Linköping studies in science and technology. Dissertations.
No. 1158

Integer Quadratic Programming for Control and Communication

Daniel Axehill

Department of Electrical Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2008

Linköping studies in science and technology. Dissertations.
No. 1158

Integer Quadratic Programming for Control and Communication

Daniel Axehill

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

ISBN 978-91-85523-03-0 ISSN 0345-7524

Copyright © 2008 Daniel Axehill

Parts of this thesis have been previously published.
Papers A, B, C, and F are reprinted with permission from IEEE. Copyright © 2004-2008 IEEE.
Paper E is reprinted with permission from Elsevier. Copyright © 2007 Elsevier.

Printed by LiU-Tryck, Linköping, Sweden 2008

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 methods is Model Predictive Control (MPC). In each sampling time, 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 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, which is known to have a computational complexity which grows exponentially in the number of binary optimization variables. In modern communication systems, multiple users share a so-called multi-access channel. To estimate the information originally sent, a maximum likelihood problem involving binary variables can be solved. The process of simultaneously estimating the information sent by multiple users is called Multiuser Detection (MUD). In this thesis, the problem to efficiently solve MIQP problems originating from MPC and MUD is addressed. Four 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 numerical experiments, the algorithm is applied to unconstrained MPC problems with a mixture of real valued and binary valued control signals, and the result shows that the performance gain can be significant compared to solving the problem using branch and bound. The preprocessing algorithm has also been applied to the MUD problem, where simulations have shown that the bit error rate can be significantly reduced 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 proposed QP solver and MIQP solver have lower computational complexity compared to corresponding generic solvers. Third, the dual active set QP algorithm is enhanced using ideas from gradient projection methods. The performance of this enhanced algorithm is shown to be comparable with the existing commercial state-of-the-art QP solver CPLEX for some random linear MPC problems. Fourth, an algorithm for efficient computation of the search directions in an SDP solver for a proposed alternative SDP relaxation applicable to MPC problems with binary control signals is presented. The SDP relaxation considered has the potential to give a tighter lower bound on the optimal objective function value compared to the QP relaxation that is traditionally used in branch and bound for these problems, and its computational performance is better than the ordinary SDP relaxation for the problem. Furthermore, the tightness of the different relaxations is investigated both theoretically and in numerical experiments.

Populärvetenskaplig sammanfattning

Denna avhandling handlar om hur vissa metoder från optimeringslära kan användas för att lösa problem inom reglerteknik och kommunikation, samt hur dessa metoder kan göras snabbare genom att specialanpassas för dessa tillämpningar. Inom reglertekniken är Modellprediktiv reglering (MPC) en av de mest använda moderna reglerstrategierna. MPC är en tidsdiskret metod, vilket betyder att viktiga storheter i systemet mäts med vissa tidsintervall. Efter varje mätning löses ett optimeringsproblem för att kunna hitta det bästa sättet att påverka systemet fram till nästa mättidpunkt.

Ett problem med MPC är att om systemet som ska styras är komplext, eller om det som ska styras ändrar sig snabbt, blir det en utmaning att lösa optimeringsproblemet på den ofta korta tid som finns tillgänglig mellan mättidpunkterna. Ett sätt att minska dessa problem är att använda optimeringsverktyg som är specialanpassade för MPC. På senare år har användningsområdet för MPC ökat och metoden har blivit användbar för alltmer avancerade uppgifter.

Denna avhandling fokuserar på styrning av system som bara kan påverkas på ett fixt antal sätt eller där det finns egenskaper hos systemet som bara kan anta ett fixt antal värden. Matematiskt representeras denna egenskap i avhandlingen som noll eller ett, det vill säga binärt. Exempelvis kan ofta elementet i en ugn bara styras i termer av ”på” eller ”av”. Ett exempel där det finns inre egenskaper hos ett system med denna egenskap är en vattentank, där egenskapen ”full” bara kan anta värdena ”sant” eller ”falskt”. Det visar sig att det är ofta en väl så svår uppgift att styra ett system som innehåller dessa binära inslag, jämfört med att styra ett system där motsvarande egenskap kan anta alla värden mellan noll och ett. Speciellt blir det svårt om det finns många sådana egenskaper i systemet, eftersom det då finns en risk att alla möjliga kombinationer av dessa egenskaper måste testas för att avgöra vilken som är bäst. Problemet får exponentiell komplexitet, vilket i det här fallet i princip innebär att problemet blir dubbelt så svårt att lösa för varje extra sådan egenskap. Optimeringsproblemet vilket måste lösas efter varje mätning blir då av typen Mixed Integer Quadratic Programming (MIQP).

I moderna kommunikationssystem delar många användare på en och samma kanal. Principen är densamma som om en telefonlinje har flera användare i varje ände. Så länge personerna i samma ände inte har samma röst går det principiellt att särskilja dem, även om de talar i mun på varandra. Denna princip kallas i telekommunikationssammanhang för Code Division Multiple Access (CDMA) och används då flera mobiltelefoner samtidigt använder samma frekvenser vid samma tidpunkter. Även detta problem kan formuleras matematiskt som en typ av MIQP-problem. Detta beror på att mobiltelefonernas kommunikation sker med ettor och nollor. Optimeringsproblemet som ska lösas kan sägas ha funktionen att avgöra om det är troligast att det var en etta eller en nolla som den sändande mobiltelefonen skickade.

I avhandlingen presenteras fyra olika algoritmer, för att lösa problem av den här typen. Den första algoritmen som presenteras i avhandlingen är en så kallad preprocessing-algoritm. Generellt sett är det en algoritm som körs före att den egentliga lösningsalgoritmen appliceras på problemet. I det här fallet utnyttjas en viss egenskap som finns hos både MPC-problemet och CDMA-problemet för att på ett snabbt och enkelt sätt kunna bestämma det optimala värdet på en eller flera binära storheter i problemet. Den andra algoritmen i avhandlingen kan användas för att snabba upp den mest beräkningskrävande delen av en generell algoritm för att lösa MPC-problem för system innehållande binära storheter. Eftersom det kan vara mycket tidskrävande att testa alla kombinationer av nollor och ettor i problemet är det önskvärt att på något sätt reducera antalet kombinationer som potentiellt sett skulle kunna vara optimala. Ett sätt är att tillfälligt släppa på kravet att storheten i problemet *antingen* ska vara noll eller ett och istället tillåta dem att vara vilket värde som helst *mellan* noll och ett. Genom att lösa ett optimeringsproblem där denna förenkling har gjorts kan information fås som kan användas för att utesluta många av de möjliga kombinationerna från att vara optimala. Ofta måste många sådana här förenklade (relaxerade) problem lösas innan lösningen till det ursprungliga problemet hittas. Därför är det viktigt att de kan lösas snabbt. Algoritmen som presenteras i avhandlingen är specialgjord för att lösa förenklade problem av den här typen som har sitt ursprung i ett MPC-problem med binära variabler. Den tredje algoritmen bygger vidare på den andra algoritmen. De ändringar som har gjorts innebär att prestandan har kunnat ökats ytterligare. Den fjärde och sista algoritmen bygger vidare på den tidigare använda metoden där ett svårt problem löses genom att lösa många förenklade problem. Idén med den fjärde algoritmen är att lösa andra typer av förenklade problem för att kunna förkasta ett ännu större antal kombinationer som annars måste testas. Även den fjärde algoritmen utnyttjar egenskaper hos MPC-problemet för att snabbare kunna lösa det resulterande optimeringsproblemet.

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.

The five years at the Automatic Control group have been very developing, not only professionally. Thanks to these years, I met Johanna! Thank you Johanna, for all the joy that you bring to my life, and for supporting me and encouraging me. I'm looking forward to spending more time with you in the future!

Also at the top of the list, my deepest gratitude goes to my supervisor Anders Hansson for his help and guidance during the past five years. I would especially like to thank Anders for his thorough reading of our articles and valuable comments on my work. Anders and I have had many interesting discussions and I have learnt a lot. Another person who has a special place on this page is Lennart Ljung. I am very grateful that he let me join the Automatic Control group in Linköping. It has been an honor to work with someone who has such a broad knowledge. I would also like to thank Fredrik Gunnarsson for an excellent and fruitful cooperation during our work on multiuser detection. Lieven Vandenberghe is gratefully acknowledged for letting me visit him at UCLA in fall 2006. Lieven is not only a person with unique knowledge, he is also a very kind and humble person. The visit at UCLA was a success, both for my research and for myself as a person. So, one more time, thank you Lieven for letting me visit you!

There are some more persons that I would like to thank specifically. 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 Daniel Ankelhed, Janne Harju Johansson, Gustaf Hendeby, Johan Löfberg, Daniel Petersson, Johan Sjöberg, Henrik Tidefelt, David Törnqvist, Johanna Wallén, Ragnar Wallin and Erik Wernholt. A specific thanks goes to Gustaf Hendeby for all help regarding LaTeX. I would also like to thank him for the excellent thesis style in which this thesis is formatted. There is one person who has, without any formal obligations, helped me and supported me a significant number of times. His name is Ragnar Wallin. I have written it before, and I will write it again, Ragnar is an extremely kind and helpful person. He has helped me with many things and answered many questions. I would also like to thank Gustaf Hendeby, Henrik Tidefelt and Daniel Petersson for many interesting research discussions. Especially, Henrik's and Daniel's help and knowledge has been invaluable during the work with the gradient projection project. I would also like to thank Ulla Salaneck who has helped me with various practical things, always with a smile on her face. I have discussed many things with our colleagues at the mathematical department. I would like to thank you all for your time! Especially, I would like to thank Oleg Burdakov and Jan Åslund, with whom I have had very fruitful discussions.

To ensure that I do not forget anyone, I would like to thank the entire Automatic Control group in Linköping. It has been a pleasure to work with you!

Last, but not least, I would like to thank VINNOVA's Center of Excellence ISIS (Information Systems for Industrial Control and Supervision), ECSEL (The Excellence Center in Computer Science and Systems Engineering in Linköping), and the Swedish Research Council for their economical support.

Linköping, January 2008
Daniel Axehill

Contents

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 for scalar x and y.
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,... placed 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 in a block 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 1.
$A_{(i,:)}$	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.
$\lVert x \rVert_Q$	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 $n(\tau)$.

Notation	Meaning
$\underset{x}{\operatorname{argmin}}\, f(x)$	The optimal solution to $\underset{x}{\min}\, f(x)$.
$\operatorname{dom} f(x)$	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.
$\operatorname{relint} \mathcal{C}$	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.
∇f	Gradient of a function f.
$\nabla^2 f$	Hessian of a function f.
$x \leftarrow y$	Assignment. x is assigned the value of y.
$\operatorname{tr} X$	Trace of a matrix X.

Sets

Notation	Meaning
$\mathbb{R}$	Set of real numbers.
$\mathbb{R}^n$	Set of real vectors with n components.
$\mathbb{R}^n_{++}$	Set of positive 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 of symmetric positive semidefinite matrices with n rows.
$\mathbb{S}^n_{++}$	Set of 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

Abbreviation	**Meaning**
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
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
SDP	Semidefinite 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 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, it is not unusual that on-off-control is used. In more advanced examples, logical rules are embedded in the system. To be able to control optimally, the controller 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 introduction, a more precise background is given in Section 1.1. The publications on which this thesis is based, are listed in Section 1.2. A thesis outline is given in Section 1.3. The chapter is concluded in Section 1.4, where the contributions constituting the foundation of this thesis are summarized.

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 online, there is a need for efficient QP optimization routines. As the optimization routines become more efficient and the hardware more powerful, larger systems at faster sampling rates become possible to control by MPC.

During the past ten years, 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 make 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 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 those systems, 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 function is to be minimized and the optimization variables are the bits sent by the senders. 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 Publications

The thesis is based on the following publications, where the author is the main contributor:

> D. Axehill and A. Hansson. A preprocessing algorithm for MIQP solvers with applications to MPC. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 2497–2502, Atlantis, Paradise Island, Bahamas, Dec. 2004.
>
> 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.
>
> D. Axehill, F. Gunnarsson, and A. Hansson. A low-complexity high-performance preprocessing algorithm for multiuser detection using Gold sequences. Provisionally accepted for publication in *IEEE Transactions on Signal Processing*, Jan. 2008.
>
> D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698, Manchester Grand Hyatt, San Diego, USA, Dec. 2006.
>
> D. Axehill and A. Hansson. A dual gradient projection quadratic programming algorithm tailored for mixed integer predictive control. Technical Report LiTH-ISY-R-2833, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Dec. 2007.
>
> D. Axehill, L. Vandenberghe, and A. Hansson. On relaxations applicable to model predictive control for systems with binary control signals. In *Preprints of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 200–205, Pretoria, South Africa, Aug. 2007.
>
> D. Axehill, A. Hansson, and L. Vandenberghe. Relaxations applicable to mixed integer predictive control – comparisons and efficient computations. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4103–4109, Hilton New Orleans Riverside, New Orleans, USA, Dec. 2007.

