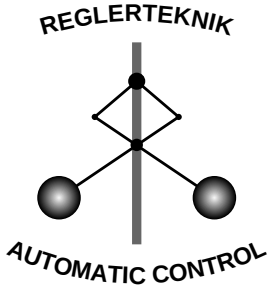
Linköping Studies in Science and Technology
Thesis No. 1218

Applications of Integer Quadratic Programming in Control and Communication

Daniel Axehill



Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
http://www.control.isy.liu.se
daniel@isy.liu.se

Linköping 2005

This is a Swedish Licentiate's Thesis.
Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree. A Doctor's Degree comprises 160 credits (4 years of full-time studies). A Licentiate's degree comprises 80 credits, of which at least 40 credits constitute a Licentiate's thesis.

Applications of Integer Quadratic Programming in Control and Communication

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping
Sweden

ISBN 91-85457-90-6 ISSN 0280-7971 LiU-TEK-LIC-2005:71

Printed by LiU-Tryck, Linköping, Sweden 2005

To my family

Abstract

The main topic of this thesis is integer quadratic programming with applications to problems arising in the areas of automatic control and communication. One of the most widespread modern control principles is the discrete-time method Model Predictive Control (MPC). The main advantage with MPC, compared to most other control principles, is that constraints on control signals and states can easily be handled. In each time step, MPC requires the solution of a Quadratic Programming (QP) problem. To be able to use MPC for large systems, and at high sampling rates, optimization routines tailored for MPC are used. In recent years, the range of application of MPC has been extended from constrained linear systems to so-called hybrid systems. Hybrid systems are systems where continuous dynamics interact with logic. When this extension is made, binary variables are introduced in the problem. As a consequence, the QP problem has to be replaced by a far more challenging Mixed Integer Quadratic Programming (MIQP) problem. Generally, for this type of optimization problems, the computational complexity is exponential in the number of binary optimization variables. In modern communication systems, multiple users share a so-called multi-access channel, where the information sent by different users is separated by using almost orthogonal codes. Since the codes are not completely orthogonal, the decoded information at the receiver is slightly correlated between different users. Further, noise is added during the transmission. To estimate the information originally sent, a maximum likelihood problem involving binary variables is solved. The process of simultaneously estimating the information sent by multiple users is called multiuser detection. In this thesis, the problem to efficiently solve MIQP problems originating from MPC is addressed. Two different algorithms are presented. First, a polynomial complexity preprocessing algorithm for binary quadratic programming problems is presented. By using the algorithm, some, or all, binary variables can be computed efficiently already in the preprocessing phase. In simulations, the algorithm is applied to unconstrained MPC problems with a mixture of real and binary control signals. It has also been applied to the multiuser detection problem, where simulations have shown that the bit error rate can be significantly reduced by using the proposed algorithm as compared to using common suboptimal algorithms. Second, an MIQP algorithm tailored for MPC is presented. The algorithm uses a branch and bound method where the relaxed node problems are solved by a dual active set QP algorithm. In this QP algorithm, the KKT-systems are solved using Riccati recursions in order to decrease the computational complexity. Simulation results show that both the QP solver and the MIQP solver proposed have lower computational complexity than corresponding generic solvers.

Acknowledgments

First of all, I would like to thank my parents Gerd and Lasse for their constant support and encouragement. Without them, I would not have been where I am today.

Also in the top of the list, my deepest gratitude to my supervisor Dr. Anders Hansson for all help and guidance during the past two and a half years towards the publication of this thesis. Anders and I have had many interesting discussions and I have learned a lot.

Another person who has a special place on this page is of course Prof. Lennart Ljung. First of all I would like to thank him for letting me join the Automatic Control group. I also appreciate his care for new Ph.D. students and his desire to make sure that there is a good balance between work and spare time. For some reason, it is very easy to spend too much time at work...

I would like to thank Dr. Fredrik Gunnarsson for the interesting discussions and good ideas behind the paper about multiuser detection.

I am sincerely grateful to the persons who have used some of their valuable time to proofread various parts of this thesis. These persons are Lic. Gustaf Hendeby, Lic. Erik Geijer Lundin, Lic. Thomas Schön, Johan Sjöberg, Henrik Tidefelt, David Törnqvist and Dr. Ragnar Wallin.

To ensure that I do not forget anyone, I would like to thank the entire Automatic Control group in Linköping. There are some persons who I would like to mention more specifically. First I would like to thank Johan Sjöberg. Johan and I did our Master's thesis project together at Scania in Södertälje. During this period we had many interesting discussions, both about control theory but also about what to do after graduation. Thanks to Johan, and all other nice persons at the group at Scania, it became clear to me that it would be interesting to continue studying as a Ph.D. student. I would also like to thank Gustaf Hendeby and David Törnqvist who joined the group at approximately the same time as I did. Especially, I would like to thank Gustaf as our TEX-guru, for all help regarding LATEX. I would also like to thank him for the excellent thesis style in which this thesis is formatted.

A thank is directed to Erik Geijer Lundin for revealing the secrets behind CDMA and multiuser detection. Erik and I have also had many interesting discussions about our common interest, boats. Dr. Jacob Roll earns a thank for all help regarding MIQP. Another person who also does not slip away my directed thanks is Ragnar Wallin. Ragnar is an extremely kind and helpful person who has helped me with many things and answered many questions.

Of course I have to thank my room mates Lic. Erik Wernholt and Henrik Tidefelt. Erik guided me as a new Ph.D. student and Henrik helped me during the work with the Riccati based QP solver.

Financial support by VINNOVA's Center of Excellence ISIS (Information Systems for Industrial Control and Supervision) and fruitful discussions with the industrial partner ABB are gratefully acknowledged. In addition to this, financial support from ECSEL (The Excellence Center in Computer Science and Systems Engineering in Linköping) is also gratefully acknowledged.

Linköping, November 2005
Daniel Axehill

Contents

1 Introduction **5**
- 1.1 Background and Motivation . . . 6
- 1.2 Contributions . . . 7
- 1.3 Thesis Outline . . . 7

2 Optimization **9**
- 2.1 Introduction and Basic Concepts . . . 9
- 2.2 Duality . . . 12
  - 2.2.1 The Lagrange Dual Problem . . . 12
  - 2.2.2 Weak and Strong Duality . . . 13
- 2.3 Optimality Conditions . . . 14
- 2.4 Quadratic Programming . . . 15
  - 2.4.1 Strong Duality for Convex Quadratic Programming . . . 17
  - 2.4.2 Active Set Methods . . . 19
  - 2.4.3 Dual Active Set Quadratic Programming Methods . . . 22
- 2.5 Mixed Integer Quadratic Programming . . . 25
  - 2.5.1 Problem Definition . . . 26
  - 2.5.2 Branch and Bound . . . 26
- 2.6 Binary Quadratic Programming . . . 31

3 Mixed Integer Predictive Control **33**
- 3.1 Model Predictive Control . . . 33
- 3.2 Mixed Logical Dynamical Systems . . . 36
  - 3.2.1 Background . . . 36
  - 3.2.2 The MLD System Description . . . 37
  - 3.2.3 Controlling MLD Systems . . . 38
  - 3.2.4 Moving Horizon Estimation for MLD Systems . . . 38

3.3 Optimization in Model Predictive Control . . . 39
3.3.1 Quadratic Programming . . . 39
3.3.2 Active Set versus Interior Point . . . 40
3.3.3 Mixed Integer Quadratic Programming . . . 41
3.4 Two Examples of Mixed Logical Dynamical Systems . . . 43
3.4.1 Mass Position Control . . . 43
3.4.2 Satellite Attitude Control . . . 43

4 Multiuser Detection in a Code Division Multiple Access System 47
4.1 Multiuser Detection . . . 47
4.2 Synchronous Code Division Multiple Access . . . 48
4.2.1 System Model . . . 48
4.2.2 Derivation of the BQP Problem . . . 50

5 A Preprocessing Algorithm for Mixed Integer Quadratic Programming 51
5.1 A Preprocessing Algorithm for BQP and MIQP Problems . . . 51
5.1.1 The BQP and MIQP Problems . . . 52
5.1.2 Preprocessing for the BQP Problem . . . 52
5.1.3 Preprocessing for the MIQP Problem . . . 56
5.1.4 Implementation . . . 56
5.2 Application of the Preprocessing Algorithm to Model Predictive Control . 57
5.2.1 Using Preprocessing . . . 57
5.2.2 Simulation Results . . . 59
5.3 Application of the Preprocessing Algorithm to Multiuser Detection . . . 61
5.3.1 Using Preprocessing . . . 62
5.3.2 Interpretation of the Result . . . 62
5.3.3 Simulation Results . . . 63

6 A Mixed Integer Dual Quadratic Programming Algorithm 71
6.1 A Dual Quadratic Programming Algorithm . . . 71
6.1.1 Problem Definition . . . 71
6.1.2 Derivation of the Dual Problem . . . 72
6.1.3 Optimality Conditions for the Dual Problem . . . 76
6.1.4 Connection Between Primal and Dual Variables . . . 77
6.1.5 Solving the Dual Problem Using Riccati Recursions . . . 79
6.1.6 Handling Parallel Constraints . . . 85
6.1.7 Increasing Performance . . . 87
6.1.8 Future Extensions . . . 88
6.1.9 Simulation Results . . . 88
6.2 A Mixed Integer Quadratic Programming Algorithm . . . 90
6.2.1 Increasing Performance . . . 93
6.2.2 Future Extensions . . . 93
6.2.3 Simulation Results . . . 93

7 Concluding Remarks 97
7.1 Conclusions . . . 97
7.2 Future Work . . . 98

Bibliography **101**

A Linear Algebra **111**

B Model Predictive Control Formulations **113**
B.1 Complete Set of Variables 113
B.2 Reduced Set of Variables 114

Notational Conventions

Symbols, Operators and Functions

Notation	Meaning
$A \succ (\succeq) 0$	A positive (semi)definite matrix.
$x \geq (>) y$	Componentwise (strict) inequality for vectors x and y. Reduces to a scalar inequality when x and y are scalars.
A^T	Transpose of matrix A.
A^{-1}	Inverse of matrix A.
$\mathrm{diag}(X_1, X_2, \ldots)$	Block diagonal matrix with matrices X_1, X_2,... along the diagonal.
$\mathrm{diag}(x_1, x_2, \ldots)$	Diagonal matrix with diagonal elements x_1, x_2,....
$\mathrm{diag}(x)$	Diagonal matrix with diagonal elements x_1, x_2,....
I, (I_n)	Identity matrix (with n rows).
0	When used in a block of a matrix or of a vector, 0 denotes a matrix, or a vector, with all elements equal to zero.
$\mathbf{1}$, $(\mathbf{1}_n)$	A vector (with n components) with all components equal to one.
$A_{(i,:)}$	Row i of matrix A (MATLAB-notation).
$A_{(:,i)}$	Column i of matrix A (MATLAB-notation).
x_i	Component i of vector x.
$\mathrm{rank}\, A$	Rank of matrix A.
$\|x\|_Q$	Weighted ℓ^2-norm for vector x with weight matrix Q.
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2.
$\mathsf{cov}\,(n(t), n(\tau))$	Covariance between random variables $n(t)$ and $n(\tau)$.
$\underset{x}{\mathrm{argmin}}\, f(x)$	The optimal solution to $\min_x f(x)$.

Notation	Meaning
$\operatorname{dom} f(x)$	Domain of function f.
sign	Signum function.
L	Lagrangian.
g	Lagrange dual function.
p^*	Optimal primal objective function value.
d^*	Optimal dual objective function value.
κ	A constant.
$\operatorname{relint} \mathcal{C}$	Relative interior of set $\mathcal{C}$.
$\lvert\mathcal{C}\rvert$	Cardinality of set $\mathcal{C}$.
$\mathcal{C}^{\complement}$	Complement of set $\mathcal{C}$.
$\lfloor x \rfloor$	The floor function. Gives the largest integer less than or equal to x.

Sets

Notation	Meaning
$\mathbb{R}$	Set of real numbers.
$\mathbb{R}^n$	Set of real vectors with n components.
$\mathbb{R}^{n \times m}$	Set of real matrices with n rows and m columns.
$\mathbb{Z}$	Set of integers.
$\mathbb{S}^n$	Set of symmetric matrices with n rows.
$\mathbb{S}^n_+$	Set symmetric positive semidefinite matrices with n rows.
$\mathbb{S}^n_{++}$	Set symmetric positive definite matrices with n rows.
$\{0,1\}^n$	Set of vectors with n binary components.

Abbreviations and Acronyms

Abbreviation	Meaning
ACC	Adaptive Cruise Control
BER	Bit Error Rate
BQP	Binary Quadratic Programming
CDMA	Code Division Multiple Access
CP	Constraint Programming
DMC	Dynamic Matrix Control
ELC	Extended Linear Complementarity
FDMA	Frequency Division Multiple Access
GBD	Generalized Benders Decomposition
IP	Interior Point
KKT	Karush-Kuhn-Tucker
LC	Linear Complementarity
LP	Linear Programming

Abbreviation	Meaning
LQR	Linear Quadratic Regulator
MBQP	Mixed Binary Quadratic Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MIP	Mixed Integer Programming
MIPC	Mixed Integer Predictive Control
MIQP	Mixed Integer Quadratic Programming
ML	Maximum Likelihood
MLD	Mixed Logical Dynamical
MMPS	Max-Min-Plus-Scaling
MPC	Model Predictive Control
mp	multi-parametric
MUD	Multiuser Detection
OA	Outer Approximation
PWA	Piecewise Affine
QCQP	Quadratically Constrained Quadratic Programming
QIP	Quadratic Integer Programming
QP	Quadratic Programming
SNR	Signal to Noise Ratio
SOCP	Second Order Cone Programming
SQP	Sequential Quadratic Programming
TDMA	Time Division Multiple Access

1

Introduction

Already from the very beginning, man has had a wish to control phenomena in her surrounding. Through the years, she has learned how to act and how to affect things to make them behave as desired. As this knowledge has grown, more advanced courses of events have become possible to control. With modern technology, various kinds of processes can be controlled. Half a million years ago it was considered challenging to control fire. Today, it is considered challenging to control fusion processes and autonomous airplanes.

Without thinking about it, most people are constantly trying to do things in an optimal way. It can be anything from looking for discounts to minimize the cost at the weekly shopping tour, to finding the shortest path between two cities. When to choose between a long queue and a short queue, most people choose the short one in order to minimize the time spent in the queue. Most of these everyday problems are solved by intuition and it is often not crucial to find the absolutely best solution. These are all examples of simple optimization problems. Unfortunately, there are many important optimization problems not that easy to solve. Optimization is used in many areas and is in many cases a very powerful tool. Common, and more advanced, examples are to minimize the weight of a construction while maintaining the desired strength or to find the optimal route for an airplane to minimize the fuel consumption. In these cases, it can be impossible to solve the problems by intuition. Instead, a mathematical algorithm executed in a computer, an optimization routine, is often applied to the problem.

In this thesis, control is combined with optimization. The desire is to control optimally, in some sense. A common optimal control problem is, in words, to make the controlled process follow a desired trajectory, while minimizing the power applied. Often, the words process and system are used interchangeable. A classical controller found by optimization is the widely used so-called Linear Quadratic Regulator (LQR). In that framework, a linear system is assumed to be controlled. In practice, all systems are, more or less, non-linear. Therefore, in order to be able to fit into the LQR framework, they are treated as approximately linear systems. A very frequently occurring non-linearity in

practical applications is that it is not possible to affect the system arbitrarily much. For example, when using full throttle in a car, the car cannot accelerate any faster. Such a limitation is called a constraint. As a consequence, a desire has been to extend LQR to handle systems with constraints. In the past twenty years, such an extension has been developed and it is commonly known as Model Predictive Control (MPC).

When classical physical processes and computers interact, more advanced optimization problems have to be solved in order to use MPC. In such systems, some ways of affecting the system can only be made in the notion of on or off. In more advanced examples, logical rules are embedded in the system. To be able to control optimally, the control has to be aware of these rules and take them into account when the optimal action is being computed. How to solve these more advanced optimization problems is the main topic of this thesis.

After this popular scientific introduction, a more precise background is given in Section 1.1. The contributions constituting the foundation for some parts of this thesis are summarized in Section 1.2. This chapter is concluded by a thesis outline given in Section 1.3.

1.1 Background and Motivation

MPC is one of the most widespread modern control principles used in industry. One of the main reasons for its acceptance is that it combines the ability of controlling multivariable systems with the ability to easily add constraints on states and control signals in the system. One of the main ideas behind MPC is to formulate a discrete-time, finite horizon, control problem similar to LQR as a Quadratic Programming (QP) problem. In the QP framework, linear constraints on control signals and states can easily be formulated as linear inequalities. In order to get a control signal to apply to the system, in each discrete time step, a QP problem is solved on-line. Because the optimization is performed on-line, there is a need for efficient QP optimization routines. As the optimization routines get more efficient, and the hardware more powerful, larger systems at faster sampling rates are possible to control by MPC.

Recently, the interest of using MPC for controlling systems containing a mix of continuous dynamics and logical rules has arisen. Unfortunately, when this problem is formulated as an optimization problem, the resulting optimization problem is no longer a QP problem but a Mixed Integer Quadratic Programming (MIQP) problem. These problems involve binary variables, which makes the problem much harder to solve than an ordinary QP problem. Therefore, there has emerged a need for efficient optimization routines for MIQP problems. In the state-of-the-art QP solvers for MPC, the problem structure is utilized in order to decrease the computational effort needed. The need for efficient optimization routines for MPC involving binary variables forms the main motivation for this thesis, where MIQP methods tailored for MPC are considered.

In modern communication systems, several users share a so-called multi-access channel, where the information sent by different users is separated by the use of orthogonal, or almost orthogonal, codes. In modern communication, the information is represented as a sequence of bits, that is, zeros and ones. At the receiver, it is natural to search for the information most likely sent by the sender. This problem can be formulated using statistical

methods and the resulting problem is an optimization problem where a quadratic objective is to be minimized and the optimization variables are the bits sent by the users. If all users are considered simultaneously, the problem is a Multiuser Detection (MUD) problem. Since the bits are examples of binary variables, the resulting problem is a so-called Binary Quadratic Programming (BQP) problem, which can be considered as a special case of an MIQP problem, but where only binary optimization variables are present and where there are no constraints.

1.2 Contributions

This thesis is based on both previously published, [3, 4, 6], and previously unpublished results.

The first contribution, [3, 4], is a preprocessing algorithm for BQP problems with applications to MPC:

> D. Axehill and A. Hansson. A preprocessing algorithm for MIQP solvers with applications to MPC. In *Proceedings of the 43th IEEE Conference on Decision and Control*, pages 2497–2502, Atlantis, Paradise Island, Bahamas, Dec. 2004.

The second contribution, [6], is the application of the preprocessing algorithm to the MUD problem:

> D. Axehill, F. Gunnarsson, and A. Hansson. A preprocessing algorithm applicable to the multiuser detection problem. In *Proceedings of RadioVetenskap och Kommunikation*, Linköping, Sweden, June 2005.

These contributions form Chapter 5. The third contribution is a dual active set QP solver which is presented in Chapter 6. The solver uses Riccati recursions in order to efficiently solve the Karush-Kuhn-Tucker (KKT) systems that arise when the algorithm moves towards the optimal solution. Further, it is used as a solver for the subproblems in a branch and bound algorithm applicable to MPC involving binary variables.

1.3 Thesis Outline

This thesis is organized as follows. Chapters 2 to 4 provide background information. Chapter 2 contains the necessary optimization background. Chapter 3 starts with an introduction to linear MPC, followed by an extension of the method to Mixed Logical Dynamical (MLD) systems. Also, different optimization methods for linear MPC as well as for MPC for MLD systems are surveyed. The chapter is concluded by the introduction of two examples of MLD systems to be used throughout the thesis. It provides the necessary background information for Section 5.2 and Chapter 6. In Chapter 4, MUD and Code Division Multiple Access (CDMA) are explained. This chapter also serves as background information for Section 5.3. In Chapter 5, a preprocessing algorithm applicable to BQP problems is derived and it is applied to MPC and MUD. In Chapter 6, an MIQP solver based on branch and bound is presented. As a part of the chapter, a dual active set QP solver tailored for MPC is presented. Finally, Chapter 7 summarizes the conclusions from different parts of the thesis and proposes some possible future extensions.

2

Optimization

Optimization is the procedure of finding an optimal solution to a problem. The optimal solution is the best solution in some sense. In what sense it can be considered best is given by the choice of the objective function, or cost function. The cost function can for example be chosen as the cost for producing a product or it can be the road distance between two places. Often there are restrictions on which solutions that are allowed. For example, if the shortest road between two places is sought, it is natural to restrict the possible solutions to those not suggesting breaking the law by proposing a one-way road in the illegal direction. Such restrictions are called constraints.

In this chapter, basic notions in optimization are presented. The notation is chosen similar to the one in [34].

2.1 Introduction and Basic Concepts

This section is opened with three fundamental definitions of a convex set, a convex function and a concave function. The first two definitions are illustrated in Figure 2.1.

Definition 2.1 (Convex set). A set $\mathcal{C}$ is convex if for any $x_1, x_2 \in \mathcal{C}$ and any $\theta \in [0, 1]$

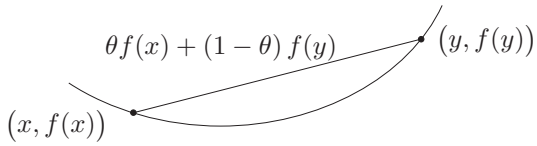
$$\theta x_1 + (1 - \theta)\, x_2 \in \mathcal{C} \tag{2.1}$$

Convex sets are thoroughly discussed in, for example, [34]. In that reference, important examples of convex functions are given as well as operations that preserve convexity.

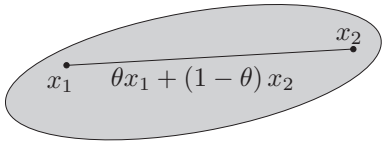
Definition 2.2 (Convex function). A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathrm{dom}\, f$ is a convex set and if for all $x, y \in \mathrm{dom}\, f$ and $\theta \in [0, 1]$

$$f\big(\theta x + (1 - \theta)\, y\big) \leq \theta f(x) + (1 - \theta)\, f(y) \tag{2.2}$$

If this inequality holds strictly whenever $x \neq y$ and $\theta \in]0, 1[$, the function f is strictly convex.



(a) *Convex function*



(b) *Convex set*

Figure 2.1: *Illustrations of the Definitions 2.1 and 2.2.*

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.

In this thesis, an optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{2.3}$$

where $f_0 : \mathbb{R}^n \to \mathbb{R}$, $f_i : \mathbb{R}^n \to \mathbb{R}$ and $h_i : \mathbb{R}^n \to \mathbb{R}$ is said to be on standard form. The function $f_0(x)$ is called the objective function, or cost function, $f_i(x),\ i = 1, \ldots, m$ denote the inequality constraint functions and $h_i(x),\ i = 1, \ldots, p$ denote the equality constraint functions. If there are no constraints, the problem is said to be unconstrained. The domain of the optimization problem is the intersection of the domains of the objective function and the constraint functions

$$\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} f_i \cap \bigcap_{i=1}^{p} \operatorname{dom} h_i \tag{2.4}$$

If a point $x \in \mathcal{D}$ satisfies all equality constraints and all inequality constraints, it is said to be feasible. If there exists at least one such point, the problem is said to be feasible. Otherwise, the problem is said to be infeasible.

An important special case of (2.3) is when the functions $f_i(x),\ i = 0, \ldots, m$ are convex and the functions $h_i(x),\ i = 1, \ldots, p$ are affine, that is $h_i(x) = a_i^T x - b_i$. Then (2.3) is called a convex optimization problem. Since the objective function is convex and the intersection of the sets defined by the constraints is convex, a convex optimization problem means that a convex function is minimized over a convex set. A fundamental property of convex optimization problems is that a local optimal solution is also a global optimal solution.

Define the optimal objective function value p^* of (2.3) as

$$p^* = \inf \{ f_0(x) \mid f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p \} \tag{2.5}$$

where p^* is allowed to take on the values $+\infty$ and $-\infty$. A point x^* is called an optimal point, or an optimal solution, to (2.3) if x^* is feasible and $f_0(x^*) = p^*$. If there exists an optimal solution to (2.3), the optimal value is said to be attained, or achieved. If there does not exist any optimal point, the optimal value is not attained. If the problem is unbounded from below, that is $p^* = -\infty$, the optimal objective function value is not attained.

Example 2.1

Consider the unconstrained optimization problem

$$\underset{x}{\text{minimize}} \quad x^2 \tag{2.6}$$

The optimal solution is $x^* = 0$ and the optimal objective function value $p^* = 0$ is attained.

Example 2.2

As an example of a problem where the optimal objective function value is not attained, consider

$$\underset{x}{\text{minimize}} \quad \arctan(x) \tag{2.7}$$

where $p^* = -\frac{\pi}{2}$, but the optimal objective function value is not attained.

The domain of a convex function can be included in the definition of the function by defining the function value to $+\infty$ outside the domain

$$\tilde{f}(x) = \begin{cases} f(x), & x \in \operatorname{dom} f \\ \infty, & x \notin \operatorname{dom} f \end{cases} \tag{2.8}$$

Here, $\tilde{f}(x)$ is called the extended-value extension of the convex function $f(x)$. The domain of the unextended function can be recovered by

$$\operatorname{dom} f = \left\{ x \mid \tilde{f}(x) < \infty \right\} \tag{2.9}$$

In this thesis, all convex functions are assumed extended. Another important concept is equivalent problems.

Definition 2.4 (Equivalent problems). Two optimization problems are said to be equivalent if the optimal solution of the two problems coincide, or the solution of the first problem can be trivially computed from the solution of the second problem and vice versa.

To illustrate Definition 2.4, a simple example of later conceptual relevance is shown.

Example 2.3

Consider the following unconstrained quadratic optimization problem

$$\underset{x}{\text{minimize}} \quad C_1 x^2 + C_2 \tag{2.10}$$

An infinite number of equivalent optimization problems to (2.10) can be found by varying $C_1 > 0$ and C_2. That is, all of them has the same optimal solution. It is extremely important to realize that they are not the same problems. For example, in this case the optimal objective function value varies with C_1 and C_2.

One application of equivalent problems is to consider a simpler, but equivalent, problem compared to the original problem. For example, choosing $C_2 = 0$ reduces problem (2.10) to a problem with only a pure quadratic term.

2.2 Duality

The concept of duality is very important in optimization. The objective by considering a dual problem is to get an alternative formulation of the optimization problem that is computationally more attractive or has some theoretical significance, [43]. When discussing duality, no assumption of convexity has to be made, even though such an assumption enables the use of more powerful results. Early work on duality for non-linear programming can be found in, for example, [38, 39, 100].

2.2.1 The Lagrange Dual Problem

In the derivation of a dual optimization problem, the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with (2.3) plays an important role. The Lagrangian is defined as

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \tag{2.11}$$

where $\operatorname{dom} L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. The variables λ_i and ν_i are the Lagrange multipliers associated with inequality constraint i and equality constraint i, respectively. The vectors of Lagrange multipliers λ and ν are called the dual variables associated with problem (2.3).

When the Lagrangian is minimized with respect to the primal variables for a given λ and ν, the Lagrange dual function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right) \tag{2.12}$$

is obtained.

An important operation conserving convexity properties is the pointwise infimum of a set of concave functions, which is a concave function. Since the Lagrange dual function is affine in (λ, ν), it is precisely a pointwise infimum of a set of concave functions and is therefore concave. This holds without any assumptions of convexity of the problem (2.3), that is, the Lagrange dual function is a concave function also in the case when (2.3) is not a convex problem.

An important property of the Lagrange dual function is that for any $\lambda \geq 0$, the following inequality holds

$$g(\lambda, \nu) \leq p^* \tag{2.13}$$

That is, the dual function gives lower bounds on the optimal objective function value. Actually, the dual function gives lower bounds on the objective function value for all feasible x. When $g(\lambda, \nu) = -\infty$, the inequality (2.13) still holds, but it is vacuous.

Since (2.13) for $\lambda \geq 0$ gives lower bounds on the optimal objective function value, it is interesting to find the pair (λ, ν) that gives the best lower bound. This pair can be found as the solution to the optimization problem

$$\begin{aligned} \underset{\lambda,\nu}{\text{maximize}} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.14}$$

This problem is called the Lagrange dual problem associated with (2.3). Note that there exist different dual problems. Example of other dual formulations for non-linear programs are the Wolfe dual, [100], and the Dorn dual for quadratic programs, [38]. In this thesis, the Lagrange dual will be used exclusively, hence the word dual will be used, without ambiguity, as short for the Lagrange dual. To summarize the terminology, (2.3) is called the primal problem and (2.14) is called the dual problem. A pair (λ, ν) is called dual feasible if $\lambda \geq 0$ and $g(\lambda, \nu) > -\infty$. Since the objective function to be maximized in (2.14) is concave and the feasible set is convex, the dual optimization problem is a convex problem independently of whether the primal problem (2.3) is convex or not. The optimal dual objective function value is denoted d^*.

2.2.2 Weak and Strong Duality

In the previous section, it was seen that by solving the dual problem, the best possible lower bound on the primal optimal objective function value can be found. The inequality (2.13) holds specifically for the dual optimal pair (λ^*, ν^*) and thus

$$d^* \leq p^* \tag{2.15}$$

This inequality is called weak duality. Weak duality holds even if the primal problem is non-convex and it still holds if d^* or p^* are infinite. Using the extended-value extension, infinite values can be interpreted as primal or dual infeasibilities. For example, if the primal is unbounded from below, that is $p^* = -\infty$, it follows from (2.15) that $d^* = -\infty$, which means that the dual is infeasible. The difference $p^* - d^*$ is called the optimal duality gap and is always non-negative.

For some problems the inequality (2.15) holds with equality, that is,

$$d^* = p^* \tag{2.16}$$

which means that the lower bound found from the dual problem is tight and the duality gap is zero. This important property is called strong duality. Unfortunately, strong duality does not hold in general. It often holds for convex problems, but not necessarily. Conditions guaranteeing strong duality are called constraint qualifications. One well-known constraint qualification is Slater's condition. For a convex optimization problem, Slater's theorem stated below holds. Let the primal problem be of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \qquad i = 1, \ldots, m \\ & Ax = b \end{aligned} \tag{2.17}$$

where $f_0, \ldots, f_m$ are convex functions. Slater's theorem can then provide conditions under which strong duality holds. The domain is assumed $\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} f_i$. The following two theorems are given without proofs and are based on the discussion in [34, p. 226].

Theorem 2.1 (Slater's theorem)
For the convex optimization problem (2.17)*, strong duality holds if there exists an* $x \in \operatorname{relint} \mathcal{D}$ *such that*

$$f_i(x) < 0, \ i = 1, \ldots, m, \qquad Ax = b \tag{2.18}$$

A verbal formulation of Theorem 2.1 is that if there exist strictly feasible primal points, strong duality holds. If some of the inequality constraints are affine, then it is sufficient that a weaker condition holds. Theorem 2.1 can be refined for the case when the inequality functions are affine. In that case, the affine inequalities do not need to hold strictly.

Theorem 2.2 (Slater's theorem, refined)
For the convex optimization problem (2.17)*, strong duality holds if there exists an* $x \in \operatorname{relint} \mathcal{D}$ *such that*

$$f_i(x) \leq 0, \ i = 1, \ldots, k, \qquad f_i(x) < 0, \ i = k+1, \ldots, m, \qquad Ax = b \tag{2.19}$$

where $f_i(x), \ i = 1, \ldots, k$ *are affine functions.*

Remark 2.1. If all constraints are affine, and $\operatorname{dom} f_0$ is open, then condition (2.19) in Theorem 2.2 is reduced to feasibility of the primal problem.

A consequence of Theorem 2.1 and Theorem 2.2 is that the dual optimal objective function value is attained whenever $d^* > -\infty$, that is, whenever the dual is feasible.

2.3 Optimality Conditions

In this thesis the so-called Karush-Kuhn-Tucker (KKT) conditions for optimality are used. In the general setup, they can be used as necessary conditions for optimality for any optimization problem with differentiable objective function and constraint functions for which strong duality holds. If the problem is convex they are also sufficient according to the following theorem, based on the discussion in [34, pp. 243–244].

Theorem 2.3 (KKT)
Consider the optimization problem (2.3)*. Assume that it is convex, that* $f_i(x), \ i = 0, \ldots, m$ *are differentiable and that strong duality holds. Then the following so-called Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions for* x^* *and* (λ^*, ν^*) *to be primal respectively dual optimal points*

$$f_i(x^*) \leq 0, \qquad i = 1, \ldots, m \tag{2.20a}$$
$$h_i(x^*) = 0, \qquad i = 1, \ldots, p \tag{2.20b}$$
$$\lambda_i^* \geq 0, \qquad i = 1, \ldots, m \tag{2.20c}$$
$$\lambda_i^* f_i(x^*) = 0, \qquad i = 1, \ldots, m \tag{2.20d}$$
$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0 \tag{2.20e}$$

Proof: See [34, p. 244]. □

2.4 Quadratic Programming

As a prelude to this section, a general form of a Quadratic Programming (QP) problem is introduced

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}} \\ & A_{\mathcal{I}} x \leq b_{\mathcal{I}} \end{aligned} \tag{2.21}$$

where $x \in \mathbb{R}^n$, $H \in \mathbb{S}^n_{++}$, $f \in \mathbb{R}^n$ and the rows in $A_{\mathcal{E}} \in \mathbb{R}^{p \times n}$ are given by the vectors in $\{a_i \in \mathbb{R}^n \mid i \in \mathcal{E}\}$ and the rows in $A_{\mathcal{I}} \in \mathbb{R}^{m \times n}$ are given by the vectors in $\{a_i \in \mathbb{R}^n \mid i \in \mathcal{I}\}$. The column vectors $b_{\mathcal{E}}$ and $b_{\mathcal{I}}$ are analogously defined. The sets $\mathcal{I}$ and $\mathcal{E}$ are finite sets of indices. The Lagrange dual function to (2.21) can be found by first forming the Lagrangian

$$L(x, \lambda, \nu) = \frac{1}{2}x^T H x + f^T x + \lambda^T \left(A_{\mathcal{I}} x - b_{\mathcal{I}}\right) + \nu^T \left(A_{\mathcal{E}} x - b_{\mathcal{E}}\right) \tag{2.22}$$

and then minimizing with respect to the primal variables. Since the $H \succ 0$, the unique minimizer can be found from the first order necessary and sufficient conditions of optimality

$$\frac{\partial L(x, \lambda, \nu)}{\partial x} = Hx + f + A_{\mathcal{I}}^T \lambda + A_{\mathcal{E}}^T \nu = 0 \;\Leftrightarrow\; x = -H^{-1}\left(f + A_{\mathcal{I}}^T \lambda + A_{\mathcal{E}}^T \nu\right) \tag{2.23}$$

Inserting (2.23) into (2.22) gives the following expression for the dual function

$$\begin{aligned} g(\lambda, \nu) = & -\frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} \\ & - \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \frac{1}{2}f^T H^{-1} f \end{aligned} \tag{2.24}$$

Using (2.14), the dual problem is found to be

$$\begin{aligned} \underset{\lambda,\nu}{\text{maximize}} \quad & -\frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \\ & - \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \frac{1}{2}f^T H^{-1} f \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.25}$$

By changing the sign of the objective function and ignoring the constant term, (2.25) can be written as an equivalent (see Definition 2.4) minimization problem

$$\begin{aligned} \underset{\lambda,\nu}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} + \\ & + \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.26}$$

Remark 2.2. Note that the optimal solutions of (2.25) and (2.26) coincide. But, the optimal objective function *values* do not generally coincide. This is extremely important to remember when working with weak and strong duality results. These results relate the optimal objective function value of (2.21) to the optimal objective function value of (2.25), but not to the optimal objective function value of (2.26).

