directions within Ip and As type solvers when applied
to solve CFTOC problems. In Wright (1991) the authors argue that exploiting
parallelism like this at the level of numerical linear algebra is more flexible than
exploiting it directly for the CFTOC problem.

In this chapter, different algorithms for solving the UFTOC problems in parallel are
presented. Hence, all the proposed parallel algorithms in this thesis are used at
the level of the numerical linear algebra. The algorithms reduce a UFTOC problem
of prediction horizon $N$ in parallel to a new, smaller master UFTOC problem in the
same form but with prediction horizon $\hat{N} < N$. Unlike for example the partitioned
DP algorithm in Wright (1991), the UFTOC problem structure is retained in the
master problem. Hence, the same parallel algorithm can be applied recursively
without changing the structure of the master problems. Furthermore, in the al-
gorithms presented here the solution is computed directly without involving any
iterative updates of variables, as is usually done in parallel first-order-methods,
such as the alternating direction method of multipliers (Boyd et al., 2011). The
material presented in this chapter is based on the publications Nielsen and Axehill
(2014, 2015), Nielsen (2015), and Nielsen and Axehill (2016b). However, some new
material is added, mainly to Section 6.3, and the presentation of the results from
the listed publications have been edited and combined into a common framework.
The UFTOC problem (3.30) that corresponds to the second-order search directions in many IP and AS solvers when applied to solve CFTOC problems is repeated here for convenience:

\[
\begin{align*}
\text{minimize} & \quad \sum_{t=0}^{N-1} \left( \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} Q_{x,t} & Q_{xu,t} \\ Q_{xu,t}^T & Q_{u,t} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} l_{x,t} \\ l_{u,t} \end{bmatrix}^T \begin{bmatrix} x_t \\ u_t \end{bmatrix} + c_t \right) + \\
& \quad \frac{1}{2} x_N^T Q_{x,N} x_N + l_{x,N}^T x_N + c_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_{t+1} = A_t x_t + B_t u_t + a_t, \quad t \in \mathbb{Z}_{0,N-1}.
\end{align*}
\]  

(6.1)

Let assumptions 3.3 and 3.4 hold, let \( \mathcal{P}(N) \) denote an optimization problem with the same structure as in (6.1), and let the dual variables corresponding to the equality constraint in (6.1) be defined as in (3.32).

In Section 6.1, the main steps of the parallel algorithms are outlined. In Section 6.2 and Section 6.3, the different ways of reducing and solving the UFTOC problem (6.1) are presented. How to recursively reduce the UFTOC problem to obtain \( \mathcal{O}(\log N) \) computational complexity growth is presented in Section 6.4.

### 6.1 Computing the Search Direction via a Master Problem

The original UFTOC problem (6.1) can be reduced in different ways to a master problem in the same UFTOC form but with shorter prediction horizon and possibly fewer control inputs using different techniques. Furthermore, the solution of the original UFTOC problem can be computed from the solution of the master problem in different ways. The reduction and solution of the original UFTOC problem can be described by four main steps:

1. **Split into subproblems:** Split the original UFTOC problem \( \mathcal{P}(N) \) in time into subproblems \( \mathcal{P}_i(N_i) \) for \( i \in \mathbb{Z}_{0,\hat{N}} \).

2. **Reduce the subproblems:** Reduce the subproblem to depend on fewer variables using any of the algorithms presented in Section 6.2 or Section 6.3.

3. **Form and solve the master problem:** Form the master UFTOC problem \( \mathcal{P}(\hat{N}) \) using the reduced subproblems. Solve the master UFTOC problem.

4. **Solve subproblems and retrieve solution:** Solve the subproblems using information from the solution to the master UFTOC problem. Retrieve the solution to the original UFTOC problem \( \mathcal{P}(N) \) using the solutions to the subproblems \( \mathcal{P}_i(N_i) \) for \( i \in \mathbb{Z}_{0,\hat{N}} \).

The structure of the reduction and solution process is presented in Figure 6.1, where the dots represent repetition of the structure. The problem at the lower level is the split original UFTOC problem (6.1) and the problem at the upper level is the master UFTOC problem.
6.2 Parametric Programming Approach

In this section, the parallel algorithm for solving UFTOC problems using the parametric programming approach in Nielsen and Axehill (2014) is presented. This approach extends the original UFTOC problem with extra complicating variables, and splits it into smaller subproblems. In each subproblem (except the last), the final state has to satisfy a terminal constraint which is dependent on the complicating variables. The connections between the subproblems are given by extra coupling constraints in the complicating variables, and the subproblems can be solved parametrically as a function of these variables. The complicating variables and the coupling constraints are chosen such that when using the parametric solutions to the subproblems to eliminate variables, a master problem which is in the UFTOC form is obtained.
This approach is similar to primal decomposition (Lasdon, 1970; Boyd et al., 2008), and to the hierarchical approach presented in Chang (1986) and Chang et al. (1989) in the sense that the problem is decomposed in time into $N + 1$ subproblems that share complicating variables. Here, however, the parallel algorithm is applied at the level of the numerical linear algebra instead of to the inequality constrained problem. Furthermore, the introduction of the complicating variables and constraints is made such that feasible subproblems are always obtained, and the master problem is in the form of a UFTOC problem. Since the master problem is in the same UFTOC form as the original UFTOC problem, the proposed parallel algorithm can be applied recursively without making any modifications to it.

### 6.2.1 Problem decomposition and variable elimination

The structure of the UFTOC problem (6.1) can be exploited by splitting it into smaller subproblems that only share a small number of complicating variables. Given the value of the complicating variables, the subproblems can be solved independently. These smaller subproblems are obtained by splitting the prediction horizon in $N + 1$ intervals $i \in \mathbb{Z}_{0,N}$, where each is of length $N_i \in \mathbb{Z}_{+1}$ such that $\sum_{i=0}^{N} = N$. Let $t_0 = 0$ and $t_i = \sum_{k=0}^{i-1} N_k$ for $i \in \mathbb{Z}_{1,N}$, and introduce the local problem matrices and vectors for $i \in \mathbb{Z}_{0,N}$ and $t \in \mathbb{Z}_{0,N_i-1}$ as

$$
Q_{x,t,i} \triangleq Q_{x,t_i,t+t}, \quad Q_{xu,t,i} \triangleq Q_{xu,t_i,t+t}, \quad Q_{u,t,i} \triangleq Q_{u,t_i+t}, \quad l_{x,t,i} \triangleq l_{x,t_i+t}, \quad \text{(6.2a)}
$$

$$
l_{u,t,i} \triangleq l_{u,t_i+t}, \quad c_{t,i} \triangleq c_{t_i+t}, \quad A_{t,i} \triangleq A_{t_i+t}, \quad B_{t,i} \triangleq B_{t_i+t}, \quad a_{t,i} \triangleq a_{t_i+t}. \quad \text{(6.2b)}
$$

Furthermore, introduce the local variables

$$
x_i \triangleq \begin{bmatrix} x_{0,i} \\ \vdots \\ x_{N_i,i} \end{bmatrix}, \quad u_i \triangleq \begin{bmatrix} u_{0,i} \\ \vdots \\ u_{N_i-1,i} \end{bmatrix},
$$

(6.3)

the initial constraints $x_{0,i} = \hat{x}_i$ for each subproblem, and the terminal constraints $x_{N_i,i} = d_i$ for $i \in \mathbb{Z}_{0,N-1}$. Here, $d_i$ will only be used as a dummy variable in an intermediate step. The connections between the subproblems are introduced as the coupling constraints $\hat{x}_{i+1} = d_i$ for $i \in \mathbb{Z}_{0,N-1}$. Now, let $d_i$ be parametrized as

$$
d_i = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i,
$$

(6.4)

where the parameters $\hat{x}_i \in \mathbb{R}^{n_x}$ and $\hat{u}_i \in \mathbb{R}^{n_{\hat{u},i}}$ with $n_{\hat{u},i} \leq n_x$ are introduced as the complicating variables. How, and also the reason why, to choose the parametrization $\hat{A}_i$, $\hat{B}_i$ and $\hat{a}_i$ will soon be explained.

Now, define the number of variables $n_i$, the number of equality constraints $m_i$ and the number of parameters $p_i$ for $i \in \mathbb{Z}_{0,N-1}$ as

$$
n_i \triangleq (N_i + 1)n_x + \sum_{t=0}^{N_i-1} n_{u,i,t}, \quad m_i \triangleq (N + 2)n_x, \quad p_i \triangleq n_x + n_{\hat{u},i}.
$$

(6.5)
Furthermore, define $H_i \in S_{n_i}$, $l_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}$, $A_i \in \mathbb{R}^{m_i \times n_i}$, $b_i \in \mathbb{R}^{m_i}$, and $G_i \in \mathbb{R}^{m_i \times p_i}$ for $i \in \mathbb{Z}_{0,N-1}$ as

$$H_i \triangleq \begin{bmatrix} Q_{x,0,i} & Q_{xu,0,i} & 0 & \cdots & 0 \\ Q_{xu,0,i}^T & Q_{u,0,i} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & Q_{x,N_i-1,i} & 0 \\ 0 & 0 & \cdots & Q_{xu,N_i-1,i} & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad l_i \triangleq \begin{bmatrix} l_{x,0,i} \\ l_{xu,0,i} \\ \vdots \\ l_{x,N_i-1,i} \\ l_{u,N_i-1,i} \\ 0 \end{bmatrix},$$

$$(6.6a)$$

$$c_i \triangleq \sum_{t=0}^{N_i-1} c_{t,i}, \quad A_i \triangleq \begin{bmatrix} 0 \\ -a_{0,i} \\ \vdots \\ -a_{N_i-1,i} \\ -\hat{a}_i \end{bmatrix}, \quad G_i \triangleq \begin{bmatrix} -I \\ 0 \end{bmatrix}, \quad X_i \triangleq \begin{bmatrix} x_{0,i} \\ u_{0,i} \\ \vdots \\ x_{N_i-1,i} \\ u_{N_i-1,i} \end{bmatrix}.$$  

$$(6.6b)$$

$$b_i \triangleq \begin{bmatrix} 0 \\ -a_{0,i} \\ \vdots \\ -a_{N_i-1,i} \end{bmatrix}, \quad X_i \triangleq \begin{bmatrix} -I \\ 0 \end{bmatrix}, \quad X_i \triangleq \begin{bmatrix} x_{0,i} \\ u_{0,i} \\ \vdots \\ x_{N_i-1,i} \end{bmatrix}.$$  

$$(6.6c)$$

For $i = \hat{N}$, the matrices are defined similarly as in (6.6) but including the cost for $x_{N_i, \hat{N}}$ in $H_{\hat{N}}$, $l_{\hat{N}}$ and $c_{\hat{N}}$, removing the last block row in $A_{\hat{N}}$, $b_{\hat{N}}$, and $G_{\hat{N}}$, and removing the last block column in $G_{\hat{N}}$ since no terminal constraint is introduced in this subproblem.

Then, by using the definitions in (6.6) together with the coupling constraints and the complicating variables, an extended optimization problem can be constructed from (6.1), given in the form

$$\text{minimize} \quad \sum_{i=0}^{\hat{N}} \left( \frac{1}{2} x_{i}^T H_i X_i + l_i^T X_i + c_i \right)$$

subject to

$$A_i X_i = b_i + G_i \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix}, \quad i \in \mathbb{Z}_{0,\hat{N}-1}$$

$$A_{\hat{N}} X_{\hat{N}} = b_{\hat{N}} + G_{\hat{N}} \hat{x}_{\hat{N}}$$

$$\hat{x}_0 = \bar{x}$$

$$\hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i, \quad i \in \mathbb{Z}_{0,\hat{N}-1}.$$  

$$(6.7)$$

The term extended is used to indicate that the UFTOC problem (6.1) is extended with both extra variables and constraints to form (6.7).
The extended problem (6.7) can be solved by first optimizing over the local variables $X_i$ while considering the complicating variables as parameters. By defining

$$\theta_i \triangleq \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix}, \quad i \in \mathbb{Z}_{0,N-1}, \quad \theta_N \triangleq \hat{x}_N,$$

$X_i$ for $i \in \mathbb{Z}_{0,N}$ can be computed as a function of the parameter $\theta_i$ by solving the equality constrained mp-qp problem

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} X_i^T H_i X_i + l_i^T X_i + c_i \\
\text{subject to} & \quad A_i X_i = b_i + G_i \theta_i.
\end{align*}$$

(6.9)

Lemma 6.1. The reduced Hessian of (6.9) is positive definite and hence satisfies $V^T H_i V \succ 0$, where the columns of $V$ form a basis for $\mathcal{N}(A_i)$.

Proof: Consider a subproblem $i \in \mathbb{Z}_{0,N-1}$. Let $\bar{A}_i$, $\bar{b}_i$ and $\bar{G}_i$ denote all the rows in $A_i$, $b_i$ and $G_i$ except the last block row. Now consider a problem as in (6.9), but with the equality constraints $\bar{A}_i X_i = \bar{b}_i + \bar{G}_i \theta_i$ instead. This problem is a uftoc problem in the form (3.30) where assumptions 3.3 and 3.4 hold, which has a unique solution (Maciejowski, 2002). Hence, the reduced Hessian $\bar{V}^T H_i \bar{V}$ is positive definite, where the columns of $\bar{V}$ form a basis for $\mathcal{N}(\bar{A}_i)$ (Boyd and Vandenberghe, 2004). Since $\mathcal{N}(A_i) \subset \mathcal{N}(\bar{A}_i)$, the result follows. For $i = N$ the result follows immediately since it is in the uftoc form where assumptions 3.3 and 3.4 hold. \qed

The subproblem (6.9) is only feasible when $b_i + G_i \theta_i \in \mathcal{R}(A_i)$ (Lancaster and Tismenetsky, 1985). Hence, depending on the parametrization and the subproblem, the introduction of the terminal constraints might result in infeasible subproblems. In order to obtain feasible subproblems in (6.7) for every choice of the parameters, the parametrizations of the terminal constraints must be chosen such that it is always possible to reach the desired final state from the initial state of the subproblem. The final state can be computed using the dynamics constraints as

$$x_{N,i} = \bar{A}_i \hat{x}_i + \bar{B}_i u_i + \bar{D}_i a_i,$$

(6.10)

where

$$\begin{align*}
\bar{A}_i & \triangleq \prod_{t=0}^{N_i-1} A_{t,i}, & \bar{D}_i & \triangleq \begin{bmatrix} \prod_{t=1}^{N_i-1} A_{t,i} & \cdots & A_{N_i-1,i} & I \end{bmatrix}, \\
\bar{B}_i & \triangleq \begin{bmatrix} \prod_{t=1}^{N_i-1} A_{t,i} B_{0,i} & \cdots & A_{N_i-1,i} B_{N_i-2,i} & B_{N_i-1,i} \end{bmatrix},
\end{align*}$$

(6.11a)

and $a_i$ and $u_i$ are the stacked $a_{t,i}$ and $u_{t,i}$ for $t \in \mathbb{Z}_{0,N_i-1}$, respectively. Note that $\bar{B}_i$ is the reachability matrix (Rugh, 1996). The product operator is introduced in Definition 6.2 given below.
**Definition 6.2 (Product operator).** The product operator for a sequence of matrices $A_l, A_{l+1}, \ldots, A_k$ is defined as

\[
\prod_{t=l}^{k} A_t \triangleq \begin{cases} 
A_k \cdots A_{l+1} A_l, & l \leq k, \\
I, & l > k.
\end{cases} \tag{6.12}
\]

The feasibility of a subproblem can be managed by parametrizing the problem such that the parameter $d_i = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i$ is reachable from $\hat{x}_i$. This can be assured by choosing $\hat{A}_i \triangleq \tilde{A}_i$, $\hat{B}_i \triangleq \mathcal{T}_i$, $\hat{a}_i \triangleq \tilde{D}_i a_i$, (6.13)

where the columns of $\mathcal{T}_i$ form a basis for $\mathcal{R} (\tilde{B}_i)$. With this choice of parametrization, it is always possible to satisfy the terminal constraint for any choice of parameters $\hat{x}_i$ and $\hat{u}_i$. Note that by introducing this parametrization of the subproblem, the problem of choosing parameters in the master problem that satisfy the range constraints

\[
b_i + G_i \theta_i \in \mathcal{R} (A_i) \quad \text{for} \quad i \in \mathbb{Z}_{0,N-1}
\]

to give feasible subproblems is solved already in the subproblems. For a subproblem where $\tilde{B}_i$ has full row rank, $\hat{A}_i = 0$, $\hat{B}_i = I$ and $\hat{a}_i = 0$ in the parametrization of the terminal constraint are valid choices for the parametrization.

**Remark 6.3.** The terminal state $x_{N_i,i}$ in the subproblems $i \in \mathbb{Z}_{0,N-1}$ can be discarded in practice, and the two constraints

\[
\begin{align*}
x_{N_i,i} &= A_{N_i-1,i} x_{N_i-1,i} + B_{N_i-1,i} u_{N_i-1,i} + a_{N_i-1,i}, \\
x_{N_i,i} &= \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i,
\end{align*} \tag{6.14}
\]

can be combined into a single constraint

\[
\hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i = A_{N_i-1,i} x_{N_i-1,i} + B_{N_i-1,i} u_{N_i-1,i} + a_{N_i-1,i}, \tag{6.15}
\]

and the corresponding parts in the objective function can be discarded. However, for notational reasons the terminal state $x_{N_i,i}$ is used as a local variable in the subproblems.

Now, introduce the dual variables $\Lambda_i$ corresponding to the equality constraints in (6.9) as

\[
\Lambda_i \triangleq \begin{bmatrix} 
\lambda_{0,i} \\
\lambda_{1,i} \\
\vdots \\
\lambda_{N_i,i} \\
\lambda_{tc,i}
\end{bmatrix} \tag{6.16}
\]

where

\[
\begin{align*}
\lambda_{0,i} &\leftrightarrow -x_{0,i} + \hat{x}_i = 0, \tag{6.17a} \\
\lambda_{t+1,i} &\leftrightarrow -x_{t+1,i} + A_{t,i} x_{t,i} + B_{t,i} u_{t,i} + a_{t,i} = 0, \quad t \in \mathbb{Z}_{0,N_i-1}, \tag{6.17b} \\
\lambda_{tc,i} &\leftrightarrow -x_{N_i,i} + \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i = 0. \tag{6.17c}
\end{align*}
\]
The use of the notation ↔ is introduced in Definition 3.5. The subscript “tc” is an abbreviation of terminal constraint.

The subproblem (6.9) is a very simple mp-QP problem with parameters \( \theta_i \) and only equality constraints. Hence the optimal primal and dual solutions to this problem are both affine functions of the parameters \( \theta_i \) and given by

\[
X^*_i(\theta_i) = K^x_i \theta_i + k^x_i, \\
\Lambda^*_i(\theta_i) = K^\lambda_i \theta_i + k^\lambda_i + \lambda_i^N, \tag{6.18a}
\]

for some \( K^x_i \in \mathbb{R}^{n_i \times p_i}, k^x_i \in \mathbb{R}^{n_i}, K^\lambda_i \in \mathbb{R}^{m_i \times p_i}, k^\lambda_i \in \mathbb{R}^{m_i} \) and \( \lambda_i^N \in \mathcal{N}(A^T_i) \). Here \( i \) denotes the index of the subproblem. See for example Tøndel et al. (2003a) for the details on how compute the solution. The primal solution is unique since the reduced Hessian is positive definite according to Lemma 6.1. When LICQ holds for the subproblem (6.9) it follows that \( \mathcal{N}(A^T_i) = \{0\} \), and hence also the dual solution is unique (Tøndel et al., 2003a). Since the simple mp-QP problem (6.9) is subject to equality constraints only, it is affine, rather than piecewise affine, in the parameters. Hence, the solution to this equality constrained mp-QP problem can be computed cheaply compared to the cost of solving an inequality constrained mp-QP problem, and it does not suffer from the complexity issues of a general mp-QP problem.

The local variables \( X_i \) can be eliminated from the extended problem (6.7) by using the parametric solution (6.18a). The value function of (6.9) can be expressed in the parameter \( \theta_i \) by inserting the parametric primal optimal solution (6.18a) into the objective function in (6.9), resulting in

\[
\hat{V}_i(\theta_i) \triangleq \frac{1}{2} \theta_i^T \hat{Q}_i \theta_i + \hat{l}_i^T \theta_i + \hat{c}_i, \tag{6.19}
\]

where

\[
\hat{Q}_i \triangleq (K^x_i)^T H_i K^x_i \in \mathbb{S}^n_{++}, \tag{6.20a}
\]

\[
\hat{l}_i \triangleq (K^x_i)^T l_i + (K^\lambda_i)^T H_i k^x_i \in \mathbb{R}^{n_x + n_{\hat{a}},i}, \tag{6.20b}
\]

\[
\hat{c}_i \triangleq c_i + \frac{1}{2} (k^x_i)^T H_i k^x_i + l_i^T k^x_i \in \mathbb{R}. \tag{6.20c}
\]

Lemma 6.4. Let the relation between \( \theta_i \) and \( \hat{x}_i, \hat{u}_i \) be given by (6.8), and let \( \hat{Q}_i \) in (6.20a) be partitioned into blocks corresponding to the size of \( \hat{x}_i \) and \( \hat{u}_i \) as

\[
\begin{bmatrix}
\hat{Q}_{x,i} & \hat{Q}_{xu,i} \\
\hat{Q}_{ux,i}^T & \hat{Q}_{u,i}
\end{bmatrix} \triangleq \hat{Q}_i. \tag{6.21}
\]

Then it holds that \( \hat{Q}_{u,i} \in \mathbb{S}^n_{++} \) for \( i \in \mathbb{Z}_{0,N-1} \).

**Proof:** The proof of Lemma 6.4 is presented in Appendix 6.B.1. \( \square \)
### 6.2.2 Constructing the master problem

By eliminating all the local variables \( X_i \) for \( i \in \mathbb{Z}_{0,N} \) in (6.7), a UFTOC problem in the form (6.1) but with horizon \( N < N \) is obtained, which is presented in Theorem 6.5.

**Theorem 6.5.** Consider a UFTOC problem \( P(N) \) as defined in (6.1). Then, this UFTOC problem can be reduced in parallel to a smaller UFTOC problem \( P(\hat{N}) \) with \( \hat{N} < N \), given by

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=0}^{\hat{N}-1} \left( \frac{1}{2} \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix}^T \begin{bmatrix} Q_{x,i} & Q_{xu,i} \\ Q_{xu,i}^T & Q_{u,i} \end{bmatrix} \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix} + \begin{bmatrix} \hat{l}_{x,i} \\ \hat{l}_{u,i} \end{bmatrix}^T \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix} + \hat{c}_i \right) + \frac{1}{2} \hat{x}_{\hat{N}}^T \hat{Q}_{\hat{N}} \hat{x}_{\hat{N}} + \hat{l}_{\hat{N}}^T \hat{x}_{\hat{N}} + \hat{c}_{\hat{N}} \\
\text{subject to} & \quad \hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i, \quad i \in \mathbb{Z}_{0,\hat{N}-1},
\end{align*}
\]  

where \( \hat{A}_i \in \mathbb{R}^{nx \times nx}, \hat{B}_i \in \mathbb{R}^{nx \times nu,i} \) and \( \hat{a}_i \in \mathbb{R}^{nx} \) are defined in (6.13), \( \hat{Q}_i \in S^{nx+nu,i} \), \( \hat{Q}_{u,i} \in S^{nu,i} \), \( \hat{l}_i \in \mathbb{R}^{nx+nu,i} \) and \( \hat{c}_i \in \mathbb{R} \) are defined in (6.20), and \( nu,i \leq nx \).

**Proof:** Given a UFTOC problem (6.1), the construction of the extended problem (6.7) can be done as described in Section 6.2.1. All local variables can be eliminated by solving the subproblems (6.9) parametrically as functions of the complicating variables, and substituting the parametric solution into the extended problem (6.7). Then, using the definitions in (6.20) it follows that the extended problem (6.7) is reduced to (6.22). \( \hat{A}_i, \hat{B}_i \) and \( \hat{a}_i \) are chosen according to (6.13), and the parametrization is introduced such that \( nu,i \leq nx \). Positive semidefiniteness of \( \hat{Q} \) follows by construction, and \( \hat{Q}_{u,i} \in S^{nu,i} \) follows from Lemma 6.4. Furthermore, since all subproblems can be solved parametrically independently of each other, it is possible to reduce the UFTOC problem (6.1) to (6.22) in parallel, which concludes the proof.

Let the dual variables for the master UFTOC problem in (6.22) be defined as

\[
\begin{align*}
\hat{\lambda}_0 & \leftrightarrow -\hat{x}_0 + \bar{x} = 0, \quad (6.23a) \\
\hat{\lambda}_{i+1} & \leftrightarrow -\hat{x}_{i+1} + \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i = 0, \quad i \in \mathbb{Z}_{0,\hat{N}-1}. \quad (6.23b)
\end{align*}
\]

Then, the primal and dual solution given by \( \hat{x}_i^* \) for \( i \in \mathbb{Z}_{0,\hat{N}} \), \( \hat{u}_i^* \) for \( i \in \mathbb{Z}_{0,\hat{N}-1} \) and \( \hat{\lambda}_i^* \) for \( i \in \mathbb{Z}_{0,\hat{N}} \) is unique since LICQ and assumptions 3.3 and 3.4 hold for the master problem. The solution can be computed using any method that is applicable to UFTOC problems in the form (6.22).
6.2.3 Computing the solution

Once the solution to the master problem \(6.22\) is computed, the solutions to the subproblems \(6.9\) can be computed using these optimal values of the parameters \(\theta_i\). The solutions to the subproblems can be computed using \((6.18)\), where \(X_i^*\) is unique since the reduced Hessian of \((6.9)\) is positive definite (Tøndel et al., 2003a). Furthermore, when LICQ holds, meaning that \(A_i\) has full row rank, also the dual solution is unique since \(\mathcal{N}(A_i^T) = \{0\}\) and thus \(A_i^T \lambda_i^N = 0 \iff \lambda_i^N = 0\) holds. Hence, in the case where LICQ holds, the primal and dual solution can be computed uniquely from the optimal parameters \(\theta_i^*\) as

\[
X_i^* = K_i^T \theta_i^* + k_i^x, \quad (6.24a)
\]

\[
\Lambda_i^* = K_i^\lambda \theta_i^* + k_i^\lambda. \quad (6.24b)
\]

However, the terminal constraint in a subproblem introduces \(n_x\) new constraints, which might result in violation of LICQ for the subproblem even though this is not the case in the original UFTOC problem \((6.1)\). According to Definition 2.13, violation of LICQ is known as primal degeneracy and the dual variables \((6.18b)\) for a primal degenerate problem are non-unique (Tøndel et al., 2003a). Here it will be shown how to choose dual variables in the subproblems that satisfy the KKT optimality conditions of the original UFTOC problem \((6.1)\), even in the case with non-unique dual variables in the subproblems.

Note that if the subproblem violates LICQ when \(x_{N_i,i}\) is included as a local variable in \((6.9)\), it violates LICQ even when \(x_{N_i,i}\) is excluded as was mentioned in Remark 6.3 and the constraints are combined as in \((6.15)\).

In the case when LICQ is violated, the dual variables are computed as in \((6.18b)\) with \(\lambda_i^N \in \mathcal{N}(A_i^T)\), where the nullspace of \(A_i\) is given in Lemma 6.6.

**Lemma 6.6.** The nullspace of \(A_i^T\) is given by

\[
\mathcal{N}(A_i^T) \triangleq \left\{ z \in \mathbb{R}^{m_i} \mid z = Z_i w_i, \text{ with } w_i \in \mathcal{N}\left(\tilde{B}_i^T\right) \right\}, \quad (6.25)
\]

where

\[
Z_i \triangleq \begin{bmatrix}
-\tilde{A}_i^T \\
-\tilde{D}_i^T \\
I
\end{bmatrix}, \quad (6.26)
\]

and \(\tilde{A}_i\) and \(\tilde{D}_i\) are defined in \((6.11a)\), and \(\tilde{B}_i\) is the reachability matrix defined in \((6.11b)\).

**Proof:** For the proof of Lemma 6.6, see Appendix 6.B.3.

Note that \(Z_i\) is computed cheaply since the matrices \(\tilde{A}_i\) and \(\tilde{D}_i\) are already computed when they are used in the computations of \(\hat{A}_i\) and \(\hat{a}_i\) defined in \((6.13)\).