1.3 Thesis Outline

The thesis is organized as follows. Part I contains an overview of the optimization theory, control theory and communication theory related to the results presented in this thesis. Parts of the material have previously been published in [3]. Part I serves as a background for Part II, which contains a collection of papers containing the results of this thesis.

1.3.1 Outline of Part I

This part provides 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. In the end of the chapter, two systems in MLD form are introduced that will serve as benchmark examples in several of the papers presented in Part II. In Chapter 4, MUD and Code Division Multiple Access (CDMA) are explained.

1.3.2 Outline of Part II

In this part, the results of this thesis are summarized in the form of a collection of papers. The papers are:

Paper A: A Preprocessing Algorithm for MIQP Solvers with Applications to MPC

This paper is an edited version of [4]. In this paper, a preprocessing algorithm applicable to unconstrained BQP and MIQP problems is presented. The algorithm is applied to an unconstrained MPC problem with a mix of binary valued and real valued control signals. Simulation results indicate that a significant amount of computational time can be saved by combining branch and bound with the proposed preprocessing algorithm compared to using branch and bound without the preprocessing algorithm.

Paper B: A Low-Complexity High-Performance Preprocessing Algorithm for Multiuser Detection using Gold Sequences

This paper is an edited version of [12]. In this paper, the BQP preprocessing algorithm presented in Paper A is applied to the MUD problem. It is shown that the performance in the synchronous MUD problem is very good compared to some ordinary suboptimal detectors when signature sequences with low cross-correlation are used.

Paper C: A Mixed Integer Dual Quadratic Programming Algorithm Tailored for MPC

This paper is an edited version of [6]. In this paper, a dual active-set quadratic programming algorithm is developed. The algorithm is tailored for MPC problems and for solving similar problems repeatedly, that is, so-called warm starts can be performed very efficiently. The solver is incorporated as a part of a branch and bound algorithm for MIQP problems in order to be able to efficiently solve the optimization problems originating from MPC problems for hybrid systems. The results from numerical experiments indicate that the computational complexity is significantly decreased compared to a generic comparable solver, both in the case when the proposed QP solver used stand-alone, and when it is incorporated as part of an MIQP solver.

Paper D: A Dual Gradient Projection Quadratic Programming Algorithm Tailored for Mixed Integer Predictive Control

This paper is an edited version of [7]. In this paper, the performance of the algorithm in Paper C is significantly increased by combining the ideas in Paper C with ideas from gradient projection methods. The result is a high-performing QP algorithm which is tailored for MPC, has good warm-start properties, and has good performance also in the case when many inequality constraints are present in the problem.

Paper E: On Relaxations Applicable to Model Predictive Control for Systems with Binary Control Signals

This paper is an edited version of [11]. In this paper, Semidefinite Programming (SDP) relaxations are considered for unconstrained MPC problems with binary inputs. An alternative relaxation that is new to this application is introduced. It is shown to have the potential to provide tighter bounds than the ordinary QP relaxation and lower computational complexity than the ordinary SDP relaxation. Furthermore, different relaxations applicable to the problem are related both theoretically and in computational experiments.

Paper F: Relaxations Applicable to Mixed Integer Predictive Control – Comparisons and Efficient Computations

This paper is an edited version of [10]. In this paper, the work in Paper E is extended. After the extension, also problems involving inequality constrained systems can be solved. Furthermore, it is shown how the SDP relaxation introduced in Paper E can be very efficiently computed by utilizing the specific structure in the problem, which originates from the fact that the underlying problem is an MPC problem. It is shown that the computational complexity in practice will grow roughly linearly in the prediction horizon if the proposed algorithm is used.

1.4 Contributions

This thesis is based on both previously published, [4–6, 9–11], and previously unpublished results. The main contributions of this thesis are:

- A new preprocessing algorithm applicable to the BQP and MIQP problems is presented. The algorithm is introduced in [4], where it is applied to MPC problems with a mix of binary valued and real valued control signals. In [9, 12] the algorithm is applied to the MUD problem where it is shown that it can improve the Bit Error Rate (BER) performance of existing suboptimal detectors at a moderate computational cost. Furthermore, it is also shown how it can be combined with an optimal detector in order to solve the MUD problem exactly with a reduced computational cost.

- In [6] a new active set QP algorithm tailored for MPC is presented. It is shown that if the dual QP problem is explicitly derived, it gets a structure which makes it possible to apply an algorithm built on similar ideas as the one presented in [68].

Furthermore, it is shown that this dual active set algorithm can be useful both in itself as a QP solver tailored for linear MPC problems, but also as a part of an MIQP solver built on the branch and bound method. In [7], the ideas from [6] are combined with ideas from gradient projection methods into a new algorithm which is the first algorithm that combines ideas from gradient projection, utilizes the MPC problem structure, and is working in the dual space. It is also shown how it can be used as an important part of an MIQP solver tailored for MPC.

- In [10] and [11], simulation studies are performed to evaluate the performance of different types of relaxations applicable to MPC for hybrid systems. Furthermore, an alternative SDP relaxation for the hybrid MPC problem is introduced. The optimization ideas used to derive this new relaxation are not new, but the resulting relaxation has not previously been used in the context of MPC problems involving binary variables. It is shown that it can be expressed as a QP problem in a special case, and that it in this special case can improve the computational performance of a branch and bound method using the ordinary QP relaxation. Furthermore, it is shown in [10] that the computational performance of the alternative SDP relaxation can be improved by utilizing problem structure.

Part I

Background

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 an 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 [39].

2.1 Introduction and Basic Concepts

This section is opened with the 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 $\theta \in [0,1]$

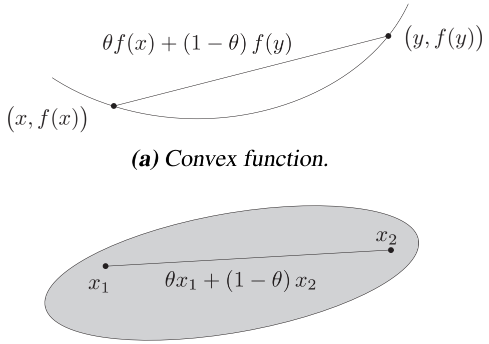
$$\theta x_1 + (1-\theta)\, x_2 \in \mathcal{C} \tag{2.1}$$

Convex sets are thoroughly discussed in, for example, [39].

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


$\theta f(x) + (1-\theta) f(y)$
$(y, f(y))$
$(x, f(x))$
x_2
x_1
$\theta x_1 + (1-\theta) x_2$

***(a)** Convex function.*

***(b)** Convex set.*

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

Important examples of convex functions, as well as operations that preserve convexity, can be found in [39].

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

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

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

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

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 reason to consider a dual problem is to get an alternative formulation of the optimization problem that is more computationally attractive or has some theoretical significance, [51]. 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 nonlinear programming can be found in, for example, [46, 47, 105].

2.2.1 The Lagrange Dual Problem

In the derivation of a dual optimization problem, the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$ 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 $g : \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}$

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

is obtained.

An important operation that conserves convexity 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

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

This problem is called the Lagrange dual problem associated with (2.3). Note that there exist other dual problems. Example of other dual formulations for nonlinear programs are the Wolfe dual, [105], and the Dorn dual for quadratic programs, [46]. 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, or 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. Let the primal problem be in 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. 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 [39, 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 2.2 (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. They are used as necessary conditions for optimality for optimization problems with differentiable objective function and constraint functions, and for which strong duality holds. If the problem is convex they are also sufficient according to the following theorem, based on the discussion in [39, pp. 243–244].

Theorem 2.3 (KKT)
Consider the optimization problem (2.3)*. Assume that it is convex, that* $f_i(x),\ i = 0, \ldots, m$ *and* $h_i(x),\ i = 1, \ldots, p$ *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

$$f_i(x^*) \leq 0, \quad i = 1, \ldots, m \tag{2.20a}$$

$$h_i(x^*) = 0, \quad i = 1, \ldots, p \tag{2.20b}$$

$$\lambda_i^* \geq 0, \quad i = 1, \ldots, m \tag{2.20c}$$

$$\lambda_i^* f_i(x^*) = 0, \quad i = 1, \ldots, m \tag{2.20d}$$

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0 \tag{2.20e}$$

Proof: See [39, p. 244]. □

2.4 Steepest Descent and Newton Methods

So far, it has not been mentioned *how* an optimal solution to an optimization problem can be found. In this section, some fundamental algorithms that perform this task are introduced, namely descent methods. This section serves as a short background to the discussion in Section 2.5.3 and in Section 2.6. The presentation is based on the ones found in [39] and in [84]. Descent methods are methods that generate a sequence x_k, $k = 1, \ldots$ where

$$x_{k+1} = x_k + \alpha_k s_k \tag{2.21}$$

and where $\alpha_k > 0$ except when x_k is optimal. The vector s_k is called the step or search direction. The scalar α_k is called the step length at step k, that is, the length of the step from the point x_k in the direction s_k. The sequence that is produced satisfies

$$f(x_{k+1}) < f(x_k) \tag{2.22}$$

until an optimal x_k has been found. There are several possible choices of the search direction s_k. However, in order to satisfy the condition in (2.22), it must satisfy

$$\nabla f(x_k)^T s_k < 0 \tag{2.23}$$

that is, it must make an angle strictly less than $\pi/2$ radians with the negative gradient. If s_k satisfies this condition, it is called a descent direction. An outline of a generic descent method is found in Algorithm 2.1. The two main operations in this algorithm are the computation of the search direction s_k and the step length α_k. Hence, these two will now be further described.

During the line search in iteration k, the objective function is either approximately or exactly minimized along the ray defined by $\{x_k + \alpha_k s_k | \alpha_k \geq 0\}$. In practice, it is most common to use inexact line searches. However, for some optimization problems, the minimizer α_k^* can be found efficiently or analytically. One such case is described in Paper D. In the inexact line search methods, s_k is chosen such that the objective function reduction is “sufficient”. For a more thorough description of these methods, see [39] or [84].

Algorithm 2.1 Generic descent algorithm

```
Compute a feasible starting point $x_0$.
while Stopping criterion is not satisfied do
    Compute a search direction $s_k$.
    Compute $\alpha_k$ by performing a line search in the direction $s_k$.
    $x_{k+1} \leftarrow x_k + \alpha s_k$
    $k \leftarrow k + 1$
end while
```

As discussed earlier, there is a freedom in the choice of the search direction s_k. Two commonly used search directions are the steepest descent direction and the Newton direction. The steepest descent direction is the direction in which the objective function decreases most rapidly in a neighborhood of the current point x_k. By using the definition of the scalar product, it is easy to see that this direction is

$$s_k = -\frac{\nabla f(x_k)}{\|f(x_k)\|} \tag{2.24}$$

The drawback with this search direction is that it can result in slow convergence on difficult problems, specifically when the Hessian of $f(x)$ is ill-conditioned around the optimal point. The convergence rate when the steepest descent direction is used is linear, which means that the distance to the optimum decreases with at least a positive constant factor strictly less than one in each iteration. See Appendix C for a more thorough description of convergence rates. An advantage with the method is that it does not require computation of $\nabla^2 f(x_k)$. The Newton direction is found from the second-order Taylor-series expansion of the objective function

$$f(x_k + s) \approx f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s \triangleq m_k(s) \tag{2.25}$$

around the current point x_k. This approximation serves as a quadratic model of the original objective function. Assume that $\nabla^2 f(x_k) \succ 0$. Then, the Newton step s_k is defined as the minimizer to the quadratic model $m_k(s)$, that is

$$s_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \tag{2.26}$$

The main advantage of the Newton method is that the convergence rate becomes quadratic instead of linear as for the steepest descent method. The major drawback of using the Newton direction is that it can be rather computationally demanding since it requires the computation of the Hessian of the original objective function. Note that, if the original objective function is quadratic and strictly convex, the quadratic model is no approximation but exact. As a consequence, the minimizer to the original problem is in this case found in one iteration of Algorithm 2.1. A compromise between the simplicity of the steepest descent method and the rapid convergence of the Newton method is offered by the quasi-Newton methods, where an approximation of the Hessian is used to obtain a superlinear convergence rate.