The great structural advantage with the dual problem (2.25), or (2.26), compared to the primal problem (2.21), is that the latter only has simple non-negativity constraints. A consequence of the simple constraint structure is that the origin is always a feasible solution. Furthermore, the simple structure of the constraints enables the use of the efficient gradient projection algorithm, which allows more rapid changes to the working set (see Section 2.4.2) compared to a classical active set algorithm, [77]. More advantages of the dual formulation are presented in Section 2.4.3.

In this thesis, a variant of (2.21) will be of great interest

$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix} x_1^T & x_2^T \end{bmatrix}\begin{bmatrix} \tilde{H} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \tilde{f}^T & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} A_{\mathcal{E},1} & A_{\mathcal{E},2} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = b_{\mathcal{E}} \\ & \begin{bmatrix} A_{\mathcal{I},1} & A_{\mathcal{I},2} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq b_{\mathcal{I}} \end{aligned} \tag{2.27}$$

where $x_1 \in \mathbb{R}^{n_1}$, $x_2 \in \mathbb{R}^{n_2}$, $\tilde{H} \in \mathbb{S}^{n_1}_{++}$ and $\tilde{f} \in \mathbb{R}^{n_1}$. The dual problem of (2.27) is derived in a similar manner as the dual problem of (2.21), but the derivation becomes slightly complicated by the fact that the Hessian is not positive definite. First, the Lagrangian is derived

$$\begin{aligned} & L(x_1, x_2, \lambda, \nu) \\ & = \frac{1}{2}x_1^T\tilde{H}x_1 + \tilde{f}^T x_1 + \lambda^T\left(A_{\mathcal{I}}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - b_{\mathcal{I}}\right) + \nu^T\left(A_{\mathcal{E}}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - b_{\mathcal{E}}\right) \\ & = \frac{1}{2}x_1^T\tilde{H}x_1 + \tilde{f}^T x_1 + \lambda^T A_{\mathcal{I},1}x_1 - \lambda^T b_{\mathcal{I}} + \nu^T A_{\mathcal{E},1}x_1 - \nu^T b_{\mathcal{E}} \\ & + \left(\lambda^T A_{\mathcal{I},2} + \nu^T A_{\mathcal{E},2}\right)x_2 \end{aligned} \tag{2.28}$$

The Lagrange dual function is found by minimizing the Lagrangian with respect to the primal variables

$$\begin{aligned} g(\lambda,\nu) &= \inf_{x_1,x_2\in\mathcal{D}} L(x_1,x_2,\lambda,\nu) \\ &= \begin{cases} \inf\limits_{x_1\in\mathbb{R}^{n_1}} \frac{1}{2}x_1^T\tilde{H}x_1 + \left(\tilde{f}^T + \lambda^T A_{\mathcal{I},1} + \nu^T A_{\mathcal{E},1}\right)x_1 - \lambda^T b_{\mathcal{I}} - \nu^T b_{\mathcal{E}}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } \lambda^T A_{\mathcal{I},2} + \nu^T A_{\mathcal{E},2} = 0 \\ -\infty, \text{ otherwise} \end{cases} \end{aligned} \tag{2.29}$$

According to (2.29), if L is to be bounded from below the following condition has to be fulfilled

$$\lambda^T A_{\mathcal{I},2} + \nu^T A_{\mathcal{E},2} = 0 \tag{2.30}$$

If (2.30) is inserted into (2.28), the Lagrangian becomes a strictly convex function of x_1, λ and ν. First order necessary and sufficient conditions for optimality with respect to x_1 are

$$\frac{\partial L(x_1, \lambda, \nu)}{\partial x_1} = \tilde{H}x_1 + \tilde{f} + A_{\mathcal{I},1}^T\lambda + A_{\mathcal{E},1}^T\nu = 0 \tag{2.31}$$

or equivalently

$$x_1 = -\tilde{H}^{-1}\left(\tilde{f} + A_{\mathcal{I},1}^T\lambda + A_{\mathcal{E},1}^T\nu\right) \tag{2.32}$$

When formulating the dual problem, the implicit constraint in (2.30) is made explicit by adding it to the list of constraints. After inserting (2.32) into (2.29), the dual problem is concluded to be

$$\begin{aligned} \underset{\lambda,\nu}{\text{maximize}} \quad & -\frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I},1}\\ A_{\mathcal{E},1}\end{bmatrix}\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix}\begin{bmatrix}\lambda\\ \nu\end{bmatrix} - \\ & -\left(\tilde{f}^T\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda\\ \nu\end{bmatrix} - \frac{1}{2}\tilde{f}^T\tilde{H}^{-1}\tilde{f} \\ \text{subject to} \quad & A_{\mathcal{I},2}^T\lambda + A_{\mathcal{E},2}^T\nu = 0 \\ & \lambda \geq 0 \end{aligned} \tag{2.33}$$

By changing the sign of the objective and removing the constant term, a problem equivalent to the dual problem is

$$\begin{aligned} \underset{\lambda,\nu}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I},1}\\ A_{\mathcal{E},1}\end{bmatrix}\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix}\begin{bmatrix}\lambda\\ \nu\end{bmatrix} + \\ & + \left(\tilde{f}^T\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda\\ \nu\end{bmatrix} \\ \text{subject to} \quad & A_{\mathcal{I},2}^T\lambda + A_{\mathcal{E},2}^T\nu = 0 \\ & \lambda \geq 0 \end{aligned} \tag{2.34}$$

For an extensive bibliography on QP, see [51].

2.4.1 Strong Duality for Convex Quadratic Programming

Early work on duality for QPs can be found in [38], [39] and [37]. Since the constraints of a QP are linear, it follows from Theorem 2.2 that if the primal problem is feasible, strong duality holds. Sometimes the primal optimal solution can be derived from the dual optimal solution. In those cases, it can sometimes be advantageous to solve the dual problem instead of the primal problem. When the dual optimal solution has been found, it can be used to easily compute the primal optimal solution. If this approach is used, it is important to know what will happen in the dual problem if the primal problem does not have any solution. In this section, the primal problem (2.27) and the dual problem (2.33) are considered. The desirable situation is that the dual problem has a solution if and only if the primal problem has a solution. The primal problem considered has an objective function that is bounded from below.

Consider feasibility of the primal and the dual problem. Four mutually exclusive cases can occur:

Case	Primal	Dual
1	Feasible	Feasible
2	Infeasible	Infeasible
3	Feasible	Infeasible
4	Infeasible	Feasible

Since the problem considered has an objective function value that is bounded from below and strong duality holds, case 3 can never occur.

From strong duality, in case 1 the primal and dual optimal objective function values coincide.

For case 4, it will now be shown that the dual optimal objective function value becomes unbounded from above. First, a strong alternative result from [34] is needed.

Lemma 2.1
The following two systems of inequalities are strong alternatives

1. $Ax \leq b$
2. $\lambda \geq 0,\ A^T\lambda = 0,\ b^T\lambda < 0$

that is, exactly one of the alternatives holds.

Proof: See [34, pp. 261–262]. □

Theorem 2.4
If the primal problem (2.27) *is infeasible, and the dual problem* (2.33) *is feasible, then the dual problem* (2.33) *is unbounded from above.*

Proof: Consider a QP problem of the type in (2.27) with only inequality constraints and define $J_D(\lambda)$ to be the dual objective function. Assume the dual problem feasible. Then $\exists\,\bar{\lambda} : A_{\mathcal{I},2}^T\bar{\lambda} = 0,\ \bar{\lambda} \geq 0$. Further, assume the primal infeasible. Then, from Lemma 2.1, it follows that $\exists\,\lambda' : \lambda' \geq 0,\ A_{\mathcal{I}}^T\lambda' = 0,\ b_{\mathcal{I}}^T\lambda' < 0$. Note that

$$A_{\mathcal{I}}^T\lambda' = \begin{bmatrix} A_{\mathcal{I},1}^T \\ A_{\mathcal{I},2}^T \end{bmatrix} \lambda' = 0 \tag{2.35}$$

Since $A_{\mathcal{I},2}^T\left(\bar{\lambda} + \alpha\lambda'\right) = 0$, the sum $\bar{\lambda} + \alpha\lambda'$ is dual feasible for every $\alpha \geq 0$. It now holds that

$$\begin{aligned} J_D(\bar{\lambda} + \alpha\lambda') &= -\frac{1}{2}\left(\bar{\lambda} + \alpha\lambda'\right)^T A_{\mathcal{I},1}\tilde{H}^{-1}A_{\mathcal{I},1}^T\left(\bar{\lambda} + \alpha\lambda'\right) \\ &- \left(\tilde{f}^T\tilde{H}^{-1}A_{\mathcal{I},1}^T + b_{\mathcal{I}}^T\right)\left(\bar{\lambda} + \alpha\lambda'\right) - \frac{1}{2}\tilde{f}^T\tilde{H}^{-1}\tilde{f} \\ &= -\frac{1}{2}\bar{\lambda}^T A_{\mathcal{I},1}\tilde{H}^{-1}A_{\mathcal{I},1}^T\bar{\lambda} - \left(\tilde{f}^T\tilde{H}^{-1}A_{\mathcal{I},1}^T + b_{\mathcal{I}}^T\right)\bar{\lambda} - \frac{1}{2}\tilde{f}^T\tilde{H}^{-1}\tilde{f} - \alpha b_{\mathcal{I}}^T\lambda' \\ &= J_D(\bar{\lambda}) - \alpha b_{\mathcal{I}}^T\lambda' \to +\infty,\ \alpha \to +\infty \end{aligned} \tag{2.36}$$

since $b_{\mathcal{I}}^T\lambda' < 0$, and where the second equality follows from $A_{\mathcal{I},1}^T\lambda' = 0$. The general case, where equality constraints are included, follows directly from the proof above by expressing an equality constraint as two inequality constraints. □

Case 2 does not need any further investigation since the desired result is immediate.

Two important conclusions can now be drawn. First, if the dual is infeasible, then the primal is infeasible. Second, if the dual is feasible, then the primal is feasible if and only if the dual optimal objective function value is bounded from above.

2.4.2 Active Set Methods

An inequality constrained QP can be solved either using an interior point method or an active set method. In this text the focus will be on an active set method. A well-known example of an active set method for linear programs is the simplex method. As soon will be apparent, the notion "active set" allude to the way the method works. To solve an equality constrained QP is rather straightforward. An active set solver reduces the problem of solving the inequality constrained problem to solving a sequence of equality constrained problems. In this text, a step in this solution sequence will be referred to as a QP iteration. The material presented in this section is based on [77] and [43]. The problem to be solved is of the type in (2.21), with $H \in \mathbb{S}_+^n$. However, in each QP iteration an equality constrained QP with $m \leq n$ number of constraints is considered:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & Ax = b \end{aligned} \tag{2.37}$$

where $A \in \mathbb{R}^{m \times n}$ has full row rank, that is $\operatorname{rank} A = m$. If A does not have full row rank, the constraints are either inconsistent or some constraints are redundant in which case they can be deleted without changing the solution to the problem. Using the equation $Ax = b$, m variables can be eliminated from the problem by expressing them in the other $n - m$ remaining variables. Choose matrices $Y \in \mathbb{R}^{n \times m}$ and $Z \in \mathbb{R}^{n \times (n-m)}$ such that $\begin{bmatrix} Y & Z \end{bmatrix}$ is nonsingular. Further, Z and Y should fulfill $AY = I$ and $AZ = 0$. That is, *one* solution to $Ax = b$ is given by $x = Yb$. Since this solution, in general, is non-unique, an arbitrary solution to $Ax = b$ can be written as

$$x = Yb + Zx_Z \tag{2.38}$$

where $x_Z \in \mathbb{R}^{n-m}$. The linearly independent columns of Z can be interpreted as a basis for the nullspace of A. If (2.38) is inserted into (2.37), the following unconstrained optimization problem is obtained

$$\underset{x_Z}{\text{minimize}} \quad \tfrac{1}{2}x_Z^T Z^T H Z x_Z + (f + HYb)^T Z x_Z + \tfrac{1}{2}b^T Y^T H Y b + f^T Y b \tag{2.39}$$

Note that the last two terms in the objective function are constants and can therefore be omitted. The result is an equivalent optimization problem which can be identified as a QP on the form (2.21) without constraints. The matrix $Z^T H Z$ is considered as the Hessian of the reduced problem and is called the reduced Hessian. Its properties is of importance when solving (2.37).

The vector Yb was chosen to be *one* solution of $Ax = b$, that is, in many cases there is freedom in this choice. This freedom can be used to choose Y such that good numerical properties are obtained.

The KKT conditions for an optimization problem on the form (2.37) can be written as a system of linear equations

$$K \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} -f \\ b \end{bmatrix} \tag{2.40}$$

The following lemma taken from [77] gives sufficient conditions for non-singularity of the KKT matrix K.

Lemma 2.2
Let A have full row rank and assume that the reduced-Hessian matrix $Z^T G Z$ is positive definite. Then the KKT matrix K in (2.40) *is non-singular and there is a unique pair of vectors (x^*, ν^*) satisfying* (2.40)*.*

Proof: See [77, 445]. □

Actually, a more powerful result can be shown. The following theorem is taken from [77].

Theorem 2.5
Suppose that the conditions of Lemma 2.2 are satisfied. Then the vector x^* satisfying (2.40) *is the unique global solution of* (2.37)*.*

Proof: See [77, p. 446]. □

Before inequality constraints are considered, a definition of the active set is necessary.

Definition 2.5 (Active set). The set

$$\mathcal{A}(x) = \mathcal{E} \cup \left\{ i \in \mathcal{I} \mid a_i^T x = b_i \right\} \tag{2.41}$$

where x is any feasible point, is called the active set at x.

The active set in optimum, $\mathcal{A}(x^*)$, is called the optimum active set. An active set solver has a set containing the indices of the constraints that are treated as equality constraints in the current iteration. This set is called the working set and is in iteration k denoted $\mathcal{W}_k$. If $\mathcal{A}(x^*)$ would have been known in advance, the problem could have been solved as an equality constrained problem of the type (2.37), where the constraints are those being indexed by $\mathcal{A}(x^*)$. If an active set solver is supplied with an initial working set $\mathcal{W}_0$, which does not differ much from $\mathcal{A}(x^*)$, the problem can often be quickly solved. This idea is used in so-called warm starts, where information from a previous optimal solution is used to quickly reoptimize after a minor change to the problem. Unfortunately, $\mathcal{A}(x^*)$ is in general not known in advance. Therefore, $\mathcal{W}_0$ has to be initialized in some way. This can be done by making a guess of $\mathcal{A}(x^*)$, or simply by taking $\mathcal{W}_0 = \mathcal{E}$.

As previously mentioned, in a QP solver a sequence of equality constrained problems of the type in (2.37) is solved. Between the solution of each such problem, an inequality constraint is either added to the working set or removed from the working set. After the working set has been changed, a new optimization problem in the sequence is considered. This is done by solving the corresponding KKT system of the type in (2.40). Therefore, it is necessary to solve systems of this type efficiently. For generic QP solvers there exist a

number of different methods for how this can be performed. For the problems considered in this thesis, the KKT system has a special structure, which makes it possible to solve it using Riccati recursions. Thus, the standard methods are not surveyed in this text. Some references to standard methods are, for example, [77] and [43].

It is important that all rows indexed by $\mathcal{W}_k$ are linearly independent, otherwise the constraints are either inconsistent or redundant. If this requirement is fulfilled for $\mathcal{W}_0$, the algorithm to be presented guarantees that it will also be fulfilled for $\mathcal{W}_k$ in all subsequent iterations, [77].

Let x_k be a feasible solution to the constraints indexed by $\mathcal{W}_k$ in iteration k. It is not known whether x_k minimizes the objective function subject to the constraints indexed by $\mathcal{W}_k$ or not. Further, let $\hat{x}_{k+1}$ denote the optimal solution subject to the constraints indexed by $\mathcal{W}_k$. The step necessary to take from x_k to reach $\hat{x}_{k+1}$ is then calculated as $p_k = \hat{x}_{k+1} - x_k$. If $\hat{x}_{k+1}$ is feasible with respect to all constraints in the original problem, x_{k+1} is computed according to $x_{k+1} = \hat{x}_{k+1}$. Otherwise, α_k in $x_{k+1} = x_k + \alpha_k p_k$ has to be chosen as large as possible in the interval $[0, 1]$ under the constraint that x_{k+1} is feasible. Since x_k and $\hat{x}_{k+1}$ satisfy the constraints in $\mathcal{W}_k$, so does x_{k+1} since

$$\begin{aligned} A_{\mathcal{W}_k} x_{k+1} &= A_{\mathcal{W}_k} \left(x_k + \alpha_k \left(\hat{x}_{k+1} - x_k \right) \right) \\ &= A_{\mathcal{W}_k} x_k + \alpha_k \left(A_{\mathcal{W}_k} \hat{x}_{k+1} - A_{\mathcal{W}_k} x_k \right) = b_{\mathcal{W}_k} \end{aligned} \tag{2.42}$$

This follows from the fact that $A_{\mathcal{W}_k} \hat{x}_{k+1} = b_{\mathcal{W}_k}$ and $A_{\mathcal{W}_k} x_k = b_{\mathcal{W}_k}$, independently of α_k. The matrix $A_{\mathcal{W}_k}$ and the vector $b_{\mathcal{W}_k}$ contain the rows corresponding to the constraints in the current working set. Hence, after a step of arbitrary length in the direction $\hat{x}_{k+1} - x_k$, the resulting point is always feasible with respect to $\mathcal{W}_k$. If $\alpha_k < 1$, there is an inequality constraint blocking the way towards the optimum. Consider the inequality constraint with index i. It can be written as $a_i^T \left(x_k + \alpha_k p_k \right) = a_i^T x_k + \alpha_k a_i^T p_k \leq b_i$. If $a_i^T p_k \leq 0$, the constraint remains to be fulfilled for an arbitrary $\alpha_k \geq 0$. On the contrary, if $a_i^T p_k > 0$, α_k has to fulfill

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k} \tag{2.43}$$

Of course, if the optimum has been reached before a constraint blocks the search, α_k is chosen to 1. Summarizing, α_k is chosen as

$$\alpha_k = \min \left\{ 1, \min_{i \notin \mathcal{W}_k,\, a_i^T p_k > 0} \left(\frac{b_i - a_i^T x_k}{a_i^T p_k} \right) \right\} \tag{2.44}$$

where $p_k = \hat{x}_{k+1} - x_k$. The constraints for which the minimum in (2.44) is achieved are called the blocking constraints. Two extremes are when $\alpha_k = 1$ or $\alpha_k = 0$. The first one is already discussed, the second one occurs if there exists an $i \notin \mathcal{W}_k$ such that $a_i^T x_k = b_i$, that is, constraint i is active in x_k, and $a_i^T p_k > 0$. The new working set $\mathcal{W}_{k+1}$ is formed by adding a blocking constraint to the old working set $\mathcal{W}_k$. The procedure is repeated, and new constraints are added until $\hat{x}_{k+1} = x_k$. When this occurs, x_k minimizes the objective function over the working set $\mathcal{W}_k$. The Lagrange multipliers for the equality constrained problem are now computed. A difference between a Lagrange multiplier for an equality constraint and an inequality constraint is that for an inequality constraint, the multiplier must be non-negative. Consequently, if $\hat{\lambda}_i \geq 0, \ \forall\, i \in \mathcal{W}_k \cap \mathcal{I}$, then the

Lagrange multipliers for all inequality constraints treated as equality constraints in the last subproblem, are feasible. As a result, the KKT condition (2.20c) is fulfilled for x_k. Lagrange multipliers for inequality constraints not in the working set, are set to zero. If there exists an index $j \in \mathcal{W}_k \cap \mathcal{I}$ such that $\hat{\lambda}_j < 0$, the KKT condition (2.20c) is not fulfilled and the objective function value can be decreased by dropping constraint j from the working set. This conclusion can be drawn from sensitivity analysis. If there exist negative multipliers, the index corresponding to one of them is removed from the working set and a new subproblem with this constraint removed is solved. In [77], it is shown that this strategy generates a search direction in the next subproblem that is feasible with respect to the dropped inequality constraint. Even though it is possible to drop any of the constraints corresponding to a negative multiplier, the most negative multiplier is often chosen in practice. This choice can be motivated using sensitivity analysis. From this analysis it follows that the decrease in the objective function value when a constraint is dropped is proportional to the multiplier associated with that constraint.

In every QP iteration, the KKT conditions (2.20a) and (2.20b) are fulfilled because the initial point x_0 is feasible and all subsequent α_k are chosen such that primal feasibility is maintained. The complementary slackness condition (2.20d) is fulfilled by the construction of the active set algorithm. In every iteration, $\hat{x}_{k+1}$ fulfills the KKT condition in (2.20e) with all $\hat{\lambda}_i,\ i \notin \mathcal{W}_k \cap \mathcal{I}$ set to zero. If the signs of all multipliers corresponding to inequality constraints in the current working set are non-negative, then also the KKT condition (2.20c) is fulfilled. In this case, all KKT conditions are fulfilled and hence a global optimal solution to the problem has been found. If $H \in \mathbb{S}^n_{++}$, then the unique global optimal solution has been found.

When implementing an active set QP algorithm, it is common to make the variable substitution $p = \hat{x}_{k+1} - x_k$, and formulate the subproblems directly in p. However, in Algorithm 2.1, $\hat{x}_{k+1}$ is explicitly computed instead of p. Apart from this modification, Algorithm 2.1 is similar to the algorithm given in [77]. In this reference, convergence properties of the algorithm are discussed. This discussion also covers cycling of the active set algorithm, which means that a sequence of additions and deletions of constraints to and from the working set is repeated without the algorithm making any progress towards the optimum. If not detected and aborted, this sequence is repeated until the maximum allowed number of iterations is reached. There are procedures to handle cycling, but according to [77], most QP implementations simply ignore the possibility of cycling.

An active set algorithm requires a feasible initial point x_0. One approach is to use a so-called Phase I method, where a linear optimization problem is solved to generate a feasible starting point for the QP. Another approach is to use the big-M method, where the constraint infeasibility is penalized by adding a weighted infinity norm of the amount of infeasibility to the objective function.

2.4.3 Dual Active Set Quadratic Programming Methods

The QP method presented in Section 2.4.2 is a primal feasible active set method. This means that it starts in a primal feasible point. Primal feasibility is thereafter maintained in all subsequent QP iterations. The main drawback with the primal method is that a primal feasible starting point has to be obtained before the actual optimization can start. As described in Section 2.4.2, if a feasible starting point cannot be obtained by, for example,

Algorithm 2.1 Active set QP algorithm for convex QP

Compute a feasible starting point x_0.
Define the maximum number of iterations as k_{max}.
Set $\mathcal{W}_0$ to be a subset of the active constraints at x_0.
$k := 0$
while $k < k_{max}$ **do**
  Given $\mathcal{W}_k$, compute $\hat{x}_{k+1}$.
  if $\hat{x}_{k+1} = x_k$ **then**
    Compute Lagrange multipliers $\hat{\lambda}_i$.
    if $\hat{\lambda}_i \geq 0,\ \forall i \in \mathcal{W}_k \cap \mathcal{I}$ **then**
      $x^* = x_k$
      STOP
    else
      $j := \underset{j \in \mathcal{W}_k \cap \mathcal{I}}{\operatorname{argmin}}\ \hat{\lambda}_j$
      $x_{k+1} := x_k$
      $\mathcal{W}_{k+1} := \mathcal{W}_k \setminus \{j\}$
    end if
  else
    Compute α_k according to (2.44).
    $x_{k+1} := x_k + \alpha_k \left(\hat{x}_{k+1} - x_k\right)$
    if $\alpha_k < 1$ **then**
      Set j to be the index of one of the blocking constraints.
      $\mathcal{W}_{k+1} := \mathcal{W}_k \cup \{j\}$
    else
      $\mathcal{W}_{k+1} := \mathcal{W}_k$
    end if
  end if
  $k := k + 1$
end while
No solution was found in k_{max} iterations.

practical knowledge of the problem, a Phase I algorithm can be applied to find such a point. According to [48], the authors computational experience indicates that on average between one-third to one-half of the total effort needed to solve a QP with "typical primal algorithms" is spent in Phase I. Comparing the primal QP problem in (2.21) and the dual QP problem in (2.25), it is clear that it is easier to find a feasible starting point to the dual problem than to the primal problem. For example, the origin is always a feasible starting point to the dual problem. To find a feasible starting point to the primal problem, the in general more difficult constraints in (2.21) have to be fulfilled. A dual method is particularly suitable for Sequential Quadratic Programming (SQP), where several similar inequality constrained QPs are solved sequently, [78]. According to the same reference, if the suggested initial working set is unsuitable, it can relatively easy be adjusted. In a primal algorithm, if the initial working set is not feasible it might be necessary to start over from an empty working set. Another important advantage of a dual active set method is that the dual inequality constraints cannot be degenerate since the gradients of the non-negativity constraints in (2.25) are linearly independent. For specific methods, this claim is supported by the references [65], [43] and [48]. According to [77], the simple structure of the constraints in the dual problem enables efficient use of the gradient projection method when solving the dual problem. The advantage with a gradient projection method, compared to a classical active set method as the one presented in Algorithm 2.1, is that rapid changes to the working set are allowed. As a consequence, the number of QP iterations can be reduced.

Early work on dual active set methods for QP can be found in [65] and [94]. The method presented in [65] is built on Dorn's dual of the type where the primal variables have been eliminated. When the primal variables are eliminated the result is a dual problem of the form (2.26) with only λ-variables present. The method can be interpreted as an active set method where dual feasibility is maintained during the active set iterations in the search for a dual optimal point. Because of the simple structure of the constraints, the origin is always found to be a feasible initial solution. This is true for all dual methods if the dual looks like (2.25). In (2.33), also the equality constraints have to be fulfilled by the starting point. In [65], a finite solution to the dual problem is required, which can be interpreted, by weak duality, as a requirement for primal feasibility. In [94], a dual method built on the so-called simplex method for quadratic programming by Dantzig and van de Panne is presented. As the name of the algorithm indicates, the method reduces to the ordinary simplex method for linear programming if the Hessian is zero. Readers interested in this early QP method are referred to the first part of the article, which covers this primal method. In the second part, the method is applied to the Dorn dual of a QP. According to [48], the dual method in [94] cannot handle problems where the primal is infeasible. This problem can though be eliminated using the modification proposed in [48].

A more recent dual active set algorithm for strictly convex QPs is presented in [48]. In this method, primal problems like (2.21) are solved in subproblems where only a subset of the inequality constraints are present. In each iteration a violated primal constraint is added to the working set and the corresponding subproblem is solved. If the subproblem is infeasible, the entire optimization problem is found to be infeasible. This can be explained by the fact that in the optimal solution there must not exist any violated constraints. Consequently, if it turns out that it is not possible to clear all constraint violations,

while maintaining the subproblems feasible, the optimization problem is not feasible. If the new subproblem is feasible, the working set is updated and the procedure is continued. A difference with this algorithm compared to [65] and [94] is that the former does not explicitly form the dual problem. The "duality" in the algorithm can be said to stem from the fact that it maintains dual feasibility instead of primal feasibility during the changes to the active set. The algorithm usually starts in the unconstrained primal optimum and is generally able to take advantage of a good initial estimate of the solution, [43]. The algorithm is equivalent to a primal algorithm applied to the dual problem, [43, 48]. In the reference, both the method presented in [65] and the method presented in [94] are compared to the algorithm. According to [48], the algorithm presented in the cited reference is more efficient and more numerically stable than [94]. A drawback with the dual algorithm is also mentioned. If the Hessian is ill-conditioned, numerical problems might occur since the dual algorithm starts from the unconstrained optimum. The numerical properties of the algorithm presented in [48] are further examined in [78], where an extension to handle ill-conditioned problems is presented and the algorithm is compared to two primal QP solvers `QPSOL` and `VE02A`. In [31], the algorithm is extended to the positive semidefinite case. In [7], the QR factorization used in [48] is replaced by the use of a Schur complement (as in [61]) for block elimination of the KKT matrix. This enables the use of solvers for linear equation systems utilizing problem structure. To be able to more easily adapt to a specific application, the code is written in object oriented C++. The routine is called `QPSchur`. As a conclusion, it can be noticed that the method based on [48], seems to have good numerical properties, as well as good performance. It should however be mentioned that the method does not allow for rapid changes in the working set, which is invited by the simple constraint structure in the dual QP problem.

An infeasible active set solver for problems with simple bounds on variables is presented in [63]. It is actually not based on a dual method, but it shares the property of not enforcing primal feasibility during the iterations. Unlike a dual method, it does not enforce dual feasibility either. In the derivation of the method, the Hessian is assumed positive definite. In the article, the results for discretized infinite-dimensional optimal control problems in [27, 28] are generalized to a general QP formulation.

The dual problem to a QP is considered in several books. Some examples are [77], [43] and [10].

2.5 Mixed Integer Quadratic Programming

Mixed Integer Quadratic Programming (MIQP) is a special case of Mixed Integer Non-Linear Programming (MINLP). At a first glance, the MIQP problem looks similar to the ordinary QP problem (2.21). There is however one important difference. The optimization variables are not only allowed to be real valued, but also integer valued. This "slight" modification turns the easily solved QP problem, into an $\mathcal{NP}$-hard problem, [101]. A common special case of MIQP is when the integer variables are constrained to be 0 or 1. To use a precise notation, this problem is called a Mixed Binary Quadratic Programming (MBQP) problem. The standard notation for MBQP seems, at least in the control literature, to be MIQP. In what follows, the problem studied will be an MBQP, but to keep the standard notation, it will be denoted MIQP. A survey considering Quadratic Integer

Programming (QIP) can be found in [98].

2.5.1 Problem Definition

The mathematical definition of an MIQP problem is

$$\begin{aligned} \underset{x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x \\ \text{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}} \\ & A_{\mathcal{I}} x \leq b_{\mathcal{I}} \end{aligned} \tag{2.45}$$

where $f \in \mathbb{R}^{n_c+n_b}$ and $H \in \mathbb{S}_+^{n_c+n_b}$. Further, let $A_{\mathcal{E}}$, $A_{\mathcal{I}}$, $b_{\mathcal{E}}$ and $b_{\mathcal{I}}$ be defined as in (2.21) with $n = n_c + n_b$.

There exist several methods for solving MIQP problems. The four most commonly used methods for these kind of problems are, [16]:

- Cutting plane methods
- Decomposition methods
- Logic-based methods
- Branch and bound methods

Several authors claim that branch and bound is the best method for mixed integer programs, [16]. In [44], a branch and bound method is compared to Generalized Benders Decomposition (GBD), Outer Approximation (OA) and LP/QP based branch and bound. The conclusion in this reference is that branch and bound is the superior method for solving MIQP problems. With a few exceptions, branch and bound is an order of magnitude faster than any of the other methods. An important explanation to why branch and bound is so fast is that the QP subproblems are very cheap to solve. This is not the case for general MINLP, where several QP problems have to be solved in each node in the branch and bound tree. In the MINLP case there exist important problem classes where branch and bound is not the best method. A review of different methods of solving MIQP problems can be found in [98]. There exist several software for solving MIQP problems. For MATLAB, free software like YALMIP or *miqp.m* can be used. A commonly used commercial software is CPLEX.

2.5.2 Branch and Bound

If computational burden is not considered, the most straightforward approach to compute the optimal solution to an optimization problem involving binary variables is to enumerate all possible combinations of the binary variables, and for each such combination, compute the optimal solution of any real variables also included in the problem. Thereafter, the objective function values are compared and the solution, or solutions, generating the best objective function value is taken as the optimal solution. However, for problems involving many binary variables the computational burden will become overwhelming, since the number of combinations of the binary variables is 2^{n_b}.

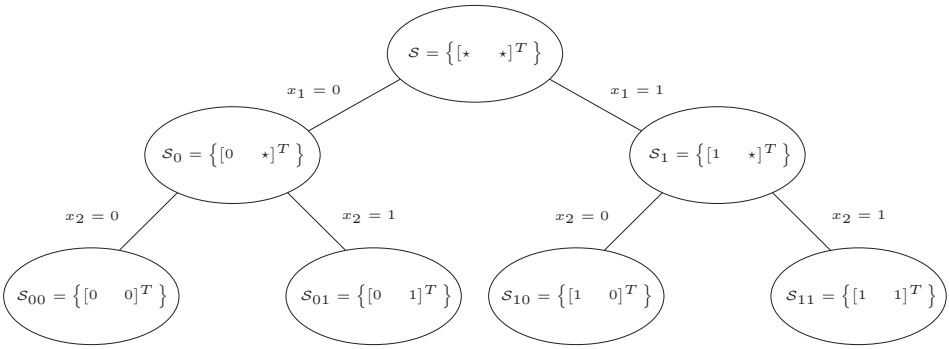


Figure 2.2: *This figure shows an example of a binary search tree for two binary variables, x_1 and x_2. In each node, represented as an ellipse, the corresponding feasible set $\mathcal{S}_i$ is shown. The symbol $\star$ is used to denote that this variable is free to be either 0 or 1.*

The conclusion from this introductory discussion is that there is a need for an algorithm that can find the optimal solution without enumerating all possible combinations of the binary variables. One such algorithm is branch and bound, where it is most often sufficient to explicitly enumerate only *some* of the possible combinations. Unfortunately, the worst case complexity is still exponential and the number of combinations necessary to enumerate, and solve an optimization problem for, is problem dependent. Most of the derivation of and the motivation for the branch and bound algorithm come from [101] and [45].

Denote the feasible set of the optimization problem considered $\mathcal{S}$. In the branch and bound method, $\mathcal{S}$ is split into K smaller sets such that

$$\mathcal{S} = \bigcup_{i=1}^{K} \mathcal{S}_i \tag{2.46}$$

This partitioning is performed in several steps. The partitioning is at first coarse, but is in later steps more and more refined. The partitioning can be represented using a tree structure. An example of a tree is given in Figure 2.2. The tree in Figure 2.2 is a so-called binary search tree, which is a special case of a general search tree and is the type of tree of interest for the MIQP problems considered in this text. The ellipses in the tree are called nodes. The rows of nodes in the tree are called levels. The top node is called the root node. In a binary search tree, all nodes except the nodes in the bottom of the tree have two nodes connected to the lower side of the node. These two nodes are called the children of the node above, and the node above is called the parent node of the two child nodes. Note that the root node does not have a parent node. Similarly, the nodes at the bottom of the tree do not have any children. These nodes are called leaves. One of the features of branch and bound is that the entire tree is not known from the beginning. Only the parts of the tree needed in the solution process are expanded.