How to compute the solution of the original UFTOC problem from the solutions of the subproblems \(X_i^*\) and \(\Lambda_i^*\) for \(i \in \mathbb{Z}_{0,N}\) is presented in Theorem 6.7.
Theorem 6.7. Let assumptions 3.3 and 3.4 hold for the \( \text{uftoc} \) problem in (6.1), and let the corresponding extended problem (6.7) be constructed as described in Section 6.2.1. Furthermore, let \( \hat{x}^*, \hat{u}^* \) and \( \hat{\lambda}^* \) be the solution to (6.22). Then, the primal solution to (6.1) can be computed as

\[
x^* = \begin{bmatrix}
x_0^* \\
x_{N_i - 1 + t_i}^* \\
\vdots \\
x_N^*
\end{bmatrix}, \quad u^* = \begin{bmatrix}
u_0^* \\
u_{t_i + N_i - 1}^* \\
\vdots \\
u_{N - 1}^*
\end{bmatrix},
\]

where the primal solutions to the subproblems are computed as in (6.24a) for \( i \in \mathbb{Z}_{0, \hat{N}} \). Furthermore, the dual solution to (6.1) can be computed as

\[
\lambda^* = \begin{bmatrix}
\lambda_0^* \\
\lambda_{t_i + N_i - 1}^* \\
\vdots \\
\lambda_{\hat{N}}^*
\end{bmatrix},
\]

where the dual solutions to the subproblems in (6.7) where LICQ holds are computed as in (6.24b). For the subproblems where LICQ does not hold the dual solutions are computed as

\[
\Lambda_i^* = K^\lambda_i \theta_i^* + k^\lambda_i - Z_i \left( \xi_{tc,i}^* + \hat{\lambda}_{i+1}^* \right),
\]

where \( \theta_i \) is defined as in (6.8), \( Z_i \) is given in Lemma 6.6, and

\[
\xi_{tc,i}^* \triangleq \begin{bmatrix}
0 & \cdots & 0
\end{bmatrix} \begin{bmatrix}
K^\lambda_i \hat{\lambda}_i^* \\
\hat{u}_i^*
\end{bmatrix} + k^\lambda_i.
\]

**Proof:** The proof of Theorem 6.7 is given in Appendix 6.B.2. \( \square \)

According to Theorem 6.7, the solution to the original \( \text{uftoc} \) problem (6.1) can be computed from the parametric solution to each subproblem (6.9) and the solution to the master problem (6.22). Since all subproblems can be solved independently of each other when \( \hat{x}^*, \hat{u}^* \) and \( \hat{\lambda}^* \) are given, it is possible to compute the solution to the \( \hat{N} + 1 \) subproblems, and hence also to the original \( \text{uftoc} \) problem, in parallel.

### 6.2.4 Communication

When solving the original \( \text{uftoc} \) problem (6.1) in parallel on different computational units, the value functions \( \hat{V}_i(\theta_i) \) and \( \hat{A}_i, \hat{B}_i \) and \( \hat{\alpha}_i \) from each subproblem must be communicated between the computational units when the master problem is
Table 6.1: The amount of data that is sent for each subproblem to form the master UFTOC problem. Here, the number of real numbers is presented, not the total number of bytes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{Q}<em>{x,i} \in \mathbb{S}</em>{++}^{n_x}$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{u,i} \in \mathbb{S}</em>{++}^{n_{\hat{u},i}}$</td>
<td>$(n_{\hat{u},i}^2 + n_{\hat{u},i}) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{xu,i} \in \mathbb{R}^{n_x \times n</em>{\hat{u},i}}$</td>
<td>$n_x n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{l}_{x,i} \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{l}<em>{u,i} \in \mathbb{R}^{n</em>{\hat{u},i}}$</td>
<td>$n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{c}_i \in \mathbb{R}$</td>
<td>1</td>
</tr>
<tr>
<td>$\hat{A}_i \in \mathbb{R}^{n_x \times n_x}$</td>
<td>$n_x^2$</td>
</tr>
<tr>
<td>$\hat{B}<em>i \in \mathbb{R}^{n_x \times n</em>{\hat{u},i}}$</td>
<td>$n_x n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{a}_i \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>Total</td>
<td>$3/2 n_x^2 + 5/2 n_x + 1/2 n_{\hat{u},i}^2 + 3/2 n_{\hat{u},i} + 2 n_x n_{\hat{u},i} + 1$</td>
</tr>
</tbody>
</table>

formed. The amount of data that is needed to communicate for each subproblem is presented in Table 6.1, where symmetry of the matrices is taken into account.

In order to compute the solution to each subproblem, the solution to the master problem must be communicated to the computational units where the subproblem is to be solved. Depending on if the subproblem is primal degenerate or not, the dual solution $\hat{\lambda}_{i+1}^*$ to the master problem might, or might not be needed in order to compute the dual solution to the subproblem. Hence, $\hat{\lambda}_{i+1}^*$ is only sent to subproblems $i$ where LICQ does not hold. The amount of data that needs to be sent is presented in Table 6.2.

### 6.3 Parallel Partial Condensing and Reduction Approach

In Axehill (2015) partial condensing was introduced as a way to lower the computation time for solving MPC problems by enabling condensing strategies that are in between the sparse MPC formulation and the fully condensed MPC formulation. As was remarked in that work, the ideas from partial condensing can also be used in a parallel setting, where for example the condensing of each block can be performed in parallel and the sizes of the blocks determine the corresponding workloads. In this section it will be shown how to exploit, and extend, the ideas from partial condensing in order to reduce a UFTOC problem in parallel to a smaller master problem in the same UFTOC form but with shorter prediction horizon and possibly fewer control inputs.
It will also be shown how the parallel Riccati recursion algorithm presented in Nielsen and Axehill (2015) and Nielsen (2015) can be included in this framework. Here, the Riccati recursion is computed in parallel in a similar way as is done in partitioned DP (Wright, 1991), but in the algorithm proposed in this thesis the reduced master problem remains in the UFTOC form, which facilitates the recursive solution of smaller UFTOC problems. It will also be shown how to compute the Riccati factorization in Algorithm 4 in parallel, which is not part of the work presented in Wright (1991). The classical serial Riccati recursion that was introduced in Section 3.4 exploits the causality of the problem. For that reason it is not obvious that it can be split and parallelized in time, especially without involving some form of iterative consensus step. In this chapter it is shown how this can be done.

The reduction from the original UFTOC problem $\mathcal{P}(N)$ to the master UFTOC problem $\mathcal{P}(\hat{N})$ will be done by first splitting the UFTOC problem into several smaller subproblems, and then performing two main steps:

- **Condense subproblem**: The condensation of a subproblem is the process of eliminating states using the corresponding dynamics constraints. This is similar to what is done when re-writing the UFTOC problem into its condensed QP form as is shown in Section 3.1 and in for example Maciejowski (2002), Jerez et al. (2012), and Axehill (2015). The different condensing techniques that are used here are presented in Section 6.3.2.

- **Reduce subproblem**: The reduction of a subproblem is the process of reducing the size of the total control input dimension over the horizon in the subproblem. Depending on the definition of the original UFTOC problem and the subproblem, it is not always possible to reduce the control input dimension. It is described in Section 6.3.3 when, and how, this can be done.

In this section, it will be described how to perform these steps in parallel, but first the decomposition of the UFTOC problem $\mathcal{P}(N)$ into subproblems is presented.

### Table 6.2: The amount of data that is communicated to each subproblem to compute the solution. Note that $\hat{\lambda}_{i+1}^*$ is only sent to subproblems $i$ where LICQ do not hold.

<table>
<thead>
<tr>
<th>Variable</th>
<th>LICQ holds</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_i^* \in \mathbb{R}^{n_x}$</td>
<td>Yes/no</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{u}<em>i^* \in \mathbb{R}^{n</em>{\hat{u},i}}$</td>
<td>Yes/no</td>
<td>$n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{\lambda}_{i+1}^* \in \mathbb{R}^{n_x}$</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>$n_x$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>Yes</td>
<td>$n_x + n_{\hat{u},i}$</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>$2n_x + n_{\hat{u},i}$</td>
</tr>
</tbody>
</table>
6.3.1 Decomposition into subproblems

The decomposition of the original UFTOC problem (6.1) can be done similarly as in Section 6.2. The prediction horizon $N$ in the UFTOC problem (6.1) is split into $\hat{N} + 1$ parts of length $N_i \in \mathbb{Z}_{++}$ for $i \in \mathbb{Z}_{0,\hat{N}}$, such that $\sum_{i=0}^{\hat{N}} N_i = N$. Let $t_0 = 0$ and $t_{\hat{N}+1} = N$, and let $t_i = \sum_{k=0}^{i-1} N_k$ for $i \in \mathbb{Z}_{1,\hat{N}}$ be the time indices where the prediction horizon is split. Furthermore, for presentational brevity, let $\hat{x}_i \triangleq x_{t_i}$ denote the state at time $t_i$ and define the local variables $x_i$ and $u_i$ for $i \in \mathbb{Z}_{0,\hat{N}}$ as

$$
x_i \triangleq \begin{bmatrix} x_{t_i+1} \\ x_{t_i+2} \\ \vdots \\ x_{t_i+N_i-1} \end{bmatrix}, \quad u_i \triangleq \begin{bmatrix} u_{t_i} \\ u_{t_i+1} \\ \vdots \\ u_{t_i+N_i-1} \end{bmatrix}. \tag{6.31}
$$

Note that the indexing starts at different values for $x_i$ and $u_i$. Then, the UFTOC problem (6.1) can be re-written in the equivalent form

$$
\min_{\hat{x}, x, u} \sum_{i=0}^{\hat{N}} \left( \frac{1}{2} \begin{bmatrix} \hat{x}_i \\ x_i \\ u_i \end{bmatrix}^T \begin{bmatrix} Q_{x,t_i} & 0 & Q_{zu,i} \\ 0 & Q_{x,i} & Q_{zu,i} \\ Q_{tu,i}^T & Q_{tu,i} & Q_{u,i} \end{bmatrix} \begin{bmatrix} \hat{x}_i \\ x_i \\ u_i \end{bmatrix} + \begin{bmatrix} l_{x,t_i} \\ l_{x,i} \\ l_{u,i} \end{bmatrix}^T \begin{bmatrix} \hat{x}_i \\ x_i \\ u_i \end{bmatrix} + \bar{c}_i \right) + \frac{1}{2} \hat{x}_{N+1}^T Q_{x,N} \hat{x}_{N+1} + \hat{D}_{x,N}^T \hat{x}_{N+1} + c_N
$$

subject to \[ \hat{x}_0 = \bar{x} \]

$$
A_i x_i = A_{0,i} \hat{x}_i + B_i u_i + a_i, \quad i \in \mathbb{Z}_{0,\hat{N}}
$$

$$
\hat{x}_{i+1} = A_{N,i} x_i + B_{N,i} u_i + a_{N,i}, \quad i \in \mathbb{Z}_{0,\hat{N}}, \tag{6.32}
$$

where the matrices are defined as in (6A.83) in Appendix 6.A.1. Note that from the definition of $A_i$ it follows that it is an invertible matrix. The notation $\hat{x}_i \triangleq x_{t_i}$ for $i \in \mathbb{Z}_{0,\hat{N}+1}$ is introduced to simplify the notation. Hence, no extra variables or constraints have been added to the original UFTOC problem (6.1), as was done in Section 6.2, to form the equivalent (6.32).

In this section, instead of introducing complicating variables and constraints as in Section 6.2, the original UFTOC problem will be decomposed into several sub-problems using the cost-to-go function used in dynamic programming, see for example Bertsekas (2000). To do this, let $J(x_t)$ denote the cost-to-go function at state $x_t$ for the UFTOC problem (6.1), which in the case of a quadratic objective function is defined by

$$
J(x_t) \triangleq x_t^T P_t x_t - \Psi_t^T x_t + \gamma_t, \tag{6.33}
$$

with $P_t \in \mathbb{S}^{n_x}_{++}$, $\Psi_t \in \mathbb{R}^{nx}$ and $\gamma_t \in \mathbb{R}$. Furthermore, let $\bar{x}_i \triangleq x_{t_i}^*$ where $x_{t_i}^*$ is obtained from the optimal solution to (6.1). Assume, for the moment, that $\bar{x}_i$ and $J(\hat{x}_{i+1})$ are known for some $i \in \mathbb{Z}_{0,\hat{N}}$. Then it is possible to exploit the principle of optimality (Bertsekas, 2000) and the structure of the equality constraints in the UFTOC problem to compute the optimal solution in the interval $t_i \leq t \leq t_{i+1}$. 
This is done by solving the smaller UFTOC problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \left[ \begin{array}{c} \hat{x}_i \\ x_i \\ u_i \end{array} \right]^T \left[ \begin{array}{ccc} Q_{x,t_i} & 0 & Q_{xu,i} \\ 0 & Q_{x,i} & Q_{xu,i} \\ Q_{xu,i}^T & Q_{xu,i} & Q_{u,i} \end{array} \right] \left[ \begin{array}{c} \hat{x}_i \\ x_i \\ u_i \end{array} \right] + \left[ \begin{array}{c} l_{x,t_i} \\ l_{x,i} \\ l_{u,i} \end{array} \right]^T \left[ \begin{array}{c} \hat{x}_i \\ x_i \\ u_i \end{array} \right] + \bar{c}_i \\
\text{subject to} & \quad \hat{x}_i = \bar{x}_i \\
& \quad A_i x_i = A_{0,i} \hat{x}_i + B_i u_i + a_i \\
& \quad \hat{x}_{i+1} = A_N i x_i + B_N i u_i + a_{N,i}.
\end{align*}
\]

(6.34)

Here, the notation \( \hat{P}_{i+1} \triangleq P_{t_{i+1}}, \Psi_{t_{i+1}} \triangleq \Psi_{t_{i+1}} \) and \( \bar{c}_{i+1} \triangleq \gamma_{t_{i+1}} \) is used for brevity. Hence, if \( \bar{x}_i \) and \( J(\hat{x}_{i+1}) \) are known for all \( i \in \mathbb{Z}_{0,\hat{N}} \), it is possible to solve the original UFTOC problem (6.1) by computing the solution in each interval \( t_i \leq t \leq t_{i+1} \), which is done by solving the \( \hat{N} + 1 \) corresponding subproblems in the form (6.34).

**Remark 6.8.** For the last subproblem with \( i = \hat{N} \), the cost-to-go function is known and defined as \( J(\hat{x}_{\hat{N}+1}) = \frac{1}{2} \hat{x}_{\hat{N}+1}^T Q_{x,N} \hat{x}_{\hat{N}+1} + l_{x,N}^T \hat{x}_{\hat{N}+1} + c_N \).

### 6.3.2 Condensing a subproblem

The optimal solutions \( \bar{x}_i \) and the cost-to-go functions \( J(\hat{x}_{i+1}) \) are not available prior to solving the problem (except for \( J(\hat{x}_{\hat{N}+1}) \) in the last subproblem \( i = \hat{N} \) as mentioned in Remark 6.8). However, it is possible to eliminate local variables from the subproblem (6.34), which is here referred to as condensing the subproblem. Here, it will be shown how to condense the subproblem in two different, but conceptually similar, ways. The first condensing technique is the one which is used in for example Maciejowski (2002), Jerez et al. (2012), and Axehill (2015) and will be referred to as regular condensing, whereas the other is based on the ideas from Nielsen and Axehill (2015) where the Riccati recursion is used to obtain a condensed subproblem that can facilitate both efficient communication and efficient computations.

**Regular condensation of a subproblem**

In regular condensing, the dynamics constraints in the UFTOC problem (6.1) are used to eliminate state variables from the problem, and describing them as a function of the initial state and the control inputs over the full horizon of the subproblem. In the regular condensing technique used here, also the final state is kept as a variable in the condensed subproblem. The details of the regular condensing of the subproblem (6.34) are given in Lemma 6.9 below.
Lemma 6.9 (Regular condensing). Consider an equality constrained QP problem in the form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} x_0^T & \mathbf{u}^T \end{bmatrix} \begin{bmatrix} Q_0 & 0 & R \\ 0 & Q_x & S \\ R^T & S^T & Q_u \end{bmatrix} \begin{bmatrix} x_0 \\ \mathbf{u} \end{bmatrix} + \begin{bmatrix} f_0^T \\ f_x^T \\ f_u^T \end{bmatrix} \begin{bmatrix} x_0 \\ \mathbf{u} \end{bmatrix} + c \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad \Omega \mathbf{x} = \Omega_0 x_0 + \Gamma \mathbf{u} + \gamma \\
& \quad x_N = \Omega_N x + \Gamma_N \mathbf{u} + \gamma_N,
\end{align*}
\]

where \( \Omega \) is invertible and

\[
\begin{bmatrix} Q_0 & 0 & R \\ 0 & Q_x & S \\ R^T & S^T & Q_u \end{bmatrix} \succeq 0, \quad Q_u \succ 0, \quad Q_N \succeq 0.
\]

Then, by eliminating the variables \( \mathbf{x} \) using the corresponding equality constraints \( \Omega \mathbf{x} = \Omega_0 x_0 + \Gamma \mathbf{u} + \gamma \), an equivalent QP problem can be obtained

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x_N^T Q_N x_N + f_N^T x_N + c_N \\
\text{subject to} & \quad x_0 = \bar{x} \\
& \quad x_N = \tilde{\Omega}_N x_0 + \tilde{\Gamma}_N \mathbf{u} + \tilde{\gamma}_N,
\end{align*}
\]

with matrices defined as in (6A.84) in Appendix 6.A.2, and

\[
\begin{bmatrix} \tilde{Q}_0 & \tilde{R} \\ \tilde{R}^T & \tilde{Q}_u \end{bmatrix} \succeq 0, \quad \tilde{Q_u} \succ 0, \quad \tilde{Q}_N \succeq 0.
\]

The eliminated variables \( \mathbf{x} \) and the dual variables \( \lambda \leftrightarrow -\Omega \mathbf{x} + \Omega_0 x_0 + \Gamma \mathbf{u} + \gamma = 0 \) can be computed as

\[
\begin{align*}
\mathbf{x} &= \Omega^{-1} \Omega_0 x_0 + \Omega^{-1} \Gamma \mathbf{u} + \Omega^{-1} \gamma, \\
\lambda &= \Omega^{-T} \left( Q_x \Omega^{-1} \Omega_0 x_0 + (S + Q_x \Omega^{-1} \Gamma) \mathbf{u} + \Omega_N^T \lambda_N + f_x + Q_x \Omega^{-1} \gamma \right).
\end{align*}
\]

Proof: The proof of Lemma 6.9 is given in the Appendix 6.C.1.

Note that if Lemma 6.9 is used to eliminate the local state variables for all sub-problems \( i \in \mathbb{Z}_{0,\bar{N}} \), it can be interpreted as partially condensing the UFTOC problem (6.1) as in Axehill (2015).

From the definition of the subproblem in (6.34) and Lemma 6.9, it follows that
the resulting condensed subproblem is given by

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \left[ \hat{x}_i \right]^T \begin{bmatrix} Q_{x,i} & Q_{xu,i} \\ Q_{xu,i}^T & Q_{u,i} \end{bmatrix} \left[ \hat{x}_i \right] + \left[ \hat{x}_i \right]^T \begin{bmatrix} \hat{t}_{i,j} \\ \hat{u}_i \end{bmatrix} + \tilde{c}_i + \\
& \quad \frac{1}{2} \hat{x}_{i+1}^T \hat{P}_{i+1} \hat{x}_{i+1} - \hat{\Psi}_{i+1}^T \hat{x}_{i+1} + \hat{c}_{i+1} \\
\text{subject to} & \quad \hat{x}_0 = \bar{x}_i, \\
& \quad \hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \hat{a}_i.
\end{align*}
\] (6.40)

The matrices are defined as in (6A.84) in Appendix 6.A.2 but using the matrices from (6.34). The condensed problem (6.40) can be interpreted as a UFTOC problem \( \mathcal{P}(1) \) with control inputs \( u_i \in \mathbb{R}^{m_i} \), where \( m_i \triangleq \sum_{t=t_i}^{t_i+N_i-1} n_{u,t} \).

**Condensation of a subproblem using the Riccati recursion**

Later it will be shown that when using the condensing technique from Nielsen and Axehill (2015), which is based on the Riccati recursion, Assumption 6.10 will be useful.

**Assumption 6.10.** For each \( t \in \mathbb{Z}_{0,N-1} \) in (6.1), at least one of the following properties holds

\( (i) \) \( Q_{u,t} \in S_{++}^{n_{u,t}} \),

\( (ii) \) \( Q_{u,t} = B_t \in S_{+}^{n_x} \) and \( l_{u,t} = 0. \)

When using this approach to condense the subproblem, Assumption 6.10 is used instead of Assumption 3.4 for the UFTOC problem (6.1). Even though the original UFTOC problem that is solved satisfies Assumption 3.4, being able to handle UFTOC problems that only satisfy this alternative assumption will be important in Section 6.4 when UFTOC problems are solved recursively. Note that if Assumption 6.10(i) holds for all \( t \in \mathbb{Z}_{0,N-1} \), then it coincides with Assumption 3.4.

The idea with this alternative condensing approach is to introduce a change of variables for the control inputs by solving the subproblem using the Riccati recursion for a preliminary choice of \( J(\hat{x}_{i+1}) \). Although this way of condensing the subproblem is conceptually the same as the regular condensing technique, it will be shown that it results in a condensed subproblem with a structure that can facilitate more efficient computations and communications.

To compute this change of variables, a preliminary feedback is computed by letting the UFTOC problem (6.34) be factored and solved using algorithms 4 and 5 for the preliminary choice \( \hat{P}_{i+1} = 0, \hat{\Psi}_{i+1} = 0 \) and \( \hat{c}_{i+1} = 0 \). Let the subindex “0” denote a variable or matrix associated with the preliminary factorization. Then, the preliminary feedback is given by \( u_{0,t} = k_{0,t+1} + K_{0,t+1} x_t \) for \( t \in \mathbb{Z}_{t_i,t_i+N_i-1} \).

Since the cost-to-go function

\[
J(\hat{x}_{i+1}) = \hat{x}_{i+1}^T \hat{P}_{i+1} \hat{x}_{i+1} - \hat{\Psi}_{i+1}^T \hat{x}_{i+1} + \hat{c}_{i+1},
\] (6.41)
might not be zero, it is necessary to capture the effect from the cost-to-go function on the solution to the subproblem. Let $\bar{u}_t \in \mathbb{R}^{n_u,t}$ be the contribution of the non-zero $J(\hat{x}_{t+1})$ on $u_t$. Then, $u_t$ can be expressed using the preliminary feedback

$$u_t = k_{0,t+1} + K_{0,t+1}x_t + \bar{u}_t, \quad t \in \mathbb{Z}_{t_i,t_i+N_i-1},$$  \hfill (6.42)

which can be interpreted as a change of variables from $u_t$ to $\bar{u}_t$. Note that $\bar{u}_t$ is a full $n_u,t$ vector, and hence there is no loss of generality by using the change of variables (6.42). Now, by using (6.42) a condensed UFTOC problem similar to (6.40) but with a different structure in the objective function can be obtained. This condensation of the subproblem is described in Lemma 6.11.

**Lemma 6.11 (Condensing using the Riccati recursion).** Consider a UFTOC problem in the form (6.34) where assumptions 3.3 and 6.10 hold, and assume that algorithms 4 and 5 have been computed for $\tilde{P}_{i+1} = 0$, $\tilde{\Psi}_{i+1} = 0$ and $\tilde{c}_{i+1} = 0$, giving $P_{0,t}, \Psi_{0,t}, K_{0,t+1}, k_{0,t+1}$ and $\tilde{c}_{0,t}$. Then, by using the change of variables (6.42), the UFTOC problem (6.34) can be condensed to a UFTOC problem $\mathcal{P}(1)$

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \hat{x}_i^T P_{0,t_i} \hat{x}_i - \Psi_{0,t_i}^T \hat{x}_i + \tilde{c}_{0,t_i} + \frac{1}{2} \bar{u}_i^T \tilde{Q}_{u,i} \bar{u}_i + \frac{1}{2} \hat{x}_{i+1}^T \tilde{P}_{i+1} \hat{x}_{i+1} - \tilde{\Psi}_{i+1}^T \hat{x}_{i+1} + \tilde{c}_{i+1} \\
\text{subject to} & \quad \hat{x}_i = \hat{x}_i \\
& \quad \hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{S}_i \bar{u}_i + \hat{\alpha}_i,
\end{align*}$$  \hfill (6.43)

with $\bar{u}_i \in \mathbb{R}^{m_i}$, $m_i \triangleq \sum_{t=t_i}^{t_i+N_i-1} n_u,t$ and

$$\begin{align*}
\tilde{Q}_{u,i} & \triangleq \begin{bmatrix} G_{0,t_i+1} & & \\ & \ddots & \\ & & G_{0,t_i+N_i-1} \end{bmatrix} \in \mathbb{S}_{++}^{m_i}, & \hat{A}_i & \triangleq \prod_{t=t_i}^{t_i+N_i-1} (A_t + B_t K_{0,t+1}), \\
S_i & \triangleq \left[ \prod_{t=t_i+1}^{t_i+N_i-1} (A_t + B_t K_{0,t+1}) B_{t_i} \ldots B_{t_i+N_i-1} \right], & \hat{\alpha}_i & \triangleq \sum_{\tau=t_i}^{t_i+N_i-1} \prod_{t=\tau+1}^{t_i+N_i-1} (A_t + B_t K_{0,t+1}) (a_\tau + B_t k_{0,\tau+1}).
\end{align*}$$  \hfill (6.44a,b,c)

If Assumption 6.10(i) holds for all $t \in \mathbb{Z}_{t_i,t_i+N_i-1}$ then $\tilde{Q}_{u,i} \in \mathbb{S}_{++}^{m_i}$.

Furthermore, by using the notation $\lambda_{t_i+1} \triangleq \lambda_{t_i+1}$, the eliminated states $x_t$ and dual variables $\lambda_t$ for $t \in \mathbb{Z}_{t_i+1,t_i+N_i-1}$ can be computed from

$$\begin{align*}
x_{t+1} & = A_t x_t + B_t u_t + a_t, \quad t \in \mathbb{Z}_{t_i+1,t_i+N_i-2}, \\
\lambda_t & = P_{0,t} x_t - \Psi_{0,t} + \left( \prod_{\tau=t}^{t_i+N_i-1} (A_\tau + B_\tau K_{0,\tau+1}) \right)^T \hat{\lambda}_{t+1}, \quad t \in \mathbb{Z}_{t_i+1,t_i+N_i-1}.
\end{align*}$$  \hfill (6.45a,b)

**Proof:** The proof of Lemma 6.11 is given in Appendix 6.C.2. $\square$
By using Lemma 6.11, it is possible to condense the UFTOC problem (6.34) into the problem (6.43), with the matrices defined as in the lemma. This condensed problem is in the UFTOC form \( P(1) \) with \( m_i \) control inputs, which is in the same form as the condensed problem (6.40). However, (6.43) possibly has \( \bar{Q}_{\bar{u}_i} \in S_{++}^{m_i} \) since Assumption 6.10 is used instead of Assumption 3.4, and the cross terms between \( x_0 \) and \( \bar{u}_i \) in the objective function in (6.43) are eliminated when using the change of variables in (6.42). Furthermore, there is no linear penalty term for \( \bar{u}_i \) and the quadratic cost matrix \( \bar{Q}_{\bar{u}_i} \) is block diagonal as can be seen in (6.44a). It will be shown that these properties are important from both a computation and communication point of view. Note that \( S_i \) is the reachability matrix for the subproblem (6.34) when using the change of variables (6.58). Hence, it is closely related to the reachability matrix \( \tilde{B}_i \) in (6.11b) for the subproblems in the parametric programming approach.

### 6.3.3 Reducing the control input dimension in a subproblem

Now it will be shown how to reduce the control input dimension in a subproblem, whenever that is possible. When using any of the two condensing techniques described in Section 6.3.2, the resulting condensed subproblems (6.40) and (6.43) are both in the same UFTOC form \( P(1) \) with \( m_i \) number of control inputs, but with different problem matrices. In some cases it is possible to reduce the number of control inputs in these condensed subproblems to obtain a subproblem with fewer variables. This can be done whenever \( \text{colrank} \tilde{B}_i < m_i \) in (6.40) or \( \text{colrank} S_i < m_i \) in (6.43), respectively. This case corresponds to subproblems which after the condensation process become over-actuated, and the reduction of the control input dimension can be interpreted as performing control allocation. It will be shown how to reduce the control input dimension in the subproblem in three different ways, where two of them are tailored for subproblems that are condensed using the Riccati recursion approach as in Lemma 6.11.

For the first reduction approach, the reduction of the control input dimension in a subproblem in the UFTOC form \( P(1) \) is shown in the following lemma:

**Lemma 6.12 (Reduce control input dimension).** Consider a UFTOC problem in the form

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} x_0 \\ u \end{bmatrix}^T \begin{bmatrix} Q_0 & R \\ R^T & Q_u \end{bmatrix} \begin{bmatrix} x_0 \\ u \end{bmatrix} + \begin{bmatrix} f_0 \\ f_u \end{bmatrix}^T \begin{bmatrix} x_0 \\ u \end{bmatrix} + c + \frac{1}{2} x_N^T Q_N x_N + f_N^T x_N + c_N \\
\text{subject to} & \quad x_0 = \bar{x}, \\
& \quad x_N = \Omega x_0 + \Gamma u + \gamma,
\end{align*}
\]

with \( x_0, x_N \in \mathbb{R}^n, u \in \mathbb{R}^m, \Omega \in \mathbb{R}^{n \times n}, \Gamma \in \mathbb{R}^{n \times m}, \gamma \in \mathbb{R}^n, f_0, f_N \in \mathbb{R}^n, f_u \in \mathbb{R}^m, \)

\[
\begin{bmatrix} Q_0 & R \\ R^T & Q_u \end{bmatrix} \in S_+^{n+m}, \quad Q_u \in S_+^m, \quad Q_N \in S_+^n,
\]

and where \( \text{colrank} \Gamma = r < m \) holds.
Then (6.46) can be reduced to an equality constrained QP problem given by

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} x_0 \end{bmatrix}^T \begin{bmatrix} \tilde{R}^T & \tilde{Q}_u \end{bmatrix} \begin{bmatrix} x_0 \end{bmatrix} + \begin{bmatrix} \tilde{f}_u \end{bmatrix}^T \begin{bmatrix} x_0 \end{bmatrix} + c, \\
\text{subject to} & \quad x_0 = \bar{x}, \\
& \quad x_N = \bar{Q} x_0 + \hat{\Gamma} \hat{u} + \hat{\gamma},
\end{align*}
\]

with \( \hat{u} \in \mathbb{R}^r \), and \( \bar{Q} \in \mathbb{R}^{n \times n} \), \( \hat{\Gamma} \in \mathbb{R}^{n \times r} \), \( \hat{\gamma} \in \mathbb{R}^n \), \( \hat{f}_0, f_N \in \mathbb{R}^n \), \( \hat{f}_u \in \mathbb{R}^r \), and

(6.48)

\[
\begin{bmatrix} \bar{Q}_0 \\ \hat{R} \\ \hat{Q}_u \end{bmatrix} \in \mathbb{S}_{++}^{n+r}, \quad \hat{Q}_u \in \mathbb{S}_{++}^r, \quad Q_N \in \mathbb{S}_+^n,
\]

are defined as in (6A.85) in Appendix 6.A.3.