The stopping criterion in Algorithm 2.1 is often of the form $\|\nabla f(x)\|_2 \leq \epsilon$, where ϵ is a small non-negative number.

2.5 Quadratic Programming

In this section, a Quadratic Programming (QP) problem in the form

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

is used, 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 (2.27) can be found by first forming the Lagrangian

$$L(x, \lambda, \nu) = \frac{1}{2}x^T H x + f^T x + \lambda^T \left(A_{\mathcal{I}} x - b_{\mathcal{I}}\right) + \nu^T \left(A_{\mathcal{E}} x - b_{\mathcal{E}}\right) \tag{2.28}$$

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

$$\frac{\partial L(x, \lambda, \nu)}{\partial x} = Hx + f + A_{\mathcal{I}}^T \lambda + A_{\mathcal{E}}^T \nu = 0 \Leftrightarrow x = -H^{-1}\left(f + A_{\mathcal{I}}^T \lambda + A_{\mathcal{E}}^T \nu\right) \tag{2.29}$$

Inserting (2.29) into (2.28) gives the following expression for the dual function

$$\begin{aligned} g(\lambda, \nu) = & -\frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} \\ & - \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \frac{1}{2} f^T H^{-1} f \end{aligned} \tag{2.30}$$

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

$$\begin{aligned} \underset{\lambda, \nu}{\text{maximize}} \quad & -\frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \\ & - \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} - \frac{1}{2} f^T H^{-1} f \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.31}$$

By changing the sign of the objective function and ignoring the constant term, (2.31) can be written as an equivalent (see Definition 2.4) minimization problem

$$\begin{aligned} \underset{\lambda, \nu}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I}} \\ A_{\mathcal{E}}\end{bmatrix} H^{-1} \begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix}\begin{bmatrix}\lambda \\ \nu\end{bmatrix} + \\ & + \left(f^T H^{-1}\begin{bmatrix}A_{\mathcal{I}}^T & A_{\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{2.32}$$

Since the constraints of a QP are linear, it follows from Theorem 2.2 that if the primal problem is feasible, strong duality holds. Furthermore, it can be shown that if the dual is feasible, then the primal is feasible if and only if the dual optimal objective function value is bounded from above. That is, the objective function values coincide also in the case the primal is infeasible. For details, see Paper D.

Remark 2.2. Note that the optimal solutions of (2.31) and (2.32) 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 (2.27) to the optimal objective function value of (2.31), but not to the optimal objective function value of (2.32).

The great structural advantage with the dual problem (2.31), or (2.32), compared to the primal problem (2.27), 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 2.5.1) compared to a classical active set algorithm, [85]. More advantages of the dual formulation are presented in Section 2.5.2. Early work on duality for QPs can be found in [46], [47] and [45]. The dual problem to a QP is considered in several books. Some examples are [85], [51] and [16]. For an extensive bibliography on QP, see [60].

2.5.1 Active Set Methods

An inequality constrained QP can be solved by, for example, an interior point method or an active set method. In this thesis, the focus will be on active set methods. 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 [85] and [51]. The problem to be solved is of the type in (2.27), with $H \in \mathbb{S}_{+}^{n}$. However, in each QP iteration an equality constrained QP with $m \leq n$ number of constraints is considered:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & Ax = b \end{aligned} \tag{2.33}$$

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 non-singular. Further, Z and Y should satisfy $AY = I$ 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

$$x = Yb + Zx_Z \tag{2.34}$$

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 (2.34) is inserted into (2.33), the following unconstrained optimization problem is obtained

$$\underset{x_Z}{\text{minimize}} \quad \tfrac{1}{2}x_Z^T Z^T H Z x_Z + (f + HYb)^T Z x_Z + \tfrac{1}{2} b^T Y^T H Y b + f^T Y b \tag{2.35}$$

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 (2.27) 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 are of importance when solving (2.33). 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 (2.33) can be written as a system of linear equations

$$K \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} -f \\ b \end{bmatrix} \tag{2.36}$$

The following lemma taken from [85] gives sufficient conditions for non-singularity of the KKT matrix K.

Lemma 2.1
Let A have full row rank and assume that the reduced-Hessian matrix $Z^T H Z$ is positive definite. Then the KKT matrix K in (2.36) *is non-singular and there is a unique pair of vectors (x^*, ν^*) satisfying* (2.36).

Proof: See [85, p. 445]. □

Actually, a more powerful result can be shown. The following theorem is taken from [85].

Theorem 2.4
Suppose that the conditions of Lemma 2.1 are satisfied. Then the vector x^* satisfying (2.36) *is the unique global solution of* (2.33).

Proof: See [85, p. 446]. □

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

Definition 2.5 (Active set). The set

$$\mathcal{A}(x) = \mathcal{E} \cup \left\{ i \in \mathcal{I} \mid a_i^T x = b_i \right\} \tag{2.37}$$

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 (2.33), 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 an active set QP solver a sequence of equality constrained problems of the type in (2.33) 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 (2.36). 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. However, 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, [85] and [51].

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 satisfied for $\mathcal{W}_0$, the algorithm to be presented guarantees that it will also be satisfied for $\mathcal{W}_k$ in all subsequent iterations, [85].

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, x_{k+1} is chosen as $x_{k+1} = x_k + \alpha_k p_k$ where α_k is the largest number in $[0, 1]$ for which 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

$$\begin{aligned} A_{\mathcal{W}_k} x_{k+1} &= A_{\mathcal{W}_k} \left(x_k + \alpha_k \left(\hat{x}_{k+1} - x_k \right) \right) \\ &= A_{\mathcal{W}_k} x_k + \alpha_k \left(A_{\mathcal{W}_k} \hat{x}_{k+1} - A_{\mathcal{W}_k} x_k \right) = b_{\mathcal{W}_k} \end{aligned} \tag{2.38}$$

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 satisfied for an arbitrary $\alpha_k \geq 0$. On the contrary, if $a_i^T p_k > 0$, then α_k has to satisfy

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k} \tag{2.39}$$

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

$$\alpha_k = \min \left\{ 1, \min_{i \notin \mathcal{W}_k,\, a_i^T p_k > 0} \left(\frac{b_i - a_i^T x_k}{a_i^T p_k} \right) \right\} \tag{2.40}$$

where $p_k = \hat{x}_{k+1} - x_k$. The constraints for which the minimum in (2.40) 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. The difference between a Lagrange multiplier corresponding to an equality constraint and a Lagrange multiplier corresponding to an inequality constraint, is that the latter 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 satisfied 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 satisfied 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 [85], 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 satisfied 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 satisfied by the construction of the active set algorithm. In every iteration, $\hat{x}_{k+1}$ satisfies 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 satisfied. In this case, all KKT conditions are satisfied 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 2.2, $\hat{x}_{k+1}$ is explicitly computed instead of p. Apart from this modification, Algorithm 2.2 is similar to the algorithm given in [85]. 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

Algorithm 2.2 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 $x_0$.
$k \leftarrow 0$
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
            $x^* \leftarrow x_k$
            STOP
        else
            $j \leftarrow \underset{j \in \mathcal{W}_k \cap \mathcal{I}}{\text{argmin}}\ \hat{\lambda}_j$
            $x_{k+1} \leftarrow x_k$
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$
        end if
    else
        Compute $\alpha_k$ according to (2.40).
        $x_{k+1} \leftarrow x_k + \alpha_k \left(\hat{x}_{k+1} - x_k\right)$
        if $\alpha_k < 1$ then
            Set $j$ to be the index of one of the blocking constraints.
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{j\}$
        else
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
        end if
    end if
    $k \leftarrow k + 1$
end while
No solution was found in $k_{max}$ iterations.
```

according to [85], 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 Phase I method, [85], 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, [85], where a weighted infinity norm of the amount of infeasibility is added to the objective function as a penalty.

2.5.2 Dual Active Set Quadratic Programming Methods

In this section, a motivation is given to why active set methods working on the *dual* QP problem are preferable in the application considered in this thesis. The QP method presented in Section 2.5.1 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 2.5.1, if a feasible starting point cannot be obtained by, for example, practical knowledge of the problem, a Phase I algorithm can be applied to find such a point. According to [58], 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 (2.27) and the dual QP problem in (2.31), 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 (2.27) have to be satisfied. According to [85], 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 2.2, is that rapid changes to the working set are allowed. As a consequence, the number of QP iterations can be reduced.

2.5.3 Gradient Projection Quadratic Programming Methods

A drawback with the active set method presented in Section 2.5.1, is that after each change of the working set (adding or dropping a constraint), a linear system of equations in the form in (2.36) has to be solved in order to get the new search direction. In problems with many constraints, there might be a considerable number of constraints that have to be added or dropped before the optimal working set has been identified, and hence, there might be a considerable number of linear systems in the form in (2.36) to solve before the optimal solution is found. In the simplest form, gradient projection methods are basically steepest descent methods where the steepest descent direction is "bent" such that is stays feasible if it hits an inequality constraint. This is illustrated in Figure 2.2. To get a more rapid convergence, the basic steepest descent method is often used in combination with another, more rapidly convergent, method. Furthermore, other search directions than the steepest descent direction may be used and projected. However, it should be mentioned that such directions have to be chosen with some care, as discussed in [34] and in Paper D. The process of "bending" the search direction is called projection of the search direction, hence the name gradient projection. This procedure makes it possible to encounter several constraints during a line search along the search direction without recomputing the search direction, that is, the sometimes computationally expensive search direction computation is utilized as much as possible. This property is especially important in optimal control applications where often many control inputs are at their boundaries at the optimum, [33].

The gradient projection idea has been considered in several articles, and it exists in different variants. The gradient projection algorithm presented in Paper D is inspired by the algorithm presented in [84]. This algorithm can be found in this section as Algorithm 2.3, and it will now be explained. During this discussion, a generic QP problem in

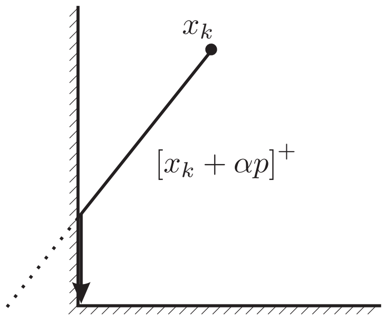


Figure 2.2: *The figure illustrates how the points along the line starting in the point x_k in the direction p are projected back onto the positive orthant during the operation $[x_k + \alpha p]^+$. The resulting path is piecewise linear.*

the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q(x) = \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & x \geq 0 \end{aligned} \tag{2.41}$$

is considered. The two main operations in the algorithm are the gradient projection operation on line 9, and the improvement operation on line 10. During the gradient projection operation, a search for the first local minimizer is performed from the starting point x_k in the negative gradient direction. If constraints are encountered during the search, the search direction is bent as described above. The local minimizer found is called the Cauchy point and is denoted x^c. The Cauchy point plays an important role for the convergence properties of an algorithm. Basically, a step that is at least as good as a step to the Cauchy point guarantees global convergence. For further details, see [84]. If a step to the next Cauchy point is taken in each iteration, without extra improvement by another algorithm, the convergence rate will be rather slow (linear) and the algorithm will basically implement a steepest descent method. Therefore, given a point x^c good enough to guarantee convergence, an attempt is performed on line 10 to improve on that point, in order to get a better convergence rate. Variables that are at their bounds at x^c will be forced to remain there during the improvement operation. During that operation, a subspace minimization problem in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q_s(x) = \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & x_i = 0, \ i \in \mathcal{A}(x^c) \\ & x_i \geq 0, i \notin \mathcal{A}(x^c) \end{aligned} \tag{2.42}$$

is solved, where $\mathcal{A}(x^c) = \{i : x_i^c = 0\}$. In order to obtain global convergence, it is not necessary to solve the problem in (2.42) exactly. The requirement on the approximate solution x^+ is that it is not worse than x^c and that it is feasible with respect to the feasible set in (2.41). Since it might be almost as difficult to solve as the original problem, it is often not desirable to solve it exactly. One possible choice is to run a conjugated gradient method on the subspace defined by the equality constraints in (2.42), and either abort

Algorithm 2.3 Gradient projection algorithm for QP, [84]