The optimal solution over the set $\mathcal{S}$ can be computed by optimizing over the smaller

sets separately according to

$$\begin{aligned} z^{i*} &= \underset{x \in \mathcal{S}_i}{\text{minimize}}\ f_0(x),\ i \in \{1, \ldots, K\} \\ z^* &= \min_{i \in \{1,\ldots,K\}} \left\{ z^{i*} \right\} \end{aligned} \tag{2.47}$$

The optimal solution over $\mathcal{S}$ is found as the optimal solution to the subproblem with the lowest optimal objective function value. Note that the leaves in the tree in Figure 2.2 contain the different combinations of the binary variables that have to be investigated if $\mathcal{S}$ in the example is to be explored by complete enumeration. Hence, it is clear that if it is necessary to solve all of the problems represented by the leaves, there is no gain in using the branch and bound method. The important question to answer is whether it is possible to use the structure of the tree in order to reduce the number of leaves necessary to explore.

To simplify what follows, make the following definitions:

- P_i denotes the optimization subproblem over the set $\mathcal{S}_i$.
- N_i denotes the node containing P_i.
- z^* is the optimal objective function value over $\mathcal{S}$.
- z^{i*} is the optimal objective function value for subproblem P_i.
- $\bar{z}$ denotes a global upper bound of the objective function value. By global it is meant it is valid for the entire tree. It is achieved by the best known feasible solution so far, which is denoted by $\bar{x}$ and is usually called the incumbent.
- $\underline{z}^i$ denotes a local lower bound of the objective function value. By local it is meant that it is valid only for the subtree with root node N_i.

The key idea to reduce the computational effort needed is to compute upper and lower bounds for the optimal objective function value for the subproblems in the nodes. Often, these bounds can be used to prune entire subtrees, which means that these subtrees do not have to be considered any more, since it can be concluded that the optimal solution cannot be found in any of them. Further, these bounds are much easier to compute than to solve the original problem to optimality. Pruning can be interpreted as an implicit enumeration, and is therefore highly desirable. An example of the use of the bounds is shown in Figure 2.3. The original problem is to minimize the objective function over the set $\mathcal{S}$. This problem is split into two subproblems. In one subproblem the binary variable x_1 is fixed to 0 and in the other it is fixed to 1. In Figure 2.3, the upper and the lower bound for a node are indicated as a super- and a subindex respectively for the circle representing the node. The computation of the bounds for problem P over the set $\mathcal{S}$ gives an upper bound of 10 and a lower bound of 1. Problem P is split into two subproblems P_0 and P_1 over the sets $\mathcal{S}_0$ and $\mathcal{S}_1$. The upper bound for P_0 is 5 and the lower bound is 4. Further, the upper bound for P_1 is 7 and the lower bound is 6. Since the best possible objective function value over $\mathcal{S}_1$ does not even reach the worst possible value over $\mathcal{S}_0$, it is no use continuing working with P_1. Therefore, the subtree with N_1 as the root node can be pruned. Another useful case occurs if it is actually possible to solve

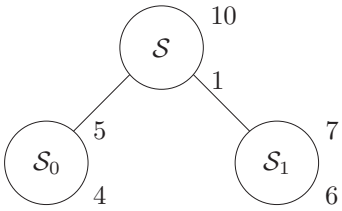


Figure 2.3: *This figure is used to illustrate how bounds can be used to prune nodes. Assume that the tree originates from a minimization problem. Since the upper bound over $\mathcal{S}_0$ is lower than the lower bound over $\mathcal{S}_1$, $\mathcal{S}_1$ cannot contain the optimal solution.*

a subproblem to optimality rather than just to compute bounds. In that case, a feasible solution to the original problem P has been found and the optimal objective function value for the subproblem is an upper bound for z^*. Further, the subtree containing this node can be pruned, since the objective function value cannot be improved by a reduction of the feasible set. Another reason for pruning is if the set $\mathcal{S}_i$ is empty.

Summarizing, there exist at least three different possibilities for pruning a subtree with root node N_i.

1. Infeasibility: $\mathcal{S}_i = \emptyset$.
2. Optimality: An optimal solution to the subproblem is found.
3. Dominance: $\underline{z}^i \geq \bar{z}$.

Note that if case 2 occurs, and if $z^{i*} < \bar{z}$, the global upper bound $\bar{z}$ should be updated. It also important to note that if $\underline{z}^i < \bar{z}$, and $\underline{x}^i$ is not feasible in $\mathcal{S}_i$, the node cannot be pruned.

To be able to apply the above scheme in practice, it has to be decided how to compute the upper and lower bounds. Usually, upper bounds are found from integer feasible solutions and lower bounds are found from relaxations or duality. In MIQP, relaxations can be created by relaxing the integer constraints to interval constraints. That is, if the binary variable indexed by j is relaxed, the constraint

$$x_j \in \{0, 1\} \tag{2.48}$$

is replaced by

$$x_j \in [0, 1] \tag{2.49}$$

In an MIQP solver built on branch and bound, ordinary constrained QP problems are solved in the nodes. As the method makes progress down in the tree, fixed integer variables are eliminated from the problem. This means that the number of optimization variables in the relaxed subproblems decreases by one for each level passed on the way down in the tree. Note that when a variable has been fixed to either 0 or 1 after a branch on that variable, this constraint is not relaxed in the nodes further down in the tree.

It is important to understand what properties of a problem that can be found from a relaxation of the problem. First, if the relaxed problem is infeasible, then the original

problem is infeasible. This can be used for pruning according to 1 above. Second, the optimal objective function value of a relaxation is lower than the optimal objective function value of the original problem. Third, if an optimal solution of the relaxed problem is feasible in the original problem, then this solution is also an optimal solution to the original problem. This can be used to prune according to 2 above. In the MIQP case, if an optimal solution of a relaxation of problem P_i satisfies the binary constraints for subproblem P_i, an optimal solution to the unrelaxed problem P_i has been found. Since this solution also is feasible in the optimization problem over $\mathcal{S}$, an upper bound for z^* has been found. In this thesis, the relaxation of P_i is denoted by P_i^R, and the relaxed solution by $\underline{x}^i$. The relaxation of the set $\mathcal{S}_i$ is denoted $\mathcal{S}_i^R$.

In a branch and bound method, there are several parameters and choices that may affect the performance drastically. Two important parameters are the choice of the next node to solve and the choice of the branch variable. The three most common criteria for node selection are

- Depth first
- Breadth first
- Best first

In depth first, the next node to solve is chosen as one of the child nodes of the current node. This process is continued until a node is pruned. After a node is pruned the so-called backtracking starts. Backtracking is the procedure of going back, towards the root node, in the search for a node with an unconsidered child node. One advantage with this strategy is that the search goes down quickly in the tree, which is good because integer feasible solutions to the relaxed problems are more likely to appear deep down in the tree. Another advantage is that similar problems are solved subsequently, making it easy to perform warm starts of the QP solver. A disadvantage with this strategy is that it is likely that many nodes have to be considered before optimality can be proven. Depth first with backtracking is the default setting in most commercial codes.

In breadth first, all nodes at each level have to be considered before a node in a new level can be considered. It is used as a basis for node selection heuristics and for certain estimates.

In best first, the next problem to consider is chosen as the one with the lowest lower bound so far. The advantage with this node selection criterion is that the number of subproblems to solve is minimized.

Choosing which node selection criterion to use is not straightforward. Usually, empirical studies have to be performed in order to choose the best criterion for a specific application. It is also common to use a mix of depth first and best first in order to prove optimality as well as to find better feasible solutions.

The next important parameter is how to select the next variable to branch. A common choice is to let the user provide a set of priorities. In that case, each integer variable is assigned a relative importance. When the system is about to branch, it chooses the integer variable with the highest assigned priority among the integer variables with fractional optimal relaxed values. Other approaches are to branch on the variable with the lowest index, or the integer variable with the largest or smallest fractional part in the solution of the relaxed problem, [15].

It is also common to be able to specify which of the branches to explore first. In the binary case, the choice is between the branch where the variable is set to 0 and the branch where it is set to 1.

According to [44], solving the subproblems using a dual active set method offers the most straightforward way to exploit the structure introduced by the branching procedure. After a branch, the solution to the parent problem is in general infeasible in the child problems. But, a dual feasible starting point for the child problems is directly available from the dual solution of the parent problem. Consequently, it is possible to warm start the active set solver using information from the solution to the parent problem. Warm starts are further discussed in Section 3.3.2. Also, since a dual active set method is an ascend method generating dual feasible points, it can use an upper bound as a cut-off value for terminating the QP solver prematurely, [44].

According to [101], active set methods (the reference considers the linear programming case) is preferable for solving the relaxed problems in branch and bound. For very large problems, Interior Point (IP) algorithms can be used to solve the first subproblem, but in the subsequent subproblems an active set method should be used.

An important step in a commercial branch and bound code is the preprocessing step. In the preprocessing step the formulation is checked to be "sensible" and as strong as possible given the available information, [101]. A strong formulation is a formulation that gives a tight lower bound on the optimal objective function value. The basic operations in preprocessing is to quickly detect and eliminate redundant constraints and variables, and to tighten bounds if it is possible. A smaller and tighter formulation is preferred, since the number of nodes necessary to consider, and the dimension of the subproblems, might be reduced.

This section is concluded with a formal algorithm for branch and bound for binary variables. This is found in Algorithm 2.2. How subproblems are put on the list and retrieved from the list is decided by the choice of the node selection criterion and the branching priority. If it, in some way, is possible to easily find an upper bound on the optimal objective function value, this bound can be used to initialize the global upper bound $\bar{z}$.

2.6 Binary Quadratic Programming

Binary Quadratic Programming (BQP) can be considered a special case of MIQP, where only binary variables are present. In this thesis, two forms of BQP problems are considered. A property shared by both forms is that no constraints are present. The first problem formulation has a pure quadratic objective

$$\underset{x\in\{0,1\}^{n_b}}{\text{minimize}} \quad x^T H x \tag{2.50}$$

where $H \in \mathbb{S}^{n_b}$. The second form is a generalization of (2.50)

$$\underset{x\in\{0,1\}^{n_b}}{\text{minimize}} \quad \tfrac{1}{2}x^T H x + f^T x \tag{2.51}$$

where a linear term has been incorporated in the objective function.

Algorithm 2.2 Branch and bound for binary variables

$\bar{z} := +\infty$
$\bar{x} :=$ **void**
Add P to $LIST$.
while $\text{length}(LIST) > 0$ **do**
  Pop P_i from $LIST$.
  Solve $P_i^R \Rightarrow \underline{z}^i$ and $\underline{x}^i$.
  if $\mathcal{S}_i^R = \emptyset$ **then**
    No feasible solution exists for P_i.
  else if $\underline{z}^i \geq \bar{z}$ **then**
    There exists no feasible solution of P_i which is better than $\bar{x}$.
  else if $\underline{x}^i \in \mathcal{S}_i$ **then**
    $\underline{x}^i$ is integer feasible and is therefore optimal also in P_i.
    $\bar{z} := \underline{z}^i$
    $\bar{x} := \underline{x}^i$
  else
    Split $\mathcal{S}_i$ into $\mathcal{S}_{i0}$ and $\mathcal{S}_{i1}$.
    Push P_{i0} and P_{i1} to $LIST$.
  end if
end while

The BQP problem is known to be $\mathcal{NP}$-hard, [62]. Most algorithms for this kind of problems either focus on producing approximative solutions or on only handling various special cases of the general problem, [46]. Some approximative heuristic algorithms can be found in, for example, [62], [11], [72] and [47].

After a reformulation, several combinatorial optimization problems such as the maximum cut problem, the maximum clique problem, the maximum vertex packing problem and the maximum independent set problem can all be written as BQP problems, [72].

3

Mixed Integer Predictive Control

In this chapter, Model Predictive Control (MPC) and hybrid systems on the Mixed Logical Dynamical (MLD) form are introduced. In Section 3.1, basic MPC is presented and the problem is formulated as an optimization problem. In Section 3.2, MLD systems are introduced and their range of application is discussed. Further, control of MLD systems is considered. Optimization in MPC is discussed in Section 3.3. The chapter is concluded with Section 3.4, where two examples of systems on MLD form are presented. These examples will be used as benchmark problems throughout the thesis.

3.1 Model Predictive Control

Model Predictive Control (MPC) has been used in a broad spectrum of applications for a long time. It is hard to say exactly when MPC was invented, but probably the first patent was granted to Martin-Sanchez in 1976, [70]. An early academic publication containing the basic ideas was presented by Propoi 1963, [70]. There are also some methods similar to MPC, but with different names. One of the most well-known is Dynamic Matrix Control (DMC), [35].

The most commonly used variant of MPC is so-called linear MPC, where the dynamics is linear and a quadratic objective similar to the one used in Linear Quadratic (LQ) control is used. A difference compared to LQ is that it is also possible to consider linear constraints on the states and control signals. With the word linear in front of MPC, it is emphasized that a linear model of the controlled system is used. A discrete-time linear time-invariant model on state space form is given by

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.1}$$

where $t \in \mathbb{Z}$ is the discrete time, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the control input and

$y(t) \in \mathbb{R}^p$ is the controlled output. The objective, or performance measure, to minimize is a quadratic function like

$$\begin{aligned} J(t_0) = \sum_{s=0}^{N-1} \left(\|y(t_0+s) - r(t_0+s)\|_{Q_e}^2 + \|u(t_0+s)\|_{Q_u}^2 \right) \\ + \|y(t_0+N) - r(t_0+N)\|_{Q_e}^2 \end{aligned} \tag{3.2}$$

where $Q_e \in \mathbb{S}_{++}^p$ and $Q_u \in \mathbb{S}_{++}^m$, and $r(t) \in \mathbb{R}^p$ is the reference signal. Often, the constraints are defined as

$$H_u(t)u(t) + H_x(t)x(t) + h(t) \leq 0 \tag{3.3}$$

A common variant of the constraint formulation (3.3) is to allow constraints involving states and control signals from different time steps. This modification enables the use of, for example, rate limits on states or control signals. However, this can also be enabled in the formulation in (3.3) by augmenting the state vector with states and control signals from previous time instants. In this thesis, the special case of MPC when there are no inequality constraints like (3.3) is called the unconstrained MPC problem.

In MPC, the future behavior of the system is predicted N time steps ahead. In this context, prediction means that a system model like (3.1) is used to calculate how the system will react to control inputs and thereby what will happen in the future if a certain control input is applied to the system. Not surprisingly, N is called the prediction horizon, which in practice is chosen long enough to cover a normal transient of the controlled system.

There are several different ways to cast (3.1), (3.2) and (3.3) on the form of a formal optimization problem. The most common variants are presented and evaluated in [67]. If the system is linear and the objective is quadratic, the resulting optimization problem is a QP (see Section 2.4), for which there exist well developed optimization routines. Hence, for linear MPC the optimization problem is considered easy to solve. In this thesis two formulations are used where the difference lies in the representation of the dynamics. In the first formulation in (3.4), the dynamics is represented as equality constraints

$$\begin{aligned} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T\end{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & \mathsf{Q}_u & 0\\ 0 & 0 & \mathsf{Q}_e\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix}\mathsf{A} & \mathsf{B} & 0\\ \mathsf{C} & 0 & -I\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} = \begin{bmatrix}b\\ \mathsf{r}\end{bmatrix} \\ & \begin{bmatrix}\mathsf{H}_x & \mathsf{H}_u & 0\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} \leq -\mathsf{h} \end{aligned} \tag{3.4}$$

and in the second formulation in (3.5), the states x and control error e have been eliminated using the equality constraints.

$$\begin{aligned} \underset{\mathsf{u}}{\text{minimize}} \quad & \frac{1}{2}\mathsf{u}^T\left(\mathsf{S}_u^T\mathsf{C}^T\mathsf{Q}_e\mathsf{C}\mathsf{S}_u + \mathsf{Q}_u\right)\mathsf{u} + \left(\mathsf{S}_u^T\mathsf{C}^T\mathsf{Q}_e\left(\mathsf{C}\mathsf{S}_x x_0 - R\right)\right)^T\mathsf{u} \\ \text{subject to} \quad & \left(\mathsf{H}_x\mathsf{S}_u + \mathsf{H}_u\right)\mathsf{u} \leq -\mathsf{h} - \mathsf{H}_x\mathsf{S}_x x_0 \end{aligned} \tag{3.5}$$

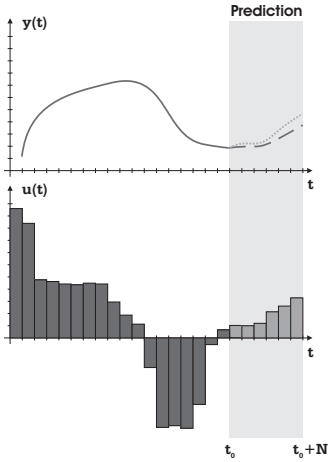


Figure 3.1: *In this figure, an example of a control input $u(t)$ and the corresponding controlled output $y(t)$ is given. The controller is about to compute the control signal in time t_0. The figure illustrates how the behavior of the system is predicted N steps. In the predicted interval, the dotted predicted output can be compared with the dashed actual output. As a consequence of unknown disturbances and modeling errors, these curves do not completely coincide.*

Algorithm 3.1 Basic MPC controller

1: Measure or estimate the state of the controlled process x_0 in time instant t_0.
2: Obtain u by minimizing (3.2) with respect to u subject to the constraints (3.1), (3.3) and the initial constraint $x(t_0) = x_0$.
3: Apply the first element $u(t_0)$ in u to the controlled process.
4: Set $t_0 := t_0 + 1$ and repeat the procedure.

The notation and the derivations of the formulations can be found in Appendix B. The two optimization problems are equivalent, but from a computational view (3.4) gives a sparse optimization problem with $(N + 1)(n + p) + Nm$ optimization variables, while (3.5) gives a dense optimization problem with Nm variables.

In order to get closed-loop control, the approach above is used in a receding horizon fashion, which means that the prediction interval is moved one step forward after each completed optimization. After the optimization has been performed, only the first control signal in the optimal control signal sequence computed is applied to the system and the others are ignored. In the next time step, a new optimization is performed and the procedure is repeated. Due to modeling errors and unknown disturbances, the predicted behavior and the actual behavior of the system do not usually completely coincide. Such errors are, if they are sufficiently small, handled by the feedback in the algorithm. The procedure is visualized in Figure 3.1 and the conceptual steps are summarized in Algorithm 3.1. In this thesis, t_0 in (3.2) is often assumed zero.

An already explored extension to linear MPC is non-linear MPC. This extension handles non-linear systems and a general non-linear norm in the objective function. Unfortu-

nately, the resulting optimization problem is more difficult to solve in general.

A special case of non-linear MPC is to handle systems described partly by logics. These are called hybrid systems and provides a unified framework for describing processes evolving according to continuous dynamics, discrete dynamics and logic rules, [20]. This class of systems is especially important when analyzing and controlling systems arising in the growing interaction between physical processes and digital controllers.

A survey covering both linear and non-linear MPC is found in [71]. A reference book covering most of MPC is [70].

3.2 Mixed Logical Dynamical Systems

Mixed Logical Dynamical (MLD) systems is one way of describing an important class of hybrid systems defined by linear dynamic equations subject to linear mixed integer inequalities, that is, inequalities involving both continuous and binary variables. Binary variables are sometimes also denoted logical or 0-1 variables. The MLD description is a very general model class capable of describing a broad spectrum of systems. In this thesis, only discrete-time systems are considered.

3.2.1 Background

The initial interest in hybrid systems has been concentrated to the field of verification and safety analysis, for which many results and techniques are now available, [22]. In [16], an MPC framework used for systems described by physical laws, logic rules and operating constraints is presented. An important part of this framework consists of the definition of MLD systems. This class of systems includes linear hybrid systems, finite state machines, some classes of discrete event systems, constrained linear systems and non-linear systems which can be exactly or approximately described by piecewise linear functions.

Although the MLD description is quite new, there are several applications for MLD systems reported in the literature. For example, in [16] it is described how the by-products from a steel-works are used to produce electric power. In order to produce the electricity, the by-products are burnt in furnaces. Not all furnaces can burn all by-products, so the MPC controller has to choose which furnaces to use to be able to use as much of the by-products as possible and limit the use of non-by-products in form of heavy oil. In [41] and [42] a power plant is modeled as an MLD system and controlled by an MPC controller. In this application, the flaps and gates are controlled by sending discrete commands to stepper motors. The dynamics of the different subsystems varies with the logical state of the model and there is also a desire to use the actuators with a certain priority. Another example motivating the use of the MLD description in this application is that some elements cannot be opened or closed for an arbitrarily short time. In [5], an Adaptive Cruise Control (ACC) problem for heavy vehicles is studied. In the reference, an MPC controller is used to control the distance to the vehicle in front of the ACC equipped vehicle. The main difficulty in the problem is to prohibit simultaneous use of throttle and brakes. This condition can easily be formulated by introducing a mixed integer linear inequality, which is a linear inequality involving real and binary variables. Other examples of systems requiring a hybrid model are systems with binary control signals as valves and

hatches.

3.2.2 The MLD System Description

An MLD system can be described by the following linear relations, [16],

$$\begin{aligned} x(t+1) &= A(t)x(t) + B_u(t)u(t) + B_\delta(t)\delta(t) + B_z(t)z(t) \\ y(t) &= C(t)x(t) + D_u(t)u(t) + D_\delta(t)\delta(t) + D_z(t)z(t) \\ H_u(t)u(t) &+ H_x(t)x(t) + h(t) \leq H_\delta(t)\delta(t) + H_z(t)z(t) \end{aligned} \tag{3.6}$$

where $t \in \mathbb{Z}$ and

$$x(t) = \begin{bmatrix} x_c(t) \\ x_b(t) \end{bmatrix},\ x_c(t) \in \mathbb{R}^{n_c},\ x_b(t) \in \{0,1\}^{n_b},\ n = n_c + n_b \tag{3.7}$$

denotes the state of the system, partitioned into continuous states $x_c(t)$ and logical (binary) states $x_b(t)$. The controlled output is

$$y(t) = \begin{bmatrix} y_c(t) \\ y_b(t) \end{bmatrix},\ y_c(t) \in \mathbb{R}^{p_c},\ y_b(t) \in \{0,1\}^{p_b},\ p = p_c + p_b \tag{3.8}$$

The control input is also partitioned similarly

$$u(t) = \begin{bmatrix} u_c(t) \\ u_b(t) \end{bmatrix},\ u_c(t) \in \mathbb{R}^{m_c},\ u_b(t) \in \{0,1\}^{m_b},\ m = m_c + m_b \tag{3.9}$$

where $u_c(t)$ denotes the continuous inputs and $u_b(t)$ the logical inputs. Finally, $\delta(t) \in \{0,1\}^{r_b}$ and $z(t) \in \mathbb{R}^{r_c}$ represent auxiliary logical and continuous variables respectively. In order to be able to use a notation as uniform as possible throughout the thesis, the notation in (3.6) has been slightly modified compared to the one used in [16]. If the desired finite alphabet is not binary as here, it can always be coded using binary variables.

MLD systems is just *one* way of modeling hybrid systems, [13]. In [21], the formal equivalence between MLD systems and Piecewise Affine (PWA) systems is established. In [58, 59], the equivalence between the following five classes of hybrid systems is, under certain conditions, established: MLD systems, Linear Complementarity (LC) systems, Extended Linear Complementarity (ELC) systems, PWA systems and Max-Min-Plus-Scaling (MMPS) systems. The equivalence result between MMPS systems and PWA systems is refined in [36]. The important result of these equivalences is that derived theoretical properties and tools can easily be transferred from one class to another, [24]. Each of these subclasses has its advantages. For optimal control and state estimation, the MLD description is proposed, while most other hybrid techniques are built on a PWA representation, [13]. Also, simulation of hybrid systems can be performed much more easily in PWA form compared to in MLD and LC form. Even though different theoretically equivalent forms exist, it is not necessarily an easy task to convert from one form to another. For example, transforming from PWA to MLD is easy but the other way around can have high computational complexity, [13]. The note [13] presents an efficient conversion algorithm from MLD to PWA form. In [16], it is shown how different types of systems can be rewritten as explicit MLD systems.

One way of modeling a system on MLD form is to by hand derive a model on the form (3.6). In that case, it might be necessary to convert a logic description to an MLD description. How this is performed is discussed in [73]. An alternative approach is to create the MLD formulation automatically by using the high-level modeling language HYSDEL (Hybrid Systems Description Language), [92].

3.2.3 Controlling MLD Systems

In [16], both optimal control and receding horizon estimation for MLD systems is discussed. The control signal is found by minimizing a quadratic performance criterion of the form

$$\begin{aligned} J_{\text{MLD}} = \sum_{s=0}^{N-1} &\|u(s) - u_f(s)\|_{Q_u}^2 + \|\delta(s) - \delta_f(s)\|_{Q_\delta}^2 + \|z(s) - z_f(s)\|_{Q_z}^2 \\ &+ \|x(s) - x_f(s)\|_{Q_x}^2 + \|y(s) - y_f(s)\|_{Q_y}^2 \end{aligned} \tag{3.10}$$

subject to $x(0) = x_0$, $x(N) = x_f(N)$ and dynamics (3.6), where $Q_u \in \mathbb{S}_{++}^m$, $Q_\delta \in \mathbb{S}_+^{r_b}$, $Q_z \in \mathbb{S}_+^{r_c}$, $Q_x \in \mathbb{S}_{++}^n$ and $Q_y \in \mathbb{S}_+^p$. The variables in (3.10) with subscript f denote reference signals. This MPC problem can be rewritten as an optimization problem as described for linear MPC in Section 3.1. Consequently, there is a choice between a sparse formulation similar to the one in (3.4) or a dense formulation similar to the one in (3.5). Independently of the formulation chosen, the optimization problem can be solved as a Mixed Integer Quadratic Programming (MIQP) problem, [16].

As in linear MPC, the algorithm is implemented in a receding horizon fashion. The difference is that it is much more complicated to find the optimal control signal sequence, since the system is neither linear nor smooth, [16]. One way of reducing the computational complexity is to use tailored MIQP solvers. This is further discussed in Section 3.3. The control law found is in the literature sometimes referred to as a Mixed Integer Predictive Control (MIPC) law, [16].

Interesting work is presented in [79], where the geometric structure of the solution to MPC problems with finite input constraint sets is studied. A finite input constraint set means that the control signal may be chosen only from a finite alphabet and not continuously as usually. An important special case is when the control signal is constrained to be binary. The main idea in the approach is to treat the problem as a norm minimization problem. By making a variable substitution in order to get the “right coordinates”, the norm becomes the Euclidean norm and the minimization can be performed by vector quantization of the unconstrained solution. It is pointed out in [80] that the solution obtained when directly quantizing the unconstrained solution is different from the one obtained when the variable substitution first is performed.

3.2.4 Moving Horizon Estimation for MLD Systems

The MLD structure is useful not only for control. The model structure can also be used for estimation of states and faults in hybrid systems, [18]. A difference compared to the control problem is that the horizon extends backwards in time, that is, at time t_0 the

interesting quantities are estimated at times prior to t_0. The computational complexity is also here a great problem.

The problem of system identification of hybrid systems is addressed in [23]. Often, also these problems end up in either a Mixed Integer Linear Programming (MILP) problem or an MIQP problem.

3.3 Optimization in Model Predictive Control

Since the MPC algorithm is executed on-line, it is of great relevance to be able to quickly solve the optimization problem in step 2 in Algorithm 3.1. It is the amount of time consumed in step 2 that limits the use of MPC. As the optimization routines get more efficient, implementations at faster sampling rates, on slower hardware and for larger systems is possible. For linear MPC, solvers with high performance exist today. To be able to increase performance, the structure of the optimization problem can be used, [82]. Because of the increased complexity, this is even more important for MIPC. The extension of the QP problem that has to be solved is an MIQP problem. It can, for example, be solved using a branch and bound algorithm where QP relaxations are solved in the nodes. Branch and bound is further described in Section 2.5.2.

Tailored solvers for MPC are not only interesting for MPC applications. They can also be used for state estimation, fault detection and verification, see for example [17].

3.3.1 Quadratic Programming

In many applications, even linear MPC is considered computationally expensive. At each time step, either a QP of the form (3.4) or (3.5) has to be solved. One way of speeding up the solution of the optimization problem is to use QP solvers tailored for MPC, where the special structure of the KKT system is used to decrease the complexity of the algorithm. Basically two approaches can be used. First, general methods utilizing the structure in block-banded equation systems can be used. Second, Riccati recursions can be used. In [61], an active set method utilizing Riccati recursions for solving parts of the KKT system is used. The method is first derived for linear MPC problems, and then extended to non-linear problems. A similar method is presented in [2]. The special structure of the problem has also been used in Interior Point (IP) methods. In [102], an IP solver utilizing block-bandedness is presented. The approach is refined in [103], where the feasible IP method has been replaced by an infeasible IP method. In the reference, also an active set method utilizing the block-banded structure is presented. No actual performance comparison between the two methods is presented, but active set methods are considered to be more suitable for warm starts. This is further discussed in Section 3.3.2. In [55], an infeasible primal-dual IP method utilizing Riccati recursions for the computation of the search directions has been applied to robust MPC. In [30] Riccati recursions have been used in an interior point solver for stochastic programming. SQP methods using active set and IP solvers utilizing problem structure are presented in [86]. Another reference on the same topic is [29]. One way of establishing stability for MPC is to introduce an ellipsoidal terminal state constraint [105]. The resulting optimization problem is a variant of a QP, namely a Quadratically Constrained Quadratic Program (QCQP). After

rewriting the QCQP into a Second Order Cone Program (SOCP), it is shown in [105] how Riccati recursion can be used to decrease the complexity of the calculations of the search directions in the IP algorithm. In [95, 96], the Riccati recursion is thoroughly derived and it is applied to IP LP solvers for solving MPC problems with linear objective function. A summary of different optimization formulations of MPC, and some optimization routines suitable for optimization problems originating from MPC, can be found in [99].

A thorough comparison between four QP solvers for MPC control of the cross directional control in a paper machine is performed in [9]. The methods compared are one primal IP method, one primal active set method and two dual active set methods. The methods found to perform best are the primal active set algorithm `QPOPT` and the dual active set algorithm `QPSchur`. `QPSchur` is the method proposed in the paper and it turns out to be the overall winner. In this method the states are eliminated and the banded structure of the reduced Hessian matrix is utilized. This property is a result of the highly structured process considered. According to the authors, Riccati methods as presented in [82] are not suitable for this process, since the dimensionality of the state vector is high. It can also be noticed that the prediction horizon is very short (three steps or less). Since the bandedness of the reduced Hessian matrix is reduced as the prediction horizon grows larger than one, the usefulness of a solver optimized for banded matrices decreases. A thorough description of `QPSchur` can be found in [7].

Another way of reducing the on-line computational effort is to precalculate the control law. Briefly, the procedure can be explained as given the state of the system, the controller gain is retrieved from a table. The control law has been computed off-line by solving a multi-parametric programming problem, where multi-parametric means that the problem depends on a vector of parameters [91]. This type of MPC is often referred to as explicit MPC. A drawback with explicit MPC is that the complexity of the state space partition often increases rapidly with the number of states [53]. Therefore, several approaches have been developed in order to reduce the complexity. Some references are [53, 54, 90, 91]. Apart from being a tool for reducing the on-line computational effort required, the solution from explicit MPC can give insight into the behavior of the controller in different regions of the state space [25], for example, regions where saturation occurs can be detected. A summary of the theory behind explicit MPC for quadratic objective is found in [26]. The counterpart for problems with linear objective is found in [25].

In [32], an efficient algorithm based on a combination of dynamic programming and multi-parametric quadratic programming for the off-line calculations of the explicit MPC control law is described.

3.3.2 Active Set versus Interior Point

In comparison with active set methods, IP methods are said to be preferable for problems with large values of N, [102]. This statement is motivated by claiming that the number of active set iterations is proportional to the number of constraints, which in turn are proportional to N. Each QP iteration involves the solution of a narrow-banded linear system with complexity $\mathcal{O}(N)$. The total complexity is therefore expected to be $\mathcal{O}(N^2)$, [102]. The reference then comments that an alternative solution is to use a so-called gradient projection algorithm, but rejects this alternative because of the complex nature of the constraints involving both states and control signals. According to [77], the purpose with gradient

projection algorithms is to accelerate the solution process by allowing rapid changes to the active set, but they are most efficient when there are only bound constraints on the variables. That is, MPC problems involving only upper and lower bounds on the control input is expected to be possible to solve efficiently by a gradient projection algorithm. The computational complexity for the IP algorithm presented in [102] is between $\mathcal{O}(N)$ and $\mathcal{O}(N^{\frac{3}{2}})$. The main motivation for using IP algorithms for large problems is that a fixed price is being paid for the number of active constraints, [8]. On the other hand, the active set algorithm is a combinatorial algorithm which in the worst case has a complexity higher than polynomial, [50]. When the number of active constraints is small, the active set algorithm is expected to perform better than an IP algorithm, while when the number of active constraints is large, the IP algorithm is expected to perform better than the active set algorithm. Other references also proposing IP algorithms for large-scale MPC problems are [1, 50]. In these references they are used in combination with a Sequential Quadratic Programming (SQP) solver for non-linear MPC.

Even though the complexity for a standard implementation of an active set solver is higher than for a corresponding IP solver, there is at least one important advantage with an active set algorithm. Often in MPC, several similar optimization problems are to be solved. It is then possible to use information from the solution of a previous problem in order to be able to quickly find the solution to a slightly modified problem. This procedure is called warm start (or hot start). According to [103], active set methods gain more from warm starts than IP methods. According to [8], warms starts still present many open questions for IP methods. The motivation for the effectiveness of the active set method when using warm starts is that if the optimal active set is almost known, often only a few active set iterations are required to reach the optimal active set, [103]. Based on this fact, it seems natural to choose an active set approach when several similar optimization problems are to be solved consecutively. Unfortunately, the last solution is not always feasible in the new problem. In [103], this is pointed out as a drawback with the presented primal active set solver. In this reference, an infeasible IP algorithm is also considered. The latter algorithm handles infeasible starting points without problems. A similar idea is used in [74, 76], where an infeasible active set algorithm is presented. This algorithm can activate and deactivate entire blocks of constraints during one active set iteration. This idea is similar to the gradient projection algorithm. Further, it focuses on removing infeasibilities occurring early in the predicted interval. The authors claim that this will give better suboptimal solutions if the algorithm is prematurely aborted, [75]. Since the algorithm works with an infeasible primal, it is possible to use the unconstrained solution as an initial solution to the optimization problem, that is, finding a primal feasible solution is no problem.

A conclusion from this discussion is that the best algorithm when solving several similar optimization problems is an active set algorithm, which can handle primal infeasible starting points.

3.3.3 Mixed Integer Quadratic Programming

When ordinary linear MPC is extended to MIPC, a Mixed Integer Programming (MIP) problem has to be solved. Often, these are classified as $\mathcal{NP}$-hard. This means that, in worst case, the solution time grows exponentially with the number of integer variables.