The variable \( u \) can be recovered from

\[
u = \left( I - V (V^T Q_u V)^{-1} V^T Q_u \right) U \hat{u} - V (V^T Q_u V)^{-1} V^T R^T x_0 - V (V^T Q_u V)^{-1} V^T f_u,
\]

where \( U \) and \( V \) are defined as in the proof in Appendix 6.C.3.

**Proof:** The proof of Lemma 6.12 is given in Appendix 6.C.3.\(\square\)

From Lemma 6.12 it follows that the subproblems \( \mathcal{P} (1) \) in (6.40) with \( \text{colrank} \, \hat{B}_i < m_i \), and in (6.43) with \( \text{colrank} \, S_i < m_i \) and \( \hat{Q}_{\bar{u},i} \in \mathbb{S}_{++}^{m_i} \), can be reduced to a USFQ problem with fewer variables, given by

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} \hat{x}_i \end{bmatrix}^T \begin{bmatrix} \hat{Q}_{x,i} & \hat{Q}_{xu,i} \\ \hat{Q}_{xu,i}^T & \hat{Q}_{u,i} \end{bmatrix} \begin{bmatrix} \hat{x}_i \end{bmatrix} + \begin{bmatrix} \hat{I}_{x,i} \end{bmatrix}^T \begin{bmatrix} \hat{x}_i \end{bmatrix} + \tilde{c}_i + \\
\text{subject to} & \quad \frac{1}{2} \hat{x}_{i+1}^T \hat{P}_{i+1} \hat{x}_{i+1} - \hat{Q}_{x,i}^T \hat{x}_{i+1} - \tilde{Q}_{xu,i}^T \hat{x}_{i+1} + \hat{c}_{i+1}
\end{align*}
\]

(6.51)

subject to

\[
\hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_i \hat{u}_i + \tilde{a}_i,
\]

where \( \hat{A}_i \in \mathbb{R}^{n_x \times n_x} \), \( \hat{B}_i \in \mathbb{R}^{n_x \times n_{\tilde{u},i}} \), \( \tilde{a}_i \in \mathbb{R}^{n_x} \), \( \hat{I}_{x,i} \in \mathbb{R}^{n_x} \), \( \hat{I}_{u,i} \in \mathbb{R}^{n_{\tilde{u},i}} \),

\[
\begin{bmatrix} \hat{Q}_{x,i} & \hat{Q}_{xu,i} \\ \hat{Q}_{xu,i}^T & \hat{Q}_{u,i} \end{bmatrix} \in \mathbb{S}_{++}^{n_x + n_{\tilde{u},i}}, \quad \hat{Q}_{u,i} \in \mathbb{S}_{++}^{n_{\tilde{u},i}},
\]

(6.52)

are defined as in the lemma. Here \( n_{\tilde{u},i} \triangleq \text{colrank} \, \hat{B}_i = \text{colrank} \, \hat{B}_i = \text{colrank} \, S_i \). Note that \( n_{\tilde{u},i} \leq n_x \) does always hold. When (6.43) is reduced, it follows from the definition of the problem and Lemma 6.12 that \( \hat{Q}_{xu,i} = 0 \) and \( \hat{I}_{u,i} = 0 \). This is a consequence of using the change of variables (6.42).
For the case with colrank $\tilde{B}_i = m_i$ in (6.40), the reduced problem (6.51) is obtained directly by defining

$$
\begin{align*}
\hat{Q}_{x,i} & \triangleq \tilde{Q}_{x,i}, \quad \hat{Q}_{xu,i} \triangleq \tilde{Q}_{xu,i}, \quad \hat{Q}_{u,i} \triangleq \tilde{Q}_{u,i}, \quad \hat{l}_{x,i} \triangleq \tilde{l}_{x,i}, \quad \hat{l}_{u,i} \triangleq \tilde{l}_{u,i}, \\
\hat{c}_i & \triangleq \tilde{c}_i, \quad \hat{A}_i \triangleq \tilde{A}_i, \quad \hat{B}_i \triangleq \tilde{B}_i, \quad \hat{a}_i \triangleq \tilde{a}_i,
\end{align*}
$$

(6.53a)

and when colrank $S_i = m_i$ and $\hat{Q}_{\tilde{u},i} \in S_{++}^{m_i}$ in (6.43), the reduced problem (6.51) is obtained directly by defining

$$
\begin{align*}
\hat{Q}_{x,i} & \triangleq P_{0,t_i}, \quad \hat{Q}_{xu,i} \triangleq 0, \quad \hat{Q}_{u,i} \triangleq \hat{Q}_{\tilde{u},i}, \quad \hat{l}_{x,i} \triangleq -\Psi_{0,t_i}, \quad \hat{l}_{u,i} \triangleq 0, \\
\hat{c}_i & \triangleq \tilde{c}_{0,t_i}, \quad \hat{B}_i \triangleq S_i.
\end{align*}
$$

(6.54a)

For the last subproblem $i = \tilde{N}$ the cost-to-go function is always known according to Remark 6.8. Hence, all variables except $\hat{x}_{\tilde{N}}$ can be eliminated from the last subproblem, and the optimal cost-to-go function $J(\hat{x}_{\tilde{N}})$ for this problem can be computed directly using the Riccati recursion presented in Section 3.4. By using the Riccati recursion for the problem $P(1)$ in (6.40), the cost-to-go function and the solution are computed from $\hat{x}_{\tilde{N}}$ as

$$
\begin{align*}
J(\hat{x}_{\tilde{N}}) &= \hat{x}_{\tilde{N}}^T \hat{P}_{\tilde{N}} \hat{x}_{\tilde{N}} - \hat{\Psi}_{\tilde{N}}^T \hat{x}_{\tilde{N}} + \hat{c}_{\tilde{N}}, \\
u_{\tilde{N}} &= \hat{K}_{\tilde{N}} \hat{x}_{\tilde{N}} + \hat{k}_{\tilde{N}}, \\
x_{\tilde{N}} &= (\hat{A}_{\tilde{N}} + \hat{B}_{\tilde{N}} \hat{K}_{\tilde{N}}) \hat{x}_{\tilde{N}} + \hat{B}_{\tilde{N}} \hat{k}_{\tilde{N}}, \\
\lambda_0 &= \hat{P}_{\tilde{N}} \hat{x}_{\tilde{N}} - \hat{\Psi}_{\tilde{N}}, \\
\lambda_{\tilde{N}} &= \hat{P}_{\tilde{N}+1} \hat{x}_{\tilde{N}+1} - \hat{\Psi}_{\tilde{N}+1} = Q_{\tilde{N}} x_{\tilde{N}} + f_{\tilde{N}}.
\end{align*}
$$

(6.55a)

Here, $\hat{c}_{\tilde{N}}$ is computed by inserting the solution into the objective function in (6.40). The solution for the last subproblem when the Riccati based condensing technique is used can be derived analogously.

**Tailored reduction of the subproblems**

The second way of reducing the control input dimension in a subproblem (6.43), which also works when $\hat{Q}_{\tilde{u},i} \in S_{++}^{m_i}$ due to Assumption 6.10, is presented next. This approach avoids computing the orthonormal basis $V$ which is required to form the reduced subproblem in Lemma 6.12. To derive this alternative reduction strategy, Lemma 6.13 will be used.

**Lemma 6.13.** Let $Q_{u,t} = B_t \in S_{++}^{nx}$. Then $\mathcal{N}(G_{t+1}) = \mathcal{N}(Q_{u,t}) = \mathcal{N}(B_t)$, where $G_{t+1} \in S_{++}^{nx}$ is defined similarly as in (3.41c).

**Proof:** Let $z \in \mathcal{N}(Q_{u,t})$ be arbitrary. Then, $G_{t+1} z = Q_{u,t} z + B_t^T P_{t+1} B_t z = 0$ since $Q_{u,t} = B_t$. Hence $\mathcal{N}(Q_{u,t}) \subset \mathcal{N}(G_{t+1})$. Now, let $z \in \mathcal{N}(G_{t+1})$ be arbitrary. Then $z \in \mathcal{N}(G_{t+1}) \iff G_{t+1} z = 0 \iff z^T G_{t+1} z = 0$, and it follows that $z^T (Q_{u,t} + B_t^T P_{t+1} B_t) z = 0 \iff z^T Q_{u,t} z = 0$ since $B_t^T P_{t+1} B_t \in S_{++}^{nx}$. However, since $Q_{u,t} \in S_{++}^{nx}$ it follows from Lemma A.2 that $z^T Q_{u,t} z = 0 \iff Q_{u,t} z = 0$. Hence, $\mathcal{N}(G_{t+1}) \subset \mathcal{N}(Q_{u,t})$. Since also $\mathcal{N}(Q_{u,t}) \subset \mathcal{N}(G_{t+1})$ it follows that
\( \mathcal{N}(G_{t+1}) = \mathcal{N}(Q_{u,t}) \). From \( Q_{u,t} = B_t \) it directly follows that \( \mathcal{N}(Q_{u,t}) = \mathcal{N}(B_t) \), which concludes the proof.

When using the condensing technique which is based on the Riccati recursion, the condensed subproblem (6.43) does not have any cross-terms between \( \hat{x}_i \) and \( \bar{u}_i \). This will be exploited here to derive an alternative reduction technique for a UFTOC problem in the form (6.43). By forming the KKT optimality conditions for the condensed UFTOC problem (6.43), it can be seen that the solution to this problem is obtained by computing the solution to the KKT system

\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & P_{0,t_i} & 0 & \hat{A}_i^T & 0 \\
0 & 0 & \bar{Q}_{\bar{u},i} & S_i^T & 0 \\
0 & \hat{A}_i & S_i & 0 & -I \\
0 & 0 & 0 & -I & \bar{P}_{t+1}
\end{bmatrix} \begin{bmatrix}
\hat{\lambda}_i \\
\hat{x}_i \\
\bar{u}_i \\
\hat{\lambda}_{i+1} \\
\hat{x}_{i+1}
\end{bmatrix} = \begin{bmatrix}
\hat{x}_i \\
\Psi_{0,t_i} \\
\bar{u}_i \\
\hat{\lambda}_{i+1} \\
\hat{x}_{i+1}
\end{bmatrix}, \tag{6.56}
\]

For the case with \( r_i = \text{colrank} S_i < m_i \) it is possible to reduce the size of the system of equations (6.56) into a smaller system of equations with the same symmetric KKT structure. This will be done by manipulating the KKT matrix (6.56) in several steps to obtain a KKT system for a smaller UFTOC problem. Let \([U \ V]\) be an orthogonal matrix where the columns of \( U \in \mathbb{R}^{m_i \times r_i} \) form an orthonormal basis for \( \mathcal{R}(S_i^T) \) and the columns of \( V \in \mathbb{R}^{m_i \times (m_i - r_i)} \) form an orthonormal basis for \( \mathcal{N}(S_i) \), giving \( \mathbb{R}^{m_i} = U \oplus V \) (Lancaster and Tismenetsky, 1985). Then, by multiplying the third block row in the KKT system (6.56) with \([U \ V]^T \) from the left, the system of equations can equivalently be written

\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & P_{0,t_i} & 0 & \hat{A}_i^T & 0 \\
0 & 0 & U^T \bar{Q}_{\bar{u},i} & U^T S_i^T & 0 \\
0 & 0 & V^T \bar{Q}_{\bar{u},i} & 0 & 0 \\
0 & \hat{A}_i & S_i & 0 & -I \\
0 & 0 & 0 & -I & \bar{P}_{t+1}
\end{bmatrix} \begin{bmatrix}
\hat{\lambda}_i \\
\hat{x}_i \\
\bar{u}_i \\
\hat{\lambda}_{i+1} \\
\hat{x}_{i+1}
\end{bmatrix} = \begin{bmatrix}
\hat{x}_i \\
\Psi_{0,t_i} \\
\bar{u}_i \\
\hat{\lambda}_{i+1} \\
\hat{x}_{i+1}
\end{bmatrix}, \tag{6.57}
\]

where \( V^T S_i^T = 0 \) in the \((4,4)\)-block follows from the definition of \( V \). The fourth block equation in (6.57) states that \( V^T \bar{Q}_{\bar{u},i} \bar{u}_i = 0 \) must hold. By exploiting the orthogonality of \([U \ V]\) using the property \( U^T V = 0 \), it follows that the equation \( V^T \bar{Q}_{\bar{u},i} \bar{u}_i = 0 \) holds whenever

\[
\bar{Q}_{\bar{u},i} \bar{u}_i = U \hat{u}_i \iff \bar{u}_i = \bar{Q}_{\bar{u},i}^\dagger U \hat{u}_i + \left(I - \bar{Q}_{\bar{u},i}^\dagger \bar{Q}_{\bar{u},i}\right) \hat{z}_i, \tag{6.58}
\]

where \( \bar{Q}_{\bar{u},i}^\dagger \) is the Moore-Penrose pseudo-inverse of \( \bar{Q}_{\bar{u},i} \) and \( \hat{z}_i \in \mathbb{R}^{m_i} \) is an arbitrary vector (Penrose, 1955). From Assumption 6.10, Lemma 6.13 and the definitions
of $\bar{Q}_{u,i}$ and $S_i$ it follows that whenever $Q_{u,i}$ is singular, $N(Q_{u,i}) \subset N(S_i)$ holds. Then, from Lemma A.3 it follows that $\mathcal{R}(S_i^T) \subset \mathcal{R}(Q_{u,i})$ holds. Furthermore, since the columns of $U$ form a basis for $\mathcal{R}(S_i^T) \subset \mathcal{R}(Q_{u,i})$, it implies that

$$Q_{u,i}^\dagger U = U, \quad S_i = S_i Q_{u,i}^\dagger Q_{u,i},$$  \hspace{1cm} (6.59)

hold (Penrose, 1955; Albert, 1969). By using (6.58) to eliminate $\bar{u}_i$ in (6.57), together with (6.59) and $U^T U = I$ gives the following equivalent system of equations

$$\begin{bmatrix} 0 & -I & 0 & 0 & 0 \\ -I & P_{0,t_i} & 0 & \bar{A}_i^T & 0 \\ 0 & 0 & I & U^T S_i^T & 0 \\ 0 & \bar{A}_i & S_i \bar{Q}_{u,i}^\dagger U & 0 & -I \\ 0 & 0 & 0 & -I & \bar{P}_{i+1} \end{bmatrix} \begin{bmatrix} \lambda_i \\ \bar{x}_i \\ \bar{u}_i \\ \lambda_{i+1} \\ \bar{x}_{i+1} \end{bmatrix} = \begin{bmatrix} -\bar{x}_i \\ \Psi_{0,t_i} \\ 0 \\ -\bar{u}_i \\ \Psi_{i+1} \end{bmatrix}.$$  \hspace{1cm} (6.60)

Finally, to obtain a system of equations which has the symmetric KKT structure, the third block row in (6.60) is multiplied with $U^T \bar{Q}_{u,i}^\dagger U \in S_{r+i}^{++}$ from the left. By also noting that $UU^T S_i^T = S_i^T$ follows from the definition of $U$, the system of equations (6.60) can be written in the form

$$\begin{bmatrix} 0 & -I & 0 & 0 & 0 \\ -I & P_{0,t_i} & 0 & \bar{A}_i^T & 0 \\ 0 & 0 & U^T \bar{Q}_{u,i}^\dagger U & U^T \bar{Q}_{u,i}^\dagger S_i^T & 0 \\ 0 & \bar{A}_i & S_i \bar{Q}_{u,i}^\dagger U & 0 & -I \\ 0 & 0 & 0 & -I & \bar{P}_{i+1} \end{bmatrix} \begin{bmatrix} \lambda_i \\ \bar{x}_i \\ \bar{u}_i \\ \lambda_{i+1} \\ \bar{x}_{i+1} \end{bmatrix} = \begin{bmatrix} -\bar{x}_i \\ \Psi_{0,t_i} \\ 0 \\ -\bar{u}_i \\ \Psi_{i+1} \end{bmatrix}.$$  \hspace{1cm} (6.61)

Here $U^T \bar{Q}_{u,i}^\dagger U \in S_{r+i}^{++}$ since $\mathcal{R}(U) = \mathcal{R}(S_i^T) \subset \mathcal{R}(\bar{Q}_{u,i}) = \mathcal{R}(\bar{Q}_{u,i}^\dagger)$. The last equality follows from the properties of the Moore-Penrose pseudo-inverse (Lancaster and Tismenetsky, 1985).

Note that since $\bar{Q}_{u,i}$ is block diagonal, also $\bar{Q}_{u,i}^\dagger$ is block diagonal and given by

$$\bar{Q}_{u,i}^\dagger = \text{blkdiag}(G_{0,t_i+1}^\dagger, \ldots, G_{0,t_i+N_i}^\dagger).$$  \hspace{1cm} (6.62)

Furthermore, since $G_{0,t_i+1}^\dagger = G_{0,t_i+1}^{-1}$ when $G_{0,t_i+1}$ is non-singular (Penrose, 1955), the Moore-Penrose pseudo-inverse is only required for the $G_{0,t_i+1}$ that are singular.

This technique for reducing a subproblem is presented in Lemma 6.14 below.
Lemma 6.14 (Tailored reduction of control inputs). Consider a UFTOC problem as in (6.43) in Lemma 6.11, and assume that colrank $S_i < m_i$ holds. Then, this problem can be reduced to a UFTOC problem

$$
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} \begin{bmatrix} \dot{x}_i \\ \dot{u}_i \end{bmatrix}^T \begin{bmatrix} \dot{Q}_{x,i} & 0 \\ 0 & \dot{Q}_{u,i} \end{bmatrix} \begin{bmatrix} \dot{x}_i \\ \dot{u}_i \end{bmatrix} + \begin{bmatrix} \dot{l}_{x,i} \\ \dot{l}_{u,i} \end{bmatrix}^T \begin{bmatrix} \dot{x}_i \\ \dot{u}_i \end{bmatrix} + \dot{c}_i \\
\text{subject to} & \quad \dot{x}_i = \ddot{x}_i \\
& \quad \dot{x}_{i+1} = \hat{A}_i \dot{x}_i + \hat{B}_i \dot{u}_i + \hat{a}_i,
\end{aligned}
$$

(6.63)

where

$$
\begin{gathered}
\dot{Q}_{x,i} \triangleq P_{0,t_i} \in S_{++}^{n_x}, \quad \dot{Q}_{u,i} \triangleq U^T \dot{Q}_{\hat{u},i} U \in S_{++}^{n_{\hat{u},i}}, \quad \dot{l}_{x,i} \triangleq \Psi_{0,t_i} \in \mathbb{R}^{n_x}, \quad \dot{c}_i \triangleq \bar{c}_{0,t_i} \in \mathbb{R}, \\
\hat{B}_i \triangleq S_i \dot{Q}_{\hat{u},i} U \in \mathbb{R}^{n_x \times n_{\hat{u},i}}, \text{ with } n_{\hat{u},i} = \text{colrank} \hat{B}_i = \text{colrank} S_i,
\end{gathered}
$$

(6.64a)

$$
\hat{A}_i \text{ and } \hat{a}_i \text{ are defined as in (6.44), and the columns of } U \text{ form an orthonormal basis of } \mathcal{R} \left( S_i^T \right).$$

The variable $\bar{u}_i$ can be computed from

$$
\bar{u}_i = \dot{Q}_{\hat{u},i} U \hat{u}_i + \left( I - \dot{Q}_{\hat{u},i} \dot{Q}_{\hat{u},i} \right) \hat{z}_i.
$$

(6.65)

Proof: From the derivation of (6.61) it follows that when colrank $S_i < m_i$, the KKT system (6.66) can be reduced to this equivalent system of equations with fewer variables and equations. Furthermore, by introducing the definitions (6.64), it follows that a solution to the KKT system (6.61) is also a primal and dual solution to the UFTOC problem (6.63). The solution (6.58) directly gives (6.65).

Since the use of the preliminary feedback in (6.42) results in a block diagonal $\dot{Q}_u$ with blocks given by $G_{0,t+1}$ for $t \in \mathbb{Z}_{0,N-1}$, computing $\dot{Q}_u$ and $\hat{B}$ in (6.64) can be efficiently done by block-wise computations when $U$ is computed. Furthermore, the factorizations of $G_{0,t+1}$ from the computation of $K_{0,t+1}$ can be re-used here.

For the last subproblem $i = \hat{N}$, the variables $\hat{P}_{\hat{N}+1} = Q_{x,N}$, $\hat{\Psi}_{\hat{N}+1} = -l_{x,N}$ and $\hat{c}_{\hat{N}+1} = c_N$ in (6.34) are known. Hence, the cost-to-go function for the last subproblem is given by

$$
J(\hat{x}_{\hat{N}}) = \hat{x}_{\hat{N}}^T \dot{Q}_{x,\hat{N}} \hat{x}_{\hat{N}} + \hat{l}_{x,\hat{N}}^T \hat{x}_{\hat{N}} + \hat{c}_{\hat{N}},
$$

(6.66)

where the Riccati factorization and backward recursion in algorithms 4 and 5 are used to compute $\dot{Q}_{x,\hat{N}} \triangleq P_{\hat{N}}, \hat{l}_{x,\hat{N}} \triangleq -\Psi_{\hat{N}}$ and $\dot{c}_{\hat{N}} \triangleq \bar{c}_{\hat{N}}$. 
Tailored Riccati-based reduction algorithm

Here, the third way of reducing the control input dimension in a subproblems is presented. It is similar to the second way, but a transformation \( \hat{u}_i = T_i \hat{v}_i \) is used to avoid computing the orthonormal basis \( U \).

**Lemma 6.15.** Consider a uftoc problem as in Lemma 6.14. Introduce the transformation \( \hat{u}_i = T_i \hat{v}_i \) where \( T \in \mathbb{R}^{n_{u,i} \times n_x} \) has full rank, \( UT = S^T \) and \( \hat{v}_i \in \mathbb{R}^{n_x} \). Then, the primal and dual solution given by \( \hat{x}_i^*, \hat{v}_i^*, \hat{x}_{i+1}^*, \hat{\lambda}_i^* \) and \( \hat{\lambda}_{i+1}^* \) to

\[
\begin{align*}
\min_{\hat{x}_i, \hat{v}_i, \hat{x}_{i+1}} & \quad \frac{1}{2} \left[ \hat{x}_i^T \hat{Q}_{x,i} \hat{x}_i \right] + \frac{1}{2} \hat{x}_{i+1}^T \hat{P}_{i+1} \hat{x}_{i+1} + \hat{c}_i + \hat{c}_{i+1} \\
\text{subject to} & \quad \hat{x}_i = \hat{x}_i \\
& \quad \hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_v i \hat{v}_i + \hat{a}_i,
\end{align*}
\]

with

\[
\hat{Q}_{v,i} = \hat{B}_v,i \triangleq S_i \bar{Q}_{u,i}^T S_i^T \in \mathbb{S}_{++}^{n_x},
\]

is also a solution to (6.63) with \( \hat{u}_i^* = T_i \hat{v}_i^* \), and the eliminated variable \( \bar{u}_i \) in (6.65) can instead be computed from

\[
\bar{u}_i = \bar{Q}_{u,i}^T \hat{v}_i + \left( I - \bar{Q}_{u,i}^T \bar{Q}_{u,i} \right) \hat{z}_i.
\]

Furthermore, if \( n_{\bar{u},i} = n_x \) then \( \hat{Q}_{v,i} \in \mathbb{S}_{++}^{n_x} \).

**Proof:** By using \( \hat{Q}_{v,i} = \hat{B}_v,i = S_i \bar{Q}_{u,i}^T S_i^T = T^T U^T \bar{Q}_{u,i}^T U^T \), the kkt system for the uftoc problem (6.67) is given by

\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & \hat{Q}_{x,i} & 0 & 0 & 0 \\
0 & 0 & T^T U^T \bar{Q}_{u,i}^T U^T & T^T U^T \bar{Q}_{u,i}^T \bar{S}_i^T & 0 \\
0 & \hat{A}_i & S_i \bar{Q}_{u,i}^T U^T & 0 & -I \\
0 & 0 & 0 & -I & \hat{P}_{i+1}^T
\end{bmatrix}
\begin{bmatrix}
\hat{\lambda}_i \\
\hat{x}_i \\
\hat{v}_i \\
\hat{\lambda}_{i+1} \\
\hat{\psi}_{i+1}
\end{bmatrix}
= \begin{bmatrix}
-\bar{x}_i \\
-\bar{\lambda}_x,i \\
0 \\
0 \\
0
\end{bmatrix}.
\]

Now, multiply the third block row with \( (T_i T_i^T)^{-1} T_i \) from the left. This does not change the solution set of (6.70) since rank \( U^T \bar{Q}_{u,i}^T U^T = \text{rank} T^T U^T \bar{Q}_{u,i}^T U^T = n_{\bar{u},i} \) (Lancaster and Tismenetsky, 1985). Furthermore, by using \( \hat{u}_i = T_i \hat{v}_i \), it follows that a solution to (6.70) is also a solution to the kkt system of (6.63), given by
When \( n_{\hat{u},i} = n_x \), \( T_i \in \mathbb{R}^{n_x \times n_x} \) is non-singular by definition and hence \( \hat{Q}_{v,i} \in \mathbb{S}_{++}^{n_x} \). Furthermore, by inserting \( \hat{u}_i = T_i \hat{v}_i \) into (6.65) gives (6.69), which concludes the proof.

By using Lemma 6.15, the reduced uFTOC problem (6.67) can be used instead of (6.63). This choice of reduced uFTOC problem has \( n_x \geq n_{\hat{u},i} = \text{colrank} \ S_i \) control inputs, and hence (6.67) might have more control inputs than (6.63). However, the advantage when using this choice is that \( \hat{Q}_u \) and \( \hat{B} \) can be easily computed from the definitions of \( \bar{Q}_{\hat{u},i} \) and \( S_i \) without having to compute the orthonormal basis \( U \). It will later be seen in Section 6.3.7 that less communication is usually needed in a parallel setting.

The computational procedure for both condensing and reducing the uFTOC problem using lemmas 6.11 and 6.15 is summarized in Algorithm 11. This algorithm is basically a Riccati factorization and backward recursion as in algorithms 4 and 5, respectively. Algorithm 11 is used to eliminate the local variables and to reduce the control input in each subproblem \( i \in \mathbb{Z}_{0,\hat{N} - 1} \). Algorithm 11 is similar to the partitioned DP algorithm presented in Wright (1991), but here the uFTOC structure is conserved.