```
Compute a feasible starting point $x_0$.
Define the maximum number of iterations as $k_{max}$.
$k \leftarrow 0$
while $k < k_{max}$ do
  if $x_k$ satisfies the KKT conditions for (2.41) then
    $x^* \leftarrow x_k$
    STOP
  end if
  Starting in $x_k$, find the Cauchy point $x^c$.
  Find an approximate minimizer $x^+$ to the subproblem in (2.42), such that
  $Q(x^+) \leq Q(x^c)$ and such that $x^+$ is feasible with respect to the constraints in
  the problem in (2.41).
  $x_{k+1} \leftarrow x_k$
  $k \leftarrow k + 1$
end while
No solution was found in $k_{max}$ iterations.
```

it as soon as an inequality constraint is encounter, or continue the conjugated gradient iterations while ignoring the inequality constraints and projecting this solution back onto the feasible set.

If the algorithm in Algorithm 2.3 approaches a solution x^* at which the Lagrange multipliers associated with all the active inequality constraints are non-zero, that is, strict complementarity holds, the active sets $\mathcal{A}(x^c)$ generated by the algorithm are equal to the optimal active set for all k sufficiently large, [84]. If this property does not hold, the active set might not settle down at the optimal one, [84], which means that constraint indices might repeatedly leave and enter the active set on consecutive iterations. Convergence properties of gradient projection methods are further discussed in, for example, [40], [41], and in [54].

The gradient projection algorithm has previously been successfully applied to optimal control problems. See for example [33] and [34]. In both references, Riccati recursions are used to decrease the computational complexity of certain parts of the algorithm. The conclusion in [34] is that the algorithm performs very well for large scale optimal control problems with "simple constraints" (that is, box constraints).

In Paper D, it is shown that the dual problem to an MPC problem can be interpreted as a new MPC problem with lower bound constraints on some of the "control signals". Since the resulting QP problem has simple constraints, a gradient projection method is expected to perform well when applied to this dual problem. In the paper, approximate Newton steps are used to solve the subproblems in the form in (2.42). The computations performed during this part of the algorithm are performed efficiently using Riccati recursions.

2.6 Semidefinite Programming

In this section, Semidefinite Programming (SDP) will be introduced, and some basic properties will be discussed. The presentation is based on those in [39] and [106], and the reader is referred to these two books for further details. SDP problems are optimization problems that can be written in the form

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \operatorname{tr} CX \\ \text{subject to} \quad & \operatorname{tr} A_i X = b_i,\ i = 1, \dots, p \\ & X \succeq 0 \end{aligned} \tag{2.43}$$

where the variable $X \in \mathbb{S}^n$, and the constants $C, A_1, \dots, A_p \in \mathbb{S}^n$. The problem in (2.43) is said to be an SDP problem in standard form. As in linear programming, there are several different alternative, but equivalent, standard forms. Note that, the SDP problems studied in Paper E and in Paper F can be transformed into the form in (2.43). The SDP problem can be seen as a generalization of the Linear Programming (LP) problem, and there exist many similarities between LPs and SDPs. However, SDP is much more general and a wide span of nonlinear convex optimization problems can be formulated and efficiently solved as SDPs. There are several freely available SDP solvers, for example, SDPT3 and SEDUMI. An excellent interface to a large number of solvers (SDP solvers as well as other solvers) is YALMIP, [74].

As in the QP case, it is sometimes interesting to study the dual problem. The problem dual to the one in (2.43) is, [106],

$$\begin{aligned} \underset{y,Z}{\text{maximize}} \quad & b^T y \\ \text{subject to} \quad & \sum_{i=1}^{p} y_i A_i + Z = C \\ & Z \succeq 0 \end{aligned} \tag{2.44}$$

where the variables $y \in \mathbb{R}^p$ and $Z \in \mathbb{S}^n$. Strong duality holds between the primal problem in (2.43) and the dual problem in (2.44) if the problem in (2.43) is strictly feasible.

2.6.1 Interior Point Methods

As already has been seen in Section 2.5, inequality constrained optimization problems are harder to solve than unconstrained or linear equality constrained problems. Especially, unconstrained or linear equality constrained QP problems can be solved by solving a linear system of equations. As discussed in Section 2.5.1, active set methods provide a method to solve the inequality constrained QP problem by solving a sequence of linear equality constrained QP problems. Interior Point (IP) methods provide an alternative solution strategy to inequality constrained optimization problems. The two methods share the idea of reducing the harder inequality constrained optimization problem into a sequence of easier linear equality constrained quadratic problems. IP methods are applicable to a wide variety of convex optimization problems with twice differentiable objective function, for example, LP problems, QP problems, Quadratically Constrained Quadratic Programming

Algorithm 2.4 Barrier method, [39]

```
Given $\mu > 1$, initial barrier parameter $s_0$, max number of iterations $k_{max}$, and tolerance $\varepsilon > 0$.
Compute a strictly feasible starting point $X$.
$s \leftarrow s_0$
while $k < k_{max}$ do
  Compute $X^*(s)$ by solving the problem in (2.46) using
  Newton's method starting at $X$. // Centering step
  $X \leftarrow X^*(s)$ // Update
  if $\bar{\theta}/s < \varepsilon$ then
    STOP
  end if
  $s \leftarrow \mu s$ // Improve constraint approximation
end while
```

(QCQP) problems and Geometric Programming (GP) problems. After some extensions, IP methods can also be applied to SDP problems. In this thesis, the focus will be on how they can be applied to SDP problems. For a more general presentation, see [39].

When an IP method is applied to a problem in the form in (2.43), the inequality constraints in the original problem are approximated using a barrier function. The barrier function approaches infinity as the solution approaches the boundary of the feasible set of the problem. The barrier function used for this problem is the generalized logarithmic barrier function for the positive semidefinite cone $\mathbb{S}_+^n$, that is

$$\psi(X) = \log\det X \tag{2.45}$$

By adding this barrier function to the objective function and introducing a barrier parameter $s \in \mathbb{R}_{++}$, subproblems in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & s \operatorname{tr} CX - \log\det X \\ \text{subject to} \quad & \operatorname{tr} A_i X = b_i, \ i = 1, \ldots, p \end{aligned} \tag{2.46}$$

are formed and solved for a fixed s using Newton's method, which in turn solves the nonlinear problem by solving a sequence of, possibly equality constrained, quadratic problems. Hence, the solution of the original nonlinear generalized inequality constrained optimization problem has been broken down into the solution of a sequence of linear equation systems. The accuracy of the approximation of the inequality constraints by the barrier function is controlled by the choice of the parameter s. A sequence of approximations with increasing accuracy in the approximation of the inequality constraints are solved. The problem in (2.46) is sometimes called the centering problem, and the path defined by the set of points $X^*(s)$, for $s > 0$, is called the central path. A basic algorithm for a barrier method can be found in Algorithm 2.4, where $\bar{\theta}$ denotes the sum of the degrees (see [39]) of the generalized logarithms for the cones. The degree of the generalized logarithm in (2.45), is n, [39]. For a discussion of how to choose the parameters μ and s_0, see [39].

The Lagrangian for the problem in (2.46) is

$$L(X,\nu) = s\,\mathrm{tr}\,CX - \log\det X + \sum_{i=1}^{p} \nu_i \left(\mathrm{tr}\,A_iX - b_i\right) \tag{2.47}$$

Hence, the KKT conditions for the centering problem in (2.46) are

$$\begin{aligned} sC - X^{-1} + \sum_{i=1}^{p} \nu_i A_i &= 0 \\ \mathrm{tr}\,A_iX &= b_i,\ i = 1,\ldots,p \end{aligned} \tag{2.48}$$

and to get the Newton step to the centering problem, the equations in (2.48) are linearized at a point X^0. The resulting linear system of equations is

$$\begin{aligned} X^0 \Delta X X^0 + \sum_{i=1}^{p} \nu_i A_i &= X^0 - sC \\ \mathrm{tr}\,A_iX &= b_i,\ i = 1,\ldots,p \end{aligned} \tag{2.49}$$

The solution to this system of equations gives the Newton step for the centering problem and it is during the solution process of this linear system the main effort is spent in the barrier method in Algorithm 2.4. In Paper F, it is shown how this system can be solved efficiently for the SDP relaxation of the MIPC problem by utilizing problem structure. In the barrier method and in primal-dual methods, the number of iterations necessary to reach the optimum is *in practice* roughly independent of the problem size. Hence, the computational complexity grows in practice roughly as the computational complexity of solving the problem in (2.49). It should be mentioned that even though the proposed Riccati method in Paper F for simplicity is illustrated on a basic barrier method like the one in Algorithm 2.4, it is also applicable to a primal-dual IP method as the one described in [101]. The reason for this is that the Newton-like equations necessary to solve in a primal-dual method are essentially in the same form as the ones in a barrier method (the block structure utilized in Paper F is preserved). This is further discussed in [39].

2.7 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 in (2.27). 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, [107]. 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 [103].

2.7.1 Problem Definition

The mathematical definition of an MIQP problem is

$$\begin{aligned} \underset{x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x \\ \text{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}} \\ & A_{\mathcal{I}} x \leq b_{\mathcal{I}} \end{aligned} \tag{2.50}$$

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 (2.27) 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, [22]:

- 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, [22]. In [52], 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 [103]. 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.

An important special case of MIQP is Binary Quadratic Programming (BQP), where only binary variables are present. A BQP problem in the form

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

where $H \in \mathbb{S}_+^{n_b}$ and $f \in \mathbb{R}^{n_b}$ is used in Paper A and in Paper B. The BQP problem is known to be $\mathcal{NP}$-hard, [69]. Most algorithms for this kind of problems either focus on producing approximative solutions or on only handling various special cases of the general problem, [55]. Some approximative heuristic algorithms can be found in, for example, [69], [17], [77] and [56]. 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, [77].

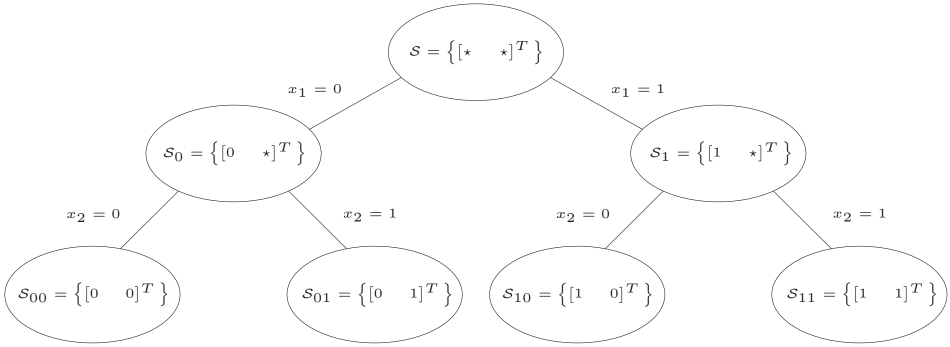

$\mathcal{S} = \left\{ [\star \quad \star]^T \right\}$
$x_1 = 0$
$x_1 = 1$
$\mathcal{S}_0 = \left\{ [0 \quad \star]^T \right\}$
$\mathcal{S}_1 = \left\{ [1 \quad \star]^T \right\}$
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
$\mathcal{S}_{00} = \left\{ [0 \quad 0]^T \right\}$
$\mathcal{S}_{01} = \left\{ [0 \quad 1]^T \right\}$
$\mathcal{S}_{10} = \left\{ [1 \quad 0]^T \right\}$
$\mathcal{S}_{11} = \left\{ [1 \quad 1]^T \right\}$

Figure 2.3: *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.

2.7.2 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}.

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 [107] and [53].

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

$$\mathcal{S} = \bigcup_{i=1}^{K} \mathcal{S}_i \tag{2.52}$$

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 2.3. The tree in Figure 2.3 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

$$\begin{aligned} z^{i*} &= \underset{x \in \mathcal{S}_i}{\text{minimize}}\; f_0(x),\ i \in \{1, \ldots, K\} \\ z^* &= \min_{i \in \{1,\ldots,K\}} \left\{ z^{i*} \right\} \end{aligned} \tag{2.53}$$

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 2.3 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, 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 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 2.4. 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 2.4, 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

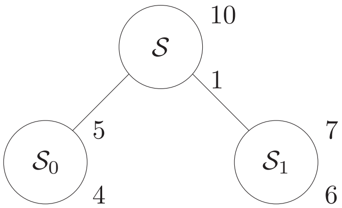


Figure 2.4: *This figure is used to illustrate how bounds can be used to prune nodes. Assume that the tree originates from a minimization problem. Since the upper bound over $\mathcal{S}_0$ is lower than the lower bound over $\mathcal{S}_1$, $\mathcal{S}_1$ cannot contain the optimal solution.*

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 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, often QP relaxations are used, where the integer constraints are relaxed to interval constraints. More about relaxations applicable to branch and bound for MIQP can be found in Section 2.8.

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 case 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 to the relaxed problem is a feasible solution to the original problem, then this solution is also an optimal solution to the original problem. This can be used to prune according to case 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 choices to make 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, [21].

It is also common to let the user 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.

Algorithm 2.5 Branch and bound for binary variables