Integer programming problems can be solved by "brute force", meaning that all possible solutions are enumerated and the best possible solutions found are finally presented as the optimal ones. Note that since the problem is non-convex, there might exist more than one optimal solution.

In [16], a commercial Fortran package has been used as an MIQP solver. The package is capable of coping with both dense and sparse MIQP problems. The control problems considered normally lead to sparse optimization problems, which means that a solver utilizing sparsity is preferable.

In [22], a branch and bound strategy based on reachability analysis is presented. Compared to an ordinary branch and bound MIQP solver, the algorithm is, according the authors, neither a depth first nor a breadth first method, but rather a best first method. In the performance test presented in the article, the derived algorithm needs to solve half as many QPs as if an ordinary branch and bound MIQP solver had been used.

One approach to a branch and bound algorithm that aims at quickly pruning entire subtrees is described in [17]. The algorithm presented is tailored for optimal control or estimation problems for MLD systems. The main motivation for the algorithm is the observation that for many systems the binary variables seldom change over the prediction horizon. For example, binary variables can be associated with conditions on the continuous states, that is $[\delta(t) = 1] \leftrightarrow [x(t) \geq 0]$. Also, if the binary variables represent the existence of irreparable faults in the system, they can at most change once during the prediction horizon. Based on this knowledge, the main idea with the algorithm is to first solve the subproblems where the binary variables switch few times. In the article, the tailored method is compared to the standard tree exploring strategies breadth first and depth first. When the proposed algorithm is used on a test problem, the number of QPs solved in the subproblems is approximately reduced by a factor 4. When a solver is used for a real-time implementation, the time available for retrieving a solution is limited. If the branch and bound algorithm has not terminated in time, it is desirable to get an acceptable suboptimal solution. The test made in [17] shows that the outside first tree exploring strategy produces, for MPC applications, better suboptimal solutions as compared to the other two standard strategies if the limitation is the number of QPs solved.

An alternative to the ordinary branch and bound algorithm is presented in [14]. The algorithm is built on a combination of Constraint Programming (CP) and MIP. By letting the CP solver deal with the logic part of the problem, it is no longer necessary to reformulate the logic part into mixed integer inequalities and the structure in the logic part can therefore be kept.

The idea of keeping the structure of the logic part of the hybrid system is also the key to the algorithm proposed in [60]. The motivation is when an automaton is converted into mixed integer inequality constraints, the relaxed problems in the branch and bound algorithm become unnecessary loose. To get a tighter relaxation, new equality constraints are introduced.

In [12], so-called temporal Lagrangian decomposition has been used to split the hybrid MPC problem in time, into several smaller subproblems. The separation in time is performed by Lagrangian relaxation of the dynamic constraints connecting the state before and after a split point. When this approach is used, the primal subproblems can be solved independently for a fixed value of the Lagrange multipliers associated with the interconnection constraints. Unfortunately, in practice the algorithm has to iterate between

solving the primal subproblems and the Lagrange multipliers connecting the subproblems. Even though not presented in the paper, the iterative procedure is expected to be computationally expensive. Some more general references on decomposition techniques in optimization can be found in, for example, [87] and [64].

Another attempt to reduce the complexity of the MIQP problem is found in [89], where the original MLD model is split into several smaller submodels, each valid in a certain region of the state space where some or all binary variables are remaining constant. The result is an MIQP problem with fewer binary variables to compute.

In [20] explicit MPC is extended to MLD systems. The performance criterion in the reference is not the 2-norm, but the 1-norm and the ∞-norm. The optimization problem that has to be solved off-line is in this case a multi-parametric MILP (mp-MILP). A similar paper is [19]. A thorough reference on the subject multi-parametric MIQP is [40], where theory and algorithms for mp-QP, mp-LP and mp-MIQP are presented.

According to [33], a drawback with multi-parametric mixed integer programming is that the solver does not exploit the structure of the optimal control problem. In the reference, a more efficient algorithm based on solving the discrete-time Hamilton-Jacobi-Bellman equation is proposed. This equation is solved using a multi-parametric quadratic programming solver.

A survey of constrained optimal control in general and specifically the explicit solution is found in [52].

3.4 Two Examples of Mixed Logical Dynamical Systems

In this section two simple examples of MLD systems are given. These systems are used throughout the thesis as benchmark examples for the algorithms to be presented in later sections.

3.4.1 Mass Position Control

In this example, a mass is controlled in one dimension by two separate forces. One force, u_c, is possible to control continuously and the other, u_b, is applied binary with a certain magnitude and direction. The states x_1 and x_2 are the velocity of the mass and the position of the mass, respectively. The continuous-time state space description is given by

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & -5 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_c \\ u_b \end{bmatrix} \\ y &= \begin{bmatrix} 0 & 1 \end{bmatrix} x \end{aligned} \tag{3.11}$$

To obtain a discrete-time system on the form (3.6), zero order hold sampling can be used.

3.4.2 Satellite Attitude Control

In this example, a satellite control problem is presented. The satellite controls its attitude by accelerating a reaction wheel inside the satellite and by using external thrusters. When

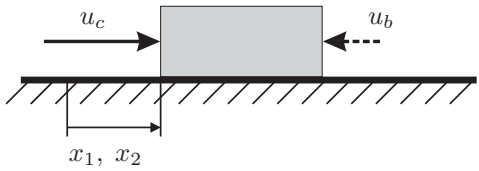


Figure 3.2: *This figure illustrates the mass modeled in* (3.11)*, where* x_1 *denotes the velocity of the mass,* x_2 *the position of the mass,* u_c *a continuously controlled force and* u_b *a binary controlled force.*

the wheel is accelerated a counter torque is produced. If several adjustments in the same direction are made, the angular velocity of the wheel finally becomes very high. To be able to slow down the wheel without affecting the attitude of the satellite, the external thrusters have to be used to compensate when the wheel is braked.

The wheel is assumed to be controlled continuously by an electric engine. Its control signal is denoted u_c. The satellite is also assumed to be equipped with two external thrusters, one in each direction. These are assumed to be controlled binary, that is, either they give full thrust or no thrust at all. The binary control signals for the thrusters are denoted $u_{b,1}$ and $u_{b,2}$.

A continuous-time state space description for the system with satellite attitude x_1, satellite angular velocity x_2 and internal wheel velocity x_3 is

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 2.5 & 1 & -1 \\ -10 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_c \\ u_{b,1} \\ u_{b,2} \end{bmatrix} \\ y &= I_3 x \end{aligned} \tag{3.12}$$

To obtain a discrete-time system on the form (3.6), zero order hold sampling can be used.

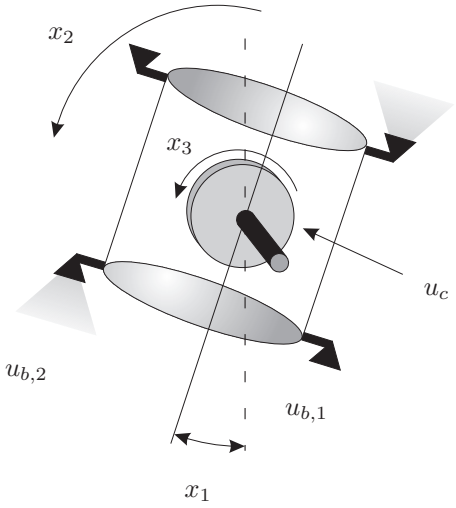


Figure 3.3: *This figure illustrates the satellite modeled in* (3.12)*, where* x_1 *is the satellite attitude,* x_2 *is the satellite angular velocity and* x_3 *the angular velocity of the reaction wheel. The control signals are* u_c*,* $u_{b,1}$ *and* $u_{b,2}$*, which control the electric engine and the two oppositely directed thrusters respectively.*

4

Multiuser Detection in a Code Division Multiple Access System

Already in early telegraph systems, it was possible for two users to share a common channel. In these systems the channel was an ordinary wire. The information from one user was coded by changes in the polarity and the information from the other user was coded by changing the absolute values. This is an example of an early multi-access communication system, where several users share a common channel. Today, there are numerous examples of such communication systems. Two common examples are mobile phones transmitting to a base station and local area networks.

In this chapter, mobile phone networks are considered. When using a radio channel, several users may coexist by assigning different frequencies to each one of them. This multi-access technique is called Frequency Division Multiple Access (FDMA). In common GSM networks, a multi-access technique called Time Division Multiple Access (TDMA) is used. In TDMA, each user is assigned a time-slot in which it is allowed to transmit. Both these approaches have in common that no more than one user may occupy a given time-frequency slot. In the third generation (3G) mobile communication systems, a multi-access method called Code Division Multiple Access (CDMA) is used. In CDMA, the users are assigned different signature sequences. These sequences are used to separate the information sent by a specific user from the information sent by other users and it can be compared with a specific frequency in FDMA and a specific time-slot in TDMA. An important difference is that in CDMA, the signature sequences overlap both in time and in frequency. Two advantages of CDMA compared to TDMA and FDMA is that it is more spectrum efficient and it allows more easily for dynamical bandwidth allocation.

4.1 Multiuser Detection

Multiuser Detection (MUD) is the process of demodulating multiple users sharing a common multi-access channel. A first approach is to demodulate each user independently and

to treat the signal from other users as additive Gaussian noise, [88]. An improvement to this strategy is to use the known correlation between users in the demodulation process. Better performance can be achieved if the detector makes the most likely decision, which formally is achieved by solving a Maximum Likelihood (ML) problem. When the optimum multiuser detection problem is cast on the form of an ML problem, it requires the solution of a Binary Quadratic Programming (BQP) problem. Unfortunately, these problems are generally known to be $\mathcal{NP}$-hard (see Section 2.6). If the signature sequences produce a cross-correlation matrix with some special structures, the problem can however sometimes turn out to have lower complexity, [83, 85, 93].

Many contributions to the area of multiuser detection have already been published. The objective is to find an algorithm which solves the multiuser detection problem in reasonable time in order to make a real-time implementation possible. So far, this has been done either by restricting the class of possible cross-correlation matrices or by employing a sub-optimal procedure. In [93], an algorithm with polynomial complexity has been derived for systems with only negative cross-correlations. A similar requirement on the cross-correlation matrix is found in [83], where the multiuser detection problem is solved with a polynomial complexity algorithm if the cross-correlation between the users are non-positive. Another paper also dealing with a special class of cross-correlations is [85], where a polynomial complexity algorithm is derived for the case of identical, or a few different, cross-correlations between the users. Thorough work in the field of approximate algorithms for multiuser detection is found in [88]. Several different algorithms, optimal as well as sub-optimal, are presented and evaluated in [56]. The sub-optimal algorithm local search is evaluated in [57]. Branch and bound methods are investigated in [69]. Another near optimal approach is presented in [66]. Also the well-known Kalman filter has been applied to the problem. This approach is presented in [68].

4.2 Synchronous Code Division Multiple Access

In this section, a synchronous CDMA model is presented. It is also shown how the multiuser detection problem can be formulated as a BQP problem.

4.2.1 System Model

Consider a CDMA channel simultaneously used by K users. The symbol length is assumed to be T seconds. Each user is assigned a certain signature sequence, a so-called chip sequence. The chip sequence is a sequence consisting of N chips, each taking a value from $\{-1, +1\}$. The constant N is known as the spreading factor, spreading gain or processing gain, [97].

The notation used in this thesis is chosen similar to the one used in [97]. The channel model used is the so-called K-user channel which consists of the sum of K antipodally modulated synchronous signature waveforms embedded in additive white Gaussian noise

$$y(t) = \sum_{k=1}^{K} A_k b_k s_k(t) + \sigma n(t),\ t \in [0, T] \tag{4.1}$$

where

- $y(t) \in \mathbb{R}$ is the received signal.
- $s_k(t) \in \mathbb{R}$ is the deterministic signature waveform assigned to user k, normalized to have unit energy, that is,

$$\int_0^T s_k^2(t)\,dt = 1 \tag{4.2}$$

  Because the waveforms are assumed to be zero outside the interval $[0, T]$, there is no inter-symbol interference.
- $A_k \in \mathbb{R}$ is the received amplitude of the signal from user k, and therefore, A_k^2 is referred to as the energy of user k.
- $b_k \in \{-1, +1\}$ is the data bit transmitted by user k.
- $n(t) \in \mathcal{N}(0, 1)$ with $\mathsf{cov}\,(n(t), n(\tau)) = \delta(n - \tau)$ is the Gaussian noise added to the channel.

The similarity of different signature waveforms is expressed in terms of the cross-correlation defined by

$$\rho_{ij} = \int_0^T s_i(t) s_j(t)\,dt \tag{4.3}$$

At the receiver, the signal $y(t)$ in (4.1) is received. After the reception, the procedure of separating the information sent by different users begins. In that procedure, low cross-correlation between the different signature sequences is useful. The separation procedure, called despreading, is performed by matched filters according to

$$\begin{aligned} y_1 &= \int_0^T y(t) s_1(t)\,dt \\ &\vdots \\ y_K &= \int_0^T y(t) s_K(t)\,dt \end{aligned} \tag{4.4}$$

Using (4.1), (4.2) and (4.3), output y_k in (4.4) can be written as

$$y_k = A_k b_k + \sum_{j \neq k} A_j b_j \rho_{jk} + n_k \tag{4.5}$$

where

$$n_k = \sigma \int_0^T n(t) s_k(t)\,dt \in \mathcal{N}(0, \sigma^2) \tag{4.6}$$

Using vector notation, this can be written more compactly as

$$y = RAb + n \tag{4.7}$$

where R is the normalized cross-correlation matrix

$$R = \int_0^T \begin{bmatrix} s_1(1) \\ \vdots \\ s_K(t) \end{bmatrix} \begin{bmatrix} s_1(1) \\ \vdots \\ s_K(t) \end{bmatrix}^T dt \tag{4.8}$$

whose diagonal elements are equal to one and is symmetric non-negative definite, and where

$$\begin{aligned} y &= [y_1, \ldots, y_K]^T \\ b &= [b_1, \ldots, b_K]^T \\ A &= \operatorname{diag}(A_1, \ldots, A_K) \end{aligned} \tag{4.9}$$

If the signature sequences are orthogonal, then $\rho_{ij} = 0$, whenever $i \neq j$. The non-orthogonal sequences usually give low cross-correlation even though the users might not be synchronized. Common choices of such sequences are Gold sequences and Kasami sequences, [97]. Furthermore, the unnormalized cross-correlation matrix is denoted as

$$H = ARA \tag{4.10}$$

Because only the synchronous case is treated in this thesis, no inter-symbol interference will occur. Hence, it is only necessary to consider one time instant and therefore time index t on y, b and n is suppressed.

4.2.2 Derivation of the BQP Problem

The matched filter output is described by equation (4.7). According to [97], the bits most likely sent by the users are given by the solution b to the ML problem

$$\underset{b}{\text{maximize}} \quad \exp\left(-\tfrac{1}{2\sigma^2} \int_0^T \left(y(t) - \textstyle\sum_{k=1}^K b_k A_k s_k(t)\right)^2 dt\right) \tag{4.11}$$

Alternatively, it is equivalent to maximize

$$\Omega(b) = 2\int_0^T \left[\sum_{k=1}^K A_k b_k s_k(t)\right] y(t)\, dt - \int_0^T \left[\sum_{k=1}^K A_k b_k s_k(t)\right]^2 dt = 2b^T Ay - b^T Hb \tag{4.12}$$

where A, H, b and y are defined in (4.9) and (4.10). By altering the sign of the objective and dividing it by two, the optimization problem can be rewritten as an equivalent minimization problem

$$\underset{b \in \{-1,+1\}^K}{\text{minimize}} \quad \tfrac{1}{2} b^T Hb - y^T A^T b \tag{4.13}$$

After a variable substitution, this problem can be identified as a BQP problem on the form (2.51).

5

A Preprocessing Algorithm for Mixed Integer Quadratic Programming

In this chapter, a preprocessing algorithm applicable to BQP problems is presented. The algorithm is derived in Section 5.1. In two steps, the algorithm is extended to handle an unconstrained special case of an MIQP problem, where neither equality constraints nor inequality constraints are present.

In Section 5.2, the algorithm is used for preprocessing of MIQP problems originating from unconstrained MPC problems involving binary control signals. Furthermore, in Section 5.3, the algorithm's applicability to BQP problems is used in a detector for Multiuser Detection (MUD). When the algorithm is applied to the MUD problem, it not only works as a preprocessing algorithm, but it also shows some important properties of the solution to the problem.

5.1 A Preprocessing Algorithm for BQP and MIQP Problems

In this section, a polynomial complexity preprocessing algorithm for BQP problems and unconstrained MIQP problems is derived. A preprocessing algorithm is an algorithm that processes the optimization problem in the step prior to the one in which the actual solver is applied. Because the algorithm to be presented in this section executes in polynomial time and the BQP solver, generally, executes in exponential time, the required CPU time can be reduced if the optimal value of variables can be computed already in the preprocessing step.

Most algorithms for solving BQP problems either focus on producing approximative solutions or only on handling various special cases of the general problem. The algorithm presented here belongs to the latter type of algorithms. For references to BQP algorithms, see Section 2.6.

Except for problems with particular structures, the MIQP problem is known to have

exponential worst case complexity, [17]. One way of reducing the average complexity is to use an algorithm that in most cases can find the global optimum without enumerating all possible solutions. A popular algorithm to use when solving MIQP problems is the branch and bound algorithm. For a thorough discussion about branch and bound, see Section 2.5.2. In this chapter, it is investigated if the performance of the branch and bound algorithm can be increased by using preprocessing. Only MIQP problems without constraints are considered. Previous work in the area of preprocessing for the MIQP problem is found in, for example, [84].

The derivation of the main result is performed in three steps. In the first part of Section 5.1.2, a BQP problem of the form (2.50) is considered. In the second part, this result is extended to a BQP problem of the form (2.51). After the final extension, which is presented in Section 5.1.3, the algorithm can be applied to problems of the form (5.3). After this extension, the preprocessing algorithm can be used for unconstrained MPC problems involving both binary and real-valued control signals.

5.1.1 The BQP and MIQP Problems

For convenience, the two BQP problem formulations (2.50) and (2.51) from Section 2.6 are repeated below, followed by an unconstrained special case of (2.45).

$$\underset{x\in\{0,1\}^{n_b}}{\text{minimize}} \quad x^T H x \tag{5.1}$$

$$\underset{x\in\{0,1\}^{n_b}}{\text{minimize}} \quad \tfrac{1}{2}x^T H x + f^T x \tag{5.2}$$

where $H \in \mathbb{S}^{n_b}$. Finally, an unconstrained version of (2.45),

$$\underset{x\in\mathbb{R}^{n_c}\times\{0,1\}^{n_b}}{\text{minimize}} \quad \tfrac{1}{2}x^T H x + f^T x \tag{5.3}$$

where $H \in \mathbb{S}^{n_c+n_b}$ and the optimization vector x contains n_c real and n_b binary variables. For future reference, define

$$H = H_d + H^+ + H^- \tag{5.4}$$

where

$$\begin{aligned} H_{d,ij} &= \begin{cases} H_{ij}, & i = j \\ 0, & i \neq j \end{cases} \\ H^+_{ij} &= \max\left(0, H_{ij} - H_{d,ij}\right) \\ H^-_{ij} &= \min\left(0, H_{ij} - H_{d,ij}\right) \end{aligned} \tag{5.5}$$

5.1.2 Preprocessing for the BQP Problem

To begin with, the case when all elements in x are binary is considered. The problem is then of BQP type, which is a purely combinatorial problem. As mentioned in Section 5.1, a characteristic property of many combinatorial problems is that they in general are very hard to solve exactly, because of the computational complexity, [72]. To be able to solve large problem instances, either an approximate algorithm or some special structure in the

problem has to be exploited. If the structure of the problem is used it might be possible to solve the problem exactly in reasonable time.

The algorithm presented in this section makes it possible to speed up the solution of a certain class of BQP problems. For this class of problems the algorithm produces an exact solution for one or more variables in polynomial time. For each binary variable the algorithm delivers one out of three possible results: 1 is the optimal value, 0 is the optimal value or nothing can be said for sure.

The preprocessing algorithm is based on the following result:

Theorem 5.1
For a BQP problem of type (5.1)*, an optimal value of one or more components x_i can be found in polynomial time if for some $i \in \{1, \ldots, n_b\}$ any of the following conditions is satisfied*

$$\begin{aligned} (i): &\qquad H_{ii} \geq -2\textstyle\sum_{j=1}^{n_b} H_{ij}^{-} \\ (ii): &\qquad H_{ii} \leq -2\textstyle\sum_{j=1}^{n_b} H_{ij}^{+} \end{aligned}$$

If any of the conditions (i) or (ii) is satisfied for a certain value of i, an optimal value of x_i is given by

$$x_i = \begin{cases} 0, & \text{if } (i) \text{ holds} \\ 1, & \text{if } (ii) \text{ holds} \end{cases}$$

Proof: Consider optimization problem (5.1). Denote the objective function by $Q(x)$ and rewrite it as follows

$$Q(x) = x^T H x = \sum_{i=1}^{n_b} \sum_{j=1}^{n_b} H_{ij} x_i x_j \tag{5.6}$$

where $x_i, x_j \in \{0, 1\},\ \forall\, i, j = \{1, \ldots, n_b\}$. For each $i \in \{1, \ldots, n_b\}$ the objective function $Q(x)$ can be written as

$$\begin{aligned} Q(x) &= H_{ii} x_i x_i + 2 x_i \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j + g_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_b}) \\ &= (H_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j) x_i + g_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_b}) \end{aligned} \tag{5.7}$$

where g_i is a function that is independent of x_i and where the last equality follows from the fact that $x_i^2 = x_i$ when $x_i \in \{0, 1\}$. Define

$$h_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_b}) = H_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j \tag{5.8}$$

Note that h_i is independent of x_i. With this definition, the objective function can be written as

$$Q(x) = h_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_b}) x_i + g_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_b}) \tag{5.9}$$

Denote an arbitrary optimal solution to (5.1) by $x^* = \left[x_1^*, \ldots, x_{n_b}^*\right]^T$. Note that since the feasible set $\{x \mid x_i = \{0,1\},\ i = 1, \ldots, n_b\}$ is non-convex, there is not necessarily a unique optimum to (5.1). For convenience, introduce

$$\begin{aligned} h_i^* &= h_i(x_1^*, x_2^*, \ldots, x_{i-1}^*, x_{i+1}^*, \ldots, x_{n_b}^*) \\ g_i^* &= g_i(x_1^*, x_2^*, \ldots, x_{i-1}^*, x_{i+1}^*, \ldots, x_{n_b}^*) \end{aligned} \tag{5.10}$$

Note that if the optimal solution x_i^* is non-unique, there is one g_i^* and one h_i^* associated with each optimal solution. It now follows that

$$\min_x Q(x) = \min_{x_i} h_i^* x_i + g_i^* = \begin{cases} g_i^*, & \text{if } h_i^* \geq 0 \\ h_i^* + g_i^*, & \text{if } h_i^* < 0 \end{cases} \tag{5.11}$$

From (5.11) the conclusion can be drawn that

$$x_i^* = \begin{cases} 0, & \text{if } h_i^* > 0 \\ 0 \text{ or } 1, & \text{if } h_i^* = 0 \\ 1, & \text{if } h_i^* < 0 \end{cases} \tag{5.12}$$

Unfortunately, h_i^* is usually not known before the optimal solution $\left[x_1^*, x_2^*, \ldots, x_{i-1}^*, x_{i+1}^*, \ldots, x_{n_b}^*\right]^T$ is known. A solution to this problem is to try to make an estimate of h_i^*. To simplify the notation, define

$$\begin{cases} \bar{h}_i = \max\limits_x h_i \\ \underline{h}_i = \min\limits_x h_i \end{cases} \tag{5.13}$$

It now holds that $\underline{h}_i \leq h_i^* \leq \bar{h}_i$, for all h_i^* corresponding to, possibly different, optimal solutions. From this observation the following implications can be stated

$$\begin{cases} \bar{h}_i < 0 & \Rightarrow h_i^* < 0 \\ \bar{h}_i = 0 & \Rightarrow h_i^* \leq 0 \\ \underline{h}_i = 0 & \Rightarrow h_i^* \geq 0 \\ \underline{h}_i > 0 & \Rightarrow h_i^* > 0 \end{cases} \tag{5.14}$$

Note that, if h_i^* has the same sign for all optimal solutions, the component x_i^* has a unique optimal solution. By combining (5.12) and (5.14), the following conclusion can be drawn about the optimal value of x_i

$$x_i^* = \begin{cases} 0, & \underline{h}_i > 0 \quad (x_i^* \text{ unique}) \\ 0, & \underline{h}_i = 0 \quad (x_i^* \text{ not necessarily unique}) \\ 0 \text{ or } 1, & \underline{h}_i = \bar{h}_i = 0 \quad (x_i^* \text{ not unique}) \\ 1, & \bar{h}_i = 0 \quad (x_i^* \text{ not necessarily unique}) \\ 1, & \bar{h}_i < 0 \quad (x_i^* \text{ unique}) \end{cases} \tag{5.15}$$

Note that if $\underline{h}_i = \bar{h}_i = 0$, then the coefficient in front of x_i is zero. Thus, this case is not of practical interest. If the uniqueness properties of the solution is not of interest, it is possible to reduce the five cases in (5.15) to two cases

$$x_i^* = \begin{cases} 0, & \underline{h}_i \geq 0 \quad (x_i^* \text{ not necessarily unique}) \\ 1, & \bar{h}_i \leq 0 \quad (x_i^* \text{ not necessarily unique}) \end{cases} \tag{5.16}$$

From (5.5) and (5.8) it follows that

$$\begin{aligned} \bar{h}_i = \max_x h_i = \max_x \left(H_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j \right) \\ = H_{ii} + 2 \max_x \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j = H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^+ \end{aligned} \tag{5.17}$$

where the last equality follows from the fact that the sign of H_{ij} determines whether the maximizing x_j is 0 or 1. Analogously it follows that

$$\underline{h}_i = \min_x h_i = H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^- \tag{5.18}$$

Finally, Equation (5.16) can be written on the desired form

$$x_i^* = \begin{cases} 0, \ H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^- \geq 0 \Leftrightarrow H_{ii} \geq -2 \sum_{j=1}^{n_b} H_{ij}^- & (i) \\ 1, \ H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^+ \leq 0 \Leftrightarrow H_{ii} \leq -2 \sum_{j=1}^{n_b} H_{ij}^+ & (ii) \end{cases} \tag{5.19}$$

From (5.19) it is clear that the computational complexity of the tests (i) and (ii) is polynomial in the number of variables, that is in n_b. □

Now the result is extended to problems of type (5.2).

Corollary 5.1
For a BQP problem of type (5.2)*, an optimal value of one or more components x_i can be found in polynomial time if for some $i \in \{1, \ldots, n_b\}$ any of the following conditions is satisfied*

$$\begin{aligned} (i): \quad & H_{ii} \geq -2f_i - 2 \textstyle\sum_{j=1}^{n_b} H_{ij}^- \\ (ii): \quad & H_{ii} \leq -2f_i - 2 \textstyle\sum_{j=1}^{n_b} H_{ij}^+ \end{aligned}$$

If any of the conditions (i) or (ii) is satisfied for a certain value of i, an optimal value of x_i is given by

$$x_i = \begin{cases} 0, & \textit{if } (i) \textit{ holds} \\ 1, & \textit{if } (ii) \textit{ holds} \end{cases}$$

Proof: The result follows directly from Theorem 5.1 by observing that

$$Q(x) = \frac{1}{2} x^T H x + f^T x = x^T \left(\frac{H + 2\,\mathrm{diag}(f)}{2} \right) x \tag{5.20}$$

for $x_i \in \{0, 1\}$. □

5.1.3 Preprocessing for the MIQP Problem

If the x-vector is allowed to contain both real and binary variables, the BQP problem becomes an MIQP problem. In this section, the problem is assumed to be of the type (5.3) and the x-vector is assumed to be of the form

$$\begin{aligned} x &= \begin{bmatrix} x_c \\ x_b \end{bmatrix} \\ x_c &\in \mathbb{R}^{n_c},\ x_b \in \{0,1\}^{n_b} \end{aligned} \tag{5.21}$$

The objective function can be expressed as

$$Q(x) = \frac{1}{2}x^T H x + f^T x = \frac{1}{2}\begin{bmatrix} x_c^T & x_b^T \end{bmatrix}\begin{bmatrix} H_{cc} & H_{cb} \\ H_{cb}^T & H_{bb} \end{bmatrix}\begin{bmatrix} x_c \\ x_b \end{bmatrix} + \begin{bmatrix} f_c^T & f_b^T \end{bmatrix}\begin{bmatrix} x_c \\ x_b \end{bmatrix} \tag{5.22}$$

Assume H_{cc} positive definite. If all components in x had been real, it would have been straightforward to use first order necessary and sufficient conditions for optimality from Theorem 2.3 to calculate an explicit optimal solution to the problem. When some of the components in x are binary this is no longer possible. However, the optimality conditions mentioned can still be used to compute the optimal values of the real variables as a function of the binary variables. The expression for the optimal real variables is then given by

$$x_c = -H_{cc}^{-1}\left(H_{cb}x_b + f_c\right) \tag{5.23}$$

Substitute this expression into (5.22). The resulting optimization problem is a pure BQP problem. The objective function, disregarding constant terms, can be written as

$$\frac{1}{2}x_b^T \tilde{H} x_b + \tilde{f}^T x_b \triangleq \frac{1}{2}x_b^T\left(H_{bb} - H_{cb}^T H_{cc}^{-1} H_{cb}\right)x_b + \left(f_b - H_{cb}^T H_{cc}^{-1} f_c\right)^T x_b \tag{5.24}$$

In the calculations the symmetry of H and H^{-1} has been used. When the objective function is written on the form (5.24), Corollary 5.1 can be applied.

5.1.4 Implementation

Theorem 5.1 and Corollary 5.1 can be used in a straightforward way to implement a preprocessing algorithm for BQP problems. In its simplest version, the conditions (i) and (ii) are tested for each $i \in \{1, \ldots, n_b\}$. When a condition is satisfied for a certain value of i, an optimal value of the corresponding component x_i has been found. Practical experience shows that if the preprocessing algorithm is implemented in this straightforward fashion it becomes rather conservative, and as a result, often only few variables are possible to compute by preprocessing.

To increase the number of variables possible to compute in the preprocessing step, the algorithm can be enhanced in two steps. Both steps are built upon the idea to tighten the bounds $\underline{h}_i$ and $\bar{h}_i$. In the first step, any optimal values found for x_i are used in the subsequent computations for elements remaining to be computed. The algorithm, including this first improvement, is presented in Algorithm 5.1. In the second step, after the conditions have been tested once for each optimization variable, and if there are more variables

Algorithm 5.1 BQP Preprocessing

$\tilde{x} := -\mathbf{1}_{n_b}$ // Variables not computed are assigned value -1.
$\tilde{H} := H$
$\tilde{f} := f$
for $i = 1$ to n_b **do**
  $s^- := \tilde{H}(i,i) + 2\tilde{f}(i) + 2\sum_{j=1}^{n_b} \tilde{H}^-(i,j)$
  $s^+ := \tilde{H}(i,i) + 2\tilde{f}(i) + 2\sum_{j=1}^{n_b} \tilde{H}^+(i,j)$
  if $s^+ \leq 0$ **then**
    $\tilde{x}(i) := 1$
    $\tilde{f} := \tilde{f} + \tilde{H}(:,i)$
    $\tilde{H}(:,i) = 0$
  else if $s^- \geq 0$ **then**
    $\tilde{x}(i) := 0$
    $\tilde{H}(:,i) := 0$
  end if
end for

remaining to be computed, a BQP problem of lower dimension is constructed by incorporating the knowledge gained from previous computations. This procedure is repeated for smaller and smaller BQP problems until either no new variables can be computed during an iteration or until the optimal value for all variables in the problem have been found. This implementation of the algorithm is referred to as the "iterated implementation" and is presented as Algorithm 5.2. Both improvement steps result in sharper tests, since optimal values replace the worst case estimates previously used.

After the preprocessing algorithm has terminated, an ordinary MIQP solver, or BQP solver, may be applied to compute any remaining variables. Depending on the time available, the method used in the second step can either produce optimal solutions or suboptimal solutions.

5.2 Application of the Preprocessing Algorithm to Model Predictive Control

In this section, Algorithm 5.2 is applied to the unconstrained MPC problem when binary control signals are present. To be able to apply the preprocessing algorithm to this MPC problem, it has to be formulated as an optimization problem on the form (5.3). This is discussed in Section 5.2.1. Results from simulations are presented in Section 5.2.2.

5.2.1 Using Preprocessing

The results from Section 5.1.3 is in this section applied to an unconstrained MPC problem. In this problem, it is desirable to find the control signal sequence that minimizes a certain

Algorithm 5.2 BQP Iterative Preprocessing

$\tilde{x} := -\mathbf{1}_{n_b}$ `// Variables not computed are assigned value -1.`
$\mathcal{I}_{us} := \{i \mid i \in \mathbb{Z}, 1 \leq i \leq n_b\}$
$H_{tmp} := H$
$f_{tmp} := f$
while $\mathcal{I}_{us} \neq \emptyset$ **do**
  Compute $\tilde{x}(\mathcal{I}_{us})$ by using Algorithm 5.1 with $H = H_{tmp}$ and $f = f_{tmp}$
  $n_s := |\{i \mid i \in \mathcal{I}_{us}, \tilde{x}(i) \neq -1\}|$
  $\mathcal{I}_{us} := \{i \mid \tilde{x}(i) = -1\}$
  if $(n_s > 0)$ and $(\mathcal{I}_{us} \neq \emptyset)$ **then**
    $\mathcal{I}_s := \{i \mid \tilde{x}(i) \neq -1\}$
    $H_{tmp} := H(\mathcal{I}_{us}, \mathcal{I}_{us})$
    $f_{tmp} := H^T(\mathcal{I}_s, \mathcal{I}_{us})\tilde{x}(\mathcal{I}_s) + f(\mathcal{I}_{us})$
  else
    STOP
  end if
end while

criterion for a system on a form similar to (3.6) but with some matrices equal to zero

$$\begin{aligned} x(t+1) &= Ax(t) + B_u u(t) \\ y(t) &= Cx(t) \end{aligned} \tag{5.25}$$

where

$$B_u = \begin{bmatrix} B_{u_c} & B_{u_b} \end{bmatrix}, \; u(t) = \begin{bmatrix} u_c^T(t) & u_b^T(t) \end{bmatrix}^T \tag{5.26}$$

Here B_u is split into one part for continuous control signals and one for binary control signals. Furthermore, $u_c(t) \in \mathbb{R}^{m_c}$ denotes real-valued control signals and $u_b(t) \in \{0,1\}^{m_b}$ denotes binary control signals. The signal $y(t) \in \mathbb{R}^p$ denotes the controlled output.