### 6.3.4 Constructing the master problem

In Section 6.3.2 it was shown how a subproblem in the form (6.34) can be condensed to either (6.40) or (6.43), depending on which condensation technique that is used. Furthermore, in Section 6.3.3 it was shown that it sometimes is possible to reduce the size of the control input in different ways to obtain a smaller uFTOC problem in either of the forms (6.51), (6.63) or (6.67), depending on the reduction technique. Furthermore, the last subproblem \( i = \hat{N} \) can be described by the cost-to-go function \( J(\hat{x}_N) \) according to (6.55) and (6.66), respectively. Hence, by using the definition of \( \hat{x}_i = x_{t_i} \) and \( J(\hat{x}_{i+1}) \), it is possible to construct a master problem with shorter prediction horizon \( \hat{N} < N \) from the reduced subproblems. This is described in Theorem 6.16 below.
Algorithm 11 Condensation and reduction using the Riccati factorization

1: $P_N := 0$, $\Psi_N := 0$, $\bar{c}_N := 0$
2: $Q_u := 0$, $D_N := I$, $d_N := 0$
3: for $t = N - 1, \ldots, 0$ do
4: $F_{t+1} := Q_{x,t} + A_t^T P_{t+1} A_t$
5: $G_{t+1} := Q_{u,t} + B_t^T P_{t+1} B_t$
6: $H_{t+1} := Q_{xu,t} + A_t^T P_{t+1} B_t$
7: Compute and store a factorization of $G_{t+1}$
8: Compute a solution $K_{t+1}$ to $G_{t+1} K_{t+1} = -H_t^T$
9: Compute a solution $k_{t+1}$ to $G_{t+1} k_{t+1} = B_t^T \Psi_{t+1} - l_{u,t} - B_t^T P_{t+1} a_t$
10: $\Psi_t := A_t^T \Psi_{t+1} - H_{t+1} k_{t+1} - l_{x,t} - A_t^T P_{t+1} a_t$
11: $P_t := F_{t+1} - K_{t+1} G_{t+1} K_{t+1}$
12: Compute a solution $L_{t+1}$ to $G_{t+1} L_{t+1} = -B_t D_{t+1}$
13: $D_t := (A_t^T + K_{t+1} B_t^T) D_{t+1}$
14: $d_t := d_{t+1} + D_t^T (a_t + B_t k_{t+1})$
15: $\hat{Q}_u := \hat{Q}_u + L_{t+1} G_{t+1} L_{t+1}$
16: $A := D_0^T$, $\hat{B} := \hat{Q}_u$, $\hat{a} := d_0$
17: $\hat{Q}_x := P_0$, $\hat{l}_x := -\Psi_0$, $\hat{c} := \bar{c}_0$

Theorem 6.16 (Construction of the master problem). Consider a uftoc problem $\mathcal{P}(N)$ given in (6.1) where Assumption 3.3, and either Assumption 3.4 or 6.10, hold. Then, this problem can be reduced in parallel to the smaller uftoc problem

$$
\min_{\hat{x},u} \sum_{i=0}^{\hat{N}-1} \left( \frac{1}{2} \left[ \hat{x}_i \bar{u}_i \right]^T \left[ \hat{Q}_{x,i} \hat{Q}_{xu,i} \hat{Q}_{u,i} \right] \left[ \hat{x}_i \bar{u}_i \right] + \left[ \hat{l}_{x,i} \right]^T \hat{x}_i + \hat{c}_i \right) + \frac{1}{2} \hat{\bar{x}}_N^T \hat{Q}_{x,N} \hat{\bar{x}}_N + \hat{l}_{x,N}^T \hat{\bar{x}}_N + \hat{c}_N \quad \text{subject to} \quad \hat{x}_0 = \bar{x}, \quad \hat{x}_{i+1} = \hat{A}_i \hat{x}_i + \hat{B}_i \bar{u}_i + \hat{a}_i, \quad i \in \mathbb{Z}_{0,\hat{N}-1},
$$

(6.72)

with $\hat{x}_i \in \mathbb{R}^{n_x}$, $\bar{u}_i \in \mathbb{R}^{n_{\bar{u},i}}$, and $n_{\bar{u},i} \leq n_x$.

Proof: Partition the uftoc problem (6.1) into $\hat{N} + 1$ subproblems in the form (6.32). Then, each subproblem can be condensed in either of the ways presented in Section 6.3.2, and reduced as described in Section 6.3.3 to a subproblem with $n_{\bar{u},i} \leq n_x$ control inputs. Since $J(\hat{x}_N)$ is known from (6.66), and it follows from Bertsekas (2000) that $J(\hat{x}_i)$ for all $i \in \mathbb{Z}_{0,\hat{N}-1}$ can be computed from the solution of
6 Parallel Computations of Search Directions

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \hat{x}_{i}^{T} \hat{Q}_{x,i} \hat{x}_{i} + \hat{u}_{i}^{T} \hat{Q}_{u,i} \hat{u}_{i} + \hat{r}_{i}^{T} \hat{x}_{i} + \hat{c}_{i} + J(\hat{x}_{i+1}) \\
\text{subject to} & \quad \hat{x}_{i+1} = \hat{A}_{i} \hat{x}_{i} + \hat{B}_{i} \hat{u}_{i} + \hat{a}_{i},
\end{align*}
\]

(6.73)

it follows by induction that the master problem can be defined as (6.72). Furthermore, the condensation and reduction of each subproblem can be performed independently of each other, which concludes the proof.

The structure of the master UFTOC problem in Theorem 6.16 is different depending on which condensation and reduction technique that has been used. When the condensing technique based on the Riccati recursion is used, the condensed problem is given by (6.43) and hence \( \hat{Q}_{u,i} = 0 \) and \( \hat{l}_{u,i} = 0 \) for such problems. Besides this, if the Riccati-based condensation and reduction algorithm presented in Algorithm 11 is used, then also \( \hat{Q}_{u,i} = \hat{B}_{i} \in S_{+}^{n_{u,i}} \) holds. Furthermore, if the original UFTOC problem (6.1) satisfies Assumption 3.4, then also the master problem satisfies Assumption 3.4 unless the subproblems are reduced using Algorithm 11. When the subproblems are reduced using Algorithm 11, then the master UFTOC problem instead satisfies Assumption 6.10.

The master problem (6.72) can be solved using any suitable method to compute the solution \( \hat{u}_{i} \) for \( i \in \mathbb{Z}_{0,N-1} \), and \( \hat{x}_{i}^{\ast} \) and \( \hat{\lambda}_{i}^{\ast} \) for \( i \in \mathbb{Z}_{0,N} \). If the Riccati recursion is used then also the cost-to-go function at each stage \( i \), defined by \( \hat{P}_{i}, \hat{\Psi}_{i} \) and \( \hat{c}_{i} \), is computed. When some of the \( \hat{Q}_{u,i} \in S_{+}^{n_{u,i}} \) are singular, the Riccati recursion can be computed as presented in Section 3.4.2.

**Remark 6.17.** When Lemma 6.15 is used to reduce a subproblem it is possible that some \( \hat{Q}_{u,i} = \hat{Q}_{v,i} \) are singular. Even in such cases, there will always exist a solution to the master problem since \( \hat{Q}_{u,i} = \hat{B}_{i} \) and \( \hat{l}_{u,i} = 0 \) according to Assumption 6.10. However, since the optimal control input will be computed as in Section 3.4.2, i.e.,

\[
\hat{u}_{i} = \hat{k}_{i+1} + \hat{K}_{i+1} \hat{x}_{i} + \hat{u}_{N,i}, \quad \hat{u}_{N,i} \in \mathcal{N}(\hat{G}_{i+1}),
\]

(6.74)

it is not unique. From Lemma 6.13 it follows that \( \mathcal{N}(\hat{G}_{i+1}) = \mathcal{N}(\hat{B}_{i}) \). Since \( \hat{u}_{N,i} \in \mathcal{N}(\hat{G}_{i+1}) \) \( \iff \hat{u}_{N,i} \in \mathcal{N}(\hat{B}_{i}) \) it follows that

\[
\hat{B}_{i} \hat{u}_{i}^{\ast} = \hat{B}_{i} \left( \hat{k}_{i+1} + \hat{K}_{i+1} \hat{x}_{i}^{\ast} + \hat{u}_{N,i} \right) = \hat{B}_{i} \left( \hat{k}_{i+1} + \hat{K}_{i+1} \hat{x}_{i}^{\ast} \right).
\]

(6.75)

Hence, by using the state recursion in (3.56) it follows that the optimal states and dual variables \( \hat{x}_{i+1}^{\ast} \) and \( \hat{\lambda}_{i+1}^{\ast} \) are still unique even in this case.

Once the master UFTOC problem is solved, the solution to the original UFTOC problem (6.1) can be obtained from the solutions to all subproblems (6.34) by using the definitions of \( x_{i} \) and \( u_{i} \) in (6.31), together with the definitions \( \hat{x}_{i} = x_{i} \) and \( \hat{\lambda}_{i} = \lambda_{i} \). Hence, the optimal primal and dual solution to (6.1) is obtained as
\[ x^* = \begin{bmatrix} \hat{x}^*_0 \\ x^*_0 \\ \vdots \\ \hat{x}^*_N \\ x^*_N \\ \hat{x}^*_{N+1} \end{bmatrix}, \quad u^* = \begin{bmatrix} u^*_0 \\ u^*_1 \\ \vdots \\ u^*_N \end{bmatrix}, \quad \lambda^* = \begin{bmatrix} \lambda^*_0 \\ \lambda^*_1 \\ \vdots \\ \lambda^*_N \\ \lambda^*_{N+1} \end{bmatrix}. \] (6.76)

The solutions \( x^*_i, u^*_i \) and \( \lambda^*_i \) of the subproblems can be computed in two different ways: by substitution of the optimal solution to the master problem, or by resolving the subproblems in the form in (6.40) using \( \hat{x}^*_i \) and the (now known) cost-to-go function \( J(\hat{x}_{i+1}) \). How to solve the original \( \text{uftoc} \) problem using the first approach will be described in Section 6.3.5 and using the second approach will be described in Section 6.3.6.

**Remark 6.18.** The solution of each subproblem can be computed independently of the other subproblems. Hence, the solution to the original \( \text{uftoc} \) problem (6.1) can be computed in parallel, given the solution to the master problem (6.72).

### 6.3.5 Solving the subproblems using the solution to the master problem

Depending on which approach that has been used to condense and reduce a subproblem, the solution to the subproblem can be computed in different ways using the solution to the master problem. Here, these are presented in detail.

**The control input dimension is reduced using Lemma 6.12**

When a subproblem has been reduced using Lemma 6.12, the optimal control input for subproblem \( i \) can be computed directly from \( \hat{x}^*_i \) and \( \hat{u}^*_i \) using (6.50). From this equation it follows that the optimal control input vector \( u^*_i \) can be computed from

\[ u^*_i = \left( I - V \left( V^T \hat{Q}_{u,i} V \right)^{-1} V^T \hat{Q}_{u,i} \right) \hat{u}^*_i - V \left( V^T \hat{Q}_{u,i} V \right)^{-1} V^T \tilde{x}^*_{i-1} - V \left( V^T \hat{Q}_{u,i} V \right)^{-1} V^T \tilde{x}^*_{i-1} - V \left( V^T \hat{Q}_{u,i} V \right)^{-1} V^T \tilde{x}^*_{i-1} - V \left( V^T \hat{Q}_{u,i} V \right)^{-1} V^T \tilde{x}^*_{i-1}, \] (6.77)

where \( U \) and \( V \) are defined as in the definitions of the matrices in (6A.84). Using \( \tilde{x}^*_i, u^*_i \) and \( \tilde{\lambda}^*_i \), the optimal local variables \( x^*_i \) and the optimal dual variables \( \lambda^*_i \) corresponding to the local constraints can be computed from (6.39), giving

\[ x^*_i = A_i^{-1} A_{0,i} \hat{x}^*_i + A_i^{-1} B_i u^*_i + A_i^{-1} a_i, \] (6.78a)

\[ \lambda^*_i = A_i^{-T} \left( Q_{x,i} A_i^{-1} A_{0,i} \hat{x}^*_i + (Q_{x,i} + Q_{x,i} A_i^{-1} B_i) u^*_i + A_i^{-T} \tilde{\lambda}^*_{i+1} + l_{x,i} + Q_{x,i} A_i^{-1} a_i \right). \] (6.78b)

For subproblems where Assumption 3.4 holds and that are condensed using the tailored approach presented in Lemma 6.11, the solution is computed analogously but where the specific structure of the condensed problem (6.43) is exploited. For the last subproblem \( i = \hat{N} \), the solution from (6.55) for \( \hat{x}^*_N \) can be used.
Using the tailored condensation and reduction techniques

For problems that have been condensed using Lemma 6.11 and reduced using the approaches in lemmas 6.14 or 6.15, the optimal \( \tilde{u}_i^* \) can be computed from (6.65) or (6.69) as

\[
\tilde{u}_i^* = \begin{bmatrix}
\tilde{u}_{t_i}^* \\
\vdots \\
\tilde{u}_{t_i+N_i-1}^*
\end{bmatrix} = \tilde{Q}_{u,i}^T M \tilde{u}_i^* + \left( I - \tilde{Q}_{u,i}^T \tilde{Q}_{u,i} \right) \tilde{z}_i,
\]  

(6.79)

for some \( \tilde{z}_i \in \mathbb{R}^{m_i} \). Here, \( M \) in (6.79) is given by \( M \triangleq U \) if Lemma 6.14 is used to reduce the subproblem, and by \( M \triangleq S_i^T \) if Lemma 6.15 is used. When Lemma 6.15 is used, and if \( \tilde{Q}_{u,i} \) is singular, then the optimal control input \( \tilde{u}_i^* \) for the master problem is not unique according to Section 3.4.2 and Remark 6.17. However, \( \tilde{Q}_{u,i}^T S_i^T \tilde{u}_i^* \) is still unique, which follows from Lemma 6.19.

**Lemma 6.19.** Let \( \tilde{G}_{i+1} = \tilde{Q}_{u,i} + \tilde{B}_i^T \tilde{P}_{i+1} \tilde{B}_i \). Then \( \mathcal{N} \left( \tilde{G}_{i+1} \right) \subset \mathcal{N} \left( \tilde{Q}_{u,i}^T S_i \right) \).

**Proof:** When Lemma 6.15 is used to reduce the subproblem it follows that \( \tilde{B}_i = \tilde{Q}_{u,i} = S_i \tilde{Q}_{u,i}^T S_i \). Hence, Lemma 6.13 states that \( \mathcal{N} \left( \tilde{G}_{i+1} \right) = \mathcal{N} \left( \tilde{B}_i \right) \) holds. Let \( z \in \mathcal{N} \left( \tilde{G}_{i+1} \right) \) be arbitrary. Then, \( z \in \mathcal{N} \left( \tilde{B}_{i+1} \right) \iff z \in \mathcal{N} \left( S_i \tilde{Q}_{u,i}^T S_i \right) \). Since \( \tilde{Q}_{u,i} \in S_{+}^{m_i} \), Lemma A.2 states that \( \tilde{Q}_{u,i}^T S_i z = 0 \), and since \( z \in \mathcal{N} \left( \tilde{G}_{i+1} \right) \) was arbitrary it follows that \( \mathcal{N} \left( \tilde{G}_{i+1} \right) \subset \mathcal{N} \left( \tilde{Q}_{u,i}^T S_i \right) \). Furthermore, if \( \tilde{Q}_{u,i} \in S_{++}^{m_i} \) it follows that \( \mathcal{N} \left( \tilde{G}_{i+1} \right) = \{0\} \), which concludes the proof. \( \square \)

By using the solution (6.79) in (6.42), the state recursion on Line 4 in Algorithm 6 and the equation for the dual variables in (6.45), and also noting that \( x_t^* = \tilde{x}_t^* \) by definition, the primal and dual solution of subproblem \( i \) can be computed from

\[
\begin{align*}
\bar{u}_t^* &= k_{0,t+1} + K_{0,t+1} x_t^* + \tilde{v}_t^*, \quad t \in \mathbb{Z}_{t_i,t_i+N_i-1}, \\
x_{t+1}^* &= A_t x_t^* + B_t \bar{u}_t^* + \alpha_t, \quad t \in \mathbb{Z}_{t_i,t_i+N_i-2}, \\
\lambda_t^* &= P_{0,t} x_t^* - \Psi_{0,t} + D_t^T \tilde{\lambda}_{t+1}^*, \quad t \in \mathbb{Z}_{t_i+1,t_i+N_i-1}.
\end{align*}
\]

(6.80a, 6.80b, 6.80c)

Here, \( P_{0,t} \), \( K_{0,t+1} \), and \( k_{0,t+1} \) were computed when the subproblem was condensed, and

\[
D_t = \prod_{\tau=t}^{t_i+N_i-1} \left( A_\tau + B_\tau K_{0,\tau+1} \right),
\]

(6.81)
is computed as in Algorithm 11. Note that the Riccati factorization and backward recursion in algorithms 4 and 5 do not need to be re-computed.

For the last subproblem \( i = \bar{N} \), the solution can be computed using the state recursion in Algorithm 6 given the optimal \( \tilde{x}_N^* = x_N^* \), and the already computed
Riccati factorization as

\[ u_t^* = k_{0,t+1} + K_{0,t+1}x_t^*, \quad t \in \mathbb{Z}_{\hat{N},\hat{N}+N_{\hat{N}}-1}, \quad (6.82a) \]

\[ x_{t+1} = A_tx_t^* + B_tu_t^* + a_t, \quad t \in \mathbb{Z}_{t_{\hat{N}},t_{\hat{N}}+N_{\hat{N}}-2}, \quad (6.82b) \]

\[ \lambda_t^* = P_{0,t}x_t^* - \Psi_{0,t}, \quad t \in \mathbb{Z}_{t_{\hat{N}+1},t_{\hat{N}}+N_{\hat{N}}-1}. \quad (6.82c) \]

According to Assumption 6.10, whenever \( u_t^* \) is not unique it follows that \( Q_{u,t} = B_t \). Hence, from Lemma 6.13 it follows that \( N(G_{0,t+1}) = N(B_t) \). Using this and the fact that \( \hat{x}_i^* \) and \( \lambda_{i+1}^* \) are unique, it follows that the optimal states \( x_t^* \) and dual variables \( \lambda_t^* \) for \( t \in \mathbb{Z}_{t_i,t_i+N_i} \) are still unique.

### 6.3.6 Solving the subproblems using the cost-to-go function

The other approach to compute the solution to a subproblem uses the cost-to-go functions \( J(\hat{x}_{i+1}) \), which must be returned from the algorithm that is used to solve the master UFTOC problem in (6.72). The cost-to-go functions can be obtained by for example solving the master UFTOC problem using the Riccati recursion. Once the solution and cost-to-go functions have been obtained from the master UFTOC problem, the solution to each subproblem can be computed by solving (6.34) using the now known cost-to-go function \( J(\hat{x}_{i+1}) \) and initial value \( \hat{x}_i = \hat{x}_i^* \) from the master problem. Note that for the last subproblem, the solution can be obtained from (6.55) since the cost-to-go function is always known for the last subproblem according to Remark 6.8.

One way to solve the subproblems once \( \hat{x}_i^* \) and \( J(\hat{x}_{i+1}) \) are known is to use the Riccati recursion. From the definition of the subproblems and the notation \( \hat{P}_i = P_{t_i} \), etc., it follows that it is possible to compute the Riccati recursion for the original UFTOC problem in the interval \( t_i \leq t \leq t_{i+1} \). Hence, the Riccati recursion for the original UFTOC problem can be computed from the \( \hat{N} + 1 \) intervals \( i \in \mathbb{Z}_{0,\hat{N}} \) in parallel. Note that only the uniquely defined \( \hat{x}_i^* \) and \( J(\hat{x}_{i+1}) \) are used to compute the solution to the subproblem. Hence, the possibly non-unique \( \hat{u}_i^* \) in the master problem does not need to be considered when re-solving the subproblems using the cost-to-go function.

### 6.3.7 Communication

According to Theorem 6.16 and Remark 6.18 it is possible to form the master problem by condensing and reducing the subproblems in parallel, and to compute the solutions to the subproblems in parallel. Hence, the computations can be distributed on several different computational units which will need to communicate data with each other.

For each subproblem \( i \), the objective function and \( \hat{A}_i, \hat{B}_i \) and \( \hat{a}_i \) of the reduced subproblem need to be sent to the computation unit that is forming the master problem. While exploiting that some of the matrices are symmetric, the amount of data that needs to be sent from each subproblem when it is reduced as in Lemma 6.12 is presented in Table 6.3. In Table 6.4 the amount of data that needs
Table 6.3: The amount of data that is sent from each subproblem to form the master problem when Lemma 6.12 is used. Here, the number of real numbers is presented, not the total number of bytes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{Q}<em>{x,i} \in S</em>{++}^{n_x}$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{u,i} \in S</em>{++}^{n_{\hat{u},i}}$</td>
<td>$(n_{\hat{u},i}^2 + n_{\hat{u},i}) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{xu,i} \in \mathbb{R}^{n_x \times n</em>{\hat{u},i}}$</td>
<td>$n_x n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{l}_{x,i} \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{l}<em>{u,i} \in \mathbb{R}^{n</em>{\hat{u},i}}$</td>
<td>$n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{c}_i \in \mathbb{R}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\hat{A}_i \in \mathbb{R}^{n_x \times n_x}$</td>
<td>$n_x^2$</td>
</tr>
<tr>
<td>$\hat{B}<em>i \in \mathbb{R}^{n_x \times n</em>{\hat{u},i}}$</td>
<td>$n_x n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{a}_i \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>Total</td>
<td>$3/2n_x^2 + 5/2n_x + 1/2n_{\hat{u},i}^2 + 3/2n_{\hat{u},i} + 2n_x n_{\hat{u},i} + 1$</td>
</tr>
</tbody>
</table>

to be communicated when Lemma 6.14 is used to reduce a subproblem is presented, and in Table 6.5 the data that need to be communicated when Lemma 6.15 is used to reduce a subproblem is presented. Note that when Lemma 6.15 is used, $B_{v,i}$ is not sent since $B_{v,i} = \hat{Q}_{u,i}$ by definition.

Depending on which approach that is used to solve a subproblem, different data is required from the master problem. The data that is used when solving a subproblem using the solution of the master problem is presented in Table 6.6, and the data that is needed when re-solving the subproblem using the cost-to-go function is presented in Table 6.7, respectively. For the last subproblem, only the initial state $\hat{x}_N^* \in \mathbb{R}^{n_x}$ needs to be communicated.

6.4 Computing the Search Direction in Parallel

In sections 6.2 and 6.3, different approaches for both reducing the original UFTOC problem (6.1) to a master UFTOC problem, and for computing the solution to the original UFTOC problem using the solution to the master UFTOC problem were presented. Since all subproblems can be reduced independently of each other, the construction of the master UFTOC problem can be computed in parallel on several computational units. Furthermore, once the solution to the master problem is computed, the subproblems can be solved independently of each other, which can also be done in parallel on several computational units. Hence, both forming the master problem and solving the subproblems can be done in parallel. The solution
Table 6.4: The amount of data that is sent from each subproblem to form the master problem when Lemma 6.14 is used. Here, the number of real numbers is presented, not the total number of bytes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{Q}<em>{x,i} \in S^{nx}</em>+$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{u,i} \in S^{n\hat{u},i}</em>{++}$</td>
<td>$(n_{\hat{u},i}^2 + n_{\hat{u},i}) / 2$</td>
</tr>
<tr>
<td>$\hat{l}_{x,i} \in \mathbb{R}^{nx}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{c}_i \in \mathbb{R}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\hat{A}_i \in \mathbb{R}^{nx \times nx}$</td>
<td>$n_x^2$</td>
</tr>
<tr>
<td>$\hat{B}_i \in \mathbb{R}^{nx \times n\hat{u},i}$</td>
<td>$n_x n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{a}_i \in \mathbb{R}^{nx}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$3/2n_x^2 + 5/2n_x + 1/2n_{\hat{u},i}^2 + 1/2n_{\hat{u},i} + n_x n_{\hat{u},i} + 1$</td>
</tr>
</tbody>
</table>

Table 6.5: The amount of data that is sent from each subproblem to form the master problem when using Lemma 6.15. Here, the number of real numbers is presented, not the number of bytes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{Q}<em>{x,i} \in S^{nx}</em>+$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td>$\hat{Q}<em>{u,i} \in S^{nx}</em>+$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td>$\hat{l}_{x,i} \in \mathbb{R}^{nx}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{c}_i \in \mathbb{R}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\hat{A}_i \in \mathbb{R}^{nx \times nx}$</td>
<td>$n_x^2$</td>
</tr>
<tr>
<td>$\hat{a}_i \in \mathbb{R}^{nx}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$2n_x^2 + 3n_x + 1$</td>
</tr>
</tbody>
</table>
Table 6.6: The amount of data sent to the subproblem when solving it with the approach using the solution from the master problem. Here, the number of real numbers is presented, not the total number of bytes. Note that for the case where Lemma 6.15 is used $n_{\hat{u},i} = n_x$ by definition.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_i^* \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{u}<em>i^* \in \mathbb{R}^{n</em>{\hat{u},i}}$</td>
<td>$n_{\hat{u},i}$</td>
</tr>
<tr>
<td>$\hat{\lambda}_{i+1}^* \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$2n_x + n_{\hat{u},i}$</td>
</tr>
</tbody>
</table>

Table 6.7: The amount of data sent to the subproblem when solving it with the approach using the cost-to-go function. Here, the number of real numbers is presented, not the total number of bytes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number of real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_i^* \in \mathbb{R}^{n_x}$</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$\hat{P}<em>{i+1} \in \mathbb{S}^{n_x}</em>+$</td>
<td>$(n_x^2 + n_x) / 2$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$1/2n_x^2 + 3/2n_x$</td>
</tr>
</tbody>
</table>
6.4 Computing the Search Direction in Parallel

\[ \mathcal{P}(\hat{N}_{m-1}) : \quad \mathcal{P}_0^m(N_0^m) \]

\[ \mathcal{P}(\hat{N}_0) : \quad \mathcal{P}_0^1(N_0^1) \quad \ldots \quad \mathcal{P}_{N_1}^1(N_1^1) \]

\[ \mathcal{P}(N) : \quad \mathcal{P}_0^0(N_0^0) \quad \ldots \quad \mathcal{P}_i^0(N_i^0) \quad \mathcal{P}_j^0(N_j^0) \quad \ldots \quad \mathcal{P}_{N_0}^0(N_0^0) \]

**Figure 6.2:** The tree structure that is obtained when a UFTOC problem is reduced in several steps. Each level in the tree forms a UFTOC problem that is again split into several smaller problems. \( \mathcal{P}_k^i(N_k^i) \) denotes a subproblem at level \( k \) in the tree in either of the forms (6.9) or (6.34) with prediction horizon \( N_k^i \).

One of the key benefits with the reduction techniques presented in this thesis and in Nielsen and Axehill (2014), Nielsen (2015), and Nielsen and Axehill (2015) is that the master problem is in the same form as the original UFTOC problem. Furthermore, the master problem also satisfies the same assumptions as the UFTOC problem that is reduced. As a consequence, instead of solving the master UFTOC problem in Figure 6.1 serially, it can itself be reduced recursively in several a priori determined steps to an even smaller UFTOC problem. This small UFTOC problem can be solved serially, and the solution can be propagated to all the intermediate UFTOC problems to finally compute the solution to the original UFTOC problem. This procedure can be described by the tree in Figure 6.2, where each level \( k \) in the tree is a UFTOC problem with horizon \( \hat{N}_{k-1} \) and the original UFTOC problem (6.1) is at the bottom level in the tree. Furthermore, the UFTOC problem at each level is split into \( \hat{N}_k \) subproblems which are denoted \( \mathcal{P}_i^k(N_k^i) \) for \( i \in \mathbb{Z}_{0,\hat{N}_k} \) and are depicted as the rounded boxes in the tree in Figure 6.2. Hence, each level \( k \) and its parent level \( k+1 \) in the tree has the structure as in Figure 6.1. Communication is only needed between the parent and child nodes in the tree, and hence each level \( k \) can be solved completely in parallel provided that \( \hat{N}_k \) computational units are available.

**Remark 6.20.** Note that the sizes of different subproblems in Figure 6.2 can be different, both within the same level and between levels. Hence, depending on for example the communication layout on the computational cluster, different sizes of the subproblems can be used to reduce the total computation time.
6.4.1 Algorithms for parallel computation of the search directions

The parallel computation of the search directions by recursively solving UFTOC problems of decreasing size can be divided into two main steps:

1. **Reduction step**: This step consists of recursively reducing the UFTOC problems in \( m \) steps upwards in the tree in Figure 6.2, which is summarized in Algorithm 12.

2. **Solution step**: This step consists of propagating the solution to the UFTOC problems in \( m \) steps downwards in the tree in Figure 6.2, which is summarized in Algorithm 13.

Note that the parfor-loops in algorithms 12 and 13 can be computed in parallel using several computation units.

**Algorithm 12** Parallel recursive reduction of UFTOC problems

1: Set the maximum level number \( m \)
2: Set the number of subproblems \( \hat{N}_k + 1 \) for each level \( k \in \mathbb{Z}_{0,m} \), with \( \hat{N}_m = 0 \)
3: for \( k = 0, \ldots, m - 1 \) do
4:     parfor \( i = 0, \ldots, \hat{N}_k \) do
5:         Create subproblem \( P^k_i(N^k_i) \)
6:         Reduce subproblem \( P^k_i(N^k_i) \)
7:         Send the reduced problem matrices to the parent
8:     end parfor
9: end for
10: Create the top subproblem \( P^m_0(N^m_0) \)

**Algorithm 13** Parallel propagation of solutions to UFTOC problems

1: Get the maximum level number \( m \) and all \( \hat{N}_k \)'s from Algorithm 12
2: for \( k = m, m - 1, \ldots, 0 \) do
3:     parfor \( i = 0, \ldots, \hat{N}_k \) do
4:         Solve subproblem \( P^k_i(N^k_i) \)
5:         if \( k \neq 0 \) then
6:             Send solution information to each children at level \( k - 1 \)
7:         end if
8:     end parfor
9: end for
10: Retrieve the solution of (6.1) from the solutions of \( P^0_i(N^0_i) \) for \( i \in \mathbb{Z}_{0,\hat{N}_0} \)

When computing the search direction in an AS method as described in Section 3.2.2, the eliminated dual variables are computed as in Section 3.5 using Algorithm 7. Given the primal and dual solution to the UFTOC problem, it can be seen that also Algorithm 7 can be executed in parallel for the independent intervals \( i \in \mathbb{Z}_{0,\hat{N}} \).
So far no assumptions on the length of the prediction horizon of each subproblem have been made. Now, assume for simplicity that the length of the subproblems are all equal to $N_s$, and that $N = N_s^{m+1}$ for some $m \in \mathbb{Z}_{++}$. Then, if there are $N_s^n$ computational units available, both the reduction and the propagation of the solution can be done in $m$ steps. Hence, the solution to the original UFTOCP problem can be computed in $O(\log N)$ computational complexity growth. The optimal choice of $N_s$ for each subproblem can be adjusted to fit the hardware on which the algorithms are implemented. Depending on for example the number of computational units, the available memory and the communication overhead between processors, the size of $N_s$ can be chosen differently. How to optimally adjust the size of the subproblems is not investigated here, but it could possibly be done by for example benchmarking the algorithms using the desired hardware and mapping the performance versus different sizes of the subproblems.