```
$\bar{z} \leftarrow +\infty$
$\bar{x} \leftarrow$ **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 $P_i$.
    $\bar{z} \leftarrow \underline{z}^i$
    $\bar{x} \leftarrow \underline{x}^i$
  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
```

According to [52], 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, [52].

According to [107], 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, [107]. 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 2.5. 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 $\bar{z}$.

2.8 Relaxations Applicable to BQP and MIQP Problems

Consider the BQP problem in (2.51), which is here repeated for convenience,

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

For simplicity, a pure binary example is considered, but the results are also applicable to the more general MIQP problem in (2.50). Both problems are non-convex and are in general hard to solve exactly. As mentioned before, problems in this form can be solved using branch and bound where relaxations are solved in the nodes of a search tree. As discussed in Section 2.7.2, a branch and bound algorithm relies on efficient computations of upper and lower bounds in order to cut away parts of the tree. If the relaxation is able to provide a stronger bound, there is a higher possibility that condition 3 on page 36 can be satisfied. The more successful this operation is, the smaller part of the tree has to be explicitly explored and fewer relaxations have to be solved. Basically, upper bounds are found from integer feasible solutions and lower bounds are found from relaxations or duality. There are several non-trivial choices to make when designing a branch and bound algorithm. For example, when choosing the type of relaxation, a trade-off often has to be made between low computational effort and strength of the computed bound, [107].

In this section, two different alternative relaxations are presented. The QP relaxation is presented in Section 2.8.1 and the SDP relaxation is presented in Section 2.8.2. On one hand, the QP relaxation is relatively cheap to compute, but it is expected to give a rather loose lower bound. On the other hand, the SDP relaxation is expected to give a sharper lower bound, but it is far more computationally demanding. These expectations are verified by the numerical experiments in Paper E and in Paper F. The idea in these two papers is to try to find a way to compute the SDP relaxation more efficiently in order to make it less computationally demanding, and possibly in the future, make it useful in branch and bound. Note that, even thought the computational time can be reduced as presented in Paper E and in Paper F, more work remains to make it useful in branch and bound. For example, it is probably necessary that the warm start properties of IP methods are improved, which is still a research area. In Paper E, its capability of reducing the number of nodes explored in the branch and bound tree is illustrated for a special case.

2.8.1 QP Relaxations

As discussed in Section 2.7, commonly the relaxations used in branch and bound for MIQP problems are of QP type. The QP relaxation is created by relaxing the integer constraints to interval constraints. That is, if the binary variable indexed by j is relaxed, the constraint

$$x_j \in \{0, 1\} \tag{2.54}$$

is replaced by

$$x_j \in [0,1] \tag{2.55}$$

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.

2.8.2 SDP Relaxations

Recent research has shown that the so-called SDP relaxation, or moment relaxation, [72, 106], of the BQP problem can provide good bounds for certain instances of the problem, [57]. The first work was presented for the Max Cut problem in [57], and this result has been refined in several articles, for example, [82], [83] and [110]. Furthermore, there exist randomization methods to produce sub-optimal solutions from the solution to the relaxed problem, [57]. However, a drawback with the standard SDP relaxation is that the number of variables grows fast and, as a consequence, it soon becomes rather computationally demanding. In Paper E, sparsity in the problem is used to lower the computational complexity of the computation of the SDP relaxation. Similar sparsity has previously been used for large SDP relaxations in, for example, [70], but it has not until now been used for relaxations of control problems involving binary variables.

By rewriting the binary constraint on variable i in the equivalent form $x_i^2 = x_i$ and introducing a Lagrangian multiplier vector $\gamma \in \mathbb{R}^{n_b}$, the dual problem to the one in (2.51), can be found to be

$$\begin{aligned} \underset{\gamma}{\text{maximize}} \quad & -\frac{1}{2}(f-\gamma)^T (H + 2\operatorname{diag}\gamma)^{\dagger} (f-\gamma) \\ \text{subject to} \quad & H + 2\operatorname{diag}(\gamma) \succeq 0 \\ & f - \gamma \in \operatorname{range}(H + 2\operatorname{diag}(\gamma)\,) \end{aligned} \tag{2.56}$$

which can be written in epigraph form as

$$\begin{aligned} \underset{t}{\text{minimize}} \quad & \frac{1}{2} t \\ \text{subject to} \quad & \begin{bmatrix} H + 2\operatorname{diag}(\gamma) & f-\gamma \\ (f-\gamma)^T & t \end{bmatrix} \succeq 0 \end{aligned} \tag{2.57}$$

by using the Schur complement formula in Lemma A.2, where $t \in \mathbb{R}$. Note that, strong duality does not in general hold between the non-convex primal problem in (2.51) and the dual problem in (2.57). However, since weak duality always holds, the optimal objective function value of the dual problem in (2.57) will always be lower or equal to the one of the problem in (2.51). Hence, a lower bound on the optimal objective function value of the non-convex optimization problem in (2.51) can be found from the optimal solution to the convex optimization problem in (2.57).

The dual problem of the dual problem in (2.57) is

$$\begin{array}{ll} \underset{Z,z}{\text{minimize}} & \operatorname{tr}(HZ) + 2f^T z \\ \text{subject to} & Z_{ii} = z_i,\ i = 1, \dots, n_b \\ & \begin{bmatrix} Z & z \\ z^T & \frac{1}{2} \end{bmatrix} \succeq 0 \end{array} \tag{2.58}$$

where $Z \in \mathbb{S}^{n_b}$ and $z \in \mathbb{R}^{n_b}$, or equivalently

$$\begin{array}{ll} \underset{Y,y}{\text{minimize}} & \frac{1}{2} \operatorname{tr}(HY) + f^T y \\ \text{subject to} & Y_{ii} = y_i,\ i = 1, \dots, n_b \\ & \begin{bmatrix} Y & y \\ y^T & 1 \end{bmatrix} \succeq 0 \end{array} \tag{2.59}$$

where $Y \in \mathbb{S}^{n_b}$ and $y \in \mathbb{R}^{n_b}$. Strong duality holds between the convex problems in (2.57) and in (2.58). Hence, the optimal objective function value of these two problems will coincide. However, since strong duality does not in general hold between the problem in (2.51) and the dual problem in (2.57), the optimal objective function value of (2.59) will only provide a lower bound of the optimal solution to the problem in (2.51). An interpretation of the problem in (2.59) is that it is a semidefinite rank-relaxation of the original non-convex optimization problem, [43]. The problem in (2.59) is commonly called the SDP relaxation, or moment relaxation, of the problem in (2.51). In Paper E and in Paper F, efficient computations of these SDP relaxations are considered for the MIPC problem.

3

Mixed Integer Predictive Control

In this chapter, Model Predictive Control (MPC) and hybrid systems in 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, [75]. An early academic publication containing the basic ideas was presented by Propoi 1963, [75]. There are also some methods similar to MPC, but with different names. One of the most well-known is Dynamic Matrix Control (DMC), [42].

The most commonly used variant of MPC is 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 control 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

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

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) in the form of a formal optimization problem. The most common variants are presented and evaluated in [73]. If the system is linear and the objective is quadratic, the resulting optimization problem is a QP (see Section 2.5), 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.

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

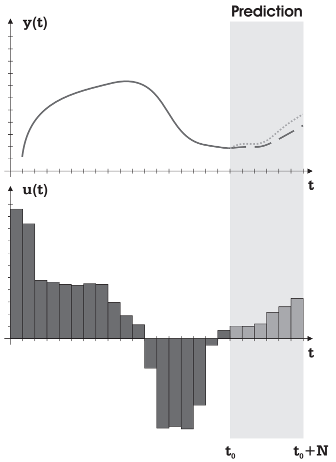


Figure 3.1: *In this figure, an example of a control input $u(t)$ and the corresponding controlled output $y(t)$ is given. The controller is about to compute the control signal in time t_0. The figure illustrates how the behavior of the system is predicted N steps. In the predicted interval, the dotted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

The notation and the derivations of the formulations can be found in Appendix B. The two optimization problems are equivalent (see Definition 2.4). However, from a computational point of view, the formulation in (3.4) gives a sparse optimization problem with $(N+1)(n+p)+Nm$ optimization variables, while the one in (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 nonlinear MPC. This extension handles nonlinear systems and a general nonlinear performance measure in the objective function. Unfortunately, the resulting optimization problem is more difficult to solve in general.

A special case of nonlinear 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, [22]. This class of systems is especially important when analyzing and controlling sys-

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: $t_0 \leftarrow t_0 + 1$
5: Repeat the procedure.
```

tems arising in the growing interaction between physical processes and digital controllers.

A survey covering both linear and nonlinear MPC is found in [76]. A reference book covering most of MPC is [75].

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, [28]. In [22], 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 nonlinear 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 [22] 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 [49] and [50] 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 [8], 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, [22],

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

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 [22]. 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, [19]. In [27], the formal equivalence between MLD systems and Piecewise Affine (PWA) systems is established. In [65, 66], 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 [44]. The important result of these equivalences is that derived theoretical properties and tools can easily be transferred from one class to another, [30]. 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, [19]. 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, [19]. The note [19] presents an efficient conversion algorithm from MLD to PWA form. In [22], 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 [78]. An alternative approach is to create the MLD formulation automatically by using the high-level modeling language HYSDEL (Hybrid Systems Description Language), [97].

3.2.3 Controlling MLD Systems

In [22], both optimal control and receding horizon estimation for MLD systems is discussed. The control signal is found by minimizing a quadratic performance criterion in the form

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

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, [22].

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, [22]. 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, [22].

Interesting work is presented in [86], 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 [87] 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, [24]. 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 [29]. 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, [88]. 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. For a further description of branch and bound, see Section 2.7.2.

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 [23].

3.3.1 Quadratic Programming

In many applications, even linear MPC is considered computationally expensive. At each time step, either a QP in 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 [68], 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 nonlinear problems. A similar method is presented in [2]. The special structure of the problem has also been used in Interior Point (IP) methods. In [108], an IP solver utilizing block-bandedness is presented. The approach is refined in [109], 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 [64], an infeasible primal-dual IP method utilizing Riccati recursions for the computation of the search directions has been applied to robust MPC. In [36] 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 [91]. Another reference on the same topic is [35]. One way of establishing stability for MPC is to introduce an ellipsoidal terminal state constraint, [112]. 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 [112] how Riccati recursion can be used to decrease the complexity of the calculations of the search directions in the IP algorithm. In [99, 100], 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 [104].

A thorough comparison between four QP solvers for MPC control of the cross directional control in a paper machine is performed in [15]. 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 [88] 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 [13].

Another way of reducing the on-line computational effort is to precompute the control law. Briefly, the procedure can be described as that the state-space is partitioned into a number of regions, where in each a different affine control law is optimal. The partitioning and the parameters in the affine control laws are computed off-line, by solving a multi-parametric programming problem, where multi-parametric means that the problem depends on a vector of parameters, [96]. The on-line computations are basically to identify the correct region in the state-space, retrieve the control law parameters for that region, and to evaluate the affine control law given the precomputed parameters. 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, [62]. Therefore, several approaches have been developed in order to reduce the complexity. Some references are [62, 63, 95, 96]. 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 [31], for example, regions where saturation occurs can be detected. A summary of the theory behind explicit MPC for quadratic objective is found in [32]. The counterpart for problems with linear objective is found in [31].

In [37], 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, [108]. 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 $\mathcal{O}(N^2)$, [108]. The computational complexity for the IP algorithm presented in [108] 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, [14], while the active set algorithm is a combinatorial algorithm which in the worst case has a complexity higher than polynomial, [59]. 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, 59]. In these references they are used in combination with a Sequential Quadratic Programming (SQP) solver for nonlinear MPC. In [108], the author comments that gradient projection algorithms are advantageous for the optimal control application, especially when the constraints are in a simple form with upper and lower bound constraints on the control signals. In Paper D, this property is utilized for MPC problems with more general constraints. In that report, this is accomplished by considering the dual problem, which can be interpreted as another MPC problem, in which the constraints are of the type that is suitable for the gradient projection method.