The objective function to minimize is of the type (3.10), but a terminal state weight has been included. The optimization problem can be written on the form (3.5), by applying a similar procedure as in Appendix B. By ignoring constants and dividing the objective function by two, it can be written on the form (5.22) with

$$\begin{aligned} x_c &= \mathsf{u}_c \\ x_b &= \mathsf{u}_b \\ H_{cc} &= S_{u_c}^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_{u_c} + \mathsf{Q}_{u_c} \\ H_{cb} &= S_{u_c}^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_{u_b} \\ H_{bb} &= S_{u_b}^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_{u_b} + \mathsf{Q}_{u_b} \\ f_c &= S_{u_c}^T \mathsf{C}^T \mathsf{Q}_e \left(\mathsf{C} S_x x_0 - \mathsf{r}\right) \\ f_b &= S_{u_b}^T \mathsf{C}^T \mathsf{Q}_e \left(\mathsf{C} S_x x_0 - \mathsf{r}\right) \end{aligned} \tag{5.27}$$

where u_c, u_b, S_x, S_{u_c}, S_{u_b}, C, Q_e, Q_{u_c} and Q_{u_b} are defined in analogy with Appendix B and x_0 is the measured or estimated state of the system. The optimization problem can then of course also be expressed as a BQP problem on the form (5.24).

If the control signals are ordered according to their appearance in time, it can be seen that the H-matrix gets a structure where the magnitude of the matrix elements descent by the distance to the diagonal. How fast the magnitude of the matrix elements descent seems to be dependent of the position of the poles of the controlled system. For example, stable real poles give a non-oscillating fade off while complex poles give an oscillating fade off. Unstable real poles also give elements that fade out. This behavior is subject to further investigation. This structure of the H-matrix in the MPC problem makes it easier to satisfy condition (i) or (ii) in Corollary 5.1.

5.2.2 Simulation Results

In this section, Algorithm 5.2 is used as an optimization preprocessing algorithm in a mixed integer predictive controller applied to the examples presented in Section 3.4. In both examples, the problem has been solved using three different approaches and the corresponding computational times are presented in a table. In Approach I, an ordinary MIQP solver has been used. In Approach II, real variables have been eliminated, as described in Section 5.1.3, before an ordinary MIQP solver has been used. No other preprocessing has been performed. By doing this reformulation of the problem, n_c variables less have to be computed in each node in the branch and bound tree. This is not done completely for free. The expressions for $\tilde{H}$ and $\tilde{f}$ have to be calculated. In the examples, these calculations are included in the solution times presented. In a real world MPC problem, some of these calculations could probably have been re-used in several consecutive, or all, MPC optimizations. Finally, in Approach III, real variables have first been eliminated according to Section 5.1.3 and then the preprocessing algorithm has been applied to compute as many variables as possible. Generally, any variables then remaining are computed using an ordinary MIQP, or BQP, solver. In the examples in this section, the preprocessing algorithm determines the optimal value of all variables. Therefore, the MIQP solver is actually never used in Approach III. In all three approaches a slightly modified version of the MIQP solver `miqp.m` presented in [15] has been used. To be able to make a fair comparison, all available combinations of the settings ”method” and ”branchrule” in the solver have been tested. In the table, only the shortest solution time achieved is presented. For further information of available settings, see [15].

The preprocessing algorithm can be implemented in several different ways. In MATLAB, an implementation with vectorized expressions has better performance than one with for-loops. The code in `miqp.m` is not vectorized. Therefore, to be able to make a fair comparison between the MIQP solver and the preprocessing algorithm, the tests of the conditions from Corollary 5.1 in the preprocessing algorithm are performed one row at a time by using for-loops.

All tests have been performed on a Sun UltraSPARC-IIe 500 MHz with 640 Mb RAM running SunOS 5.8 and MATLAB version 6.5.1. The time measurements have been performed using the MATLAB functions `tic` and `toc`.

Mass Position Control

The control problem described in this section is the mass position control problem introduced in Section 3.4.1. In this example, the position of the mass is supposed to follow a

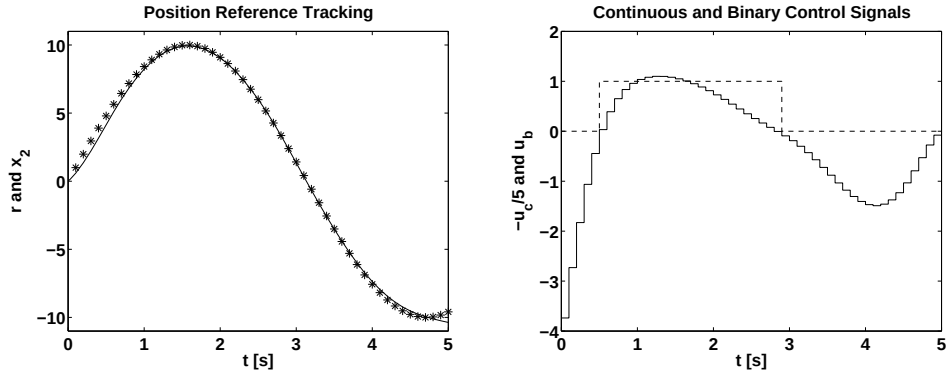


Figure 5.1: *The left plot shows how the position of the mass, x_2 (solid), follows the sampled reference signal, r (starred), when the calculated optimal control signal sequence is applied to the continuous model of the system. The right plot shows the control signals, u_c (solid) and u_b (dashed). Note that u_c is scaled in the plot.*

reference signal which is $r(t) = 10\sin(t)$. The state x_1 is the velocity of the mass and the state x_2 is the position. To obtain a system on the form (5.25), zero order hold sampling has been used with the sampling time 0.1 s. The problem is solved over a time horizon of 50 steps. The control signal cost is chosen in a way that makes it beneficial to use the binary control signal, when it is possible. The cost function used in this example is of the type described in Section 5.2.1, with

$$Q_e = 100,\ Q_{u_c} = 1,\ Q_{u_b} = 1 \tag{5.28}$$

The initial state is

$$x_1(0) = 5 \text{ and } x_2(0) = 0 \tag{5.29}$$

The result is shown in Figure 5.1. The computational time for optimizing 50 real and 50 binary variables is presented in Table 5.1. The tree exploring strategy used in the first and second approaches was the standard breadth first strategy. The node selection strategy was chosen to ”max”, see [15]. This combination of settings was one of the best choices available for this problem (the exploring strategies breadth first, best first and normalized best first gave approximately the same performance). In the third approach, all 50 binary variables were determined by Algorithm 5.2. It can be noticed that the computational time is reduced with a factor of about 180, compared to using Approach I.

Table 5.1: *Performance tests.*

Optimization method	**Solution time [s]**
Approach I	15.861
Approach II	4.618
Approach III	0.0868

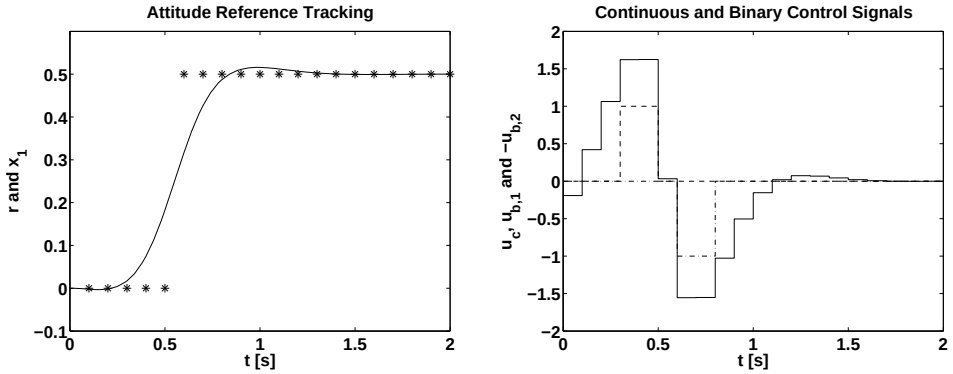


Figure 5.2: *The left plot shows how the attitude of the satellite, x_1 (solid), follows the sampled reference signal, r (starred), when the calculated optimal control signal sequence is applied to the continuous model of the system. The right plot shows the control signals, u_c (solid), $u_{b,1}$ (dashed) and $u_{b,2}$ (dash-dotted).*

Satellite Attitude Control

The control problem described in this section is the satellite attitude control problem introduced in Section 3.4.2. The states of the system are chosen to be the satellite attitude x_1, the satellite angular velocity x_2 and the internal wheel velocity x_3. To obtain a system on the form (5.25), zero order hold sampling with the sampling time 0.1 s has been used. The time horizon used is 20 samples. The cost function used in this example is of the type described in Section 5.2.1, with

$$Q_e = \text{diag}\left(0.5 \cdot 10^4, 10^{-2}, 10^{-1}\right),\ Q_{u_c} = 10,\ Q_{u_b} = 10 \cdot I_2 \tag{5.30}$$

The initial state is

$$x_1(0) = 0, x_2(0) = 0 \text{ and } x_3(0) = 0 \tag{5.31}$$

In this example, the reference signal for the attitude of the satellite is a step function with the amplitude 0.5. The reference signals for the other states are chosen to zero. The optimal control signal sequence and the attitude of the satellite is shown in Figure 5.2. The computational time for optimizing 20 real and 40 binary variables is found in Table 5.2. The tree exploring strategy used in the first and second approaches was ordinary depth first, and the node selection strategy was chosen to ”min”, see [15]. This combination of settings was the best choice available for this problem. In this example the preprocessing algorithm determined 40 out of 40 binary variables. The computational time was reduced with a factor of about 275, compared to Approach I.

5.3 Application of the Preprocessing Algorithm to Multiuser Detection

In this section, it is shown how to apply Algorithm 5.2 to the BQP problem (4.13) and how the result can be interpreted.

Table 5.2: *Performance tests.*

Optimization method	Solution time [s]
Approach I	11.449
Approach II	6.916
Approach III	0.0414

5.3.1 Using Preprocessing

In order to be able to apply the preprocessing algorithm, the optimization problem (4.13) has to rewritten on the BQP form (5.2). Note especially the domain of the optimization variable x. In order to convert (4.13) to an optimization problem with binary variables, the following variable substitution is performed

$$b = 2\bar{b} - \mathbf{1} \tag{5.32}$$

where $\bar{b} \in \{0,1\}^K$, $b \in \{-1,+1\}^K$, $\mathbf{1}$ denotes a column vector with all elements equal to one and K is the number of simultaneous users in the system. Using (5.32), neglecting constant terms and dividing by 4, the objective function in (4.13) can be rewritten as

$$\frac{1}{2}\bar{b}^T H \bar{b} + \tilde{f}^T \bar{b} \tag{5.33}$$

where

$$\tilde{f} = -\frac{1}{2}H\mathbf{1} - \frac{1}{2}Ay \tag{5.34}$$

The problem is now on the form (5.2), on which preprocessing can be performed.

5.3.2 Interpretation of the Result

Using the notation in the multiuser detection problem, the conditions (i) and (ii) in Corollary 5.1 can after simplification be written as

$$\begin{cases} A_i y_i \leq -\sum_{j \neq i} |H_{ij}| & (i) \\ A_i y_i \geq \sum_{j \neq i} |H_{ij}| & (ii) \end{cases} \tag{5.35}$$

Combining (5.32) and (5.35), it follows that the optimal choice of b_i is given by

$$b_i^* = \begin{cases} -1, & \text{if } (i) \text{ holds} \\ 1, & \text{if } (ii) \text{ holds} \end{cases} \tag{5.36}$$

An interpretation of the conditions in (5.35) is that the sum of the non-diagonal terms represents the maximum sum of interference energy possibly affecting user i. Because all terms in the sum always are positive, it can be interpreted as if all users sent the worst possible choice from $\{-1,+1\}$. If $A_i y_i$ is larger than this maximum known disturbance energy, then it is most likely that user i sent the symbol 1. Analogously, if $A_i y_i$ is smaller than the negative maximum known disturbance energy, then it is most likely that user i

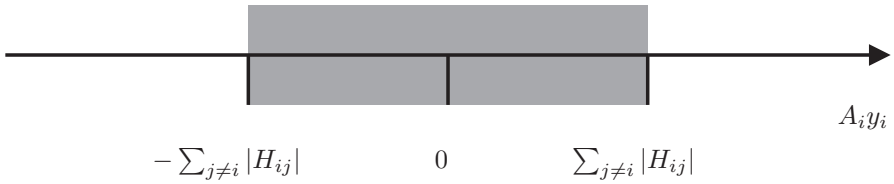


Figure 5.3: *This figure shows the decision regions for the preprocessing algorithm, when the non-iterative Algorithm 5.1 is used. The grey region shows the region where variable i cannot be computed by non-iterative preprocessing. The size of this region might be reduced when the iterated version of the algorithm in Algorithm 5.2 is used.*

sent the symbol -1. This behavior seems reasonable since the noise is assumed to have zero mean. The decision strategy is illustrated in Figure 5.3.

If (5.35) is investigated, it can be realized that a necessary property of the cross-correlation matrix in order to be able to successfully use the algorithm, is that the chip sequences give low cross-correlations between different users. This is typically the case for, for example, Gold Sequences. Compared to previous optimal low complexity methods presented in [83, 85, 93], the algorithm presented in this thesis does not introduce any requirements on the sign of the cross-correlations or that the cross-correlations between users are equal.

5.3.3 Simulation Results

In this section, the preprocessing algorithm is applied to the multiuser detection problem and tested in Monte Carlo simulations. In the first simulations, the joint Bit Error Rate (BER) for the optimal detector implemented by using the preprocessing algorithm is compared to the joint BER of the conventional detector, [56],

$$\hat{b} = \text{sign}(y) \tag{5.37}$$

and to the joint BER of the decorrelating detector, [56],

$$\hat{b} = \text{sign}(H^{-1}y) \tag{5.38}$$

where y denotes the output from the matched filters as described in Section 4.2.1. To be able to make a fair comparison, in all but the last example only the variables computed by the preprocessing algorithm were used in the BER calculations for all methods in the comparisons. The tests were performed with Gold Sequences of length 128. The algorithms were compared for the loads 1 to 127 users. Each load was tested 10000 times. In each test, a new noise realization and a new random bit was assigned to each user. In Figure 5.4 it can be noticed that the preprocessing algorithm computes nearly all variables in average. In the worst case, 75 % of the variables were computed. The next issue to verify is that the problem instances tested were not “trivial” in the sense that the existing lowest complexity algorithms also could compute them optimally. This verification is performed by calculating the average BER for the different methods during the Monte

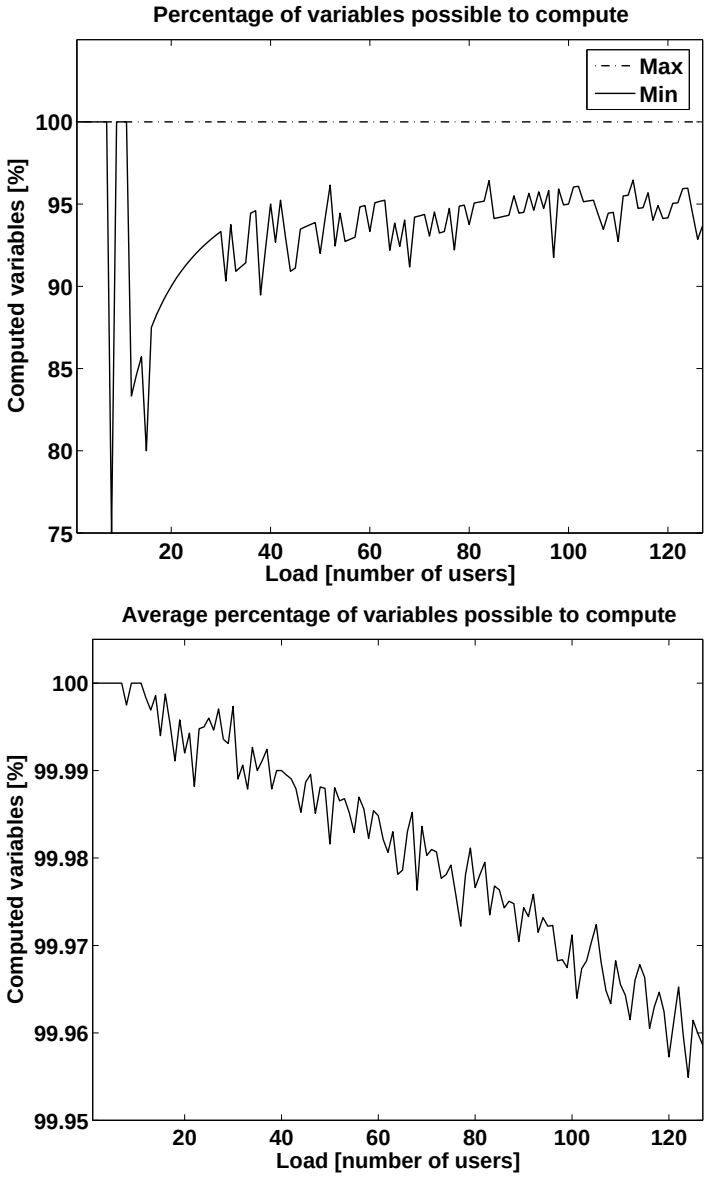


Figure 5.4: *The plots show how many percents of the variables that were computed by preprocessing for different loads. In the upper plot two curves are shown: The min-curve shows the lowest amount of computed variables during the 10000 simulations. The max-curve shows the greatest amount of computed variables during the simulations. In the lower plot, the average of the amount of computed variables over the 10000 realizations is shown. Note that the axis scaling are different in the upper and in the lower plot.*

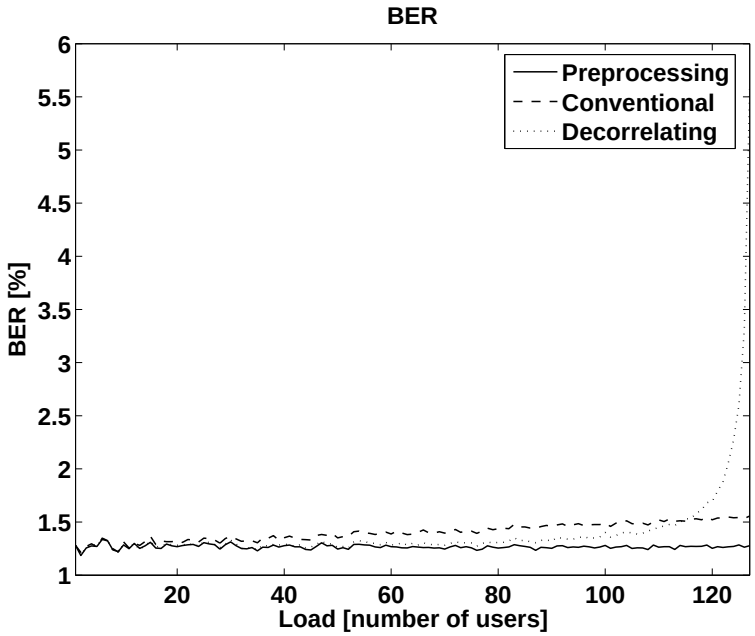


Figure 5.5: *The plot shows the BER of the different methods as a function of the load when Gold sequences of length 128 are used. It can be seen that the optimal multiuser detector implemented by the preprocessing algorithm has the lowest BER. Only variables computed by the preprocessing algorithm are considered in the comparison. Note that since on average more than* 99.95 *% of the variables are computed by the preprocessing algorithm, the variables not computed by the algorithm do not change the presented result significantly.*

Carlo simulations. The result from this simulation is shown in Figure 5.5. In the test, the Signal to Noise Ratio (SNR) for user 1 varied from 7 dB to 6.7 dB. The conclusion drawn from the simulation is that the optimal multiuser detector implemented by the preprocessing algorithm gives lower BER than the two other algorithms. Note that the plots for all three methods only include bits possible to calculate with the preprocessing algorithm. The same plot also verifies that the iterated algorithm in Algorithm 5.2, enables more "wise" decisions than Algorithm 5.1 which only uses a single iteration. In the non-iterated case, it follows from (5.35) that the optimal decision coincides with the decision taken by the detector in (5.37), in the region where variables can be computed by the preprocessing algorithm. The region where the solution from the algorithms coincide is the region outside the grey region in Figure 5.3. When the iterated implementation of the preprocessing algorithm is used it can sometimes be possible to decrease the size of the grey area for some variables in the problem. This means that in those cases more variables are possible to compute by preprocessing.

The computational time is illustrated in Figure 5.6. The conventional detector (5.37) is not shown in the plot because its computational time is negligible in comparison with

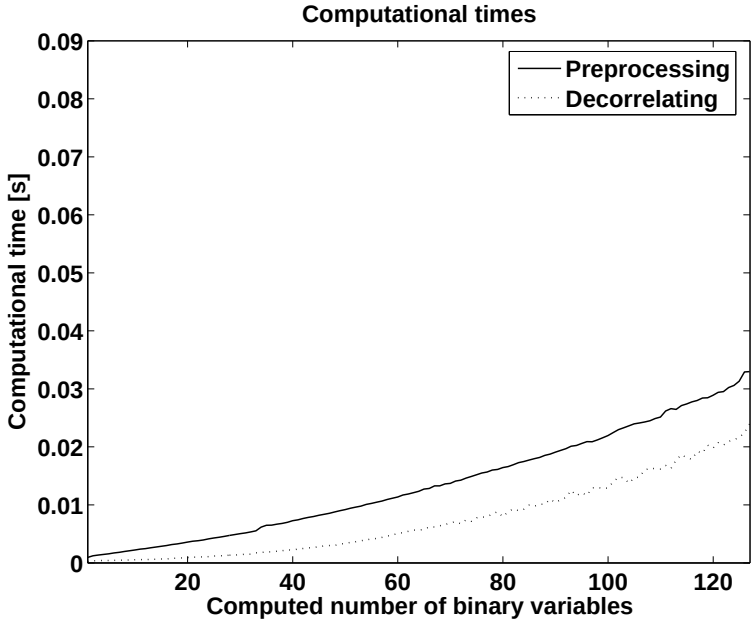


Figure 5.6: *In the plot, the computational time for preprocessing and the decorrelating detector are presented. To be able to make a fair comparison, the computational time shown for the decorrelating detector is only the time it takes to compute the variables computable by preprocessing. The conventional detector has significantly lower computational time and it has therefore been excluded from this plot.*

the other two. The time measurements have been performed using the MATLAB functions `tic` and `toc`. The conclusion is drawn that the computational complexity for the preprocessing algorithm is similar to the one for the decorrelating detector (5.38). It should be mentioned that the matrix inversion performed in (5.38) in MATLAB is implemented much more efficiently than the preprocessing algorithm. For example, by implementing the preprocessing algorithm in C, a significant reduction of the computational time is expected. The tests of the computational times were performed on a Sun UltraSPARC-IIe 500 MHz with 640 Mb RAM running SunOS 5.9 and MATLAB 7.0.1.

Gold sequences have very low cross-correlation. To test the algorithm as the cross-correlation increases, the cross-correlation matrix was manually modified. This was done by adding $0.01, 0.02, \ldots, 0.6$ in 60 steps to 16 symmetric off-diagonal elements (8 elements on each side of the diagonal). For each of the 60 steps, 10000 Monte Carlo simulations were performed. The test was performed for 100 users and the result from this simulation can be found in Figure 5.7. The number of variables possible to compute by preprocessing when the correlation is increased is illustrated by Figure 5.8.

An important property of the solution delivered by the proposed algorithm is that both Algorithm 5.1 and Algorithm 5.2 provide a certificate of optimality, that is, even though the solution sometimes coincides with the algorithms (5.37) and (5.38), the solution computed by the latter algorithms cannot be guaranteed to be optimal. Note that, if the solution

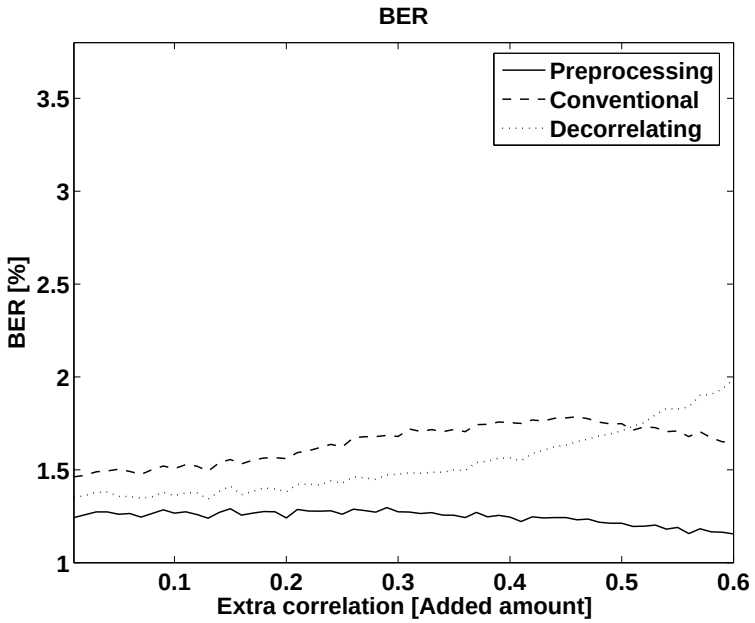


Figure 5.7: *This figure shows BER as a function of the extra added off-diagonal correlation. Only variables solved by the preprocessing algorithm is presented in the comparison. The average number of variables solved for each added correlation is shown in Figure 5.8.*

of the preprocessing algorithm coincide with the solution of a suboptimal detector for a specific region of y_i, this implies that the suboptimal detector actually takes optimal decisions in the region considered. With the certificate of optimality it can be, from case to case, worth to apply a higher complexity algorithm to compute any remaining variables and hence compute the optimal solution to the entire problem. The alternative is to apply a low complexity suboptimal algorithm to compute these remaining variables. In the last simulation, the variables not possible to compute by the preprocessing algorithm were computed by the conventional detector (5.37) and by the decorrelating detector (5.38). This was performed by first subtracting the part of y originating from the bits computed by preprocessing. Denote the vector containing these bits $\hat{b}_p$. Using (4.7), the remaining signal was computed as

$$y_{remain} = y(r) - R(r,c)A(c,c)\hat{b}_p \tag{5.39}$$

where r contains the indices in the original problem remaining to be computed and c contains the indices in the original problem of the variables computed by preprocessing. The detectors (5.37) and (5.38) were finally applied to the remaining signal, y_{remain}. The BER is illustrated in Figure 5.9. The conclusion from the result in the figure is that the preprocessing algorithm followed by a simple suboptimal algorithm will reduce the BER significantly compared to using only the suboptimal algorithm.

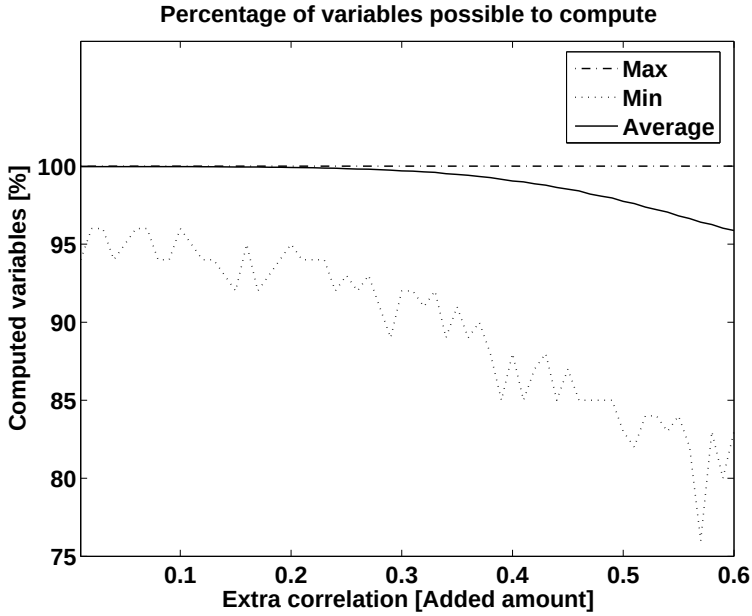


Figure 5.8: *In the plot the number of variables possible to solve for different modifications of the correlation is presented.*

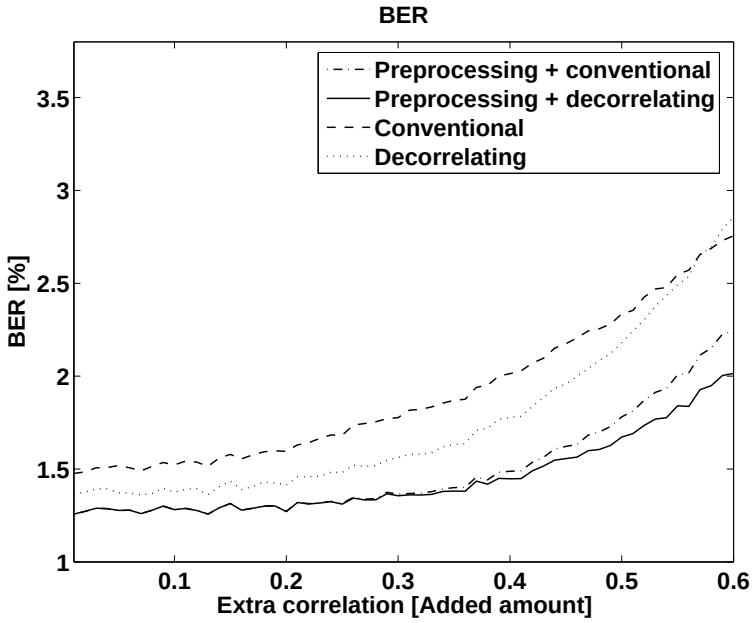


Figure 5.9: *This figure shows BER as a function of the extra added off-diagonal correlation when the variables not solved by preprocessing are solved by two suboptimal approaches, the conventional algorithm* (5.37) *and the decorrelating algorithm* (5.38)*. In this plot, all bits sent are considered in the BER calculation.*

6

A Mixed Integer Dual Quadratic Programming Algorithm

As described in Section 3, when linear MPC is extended to handle hybrid systems, more challenging problems have to be solved. The method most commonly used is branch and bound, which is described in Section 2.4. In branch and bound for MIQP problems, many QP subproblems have to be solved in the nodes of the tree. It can then be advantageous to consider QP solvers that can be warm started efficiently. By warm start it is meant that the solver is supplied with information from a previously solved problem similar to the one to be solved. The idea is that this information will reduce the computational effort needed when reoptimizing after a minor modification of the problem. In Section 6.1, a dual QP solver is presented. This solver is tailored for linear MPC and can easily be warm started. In Section 6.2, this solver is used in a branch and bound method. It is shown how warm starts can be applied and the performance of the algorithm is investigated.

6.1 A Dual Quadratic Programming Algorithm

In this section, a solver tailored for linear MPC is derived. The aim is to derive a solver which can solve the subproblems in a branch and bound algorithm efficiently. From the discussions in Section 2.5.2 and Section 3.3.2, it can be concluded that because of its good warm start abilities, a dual active set QP solver would probably be a very good choice for solving the node problems. Therefore, the solver presented in this section is working in the dual space.

6.1.1 Problem Definition

As described in Section 3, the MPC optimization problem can be cast on the QP form in two different ways. Either, it can be written as a QP problem with only control signals as optimization variables or it can be written as a QP where control signals, states and

control errors all are optimization variables. In the first approach, the dynamics is embedded in the objective function, while in the latter approach, the underlying MPC problem is more directly visible, but it involves more variables. Both formulations are derived in Appendix B for a general linear MPC problem. If the resulting KKT systems are examined, it can be seen that the linear part for the first approach involves a dense system while the second approach involves an almost block diagonal system. In this section, it will be shown that the latter system can be solved using a Riccati recursion. It will be seen that this is advantageous from a computational point of view.

Consider a linear MPC problem similar to the one presented in (3.1) on the form

$$\begin{aligned}
\underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & J_P(\mathsf{x},\mathsf{u},\mathsf{e}) = \frac{1}{2}\sum_{t=0}^{N-1} e^T(t)Q_e(t)e(t) + u^T(t)Q_u(t)u(t) + \\
& \qquad + \frac{1}{2}e^T(N)Q_e(N)e(N) \\
\text{subject to} \quad & x(0) = x_0 \\
& x(t+1) = A(t)x(t) + B(t)u(t), \qquad t = 1,\ldots,N-1 \\
& e(t) = M(t)x(t), \qquad t = 1,\ldots,N \\
& h(0) + H_u(0)u(0) \leq 0 \\
& h(t) + H_x(t)x(t) + H_u(t)u(t) \leq 0, \qquad t = 1,\ldots,N-1
\end{aligned} \tag{6.1}$$

where x, u and e are defined as in Appendix B with $t_0 = 0$, and $H_x(t) \in \mathbb{R}^{c(t)\times n}$, $H_u(t) \in \mathbb{R}^{c(t)\times m}$ and $h(t) \in \mathbb{R}^{c(t)}$ where $c(t)$ denotes the number of inequality constraints at time t. Furthermore, the following assumptions are made

Assumption A1. $Q_e(t) \in \mathbb{S}^p_{++},\ t = 0,\ldots,N$

Assumption A2. $Q_u(t) \in \mathbb{S}^m_{++},\ t = 0,\ldots,N-1$

6.1.2 Derivation of the Dual Problem

In order to design a solver working on the problem dual to (6.1), the dual optimization problem has to be derived. The optimization problem in (6.1) is on the form (2.27). Therefore, deriving the dual problem to (6.1) is a special case of the procedure in Section 2.4. The first step in the procedure is to form the Lagrangian.

$$\begin{aligned}
L_P(\mathsf{x},\mathsf{u},\mathsf{e},\alpha,\beta,\gamma) &= \frac{1}{2}\sum_{t=0}^{N-1} e^T(t)Q_e(t)e(t) + u^T(t)Q_u(t)u(t) + \frac{1}{2}e^T(N)Q_e(N)e(N) \\
&+ \alpha^T(0)\big(x_0 - x(0)\big) + \sum_{t=0}^{N-1} \alpha^T(t+1)\big(A(t)x(t) + B(t)u(t) - x(t+1)\big) \\
&+ \sum_{t=0}^{N} \beta^T(t)\big(M(t)x(t) - e(t)\big) + \gamma^T(0)\big(h(0) + H_u(0)u(0)\big) \\
&+ \sum_{t=1}^{N-1} \gamma^T(t)\big(h(t) + H_x(t)x(t) + H_u(t)u(t)\big)
\end{aligned} \tag{6.2}$$

where α, β and γ are the Lagrange multiplier vectors associated with the optimization problem (6.1)

$$\begin{aligned}\alpha &= [\alpha^T(0), \ldots, \alpha^T(N)]^T \\ \beta &= [\beta^T(0), \ldots, \beta^T(N)]^T \\ \gamma &= [\gamma^T(0), \ldots, \gamma^T(N-1)]^T\end{aligned} \tag{6.3}$$

These are also known as the dual variables. Following (2.12), the Lagrange dual function for this problem is

$$g(\alpha, \beta, \gamma) = \inf_{\mathsf{x},\mathsf{u},\mathsf{e}} L(\mathsf{x}, \mathsf{u}, \mathsf{e}, \alpha, \beta, \gamma) \tag{6.4}$$

Hence, the Lagrange dual problem associated with (6.1) is

$$\begin{aligned}\underset{\alpha,\beta,\gamma}{\text{maximize}} \quad & g(\alpha, \beta, \gamma) \\ \text{subject to} \quad & \gamma \geq 0\end{aligned} \tag{6.5}$$

In order to write down an explicit expression for (6.4), the Lagrangian function in (6.2) is rewritten on the following form

$$\begin{aligned}&L_P(\mathsf{x}, \mathsf{u}, \mathsf{e}, \alpha, \beta, \gamma) \\ &= \frac{1}{2}\sum_{t=0}^{N-1} e^T(t)Q_e(t)e(t) + u^T(t)Q_u(t)u(t) + \frac{1}{2}e^T(N)Q_e(N)e(N) \\ &+ \big(\alpha^T(1)A(0) - \alpha^T(0) + \beta^T(0)M(0)\big)x(0) \\ &+ \sum_{t=1}^{N-1}\big(\alpha^T(t+1)A(t) - \alpha^T(t) + \beta^T(t)M(t) + \gamma^T(t)H_x(t)\big)x(t) \\ &+ \sum_{t=0}^{N-1}\big(\alpha^T(t+1)B(t)u(t) - \beta^T(t)e(t) + \gamma^T(t)h(t) + \gamma^T(t)H_u(t)u(t)\big) \\ &+ \alpha^T(0)x_0 + \big(-\alpha^T(N) + \beta^T(N)M(N)\big)x(N) - \beta^T(N)e(N)\end{aligned} \tag{6.6}$$

As for the optimization problem (2.27), the Lagrangian is a linear function of the primal optimization variables not explicitly present in the primal objective function. Hence, when minimizing the Lagrangian with respect to all primal variables, an implicit equality constraint similar to the one found in (2.28) can be found also in (6.6).