At level $k$ in the tree in Figure 6.2 only $\hat{N}_k$ computational units are used for reducing or solving subproblems. However, to use the computational resources more efficiently, it is possible to for example use standard parallel linear algebra routines in the reduction and the solution of the subproblems to make sure that all computational units are always busy.

The main steps in algorithms 12 and 13 are the same for the approaches presented in Section 6.2 and Section 6.3. However, the details on what is sent and how the reduction and propagation of the solutions are made differ between them. These details are given below.

**Parametric programming approach**

For the parametric programming approach presented in Section 6.2, the reduction of the subproblems at Line 6 in Algorithm 12 is done as described in Section 6.2.1 to form the master UFTOCP problem as in Theorem 6.5. The data that is communicated to the parent at Line 7 in Algorithm 12 is given in Table 6.1.

The UFTOCP problem $\mathcal{P}(\hat{N}_{m-1})$ at the top level $m$ in the tree in Figure 6.2 can be solved using any method for solving UFTOCP problems in the form (6.1). The UFTOCP problem at the level below in the tree can be solved as in Theorem 6.7. Hence, the primal solutions $\theta^*_i$ need to be communicated to each child in the tree. For each child that is primal degenerate, also the dual solution needs to be communicated. The data that is communicated to the children at Line 6 in Algorithm 13 is given in Table 6.2.

**Parallel partial condensing and reduction approach**

In the approach presented in Section 6.3 it is not only shown how to compute the solution to the UFTOCP problem (6.1), but also how to compute the Riccati factorization in parallel. Depending on which condensing and reduction technique from Section 6.3 that has been used, different sets of data are used in Algorithm 12 and in Algorithm 13. Furthermore, at Line 10 in Algorithm 13 the solution to the original UFTOCP problem is retrieved as in (6.76). Note that if the subproblems at the bottom level in the tree in Figure 6.2 are solved using the Riccati recursion,
then it follows that both the Riccati factorization and the full Riccati recursion for the original UFTOC problem (6.1) are computed in parallel.

Communication in Algorithm 12:

When a subproblem is condensed and the control input dimension is reduced using lemmas 6.9 and 6.12 at Line 6 in Algorithm 12, the master UFTOC problem satisfies Assumption 3.4. Hence, a recursive reduction as in Algorithm 12 is possible. The data that is communicated to the parent node at Line 7 in Algorithm 12 is given in Table 6.3.

If the condensation and the reduction of the control input dimensions at Line 6 in Algorithm 12 is done using Lemma 6.11, and either Lemma 6.12 (when possible) or Lemma 6.14, then the data that is communicated at Line 7 in Algorithm 12 is given in Table 6.4. Since $\hat{Q}_{xu,i} = 0$ and $\hat{l}_{u,i} = 0$, less data need to be communicated compared to the parametric programming approach and the regular condensing as in Lemma 6.9. This condensation and reduction is done such that the master UFTOC problem satisfies Assumption 3.4, and hence it can be applied recursively.

When the reduction of the subproblems at Line 6 in Algorithm 12 is done using Algorithm 11, the condensation and reduction of the control input dimensions are done as in Lemma 6.11 and Lemma 6.15. For this case it is not certain that $\hat{Q}_{u,i} \in S_{++}^{n_x}$. For subproblems with colrank $S_i < n_x$ it follows from Lemma 6.15 that $\hat{Q}_{u,i} = \hat{B}_i \in S_{++}^{n_x}$, and hence the master UFTOC problem satisfies Assumption 6.10. In order to apply Algorithm 11 recursively as in Figure 6.2, the case where Assumption 6.10 holds must be handled. This is the motivation why Assumption 6.10 is considered in this thesis. In Table 6.4, the data that needs to be communicated to the parent node is presented. Besides exploiting that $\hat{Q}_{xu,i} = 0$ and $\hat{l}_{u,i} = 0$, it also follows that $\hat{Q}_{u,i} = \hat{B}_i$ and hence there might be even more savings in terms of communication compared to the other approaches, as is seen in Table 6.5.

Communication in Algorithm 13:

The solution of the subproblems in the tree can be computed in two different ways for this approach: either by substituting the solution from the problem at the level above as described in Section 6.3.5, or by re-solving the problem using the cost-to-go function $J(\hat{x}_{i+1})$ as described in Section 6.3.6.

For the first case, the top problem $\mathcal{P}(\hat{N}_{m-1})$ can be computed using any method for UFTOC problems in the form (6.1). For any subproblem in the tree, the solution can be computed from (6.78) or (6.80), depending on the condensation and reduction technique that was applied in Algorithm 12. For the last subproblem at each level in the tree, the solution is instead computed from (6.55) or (6.82). The data that is communicated to the children at Line 6 in Algorithm 13 for this case is given in Table 6.6. Note that for the last subproblem, only $\hat{x}_N^* \in \mathbb{R}^{n_x}$ needs to be sent.

If the subproblems are re-solved using the cost-to-go function, then all subproblems, including the top problem $\mathcal{P}(\hat{N}_{m-1})$, must be solved with an algorithm that computes the cost-to-go function for each time instance in the UFTOC pro
blem. One example of such a method is the Riccati recursion. For level $k$ in the tree, at Line 4 in Algorithm 13 the subproblems $\mathcal{P}_i^k(N_i^k)$ given by (6.34) are hence solved using the cost-to-go function $J(\hat{x}_{i+1})$, which is provided by the parent at level $k + 1$. At Line 6 in Algorithm 13 the optimal state and cost-to-go function computed at level $k$ need to be sent to the children, see Table 6.7.

6.5 Numerical Results

In this section the numerical evaluations of the algorithms are presented. The algorithm based on the parametric approach has been implemented in MATLAB, and the algorithm based on the parallel Riccati approach using Algorithm 11 for the reduction of the subproblems and re-computing the solution using the cost-to-go function has been implemented in both MATLAB and ANSI-C. The parallel algorithms are compared to the state-of-the-art serial Riccati recursion which is implemented in both MATLAB and ANSI-C. Furthermore, in the numerical evaluation of the algorithms for linear MHE problems they are also compared to the well-known RTS smoother (Kailath et al., 2000) in MATLAB. Note, however, that although the Riccati algorithms compute the same estimate as the RTS smoother, they do not compute the arrival cost which is done in the RTS smoother. All of the implementations are done by the author of this thesis. In all numerical experiments, the length of each subproblem $N_s = 2$ and $N/2$ number of computational units have been used. Note, however, that different batch lengths can be used for each subproblem in the tree as is mentioned in Remark 6.20. Similarly as in Axehill (2015), the optimal choice probably depends on for example the problem and the hardware on which the algorithm is implemented. How to choose these to minimize computation time is outside the scope of this thesis and left as future work.

For the evaluation of the MATLAB implementation of the algorithms, the parallelism is simulated by executing the algorithms serially using only one computational thread but still using the same information flow as for an actual parallel execution. The computation time for these evaluations are estimated by computing the sum of the maximum computation times for each level in the tree in Figure 6.2. Hence, for the evaluations in MATLAB the communication overhead is neglected.

The ANSI-C implementation of the parallel Riccati algorithm has been executed truly in parallel on a computer cluster consisting of nodes with 8-core Intel Xeon E5-2660 @ 2.2 GHz cpus with communication over TCP/IP on Gigabit Ethernet or Mellanox InfiniBand FDR high-speed interconnect (depending on choice). The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at NSC. Hence, the evaluation of the ANSI-C implementation of the parallel Riccati algorithm includes also the communication overhead and shows the actual potential benefits using the parallel Riccati recursion compared to the serial one. The ANSI-C implementation uses two different implementations for communication: one based on TCP/IP and one based on communication over infiniband (IB) using the message-passing interface (MPI). Both
implementations are rudimentary, where for example no tuning of MPI parameters or TCP/IP settings have been made. However, the implementations serve as a proof-of-concept that the proposed parallel Riccati algorithm actually increases performance in terms of computation time when executed truly in parallel on physical hardware, where the communication time is taken into consideration. The computation times for the ANSI-C implementation have been calculated using wall-clock time. Note that the scaling on the y-axis of Figure 3 in Nielsen and Axehill (2015) and Figure 6.5 in Nielsen (2015) are incorrect and must be multiplied by a factor $10^3$ to be correct.

In MATLAB, the algorithms have been evaluated by computing the solution to UFTOC problems with stable LTI systems of varying sizes and with varying prediction horizons. The evaluations have been performed on an Intel Core i7-3517U CPU @ 1.9GHz running Windows 7 (version 6.1, build 7601: Service Pack 1) and MATLAB (8.0.0.783, R2012b). The computation time is obtained by computing the average time for solving 10 problems of the same dimension. The resulting computation times for a UFTOC problem with $n_x = 7$ and $n_u = 5$ are presented in Figure 6.3, and for a UFTOC problem with $n_x = 20$ and $n_u = 20$ in Figure 6.4. From the figures it is clear that the parallel Riccati algorithm outperforms the serial Riccati recursion for $N \geq 18 - 20$ for both problem sizes. The parallel Riccati algorithm is outperforming the parallel algorithm based on the parametric programming approach presented in Section 6.2 for both problem sizes. For the problem in Figure 6.4 the parallel Riccati algorithm is approximately three times faster than the parallel algorithm based in Section 6.2. This significant boost in performance is possible since the parallel Riccati recursion presented in Section 6.3 utilizes the structure in the subproblems.

The evaluation of the MATLAB implementations of the algorithms for linear MHE problems as in (3.9) are presented in Figure 6.5, where stable LTI systems of dimension $n_x = 20$, $n_w = 20$ and $n_y = 20$ have been used. In the figure, the computation times for the parallel Riccati recursion, the serial Riccati recursion and the commonly used RTS smoother are presented. To make a fair comparison of the algorithms, the computation time that is required to form the UFTOC problem from the MHE problem is included in the reported computation times for the parallel and serial Riccati recursions. It is seen that for this implementation of the parallel Riccati algorithm, it outperforms the serial Riccati for $N \geq 20$ and the RTS smoother for $N \geq 30$.

The ANSI-C implementation of the parallel Riccati algorithm has been evaluated by averaging the computation time when solving 15 UFTOC problems of the same size for stable LTI systems of dimension $n_x = 20$ and $n_u = 20$ for different prediction horizons. Both of the communication implementations have been evaluated and compared to a serial Riccati recursion, see Figure 6.6. In this figure, the communication times are included and for the implementation using TCP/IP the communication overhead is around 20% of the total computation time for this problem size. From the figure it can be seen that the implementation using TCP/IP solves a problem with $N = 512$ approximately as fast as the serial Riccati solves a
6.5 Numerical Results

**Figure 6.3:** Computation times when solving uftoc problems of order \( n_x = 7 \) and \( n_u = 5 \). The parallel Riccati outperforms the serial Riccati for \( N \gtrsim 18 - 20 \), and it is always faster than the parallel algorithm based on the parametric programming approach presented in Section 6.2. © 2015 IEEE

The parallel Riccati algorithm using TCP/IP is faster than the serial Riccati for \( N \gtrsim 23 \). From Figure 6.6 it can also be seen that the parallel Riccati algorithm using MPI for communication only requires around 65\% - 85\% of the total computation time required by the implementation using TCP/IP, with the largest improvement for long horizons. The implementation using MPI solves a uftoc problem with \( N = 512 \) roughly as fast as the implementation using TCP/IP solves one with \( N = 150 \), and the serial Riccati solves one with \( N = 45 \). The parallel Riccati algorithm using MPI outperforms the serial Riccati recursion for \( N \gtrsim 18 \), which is similar to the results for the MATLAB implementations where the communication overhead is neglected. This speed-up can be important in for example optimal control for motion planning problems (LaValle, 2006; Ljungqvist et al., 2017; Bergman and Axehill, 2017), and MHE problems where long horizons are often used (Rao et al., 1998).
Figure 6.4: Computation times when solving UFTOC problems of order $n_x = 20$ and $n_u = 20$. The parallel Riccati outperforms the serial Riccati for $N \gtrsim 18 - 20$, and it is significantly faster than the parallel algorithm based on the parametric programming approach presented in Section 6.2. © 2015 IEEE
Figure 6.5: Computation times for solving linear MHE problems of dimension $n_x = n_w = n_y = 20$. For the parallel and serial Riccati recursions, the time it takes to convert the linear MHE problem into a UFTOC problem is included in the computation times to make a fair comparison against the RTS smoother. The parallel Riccati recursion outperforms the commonly used RTS algorithm for $N \gtrsim 32$ and it solves a linear MHE problem with $N = 512$ approximately as fast as the RTS algorithm solves one for $N = 64$. © 2016 IEEE
Figure 6.6: Computation times for the ANSI-C implementation of the parallel Riccati algorithm using the different communication implementations. It can be seen that the parallel Riccati using MPI over IB only requires around 65%–85% of the total computation time when using the TCP/IP implementation. Furthermore, the parallel Riccati is outperforming the serial Riccati for \( N \gtrapprox 18 \) when communicating over IB using MPI, which is in the same order as the results for the MATLAB implementation in Figure 6.4. The parallel Riccati using MPI solves a UFTOC problem with \( N = 512 \) roughly as fast as the serial Riccati solves one with \( N = 45 \). © 2016 IEEE
Appendix

In this appendix, the matrices, vectors and constants for some of the problems introduced in this chapter are presented. Furthermore, the proofs for some of the lemmas and theorems are also provided in this appendix.

6.A Definition of Problem Matrices

In this section, the definitions of the problem matrices, vectors and other variables in Chapter 6 are presented.

6.A.1 Definition of the matrices in the UFTOC problem (6.34)

The problem matrices in the UFTOC problem (6.34) are defined as

\[ Q_{x,i} \triangleq \begin{bmatrix} Q_{x,t_i+1} & \cdots & Q_{x,t_i+N_i-1} \end{bmatrix}, \quad Q_{u,i} \triangleq \begin{bmatrix} Q_{u,t_i} & \cdots & Q_{u,t_i+N_i-1} \end{bmatrix}, \]

\[ Q_{xu,i} \triangleq \begin{bmatrix} Q_{xu,t_i} & \cdots & 0 \end{bmatrix}, \quad Q_{xu,i} \triangleq \begin{bmatrix} Q_{xu,t_i+1} & \cdots & Q_{xu,t_i+N_i-1} \end{bmatrix}, \]

\[ \bar{c}_i \triangleq \sum_{t=t_i}^{t_i+N_i-1} c_t, \quad A_i \triangleq \begin{bmatrix} I & \cdots & I \\ -A_{t_i+1} & I & \cdots \\ \cdots & \cdots & \cdots \\ -A_{t_i+N_i-2} & I \\ 0 & \cdots & 0 \end{bmatrix}, \quad A_{0,i} \triangleq \begin{bmatrix} A_{t_i} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \]

(6A.83a)
\[ \begin{bmatrix} B_{t_i} \\ \vdots \\ B_{t_i+N_i-2} \end{bmatrix}, \quad a_i \triangleq \begin{bmatrix} a_{t_i} \\ \vdots \\ a_{t_i+N_i-2} \end{bmatrix}, \quad (6A.83d) \]

\[ A_{N,i} \triangleq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \quad B_{N,i} \triangleq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \quad a_{N,i} \triangleq a_{t_i+N_i-1}. \quad (6A.83e) \]

### 6.A.2 Definition of the matrices used in Lemma 6.9

The matrices in the problem (6.37) in Lemma 6.9 are defined as

\[ \tilde{Q}_0 \triangleq Q_0 + \Omega_0^T \Omega^{-T} Q_x \Omega^{-1} \Omega_0, \quad (6A.84a) \]
\[ \tilde{Q}_u \triangleq Q_u + S^T \Omega^{-1} \Gamma + \Gamma^T \Omega^{-T} S + \Gamma^T \Omega^{-T} Q_x \Omega^{-1} \Gamma, \quad (6A.84b) \]
\[ \tilde{R} \triangleq R + \Omega_0^T \Omega^{-T} S + \Omega_0^T \Omega^{-T} Q_x \Omega^{-1} \Gamma, \quad (6A.84c) \]
\[ \tilde{f}_0 \triangleq f_0 + \Omega_0^T \Omega^{-T} f_x + \Omega_0^T \Omega^{-T} Q_x \Omega^{-1} \gamma, \quad (6A.84d) \]
\[ \tilde{f}_u \triangleq f_u + S^T \Omega^{-1} \gamma + \Gamma^T \Omega^{-T} f_x + \Gamma^T \Omega^{-T} Q_x \Omega^{-1} \gamma, \quad (6A.84e) \]
\[ \tilde{c} \triangleq c + \frac{1}{2} f_x^T \Omega^{-1} \gamma + \frac{1}{2} \gamma^T \Omega^{-T} Q_x \Omega^{-1} \gamma, \quad (6A.84f) \]
\[ \tilde{\Omega}_N \triangleq \Omega_N \Omega^{-1} \Omega_0, \quad \tilde{\Gamma}_N \triangleq \Gamma_N + \Omega_N \Omega^{-1} \Gamma, \quad \tilde{\gamma}_N \triangleq \Omega_N \Omega^{-1} \gamma. \quad (6A.84g) \]

### 6.A.3 Definition of the matrices used in Lemma 6.12

The matrices in the problem (6.46) in Lemma 6.12 are defined as

\[ \hat{Q}_0 \triangleq Q_0 - RV \left( V^T Q_u V \right)^{-1} V^T R, \quad (6A.85a) \]
\[ \hat{Q}_u \triangleq U^T Q_u U - U^T Q_u V \left( V^T Q_u V \right)^{-1} V^T Q_u U, \quad (6A.85b) \]
\[ \hat{R} \triangleq RU - RV \left( V^T Q_u V \right)^{-1} V^T Q_u U, \quad (6A.85c) \]
\[ \hat{f}_0 \triangleq f_0 - RV \left( V^T Q_u V \right)^{-1} V^T f_u, \quad (6A.85d) \]
\[ \hat{f}_u \triangleq U^T f_u - U^T Q_u V \left( V^T Q_u V \right)^{-1} V^T f_u, \quad (6A.85e) \]
\[ \hat{c} \triangleq c - \frac{1}{2} f_u^T V \left( V^T Q_u V \right)^{-1} V^T f_u, \quad (6A.85f) \]
\[ \hat{\Omega} \triangleq \Omega, \quad \hat{\Gamma} \triangleq \Gamma U, \quad \hat{\gamma} \triangleq \gamma, \quad (6A.85g) \]

where the columns of \( U \) form an orthonormal basis for \( \mathcal{R}(\Gamma^T) \) and the columns of \( V \) form an orthonormal basis for \( \mathcal{N}(\Gamma) \), with \( \Gamma \) from (6.46).
6.B Proofs of theorems and lemmas in Section 6.2

In this section, the proofs of some of the lemmas and theorems for the parametric programming approach are presented.

6.B.1 Proof of Lemma 6.4

The proof of Lemma 6.4 will be done in three main steps: re-write (6.9) into an equivalent mp-QP problem, relate $\hat{V}_i(\theta)$ in (6.20a) between the equivalent problems, and finally show that $\hat{Q}_u \succ 0$ must hold due to the structure of the introduced parametrization.

**Proof:** Consider the problem (6.9) where the subindex “i” is omitted for brevity. Let $\Pi$ be a permutation matrix, satisfying $\Pi^T\Pi = I$, such that

$$
\Pi X = \begin{bmatrix} x \\ u \end{bmatrix}, \quad \Pi H \Pi^T = \begin{bmatrix} H_x & H_{xu} \\ H_{xu} & H_u \end{bmatrix},
$$

where the elements in $x$ and $u$ are the vectors $x_t$ and $u_t$, respectively, and $H_u = \text{blkdiag}(Q_{u,0}, \ldots, Q_{u,N-1}) \succ 0$ by definition. In Åström and Wittenmark (1984) it is shown that by introducing the change of variables

$$
\begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} I & 0 \\ -H_u^{-1}H_{xu} & I \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} \quad \Leftrightarrow \quad \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} = \begin{bmatrix} I & 0 \\ H_u^{-1}H_{xu} & I \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix},
$$

the problem (6.9) can be re-formulated into the equivalent problem

$$
\begin{align*}
\text{minimize} & \quad \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix}^T \begin{bmatrix} H/H_u & 0 \\ 0 & H_u \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} + l^T \Pi^T T \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} + c \\
\text{subject to} & \quad A \Pi^T T \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} = b + G \theta,
\end{align*}
$$

where $H/H_u \succeq 0$, $H_u \succ 0$, and the cross-terms are zero. From Tøndel et al. (2003a) it follows that the unique primal parametric solution to (6B.88) is given by an affine function

$$
\begin{bmatrix} \bar{x}^*(\hat{x}, \hat{u}) \\ \bar{u}^*(\hat{x}, \hat{u}) \end{bmatrix} \triangleq \begin{bmatrix} \Omega_x \\ \Omega_u \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{u} \end{bmatrix} + \begin{bmatrix} \omega \\ \gamma \end{bmatrix},
$$

where $\theta = [\hat{x}^T \hat{u}^T]^T$ is used. Now, by using the definition of $\hat{V}(\theta)$ in (6.20a), the permutation matrix $\Pi$, and the transformation $T$ in (6B.87), it follows that

$$
\hat{V}(\hat{x}, \hat{u}) \triangleq \hat{V}(\theta) = \frac{1}{2} \begin{bmatrix} x^*(\theta) \\ u^*(\theta) \end{bmatrix}^T \begin{bmatrix} H_x & H_{xu} \\ H_{xu} & H_u \end{bmatrix} \begin{bmatrix} x^*(\theta) \\ u^*(\theta) \end{bmatrix} + l^T \Pi^T T \begin{bmatrix} x^*(\theta) \\ u^*(\theta) \end{bmatrix} + c \quad \text{(6B.87)}
$$

By substituting the parametric solution (6B.89) into (6B.90) and expanding all
expressions, the Hessian of $\hat{V}(\hat{x}, \hat{u})$ with respect to $\hat{u}$ can be identified as
\[
\nabla^2 \hat{V}(\hat{x}, \hat{u}) = \hat{Q}_u = \Gamma^T_u H_u \Gamma_u + \Omega^T_u (H/H_u) \Omega_u. \tag{6B.91}
\]

Hence, if $\Gamma^T_u H_u \Gamma_u \succ 0$ then $\hat{Q}_u \succ 0$ since $\Omega^T_u (H/H_u) \Omega_u \succeq 0$ by definition.

The final step is to show that $\Gamma^T_u H_u \Gamma_u \succ 0$ is satisfied, which is equivalent to that $\mathcal{N}(\Gamma_u) = \{0\}$ since $H_u \succ 0$ by definition (Lancaster and Tismenetsky, 1985). Now assume, for contradiction, that there exists a $z \in \mathcal{N}(\Gamma_u)$ such that $z \neq 0$. Let $\hat{u}$ and $\hat{x}$ be arbitrary parameters, and define the parameter $\hat{z} \triangleq \hat{u} + z$. Then,
\[
\hat{u}^*(\hat{x}, \hat{z}) = \Gamma x \hat{x} + \Gamma u \hat{u} + z + \gamma = \Gamma x \hat{x} + \Gamma u \hat{u} + \gamma = \hat{u}^*(\hat{x}, \hat{u}), \tag{6B.92}
\]
holds. By expressing the terminal constraint in (6B.88) equivalently as in (6.10) and exploiting that $\hat{u}^*(\theta)$ satisfies the equality constraints (specifically the terminal constraint) in the mp-qp problem (6B.88), it follows from (6B.92) that
\[
\hat{A} \hat{x} + \hat{B} \hat{u} + \hat{a} = \hat{A} \hat{x} + \hat{S} \hat{u}^*(\hat{x}, \hat{u}) + \hat{D} \hat{a} = \hat{A} \hat{x} + \hat{B} \hat{z} + \hat{a} \implies \hat{B} \hat{u} = \hat{B} \hat{z}. \tag{6B.93}
\]

By inserting the definition of $\hat{z}$ into the last equation in (6B.93), it follows that
\[
\hat{B} \hat{u} = \hat{B} \hat{u} + \hat{B} \hat{z} \iff \hat{B} \hat{z} = 0 \iff z = 0, \tag{6B.94}
\]
where the last equivalence follows from that $\hat{B}$ has full column rank from the definition of $\hat{B}$ in (6.13). However, $z = 0$ contradicts the assumption of the existence of a $z \in \mathcal{N}(\Gamma_u)$ such that $z \neq 0$. Hence, it follows that $\mathcal{N}(\Gamma_u) = \{0\} \iff \Gamma^T_u H_u \Gamma_u \succ 0 \iff \hat{Q}_u \succ 0$, which concludes the proof. \hfill \Box

### 6B.2 Proof of Theorem 6.7

The original UFTOC problem is given by (6.1). Let the dual variables $\lambda_t$, corresponding to the dynamics constraints, be defined as in (3.32). By writing down the equations of the KKT system for the original UFTOC problem (6.1) it is seen that the following equations hold for a primal and dual solution
\[
\begin{align*}
Q_{x,t}x_t + Q_{u,t}u_t + l_{x,t} - \lambda_t + A^T_t \lambda_{t+1} &= 0, \quad t \in \mathbb{Z}_{0,N-1}, \quad \text{(6B.95a)} \\
Q^T_{x,t}x_t + Q_{u,t}u_t + l_{u,t} + B^T_t \lambda_{t+1} &= 0, \quad t \in \mathbb{Z}_{0,N-1}, \quad \text{(6B.95b)} \\
Q_{x,N}x_N + l_{x,N} - \lambda_N &= 0, \quad \text{(6B.95c)} \\
x_{t+1} = A_t x_t + B_t u_t + a_t, \quad t \in \mathbb{Z}_{0,N-1}, \quad \text{(6B.95d)} \\
x_0 &= \bar{x}_0. \quad \text{(6B.95e)}
\end{align*}
\]

The KKT conditions for the extended problem (6.7) are given by the set of equations
\[
\begin{align*}
H_i X_i + A^T_i \Lambda_i &= -l_i, \quad i \in \mathbb{Z}_{0,N}, \quad \text{(6B.96a)} \\
A_i X_i - G_i \begin{bmatrix} \hat{x}_i \\ \hat{u}_i \end{bmatrix} &= b_i, \quad i \in \mathbb{Z}_{0,N-1}, \quad \text{(6B.96b)}
\end{align*}
\]
where $X_i$, $\Lambda_i$ and $\hat{\lambda}_i$ are defined as in (6.6c), (6.16) and (6.23), respectively.

**Proof:** From the definition of the initial and terminal constraint in each subproblem (6.9), and the coupling constraints in (6.7) it follows that

$$x_{N_i,i}^* = \hat{A}_i \hat{x}_i^* + \hat{B}_i \hat{u}_i^* + \hat{a}_i = \hat{x}_{i+1}^* = x_{0,i+1}^*, \quad i \in Z_{0,N-1},$$

(6B.97)

and from the last block row in (6B.96a) it follows that

$$\lambda_{N_i,i}^* = -\lambda_{tc,i}^*, \quad i \in Z_{0,N-1}.$$

(6B.98)

Furthermore, for subproblems where LICQ holds it follows that rank $\tilde{B}_i = n_x$ which gives rank $\tilde{B}_i = n_x$, and hence from (6B.96h) and (6B.96f) it follows that

$$\lambda_{tc,i}^* = -\hat{\lambda}_{i+1}^*, \quad \hat{\lambda}_i^* = \lambda_{0,i}^*.$$  

(6B.99)

For subproblems where LICQ is violated, the dual solution is computed as in (6.29) with $\xi_{tc,i}^*$ defined as in (6.30). Using this choice of dual variables in the subproblem results in $\lambda_{tc,i}^* = \hat{\lambda}_{i+1}^*$ due to the definition of $Z_i$ and $\xi_{tc,i}^*$, and hence (6B.99) hold also for the subproblems where LICQ is violated.

Since $X_i^*$ for $i \in Z_{0,N}$ are the solutions to the subproblems, it follows that they satisfy the dynamics constraints in the subproblems

$$x_{t+1,i}^* = A_{t,i} x_{t,i}^* + B_{t,i} u_{t,i}^* + a_{t,i}, \quad t \in Z_{0,N_i-1}.$$  

(6B.100)

Specifically, due to (6B.97),

$$x_{0,i+1}^* = A_{N_i-1,i} x_{N_i-1,i}^* + B_{N_i-1,i} u_{N_i-1,i}^* + a_{N_i-1,i},$$  

(6B.101)

holds for all $i \in Z_{0,N-1}$. Note that variables from adjacent subproblems are used in (6B.101). Hence, using the definitions of $A_{t,i}$, $B_{t,i}$ and $a_{t,i}$ from (6.2) it follows that the primal solution (6.27) satisfies the dynamics constraints (6B.95d), and the initial constraint (6B.95e) is trivially satisfied.