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 [109], active set methods gain more from warm starts than IP methods. Warms starts for IP methods is still an open question, [14]. 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, [109]. 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 [109], 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 [79, 81], 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, [80]. 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 [22], a commercial Fortran package has been used as an MIQP solver. The package is capable of coping with dense as well as sparse MIQP problems. The control problems considered normally lead to sparse optimization problems, which means that a solver utilizing sparsity is preferable.

In [28], 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 [23]. 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 [23] shows that the outside first tree exploring strategy produces, for MPC applications, better suboptimal solutions 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 [20]. 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 [67]. The motivation is when an automaton is converted into mixed integer inequality constraints, the relaxed problems in the branch and bound algorithm become unnecessarily loose. To get a tighter relaxation, new equality constraints are introduced.

In [18], 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, [92] and [71].

Another attempt to reduce the complexity of the MIQP problem is found in [94], 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 remain constant. The result is an MIQP problem with fewer binary variables to compute.

In [26] 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 [25]. A thorough reference on the subject multi-parametric MIQP is [48], where theory and algorithms for mp-QP, mp-LP and mp-MIQP are presented.

According to [38], 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 [61].

3.4 Two Examples of Mixed Logical Dynamical Systems

In this section, two simple examples of MLD systems are given. These systems are used as benchmark examples in several papers in Part II of this thesis.

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 in a binary way 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

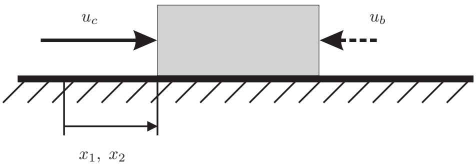


Figure 3.2: *This figure illustrates the mass modeled in* (3.11)*, where x_1 denotes the velocity of the mass, x_2 the position of the mass, u_c a continuously controlled force and u_b a binary controlled force.*

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. This example is used in Paper A and Paper C.

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 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 for the inertial forces induced when the wheel is braked.

The wheel is controlled continuously by an electric motor. 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 has been used. This example is used in Paper A, Paper C and Paper D.

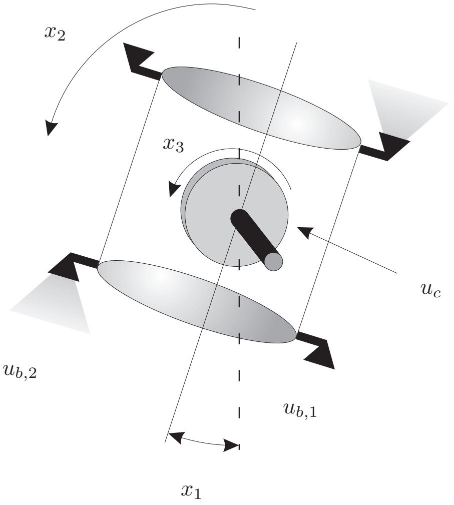


Figure 3.3: *This figure illustrates the satellite modeled in* (3.12)*, where x_1 is the satellite attitude, x_2 is the satellite angular velocity and x_3 the angular velocity of the reaction wheel. The control signals are u_c, $u_{b,1}$ and $u_{b,2}$, which control the electric motor and the two oppositely 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, [93]. 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 in 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 2.7). If the signature sequences produce a cross-correlation matrix with some special structures, the problem can however sometimes turn out to have lower complexity, [89, 90, 98].

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, [102].

The notation used in this thesis is chosen similar to the one used in [102]. 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

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

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, [102]. Furthermore, the unnormalized cross-correlation matrix is denoted as

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

In the synchronous case, 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. However, there is sometimes also a need to study an asynchronous model. The basic ideas in this model are shared with the synchronous model, but users do not have to be synchronized in time. This results in a more complicated cross-correlation between the users. Both models are consider in Paper B.

4.2.2 Derivation of the BQP Problem

The matched filter output is described in the equation in (4.7). According to [102], 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

$$\Omega(b) = 2\int\limits_0^T \left[\sum_{k=1}^K A_k b_k s_k(t)\right] y(t)\, dt - \int\limits_0^T \left[\sum_{k=1}^K A_k b_k s_k(t)\right]^2 dt = 2b^T A y - b^T H b \tag{4.12}$$

where A, H, b and y are defined in (4.9) and in (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

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

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

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 [39, 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 Schur complement formula has the following generalization in the case when T_{11} is singular. It is based on the discussion in [39, p. 651] and given without any proof.

Lemma A.2 (Schur complement formula with singular T_{11})
Assume T symmetric. Then
$T \succeq 0 \Leftrightarrow T_{11} \succeq 0,\ \left(I - T_{11} T_{11}^{\dagger}\right) T_{12} = 0,\ T_{22} - T_{12}^{T} T_{11}^{\dagger} T_{12} \succeq 0.$

The following matrix inversion lemma is taken from [111].

Lemma A.3 (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.4
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 [39].

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 in (3.2) subject to the dynamics in (3.1) and the constraints in (3.3) is cast in the form of a QP problem, that is (2.27). This can be done in several ways. See, for example, [73]. 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:

$$\begin{aligned}
\mathsf{x} &= \left[x^T(t_0), x^T(t_0+1), \ldots, x^T(t_0+N)\right]^T \\
\mathsf{u} &= \left[u^T(t_0), u^T(t_0+1), \ldots, u^T(t_0+N-1)\right]^T \\
\mathsf{r} &= \left[r^T(t_0), r^T(t_0+1), \ldots, r^T(t_0+N)\right]^T \\
\mathsf{Q_e} &= \operatorname{diag}\left(Q_e, \ldots, Q_e\right),\ \mathsf{Q_u} = \operatorname{diag}\left(Q_u, \ldots, Q_u\right),\ \mathsf{C} = \operatorname{diag}\left(C, \ldots, C\right) \\
\mathsf{H_u} &= \operatorname{diag}\left(H_u(0), \ldots, H_u(N-1)\right),\ \mathsf{H_x} = \operatorname{diag}\left(H_x(0), \ldots, H_x(N)\right) \\
\mathsf{h} &= \left[h^T(0), \ldots, h^T(N)\right]^T
\end{aligned} \tag{B.1}$$

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. Furthermore, the Sans Serif font is used to indicate that a matrix or a vector is, in some way, composed by stacked matrices or vectors from different time instants. The stacked matrices or vectors have a similar symbol as the composed matrix, but in an ordinary font. Several examples of this notation can be found in (B.1).

B.1 Complete Set of Variables

The most straightforward way to cast the MPC problem in the form of a QP is to keep the dynamic equations as equality constraints. The MPC problem is then formulated as a QP in the form

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

where

$$
\begin{aligned}
\mathsf{e} &= \left[e^T(t_0), e^T(t_0+1), \ldots, e^T(t_0+N)\right]^T\\
b &= \begin{bmatrix}-x_0^T & 0 & \ldots & 0\end{bmatrix}^T\\
\mathsf{A} &= \begin{bmatrix}-I & 0 & 0 & \ldots & 0 & 0\\ A & -I & 0 & \ldots & 0 & 0\\ 0 & A & -I & \ldots & 0 & 0\\ 0 & 0 & A & \ldots & 0 & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \ldots & A & -I\end{bmatrix}, \quad \mathsf{B} = \begin{bmatrix}0 & 0 & \ldots & 0\\ B & 0 & \ldots & 0\\ 0 & B & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & B\end{bmatrix}
\end{aligned}
\tag{B.3}
$$

This formulation involves $N(n+m+p)$ 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

$$
\mathsf{x} = S_x x_0 + S_u \mathsf{u} \tag{B.4}
$$

where

$$S_x = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; S_u = \begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \tag{B.5}$$

The equality constraints are now eliminated and the objective function can be written as

$$\begin{aligned} J &= \left(\mathsf{C}\left(S_x x_0 + S_u \mathsf{u}\right) - \mathsf{r}\right)^T \mathsf{Q}_\mathsf{e} \left(\mathsf{C}\left(S_x x_0 + S_u \mathsf{u}\right) - \mathsf{r}\right) + \mathsf{u}^T \mathsf{Q}_\mathsf{u} \mathsf{u} \\ &= \mathsf{u}^T \left(S_u^T \mathsf{C}^T \mathsf{Q}_\mathsf{e} \mathsf{C} S_u + \mathsf{Q}_\mathsf{u}\right) \mathsf{u} + 2\left(S_u^T \mathsf{C}^T \mathsf{Q}_\mathsf{e} \mathsf{C} S_x x_0 - S_u^T \mathsf{C}^T \mathsf{Q}_\mathsf{e} \mathsf{r}\right)^T \mathsf{u} + \kappa \end{aligned} \tag{B.6}$$

where κ is a constant. By using (B.4), the inequality constraints can be written as

$$\mathsf{H}_\mathsf{x} \mathsf{x} + \mathsf{H}_\mathsf{u} \mathsf{u} + \mathsf{h} = \mathsf{H}_\mathsf{x} S_u \mathsf{u} + \mathsf{H}_\mathsf{u} \mathsf{u} + \mathsf{h} + \mathsf{H}_\mathsf{x} S_x x_0 \leq 0 \tag{B.7}$$

Ignoring the constant and dividing by two gives the equivalent optimization problem

$$\begin{aligned} \underset{\mathsf{u}}{\text{minimize}} \quad & \frac{1}{2}\mathsf{u}^T \left(S_u^T \mathsf{C}^T \mathsf{Q}_\mathsf{e} \mathsf{C} S_u + \mathsf{Q}_\mathsf{u}\right) \mathsf{u} + \left(S_u^T \mathsf{C}^T \mathsf{Q}_\mathsf{e} \left(\mathsf{C} S_x x_0 - \mathsf{r}\right)\right)^T \mathsf{u} \\ \text{subject to} \quad & \left(\mathsf{H}_\mathsf{x} S_u + \mathsf{H}_\mathsf{u}\right) \mathsf{u} + \mathsf{h} + \mathsf{H}_\mathsf{x} S_x x_0 \leq 0 \end{aligned} \tag{B.8}$$

In this formulation, the Hessian becomes dense and Nm optimization variables are involved.

C

Definition of Order of Convergence

When the performance of different optimization methods are compared, the order of convergence is often discussed. It specifies how fast the distance to the optimal solution decreases at each iteration. The definitions in this appendix are summarized from the presentation in [84]. Generally, the convergence rate is said to be p (where $p > 1$) if there exists a positive constant M such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} \leq M \tag{C.1}$$

for all k sufficiently large.

The most common orders of convergence, or convergence rates, are linear ($p = 1$), superlinear and quadratic ($p = 2$). The convergence rate is said to be superlinear if the following condition is satisfied

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0 \tag{C.2}$$

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. *Applications of Integer Quadratic Programming in Control and Communication*. Licentiate's thesis, Linköpings universitet, 2005. URL `http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263`.

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

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

[6] D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698, Manchester Grand Hyatt, San Diego, USA, Dec. 2006.

[7] D. Axehill and A. Hansson. A dual gradient projection quadratic programming algorithm tailored for mixed integer predictive control. Technical Report LiTH-ISY-R-2833, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Dec. 2007.

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

[9] 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.

[10] D. Axehill, A. Hansson, and L. Vandenberghe. Relaxations applicable to mixed integer predictive control – comparisons and efficient computations. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4103–4109, Hilton New Orleans Riverside, New Orleans, USA, Dec. 2007.

[11] D. Axehill, L. Vandenberghe, and A. Hansson. On relaxations applicable to model predictive control for systems with binary control signals. In *Preprints of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 200–205, Pretoria, South Africa, Aug. 2007.

[12] D. Axehill, F. Gunnarsson, and A. Hansson. A low-complexity high-performance preprocessing algorithm for multiuser detection using Gold sequences. Provisionally accepted for publication in *IEEE Transactions on Signal Processing*, Jan. 2008.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] D. P. Bertsekas. On the Goldstein – Levitin – Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174 – 184, Apr. 1976.

[34] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221 – 246, Mar. 1982.

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

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

[37] F. Borrelli, M. Baotić, 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.

[38] 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.

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

[40] J. V. Burke and J. J. Moré. Exposing constraints. *SIAM Journal on Optimization*, 4(3):573 – 595, Aug. 1994.

[41] A. R. Conn, N. I. M. Gould, and P. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433 – 460, Apr. 1988.

[42] 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.

[43] J. Dahl, B. H. Fleury, and L. Vandenberghe. Approximate maximum-likelihood estimation using semidefinite programming. In *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003*, volume 6, pages VI – 721–724, Apr. 2003.

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

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

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

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

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

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

[54] E. M. Gafni and D. P. Bertsekas. Convergence of a gradient projection method. Technical Report LIDS-P-1201, Laboratory for Information and Decision Systems Massachusetts Institute of Technology, 1982.