By examining the Lagrangian (6.6), it follows that L_P is a linear function of x and a strictly convex function of u and e. If the minimization is performed with a non-zero coefficient in front of x, the optimal value will not be bounded from below. However, a more explicit formulation of (6.5) can be obtained by adding constraints to (6.5), which exclude the values of (α, β, γ) for which $g(\alpha, \beta, \gamma) = -\infty$. By examining (6.6), these constraints are found to be

$$\begin{aligned}A^T(0)\alpha(1) - \alpha(0) + M^T(0)\beta(0) &= 0 \\ A^T(t)\alpha(t+1) - \alpha(t) + M^T(t)\beta(t) + H_x^T(t)\gamma(t) &= 0, \qquad t = 1, \ldots, N-1 \\ -\alpha(N) + M^T(N)\beta(N) &= 0\end{aligned} \tag{6.7}$$

Because L_P is a strictly convex continuously differentiable function of u and e, the minimization over these variables can easily be performed using the first order necessary and sufficient conditions of optimality. These are

$$\begin{aligned}\frac{\partial L_P}{\partial u(t)} &= Q_u(t)u(t) + B^T(t)\alpha(t+1) + H_u^T(t)\gamma(t) = 0, \qquad t = 0,\dots,N-1\\ \frac{\partial L_P}{\partial e(t)} &= Q_e(t)e(t) - \beta(t) = 0, \qquad t = 0,\dots,N\end{aligned} \tag{6.8}$$

By combining and rewriting (6.7) and (6.8), the following equations are obtained

$$\begin{aligned}\alpha(0) &= A^T(0)\alpha(1) + M^T(0)\beta(0)\\ \alpha(t) &= A^T(t)\alpha(t+1) + M^T(t)\beta(t) + H_x^T(t)\gamma(t), \qquad t = 1,\dots,N-1\\ \alpha(N) &= M^T(N)\beta(N)\\ u(t) &= -Q_u^{-1}(t)\big(B^T(t)\alpha(t+1) + H_u^T(t)\gamma(t)\big), \qquad t = 0,\dots,N-1\\ e(t) &= Q_e^{-1}(t)\beta(t), \qquad t = 0,\dots,N\end{aligned} \tag{6.9}$$

To obtain an explicit expression for $g(\alpha,\beta,\gamma)$, (6.7) and the expressions for $u(t)$ and $e(t)$ in (6.9) are inserted into (6.6). After simplifications the result is

$$\begin{aligned}g(\alpha,\beta,\gamma) = -\frac{1}{2}\sum_{t=0}^{N-1}\big(&\beta^T(t)Q_e^{-1}(t)\beta(t) + \alpha^T(t+1)B(t)Q_u^{-1}(t)B^T(t)\alpha(t+1)\\ &+ 2\gamma^T(t)H_u(t)Q_u^{-1}(t)B^T(t)\alpha(t+1) + \gamma^T(t)H_u(t)Q_u^{-1}(t)H_u^T(t)\gamma(t)\\ &- 2\gamma^T(t)h(t)\big) - \frac{1}{2}\beta^T(N)Q_e^{-1}(N)\beta(N) + \alpha^T(0)x_0\end{aligned} \tag{6.10}$$

By combining the first three lines in (6.9) with (6.10), an explicit expression for the problem in (6.5) can be given as

$$\begin{aligned}\underset{\alpha,\beta,\gamma}{\text{maximize}} \quad & -\frac{1}{2}\sum_{t=0}^{N-1}\big(\alpha^T(t+1)B(t)Q_u^{-1}(t)B^T(t)\alpha(t+1) +\\ & + \begin{bmatrix}\beta^T(t) & \gamma^T(t)\end{bmatrix}\begin{bmatrix}Q_e^{-1}(t) & 0\\ 0 & H_u(t)Q_u^{-1}(t)H_u^T(t)\end{bmatrix}\begin{bmatrix}\beta(t)\\ \gamma(t)\end{bmatrix} +\\ & + 2\alpha^T(t+1)\begin{bmatrix}0 & B(t)Q_u^{-1}(t)H_u^T(t)\end{bmatrix}\begin{bmatrix}\beta(t)\\ \gamma(t)\end{bmatrix} - 2h^T(t)\gamma(t)\big) -\\ & - \frac{1}{2}\beta^T(N)Q_e^{-1}(N)\beta(N) + x_0^T\alpha(0)\\ \text{subject to} \quad & \alpha(0) = A^T(0)\alpha(1) + M^T(0)\beta(0)\\ & \alpha(t) = A^T(t)\alpha(t+1) + \begin{bmatrix}M^T(t) & H_x^T(t)\end{bmatrix}\begin{bmatrix}\beta(t)\\ \gamma(t)\end{bmatrix}, \; t = 1,\dots,N-1\\ & \alpha(N) = M^T(N)\beta(N)\\ & \gamma(t) \geq 0, \qquad t = 0,\dots,N-1\end{aligned} \tag{6.11}$$

To make the dual optimization problem more look like an optimal control problem, the procedure in [81] is followed, that is, the dual variables are changed and the time is reversed according to

$$\begin{aligned} \alpha(t) = \tilde{x}(N-t),\ \beta(t) = w(N-t-1),\ \gamma(t) = v(N-t-1) \\ \tau = N - t \end{aligned} \tag{6.12}$$

and the following definitions are made

$$\begin{aligned} \tilde{u}(-1) &= w(-1) \\ \tilde{u}(\tau) &= \begin{bmatrix} w^T(\tau) & v^T(\tau) \end{bmatrix}^T, \quad \tau = 0, \ldots, N-1 \\ \tilde{Q}_{\tilde{u}}(-1) &= Q_e^{-1}(N) \\ \tilde{B}(-1) &= M^T(N) \\ \tilde{Q}_{\tilde{x}}(\tau) &= B(N-\tau-1)Q_u^{-1}(N-\tau-1)B^T(N-\tau-1) \\ \tilde{Q}_{\tilde{u}}(\tau) &= \operatorname{diag}(Q_e^{-1}(N-\tau-1), H_u(N-\tau-1)Q_u^{-1}(N-\tau-1)H_u^T(N-\tau-1)) \\ \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) &= \begin{bmatrix} 0 & B(N-\tau-1)Q_u^{-1}(N-\tau-1)H_u^T(N-\tau-1) \end{bmatrix} \\ \tilde{q}_{\tilde{u}}(\tau) &= \begin{bmatrix} 0 & -h^T(N-\tau-1) \end{bmatrix}^T \\ \tilde{A}(\tau) &= A^T(N-\tau-1) \\ \tilde{B}(\tau) &= \begin{bmatrix} M^T(N-\tau-1) & H_x^T(N-\tau-1) \end{bmatrix} \\ \tilde{q}_{\tilde{x}}(N) &= -x_0 \\ \tilde{B}(N) &= \begin{bmatrix} M^T(0) & 0 \end{bmatrix} \end{aligned} \tag{6.13}$$

where the relations hold for $\tau = 0, \ldots, N-1$ unless stated differently. Finally, in order to obtain a minimization problem, the sign of the objective is changed. The optimization problem is then on the form

$$\begin{aligned} \underset{\tilde{\mathrm{x}}, \tilde{\mathrm{u}}}{\text{minimize}} \quad & J_D(\tilde{\mathrm{x}}, \tilde{\mathrm{u}}) = \frac{1}{2}\tilde{u}^T(-1)\tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) \\ & \quad + \frac{1}{2}\sum_{\tau=0}^{N-1} \left(\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) + \tilde{u}^T(\tau)\tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) \right. \\ & \quad \left. + 2\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau) + 2\tilde{q}_{\tilde{u}}^T(\tau)\tilde{u}(\tau) \right) + \tilde{q}_{\tilde{x}}^T(N)\tilde{x}(N) \\ \text{subject to} \quad & \quad \tilde{x}(0) = \tilde{B}(-1)\tilde{u}(-1) \\ & \quad \tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}(\tau)\tilde{u}(\tau), \quad \tau = 0, \ldots, N-1 \\ & \begin{bmatrix} 0 & -I_{c(N-\tau-1)} \end{bmatrix} \tilde{u}(\tau) \leq 0, \quad \tau = 0, \ldots, N-1 \end{aligned} \tag{6.14}$$

where

$$\begin{aligned} \tilde{\mathrm{x}} &= \begin{bmatrix} \tilde{x}^T(0), \ldots, \tilde{x}^T(N) \end{bmatrix}^T \\ \tilde{\mathrm{u}} &= \begin{bmatrix} \tilde{u}^T(-1), \ldots, \tilde{u}^T(N-1) \end{bmatrix}^T \end{aligned} \tag{6.15}$$

and $\tilde{x}(\tau) \in \mathbb{R}^{\tilde{n}}$ and $\tilde{u}(\tau) \in \mathbb{R}^{\tilde{m}(\tau)}$. The dual state dimension and the dual control signal dimension are related to dimensions of primal variables by the equations $\tilde{n} = n$ and $\tilde{m}(\tau) = c(N - \tau - 1)$.

Note that it is actually abuse of notation to refer to (6.14) as the dual problem to (6.1). To be exact, (6.14) is a problem equivalent to the dual problem of (6.1). That is, the two problems have the same solution, see Definition 2.4. Despite this, (6.14) will from now on be called the dual problem of (6.1).

Remark 6.1. In the derivation of the algorithm, a reference signal has been omitted. If desired, a reference signal $r(t)$ can readily be included by setting $e(t) = M(t)x(t) - r(t)$ and redefining $\tilde{q}_{\tilde{u}}(\tau) = \begin{bmatrix} r^T(N-\tau-1) & -h^T(N-\tau-1) \end{bmatrix}^T$.

The dual problem can be interpreted as an optimal control problem where the initial state $\tilde{x}(-1)$ is fixed to the origin and with positivity constraints on some of the control signals. Also, the final state penalty is linear. An alternative interpretation is that, in some cases, it can be reformulated as an estimation problem with cross penalties between the process noise and the measurement noise, and with some of the process noise components constrained to be positive. More on duality between optimal control problems and estimation problems can be found in [81] and [49].

6.1.3 Optimality Conditions for the Dual Problem

The interest is now focused on problem (6.14). The idea is to solve the primal problem (6.1) by solving the dual problem (6.14). In order to solve the dual problem, the optimality conditions for this problem have to be derived. Define the Lagrangian of (6.14) as

$$\begin{aligned} L_D(\tilde{\mathrm{x}}, \tilde{\mathrm{u}}, \lambda, \mu) = {} & \frac{1}{2}\tilde{u}^T(-1)\tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) \\ & + \frac{1}{2}\sum_{\tau=0}^{N-1} \big(\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) + \tilde{u}(\tau)^T\tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) \\ & + 2\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau) + 2\tilde{q}_{\tilde{u}}^T(\tau)\tilde{u}(\tau)\big) + \tilde{q}_{\tilde{x}}^T(N)\tilde{x}(N) \\ & + \lambda^T(0)\big(\tilde{B}(-1)\tilde{u}(-1) - \tilde{x}(0)\big) \\ & + \sum_{\tau=0}^{N-1} \lambda^T(\tau+1)\big(\tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}(\tau)\tilde{u}(\tau) - \tilde{x}(\tau+1)\big) \\ & - \sum_{\tau=0}^{N-1} \mu^T(\tau)\begin{bmatrix} 0 & I_{c(N-\tau-1)} \end{bmatrix}\tilde{u}(\tau) \end{aligned} \tag{6.16}$$

where

$$\begin{aligned} \lambda &= \begin{bmatrix} \lambda^T(0), \ldots, \lambda^T(N) \end{bmatrix}^T \\ \mu &= \begin{bmatrix} \mu^T(0), \ldots, \mu^T(N-1) \end{bmatrix}^T \end{aligned} \tag{6.17}$$

If the results from Section 2.4.1 are applied to problem (6.1), it follows that the dual problem is always feasible if the primal problem is feasible. If the dual problem is feasible, according to Remark 2.1 strong duality holds and the KKT conditions in Theorem 2.3

are fulfilled and therefore constitute necessary and sufficient conditions of optimality for (6.14). In case the primal problem is infeasible, then according to the results in Section 2.4.1, the dual is either infeasible or unbounded. Consider the case when the primal problem is feasible and apply Theorem 2.3 to the dual problem. The equations (2.20a) and (2.20b) will then generate the following equations

$$\begin{aligned}\begin{bmatrix}0 & -I_{c(N-\tau-1)}\end{bmatrix}\tilde{u}(\tau) &\leq 0, \qquad \tau = 0,\dots,N-1\\ \tilde{x}(0) &= \tilde{B}(-1)\tilde{u}(-1)\\ \tilde{x}(\tau+1) &= \tilde{A}(\tau)\tilde{x}(\tau)+\tilde{B}(\tau)\tilde{u}(\tau), \qquad \tau = 0,\dots,N-1\end{aligned} \tag{6.18}$$

From the dual feasibility condition (2.20c),

$$\mu(\tau) \geq 0, \qquad \tau = 0,\dots,N-1 \tag{6.19}$$

The complementary slackness condition (2.20d) gives the equation

$$\mu_i(\tau)\tilde{u}_{\tilde{m}(\tau)-c(N-\tau-1)+i}(\tau) = 0, \qquad i = 1,\dots,c(N-\tau-1),\ \tau = 0,\dots,N-1 \tag{6.20}$$

Finally, from (2.20e) the following equations are obtained

$$\frac{\partial L_D}{\partial \tilde{x}(\tau)} = \tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau)+\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau)+\tilde{A}^T(\tau)\lambda(\tau+1)-\lambda(\tau) = 0, \tag{6.21a}$$

$$\tau = 0,\dots,N-1$$

$$\frac{\partial L_D}{\partial \tilde{x}(N)} = \tilde{q}_{\tilde{x}}(N)-\lambda(N) = 0 \tag{6.21b}$$

$$\frac{\partial L_D}{\partial \tilde{u}(\tau)} = \tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau)+\tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau)\tilde{x}(\tau)+\tilde{q}_{\tilde{u}}(\tau)+\tilde{B}^T(\tau)\lambda(\tau+1)-\begin{bmatrix}0\\ I\end{bmatrix}\mu(\tau) = 0, \tag{6.21c}$$

$$\tau = 0,\dots,N-1$$

$$\frac{\partial L_D}{\partial \tilde{u}(-1)} = \tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1)+\tilde{B}^T(-1)\lambda(0) = 0 \tag{6.21d}$$

6.1.4 Connection Between Primal and Dual Variables

As mentioned in the previous section, the idea is to compute the solution to the primal problem from the solution of the dual problem. How this can be performed is discussed in this section. The information in this section is expected to appear after the algorithm solving the dual problem has been presented. But the relation is also important when interpreting how the algorithm works. Thus, these relations are derived now.

In the primal problem, the primal variables are x, e and u and the dual variables are α, β and γ. In the dual problem, the primal variables are $\tilde{\text{x}}$, w and v and the dual variables are λ and μ. Note that, in the formulation (6.14), the variables w and v are stacked in the variable $\tilde{\text{u}}$. The relation between the dual variables of the primal problem, α, β and γ, and the primal variables of the dual problem, $\tilde{\text{x}}$, w and v, are given by (6.12).

By combining the fourth line in (6.9) with the first line in (6.12), the following expression for u is obtained

$$u(t) = -Q_u^{-1}(t)\big(B^T(t)\tilde{x}(N-t-1)+H_u^T(t)v(N-t-1)\big) \tag{6.22}$$

Using the expression for τ in (6.12), (6.22) can be rewritten as

$$u(N-\tau-1) = -Q_u^{-1}(N-\tau-1)\big(B^T(N-\tau-1)\tilde{x}(\tau)+H_u^T(N-\tau-1)v(\tau)\big) \quad (6.23)$$

By combining (6.21a) and (6.12), the following expression for $\lambda(N-t)$ is obtained

$$\begin{aligned}\lambda(N-t) = {} & A(t-1)\lambda(N-t+1) + B(t-1)Q_u^{-1}(t-1)B^T(t-1)\tilde{x}(N-t) \\ & + B(t-1)Q_u^{-1}(t-1)H_u^T(t-1)v(N-t)\end{aligned} \quad (6.24)$$

Identifying $u(t-1)$ according to (6.22) in (6.24) yields the following recursive expression for λ

$$\begin{aligned}\lambda(N-t) &= A(t-1)\lambda(N-t+1) - B(t-1)u(t-1) \\ \lambda(N) &= -x_0\end{aligned} \quad (6.25)$$

where the expression for $\lambda(N)$ stems from (6.21b). Now let $\bar{x}(t) = -\lambda(N-t)$ and insert into (6.25)

$$\begin{aligned}\bar{x}(0) &= x_0 \\ -\bar{x}(t) &= -A(t-1)\bar{x}(t-1) - B(t-1)u(t-1), \qquad t = 1,\dots,N\end{aligned} \quad (6.26)$$

Change the sign of the last line in (6.26) and redefine the time index

$$\begin{aligned}\bar{x}(0) &= x_0 \\ \bar{x}(t+1) &= A(t)\bar{x}(t) + B(t)u(t), \qquad t = 0,\dots,N-1\end{aligned} \quad (6.27)$$

Identifying (6.27) with (6.1) yields the relation $x(t) = \bar{x}(t)$, that is

$$x(t) = -\lambda(N-t) \quad (6.28)$$

The dual variable μ in the dual problem has also an interpretation in the primal problem. In the following calculation, (6.13) has been inserted into (6.21) in order to get a direct primal interpretation of the equations. By using (6.21d), (6.22), (6.28) and the second line in (6.12), the following equation is obtained

$$h(t) + H_x(t)x(t) + H_u(t)u(t) = -\mu(N-t-1), \qquad t = 1,\dots,N-1 \quad (6.29)$$

For $t = 0$, similar calculations give

$$h(0) + H_u(0)u(0) = -\mu(N-1) \quad (6.30)$$

If (6.29) and (6.30) are compared to (6.1), the conclusion can be drawn that the μ-variables are slack variables for the primal inequality constraints. It is also clear that $\mu \geq 0$ if the primal problem is feasible, which is exactly the condition (6.19) for dual feasibility in the dual problem.

6.1.5 Solving the Dual Problem Using Riccati Recursions

This section is opened with an assumption

Assumption A3. $H_u(t)$ has full row rank for $t = 0, \ldots, N-1$.

The dual problem is now to be solved by a primal (the dual problem is treated as a primal problem) active set QP solver as described in Algorithm 2.1. In the QP iterations, equality constrained problems of type (2.37) are solved. The KKT conditions, excluding the non-linear complementary slackness condition, is given by a linear system on the form

$$K\hat{x} = \left[\begin{array}{c:c} K_{11} & K_{12} \\ \hdashline K_{21} & K_{22} \end{array}\right] \left[\begin{array}{c} \hat{x}_1 \\ \hdashline \hat{x}_2 \end{array}\right] = \left[\begin{array}{ccc:c} H & A_{\mathcal{E}}^T & A_{\mathcal{I}\cap\mathcal{W}}^T \\ A_{\mathcal{E}} & 0 & 0 \\ \hdashline A_{\mathcal{I}\cap\mathcal{W}} & 0 & 0 \end{array}\right] \left[\begin{array}{c} \hat{x} \\ \hat{\nu} \\ \hdashline \hat{\lambda}_{\mathcal{I}\cap\mathcal{W}} \end{array}\right] = \left[\begin{array}{c} -f \\ b_{\mathcal{E}} \\ \hdashline b_{\mathcal{I}\cap\mathcal{W}} \end{array}\right] \tag{6.31}$$

where $A_{\mathcal{I}\cap\mathcal{W}}$ contains the rows in $A_{\mathcal{I}}$ corresponding to constraints contained in the working set and $b_{\mathcal{I}\cap\mathcal{W}}$ and $\hat{\lambda}_{\mathcal{I}\cap\mathcal{W}}$ are defined analogously. Note that the notation used in (6.31) is the one used in Chapter 2 for a generic QP problem and is not connected to the notation used previously in this chapter. In accordance with (2.21), the matrix $A_{\mathcal{E}}$ and the vector $b_{\mathcal{E}}$ define the equality constraints in the problem. Dual variables corresponding to inequality constraints not in the working set, that is $\left\{\hat{\lambda}_i \mid i \notin \mathcal{W}\right\}$, are set to zero by the active set algorithm. Since the upper left block in (6.31) is unchanged during the QP iterations it seems computationally efficient to proceed as in [61], where it is proposed to use block elimination (see Algorithm A.1) in order to solve (6.31). As shown in [61], the upper left block will have a structure that makes it possible to compute $K_{11}^{-1}K_{12}$ and $K_{11}^{-1}\begin{bmatrix}-f^T & b_{\mathcal{E}}^T\end{bmatrix}^T$ using Riccati recursions. Under the assumptions A1 and A2, there exists a unique solution to the KKT system for the primal problem. Therefore, $K_{11}^{-1}\begin{bmatrix}-f^T & b_{\mathcal{E}}^T\end{bmatrix}^T$ can be computed for the primal problem. Trying to follow the same approach in the dual might create problems. By simple examples, it can be shown that K_{11} in the dual problem cannot, in general, be expected to be nonsingular, and therefore, block elimination cannot be used when solving the dual problem.

Remark 6.2. If the control signal constraints are linearly independent, it is actually possible to use block elimination when solving the dual problem. Lower and upper bounds can be handled by the introduction of a variable substitution. Although it has not been further examined, it might be possible to handle general linearly dependent control signal constraints by a similar procedure.

For an MPC problem with upper and lower bound constraints, the rows containing these constraints in the $A_{\mathcal{I}}$-matrix will be linearly dependent. Hence, it is not possible to use an algorithm that requires full row rank of $A_{\mathcal{I}}$. To be able to use the algorithm in the nodes of a branch and bound algorithm, where binary constraints are relaxed to interval constraints, the algorithm has to be able to handle upper and lower bounds on variables. For example, if an MPC problem with binary inputs is to be solved by branch and bound, the binary constraints $u(t) \in \{0, 1\}$ on the input variables are relaxed to $u(t) \in [0, 1]$.

From this discussion, it is clear that the KKT system cannot be solved as in [61]. Since it is generally not possible to compute K_{11}^{-1}, an efficient algorithm that either works on the

entire system (6.31) has to be developed or the system has to be partitioned in another way than proposed in (6.31). An alternative partitioning can be found by reordering the rows and columns of K in (6.31). For what follows, it it necessary to split $\tilde{\mathsf{u}}$ into w and v, where w contains the stacked unconstrained dual control signals and v contains the stacked constrained dual control signals. Relating back to (6.14), let $\bar{x}_1 = \begin{bmatrix}\tilde{\mathsf{x}}^T & \mathsf{w}^T & \lambda^T & \mathsf{v}_{\mathcal{I}\cap\mathcal{W}^\complement}^T\end{bmatrix}^T$ and $\bar{x}_2 = \begin{bmatrix}\mathsf{v}_{\mathcal{I}\cap\mathcal{W}}^T & \mu_{\mathcal{I}\cap\mathcal{W}}^T\end{bmatrix}^T$, where the subindex indicates whether the corresponding variable contains elements related to the constraints in the working set or not. The result after reordering rows and columns in K is a system on the form

$$\bar{K}\bar{x} = \left[\begin{array}{c:c} \bar{K}_{11} & \bar{K}_{12} \\ \hdashline \bar{K}_{21} & \bar{K}_{22} \end{array}\right] \begin{bmatrix}\bar{x}_1 \\ \bar{x}_2\end{bmatrix} = \left[\begin{array}{c:cc} \bar{R}_{11} & \bar{R}_{12} & 0 \\ \hdashline \bar{R}_{21} & \bar{R}_{22} & -I \\ 0 & -I & 0 \end{array}\right] \left[\begin{array}{c} \bar{x}_1 \\ \hdashline \bar{x}_{2,1} \\ \bar{x}_{2,2} \end{array}\right] = \left[\begin{array}{c} \bar{b}_1 \\ \hdashline \bar{b}_{2,1} \\ \bar{b}_{2,2} \end{array}\right] \tag{6.32}$$

Note that $\bar{K}_{22}$ is non-singular. In (6.32), it has been used that the inequality constraints have the form $-v(\tau) \leq 0$, which implies that the matrix containing the coefficient for $\mathsf{v}_{\mathcal{I}\cap\mathcal{W}}$ is $-I$. As mentioned in Section 2.4.2, dual variables not in the working set are *set* to zero. Therefore $\mu_{\mathcal{I}\cap\mathcal{W}^\complement} = 0$. Note that $\bar{b}_{2,2} = 0$. This follows from the fact that the right hand side of the equations for the inequality constraints in the working set is zero. By using a block inversion formula for the case $\bar{K}_{22}$ is non-singular, $\bar{x}_1$ and $\bar{x}_2$ can be calculated as

$$\begin{aligned} \bar{x}_1 &= \left(\bar{K}_{11} - \bar{K}_{12}\bar{K}_{22}^{-1}\bar{K}_{21}\right)^{-1}\left(\bar{b}_1 - \bar{K}_{12}\bar{K}_{22}^{-1}\bar{b}_2\right) \\ \bar{x}_2 &= \bar{K}_{22}^{-1}\left(\bar{b}_2 - \bar{K}_{21}\bar{x}_1\right) \end{aligned} \tag{6.33}$$

It follows directly from Lemma A.3 that

$$\bar{K}_{22}^{-1} = \begin{bmatrix} 0 & -I \\ -I & -\bar{R}_{22} \end{bmatrix} \tag{6.34}$$

Furthermore,

$$\bar{K}_{12}\bar{K}_{22}^{-1} = \begin{bmatrix}\bar{R}_{12} & 0\end{bmatrix}\begin{bmatrix} 0 & -I \\ -I & -\bar{R}_{22} \end{bmatrix} = \begin{bmatrix}0 & -\bar{R}_{12}\end{bmatrix} \tag{6.35}$$

From this the next result follows directly

$$\bar{K}_{12}\bar{K}_{22}^{-1}\bar{K}_{21} = \begin{bmatrix}0 & -\bar{R}_{12}\end{bmatrix}\begin{bmatrix}\bar{R}_{21} \\ 0\end{bmatrix} = 0 \tag{6.36}$$

Using (6.35) and (6.36) in the expression for $\bar{x}_1$ in (6.33), the following simplified equation for $\bar{x}_1$ is obtained

$$\bar{R}_{11}\bar{x}_1 = \bar{b}_1 + \bar{R}_{12}\bar{b}_{2,2} = \bar{b}_1 \tag{6.37}$$

where the last equality follows from $\bar{b}_{2,2} = 0$. By using (6.34) and that $\bar{b}_{2,2} = 0$, the expression for $\bar{x}_2$ can be simplified to

$$\bar{x}_2 = \begin{bmatrix} 0 \\ \bar{R}_{21}\bar{x}_1 - \bar{b}_{2,1} \end{bmatrix} \tag{6.38}$$

As a consequence, when $\bar{x}_1$ has been computed, $\mathsf{v}_{\mathcal{I}\cap\mathcal{W}}$ and $\mu_{\mathcal{I}\cap\mathcal{W}}$ can easily be computed as

$$\mathsf{v}_{\mathcal{I}\cap\mathcal{W}} = 0, \qquad \mu_{\mathcal{I}\cap\mathcal{W}} = \bar{R}_{21}\bar{x}_1 - \bar{b}_{2,1} \tag{6.39}$$

From a practical point of view, it is important to discuss the partitioning into $\bar{R}_{11}$, $\bar{R}_{12}$, $\bar{R}_{21}$ and $\bar{R}_{22}$ expressed in terms of the variables present in problem (6.1). The rows corresponding to

$$\begin{bmatrix} \bar{R}_{21} & \bar{R}_{22} & -I \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_{2,1} \\ \bar{x}_{2,2} \end{bmatrix} = \bar{b}_{2,1} \tag{6.40}$$

contain the equations in (6.21c) involving $\mu(\tau)$. Using (6.13), these equations can be rewritten as

$$\begin{aligned} &\underbrace{\begin{bmatrix} H_u(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma) & H_x(\varsigma) \end{bmatrix}}_{\text{Included in } \bar{R}_{21}} \begin{bmatrix} \tilde{x}(\tau) \\ \lambda(\tau+1) \end{bmatrix} + \underbrace{H_u(\varsigma)Q_u^{-1}H_u^T(\varsigma)}_{\text{Included in } \bar{R}_{22}} v(\tau) \\ &- I \cdot \mu(\tau) = h(\varsigma), \qquad \tau = 0, \ldots, N-2 \end{aligned} \tag{6.41}$$

where $\varsigma = N - \tau - 1$. For each component in $v(\tau)$ there is a constraint $v_i(\tau) \geq 0$ that can be either active or inactive. If the constraint is active, $v_i(\tau) = 0$. If $v_i(\tau)$ in (6.41) has a corresponding constraint which is active, the corresponding equation in (6.41) can be simplified since $v_i(\tau)$ can be removed from the equation. If a constraint is activated, $v_i(\tau)$ is set to zero and it does not have to be computed from (6.41), but on the other hand, the corresponding dual variable $\mu_i(\tau)$ is no longer set to zero and has to be calculated. This can also be done using (6.41). That is, the use of (6.41) varies with if the constraint corresponding to the v_i-element is active or inactive according to

$$\begin{aligned} &H_{u,(i,:)}(\varsigma)Q_u^{-1}H_{u,(i,:)}^T(\varsigma)v_i(\tau) + H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma)\tilde{x}(\tau) \\ &+ H_{x,(i,:)}(\varsigma)\lambda(\tau+1) = h_i(\varsigma) \end{aligned} \tag{6.42a}$$

$$\begin{aligned} &\mu_j(\tau) = H_{u,(j,:)}(\varsigma)Q_u^{-1}H_{u,(i,:)}^T(\varsigma)v_i(\tau) + H_{u,(j,:)}(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma)\tilde{x}(\tau) \\ &+ H_{x,(j,:)}(\varsigma)\lambda(\tau+1) - h_j(\varsigma) \end{aligned} \tag{6.42b}$$

where i denotes elements in $v(\tau)$ included in inequality constraints not in the working set and j denotes elements in $v(\tau)$ included in inequality constraints in the working set. Equations of type (6.42a) are solved as a part of the Riccati recursion described below and equations of type (6.42b) are solved after (6.42a) has been solved and are used to compute $\mu(\tau)$.

Left to solve is the equation $\bar{R}_{11}\bar{x}_1 = \bar{b}_1$. This equation corresponds to solving a modified version of the equations in (6.18) and (6.21). From (6.32), it follows that (6.21) should be modified by setting components in $\mathsf{v}_{\mathcal{I}\cap\mathcal{W}}$ and $\mu_{\mathcal{I}\cap\mathcal{W}}$ to zero. After a suitable

reordering of the variables and equations, (6.37) can be written as

$$\begin{bmatrix}
\tilde{Q}_{\tilde{u}}(-1) & \tilde{B}^T(-1) & 0 & 0 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\
\tilde{B}^T(-1) & 0 & -I & 0 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\
0 & -I & \tilde{Q}_{\tilde{x}}(0) & \tilde{Q}_{\tilde{x}\tilde{u}}(0) & \tilde{A}^T(0) & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\
0 & 0 & \tilde{Q}^T_{\tilde{x}\tilde{u}}(0) & \tilde{Q}_{\tilde{u}}(0) & \tilde{B}^T(0) & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\
0 & 0 & \tilde{A}(0) & \tilde{B}(0) & 0 & -I & 0 & \dots & \dots & \dots & \dots & 0 \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \tilde{A}(N-1) & \tilde{B}(N-1) & 0 & -I \\
0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0 & -I & 0
\end{bmatrix} \bar{x}'_1 = \bar{b}'_1 \quad (6.43)$$

where

$$\bar{x}'_1 = \begin{bmatrix} \tilde{u}(-1) \\ \lambda(0) \\ \tilde{x}(0) \\ \tilde{u}(0) \\ \lambda(1) \\ \tilde{x}(1) \\ \tilde{u}(1) \\ \vdots \\ \tilde{x}(N-1) \\ \tilde{u}(N-1) \\ \lambda(N) \\ \tilde{x}(N) \end{bmatrix}, \qquad \bar{b}'_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\tilde{q}_{\tilde{u}}(0) \\ 0 \\ \vdots \\ -\tilde{q}_{\tilde{u}}(N-1) \\ -\tilde{q}_{\tilde{x}}(N) \end{bmatrix} \quad (6.44)$$

and $\tilde{u}(\tau)$ includes all components of $w(\tau)$ but only the components in $v(\tau)$ not included in a constraint in the working set.