Furthermore, since $\Lambda_i^*$ satisfies (6B.96a) for $i \in Z_{0,N}$, it follows that

$$Q_{x,t,i} x_{t,i}^* + Q_{xu,t,i} u_{t,i}^* - \lambda_{t,i}^* + A_{t,i}^T \lambda_{t+1,i}^* = -l_{x,t,i}, \quad t \in Z_{0,N_i-1},$$  

(6B.102a)

$$Q_{xu,t,i} x_{t,i}^* + Q_{u,t,i} u_{t,i}^* + B_{t,i}^T \lambda_{t+1,i}^* = -l_{u,t,i}, \quad t \in Z_{0,N_i-1}.$$  

(6B.102b)
Furthermore, from (6B.98) and (6B.99) it follows that
\[ Q_{x,N_i-1,i}x_{N_i-1,i}^* + Q_{xu,N_i-1,i}u_{t,i}^* - \lambda_{N_i-1,i}^* = -l_{x,N_i-1,i}, \]  
(6B.103a)
\[ Q_{xu,N_i-1,i}x_{N_i-1,i}^* + Q_{u,N_i-1,i}u_{N_i-1,i}^* + B_{N_i-1,i}\lambda_{0,i+1}^* = -l_{u,N_i-1,i}, \]  
(6B.103b)
hold for all \( i \in \mathbb{Z}_{0,\hat{N}-1} \). Note that (6B.98) and (6B.99) have been used to substitute \( \lambda_{N_i,i}^* = \lambda_{0,i+1}^* \). By using the definitions of \( Q_{x,t,i}, Q_{xu,t,i}, Q_{u,t,i}, l_{x,t,i} \) and \( l_{u,t,i} \) from (6.2), it follows that the primal and dual solution given by (6.27) and (6.28) satisfy (6B.95a)-(6B.95c). Hence, (6.27) and (6.28) satisfy the kkt conditions (6B.95) for the original uftoc problem, and from Theorem 2.8 it follows that \( x^* \), \( u^* \) and \( \lambda^* \) is a solution to (6.1), which concludes the proof.

6.B.3 Proof of Lemma 6.6

Proof: From the definition of \( A_i \) in (6.6b) and \( \mathcal{N}(A_i^T) = \{ \lambda_i^N \in \mathbb{R}^{m_i} | A_i^T \lambda_i^N = 0 \} \) it follows that
\[ -\lambda_{t,i}^N + A_{t,i}\lambda_{t+1,i}^N = 0, \quad t \in \mathbb{Z}_{0,N_i-1}, \]  
(6B.104a)
\[ B_{t,i}\lambda_{t+1,i}^N = 0, \quad t \in \mathbb{Z}_{0,N_i-1}, \]  
(6B.104b)
\[ \lambda_{N_i,i}^N = -\lambda_{tc,i}^N, \]  
(6B.104c)
must hold. Equations (6B.104a) and (6B.104c) can be combined into
\[ \lambda_i^N = \begin{bmatrix} \lambda_{0,i}^N \\ \vdots \\ \lambda_{N_i,i}^N \\ \lambda_{tc,i}^N \end{bmatrix} = \begin{bmatrix} \left(\prod_{t=0}^{N_i-1} A_t \right)^T \\ -I \end{bmatrix} \lambda_{tc,i}^N = \begin{bmatrix} -\tilde{A}_i^T \\ -\tilde{D}_i^T \end{bmatrix} \lambda_{tc,i}^N, \]  
(6B.105)
where \( \tilde{D}_i \) and \( \tilde{A}_i \) are defined as in (6.11a). By combining (6B.104a) and (6B.104b) it follows that
\[ \begin{bmatrix} B_{0,i} \left(\prod_{t=1}^{N_i-1} A_t \right)^T \\ \vdots \\ B_{N_i-2,i} A_{N_i-1,i} \\ B_{N_i-1,i} \end{bmatrix} \lambda_{tc,i}^N = -\tilde{B}_i^T \lambda_{tc,i}^N = 0, \]  
(6B.106)
where \( \tilde{B}_i \) is defined in (6.11b). Hence \( \lambda_{tc,i}^N \in \mathcal{N}(\tilde{B}_i^T) \) must hold. Finally, define
\[ Z_i \triangleq \begin{bmatrix} -\tilde{A}_i^T \\ -\tilde{D}_i^T \\ I \end{bmatrix}, \quad w_i \triangleq \lambda_{tc,i}^N, \]  
(6B.107)
which concludes the proof.
6.C Proofs of lemmas in Section 6.3

In this section, the proofs of some of the lemmas presented in Section 6.3 are given.

6.C.1 Proof of Lemma 6.9

Proof: By forming the KKT optimality conditions for (6.35), the KKT system is given by

\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 & 0 \\
-I & Q_0 & R & 0 & \Omega_0^T & 0 \\
0 & R^T & Q_u & S^T & \Gamma^T & \Gamma_N^T \\
0 & 0 & S & Q_x & -\Omega^T & \Omega_N^T \\
0 & 0 & 0 & \Omega_0 & \Omega & 0 \\
0 & 0 & 0 & 0 & 0 & -I \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \\
x_0 \\
u \\
x \\
\lambda_N \\
x_N \\
\end{bmatrix}
= 
\begin{bmatrix}
-x \\
f_0 \\
f_u \\
f_x \\
-\gamma_N \\
f_N \\
\end{bmatrix}.
\]

(6C.108)

Since \(\Omega\) is invertible, also the second block matrix on the diagonal is invertible with the inverse given by

\[
\begin{bmatrix}
Q_x & -\Omega^T \\
-\Omega & 0 \\
\end{bmatrix}^{-1} = \begin{bmatrix} 0 & \Omega^{-1} \\
\Omega^{-T}Q_x\Omega^{-1} & 0 \\
\end{bmatrix}.
\]

(6C.109)

Now, rows 4 and 5 in (6C.108) can be used to compute \(x\) and \(\lambda\) as

\[
\begin{bmatrix}
x \\
\lambda \\
\end{bmatrix} = \begin{bmatrix}
Q_x & -\Omega^T \\
-\Omega & 0 \\
\end{bmatrix}^{-1} \left( \begin{bmatrix} 0 & S \\
\Omega_0 & \Gamma \\
\end{bmatrix} \begin{bmatrix} x_0 \\
u \\
\end{bmatrix} + \begin{bmatrix} \Omega_N^T \\
0 \\
\end{bmatrix} \lambda_N + \begin{bmatrix} f_x \\
\gamma \\
\end{bmatrix} \right).
\]

(6C.110)

Then (6.39) follows directly from (6C.110). Furthermore, using (6C.110) to eliminate \(x\) and \(\lambda\) from the KKT system (6C.108) gives the smaller system of equations

\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 & 0 \\
-I & Q_0 & R & \Omega_0^T & 0 \\
0 & R^T & Q_u & \Gamma^T & \Gamma_N^T \\
0 & 0 & \Omega_N & \Gamma_N & 0 \\
0 & 0 & 0 & -I & Q_N \\
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \\
x_0 \\
u \\
\lambda_N \\
x_N \\
\end{bmatrix}
= 
\begin{bmatrix}
-x \\
f_0 \\
f_u \\
-\gamma_N \\
f_N \\
\end{bmatrix},
\]

(6C.111)

where the matrices are defined as in (6A.84). From the definition in (6A.84) the following factorization can be derived

\[
\begin{bmatrix}
\tilde{Q}_0 & \tilde{R} \\
\tilde{R}^T & \tilde{Q}_u \\
\end{bmatrix} = \begin{bmatrix} I & 0 & \Omega_0^T\Omega^{-T} \Omega_0 \Omega^{-T} \\
0 & I & \Gamma^T \Omega^{-T} \\
\end{bmatrix}
\begin{bmatrix} Q_0 & R & 0 \Omega_0 \Omega^{-T} \\
R^T & Q_u & S^T \Omega_x \Omega^{-T} \\
0 & S^T & Q_x \Omega^{-T} \\
\end{bmatrix}
\begin{bmatrix} I & 0 \Omega^{-1} \Omega_0 \Omega^{-1} \Gamma \\
0 & I \Omega^{-1} \Gamma \\
\end{bmatrix} \succeq 0,
\]

(6C.112)

which is positive semi-definite by construction since (6.36) holds. Furthermore,
since the KKT system (6C.108) has a unique solution due to assumptions 3.3 and 3.4 and that LICQ holds, the reduced Hessian of the corresponding UFTOC problem is positive definite (Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006). Now, define

$$Z \triangleq \begin{bmatrix} 0 \\ I \\ \Omega^{-1} \Gamma \\ \Gamma_N + \Omega_N \Omega^{-1} \Gamma \end{bmatrix}. \quad (6C.113)$$

Then, using the equality constraint matrix from the KKT system (6C.108), it is straightforward to show that

$$\mathcal{R}(Z) \subset N\left( \begin{bmatrix} -I & 0 & 0 & 0 \\ \Omega_0 & \Gamma & -\Omega & 0 \\ 0 & \Gamma_N & \Omega_N & -I \end{bmatrix} \right), \quad (6C.114)$$

and by using the definition of $\tilde{Q}_u$ and $\tilde{\Gamma}_N$ from (6A.84), it follows that

$$Z^T \begin{bmatrix} Q_0 & R & 0 \\ R^T & Q_u & S \\ 0 & S^T & Q_x \end{bmatrix} Z = \hat{Q}_u + \tilde{\Gamma}_N^T Q_N \tilde{\Gamma}_N \succ 0, \quad (6C.115)$$

since the reduced Hessian is positive definite. This holds for all $Q_N \succeq 0$, and hence it follows that $\hat{Q}_u \succ 0$ must hold.

The reduced system of equations (6C.111) can be recognized as the KKT optimality conditions for the QP problem (6.37). Hence, a solution to (6C.108) can be computed using (6C.110) from a solution to (6C.111). The solution to (6C.111) is directly obtained from a solution to (6C.108). Hence, the two problems (6.35) and (6.37) are equivalent, which concludes the proof.

### 6.C.2 Proof of Lemma 6.11

**Proof:** The KKT system for the UFTOC problem $i$ in the form (6.34) is given by the following equations

$$-x_{t_i} + \bar{x}_i = 0, \quad (6C.116a)$$

$$\begin{bmatrix} -I & Q_{x,t} & Q_{xu,t} & A_t^T & 0 \\ 0 & Q_{xu,t}^T & Q_{u,t} & B_t^T & 0 \\ 0 & A_t^T & B_t & 0 & -I \end{bmatrix} \begin{bmatrix} \lambda_t \\ x_t \\ u_t \\ \lambda_{t+1}^{x+1} \\ x_{t+1} \end{bmatrix} = \begin{bmatrix} -l_{x,t} \\ -l_{u,t} \\ -a_t \end{bmatrix}, \quad t \in \mathbb{Z}_{t_i,t_i+N_i-1}, \quad (6C.116b)$$

$$-\dot{\lambda}_{t+1} + \dot{P}_{t+1} \dot{x}_{t+1} = \dot{\Psi}_{t+1}. \quad (6C.116c)$$

Let $P_{0,t}, \Psi_{0,t}, k_{0,t}$ and $K_{0,t}$ be obtained by solving this KKT system using the Riccati recursion for the preliminary choice $\dot{P}_{t+1} = 0$ and $\dot{\Psi}_{t+1} = 0$. Then it follows that $P_{0,t_i+N_i} = 0$ and $\Psi_{0,t_i+N_i} = 0$. Note that when $Q_{u,t} \in \mathcal{S}_n^{\infty, t}$ due to
Assumption 6.10, the Riccati recursion is computed as in Section 3.4.2. Assume that the dual variables for some \( t \in \mathbb{Z}_{t_i, t_i + N_t} \) can be computed from

\[
\lambda_t = P_{0,t} x_t - \Psi_{0,t} + D_T^T \hat{\lambda}_{t+1},
\]  

(6C.117)

when \( \hat{P}_{t+1} \neq 0 \) and \( \hat{\Psi}_{t+1} \neq 0 \). Clearly this holds at the end point \( \lambda_{t_i+N_t} = \hat{\lambda}_{i+1} \) if \( D_{t_i+N_t} \triangleq I \). Now, use (6C.117) for \( t+1 \) together with \( x_{t+1} = A_t x_t + B_t u_t + a_t \) to eliminate \( \lambda_{t+1} \) from (6C.116), resulting in

\[
\begin{bmatrix}
-I & F_{0,t+1} & H_{0,t+1} & 0 & A_t^T D_{t+1}^T \\
0 & H_{0,t+1}^T & G_{0,t+1} & 0 & B_t^T D_{t+1}^T \\
0 & A_t & B_t & -I & 0
\end{bmatrix}
\begin{bmatrix}
\lambda_t \\
x_t \\
u_t \\
x_{t+1} \\
\hat{\lambda}_{t+1}
\end{bmatrix} =
\begin{bmatrix}
-l_{x,t} + A_t^T (\Psi_{0,t+1} - P_{0,t+1} a_t) \\
-l_{u,t} + B_t^T (\Psi_{0,t+1} - P_{0,t+1} a_t) \\
-a_t
\end{bmatrix},
\]

(6C.118)

where \( F_{0,t+1}, H_{0,t+1} \) and \( G_{0,t+1} \) are computed as in Algorithm 4. By inserting (6.42) into (6C.116) and using the definitions of \( P_{0,t}, \Psi_{0,t}, K_{0,t+1} \) and \( k_{0,t+1} \) gives the equivalent system of equations

\[
\begin{bmatrix}
-I & P_{0,t} & H_{0,t+1} & 0 & A_t^T D_{t+1}^T \\
0 & 0 & G_{0,t+1} & 0 & B_t^T D_{t+1}^T \\
0 & (A_t + B_t K_{0,t+1}) & B_t & -I & 0
\end{bmatrix}
\begin{bmatrix}
\lambda_t \\
x_t \\
u_t \\
x_{t+1} \\
\hat{\lambda}_{t+1}
\end{bmatrix} =
\begin{bmatrix}
\Psi_{0,t} \\
0 \\
-(a_t + B_t k_{0,t+1})
\end{bmatrix}.
\]

(6C.119)

Finally, by using \( H_{0,t+1} = -K_{0,t+1}^T G_{0,t+1} \) by the definition of \( K_{0,t+1} \), and then adding the second block equation multiplied with \( K_{0,t+1}^T \) from the left to the first block equation gives the system of equations

\[
\begin{bmatrix}
-I & P_{0,t} & 0 & 0 & (A_t + B_t K_{0,t+1})^T D_{t+1}^T \\
0 & 0 & G_{0,t+1} & 0 & B_t^T D_{t+1}^T \\
0 & (A_t + B_t K_{0,t+1}) & B_t & -I & 0
\end{bmatrix}
\begin{bmatrix}
\lambda_t \\
x_t \\
u_t \\
x_{t+1} \\
\hat{\lambda}_{t+1}
\end{bmatrix} =
\begin{bmatrix}
\Psi_{0,t} \\
0 \\
-(a_t + B_t k_{0,t+1})
\end{bmatrix}.
\]

(6C.120)

Hence, by defining

\[
D_t \triangleq D_{t+1} (A_t + B_t K_{0,t+1}),
\]  

(6C.121)

it follows that (6C.117) holds also for time \( t \). Since \( t \) is chosen arbitrary and \( D_{t_i+N_t} = I \) it follows from induction that (6.45b) holds for all \( t \in \mathbb{Z}_{t_i, t_i + N_t} \).

By using the state recursion \( x_{t+1} = (A_t + B_t K_{0,t+1}) x_t + B_t u_t + a_t + B_t k_{0,t+1}, \)
the states $x_t$ for $t \in \mathbb{Z}_{t_i+1,t_i+N_i-1}$ can be eliminated from the KKT system, which results in the system of equations

$$
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & P_{0,t_i} & 0 & \hat{A}_T^T & 0 \\
0 & 0 & \hat{Q}_{u,i} & S_i^T & 0 \\
0 & \hat{A}_i & S_i & 0 & -I \\
0 & 0 & 0 & -I & \hat{P}_{i+1}
\end{bmatrix}
\begin{bmatrix}
\lambda_i \\
\hat{x}_i \\
\hat{u}_i \\
\lambda_{i+1} \\
\hat{a}_{i+1}
\end{bmatrix}
= \begin{bmatrix}
-\bar{x}_i \\
0 \\
0 \\
-\bar{a}_{i+1}
\end{bmatrix},
$$

(6C.122)

where the matrices are defined as in Lemma 6.11. This system of equations is in the form of the KKT system of the UFTOC problem (6.43). Furthermore, $\hat{c}_{0,t_i}$ is computed from Algorithm 5 for the preliminary choice $\hat{P}_{i+1} = 0$, $\hat{\Psi}_{i+1} = 0$ and $\hat{c}_{i+1} = 0$, which concludes the proof.

### 6.C.3 Proof of Lemma 6.12

**Proof:** The solution to (6.46) is obtained by solving the corresponding KKT system

$$
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & Q_0 & R & \Omega^T & 0 \\
0 & R^T & \hat{Q}_u & \Gamma^T & 0 \\
0 & \Omega & \Gamma & 0 & -I \\
0 & 0 & 0 & -I & Q_N
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \\
\lambda_N \\
\lambda_{N+1} \\
\lambda_{N+2} \\
\lambda_{N+3}
\end{bmatrix}
= \begin{bmatrix}
-\bar{x}_i \\
-\bar{f}_u \\
-\gamma \\
-\bar{f}_u \\
-\bar{f}_u
\end{bmatrix},
$$

(6C.123)

which has a unique solution since assumptions 3.3 and 3.4, and LICQ hold. Now, since $\text{colrank } \Gamma = r < m$ holds, it is possible to find a unitary matrix

$$
[U \ V] \in \mathbb{R}^{m \times m},
$$

(6C.124)

where the columns of $U \in \mathbb{R}^{m \times r}$ form an orthonormal basis for $\mathcal{R}(\Gamma^T)$ and the columns of $V \in \mathbb{R}^{m \times (m-r)}$ form an orthonormal basis of $\mathcal{N}(\Gamma)$. Hence, since $\mathbb{R}^m = \mathcal{N}(\Gamma)^\perp \oplus \mathcal{N}(\Gamma) = \mathcal{R}(\Gamma^T) \oplus \mathcal{N}(\Gamma)$, the vector $u$ can be decomposed as

$$
u = U \hat{u} + V \hat{\bar{z}} = \begin{bmatrix} U & V \end{bmatrix} \begin{bmatrix} \hat{u} \\ \hat{\bar{z}} \end{bmatrix},
$$

(6C.125)

where $\hat{u} \in \mathbb{R}^r$ and $\hat{\bar{z}} \in \mathbb{R}^{m-r}$ (Lancaster and Tismenetsky, 1985). By using (6C.125) and multiplying the third row of the KKT system (6C.123) from the left with the invertible matrix $[U \ V]^T$ gives the equivalent system of equations

$$
\begin{bmatrix}
0 & -I & 0 & 0 & 0 & 0 \\
-I & Q_0 & RU & RV & \Omega^T & 0 \\
0 & U^T R^T & U^T \hat{Q}_u U & U^T \hat{Q}_u V & U^T \Gamma^T & 0 \\
0 & V^T R^T & V^T \hat{Q}_u U & V^T \hat{Q}_u V & 0 & 0 \\
0 & \Omega & \Gamma U & 0 & 0 & -I \\
0 & 0 & 0 & 0 & -I & Q_N
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \\
\lambda_N \\
\lambda_{N+1} \\
\lambda_{N+2} \\
\lambda_{N+3} \\
\lambda_{N+4}
\end{bmatrix}
= \begin{bmatrix}
-\bar{x} \\
-\bar{f}_0 \\
-\bar{f}_0 \\
-\bar{f}_0 \\
-\bar{f}_0 \\
-\bar{f}_0
\end{bmatrix},
$$

(6C.126)

Here the relation $\Gamma V = 0$, which follows from the definition of $V$, has been used. Since $Q_u \succ 0$ and $V$ has full rank, $V^T Q_u V \succ 0$ and $\hat{z}$ can be elimination.
ted from (6C.126) using the fourth row as
\[ \hat{z} = -\left( V^T Q_u V \right)^{-1} \left( V^T R^T x_0 + V^T Q_u U \hat{u} + V^T f_u \right), \] (6C.127)
giving the equivalent system of equations
\[
\begin{bmatrix}
0 & -I & 0 & 0 & 0 \\
-I & \hat{Q}_0 & \hat{R} & \hat{\Omega}^T & 0 \\
0 & \hat{R}^T & \hat{Q}_u & \hat{\Gamma}^T & 0 \\
0 & \hat{\Omega} & \hat{\Gamma} & 0 & -I \\
0 & 0 & 0 & -I & Q_N
\end{bmatrix}
\begin{bmatrix}
\lambda_0 \\
x_0 \\
\hat{u} \\
\lambda_N \\
x_N
\end{bmatrix}
= \begin{bmatrix}
-\bar{x} \\
-\hat{f}_0 \\
-\hat{f}_u \\
-\hat{\gamma} \\
-f_N
\end{bmatrix}. \] (6C.128)

The matrices in (6C.128) are defined as in (6A.85). Now, define
\[ H \triangleq \begin{bmatrix} Q_0 & R U \\ U^T R^T & U^T Q_u U & U^T Q_u V \end{bmatrix}, \quad \hat{H} \triangleq \begin{bmatrix} \hat{Q}_0 & \hat{R} \\ \hat{R}^T & \hat{Q}_u \end{bmatrix}, \quad G \triangleq \begin{bmatrix} U^T Q_u U & U^T Q_u V \end{bmatrix}. \] (6C.129)

Then it follows from (6A.85) and Theorem A.10 that
\[ \hat{H} = H/\left( V^T Q_u V \right) \succeq 0, \] (6C.130)
since \( H \succeq 0 \), and from Theorem A.11 that
\[ \hat{Q}_u = G/\left( V^T Q_u V \right) \succ 0, \] (6C.131)
since \( G \succ 0 \). Furthermore, (6.50) follows from (6C.125) and (6C.127), which concludes the proof. \( \square \)
In Chapter 4, explicit MPC and mp-QP were introduced and it was shown that the parametric solution to these are given by an affine function defined over $R$ critical regions. Once the PWA function is determined, the computational effort to compute the solution for a given parameter is reduced to evaluating the PWA function. How to solve the mp-QP problem has been considered in previous work by other authors and is outside the scope of this thesis, see for example Bemporad et al. (2002), Tøndel et al. (2003c), Spjøtvold et al. (2006), and Gupta et al. (2011). One of the main limitations with mp-QP and explicit MPC is the large amount of memory that is required to store the parametric solution and the critical regions. Some different techniques that can be used to reduce the memory required to store the parametric solution are discussed in the survey of related work in Section 1.1.2.

In this chapter, the limitation regarding the large memory requirements for storing the solutions is targeted. The chapter is based on the structure-exploiting approach for reducing the memory footprint in mp-QP which is introduced in Nielsen and Axehill (2016a), where the structure of the parametric solution between neighboring critical regions for an mp-QP problem is exploited. The proposed approach can significantly reduce the amount of memory that is required to store the solution and critical regions by exploiting that only low-rank modifications of the parametric solution is obtained when making minor changes to the optimal active sets. As discussed in Chapter 5, the possibility to exploit the structured low-rank modifications of the KKT matrices when modifying the working set between AS iterations has been crucial to improve the performance of methods for solving general QP problems. Here, this structure will be exploited for reducing the memory footprint when storing the parametric solution to an mp-QP problem. The proposed algorithm stores the solution in a tree structure and can be incor-
Exploiting Low-Rank Structure in Multiparametric Quadratic Programming

7 Exploiting Low-Rank Structure in Multiparametric Quadratic Programming

Exploited directly in existing mp-QP solvers, or be applied as a post-processing step to an already existing parametric solution in order to reduce the memory required to store it. Hence, the algorithm presented here can be interpreted as a data compression algorithm.

7.1 Low-Rank Modifications of the Parametric Solution

In this section it will be shown how the parametric solution in a neighboring region can efficiently be described by small, structured, modifications of the solution in the first region. Stepping over a facet between two neighboring regions corresponds to adding or removing constraints to the optimal active set (Tøndel et al., 2003c). Hence, an equivalent interpretation is that the parametric solution for one set of optimal active constraints can be used with only minor modifications to describe the solution for an optimal active set where constraints have been added to, or removed from, the first one. For notational convenience, the case when LICQ and SC hold and where only one constraint is added or removed is presented in this thesis. The case for \(k\) constraints can be shown analogously, and is used in the implementation of the proposed algorithm. When LICQ and SC do not hold, the critical regions can be overlapping and correspond to several optimal active sets (Tøndel et al., 2003c). The approach presented here can be extended to cover also these cases.

Here, an mp-QP problem in the form (4.3) with \(m\) variables, \(n\) parameters and \(p\) constraints is considered. Assume that a parametric solution to this mp-QP problem is already computed using some existing mp-QP solver. Then, the PWA function and the critical regions \(P_i\) for \(i \in \mathbb{Z}_{1,R}\) in (4.5) are given. If the corresponding optimal active sets are not returned by the parametric solver, they can be computed from the critical regions by using a fixed \(\theta \in P_i\) and solving the corresponding QP problem using an AS solver. Furthermore, define \(\sigma_i \in \mathbb{R}^p\) and \(\Gamma_i \in \mathbb{R}^{p \times n}\) for an optimal active set \(A_i\) as

\[
\sigma_i \triangleq \begin{cases} \sigma_{i,N_i} = 0, \\ \sigma_{i,A_i} = \sigma_i, \end{cases} \quad \Gamma_i \triangleq \begin{cases} \Gamma_{i,N_i} = 0, \\ \Gamma_{i,A_i} = \Gamma_i, \end{cases}
\]

(7.1)

where \(\sigma_i \in \mathbb{R}^{|A_i|}\) and \(\Gamma_i \in \mathbb{R}^{|A_i| \times n}\) are defined as in (4.8). A double index such as \(\Gamma_{i,A_i}\) is used to denote the rows in \(\Gamma_i\) that are indexed by \(A_i\). The boldface notation is introduced to denote a variable that is related to all the \(p\) constraints in the mp-QP problem (4.3), but with some components trivially equal to zero. The choice of this notation will later become clear. From (4.7), it follows that the parametric primal and dual solution to (4.3) for \(\theta \in P_i\) can be described as

\[
\lambda^*(\theta) = \sigma_i + \Gamma_i \theta,
\]

(7.2a)

\[
z^*(\theta) = k_i + K_i \theta \triangleq -H^{-1}G^T_{A_i} \left( \sigma_{i,A_i} + \Gamma_{i,A_i} \theta \right).
\]

(7.2b)
7.1 Low-Rank Modifications of the Parametric Solution

7.1.1 Adding one constraint to the optimal active set

Let the solution corresponding to an optimal active set $A_i \in A^*$ be given by (7.2). Consider the case when a constraint $l \in N_i$ is added such that the active set $A_j = A_i \cup l$ becomes optimal, and hence corresponds to the critical region $P_j$. Here $N_i$ is the inactive set introduced in Chapter 4.

**Theorem 7.1.** Let the parametric solution of (4.3) for the optimal active set $A_i \in A^*$ be given by (7.2). Then, the solution for an optimal active set $A_j = A_i \cup l$ with $l \in N_i$ is given by

$$\lambda^*(\theta) = \sigma_i + \Gamma_i \theta = \sigma_i + \Gamma_i \theta - d_j \left( c_j + v_j^T \theta \right),$$

(7.3a)

$$z^*(\theta) = k_i + K_i \theta = k_i + K_i \theta + f_j \left( c_j + v_j^T \theta \right),$$

(7.3b)

where $c_j \in \mathbb{R}$, $v_j \in \mathbb{R}^n$, $f_j \in \mathbb{R}^m$ and $d_j \in \mathbb{R}^p$ with $d_j, N_j = 0$, $d_{j,l} = -1$, and $d_{j,A_i}$ possibly non-zero.