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

[56] 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.

[57] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing, STOC'94*, pages 422–431, Montréal, Québec, Canada, May 1994.

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

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

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

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

[66] 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.

[67] 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 43rd IEEE Conference on Decision and Control*, pages 1866 – 1871, Dec. 2004.

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

[69] 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.

[70] S. Kim, M. Kojima, and H. Waki. Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, 15(3):697 – 719, 2005.

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

[72] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796 – 817, 2001.

[73] 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.

[74] J. Löfberg. Yalmip: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. URL `http://control.ee.ethz.ch/~joloef/yalmip.php`.

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

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

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

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] Y. Nesterov. Quality of semidefinite relaxation for nonconvex quadratic optimization. Technical report, CORE, Universite Catholique de Louvain, Belgium, 1997.

[83] Y. Nesterov. Global quadratic optimization via conic relaxation. Technical report, CORE, Universite Catholique de Louvain, Belgium, 1998.

[84] J. Nocedal and S. J. Wright. *Numerical Optimization – Second Edition*. Springer Verlag, 2006.

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

[86] 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.

[87] 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.

[88] 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.

[89] 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.

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

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

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

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

[94] 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.

[95] 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.

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

[97] 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.

[98] 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.

[99] 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.

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

[101] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. *Interior-point algorithms for semidefinite programming problems derived from the KYP lemma*, volume 312 of *Lecture notes in control and information sciences*, pages 195 – 238. Springer, Feb. 2005.

[102] S. Verdu. *Multiuser Detection*. Cambridge University Press, 1998.

[103] 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.

[104] 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.

[105] P. Wolfe. A duality theorem for non-linear programming. *Quarterly of Applied Mathematics*, 19(3):239 – 244, 1961.

[106] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming – Theory, Algorithms and Applications.* Kluwer, 2000.

[107] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

[108] S. J. Wright. Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77:161 – 187, 1993.

[109] 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.

[110] Y. Ye. Approximating quadratic programming with bound constraints. *Mathematical Programming*, 84:219 – 226, 1999.

[111] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, 1996.

[112] M. Åkerblad. *A Second Order Cone Programming Algorithm for Model Predictive Control.* Licentiate's thesis, Royal Institute of Technology, 2002.

Part II

Publications

Paper A

A Preprocessing Algorithm for MIQP Solvers with Applications to MPC

Authors: Daniel Axehill and Anders Hansson

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

Preliminary version: Published as Technical Report LiTH-ISY-R-2607, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

A Preprocessing Algorithm for MIQP Solvers with Applications to MPC[†]

Daniel Axehill and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,hansson}@isy.liu.se

Abstract

In this paper a preprocessing algorithm for unconstrained mixed integer quadratic programming problems and binary quadratic programming problems is presented. The optimal value for some or all integer variables can be computed without approximations in polynomial time. The algorithm is first derived for the binary quadratic programming problem and the result is then extended to the mixed integer quadratic programming problem by transforming the latter problem into the first problem. Both mentioned quadratic programming problems have several important applications. In this paper, the focus is on model predictive control problems with both real valued and binary valued control signals. As an illustration of the method, the algorithm is applied to two different problems of this type.

1 Introduction

An extension to ordinary Model Predictive Control (MPC) is to allow binary control signals, see for example [3]. When this generalization is made, the ordinary Quadratic Programming (QP) solver has to be replaced with a so-called Mixed Integer Quadratic Programming (MIQP) solver, which also is able to optimize binary variables. Except for problems with particular structures, the MIQP problem is known to have exponential worst case complexity, [4]. One way of reducing the average complexity is to use an algorithm which 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. In this report we have investigated if the performance of the branch and

[†] Research supported by the Swedish Research Council for Engineering Sciences under contract Nr. 621-2002-3822.

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, e.g., [9].

In the first part of the derivation of the main result, a so-called Binary Quadratic Programming (BQP) problem is studied. The BQP problem is known to be $\mathcal{NP}$-hard, [7]. Most algorithms for this kind of problems either focus on producing approximative solutions or on only handling various special cases of the general problem, [5]. The algorithm presented in this paper belongs to the latter type of algorithms. Some approximative heuristic algorithms can be found in, e.g., [7], [1], [8] and [6].

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, [8].

The results from the first part of the paper are then extended to an MIQP problem applicable to MPC problems with both binary valued and real valued control variables.

This paper is organized as follows. In Section 2, the BQP and MIQP problems are defined. In Section 3 the preprocessing algorithm is derived for the BQP problem. A brief discussion of the implementation of the algorithm can be found in Section 4. Section 5 shows that the preprocessing algorithm can be extended to MIQP problems. In Section 6 the results are applied to the MPC problem. In Section 7, two examples of MPC are presented. Finally, section 8 contains conclusions and suggestions to future work.

2 The BQP and MIQP Problems

The MIQP problem can be seen as an ordinary QP problem for which some variables are constrained to be binary.

In this paper the problems considered are limited to those without any constraints except that some variables should be binary. We begin by considering a purely unconstrained BQP problem

$$\mathcal{P}_1 : \begin{cases} \min\limits_{x} & x^T H x \\ \text{subject to} & x \in \{0,1\}^{n_b} \end{cases} \tag{1}$$

where $H \in \mathbb{R}^{n_b \times n_b}$ is symmetric. The result is first extended to a BQP problem in the form

$$\mathcal{P}_2 : \begin{cases} \min\limits_{x} & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} & x \in \{0,1\}^{n_b} \end{cases} \tag{2}$$

where we have incorporated a linear term in the objective function. Then, the result is extended to the unconstrained MIQP problem

$$\mathcal{P}_3 : \begin{cases} \min\limits_{x} & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} & x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b} \end{cases} \tag{3}$$

The optimization vector x contains n_c real valued and n_b binary valued variables.

For what follows it is practical to define

$$H = H_d + H^+ + H^- \tag{4}$$

where

$$\begin{aligned} H_{d,ij} &= \begin{cases} H_{ij}, & i = j \\ 0, & i \neq j \end{cases} \\ H_{ij}^{+} &= \max\left(0, H_{ij} - H_{d,ij}\right) \\ H_{ij}^{-} &= \min\left(0, H_{ij} - H_{d,ij}\right) \end{aligned} \tag{5}$$

3 Preprocessing for the BQP Problem

To begin with, we consider the problem when all elements in x are binary. The problem is then of the so-called BQP type, which is a purely combinatorial problem. As mentioned in Section 1, a characteristic property of combinatorial problems is that they in general are very hard to solve exactly, because of the computational complexity, [8]. To be able to solve large problem instances either an approximate algorithm or some special structure in the problem has to be used. If the structure in the problem is used it might be possible to solve the problems exactly in a 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 a consequence of the following theorem:

Theorem 1
For a BQP problem of type $\mathcal{P}_1$ the optimal value of one or more components x_i can be found in polynomial time if for some $i \in \{1, .., n_b\}$ any of the following conditions is satisfied

$$\begin{cases} H_{ii} \geq -2\sum_{j=1}^{n_b} H_{ij}^{-} & (i) \\ H_{ii} < -2\sum_{j=1}^{n_b} H_{ij}^{+} & (ii) \end{cases}$$

If any of the conditions (i) or (ii) is satisfied for a certain value of i, the 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: Consider the optimization problem $\mathcal{P}_1$, where H is a symmetric $n_b \times n_b$ matrix. Denote the objective function by $Q(x)$ and rewrite it as follows

$$Q(x) = x^T H x = \sum_{i=1}^{n_b} \sum_{j=1}^{n_b} H_{ij} x_i x_j \tag{6}$$

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

$$\begin{aligned} Q(x) &= H_{ii}x_ix_i + 2x_i\sum_{\substack{j=1\\j\neq i}}^{n_b} H_{ij}x_j + g_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{n_b}) \\ &= (H_{ii} + 2\sum_{\substack{j=1\\j\neq i}}^{n_b} H_{ij}x_j)x_i + g_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{n_b}) \end{aligned} \tag{7}$$

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

$$h_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{n_b}) = H_{ii} + 2\sum_{\substack{j=1\\j\neq i}}^{n_b} H_{ij}x_j \tag{8}$$

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

$$Q(x) = h_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{n_b})x_i + g_i(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{n_b}) \tag{9}$$

Denote the optimal value of x with $x^* = \left[x_1^*, ..., x_{n_b}^*\right]^T$. It is also convenient to make the following definitions

$$\begin{aligned} h_i^* &= h_i(x_1^*, x_2^*, ..., x_{i-1}^*, x_{i+1}^*, ..., x_{n_b}^*), \\ g_i^* &= g_i(x_1^*, x_2^*, ..., x_{i-1}^*, x_{i+1}^*, ..., x_{n_b}^*) \end{aligned} \tag{10}$$

It now follows that

$$\min_x Q(x) = \min_{x_i} h_i^*x_i + g_i^* = \begin{cases} g_i^*, & \text{if } h_i^* \geq 0 \\ h_i^* + g_i^*, & \text{if } h_i^* < 0 \end{cases} \tag{11}$$

From (11) the conclusion can be drawn that

$$x_i^* = \begin{cases} 0, & \text{if } h_i^* \geq 0 \\ 1, & \text{if } h_i^* < 0 \end{cases} \tag{12}$$

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

$$\begin{cases} \overline{h}_i = \max_x h_i \\ \underline{h}_i = \min_x h_i \end{cases} \tag{13}$$

It now holds that $\underline{h}_i \leq h_i^* \leq \overline{h}_i$. From this observation the following implications can be stated

$$\begin{cases} \overline{h}_i < 0 & \Rightarrow h_i^* < 0 \\ \underline{h}_i \geq 0 & \Rightarrow h_i^* \geq 0 \end{cases} \tag{14}$$

By combining (12) and (14), the following conclusion can be drawn

$$x_i^* = \begin{cases} 0, & \underline{h}_i \geq 0 \\ 1, & \overline{h}_i < 0 \end{cases} \tag{15}$$

From (5) and (8) it follows that

$$\begin{aligned} \overline{h}_i &= \max_x h_i = \max_x \left(H_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j \right) \\ &= H_{ii} + 2 \max_x \sum_{\substack{j=1 \\ j \neq i}}^{n_b} H_{ij} x_j = H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^+ \end{aligned} \tag{16}$$

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

$$\underline{h}_i = \min_x h_i = H_{ii} + 2 \sum_{j=1}^{n_b} H_{ij}^- \tag{17}$$

Equation (15) can then finally be written in the desired form

$$x_i^* = \begin{cases} 0, \; H_{ii} + 2\sum_{j=1}^{n_b} H_{ij}^- \geq 0 \Leftrightarrow H_{ii} \geq -2\sum_{j=1}^{n_b} H_{ij}^- \\ 1, \; H_{ii} + 2\sum_{j=1}^{n_b} H_{ij}^+ < 0 \Leftrightarrow H_{ii} < -2\sum_{j=1}^{n_b} H_{ij}^+ \end{cases} \tag{18}$$

From (18) it is clear that the computational complexity is polynomial in the number of variables, i.e. in n_b. □

Now the result is extended to problems of type $\mathcal{P}_2$.

Corollary 1
For a BQP problem of type $\mathcal{P}_2$ the optimal value of one or more components x_i can be found in polynomial time if for some $i \in \{1, .., n_b\}$ any of the following conditions is satisfied

$$\begin{cases} H_{ii} \geq -2f_i - 2\sum_{j=1}^{n_b} H_{ij}^- & (i) \\ H_{ii} < -2f_i - 2\sum_{j=1}^{n_b} H_{ij}^+ & (ii) \end{cases}$$

If any of the conditions (i) or (ii) is satisfied for a certain value of i, the 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 1 by recognizing the fact that

$$Q(x) = \frac{1}{2} x^T H x + f^T x = x^T \left(\frac{H + 2\text{diag}(f)}{2} \right) x \tag{19}$$

for $x_i \in \{0, 1\}$. □

4 Implementation

The preprocessing algorithm can be implemented in several different ways. If the programming language supports matrix syntax, e.g. Matlab, the implementation can be seen as a straightforward implementation of Theorem 1 or Corollary 1. Using the pseudo code command `RowSum` and the relational operators $<$ and $\geq$, a quick sketch of the implementation is given by

$$\begin{aligned} &\texttt{RowSum}(H_d + 2H^+) < 0 \\ &\texttt{RowSum}(H_d + 2H^-) \geq 0 \end{aligned} \tag{20}$$

where `RowSum` returns the sum of each row of a matrix as a column vector and the relational operators are assumed to work componentwise, producing binary vectors as results. In this way, Corollary 1 is applied to all indices i simultaneously. If the optimal value is found for at least one component in x, it is possible to repeatedly apply the above mentioned test again to a smaller problem, excluding the indices for which optimal x_i already have been found.