If (6.43) is studied, it is seen that it has an almost block diagonal structure. This structure is now going to be utilized. For each $\tau = 0, \dots, N-1$, the following relation holds

$$\begin{bmatrix} -I & \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) \\ 0 & \tilde{Q}^T_{\tilde{x}\tilde{u}}(\tau) & \tilde{Q}_{\tilde{u}}(\tau) \\ 0 & \tilde{A}(\tau) & \tilde{B}(\tau) \end{bmatrix} \begin{bmatrix} \lambda(\tau) \\ \tilde{x}(\tau) \\ \tilde{u}(\tau) \end{bmatrix} + \begin{bmatrix} \tilde{A}^T(\tau) & 0 \\ \tilde{B}^T(\tau) & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} \lambda(\tau+1) \\ \tilde{x}(\tau+1) \end{bmatrix} = \begin{bmatrix} 0 \\ -\tilde{q}_{\tilde{u}}(\tau) \\ 0 \end{bmatrix} \quad (6.45)$$

In order to represent all equations in (6.43), three more equations concerning variables at $\tau = -1$ and $\tau = N$ are needed

$$\begin{aligned} \tilde{u}(-1) &= -\tilde{Q}^{-1}_{\tilde{u}}(-1)\tilde{B}^T(-1)\lambda(0) \\ \tilde{x}(0) &= \tilde{B}(-1)\tilde{u}(-1) \\ \lambda(N) &= \tilde{q}_{\tilde{x}}(N) \end{aligned} \quad (6.46)$$

The aim is now to show that there exist symmetric positive semidefinite matrices $P(\tau)$ such that

$$P(\tau)\tilde{x}(\tau) - \lambda(\tau) = \Psi(\tau) \quad (6.47)$$

By comparing the first line in (6.46) with (6.47), it is clear that (6.47) holds at time instant $\tau = N$ with

$$\begin{aligned} P(N) &= 0 \\ \Psi(N) &= -\tilde{q}_{\tilde{x}}(N) \end{aligned} \quad (6.48)$$

Assume

$$P(\tau+1)\tilde{x}(\tau+1)-\lambda(\tau+1)=\Psi(\tau+1) \tag{6.49}$$

or equivalently, $\lambda(\tau+1)=P(\tau+1)\tilde{x}(\tau+1)-\Psi(\tau+1)$. Using this assumption, (6.45) can be reformulated as

$$\begin{aligned}
&\begin{bmatrix}-I & \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\\ 0 & \tilde{Q}^T_{\tilde{x}\tilde{u}}(\tau) & \tilde{Q}_{\tilde{u}}(\tau)\end{bmatrix}\begin{bmatrix}\lambda(\tau)\\ \tilde{x}(\tau)\\ \tilde{u}(\tau)\end{bmatrix}+\begin{bmatrix}\tilde{A}^T(\tau)\\ \tilde{B}^T(\tau)\end{bmatrix}\big(P(\tau+1)\tilde{x}(\tau+1)-\Psi(\tau+1)\big)\\
&=\begin{bmatrix}-I & \tilde{Q}_{\tilde{x}}(\tau)+\tilde{A}^T(\tau)P(\tau+1)\tilde{A}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau)+\tilde{A}^T(\tau)P(\tau+1)\tilde{B}(\tau)\\ 0 & \tilde{Q}^T_{\tilde{x}\tilde{u}}(\tau)+\tilde{B}^T(\tau)P(\tau+1)\tilde{A}(\tau) & \tilde{Q}_{\tilde{u}}(\tau)+\tilde{B}^T(\tau)P(\tau+1)\tilde{B}(\tau)\end{bmatrix}\begin{bmatrix}\lambda(\tau)\\ \tilde{x}(\tau)\\ \tilde{u}(\tau)\end{bmatrix}\\
&-\begin{bmatrix}\tilde{A}^T(\tau)\\ \tilde{B}^T(\tau)\end{bmatrix}\Psi(\tau+1)=\begin{bmatrix}0\\ -\tilde{q}_{\tilde{u}}(\tau)\end{bmatrix}
\end{aligned} \tag{6.50}$$

Define the following variables

$$\begin{aligned}
F(\tau+1)&=\tilde{Q}_{\tilde{x}}(\tau)+\tilde{A}^T(\tau)P(\tau+1)\tilde{A}(\tau)\\
G(\tau+1)&=\tilde{Q}_{\tilde{u}}(\tau)+\tilde{B}^T(\tau)P(\tau+1)\tilde{B}(\tau)\\
H(\tau+1)&=\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)+\tilde{A}^T(\tau)P(\tau+1)\tilde{B}(\tau)
\end{aligned} \tag{6.51}$$

Equation (6.50) can then be written on the simplified form

$$\begin{bmatrix}-I & F(\tau+1) & H(\tau+1)\\ 0 & H^T(\tau+1) & G(\tau+1)\end{bmatrix}\begin{bmatrix}\lambda(\tau)\\ \tilde{x}(\tau)\\ \tilde{u}(\tau)\end{bmatrix}-\begin{bmatrix}\tilde{A}^T(\tau)\\ \tilde{B}^T(\tau)\end{bmatrix}\Psi(\tau+1)=\begin{bmatrix}0\\ -\tilde{q}_{\tilde{u}}(\tau)\end{bmatrix} \tag{6.52}$$

Assume $G(\tau+1)$ non-singular. From the lower block in (6.52) an expression for $\tilde{u}(\tau)$ can be derived.

$$\tilde{u}(\tau)=G^{-1}(\tau+1)\big(\tilde{B}^T(\tau)\Psi(\tau+1)-H^T(\tau+1)\tilde{x}(\tau)-\tilde{q}_{\tilde{u}}(\tau)\big) \tag{6.53}$$

Using the first block in (6.52) and the expression for $\tilde{u}(\tau)$ an expression similar to (6.47) can be derived

$$\begin{aligned}
&-\lambda(\tau)+F(\tau+1)\tilde{x}(\tau)+H(\tau+1)\tilde{u}(\tau)-\tilde{A}^T(\tau)\Psi(\tau+1)\\
&=-\lambda(\tau)+\big(F(\tau+1)-H(\tau+1)G^{-1}(\tau+1)H^T(\tau+1)\big)\tilde{x}(\tau)\\
&+H(\tau+1)G^{-1}(\tau+1)\tilde{B}^T(\tau)\Psi(\tau+1)-H(\tau+1)G^{-1}(\tau+1)\tilde{q}_{\tilde{u}}(\tau)\\
&-\tilde{A}^T(\tau)\Psi(\tau+1)=0\Leftrightarrow\\
&\big(F(\tau+1)-H(\tau+1)G^{-1}(\tau+1)H^T(\tau+1)\big)\tilde{x}(\tau)-\lambda(\tau)\\
&=-H(\tau+1)G^{-1}(\tau+1)\tilde{B}^T(\tau)\Psi(\tau+1)+H(\tau+1)G^{-1}(\tau+1)\tilde{q}_{\tilde{u}}(\tau)\\
&+\tilde{A}^T(\tau)\Psi(\tau+1)
\end{aligned} \tag{6.54}$$

Identifying terms in (6.54) with (6.47), an expression for $P(\tau)$ and $\Psi(\tau)$ can be derived.

$$\begin{aligned}
P(\tau)&=F(\tau+1)-H(\tau+1)G^{-1}(\tau+1)H^T(\tau+1)\\
\Psi(\tau)&=\big(\tilde{A}^T(\tau)-H(\tau+1)G^{-1}(\tau+1)\tilde{B}^T(\tau)\big)\Psi(\tau+1)\\
&\quad+H(\tau+1)G^{-1}(\tau+1)\tilde{q}_{\tilde{u}}(\tau)
\end{aligned} \tag{6.55}$$

These are recursive expressions that hold for $\tau = N-1, \dots, 0$, with initial values defined in (6.48). After (6.51) has been inserted, the equation for $P(\tau)$ can be identified as a Riccati recursion. This is known to have a computational complexity that grows linearly with N, [95]. Note that $P(N) \succeq 0$. By investigating (6.51), and using the symmetric property of the weight matrices, it is clear that $P(\tau)$ is symmetric, that is, $P(\tau) = P^T(\tau)$. To show that $P(\tau) \succeq 0$, recognize the expression for $P(\tau)$ as the Schur complement of $G(\tau+1)$ in a matrix $X(\tau)$, defined below. If $G(\tau+1) \succ 0$, then, by Lemma A.1, $X \succeq 0$ if and only if $P(\tau) \succeq 0$ where

$$\begin{aligned} X(\tau) &= \begin{bmatrix} G(\tau+1) & H^T(\tau+1) \\ H(\tau+1) & F(\tau+1) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{Q}_{\tilde{u}}(\tau) + \tilde{B}^T(\tau)P(\tau+1)\tilde{B}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau) + \tilde{B}^T(\tau)P(\tau+1)\tilde{A}(\tau) \\ \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) + \tilde{A}^T(\tau)P(\tau+1)\tilde{B}(\tau) & \tilde{Q}_{\tilde{x}}(\tau) + \tilde{A}^T(\tau)P(\tau+1)\tilde{A}(\tau) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{Q}_{\tilde{u}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau) \\ \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) & \tilde{Q}_{\tilde{x}}(\tau) \end{bmatrix} + \begin{bmatrix} \tilde{B}^T(\tau) \\ \tilde{A}^T(\tau) \end{bmatrix} P(\tau+1) \begin{bmatrix} \tilde{B}(\tau) & \tilde{A}(\tau) \end{bmatrix} \end{aligned} \tag{6.56}$$

If $P(\tau+1) \succeq 0$, the entire last term in $X(\tau)$ is positive semidefinite since it is a quadratic expression. The conclusion following from induction is that if the following two conditions are satisfied for all τ

$$G(\tau+1) \succ 0 \tag{6.57a}$$

$$\begin{bmatrix} \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) \\ \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau) & \tilde{Q}_{\tilde{u}}(\tau) \end{bmatrix} \succeq 0 \tag{6.57b}$$

then $P(\tau) \succeq 0$ for $\tau = N, \dots, 0$ and therefore the Riccati recursion is concluded well-defined over the same interval. If the expression for $G(\tau+1)$ in (6.51) is investigated, a sufficient condition for (6.57a) to hold is that $\tilde{Q}_{\tilde{u}}(\tau) \succ 0$.

It is now interesting to interpret these conditions using the definitions of $\tilde{Q}_{\tilde{u}}(\tau)$ and $\tilde{B}(\tau)$ in (6.13). First, condition (6.57a) is investigated.

$$\begin{aligned} G(\tau+1) &= \tilde{Q}_{\tilde{u}}(\tau) + \tilde{B}^T(\tau)P(\tau+1)\tilde{B}(\tau) \\ &= \begin{bmatrix} Q_e^{-1}(\varsigma) & 0 \\ 0 & H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)H_{u,(i,:)}^T(\varsigma) \end{bmatrix} \\ &+ \begin{bmatrix} M(\varsigma) \\ H_{x,(i,:)}(\varsigma) \end{bmatrix} P(\tau+1) \begin{bmatrix} M^T(\varsigma) & H_{x,(i,:)}^T(\varsigma) \end{bmatrix} \succ 0 \end{aligned} \tag{6.58}$$

where $\varsigma = N - \tau - 1$ and i is defined as in the text below (6.42). Since the second term is positive semidefinite, a sufficient condition for the entire expression to be positive definite is that the first term in (6.58) is positive definite. This is satisfied when the assumptions A1, A2 and A3 are satisfied. Second, the condition (6.57b) has to be shown to hold. Inserting (6.13) gives the condition

$$\begin{aligned} &\begin{bmatrix} \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) \\ \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau) & \tilde{Q}_{\tilde{u}}(\tau) \end{bmatrix} \\ &= \begin{bmatrix} B(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma) & 0 & B(\varsigma)Q_u^{-1}(\varsigma)H_{u,(i,:)}^T(\varsigma) \\ 0 & Q_e^{-1}(\varsigma) & 0 \\ H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma) & 0 & H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)H_{u,(i,:)}^T(\varsigma) \end{bmatrix} \succeq 0 \end{aligned} \tag{6.59}$$

where i is defined as in the text below (6.42). Rearrange the rows and columns to obtain the equivalent condition

$$\begin{bmatrix} Q_e^{-1}(\varsigma) & 0 & 0 \\ 0 & B(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma) & B(\varsigma)Q_u^{-1}(\varsigma)H_{u,(i,:)}^T(\varsigma) \\ 0 & H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)B^T(\varsigma) & H_{u,(i,:)}(\varsigma)Q_u^{-1}(\varsigma)H_{u,(i,:)}^T(\varsigma) \end{bmatrix} \succeq 0 \tag{6.60}$$

Since $Q_e^{-1}(\varsigma) \succ 0$, it remains to investigate the definiteness of the blocks to the right and below.

$$\begin{bmatrix} B(\varsigma) \\ H_{u,(i,:)}(\varsigma) \end{bmatrix} Q_u^{-1}(\varsigma) \begin{bmatrix} B^T(\varsigma) & H_{u,(i,:)}^T(\varsigma) \end{bmatrix} \succeq 0 \tag{6.61}$$

Due to $Q_u^{-1}(\varsigma) \succ 0$, the block in (6.61) is always positive semidefinite. To be complete, the time instant -1 has to be considered separately. Using (6.13) and (6.46), in time instant $\tau = -1$ there is only one condition to fulfill

$$\tilde{Q}_{\tilde{u}}(-1) = Q_e^{-1}(N) \succ 0 \tag{6.62}$$

and there does not exist any $P(-1)$ to check for positive definiteness.

The conclusion from this section is that provided the assumptions A1, A2 and A3 are satisfied, the equation in (6.37) can be solved using a Riccati recursion. The procedure starts with the backward recursions in (6.55) for $P(\tau)$ and $\Psi(\tau)$, followed by forward recursions for $\tilde{x}(\tau)$, $\tilde{u}(\tau)$ and $\lambda(\tau)$, which are formed by the equations for the dynamics in (6.14) and the equations in (6.53) and (6.47), respectively. A more thorough presentation of these calculations for similar problems can be found in, for example, [95] and [105], where it is also shown that the computational complexity for these calculations grows linearly with the prediction horizon N. The primal variables are finally found according to the discussion in Section 6.1.4.

6.1.6 Handling Parallel Constraints

In Section 6.1.5, $H_u(t)$ was assumed having full row rank. In this section, Assumption A3 is to be slightly relaxed. There are situations where it is necessary to have constraints with linear dependent rows. An important case for the MPC problem is parallel constraints which occur when there are upper and lower bounds on states and control signals. The word parallel refers to the gradients of the constraints. To be precise, in this case these are anti-parallel. This type of constraints are especially important in the development of a solver for the node problems in a branch and bound algorithm. In the node problems, binary constraints are relaxed to interval constraints, which have to be solved by a solver handling parallel constraints.

Parallel constraints can be written as

$$\begin{aligned} \hat{H}_x(t)x(t) + \hat{H}_u(t)u(t) &\leq \hat{h}_+(t) \\ -\hat{H}_x(t)x(t) - \hat{H}_u(t)u(t) &\leq -\hat{h}_-(t) \end{aligned} \tag{6.63}$$

where it is assumed that there exist strictly feasible points, that is, $\hat{h}_+(t) > \hat{h}_-(t)$. Furthermore, to simplify the notation in the discussion below, it is assumed that there is only

one parallel constraint in each time instant, that is, $\hat{H}_x(t)$, $\hat{H}_u(t)$, $\hat{h}_+(t)$ and $\hat{h}_-(t)$ only contain one row. The constraints in (6.63) can be written on the form in (6.1) by making the following definitions

$$H_x(t) = \begin{bmatrix} \hat{H}_x(t) \\ -\hat{H}_x(t) \end{bmatrix}, \; H_u(t) = \begin{bmatrix} \hat{H}_u(t) \\ -\hat{H}_u(t) \end{bmatrix}, \; h(t) = \begin{bmatrix} -\hat{h}_+(t) \\ \hat{h}_-(t) \end{bmatrix} \tag{6.64}$$

It is important to be specific in *which* problems it is desirable for $H_u(t)$ to have full row rank. If the primal optimization problem has inequality constraints where $H_u(t)$ does not have full row rank, it might cause problems in the dual solver. In the derivation of the algorithm in the previous section, it was concluded that the rows of $H_u(t)$ that actually appeared in the Riccati recursion were those corresponding to inactive dual constraints. If a dual constraint is active, for example $v_i(\tau) = 0$, row i in $H_u(\tau)$ is not used in the Riccati recursion. Hence, there would not be a problem if there would exist another row $j \neq i$ in $H_u(\tau)$ which is linearly dependent with row i. What is important is that they do not appear *simultaneously* in the Riccati recursion. As will be shown, this can be guaranteed by proper initialization of the active set.

To investigate the problem, consider linear dependent constraints of the type in (6.63). Assume that the dual variables corresponding to these constraints are $v_+(N-t-1)$ and $v_-(N-t-1)$. As already discussed in the text below (6.41), if the constraint $v_+(N-t-1) = 0$ is in the working set, $\hat{H}_u(t)$ appears in an equation of the same type as in (6.42b) and not in an equation of type in (6.42a). If $v_+(N-t-1) = 0$ is not included in the working set, $\hat{H}_u(t)$ appears in an equation of the type (6.42a) and not in an equation of type (6.42b). Equations of the type (6.42a) are solved as a part of the Riccati recursion. It can be realized that if *neither* $v_+(N-t-1) = 0$ nor $v_-(N-t-1) = 0$ is present in the working set, *both* $\hat{H}_u(N-t-1)$ and $-\hat{H}_u(N-t-1)$ are included in $H_{u,(i,:)}(N-t-1)$ in (6.42a) and this will decrease the row rank of $H_{u,(i,:)}(N-t-1)$. Specifically, the row rank of $H_{u,(i,:)}(N-t-1)$ will not be full, which was a sufficient condition for the Riccati recursion in Section 6.1.5 to be well defined. It will now be shown that under the assumption $\hat{h}_+(t) > \hat{h}_-(t)$, this situation will never occur.

The two parallel constraints in (6.63), will produce two rows of the type (6.42a) as described in (6.64).

$$\begin{aligned} &\hat{H}_u Q_u^{-1} \hat{H}_u^T v_+(\tau) - \hat{H}_u Q_u^{-1} \hat{H}_u^T v_-(\tau) + \hat{H}_u Q_u^{-1} B^T \tilde{x}(\tau) + \hat{h}_+ + \hat{H}_x \lambda(\tau+1) \\ &= \mu_+(\tau) \end{aligned} \tag{6.65a}$$

$$\begin{aligned} &- \hat{H}_u Q_u^{-1} \hat{H}_u^T v_+(\tau) + \hat{H}_u Q_u^{-1} \hat{H}_u^T v_-(\tau) - \hat{H}_u Q_u^{-1} B^T \tilde{x}(\tau) - \hat{h}_- - \hat{H}_x \lambda(\tau+1) \\ &= \mu_-(\tau) \end{aligned} \tag{6.65b}$$

For simplicity, some of the time indices are omitted. By combining (6.65a) and (6.65b), the following inequality is found

$$\mu_+(\tau) - \hat{h}_+ = -\mu_-(\tau) - \hat{h}_- \;\Leftrightarrow\; \mu_+(\tau) + \mu_-(\tau) = \hat{h}_+ - \hat{h}_- > 0 \tag{6.66}$$

where the last inequality follows from the assumption that there exist strictly feasible points with respect to the upper and lower bounds. Consider a time instant with two primal parallel constraints, and two corresponding dual constraints. It is only possible to remove

one constraint from the working set in each QP-iteration. Therefore, if both constraints are to be removed from the working set, it has to be performed in two QP-iterations. In the first QP-iteration, any of the two constraints is removed. In the second QP-iteration, the decision to remove the remaining constraint has to be taken, which implies that the condition for constraint removal has to be satisfied for this constraint. According to Algorithm 2.1, this condition is that the corresponding dual variable μ should be strictly negative. Note that, here μ is the dual variable of the dual problem and the corresponding variable in Algorithm 2.1 is $\hat{\lambda}$. Assume $v_+(\tau) = 0$ is in the working set and $v_-(\tau) = 0$ is not in the working set and is therefore free. Then $\mu_-(\tau) = 0$. Only if $\mu_+(\tau) < 0$, it is possible that $v_+(\tau) = 0$ is removed from the working set. If the assumptions $\mu_-(\tau) = 0$ and $\mu_+(\tau) < 0$ are inserted into (6.66), the result is

$$\mu_+(\tau) + \mu_-(\tau) = \mu_+(\tau) + 0 < 0 \tag{6.67}$$

But this leads to a contradiction since (6.66) always holds. The result is similar if $v_-(\tau) = 0$ is chosen to belong to the working set. Hence, the desired result follows.

The conclusion is that if the algorithm is properly initialized, that is at least one of $v_+(\tau) = 0$ and $v_-(\tau) = 0$ belongs to the working set, then future changes of the working set will never remove both $v_+(\tau) = 0$ and $v_-(\tau) = 0$ from the working set. Hence, H_u will always have full row rank if all constraint gradients are linearly independent excluding the dependence between constraint couples forming parallel constraint as in (6.63). Thus, it is possible to replace Assumption A3 with the relaxed assumption Assumption A4.

Assumption A4. $H_u(t)$ has full row rank for $t = 0, \ldots, N-1$, excluding the linear dependence between pair of parallel constraints as in (6.63) with $\hat{h}_+(t) > \hat{h}_-(t)$.

6.1.7 Increasing Performance

In the active set algorithm, similar KKT systems are solved in subsequent iterations. As has been discussed earlier, to be able to take advantage of the similarities, it is common to use block inversion (see Algorithm A.1). This is possible when there is an invertible matrix which is unchanged between all iterations. In the problem considered here, the matrix block which is unchanged is singular and therefore block inversion cannot be used. Anyhow, for the method described in this chapter, there are other ways to reuse computations from previous iterations. In each iteration there is only one constraint added or removed from the working set. Seen from an MPC perspective, this means that the problem is changed only in one time instant at each iteration. Because the Riccati recursion runs backwards in time, only time instants before (in dual time τ) the time step when the constraint is added have to be recomputed. In general, if a constraint on a component of $\tilde{u}(\tau')$ is added or deleted from the working set, $P(\tau)$ has to be recomputed for $\tau = \tau', \ldots, 0$. The same holds for $\Psi(\tau)$. That is, the backward recursions can reuse old computations. Since, the forward recursions use information from the last step in the backward recursions, the forward recursions always have to be completely recalculated.

6.1.8 Future Extensions

There are several interesting and promising future extensions possible for this algorithm. In the algorithm presented, $H_u(t)$ is assumed having full row rank for all time steps. This assumption limits the use of the algorithm, since it is not possible to have pure state constraints, that is, constraints with $H_u(t) = 0$. Some successful preliminary tests have been performed for this case, but there are still some details to investigate before the algorithm can be considered to work also for this type of problems.

As been discussed previously in Section 2.4, classical active set methods, like Algorithm 2.1, do not allow rapid changes to the working set. This is a great drawback for problems with many active constraints at the optimum. One solution to this problem is to use a gradient projection algorithm, which can rapidly modify the working set. An interesting idea is to combine the tailored method from this section with the more rapid handling of the working set offered by the gradient projection algorithm. The idea of allowing rapid changes to the working set has actually been tested ad-hoc, without any theoretical considerations, in the current implementation of the algorithm. Preliminary results indicate that a very high performance increase can be expected.

In the examples, the dual algorithm has been initialized with all dual constraints active and the origin as the dual starting point. A more efficient initialization would be to first solve the unconstrained primal problem and then add *all* the violated constraints to the working set. Note that the unconstrained primal problem corresponds to, via the complementary slackness condition, the fully constrained dual problem. This strategy has been previously used by other authors with good results, [75]. A gradient projection algorithm would probably "automatically" behave in a similar manner.

6.1.9 Simulation Results

In this section, the algorithm is applied to the MPC problem to control a mass as described in Section 3.4.1. The setup is similar to the one used in Section 5.2.2 but with two important differences. First, since the algorithm presented in this section is only able to solve linear MPC problems, the binary control signal is omitted from the problem. Second, the magnitude of the real-valued control signal is constrained to be less than or equal to 9. This value has been chosen to ensure that a reasonable amount of the constraints are active at the optimal solution. For example, at the prediction horizon $N = 1500$, 601 constraints are active at the optimum. By using zero order hold sampling with sampling time $0.1\,\mathrm{s}$, the continuous time description is converted to a discrete-time description of the form (3.1) with

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{6.68}$$

The cost function used in this example is of the type described in Section 3.1, with

$$Q_e = 100,\ Q_u = 1 \tag{6.69}$$

The initial state is chosen to be

$$x_1(0) = 5 \text{ and } x_2(0) = 0 \tag{6.70}$$

and the reference signal for the position of the mass is chosen to be $r(t) = 10\sin(t)$. The problem has been solved for different prediction horizons and the computational time has

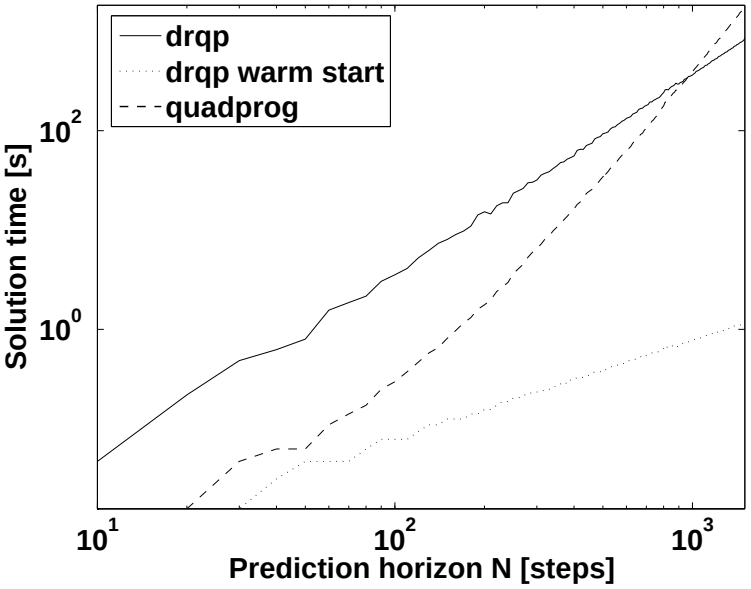


Figure 6.1: *This plot shows the computational times for two different QP solvers. The QP algorithm described in this section is implemented in the function* `drqp`*. The solid line shows the computational time when this algorithm solves the problem from scratch. The dotted line shows the computational time when a warm start is simulated when using* `drqp`*. The dashed line shows the computational time for the standard QP solver* `quadprog`*.*

been evaluated. The result from these tests are shown in Figure 6.1. In the tests, the QP algorithm presented in this chapter is used to solve the problem first from scratch and, second, given the optimal active set. This algorithm is implemented in the function `drqp`. In the latter test, the algorithm starts in the origin and it has to solve one QP subproblem before the optimal solution is found. This test is supposed to, at least roughly, simulate a warm start. As can be seen in Figure 6.1, `drqp` has significantly lower computational complexity compared to `quadprog` from the Optimization Toolbox in MATLAB. Considering prediction horizons from $N = 100$ to $N = 1500$, the computational complexity for `drqp` when cold started is in this test found to be approximately $\mathcal{O}(N^2)$. If the same algorithm is warm started, the computational complexity is reduced to approximately $\mathcal{O}(N)$. The latter result was expected since in the warm start case considered, only one Riccati recursion had to be computed and the Riccati recursion is known to have the computational complexity $\mathcal{O}(N)$, [96]. The MATLAB function `quadprog` is found to have an approximate computational complexity of $\mathcal{O}(N^{3.2})$. Therefore, the conclusion from this test is that the algorithm presented in this chapter has a significantly lower computational complexity compared to the generic algorithm `quadprog`. Also, warm starts are found to be very efficient. However, it should be emphasized that the solver was supplied with the optimal working set during the warm starts. In practice, at least a few QP iterations are expected necessary in most cases. When using `quadprog`, the MPC problem

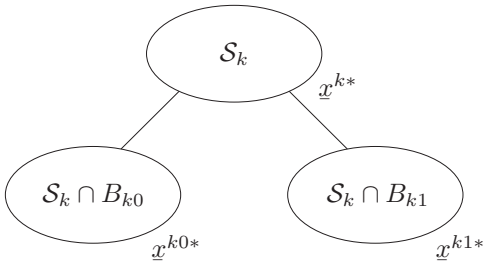


Figure 6.2: *This figure illustrates how the feasible set is changed after a branch.*

has been formulated as a QP problem using a dense optimization problem formulation as in (3.5). The test has been performed on a Intel Pentium 4 2.66 GHz with 512 Mb RAM running Microsoft Windows XP Professional Version 2002 Service Pack 2 and MATLAB 7.0.1 Service Pack 1. The computational time is calculated using the MATLAB command `cputime`.

6.2 A Mixed Integer Quadratic Programming Algorithm

The main reason for the development of the dual QP algorithm in Section 6.1, is the need for a solver that is easy to warm start. When solving an MIQP problem, one of the best methods to use is branch and bound. This method is thoroughly described in Section 2.5.2. In branch and bound, several similar subproblems are solved subsequently. Hence, it would be very useful to use the solution from one subproblem as a starting point in the next problem.

In the discussion that follows, the notation introduced in Section 2.5.2 is used. Consider a branching procedure for an arbitrary node N_k. When the node is branched, two new nodes N_{k0} and N_{k1} are created with

$$\begin{aligned} \mathcal{S}_{k0} &= \mathcal{S}_k \cap B_{k0} \\ \mathcal{S}_{k1} &= \mathcal{S}_k \cap B_{k1} \end{aligned} \tag{6.71}$$

where

$$B_{k0} = \{x | x_j = 0\}\,,\ B_{k1} = \{x | x_j = 1\} \tag{6.72}$$

The procedure is illustrated in Figure 6.2. Note that

$$\mathcal{S}_{k0} \cap \mathcal{S}_{k1} = \emptyset \tag{6.73}$$

and it is therefore clear that a solution to P_k is infeasible in, at least, one of the child problems. In the branch and bound method described in this text, the lower bounds computed by the QP relaxations in the nodes are found by relaxing the binary constraints to interval constraints. Let $\underline{x}^{k*}$ denote the optimizer to the relaxed problem P_k^R. Recapitulate from Section 2.5.2 that variables already branched in ancestor nodes are fixed to either 0 or 1 and these constraints are therefore not relaxed. That is, B_{k0} and B_{k1} cannot be relaxed.

To understand what happens when the equality constraints after a branch is introduced, three different relaxed problems are considered.

$$\begin{aligned} \underline{x}^{k*} &= \underset{x \in \mathcal{S}_k^R}{\operatorname{argmin}} \; f_0(x) \\ \underline{x}^{k0*} &= \underset{x \in \mathcal{S}_{k0}^R}{\operatorname{argmin}} \; f_0(x) \\ \underline{x}^{k1*} &= \underset{x \in \mathcal{S}_{k1}^R}{\operatorname{argmin}} \; f_0(x) \end{aligned} \tag{6.74}$$

Since the constraint differing $\mathcal{S}_{k0}^R$ from $\mathcal{S}_{k1}^R$ is not relaxed, (6.73) also holds for the relaxed sets.

$$\mathcal{S}_{k0}^R \cap \mathcal{S}_{k1}^R = \emptyset \tag{6.75}$$

Therefore, a solution $\underline{x}^{k*}$ to P_k^R is at most a feasible solution to one of the problems P_{k0}^R or P_{k1}^R. Since x_j is relaxed in P_k^R, it is very likely that $\underline{x}_j^{k*} \in]0,1[$. Only an integer solution, that is $\underline{x}_j^{k*} \in \{0,1\}$, would be feasible in one the child nodes. All other relaxed variables but j in P_k^R are still relaxed in P_{k0}^R and P_{k1}^R. The conclusion from this discussion is that it is very likely that $\underline{x}^{k*}$ cannot be used as a feasible starting point in either P_{k0}^R or P_{k1}^R. Therefore, it would be an advantage if a feasible starting point always could be easily found.

If the solution of the parent problem should be of any use in the child problem, their solutions should not differ too much. The natural question that arises is how much is "too much". At first, one interpretation of "too much" could be if the solution of the parent problem is not feasible in the child problem. But, according to the discussion above, this is very common in branch bound and there is not much to do about it. If an active set solver is used, the amount of work necessary to reach optimality is highly coupled to the number of changes necessary to the working set until optimality is reached. Therefore, a reasonable interpretation of "too much" when working with active set solvers is when the solution has changed so much that large parts of the working set has to be changed before optimality is reached. Especially, it would be very useful if the old active set cannot become infeasible when the new equality constraint is introduced.

By using a dual solver for the subproblems, a straightforward reuse of an old working set is enabled and the problem of choosing a feasible initial point is solved. This is now motivated by considering the primal problem (2.27) and its dual (2.33). In the dual solver, the problem equivalent to the dual (2.34) is actually solved. This problem is called the dual problem in this discussion. The subproblems to be solved in the nodes of the branch and bound tree will be of the type (2.34). Since ν is the Lagrange multiplier vector corresponding to the equality constraints, if one equality constraint is added to the primal problem, the number of elements in ν is increased by one. The interpretation in the dual MPC problem (6.14) is that an extra unconstrained input signal component is added. Similarly, if an extra inequality constraint is added to (2.27), a new sign-constrained variable is added to the dual problem (2.34). The interpretation in (6.14) is that a new sign-constrained control signal is added. The conclusion is that when new constraints are added to the primal problem, the dimension of the dual problem increases. Hence, a feasible dual solution is also a feasible dual solution when a constraint is added to the primal. If the added constraint is an inequality constraint, the new dual variable has

to be chosen non-negative. Because of the simple structure of the inequality constraints in the dual problem, given a feasible solution, a new feasible solution can easily be found when a constraint is added to the primal problem by simply keeping the old feasible solution and setting the new variable to zero. If no previous feasible solution is known, in the general case given in (2.34), the linear system of equations constituting the equality constraints has to be solved or a Phase I problem has to be solved. In the MPC case, the equality constraints in the dual (6.14) form the dual dynamics. Given a control signal and an initial condition, it is well-known from control theory that these equations can be used to calculate the states over the prediction horizon. Since the initial state $\tilde{x}(-1)$ in (6.14) can be interpreted as 0, a feasible solution to the equality constraints is $\tilde{\mathrm{x}} = \tilde{\mathrm{u}} = 0$. As a bonus, this solution also fulfills any non-negativity constraints on $\tilde{\mathrm{u}}$.

When a variable x_j is branched in a branch and bound method, an equality constraint $x_j = 0$ or $x_j = 1$ is added to the problem. However, since there already exist inequality constraints $x_j \geq 0$ and $x_j \leq 1$ in the parent problem, an alternative interpretation is that, for example, the inequality constraint $x_j \geq 0$ is converted into an equality constraint $x_j = 0$. This results in a dual problem similar to the previous problem, with the difference that the non-negativity constraint on the corresponding dual variable has been removed. The conclusion is again, if the primal problem is constrained, the dual problem is relaxed. If an equality constraint is added to the primal problem making the primal infeasible, the objective function value becomes $+\infty$. In the dual problem, this results in that a constraint is removed which makes the dual objective function unbounded from above (if the true dual problem (2.33) is considered, the equivalent problem (2.34) becomes unbounded from below), see also Section 2.4.1.

Since the interval $[0, 1]$ always contains interior points, both $x_j \geq 0$ and $x_j \leq 1$ cannot be active simultaneously. In branch and bound, when, for example, the inequality constraint $x_j \geq 0$ is converted into an equality constraint $x_j = 0$ it is obvious that the constraint $x_j \leq 1$ is redundant and it can therefore be removed from the problem. This means that the corresponding dual variable is also removed from the problem. Another approach is to keep the constraint, but since it never becomes active, the corresponding dual variable is fixed to zero. The first approach has better performance since variables are removed, while the latter approach is faster to implement since the problem is changed less between the nodes.