**Proof:** For $A_j \in A^*$, the solution to the mp-QP problem (4.3) is given by (7.2), but with index $j$ instead of $i$. Without loss of generality, let $G_l$ be the last row in $G_{A_j}$. To compute $\sigma_i$ and $\Gamma_i$, the block matrix inverse in Theorem A.8 is used for

$$\begin{bmatrix} W & w \\ w^T & w_0 \end{bmatrix} = G_{A_j} H^{-1} G_{A_j}^T = \begin{bmatrix} G_{A_i} H^{-1} G_{A_i}^T & G_{A_i} H^{-1} G_l^T \\ G_l H^{-1} G_{A_i}^T & G_l H^{-1} G_l^T \end{bmatrix}. \quad (7.4)$$

The dual parametric solution (7.2a) for $A_j$ is then given by

$$\lambda^*_{A_j}(\theta) = - \begin{bmatrix} W^{-1} (b_{A_i} + S_{A_i} \theta) \\ 0 \end{bmatrix} - \begin{bmatrix} W^{-1} w \left( c_j + v_j^T p \right) \\ - \left( c_j + v_j^T p \right) \end{bmatrix}, \quad (7.5)$$

where $c_j$ and $v_j$ are defined as

$$c_j \triangleq C^{-1} \left( w^T W^{-1} b_{A_i} - b_l \right) \in \mathbb{R}, \quad (7.6a)$$

$$v_j \triangleq C^{-1} \left( S_{A_i}^T W^{-1} w - S_l^T \right) \in \mathbb{R}^n, \quad (7.6b)$$

and $C$ is defined as

$$C \triangleq \begin{bmatrix} W & w \\ w^T & w_0 \end{bmatrix} / W = w_0 - w^T W^{-1} w. \quad (7.7)$$

Furthermore, from (4.8) and the definition of $W$ in (7.4), it follows that the equation $-W^{-1} (b_{A_i} + S_{A_i} \theta) = \sigma_i + \Gamma_i \theta$ holds. Hence, by defining

$$d_{j,A_i} \triangleq W^{-1} w \in \mathbb{R}^{|A_i|}, \quad d_{j,l} \triangleq -1, \quad d_{j,N_j} \triangleq 0, \quad (7.8)$$

the dual solution can be written compactly as (7.3a).
From (7.2b) it can be seen that the primal solution for the optimal active set \( A_j \) is \( z^*(\theta) = -H^{-1}G_{A_j}^T \lambda^*_A(\theta) \), giving

\[
z^*(\theta) = -H^{-1}G_{A_j}^T \left( \sigma_{i,A_j} + \Gamma_{i,A_j} \theta - d_{j,A_j} \left( c_j + v_j^T \theta \right) \right) = -H^{-1}G_{A_i}^T \left( \sigma_{i,A_i} + \Gamma_{i,A_i} \theta \right) + H^{-1}G_{A_j}^T d_{j,A_j} \left( c_j + v_j^T \theta \right). \tag{7.9}
\]

In the second equality, it is exploited that \( \sigma_{i,l} = 0 \) and \( \Gamma_{i,l} = 0 \) by definition. Using \( -H^{-1}G_{A_i}^T \left( \sigma_{i,A_i} + \Gamma_{i,A_i} \theta \right) = k_i + K_i \theta \) from (7.2b) and defining

\[
f_j \triangleq H^{-1}G_{A_j}^T d_{j,A_j} \in \mathbb{R}^m, \tag{7.10}
\]

the primal solution (7.3b) can be obtained from (7.9), which concludes the proof.

From Theorem 7.1 it can be seen that the parametric solution in \( P_j \) can be computed as a rank-one modification of the parametric solution in \( P_i \).

**Corollary 7.2.** Let \( A_i \in \mathcal{A}^* \) be an optimal active set, and let the parametric solution to (4.3) for an optimal active set \( A_j = A_i \cup l \in \mathcal{A}^* \) be given by Theorem 7.1. Then the hyperplanes that define the corresponding critical region \( P_j \) in (4.9) can equivalently be described as

\[
G_N \left( k_i + K_i \theta \right) + f_{j,N} \left( c_j + v_j^T \theta \right) \preceq b_{N_j} + S_{N_j} \theta, \tag{7.11a}
\]

\[
\sigma_{i,A_j} + \Gamma_{i,A_j} \theta - d_{j,A_j} \left( c_j + v_j^T \theta \right) \succeq 0, \tag{7.11b}
\]

where \( f_j \triangleq Gf_j \in \mathbb{R}^P \) with \( f_j, A_i = 0 \).

**Proof:** The critical region \( P_j \) is given by (4.9) but with index \( j \) instead of \( i \). Hence, the dual feasibility conditions (7.11b) for \( P_j \) follow directly from the dual solution (7.3a).

By inserting the primal solution (7.3b) into the inequality constraints of (4.3) and defining \( f_j \triangleq Gf_j \) gives

\[
G \left( k_i + K_i \theta \right) + Gf_j \left( c_j + v_j^T \theta \right) \preceq b + S \theta \iff G_N \left( k_i + K_i \theta \right) + f_{j,N} \left( c_j + v_j^T \theta \right) \preceq b_{N_j} + S_{N_j} \theta. \tag{7.12}
\]

Here it is exploited that \( G_{A_j} \left( k_i + K_i \theta \right) + G_{A_j} f_j \left( c_j + v_j^T \theta \right) = G_{A_j} \left( k_j + K_j \theta \right) = b_{A_j} + S_{A_j} \theta \) by the definition of \( A_j \). From the definitions of \( f_j \) in (7.10), \( d_j \) in (7.8), and \( W \) and \( w \) in (7.4) it follows that

\[
f_{j,A_i} = G_{A_i} f_j = G_{A_i} H^{-1} G_{A_j}^T d_{j,A_j} = G_{A_i} H^{-1} \left( G_{A_i}^T W^{-1} w - G_{f_i}^T \right) = 0, \tag{7.13}
\]

which concluded the proof. \( \square \)
Remark 7.3. The hyperplanes given by \( G_{\mathcal{N}_j} (k_i + K_j \theta) \leq b_{\mathcal{N}_j} + S_{\mathcal{N}_j} \theta \) are a subset of the primal feasibility conditions in \( \mathcal{P}_i \), and \( \mathbf{\sigma}_{i,l} = 0 \) and \( \mathbf{\Gamma}_{i,l} = 0 \). Hence, the new information in the description of \( \mathcal{P}_j \) is contained in \( \mathbf{f}_{j,\mathcal{N}_j} \in \mathbb{R}^{\mid \mathcal{N}_j \mid} \) and \( \mathbf{d}_{j,\mathcal{A}_j} \in \mathbb{R}^{\mid \mathcal{A}_j \mid} \). Note that \( c_j \) and \( v_j \) are already computed when computing \( \lambda^*(\theta) \) and \( z^*(\theta) \) as in the proof of Theorem 7.1.

7.1.2 Removing one constraint from the optimal active set

When a constraint is removed from the optimal active set, the parametric solution and the description of the critical region change in a similar way as in Section 7.1.1.

Theorem 7.4. Let the solution for the optimal active set \( \mathcal{A}_j \in \mathcal{A}^* \) be given by (7.2) but with index \( j \) instead of \( i \). Then, the parametric solution for the optimal active set \( \mathcal{A}_i = \mathcal{A}_j \setminus l \in \mathcal{A}^* \) with \( l \in \mathcal{A}_j \) is given by

\[
\lambda^*(\theta) = \mathbf{\sigma}_i + \mathbf{\Gamma}_i \theta \triangleq \mathbf{\sigma}_j + \mathbf{\Gamma}_j \theta - \mathbf{d}_i \left( c_i + v_i^T \theta \right), \tag{7.14a}
\]

\[
z^*(\theta) = k_i + K_i \theta \triangleq k_j + K_j \theta + f_i \left( c_i + v_i^T \theta \right), \tag{7.14b}
\]

where \( c_i \triangleq c_j, v_i \triangleq v_j, \mathbf{d}_i \triangleq -\mathbf{d}_j \) and \( f_i \triangleq -f_j \). Here \( c_j, v_j, \mathbf{d}_j \) and \( f_j \) are defined as in (7.6), (7.8) and (7.10) respectively.

Proof: First, note that \( \mathcal{A}_i = \mathcal{A}_j \setminus l \iff \mathcal{A}_j = \mathcal{A}_i \cup l \). Hence, Theorem 7.1 applies for the optimal active set \( \mathcal{A}_j \), and by re-ordering the terms in (7.3a) it can be seen that

\[
\mathbf{\sigma}_i + \mathbf{\Gamma}_i \theta = \mathbf{\sigma}_j + \mathbf{\Gamma}_j \theta + \mathbf{d}_j \left( c_j + v_j^T \theta \right), \tag{7.15}
\]

which, by defining the variables \( c_i \triangleq c_j, v_i \triangleq v_j \) and \( \mathbf{d}_i \triangleq -\mathbf{d}_j \) gives (7.14a). Furthermore, by re-ordering the terms in (7.3b) it follows that

\[
k_i + K_i \theta = k_j + K_j \theta - f_j \left( c_j + v_j^T \theta \right), \tag{7.16}
\]

which, using \( c_i, v_i \) and defining \( f_i \triangleq -f_j \), gives (7.14b). \( \square \)

Similarly as when adding a constraint to the optimal active set, the parametric solution in \( \mathcal{P}_i \) is a rank-one modification of the parametric solution in \( \mathcal{P}_j \).

Corollary 7.5. Let \( \mathcal{A}_j \in \mathcal{A}^* \) be an optimal active set, and let the parametric solution to (4.3) for the optimal active set \( \mathcal{A}_i = \mathcal{A}_j \setminus l \in \mathcal{A}^* \) be given by Theorem 7.4. Then the hyperplanes that define the corresponding critical region \( \mathcal{P}_i \) in (4.9) can equivalently be described as

\[
G_{\mathcal{N}_i} \left( k_j + K_j \theta \right) + \mathbf{f}_{i,\mathcal{N}_i} \left( c_i + v_i^T \theta \right) \leq b_{\mathcal{N}_i} + S_{\mathcal{N}_i} \theta, \tag{7.17a}
\]

\[
\mathbf{\sigma}_{j,\mathcal{A}_i} + \mathbf{\Gamma}_{j,\mathcal{A}_i} \theta - \mathbf{d}_{i,\mathcal{A}_i} \left( c_i + v_i^T \theta \right) \geq 0, \tag{7.17b}
\]

where \( \mathbf{f}_i \triangleq Gf_i \in \mathbb{R}^p \) with \( \mathbf{f}_{i,\mathcal{A}_i} = 0 \).
Proof: The dual feasibility conditions (7.17b) follow directly from (7.14a). Furthermore, inserting the primal parametric solution (7.14b) into the inequality constraints of the mp-QP problem (4.3) gives
\[ G_{N_i} (k_j + K_j \theta) + G_{N_i} f_i \left( c_i + v_i^T \theta \right) \preceq b_{N_i} + S_{N_i} \theta. \] (7.18)

By using the definition \( f_i \triangleq G f_i \), (7.18) gives (7.17a). From Corollary 7.2 and the definition of \( f_i \) in Theorem 7.4 it follows that \( f_i, A_i = -f_j, A_i = 0 \), which concludes the proof.

Similar to Remark 7.3, the description of \( P_j \) is re-used in \( P_i \). Furthermore, \( G_l (k_j + K_j \theta) = b_l + S_l \theta \) since \( l \in A_j \). Hence, the new information in the description of \( P_i \) is contained in \( f_{i, N_i} \in \mathbb{R}^{|N_i|} \) and \( d_{i, A_i} \in \mathbb{R}^{|A_i|} \).

7.2 Memory Efficient Storage Tree

In this section it will be shown how the theory presented in Section 7.1 can be used repeatedly to store the parametric solution and critical regions in a memory efficient manner. The storage of the parametric solution is arranged into a tree structure, henceforth denoted the storage tree. The tree structure is related to the tree in Gupta et al. (2011) and the graph in Herceg et al. (2013b). Note, however, that the results in Section 7.1 do not require a tree structure, but that the structure is an inherent property of the parametric solution. Hence, the low-rank structure between neighboring critical regions can be exploited also for more general graph structures than the tree used in this chapter.

The set of all optimal active sets \( A^* \) can be arranged in a tree structure by choosing the root node \( r \in \mathbb{Z}_{1,R} \) to correspond to the optimal active set \( A_r \in A^* \). To simplify the notation, let \( \text{pa}(i) \), \( \text{ch}(i) \), and \( \text{anc}(i) \) denote the parent, the set of children and the ordered set of ancestors of node \( i \) in the tree, respectively. Furthermore, let \( \text{desc}(i) \) denote the descendants of node \( i \), let \( P(i) = i \cup (\text{anc}(i) \setminus r) \) be the ordered set of the node and the ancestor nodes except the root, and let \( D \) be the maximum depth in the tree.

**Definition 7.6.** A storage tree of a set of optimal active sets \( A^* \) is denoted \( T(A^*, r) \). Node \( r \) corresponds to the optimal active set \( A_r \in A^* \) and is the root node in \( T(A^*, r) \).

**Assumption 7.7.** For all nodes \( i \in T(A^*, r) \setminus r \) only one constraint is added to, or removed from, the optimal active set of the parent node \( \text{pa}(i) \).

The nodes in a storage tree from Definition 7.6 correspond to the optimal active sets \( A_i \in A^* \). When Assumption 7.7 holds, they are arranged such that either one constraint is added to or removed from the optimal active set of the parent of each node. Moving between a parent and one of its children hence corresponds to stepping over a facet between adjacent critical regions (Tøndel et al., 2003c). For
Figure 7.1: An example of a partitioning with critical regions $P_i$ for $i \in \mathbb{Z}_{1,6}$ in two dimensions. The three separating hyperplanes correspond to constraints 1, 2 and 3, respectively. When moving between adjacent critical regions, the constraint corresponding to the facet is either added or removed from the optimal active set. An example of a storage tree for this partition is seen in Figure 7.2. © 2016 IEEE

the case when $k$ constraints are added or removed, Assumption 7.7 does not hold. However, this case can be derived analogously but is omitted here for brevity.

An example with two parameters and a partitioning consisting of 6 critical regions $P_i$ for $i \in \mathbb{Z}_{1,6}$ is seen in Figure 7.1, and one choice of a corresponding storage tree $T(A^*, 2)$ is presented in Figure 7.2. Each critical region $P_i$ corresponds to an optimal active set $A_i \in A^*$, where $A^* = \{\{\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\}$ in this example. Hence, $P_2$ is the critical region for the optimal active set $A_2 = \{1\}$ etc. In Figure 7.1, the hyperplane between for example $P_2$ and $P_6$ corresponds to constraint 3. Hence, moving from region $P_2$ to $P_6$ by stepping over the shared facet corresponds to adding constraint 3 to the optimal active set in $P_2$, i.e., $A_6 = A_2 \cup \{3\}$ holds. In the tree in Figure 7.2, the optimal active set $A_2 = \{1\}$ is chosen as the root node. For this choice of the storage tree it can be seen that, for example, $pa(3) = 1$, $anc(3) = \{1, 2\}$, $P(3) = \{3, 1\}$, and $ch(2) = \{1, 5, 6\}$. The maximum depth is $D = 2$. The transition from $P_2$ to $P_6$ corresponds to moving from node 2 to node 6 in the tree by adding constraint 3, indicated by the plus sign at the edge.

Note that the storage tree is not unique. For example, the tree $T(A^*, 1)$ where the optimal active set $A_1 = \{\}$ is instead the root node could also be used for this example. The choice of the particular tree structure will affect the maximum depth of the tree, and hence also the on-line performance when looking up the solution. Furthermore, depending on the problem, the structure of the parametric solution and the number of constraints that are added or removed between neighboring critical regions, it might be beneficial to store the solution using several trees. In the case with several trees, each optimal active set is only included in one of the
trees. How to choose the number of the trees and the structure of the trees to obtain maximum performance is outside the scope of this thesis. Note also that the storage tree can be constructed either after all optimal active sets have been determined, or while building the solution to the mp-QP problem in the solver. Hence, the algorithm can be used as a post-processing algorithm to compress the parametric solution, or as an integrated algorithm in the mp-QP solver.

### 7.2.1 Computing the parametric solution and critical region

By storing the full description of the critical region $P_r$ together with the affine function given by $k_r$ and $K_r$ for the root node, and repeating the theory in Section 7.1, it can be shown that the parametric solution and description of the critical region in each node $i$ can be described by a number rank-one modifications of the solution in the root node. From the definition of $T(A^*, r)$ it follows that the modifications that are used to compute the solution and critical region in node $i$ are stored in the nodes $j \in P(i)$ and the root node, i.e., all nodes along the path from node $i$ to the root node $r$.

**Theorem 7.8.** Let $A^*$ be the set of optimal active sets for the mp-QP problem (4.3), and let $T(A^*, r)$ be a storage tree for which Assumption 7.7 holds. Then the parametric solution for $\theta \in P_i$ with $i \in T(A^*, r)$ is given by

$$z^*(\theta) = k_r + K_r \theta + \sum_{j \in P(i)} f_j \left( c_j + v_j^T \theta \right),$$

(7.19)

where $c_j$, $v_j$ and $f_j$ are defined as in Theorem 7.1 or Theorem 7.4 depending on the type of edge between $\text{pa}(j)$ and node $j$ in $T(A^*, r)$. Furthermore, the critical region $P_i$ is described by the set of parameters $\theta \in \Theta$ that satisfies the inequalities

![Figure 7.2](image-url)
\[ b_{N_i} + G_{N_i} \theta + \sum_{j \in P(i)} f_j N_i \left( c_j + v_j^T \theta \right) \leq 0, \] (7.20a)

\[ -\sigma_{r,A_i} - \Gamma_{r,A_i} \theta + \sum_{j \in P(i)} d_j A_i \left( c_j + v_j^T \theta \right) \leq 0, \] (7.20b)

where \( b \triangleq Gk_r - b \) and \( G \triangleq GK_r - S \).

**Proof:** Assume that (7.19) holds for some \( s \in \mathcal{T}(A^*, r) \) with \( \text{ch}(s) \neq \emptyset \), and consider an arbitrary node \( i \in \text{ch}(s) \). Then it follows from Theorem 7.1 (or Theorem 7.4) that the parametric solution is

\[ z^*(\theta) = k_r + K_r \theta + \sum_{j \in P(s)} f_j \left( c_j + v_j^T \theta \right) + f_i \left( c_i + v_i^T \theta \right) = k_r + K_r \theta + \sum_{j \in P(i)} f_j \left( c_j + v_j^T \theta \right). \] (7.21)

Here it has been used that \( P(i) = i \cup P(s) \) since \( i \in \text{ch}(s) \). Since (7.19) holds for the root \( r \), and \( s \) and \( i \in \text{ch}(s) \) were chosen arbitrarily, (7.19) follows from induction.

The inequalities (7.20) can be shown analogously from Corollary 7.2 (or Corollary 7.5) by utilizing that, for the root node \( r \), the dual feasibility conditions (4.4c) are

\[-\sigma_r - \Gamma_r \theta \preceq 0, \] (7.22a)

and the primal feasibility conditions (4.4b) are described by

\[ G (k_r + K_r \theta) \preceq b + S \theta \iff b + G \theta \preceq 0, \] (7.22b)

follows by the definition of \( f_j \), which concluded the proof. \( \square \)

**Corollary 7.9.** The hyperplanes that describe the minimal representation of the critical region \( \mathcal{P}_i \) in (4.10) can equivalently be described as

\[ b_{H^p_i} + G_{H^p_i} \theta + \sum_{j \in P(i)} f_j H^p_i \left( c_j + v_j^T \theta \right) \leq 0, \] (7.24a)

\[ -\sigma_{r,H^d_i} - \Gamma_{r,H^d_i} \theta + \sum_{j \in P(i)} d_j H^d_i \left( c_j + v_j^T \theta \right) \leq 0, \] (7.24b)

where \( H^p_i \) and \( H^d_i \) are defined in Section 4.3.

**Proof:** The corollary follows directly from Theorem 7.8 since each hyperplane that is indexed by \( s \) in (7.20), corresponding to either the primal or dual feasibility constraints, is described only by the components and rows indexed by \( s \) in (7.20). \( \square \)
Note that the full parametric solution and description of the critical region is only stored for the root node \( r \). For the nodes \( i \in \mathcal{T}(\mathbb{A}^*, r) \backslash r \), only the rank-one modifications \( c_i, v_i, d_i, l_i, \) and \( f_i \) need to be stored.

One algorithm to compute the solution as in Theorem 7.8 is presented in Algorithm 14, where the parameter \( \theta \) and the node \( i \) are inputs, and the optimal solution \( z^*(\theta) \) is returned.

**Algorithm 14** On-line evaluation of the parametric solution

1: input \( \theta \) and \( i \)
2: Initialize \( z = k_r + K_r \theta \)
3: for \( j \in \mathcal{P}(i) \) do
4: \( z := z + f_j \left( c_j + v_j^T \theta \right) \)
5: end for
6: \( z^* := z \)

In Algorithm 15 the evaluation of a hyperplane using Corollary 7.9 is presented. The parameter \( \theta \), the node \( i \), and the hyperplane index \( s \) are given as inputs to the algorithm, and the value of the hyperplane, denoted \( q \), is returned.

**Algorithm 15** On-line evaluation of a hyperplane

1: input \( \theta \), \( i \) and \( s \)
2: if \( s \in \mathcal{H}^p_i \) then
3: Initialize \( q := b_s + G_s \theta \)
4: for \( j \in \mathcal{P}(i) \) do
5: \( q := q + f_{j,s} \left( c_j + v_j^T \theta \right) \)
6: end for
7: else if \( s \in \mathcal{H}^d_i \) then
8: Initialize \( q := -\sigma_{r,s} - \Gamma_{r,s} \theta \)
9: for \( j \in \mathcal{P}(i) \) do
10: \( q := q + d_{j,s} \left( c_j + v_j^T \theta \right) \)
11: end for
12: end if

Note that for a given parameter \( \theta \) the value of \( c_j + v_j^T \theta \) only has to be computed once for each \( \mathcal{P}_j \) when evaluating hyperplanes, and \( b_s + G_s \theta \) and \( -\sigma_{r,s} - \Gamma_{r,s} \theta \) only need to be computed once for each parameter and can be re-used by all nodes in the storage tree.

In algorithms 14 and 15 the order of summation can be chosen to facilitate the on-line performance by taking for example memory access into consideration.

### 7.2.2 Storing the parametric solution and the critical regions

From Theorem 7.8 it follows that computing the parametric solution in node \( i \) only requires the storage of \( c_j \in \mathbb{R}, v_j \in \mathbb{R}^n \) and \( f_j \in \mathbb{R}^m \) for each node \( j \in \mathcal{P}(i) \).
However, in the root node \( r \) the full parametric solution defined by \( k_r \in \mathbb{R}^m \) and \( K_r \in \mathbb{R}^{m \times n} \) needs to be stored. In Corollary 7.9 it is shown that the description of the critical region in node \( i \) only requires the describing hyperplanes. Hence, the full vectors and matrices in (7.24) do not need to be stored, but only the components and rows of \( b, \sigma_r, G, \Gamma_r, f_j \) and \( d_j \) for \( j \in P(i) \) that correspond to the describing hyperplanes.

Furthermore, from Corollary 7.9 it follows that the rank-one modification in node \( i \) is also used by all descendants \( \text{desc}(i) \subset T(A^*, r) \). Hence, the components and rows with indices corresponding to the defining hyperplanes for \( P_j \) with \( j \in \text{desc}(i) \) also need to be stored in node \( i \). Let \( S^p_i \) and \( S^d_i \) be the indices of the hyperplanes corresponding to primal and dual feasibility conditions in the description of \( P_i \) that need to be stored in node \( i \). These sets of indices are given by

\[
S^p_i \triangleq \bigcup_{j \in \text{desc}(i) \cup i} \mathcal{H}^p_j, \quad S^d_i \triangleq \bigcup_{j \in \text{desc}(i) \cup i} \mathcal{H}^d_j. \tag{7.25}
\]

The components in \( f_i \) and \( d_i \) that are equal to zero by definition should not be taken into consideration when computing \( S^p_i \) and \( S^d_i \) in (7.25), since they do not need to be stored. In each node \( i \in T(A^*, r) \setminus r \), the rank-one modifications \( c_i \in \mathbb{R} \), \( v_i \in \mathbb{R}^n \), \( f_i \in \mathbb{R}^m \), \( f_i, S^p_i \in \mathbb{R}^{|S^p_i|} \) and \( d_i, S^d_i \in \mathbb{R}^{|S^d_i|} \) need to be stored. For node \( i \) this requires

\[
1 + n + m + |S^p_i| + |S^d_i|, \tag{7.26}
\]

real numbers to be stored. For the root node \( r \), the vectors \( k_r \in \mathbb{R}^m \), \( b_{S^p_r} \in \mathbb{R}^{|S^p_r|} \), and \( \sigma_r, S^d_r \in \mathbb{R}^{|S^d_r|} \), and the matrices \( K_r \in \mathbb{R}^{m \times n} \), \( G_{S^p_r} \in \mathbb{R}^{|S^p_r| \times n} \) and \( \Gamma_r, S^d_r \in \mathbb{R}^{|S^d_r| \times n} \) need to be stored. This requires

\[
m(n + 1) + \left(|S^p_r| + |S^d_r|\right) (n + 1), \tag{7.27}
\]

real numbers to be stored. Hence, the total number \( M_{\text{LR}} \) of stored real numbers for \( T(A^*, r) \) is

\[
M_{\text{LR}} = m(n + R) + M_{\text{LF}}^c, \tag{7.28}
\]

where \( M_{\text{LF}}^c \) is the number of real numbers required to store the critical regions, defined as

\[
M_{\text{LF}}^c \triangleq \left(|S^p_r| + |S^d_r|\right) (n + 1) + (R - 1) (1 + n) + \sum_{i \in T(A^*, r) \setminus r} \left(|S^p_i| + |S^d_i|\right). \tag{7.29}
\]

Note that the storage of \( c_i \) and \( v_i \) is included in \( M_{\text{LF}}^c \).

Storing the full parametric solution and the minimal description of the critical regions that were presented in Chapter 4 require

\[
M_F \triangleq Rm(n + 1) + M_{\text{LF}}^c, \tag{7.30}
\]

real numbers to be stored. Here \( M_F^c \) is the number of real numbers that is required
in order to store the critical regions, given by

$$M_F^{cr} = \sum_{i=1}^{R} \left( H_i^p + H_i^d \right) (n + 1).$$

(7.31)

In Section 4.4 explicit MPC was presented, and it was shown that the CFTOC problem (4.11) that arises in linear MPC for LTI systems can be re-written as an mp-QP problem. Hence, storing the explicit solution of (4.11) can be done using the theory presented in this chapter. However, only the first control input $u_0^*$ is used as input to the plant in the MPC control loop, and hence the full parametric solution does not need to be stored in the case of explicit MPC. Instead only the first $n_u$ rows of $k_i$ and $K_i$ are stored, respectively. For the traditional, non-compressed, solution this results in that

$$M_F^{mpc} = Rn_u (n + 1) + M_F^{cr},$$

(7.32)

real numbers are stored. This is a slightly modified version of $M_F$ in (7.30). Similarly for the compressed solution, only the rank-one modifications that are affecting the first $n_u$ rows need to be stored in $T(A^*, r)$. Hence, for the explicit MPC solution, the number of stored real numbers is

$$M_L^{mpc} = n_u (n + R) + M_L^{cr},$$

(7.33)

which is a slightly modified version of $M_L$ in (7.28).

### 7.3 Experimental Evaluation

In this section, the memory requirement for storing the solution and the critical regions when using the proposed algorithm is compared to storing the full solution and critical regions. The mp-QP problems have been solved using MPT (version: 3.1.2 (R2011a) 28.10.2015) in MATLAB (version: 8.4.0.150421 (R2014b)). Although the optimal active sets are computed in the mp-QP solver, it was not possible to access them in the solution returned by the solver. They have instead been retrieved by solving a QP problem for each critical region with the corresponding Chebychev center as parameter.

#### 7.3.1 Defining the problems

The comparison has been made for three different examples where explicit MPC controllers are applied to stable LTI systems. In the first example, referred to as Problem 1, the continuous-time system $1/(s + 1)^{n_x}$ which is studied in Kvasnica et al. (2015) is used. This system has $n_x$ states and $n_u = 1$ control inputs. The transfer function has been discretized using a unit sampling time, the weight matrices in (4.11) are $\tilde{Q}_x = I$ and $\tilde{Q}_u = 1$, and the terminal cost $\tilde{P}_N$ is chosen as the discrete time LQ cost. The states and control inputs are subject to the constraints

$$-10 \cdot 1_{n_x} \leq x_t \leq 10 \cdot 1_{n_x}, \quad t \in \mathbb{Z}_{0,N}, \quad -1 \leq u_t \leq 1, \quad t \in \mathbb{Z}_{0,N-1}.$$

(7.34)
After re-writing this explicit MPC problem into an equivalent mp-QP problem as in Section 4.4, the problem has \( n = n_x \) parameters and \( m = N n_u = N \) variables.

Both the second and third problems use a system which is similar to the one used in for example Wang and Boyd (2010) and Axehill (2015). It consists of \( n_M \) unit masses which are coupled with springs and dampers. The spring constants are chosen as 1, the damping constants as 0, the weight matrices in (4.11) to \( \tilde{Q}_x = 100 \cdot I \) and \( \tilde{Q}_u = I \) and the terminal cost \( \tilde{P}_N \) is chosen as the discrete time LQ cost. The continuous-time system is discretized using the sampling time 0.5 seconds. Two different cases have been studied, referred to as Problem 2 and Problem 3. In Problem 2, the control input is a force acting between terra firma and the first mass, and in Problem 3 there is also an extra control input acting as a force applied between the first two masses. In both problems, the states and control inputs are subject to the inequality constraints

\[
-4 \cdot 1_{n_x} \preceq x_t \preceq 4 \cdot 1_{n_x}, \quad t \in \mathbb{Z}_{0,N}, \quad -0.5 \cdot 1_{n_u} \preceq u_t \preceq 0.5 \cdot 1_{n_u}, \quad t \in \mathbb{Z}_{0,N-1}.
\]

(7.35)

Each mass introduces two states, and \( n_u = 1 \) in Problem 2 and \( n_u = 2 \) in Problem 3 by construction. Hence, by re-writing the MPC problem into the equivalent mp-QP problem as in Section 4.4, the corresponding mp-QP problem has \( n = 2 n_M \) parameters and \( m = N n_u \) variables.

### 7.3.2 Experimental results

The relative memory reduction has been computed for the three problems for different parameter dimensions and prediction horizons, and the results are summarized in tables 7.1-7.3. The storage tree \( T(A^*, r) \) is chosen such that the root node corresponds to the optimal active set \( A_r = \{\} \), i.e., the unconstrained minimum, and as few constraints as possible (preferably one constraint) are added to each child of the nodes. When the mp-QP problem is defined, all redundant constraints are removed, giving \( p \) number of non-redundant constraints. Furthermore, \( R \) is the number of regions, \( D \) is the maximum depth of \( T(A^*, r) \), and

\[
\Delta_{cr} \triangleq \frac{M_{cr}}{M_F}, \quad \Delta \triangleq \frac{M_{LR}}{M_F}, \quad \Delta_{mpc} \triangleq \frac{M_{mpc}}{M_{mpc}};
\]

(7.36)

are the relative reductions in the number of stored real numbers for only the critical regions, for the full solution and the critical regions, and for storing the first \( n_u \) control inputs and the critical regions, respectively. Hence, for the case of explicit MPC, \( \Delta_{mpc} \) determines the total relative memory reduction, whereas for a full mp-QP problem it is given by \( \Delta \). No symmetry or other properties which are inherited from the explicit MPC problems are exploited in the comparison. In some cases, for large problems with many parameters and variables, the solution computed by the proposed algorithm is inaccurate in a few regions. This is probably a consequence of difficulties with finding the correct optimal active set given the critical region, and has not been investigated further in this thesis.

In Table 7.1, the result for Problem 1 is presented, and it can be seen that the relative memory reduction becomes increasingly beneficial with increasing parameter
Table 7.1: Experimental results for the memory reduction using the proposed algorithm for Problem 1. © 2016 IEEE

<table>
<thead>
<tr>
<th>n/N</th>
<th>p</th>
<th>R</th>
<th>D</th>
<th>Δ_{cr}</th>
<th>Δ</th>
<th>Δ_{mpc}</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/2</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>0.909</td>
<td>0.729</td>
<td>0.827</td>
</tr>
<tr>
<td>4/2</td>
<td>20</td>
<td>11</td>
<td>2</td>
<td>0.446</td>
<td>0.403</td>
<td>0.431</td>
</tr>
<tr>
<td>6/2</td>
<td>28</td>
<td>45</td>
<td>2</td>
<td>0.258</td>
<td>0.239</td>
<td>0.252</td>
</tr>
<tr>
<td>8/2</td>
<td>44</td>
<td>153</td>
<td>2</td>
<td>0.176</td>
<td>0.167</td>
<td>0.173</td>
</tr>
<tr>
<td>10/2</td>
<td>52</td>
<td>192</td>
<td>2</td>
<td>0.130</td>
<td>0.126</td>
<td>0.129</td>
</tr>
<tr>
<td>12/2</td>
<td>66</td>
<td>255</td>
<td>2</td>
<td>0.107</td>
<td>0.105</td>
<td>0.107</td>
</tr>
<tr>
<td>14/2</td>
<td>80</td>
<td>336</td>
<td>2</td>
<td>0.090</td>
<td>0.088</td>
<td>0.090</td>
</tr>
<tr>
<td>2/3</td>
<td>12</td>
<td>5</td>
<td>2</td>
<td>0.909</td>
<td>0.694</td>
<td>0.827</td>
</tr>
<tr>
<td>4/3</td>
<td>24</td>
<td>13</td>
<td>3</td>
<td>0.476</td>
<td>0.411</td>
<td>0.456</td>
</tr>
<tr>
<td>6/3</td>
<td>36</td>
<td>89</td>
<td>3</td>
<td>0.258</td>
<td>0.234</td>
<td>0.252</td>
</tr>
<tr>
<td>8/3</td>
<td>56</td>
<td>575</td>
<td>3</td>
<td>0.187</td>
<td>0.175</td>
<td>0.184</td>
</tr>
<tr>
<td>10/3</td>
<td>66</td>
<td>1186</td>
<td>3</td>
<td>0.139</td>
<td>0.133</td>
<td>0.138</td>
</tr>
<tr>
<td>12/3</td>
<td>86</td>
<td>1679</td>
<td>3</td>
<td>0.124</td>
<td>0.119</td>
<td>0.123</td>
</tr>
<tr>
<td>14/3</td>
<td>102</td>
<td>2664</td>
<td>3</td>
<td>0.115</td>
<td>0.111</td>
<td>0.114</td>
</tr>
<tr>
<td>2/4</td>
<td>14</td>
<td>5</td>
<td>2</td>
<td>0.909</td>
<td>0.667</td>
<td>0.827</td>
</tr>
<tr>
<td>4/4</td>
<td>26</td>
<td>13</td>
<td>3</td>
<td>0.476</td>
<td>0.400</td>
<td>0.456</td>
</tr>
<tr>
<td>6/4</td>
<td>42</td>
<td>129</td>
<td>4</td>
<td>0.252</td>
<td>0.226</td>
<td>0.246</td>
</tr>
<tr>
<td>8/4</td>
<td>66</td>
<td>1222</td>
<td>4</td>
<td>0.207</td>
<td>0.189</td>
<td>0.203</td>
</tr>
<tr>
<td>10/4</td>
<td>80</td>
<td>4300</td>
<td>4</td>
<td>0.161</td>
<td>0.151</td>
<td>0.159</td>
</tr>
<tr>
<td>12/4</td>
<td>104</td>
<td>5408</td>
<td>4</td>
<td>0.181</td>
<td>0.172</td>
<td>0.179</td>
</tr>
</tbody>
</table>

dimension. For N = 2 and N = 3, the memory is reduced by approximately an order of magnitude for the parameter dimensions n = 12 and n = 14. The result for Problem 2 is presented in Table 7.2, and for this problem the relative memory reduction is an order of magnitude for problems with n_M ≥ 6. In Table 7.3 the numerical results for Problem 3 are presented, and it can be seen that also for this problem the relative memory reduction is increased for larger parameter dimensions. Note that for all evaluated problem dimensions the memory required for the storage tree is lower than for storing the full solution.
### Table 7.2: Experimental results for the memory reduction using the proposed algorithm for Problem 2. © 2016 IEEE

<table>
<thead>
<tr>
<th>$n_M/N$</th>
<th>$p$</th>
<th>$R$</th>
<th>$D$</th>
<th>$\Delta^{cr}$</th>
<th>$\Delta$</th>
<th>$\Delta^{mpc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/2</td>
<td>28</td>
<td>45</td>
<td>2</td>
<td>0.392</td>
<td>0.351</td>
<td>0.378</td>
</tr>
<tr>
<td>3/2</td>
<td>40</td>
<td>161</td>
<td>2</td>
<td>0.220</td>
<td>0.206</td>
<td>0.217</td>
</tr>
<tr>
<td>4/2</td>
<td>52</td>
<td>225</td>
<td>2</td>
<td>0.159</td>
<td>0.153</td>
<td>0.158</td>
</tr>
<tr>
<td>5/2</td>
<td>64</td>
<td>229</td>
<td>2</td>
<td>0.131</td>
<td>0.127</td>
<td>0.130</td>
</tr>
<tr>
<td>6/2</td>
<td>76</td>
<td>238</td>
<td>2</td>
<td>0.102</td>
<td>0.100</td>
<td>0.102</td>
</tr>
<tr>
<td>7/2</td>
<td>88</td>
<td>239</td>
<td>2</td>
<td>0.087</td>
<td>0.086</td>
<td>0.087</td>
</tr>
<tr>
<td>8/2</td>
<td>100</td>
<td>238</td>
<td>2</td>
<td>0.082</td>
<td>0.081</td>
<td>0.082</td>
</tr>
<tr>
<td>2/3</td>
<td>38</td>
<td>127</td>
<td>3</td>
<td>0.393</td>
<td>0.341</td>
<td>0.379</td>
</tr>
<tr>
<td>3/3</td>
<td>54</td>
<td>920</td>
<td>3</td>
<td>0.244</td>
<td>0.222</td>
<td>0.239</td>
</tr>
<tr>
<td>4/3</td>
<td>70</td>
<td>1953</td>
<td>3</td>
<td>0.169</td>
<td>0.159</td>
<td>0.167</td>
</tr>
<tr>
<td>5/3</td>
<td>86</td>
<td>2577</td>
<td>3</td>
<td>0.132</td>
<td>0.127</td>
<td>0.131</td>
</tr>
<tr>
<td>6/3</td>
<td>102</td>
<td>2861</td>
<td>3</td>
<td>0.102</td>
<td>0.100</td>
<td>0.102</td>
</tr>
<tr>
<td>7/3</td>
<td>118</td>
<td>3096</td>
<td>3</td>
<td>0.086</td>
<td>0.085</td>
<td>0.086</td>
</tr>
<tr>
<td>8/3</td>
<td>134</td>
<td>3084</td>
<td>3</td>
<td>0.078</td>
<td>0.077</td>
<td>0.078</td>
</tr>
<tr>
<td>2/4</td>
<td>48</td>
<td>282</td>
<td>4</td>
<td>0.406</td>
<td>0.336</td>
<td>0.388</td>
</tr>
<tr>
<td>3/4</td>
<td>68</td>
<td>2593</td>
<td>4</td>
<td>0.275</td>
<td>0.242</td>
<td>0.268</td>
</tr>
<tr>
<td>4/4</td>
<td>88</td>
<td>9479</td>
<td>4</td>
<td>0.203</td>
<td>0.187</td>
<td>0.200</td>
</tr>
<tr>
<td>5/4</td>
<td>108</td>
<td>18707</td>
<td>4</td>
<td>0.148</td>
<td>0.140</td>
<td>0.146</td>
</tr>
<tr>
<td>6/4</td>
<td>128</td>
<td>24629</td>
<td>4</td>
<td>0.111</td>
<td>0.108</td>
<td>0.111</td>
</tr>
</tbody>
</table>

### Table 7.3: Experimental results for the memory reduction using the proposed algorithm for Problem 3. © 2016 IEEE

<table>
<thead>
<tr>
<th>$n_M/N$</th>
<th>$p$</th>
<th>$R$</th>
<th>$D$</th>
<th>$\Delta^{cr}$</th>
<th>$\Delta$</th>
<th>$\Delta^{mpc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/2</td>
<td>32</td>
<td>267</td>
<td>4</td>
<td>0.360</td>
<td>0.299</td>
<td>0.333</td>
</tr>
<tr>
<td>3/2</td>
<td>44</td>
<td>1589</td>
<td>4</td>
<td>0.250</td>
<td>0.221</td>
<td>0.238</td>
</tr>
<tr>
<td>4/2</td>
<td>56</td>
<td>3765</td>
<td>4</td>
<td>0.173</td>
<td>0.160</td>
<td>0.168</td>
</tr>
<tr>
<td>5/2</td>
<td>68</td>
<td>4922</td>
<td>4</td>
<td>0.135</td>
<td>0.128</td>
<td>0.133</td>
</tr>
<tr>
<td>2/3</td>
<td>44</td>
<td>1114</td>
<td>6</td>
<td>0.381</td>
<td>0.296</td>
<td>0.348</td>
</tr>
<tr>
<td>3/3</td>
<td>60</td>
<td>12712</td>
<td>6</td>
<td>0.295</td>
<td>0.244</td>
<td>0.277</td>
</tr>
<tr>
<td>4/3</td>
<td>76</td>
<td>60206</td>
<td>6</td>
<td>0.236</td>
<td>0.205</td>
<td>0.226</td>
</tr>
<tr>
<td>2/4</td>
<td>56</td>
<td>3119</td>
<td>8</td>
<td>0.400</td>
<td>0.294</td>
<td>0.363</td>
</tr>
<tr>
<td>3/4</td>
<td>76</td>
<td>47797</td>
<td>8</td>
<td>0.347</td>
<td>0.267</td>
<td>0.322</td>
</tr>
</tbody>
</table>
8

Conclusions and Future Work

There are two main focuses in this thesis: structure-exploiting algorithms that can be used as important subroutines in popular second-order methods for solving certain types of optimal control problems, and an algorithm that can be used to reduce the memory footprint of mp-QP and explicit MPC solutions. Here, the conclusions of the thesis are presented together with some suggestions for future work that can extend the contributions presented in this thesis.

8.1 Conclusions

In this thesis, it is shown that it is possible to perform structured low-rank modifications of the Riccati factorization when the Riccati recursion is used to solve the UFTOC problems that correspond to the second-order search directions in an AS type solver applied to solve a CFTOC problem. By exploiting the structured modifications of the working set between AS iterations, the computational complexity for computing the Riccati factorization of the corresponding KKT matrix is reduced from cubic to only quadratic in the state and control input dimensions. Furthermore, the full Riccati factorization does not always need to be modified, which increases performance even further. Numerical results from a MATLAB implementation indicate large performance gains for relevant problem sizes.

Furthermore, parallel algorithms for solving UFTOC problems that arise as important subproblems in many methods for solving CFTOC problems are presented. This results in a logarithmic computational complexity growth in the prediction horizon length. The parallel algorithms decompose the UFTOC problem in time, and it is shown how to reduce a UFTOC problem into a master UFTOC problem of shorter prediction horizon. When the proposed algorithms are applied recursively, the logarithmic computational complexity growth is obtained. It is also shown how to compute the Riccati factorization and the Riccati recursion in parallel by...
using the time decomposition. Numerical results from MATLAB and ANSI-C implementations indicate significant performance gains in terms of computation time over serial state-of-the-art algorithms for problems with long horizons. The ANSI-C implementation has been executed truly in parallel on a computer cluster, showing the actual performance gains when also communication times are considered.

Finally, it is shown how structure in the parametric solution of an mp-QP problem can be exploited to reduce the amount of memory that is required to store the parametric solution. It is shown that for all but one critical region, only low-rank modification terms need to be stored instead of the full solution and full description of the critical regions as is standard today. Numerical results from example problems show gains of an order of magnitude for relevant problem sizes.

### 8.2 Future Work

Some suggestions for future work in the direction of the contributions presented in this thesis are:

- Investigate when low-rank modifications of the Riccati factorization should be performed, and when it is more beneficial to re-compute the factorization.
- Investigate how to choose the sizes of the subproblems in the parallel algorithms that are presented in this thesis. This can be done either theoretically, or by using some automatic benchmarking procedure.
- Investigate how the ideas from the parallel Riccati algorithm can be extended to develop a parallel RTS smoother using a similar time decomposition.
- Extend the parallel Riccati algorithm to perform parallel low-rank modifications of the Riccati factorization.
- Fully exploit problem structure when solving the subproblems in the parallel algorithm that is based on parametric programming.
- Investigate how to perform low-rank modifications of the Riccati factorization in parallel when several modifications are made in the same AS iteration.
- Investigate how to choose the number of trees and the tree structure for the storage tree for mp-QP problems. This is probably dependent on for example the structure of the problem and which point location algorithm that is used.
- Investigate which point location algorithm that is most efficient to combine with the storage tree, and if more general graph structures can be beneficial for the performance of the point location algorithm.
- Develop high-performance implementations of the proposed algorithms, and investigate if FPGAs or GPUs can be used to further increase the performance of the implementations of the parallel algorithms.
- Develop the theory for the primal AS solver that computes the search direction by solving a dual subproblem.
In this appendix, some notation and results from linear algebra that are used throughout the thesis are listed. For further details, the reader is referred to standard literature such as Lancaster and Tismenetsky (1985), Golub and Van Loan (1996), and Stewart (1998).

**Definition A.1 (Moore-Penrose pseudo-inverse).** Consider a matrix $A \in \mathbb{R}^{m \times n}$. Then the matrix $A^\dagger \in \mathbb{R}^{n \times m}$ that satisfies the four equations

\begin{align*}
AA^\dagger A &= A, \quad &\text{(A.1a)} \\
A^\dagger AA^\dagger &= A^\dagger, \quad &\text{(A.1b)} \\
(AA^\dagger)^T &= AA^\dagger, \quad &\text{(A.1c)} \\
(A^\dagger A)^T &= A^\dagger A, \quad &\text{(A.1d)}
\end{align*}

is the Moore-Penrose pseudo-inverse, or simply the pseudo-inverse of $A$. Furthermore, $A^\dagger$ is unique and if $A$ is non-singular then $A^\dagger = A^{-1}$ (Penrose, 1955).

**Lemma A.2.** Consider a positive semidefinite matrix $Q \in \mathbb{S}_+^m$. Then

$$z^T Q z = 0 \iff Q z = 0, \quad z \in \mathbb{R}^m.$$ \hspace{1cm} (A.2)

**Proof:** The relation $Q z = 0 \implies z^T Q z = 0$ is trivial. Furthermore,

$$z^T Q z = 0 \implies Q z = 0, \quad z \in \mathbb{R}^m,$$ \hspace{1cm} (A.3)

is stated as an exercise in Lancaster and Tismenetsky (1985). $\Box$
Lemma A.3. Consider the matrices $Q \in \mathbb{S}_+^m$ and $M \in \mathbb{R}^{n \times m}$. Then
\[ \mathcal{N}(Q) \subset \mathcal{N}(M) \iff \mathcal{R}\left(M^T\right) \subset \mathcal{R}(Q). \quad (A.4) \]

**Proof:** From Theorem 3.17 in Stewart (1998) and the properties of the orthogonal sum $\oplus$ described in Lancaster and Tismenetsky (1985) it follows that
\[ \mathcal{N}(Q)^\perp = \mathcal{R}\left(Q^T\right) = \mathcal{R}(Q), \quad \mathcal{N}(M)^\perp = \mathcal{R}\left(M^T\right). \quad (A.5) \]
Furthermore, for two complementary orthogonal subspaces $S_1$ and $S_2$, the relation
\[ S_1 \subset S_2 \implies S_2^\perp \subset S_1^\perp, \quad (A.6) \]
is given as an exercise in Lancaster and Tismenetsky (1985). Hence, by first choosing $S_1 = \mathcal{N}(Q)$ and $S_2 = \mathcal{N}(M)$, the $\implies$ direction in (A.4) follows. Furthermore, by exploiting that $(S^\perp)^\perp = S$ for any subspace $S$ of a unitary space (Lancaster and Tismenetsky, 1985) and choosing $S_1 = \mathcal{R}(M^T)$ and $S_2 = \mathcal{R}(Q)$, it follows that
\[ \mathcal{R}\left(M^T\right) \subset \mathcal{R}(Q) \implies \mathcal{N}(Q) \subset \mathcal{N}(M), \quad (A.7) \]
which concludes the proof. \hfill $\square$

Lemma A.4. The system of equations
\[ AX = B, \quad (A.8) \]
with $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times p}$ is solvable if and only if
\[ AA^\dagger B = B \iff \mathcal{R}(B) \subset \mathcal{R}(A). \quad (A.9) \]

**Proof:** From Theorem 2 in Penrose (1955) it follows that a solution to (A.8) exists if and only if $AA^\dagger B = B$. Since $AA^\dagger$ is the orthogonal projector onto $\mathcal{R}(A)$ (Lancaster and Tismenetsky, 1985), it follows that $AA^\dagger B = B \iff \mathcal{R}(B) \subset \mathcal{R}(A).$ \hfill $\square$

Now study a square matrix $M$ which is partitioned into blocks as
\[ M \triangleq \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, \quad (A.10) \]
where $M_{11}$ and $M_{22}$ are square matrices.

The definition of the generalized Schur complement (gsc) in Definition A.5 is repeated from Carlson et al. (1974), but with slightly different notation.

**Definition A.5 (Generalized Schur complement).** The generalized Schur complement of $M_{22}$ in $M$ is denoted $M/\dagger M_{22}$ and is defined as
\[ M/\dagger M_{22} \triangleq M_{11} - M_{12}M_{22}^\dagger M_{21}. \quad (A.11) \]
Similarly, the generalized Schur complement of \( M_{11} \) in \( M \) is defined as
\[
M / M_{11} \triangleq M_{22} - M_{21} M_{11}^T M_{12}. \tag{A.12}
\]

When \( M_{22} \), or \( M_{11} \), is non-singular the Schur complement is defined as in Definition A.6.

**Definition A.6 (Schur complement).** The Schur complement of \( M_{22} \) in \( M \) is denoted \( M / M_{22} \) and is defined as
\[
M / M_{22} \triangleq M_{11} - M_{12} M_{22}^{-1} M_{21}. \tag{A.13}
\]
Similarly, the Schur complement of \( M_{11} \) in \( M \) is defined as
\[
M / M_{11} \triangleq M_{22} - M_{21} M_{11}^{-1} M_{12}. \tag{A.14}
\]

**Assumption A.7.** The matrix \( M \) in (A.10) is non-singular.

**Theorem A.8.** Let Assumption A.7 hold. Then the inverse of \( M \) in (A.10) is
\[
M^{-1} = \begin{bmatrix}
M^{-1}_{11} + M^{-1}_{11} M_{12} (M/M_{11})^{-1} M_{21} M_{11}^{-1} & -M^{-1}_{11} M_{12} (M/M_{11})^{-1} \\
-(M/M_{11})^{-1} M_{21} M_{11}^{-1} & (M/M_{11})^{-1}
\end{bmatrix}.
\tag{A.15}
\]

**Proof:** Write \( M \) as
\[
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix} = \begin{bmatrix}
I & 0 \\
M_{21} M_{11}^{-1} & I
\end{bmatrix} \begin{bmatrix}
M_{11} & 0 \\
0 & M/M_{11}
\end{bmatrix} \begin{bmatrix}
I & M_{11}^{-1} M_{12} \\
0 & I
\end{bmatrix},
\tag{A.16}
\]
where the outer matrices on the right-hand-side are invertible. Taking the inverse of both sides gives
\[
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}^{-1} = \begin{bmatrix}
I & -M^{-1}_{11} M_{12} \\
0 & I
\end{bmatrix} \begin{bmatrix}
M^{-1}_{11} & 0 \\
0 & (M/M_{11})^{-1}
\end{bmatrix} \begin{bmatrix}
I & 0 \\
0 & M_{21} M_{11}^{-1}
\end{bmatrix} = \\
\begin{bmatrix}
M^{-1}_{11} + M^{-1}_{11} M_{12} (M/M_{11})^{-1} M_{21} M_{11}^{-1} & -M^{-1}_{11} M_{12} (M/M_{11})^{-1} \\
-(M/M_{11})^{-1} M_{21} M_{11}^{-1} & (M/M_{11})^{-1}
\end{bmatrix},
\tag{A.17}
\]
which completes the proof. \( \square \)

**Assumption A.9.** The matrix \( M \) in (A.10) is symmetric, giving
\[
M_{21} = M_{12}^T, \quad M_{11} = M_{11}^T, \quad M_{22} = M_{22}^T. \tag{A.18}
\]

The following two theorems are repeated from Albert (1969).
Theorem A.10. Consider a block partitioned matrix $M$ in (A.10) where Assumption A.9 holds. Then the following hold

\[
M \succeq 0 \iff M_{11} \succeq 0, \quad (I - M_{11}M_{11}^\dagger)M_{12} = 0, \quad M/M_{11} \succeq 0, \quad M_{12} = 0, \quad (I - M_{12}M_{12}^\dagger)M_{11} = 0.
\]  

(A.19a)

\[
M \succeq 0 \iff M_{22} \succeq 0, \quad (I - M_{22}M_{22}^\dagger)M_{12} = 0, \quad M/M_{22} \succeq 0, \quad M_{T} = 0.
\]  

(A.19b)

Proof: The proof follows from Theorem 1(i) in Albert (1969).

Theorem A.11. Consider a block partitioned matrix $M$ in (A.10) where Assumption A.9 holds. Then the following hold

\[
M \succ 0 \iff M_{11} \succ 0, \quad M_{11} = M_{22} - M_{12}M_{12}^\dagger M_{11} \succ 0, \quad M_{12} = 0, \quad (I - M_{12}M_{12}^\dagger)M_{11} = 0.
\]  

(A.20a)

\[
M \succ 0 \iff M_{22} \succ 0, \quad M_{22} = M_{11} - M_{12}M_{12}^\dagger M_{11} \succ 0.
\]  

(A.20b)

Proof: The proof follows from Theorem 1(ii) in Albert (1969).

Now consider the system of equations

\[
\begin{bmatrix}
A_1 & A_2 \\
A_2^T & A_3
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2
\end{bmatrix}
= \begin{bmatrix}
H_1 \\
H_2
\end{bmatrix}.
\]  

(A.21)

Assumption A.12. The system of equations given in (A.21) satisfies,

\[
\mathcal{R}(H) \subset \mathcal{R}(A) \iff AA^\dagger = H.
\]  

(A.22)

Assumption A.13. The matrix $A$ in (A.21) satisfies $A \succeq 0$.

Theorem A.14. Consider the system of equations in (A.21) where assumptions A.12 and A.13 hold. Then one solution $X^*$ is given by

\[
\begin{bmatrix}
X_1^* \\
X_2^*
\end{bmatrix}
= \begin{bmatrix}
A_1^\dagger & A_2 \\
A_2^T A_3
\end{bmatrix}
\begin{bmatrix}
H_1 \\
H_2
\end{bmatrix}.
\]  

(A.23)

and the following properties hold

\[
\mathcal{R}(H_1 - A_2A_3^\dagger H_2) \subset \mathcal{R}(A_1 - A_2A_3A_2^\dagger) \
\mathcal{R}(H_2 - A_2^T A_1^\dagger H_1) \subset \mathcal{R}(A_3 - A_2^T A_1 A_2).
\]  

(A.24a)

(A.24b)

Proof: From Assumption A.12 and Lemma A.4 it follows that the system of equa-
tions (A.21) is solvable. Now, factor $A$ as
\[
\begin{bmatrix}
I & 0 \\
A_2^T A_1^\dagger & I \\
0 & A/\dagger A_1 \\
A_1^\dagger A_2 & I
\end{bmatrix}
= U \tilde{A} U^T.
\] (A.25)

Note that $U$ is invertible with
\[
U^{-1} = \begin{bmatrix}
I & 0 \\
-A_2^T A_1^\dagger & I
\end{bmatrix}.
\] (A.26)

Using this factorization of $A$, re-write the system of equations (A.21) as
\[
AX = \begin{bmatrix}
I & 0 \\
A_2^T A_1^\dagger & I \\
0 & A/\dagger A_1 \\
A_1^\dagger A_2 & I
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2
\end{bmatrix} = \begin{bmatrix}
H_1 \\
H_2
\end{bmatrix}.
\] (A.27)

Since $U$ is invertible it follows that the system of equations
\[
\begin{bmatrix}
A_1 & 0 \\
0 & A/\dagger A_1
\end{bmatrix}
\begin{bmatrix}
Z_1 \\
Z_2
\end{bmatrix} = \begin{bmatrix}
W_1 \\
W_2
\end{bmatrix},
\] (A.28)

with
\[
Z \triangleq U^T X = \begin{bmatrix}
X_1 + A_1^\dagger A_2 X_2 \\
X_2
\end{bmatrix}, \quad W \triangleq U^{-1} H = \begin{bmatrix}
H_1 \\
-A_2^T A_1^\dagger H_1 + H_2
\end{bmatrix},
\] (A.29)
is also solvable (Lancaster and Tismenetsky, 1985). In Penrose (1955); Lancaster and Tismenetsky (1985) it is shown that all solutions $Z^*$ are given by
\[
\begin{bmatrix}
Z_1^* \\
Z_2^*
\end{bmatrix} = \begin{bmatrix}
A_1^\dagger W_1 + \nu_1 \\
(A/\dagger A_1)^{\dagger} W_2 + \nu_2
\end{bmatrix}, \quad \nu_1 \in \mathcal{N}(A_1), \quad \nu_2 \in \mathcal{N}(A/\dagger A_1).
\] (A.30)

By using the definition of $Z$ and $W$, $X^*$ can be computed from $Z^*$ as
\[
\begin{bmatrix}
X_1^* \\
X_2^*
\end{bmatrix} = \begin{bmatrix}
A_1^\dagger + A_1^\dagger A_2 (A/\dagger A_1)^{\dagger} A_2^T A_1^\dagger - A_1^\dagger A_2 (A/\dagger A_1)^{\dagger} \\
- (A/\dagger A_1)^{\dagger} A_2^T A_1^\dagger (A/\dagger A_1)^{\dagger}
\end{bmatrix}
\begin{bmatrix}
H_1 \\
H_2
\end{bmatrix} + \begin{bmatrix}
I & -A_1^\dagger A_2 \\
0 & I
\end{bmatrix}
\begin{bmatrix}
\nu_1 \\
\nu_2
\end{bmatrix}.
\] (A.31)

Hence, by choosing the solution corresponding to $\nu = 0$ it is clear that one solution to (A.21) is given by (A.23).

Furthermore, (A.24b) follows from the fact that (A.28) is solvable, and hence
\[
\mathcal{R}(W_2) \subset \mathcal{R}(A/\dagger A_1) \iff \mathcal{R}(H_2 - A_2^T A_1^\dagger H_1) \subset \mathcal{R}(A/\dagger A_1),
\] (A.32)
follows from Lemma A.4. Analogous calculation proves (A.24a). This concludes the proof. \hfill \Box

Note that the solution (A.23) is not in general the same as $X^* = A^\dagger H$. The requirements for them to coincide are listed in for example Burns et al. (1974).