When variable x_j is branched in subproblem P_k, two subproblems P_{k0} and P_{k1} are created. In P_{k0}^R the constraint $x_j \geq 0$ from P_k^R is converted into $x_j = 0$ and in P_{k1}^R the constraint $x_j \leq 1$ is converted into $x_j = 1$. Since the optimal working set is passed from the parent problem to the child problems, one way of easily performing this constraint conversion is by deactivating the corresponding dual inequality constraint and setting a flag indicating that this constraint is now an equality constraint and should therefore never be included in the working set. If this approach is chosen, the constraint for the dual variable corresponding to the other parallel constraint has to be added to the working set. Otherwise, none of the dual constraints corresponding to the parallel primal constraints are active and the problems discussed in Section 6.1.6 will occur.

The conclusion is that a dual solver for the relaxed problems in the nodes will make warm starts easy. Given the old optimal working set, hopefully not so many QP iterations have to be performed until the optimal working set of the new problem is found. Relating back to what was previously discussed, the optimal solution to the child problems are

expected not to differ "too much" from the solution to the parent problem. However, this is problem dependent and the dual algorithm presented in this thesis will not change this fact.

6.2.1 Increasing Performance

The ideas presented in Section 6.1.7 can also be used in the MIQP solver. Since the child problems are only changed in a single time step compared to the parent problem, the backward recursions need only to be recalculated for the time step the constraint is added and backwards. The disadvantage of using this idea at the branch and bound level is that if a branch strategy is used where a branch can be "suspended" before it is pruned, it might occur that several non-pruned subtrees exist and therefore a great amount of data might have to be stored for the recalculations. If the storage space is small, a compromise between storing all recomputation data and storing no data at all is to store the data from the "most promising" nodes.

6.2.2 Future Extensions

Probably, the most effective way of increasing the performance of the algorithm would be to design some kind of preprocessing algorithm similar to the one presented in Chapter 5. No matter how efficiently the node problems can be solved, the number of nodes to explore seems to explode as the time horizon grows. It is therefore very important to cut away uninteresting sequences of binary variables at an early stage.

In the current implementation, the subproblems are created from the parent problem by only changing the active set and using flags to indicate whether a primal inequality constraint has been converted to a primal equality constraint. As discussed previously, the alternative is to explicitly rewrite the problem and explicitly remove control signals in the dual. This alternative is expected to give, at least slightly, better performance.

When using a dual feasible solver, the inequality (2.13) can be used to estimate the optimal objective function value. Every dual feasible solution will give a lower bound on the optimal objective function value. It is straightforward to use this bound to prematurely abort the solution of a subproblem as soon as a dual feasible solution is known to the current subproblem and this solution gives an optimal objective function value worse than the best known upper bound in the branch and bound tree.

6.2.3 Simulation Results

In this section, the algorithm is applied to the MPC problem to control the satellite described in Section 3.4.2. The setup is similar to the one used in Section 5.2.2, but in this example, the magnitude of the real-valued control signal is limited to be less than or equal to 1. As in Section 5.2.2, the states of the system are chosen to be the satellite attitude x_1, the satellite angular velocity x_2 and the internal wheel velocity x_3. To obtain a system description on the MLD form (3.6), zero order hold sampling with the sampling time $0.1\,\mathrm{s}$ has been used. The cost function used in this example is of the type described in Section 3.10, with

$$Q_e = \mathrm{diag}\left(0.5 \cdot 10^4, 10^{-2}, 10^{-1}\right),\ Q_{u_c} = 10,\ Q_{u_b} = 10 \cdot I_2 \tag{6.76}$$

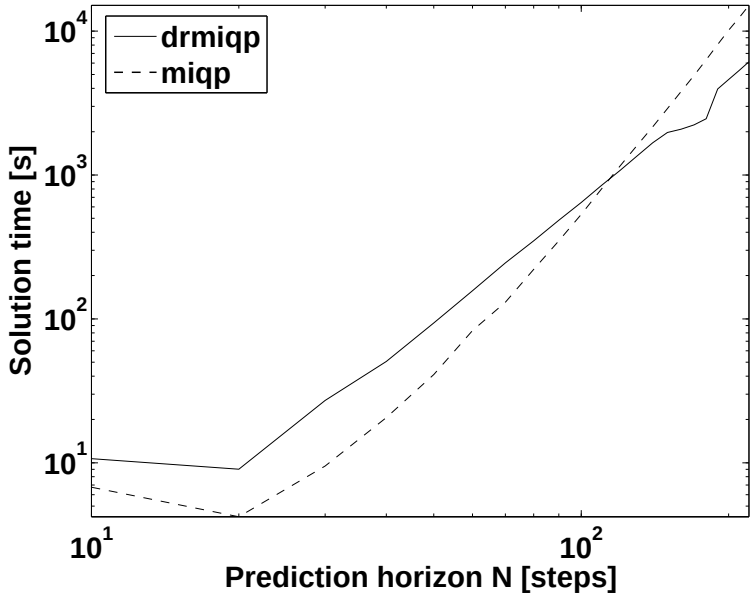


Figure 6.3: *This plot shows the computational times for two different MIQP solvers. The MIQP algorithm described in this chapter is implemented in the function* `drmiqp`*. The solid line shows the computational time when this algorithm is used. The dashed line shows the computational time for the standard MIQP solver* `miqp`*.*

The initial state is

$$x_1(0) = 0, x_2(0) = 0 \text{ and } x_3(0) = 0 \tag{6.77}$$

The reference signal is chosen as

$$r(t) = \begin{bmatrix} \theta_r(t) \\ 0 \\ 0 \end{bmatrix} \tag{6.78}$$

where

$$\theta_r(t) = \begin{cases} 0, \ t \leq \lfloor \frac{N}{4} \rfloor \\ 0.5, \ t > \lfloor \frac{N}{4} \rfloor \end{cases} \tag{6.79}$$

that is, a step in the satellite attitude of 0.5 radians is given when one fourth of the prediction horizon has elapsed. The problem has been solved for several different prediction horizons in the range $N = 10$ to $N = 220$ and the corresponding computational times are presented in Figure 6.3. In this example, for prediction horizons longer than 20 time steps, the algorithm presented in this section has an approximate computational complexity of $\mathcal{O}(N^{2.7})$, while the standard function `miqp`, using the QP solver `quadprog`, has an approximate computational complexity of $\mathcal{O}(N^{3.4})$. The MIQP algorithm presented in this chapter is implemented in the function `drmiqp`. The implementation of the branch and bound algorithm in `drmiqp` has been based on the code in `miqp`, which has been

modified in order to be able to use the dual algorithm previously presented and to enable the use of warm starts. Hence, it is exactly the same branch and bound code used in the results for `drqp` and `miqp`, and therefore, the branch and bound tree has been explored in exactly the same way. When using `miqp`, the MPC problem has been formulated as an MIQP problem using a dense optimization problem formulation similar to the one in (3.5). The test has been performed on a Intel Pentium 4 2.66 GHz with 512 Mb RAM running Microsoft Windows XP Professional Version 2002 Service Pack 2 and MATLAB 7.0.1 Service Pack 1. The computational time is calculated using the MATLAB command `cputime`.

7

Concluding Remarks

In this thesis, two approaches for how to more efficiently compute the optimal solution to Quadratic Programming (QP) problems involving binary variables in control and communication have been presented. The aim has not been to design a complete state-of-the-art solver, since this is an extremely complex task which involves several parts that each of them can be considered as a research area. Instead the aim has been to present some new approaches which can improve the performance of a specific part of the solver. This improvement has been focused on utilizing the specific problem structure arising from the underlying Model Predictive Control (MPC) problem, even though some results are more generally applicable.

7.1 Conclusions

The work presented in this thesis has generated two algorithms applicable to MPC. The first algorithm is also applicable to the Multiuser Detection (MUD) problem.

The first algorithm presented in this thesis is a preprocessing algorithm for Binary Quadratic Programming (BQP) problems having large diagonal terms compared to the non-diagonal terms. These problems are generally known to have exponential computational complexity in the number of variables. Simulations have shown that MPC tends to generate optimization problems where the algorithm can be successfully applied. In one example, the computational time was reduced with a factor of 275.

The preprocessing algorithm has also been successfully applied to the MUD problem for synchronous Code Division Multiple Access (CDMA). After the optimal MUD problem has been formulated as a BQP problem, the preprocessing algorithm has been applied to compute as many variables as possible in the problem. In simulations with up to 127 users, more than 99.95 % of the variables were found in the preprocessing step. The Bit Error Rate (BER) for the proposed algorithm was compared to the conventional detector

and the decorrelating detector. Simulations have shown that the BER can be significantly reduced by using the preprocessing algorithm for systems with high loads. The computational complexity is found to be higher than the conventional detector but approximately the same as the decorrelating detector.

The second algorithm presented in this thesis is a solver for Mixed Integer Quadratic Programming (MIQP) problems. The algorithm is built on a branch and bound framework, where different QP relaxations of the original QP problem are solved in the nodes. The contribution in this thesis is a dual QP solver tailored for MPC. By using Riccati recursions, the computational complexity of an internal QP iteration is reduced from approximately between $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ for an algorithm not utilizing structure to $\mathcal{O}(N)$, where N is the prediction horizon. In simulations, the overall complexity has been found to be lower than the overall complexity for a generic primal solver. Because the solver works in the dual space, warm starts can easily be used. This property is used when the algorithm is employed for solving the node problems in a branch and bound method. In simulations, it is shown how the algorithm decreases the practical complexity when solving an MIQP problem arising in MPC involving binary variables.

A summarizing conclusion from all work presented in this thesis is that there is much to gain from utilizing problem specific structure when solving problems involving binary variables, both in control and in communication.

7.2 Future Work

The area of integer optimization algorithms tailored for control and communications is far from being completely explored. On the contrary, the work presented in this thesis, and by many other authors, has just opened the door to an interesting and important research area. The importance of this research grows continuously, since the interaction between classical physical systems and computers is becoming more and more sophisticated. There are several future extensions possible regarding the algorithms presented in this thesis. Some of the more important are:

- The properties of the preprocessing algorithm could be further investigated.
- The preprocessing algorithm could be tested for a MUD problem when a more sophisticated model of the channel is taken into account.
- The use of the preprocessing algorithm when constraints are present remains to be explored. For example, it might be possible to use it to generate bounds on the optimal objective function value in a branch and bound method and to supply the method with a good initial guess of the optimal solution.
- The dual active set solver could possibly be extended to handle pure state constraints.
- In this thesis, the dual active set solver has been based on a classical active set method, where only one constraint is allowed to be added or removed from the working set in each internal QP iteration. An alternative approach is to try to incorporate the ideas presented in this thesis in a gradient projection algorithm, where more rapid changes to the active set are allowed.

- The fact that the subproblems in the branch and bound method are solved by a dual feasible solver has not been fully utilized in this thesis. Every dual feasible point can be used to generate a lower bound on the optimal objective function value of the current subproblem, which in some cases can be used to abort the solution of a subproblem prematurely.

Bibliography

[1] J. Albuquerque, V. Gopal, G. Staus, L. T. Biegler, and B. E. Ydstie. Interior point SQP strategies for large-scale, structured process optimization problems. *Computers and Chemical Engineering*, 23:543 – 554, 1999.

[2] E. Arnold and H. Puta. An SQP-type solution method for constrained discrete-time optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 127 – 136. Birkhäuser Verlag, 1994.

[3] D. Axehill and A. Hansson. A preprocessing algorithm for MIQP solvers with applications to MPC. In *Proceedings of the 43th IEEE Conference on Decision and Control*, pages 2497–2502, Atlantis, Paradise Island, Bahamas, Dec. 2004.

[4] D. Axehill and A. Hansson. A preprocessing algorithm for MIQP solvers with applications to MPC. In *Proceedings of Reglermöte 2004*, Göteborg, Sweden, May 2004.

[5] D. Axehill and J. Sjöberg. Adaptive cruise control for heavy vehicles - hybrid control and MPC. Master's thesis, Linköpings universitet, Feb. 2003.

[6] D. Axehill, F. Gunnarsson, and A. Hansson. A preprocessing algorithm applicable to the multiuser detection problem. In *Proceedings of RadioVetenskap och Kommunikation*, Linköping, Sweden, June 2005.

[7] R. A. Bartlett and L. T. Biegler. A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. Technical report, Department of Chemical Engineering, Carnegie Mellon University, 2004.

[8] R. A. Bartlett, A. Wächter, and L. T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, June 2000.

[9] R. A. Bartlett, L. T. Biegler, J. Backstrom, and V. Gopal. Quadratic programming algorithms for large-scale model predictive control. *Journal of Process Control*, 12: 775 – 795, 2002.

[10] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming.* John Wiley & Sons, Inc., 1979.

[11] J. E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, Management School, Imperial College, UK, Dec. 1998.

[12] A. G. Beccuti, T. Geyer, and M. Morari. Temporal lagrangian decomposition of model predictive control for hybrid systems. In *Proceedings of the 43th IEEE Conference on Decision and Control*, pages 2509 – 2514, Dec. 2004.

[13] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Transactions on Audio and Electroacoustics*, 49(5):832 – 838, May 2004.

[14] A. Bemporad and N. Giorgetti. A logic-based hybrid solver for optimal control of hybrid systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 640 – 645, Dec. 2003.

[15] A. Bemporad and D. Mignone. A Matlab function for solving mixed integer quadratic programs version 1.02 user guide. Technical report, Institut für Automatik, ETH, 2000.

[16] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407 – 427, 1999.

[17] A. Bemporad, D. Mignone, and M. Morari. An efficient branch and bound algorithm for state estimation and control of hybrid systems. In *Proceedings of the European Control Conference*, Aug. 1999.

[18] A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proceedings of the American Control Conference*, June 1999.

[19] A. Bemporad, F. Borelli, and M. Morari. Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Dec. 2000.

[20] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proceedings of the American Control Conference*, volume 2, pages 1190 – 1194, 2000.

[21] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controlability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10), Oct. 2000.

[22] A. Bemporad, L. Giovanardi, and F. Torrisi. Performance driven reachability analysis for optimal scheduling and control of hybrid systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 969 – 974, Dec. 2000.

[23] A. Bemporad, J. Roll, and L. Ljung. Identification of hybrid systems via mixed-integer programming. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 786 – 792, Dec. 2001.

[24] A. Bemporad, F. Borrelli, and M. Morari. On the optimal control law for linear discrete time hybrid systems. In *Lecture Notes in Computer Science*, volume 2289, pages 105 – 119, Mar. 2002.

[25] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. *IEEE Transactions on Automatic Control*, 47 (12):1974 – 1985, Dec. 2002.

[26] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3 – 20, 2002.

[27] M. Bergounioux, K. Ito, and K. Kunisch. Primal-dual strategy for constrained optimal control problems. *SIAM Journal on Control and Optimization*, 37(4):1176 – 1194, 1999.

[28] M. Bergounioux, M. Haddou, M. Hintermüller, and K. Kunisch. A comparison of a moreau-yosida-based active set strategy and interior point methods for constrained optimal control problems. *SIAM Journal on Optimization*, 11(2):495 – 521, 2000.

[29] L. T. Biegler. Advances in nonlinear programming concepts for process control. *Journal of Process Control*, 8(5 – 6):301 – 311, Oct. – Dec. 1998.

[30] J. Blomvall. *Optimization of Financial Decisions using a new Stochastic Programming Method.* PhD thesis, Linköpings universitet, 2001.

[31] N. L. Boland. A dual-active-set algorithm for positive semi-definite quadratic programming. *Mathematical Programming*, 78:1 – 27, 1997.

[32] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. In *Proceedings of the 2003 American Control Conference*, volume 6, pages 4717 – 4722, June 2003.

[33] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41:1709 – 1721, 2005.

[34] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[35] C. R. Cutler and B. L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings of the AIChE National Meeting*, Houston, Texas, Apr. 1979.

[36] B. De Schutter and T. J. J. van den Boom. MPC for continuous piecewise-affine systems. *Systems & Control Letters*, 52:179 – 192, 2004.

[37] W. S. Dorn. A symmetric dual theorem for quadratic programs. *Journal of the Operations Research Society of Japan*, 2(3):93 – 97, Jan. 1960.

[38] W. S. Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155 – 162, 1960.

[39] W. S. Dorn. Self-dual quadratic programs. *Journal of the Society for Industrial and Applied Mathematics*, 9(1):51 – 54, Mar. 1961.

[40] V. Dua, N. A. Bozinis, and E. N. Pistikopoulos. A multiparametric programming approach for mixed-integer quadratic engineering problems. *Computers and Chemical Engineering*, 26:715 – 733, 2002.

[41] G. Ferrari-Trecate, D. Mignone, D. Castagnoli, and M. Morari. Hybrid modeling and control of a hydroelectric power plant. Technical report, Institut für Automatik, 2000.

[42] G. Ferrari-Trecate, D. Mignone, D. Castagnoli, and M. Morari. Mixed logical dynamical model of a hydroelectric power plant. In *Proceedings of the 4th International Conference Automation of Mixed Processes: Hybrid Dynamic Systems*, 2000.

[43] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons Ltd., 1987.

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

[45] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization.* Oxford University Press, 1995.

[46] M. Garey and D. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness.* Freeman, 1979.

[47] F. Glover, B. Alidaee, C. Rego, and G. Kochenberger. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research*, 137(2):272–287, Mar. 2002.

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

[49] G. C. Goodwin, M. M. Seron, and J. A. De Doná. *Constrained Control and Estimation – An Optimisation Approach.* Springer Verlag, 2005.

[50] V. Gopal and L. T. Biegler. Large scale inequality constrained optimization and control. *IEEE Control Systems Magazine*, 18(6):59 – 68, Dec. 1998.

[51] N. I. M. Gould and P. L. Toint. A quadratic programming bibliography. Technical report, Numerical Analysis Group, Rutherford Appleton Laboratory, Feb. 2001.

[52] A. Grancharova and T. A. Johansen. Survey of explicit approaches to constrained optimal control. In *Switching and Learning in Feedback Systems. European Summer School on Multi-Agent Control. Revised Lectures and Selected Papers. (Lecture Notes in Computer Science Vol. 3355)*, Sept. 2003.

[53] A. Grancharova, T. A. Johansen, J. Kocijan, and D. Vrančić. Design of reduced dimension explicit model predictive controller for a gas-liquid separation plant. In *Proceedings of IEEE Region 8 EUROCON 2003. Computer as a Tool.*, 2003.

[54] A. Grancharova, T. A. Johansen, and J. Kocijan. Explicit model predictive control of gas-liquid separation plant via orthogonal search tree partitioning. *Computers and Chemical Engineering*, 28:2481 – 2491, 2004.

[55] A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Transactions on Automatic Control*, 45(9):1639 – 1655, Sept. 2000.

[56] F. Hasegawa, J. Luo, K. Pattipati, and P. Willett. Speed and accuracy comparison of techniques to solve a binary quadratic programming problem with applications to synchronous CDMA. In *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, 2001.

[57] G. He, A. Kot, and T. Qi. Decorrelator-based neighbour-searching multiuser detection in CDMA systems. *Electronics Letters*, 32(25), Dec. 1996.

[58] W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37:1085 – 1091, July 2001.

[59] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 364 – 369, Dec. 2001.

[60] M. Ishikawa, T. Seki, J.-i. Imura, and S. Hara. An efficient algorithm for optimal control of hybrid dynamical systems utilizing mode transition rule. In *Proceedings of the 43th IEEE Conference on Decision and Control*, pages 1866 – 1871, Dec. 2004.

[61] H. Jonson. *A Newton method for solving non-linear optimal control problems with general constraints.* PhD thesis, Linköpings Tekniska Högskola, 1983.

[62] K. Katayama and H. Narihisa. Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, 134(1):103–119, Oct. 2001.

[63] K. Kunisch and F. Rendl. An infeasible active set method for quadratic problems with simple bounds. *SIAM Journal on Optimization*, 14(1):35 – 52, 2003.

[64] L. S. Lasdon. *Optimization Theory for Large Systems.* MacMillan Publishing Co., Inc, 1970.

[65] C. E. Lemke. A method of solution for quadratic programs. *Management Science*, 8(4):442 – 453, July 1962.

[66] Y.-L. Li and Y. Lee. A novel low-complexity near-ML multiuser detector for DS-CDMA and MC-CDMA systems. In *Proceedings of GLOBECOM'02 - IEEE Global Telecommunications Conference*, volume 1, pages 493–498, Nov. 2002.

[67] B. Lie, M. D. Díez, and T. A. Hauge. A comparison of implementation strategies for MPC. *Modeling, identification and control*, 26(1):39 – 50, Jan. 2005.

[68] T. J. Lim, L. K. Rasmussen, and H. Sugimoto. An asynchronous multiuser CDMA detector based on the Kalman filter. *IEEE Journal on Selected Areas in Communications*, 16(9), Dec. 1998.

[69] J. Luo, K. R. Pattipati, P. Willet, and G. M. Levchuk. Fast optimal and suboptimal any-time algorithms for CDMA multiuser detection based on branch and bound. *IEEE Transactions on Communications*, 52(4), Apr. 2004.

[70] J. M. Maciejowski. *Predictive Control with Constraints.* Pearson Education Limited, 2002.

[71] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789 – 814, 2000.

[72] P. Merz and B. Freisleben. Greedy and local search heuristics for unconstrained binary quadratic programming. *Journal of Heuristics*, 8(2):197–213, Mar. 2002.

[73] D. Mignone, A. Bemporad, and M. Morari. A framework for control, fault detection, state estimation, and verification of hybrid systems. In *Proceedings of the American Control Conference*, June 1999.

[74] R. Milman and E. J. Davison. A proposed new non-feasible active set method with applications to model predictive control. Technical Report 0201, Department of Electrical and Computer Engineering, University of Toronto, 2002.

[75] R. Milman and E. J. Davison. Evaluation of a new algorithm for model predictive control based on non-feasible search directions using premature termination. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2216 – 2221, Dec. 2003.

[76] R. Milman and E. J. Davison. Fast computation of the quadratic programming subproblem in model predictive control. In *Proceedings of the American Control Conference*, pages 4723 – 4729, June 2003.

[77] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Verlag, 1999.

[78] M. J. D. Powell. On the quadratic programming algorithm of Goldfarb and Idnani. *Mathematical Programming Study*, 25:46 – 61, 1985.

[79] D. E. Quevedo, J. A. De Doná, and G. C. Goodwin. Receding horizon linear quadratic control with finite input constraint sets. In *Proceedings of the 15th IFAC World Congress*, July 2002.

[80] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná. Finite constraint set receding horizon quadratic control. *International Journal of Robust and Nonlinear Control*, 14:355 – 377, 2004.

[81] C. V. Rao. *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems.* PhD thesis, University of Wisconsin-Madison, 2000.

[82] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. Preprint ANL/MCS-P664-0597, Mathematics and Computer Science Division, Argonne National Laboratory, May 1997.

[83] C. Sankaran and A. Ephremides. Solving a class of optimum multiuser detection problems with polynomial complexity. *IEEE Transactions on Information Theory*, 44(5), Sept. 1998.

[84] M. W. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 1994.

[85] C. Schlegel and A. Grant. Polynomial complexity optimal detection of certain multiple-access systems. *IEEE Transactions on Information Theory*, 46(6), Sept. 2000.

[86] M. C. Steinbach. Structured interior point SQP methods in optimal control. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76:59 – 62, 1996.

[87] T. Stoilov and K. Stoilova. *Noniterative Coordination in Multilevel Systems.* Kluwer Academic Publishers, 1999.

[88] P. H. Tan. *Multiuser Detection in CDMA — Combinatorial Optimization Methods.* Licentiate's thesis, Chalmers University of Technology, Nov. 2001.

[89] J. Thomas, D. Dumur, and J. Buisson. Predictive control of hybrid systems under a multi-MLD formalism with state space polyhedral partition. In *Proceedings of the 2004 American Control Conference*, pages 2516 – 2521, June 2004.

[90] P. Tøndel, T. A. Johansen, and A. Bemporad. Computation and approximation of piecewise affine control laws via binary search trees. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Dec. 2002.

[91] P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39:489 – 497, 2003.

[92] F. D. Torrisi and A. Bemporad. HYSDEL – a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2):235 – 249, Mar. 2004.

[93] S. Ulukus and R. D. Yates. Optimum multiuser detection is tractable for synchronous CDMA systems using M-sequences. *IEEE Communications Letters*, 2 (4), Apr. 1998.

[94] C. van de Panne and A. Whinston. The simplex and the dual method for quadratic programming. *Operational Research Quarterly*, 15(4):355 – 388, 1964.

[95] L. Vandenberghe, S. Boyd, and M. Nouralishahi. Robust linear programming and optimal control. Technical report, Department of Electrical Engineering, University of California Los Angeles, 2002.

[96] L. Vandenberghe, S. Boyd, and M. Nouralishahi. Robust linear programming and optimal control. In *Proceedings of IFAC 2002 World Congress*, 2002.

[97] S. Verdu. *Multiuser Detection.* Cambridge University Press, 1998.

[98] O. V. Volkovich, V. A. Roshchin, and I. V. Sergienko. Models and methods of solution of quadratic integer programming problems. *Cybernetics*, 23:289 – 305, 1987.

[99] A. Wills and W. P. Heath. Interior-point methods of linear model predictive control. Technical report, Centre for Integrated Dynamics and Control, University of Newcastle, Apr. 2003.

[100] P. Wolfe. A duality theorem for non-linear programming. *Quarterly of Applied Mathematics*, 19(3):239 – 244, 1961.

[101] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

[102] S. J. Wright. Interior-point methods for optimal control of discrete-time systems. *Journal of Optimization Theory and Applications*, 77:161 – 187, 1993.

[103] S. J. Wright. Applying new optimization algorithms to model predictive control. In J. C. Kantor, C. E. García, and B. Carnahan, editors, *Chemical Process Control – V*, pages 147 – 155, 1997.

[104] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, 1996.

[105] M. Åkerblad. *A Second Order Cone Programming Algorithm for Model Predictive Control.* Licentiate's thesis, Royal Institute of Technology, 2002.

Appendices

A

Linear Algebra

In this appendix, some linear algebra results are presented. Let T be a square matrix partitioned according to

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \tag{A.1}$$

Definition A.1 (Schur complement). Suppose T_{11} is nonsingular, then the matrix S in

$$S = T_{22} - T_{21}T_{11}^{-1}T_{12} \tag{A.2}$$

is called the Schur complement of T_{11} in T.

The following lemma is called the Schur complement formula and is based on the discussion in [34, pp. 650–651]. It is given without any proof.

Lemma A.1 (Schur complement formula)
Assume T symmetric. Then

- $T \succ 0$ *if and only if* $T_{11} \succ 0$ *and* $S \succ 0$.
- *If* $T_{11} \succ 0$, *then* $T \succeq 0$ *if and only if* $S \succeq 0$.

The following matrix inversion lemma is taken from [104].

Lemma A.2 (Matrix inversion lemma)
If T and T_{11} are nonsingular, then

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}^{-1} = \begin{bmatrix} T_{11}^{-1} + T_{11}^{-1}T_{12}S^{-1}T_{21}T_{11}^{-1} & -T_{11}^{-1}T_{12}S^{-1} \\ -S^{-1}T_{21}T_{11}^{-1} & S^{-1} \end{bmatrix} \tag{A.3}$$

where S is defined in Definition A.1.

Proof:

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} T_{11}^{-1} + T_{11}^{-1}T_{12}S^{-1}T_{21}T_{11}^{-1} & -T_{11}^{-1}T_{12}S^{-1} \\ -S^{-1}T_{21}T_{11}^{-1} & S^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{A.4}$$

□

Lemma A.3
Given a square matrix $X \in \mathbb{R}^{n \times n}$ the following inversion formula holds

$$\begin{bmatrix} X & -I \\ -I & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & -I \\ -I & -X \end{bmatrix} \tag{A.5}$$

Proof:

$$\begin{bmatrix} X & -I \\ -I & 0 \end{bmatrix} \begin{bmatrix} 0 & -I \\ -I & -X \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{A.6}$$

□

Algorithm A.1 provides a method for solving a linear system with a non-singular submatrix T_{11} in two steps by partitioning the coefficient matrix T in four blocks. The algorithm is taken from [34].

Algorithm A.1 Block elimination

Given a linear equation system $Tx = b$, where T is nonsingular and partitioned as in (A.1). Assume T_{11} nonsingular and make the partitionings $x = \begin{bmatrix} x_1^T & x_2^T \end{bmatrix}^T$ and $b = \begin{bmatrix} b_1^T & b_2^T \end{bmatrix}^T$.
Form $T_{11}^{-1}T_{12}$ and $T_{11}^{-1}b_1$.
Form $S = T_{22} - T_{21}T_{11}^{-1}T_{12}$ and $\tilde{b} = b_2 - T_{21}T_{11}^{-1}b_1$.
Determine x_2 by solving $Sx_2 = \tilde{b}$.
Determine x_1 by solving $T_{11}x_1 = b_1 - T_{12}x_2$

B

Model Predictive Control Formulations

In this appendix, the MPC problem to minimize the objective function (3.2) subject to the dynamics 3.1 and the constraints 3.3 is cast on the form of a QP problem, that is (2.21). This can be done in several ways. See, for example, [67]. The optimization problems are formulated for time step t_0, which means that x_0 is the measured or estimated state of the true system at time step t_0.

In this thesis, two different formulations are used. The main difference between the two formulations is the formulation of the dynamic equations. Some notations are shared by the two formulations:

$$\begin{aligned}
\mathsf{x} &= \left[x^T(t_0), x^T(t_0+1), \ldots, x^T(t_0+N)\right]^T \\
\mathsf{u} &= \left[u^T(t_0), u^T(t_0+1), \ldots, u^T(t_0+N-1)\right]^T \\
\mathsf{r} &= \left[r^T(t_0), r^T(t_0+1), \ldots, r^T(t_0+N)\right]^T \\
\mathsf{Q}_e &= \operatorname{diag}\left(Q_e, \ldots, Q_e\right),\ \mathsf{Q}_u = \operatorname{diag}\left(Q_u, \ldots, Q_u\right),\ \mathsf{C} = \operatorname{diag}\left(C, \ldots, C\right) \\
\mathsf{H}_u &= \operatorname{diag}\left(H_u(0), \ldots, H_u(N-1)\right),\ \mathsf{H}_x = \operatorname{diag}\left(H_x(0), \ldots, H_x(N)\right) \\
\mathsf{h} &= \operatorname{diag}\left(h(0), \ldots, h(N)\right)
\end{aligned} \tag{B.1}$$

where $x(t) \in \mathbb{R}^n$ are the predicted states, $u(t) \in \mathbb{R}^m$ are the computed optimal control inputs and $r(t) \in \mathbb{R}^p$ is the reference signal. Note that it is not necessary to include $x(t_0)$ in x, since it is only set to the constant x_0.

B.1 Complete Set of Variables

The most straightforward way to cast the MPC problem on the form of a QP is to keep the dynamic equations as equality constraints. The MPC problem is then formulated as a

QP on the form

$$\begin{aligned} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T\end{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & \mathsf{Q}_u & 0\\ 0 & 0 & \mathsf{Q}_e\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix}\mathsf{A} & \mathsf{B} & 0\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} = b \\ & \begin{bmatrix}\mathsf{C} & 0 & -I\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} - \mathsf{r} = 0 \\ & \begin{bmatrix}\mathsf{H}_x & \mathsf{H}_u & 0\end{bmatrix}\begin{bmatrix}\mathsf{x}\\ \mathsf{u}\\ \mathsf{e}\end{bmatrix} + \mathsf{h} \leq 0 \end{aligned} \tag{B.2}$$

where

$$\begin{aligned} \mathsf{e} &= \begin{bmatrix}e^T(t_0), e^T(t_0+1), \ldots, e^T(t_0+N)\end{bmatrix}^T \\ b &= \begin{bmatrix}-x_0^T & 0 & \ldots & 0\end{bmatrix}^T \\ \mathsf{A} &= \begin{bmatrix} -I & 0 & 0 & \ldots & 0 & 0 \\ A & -I & 0 & \ldots & 0 & 0 \\ 0 & A & -I & \ldots & 0 & 0 \\ 0 & 0 & A & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & A & -I \end{bmatrix}, \; \mathsf{B} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ldots & 0 \\ 0 & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & B \end{bmatrix} \end{aligned} \tag{B.3}$$

This formulation requires $N\,(n+m+p)$ variables and gives a sparse Hessian matrix and a sparse constraint matrix. If this formulation is used, a solver either utilizing sparsity or the causality structure should be used.

B.2 Reduced Set of Variables

In the other formulation, x is expressed as a function of the initial state x_0 and the control inputs u. The vector x containing the states can then be inserted into the equations for the control error e. Finally, e is inserted into the objective function. By using the system equations, x can be found as

$$\mathsf{x} = S_x x_0 + \mathsf{S}_u \mathsf{u} \tag{B.4}$$

where

$$\mathsf{S}_x = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; \mathsf{S}_u = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \ldots & B \end{bmatrix} \tag{B.5}$$

The equality constraints are now eliminated and the objective function can be written as

$$\begin{aligned} J &= \left(\mathsf{C}\left(S_x x_0 + S_u \mathsf{u}\right) - \mathsf{r}\right)^T \mathsf{Q}_e \left(\mathsf{C}\left(S_x x_0 + S_u \mathsf{u}\right) - \mathsf{r}\right) + \mathsf{u}^T \mathsf{Q}_u \mathsf{u} \\ &= \mathsf{u}^T \left(S_u^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_u + \mathsf{Q}_u\right) \mathsf{u} + 2\left(S_u^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_x x_0 - S_u^T \mathsf{C}^T \mathsf{Q}_e \mathsf{r}\right)^T \mathsf{u} + \kappa \end{aligned} \tag{B.6}$$

where κ is a constant. By using (B.4), the inequality constraints can be written as

$$\mathsf{H}_x \mathsf{x} + \mathsf{H}_u \mathsf{u} + \mathsf{h} = \mathsf{H}_x S_u \mathsf{u} + \mathsf{H}_u \mathsf{u} + \mathsf{h} + \mathsf{H}_x S_x x_0 \leq 0 \tag{B.7}$$

Ignoring the constant and dividing by two gives the equivalent optimization problem

$$\begin{aligned} \underset{\mathsf{u}}{\text{minimize}} \quad & \frac{1}{2} \mathsf{u}^T \left(S_u^T \mathsf{C}^T \mathsf{Q}_e \mathsf{C} S_u + \mathsf{Q}_u\right) \mathsf{u} + \left(S_u^T \mathsf{C}^T \mathsf{Q}_e \left(\mathsf{C} S_x x_0 - \mathsf{r}\right)\right)^T \mathsf{u} \\ \text{subject to} \quad & \left(\mathsf{H}_x S_u + \mathsf{H}_u\right) \mathsf{u} + \mathsf{h} + \mathsf{H}_x S_x x_0 \leq 0 \end{aligned} \tag{B.8}$$

In this formulation, the Hessian becomes dense and Nm optimization variables are required.

Licentiate Theses
Division of Automatic Control
Linköpings universitet

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.
B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.
A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.
G. Malmberg: A Study of Adaptive Control Missiles. Thesis No. 76, 1986.
S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.
M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.
K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.
A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.
P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.
K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.
I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.
F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.
H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.
S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.
K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.
J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.
P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.
H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.
T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.
T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.
P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.
J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.
J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.
A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.
M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.
K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.
J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.
A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.
M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.
U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.

M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

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

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

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

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