It is also possible to perform the tests in (20) one row at a time. Then, if an optimal value of a variable has been found for some index i, this value can be used to remove a column from H in the remaining tests, assuming that the other columns of H are modified appropriately. This results in a sharper test, since the already found optimal value is then no longer estimated by the worst case value.

After the algorithm has computed the optimal value of all components that it is capable of, an ordinary MIQP solver, or BQP solver, may be applied to compute the remaining ones.

5 Preprocessing for the MIQP Problem

If the x-vector is allowed to contain both real valued and binary valued variables, the BQP problem becomes an MIQP problem. In this section, the problem is assumed to be of the type $\mathcal{P}_3$ and the x-vector is assumed to be in the form

$$\begin{gathered} x = \begin{bmatrix} x_c \\ x_b \end{bmatrix} \\ x_c \in \mathbb{R}^{n_c},\, x_b \in \{0, 1\}^{n_b} \end{gathered} \tag{21}$$

The objective function can be expressed as

$$Q(x) = \frac{1}{2}x^T H x + f^T x = \frac{1}{2}\begin{bmatrix} x_c^T & x_b^T \end{bmatrix} \begin{bmatrix} H_{cc} & H_{cb} \\ H_{cb}^T & H_{bb} \end{bmatrix} \begin{bmatrix} x_c \\ x_b \end{bmatrix} + \begin{bmatrix} f_c^T & f_b^T \end{bmatrix} \begin{bmatrix} x_c \\ x_b \end{bmatrix} \tag{22}$$

We now assume H_{cc} to be positive definite. If all components in x had been real valued it would have been straightforward to use the first order necessary and sufficient conditions for optimality to calculate an algebraic 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 anyway be used to express the optimal values of the real valued variables as a function of the binary valued variables. The expression for the optimal real valued variables is then given by

$$x_c = -H_{cc}^{-1}\left(H_{cb}x_b + f_c\right) \tag{23}$$

This expression is substituted into (22). The resulting optimization problem is a pure BQP problem. The objective function, disregarding constant terms, can be written as

$$\frac{1}{2}x_b^T\tilde{H}x_b + \tilde{f}^T x_b \triangleq \frac{1}{2}x_b^T\left(H_{bb} - H_{cb}^T H_{cc}^{-T} H_{cb}\right)x_b + \left(f_b - H_{cb}^T H_{cc}^{-1} f_c\right)^T x_b \tag{24}$$

In the calculations the symmetry of H and H^{-1} has been used. When the objective function is written in the form in (24), Corollary 1 can be applied.

6 Application of the Results to MPC

The results from Section 5 is in this section applied to the MPC problem. It is desirable to find the control signal sequence which minimizes a certain criterion for a system in the form

$$\begin{aligned} x(k+1) &= Ax(k) + B_c u_c(k) + B_b u_b(k) + B_d d(k) \\ z(k) &= Mx(k) \end{aligned} \tag{25}$$

where $u_c \in \mathbb{R}^{n_{uc}}$ denotes real valued control signals, $u_b \in \{0,1\}^{n_{ub}}$ denotes binary valued control signals and $d \in \mathbb{R}^{n_d}$ denotes disturbances. The signal $z \in \mathbb{R}^{n_z}$ denotes the control objective.

In MPC, an optimization problem is solved at each sampling time k, based on the state $x(k)$. At time k the optimization routine returns the optimal control signal sequence for all samples from time k to time $k + H_p - 1$, where H_p denotes the length of the prediction horizon. Although the optimal control signal sequence is known for H_p samples, only $u(k)$ is actually applied to the system. In the next sample a new optimization is performed and the first component in the new optimal control signal sequence is applied, and so on. The criterion which is often minimized in MPC problems is for time k

$$\begin{aligned} &\sum_{j=0}^{H_p-1} ||z(k+j+1) - r(k+j+1)||_{Q_r}^2 + ||u_c(k+j)||_{Q_c}^2 + ||u_b(k+j)||_{Q_b}^2 \\ &= \left(\tilde{M}\left(Hx(k) + S_cU_c + S_bU_b + S_dD\right) - R\right)^T \times \tilde{Q}_r \\ &\times \left(\tilde{M}\left(Hx(k) + S_cU_c + S_bU_b + S_dD\right) - R\right) + U_c^T\tilde{Q}_cU_c + U_b^T\tilde{Q}_bU_b \\ &= \left(\alpha + \tilde{M}S_bU_b\right)^T \tilde{Q}_r \left(\alpha + \tilde{M}S_bU_b\right) + 2U_c^TS_c^T\tilde{M}^T\tilde{Q}_r\left(\alpha + \tilde{M}S_bU_b\right) \\ &+ U_c^TS_c^T\tilde{M}^T\tilde{Q}_r\tilde{M}S_cU_c + U_c^T\tilde{Q}_cU_c + U_b^T\tilde{Q}_bU_b \end{aligned} \tag{26}$$

In the first equality, the definitions made in the Appendix have been used, and in the second equality the definition $\alpha = \tilde{M}\left(Hx(k) + S_dD\right) - R$ has been used.

Compared to MPC for real valued control signals, the problem to minimize (26) is no longer a QP problem, since the control signals in U_b are binary variables. Instead, the problem is an MIQP problem of the type $\mathcal{P}_3$. By ignoring constants and dividing the objective function (26) by two, it can be written in the form (22) with

$$\begin{aligned}
x_c &= U_c \\
x_b &= U_b \\
H_{cc} &= S_c^T \tilde{M}^T \tilde{Q}_r \tilde{M} S_c + \tilde{Q}_c \\
H_{cb} &= S_c^T \tilde{M}^T \tilde{Q}_r \tilde{M} S_b \\
H_{bb} &= S_b^T \tilde{M}^T \tilde{Q}_r \tilde{M} S_b + \tilde{Q}_b \\
f_c &= S_c^T \tilde{M}^T \tilde{Q}_r \alpha \\
f_b &= S_b^T \tilde{M}^T \tilde{Q}_r \alpha
\end{aligned} \tag{27}$$

The optimization problem can then of course also be expressed as a BQP problem in the form (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 envelope of the matrix elements descents horizontally and vertically away from the diagonal. How fast the modulus 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 do also give elements that fade out. This behavior has to be further investigated. This structure of the H-matrix in the MPC problem makes it easier to satisfy condition (i) or (ii) in Corollary 1.

7 Examples

In the following subsections, two examples are presented to which the preprocessing algorithm is applied. 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 valued variables have been eliminated, as described in Section 5, 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 computed. 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 valued variables first have been eliminated according to Section 5 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 Sections 7.1 and 7.2, 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 [2] has been used. To be able to make a fair comparison, all available combinations of the settings “method” and “branchrule” in the solver has been tested. In the table, only the shortest solution time achieved is presented. For further information about available settings, see [2].

As mentioned in Section 4, the preprocessing algorithm can be implemented in several different ways. In Matlab, an implementation with vectorized expressions has much 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 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.

7.1 Mass Position Control

In this example, a mass is controlled in one dimension by two separate forces. One force is possible to control continuously and the other is applied in a binary way with a certain magnitude and direction. The position of the mass is supposed to follow a 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. The continuous state space description is given by

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & -5 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_c \\ u_b \end{bmatrix} \\ z &= \begin{bmatrix} 0 & 1 \end{bmatrix} x \end{aligned} \tag{28}$$

To obtain a system in the form in (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 samples. 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 in (26) with

$$\begin{aligned} Q_r &= 100 \\ Q_c &= 1 \\ Q_b &= 1 \end{aligned} \tag{29}$$

The initial state is

$$x_1(0) = 5 \text{ and } x_2(0) = 0 \tag{30}$$

The result is shown in Figure 1. The computational time for optimizing 50 real valued and 50 binary valued variables is presented in Table 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 [2]. 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 the preprocessing algorithm based on Corollary 1. It can also be noticed that the computational time is reduced with a factor of about 180.

7.2 Satellite Attitude Control

In this example the optimal control signal sequence for the attitude control of a satellite is calculated. The satellite controls its attitude by accelerating a reaction wheel inside the

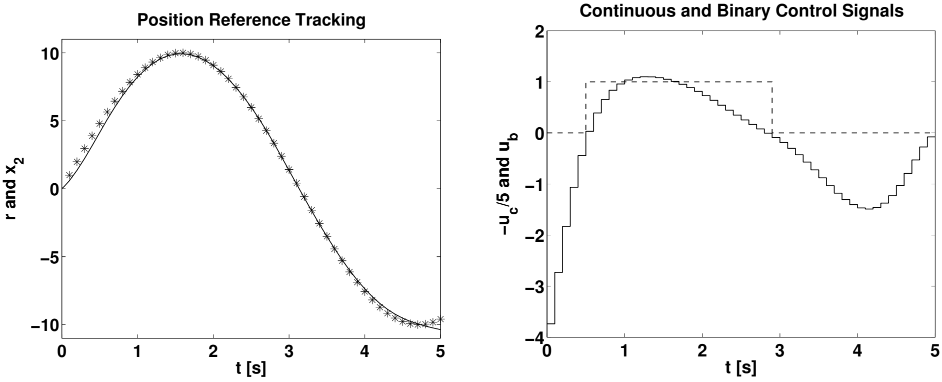


Figure 1: *The left plot shows how the position of the mass, x_2 (solid), follows the sampled reference signal, r (starred), when the calculated optimal control signal sequence is applied to the continuous model of the system. The right plot shows the control signals, u_c (solid) and u_b (dashed). Note that u_c is scaled in the plot.*

Table 1: *Performance tests*

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

satellite and by using external thrusters. When 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, or at least high enough to be power consuming in steady state. 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 motor. 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, *i.e.*, 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

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 2.5 & 1 & -1 \\ -10 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_c \\ u_{b,1} \\ u_{b,2} \end{bmatrix} \\ z &= I_3 x \end{aligned} \tag{31}$$

To obtain a system in the form in (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 in (26) with

$$\begin{aligned} Q_r &= \begin{bmatrix} 0.5 \cdot 10^4 & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 10^{-1} \end{bmatrix} \\ Q_c &= 10 \\ Q_b &= 10 \cdot I_2 \end{aligned} \tag{32}$$

The initial state is

$$x_1(0) = 0,\, x_2(0) = 0 \text{ and } x_3(0) = 0 \tag{33}$$

In this example, the reference signal for the attitude of the satellite is a step function with the amplitude 0.5. The optimal control signal sequence and the attitude of the satellite is shown in Figure 2. The computational time for optimizing 20 real valued and 40 binary variables is found in Table 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 [2]. 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.

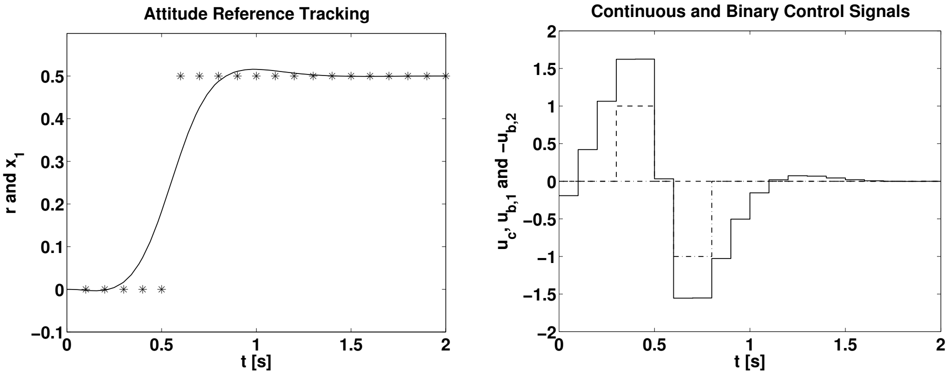


Figure 2: *The left plot shows how the attitude of the satellite, x_1 (solid), follows the sampled reference signal, r (starred), when the calculated optimal control signal sequence is applied to the continuous model of the system. The right plot shows the control signals, u_c (solid), $u_{b,1}$ (dashed)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

Table 2: *Performance tests*

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

8 Conclusions

In this paper a preprocessing algorithm for MIQP and BQP problems has been derived. These problems are generally known to have exponential complexity. The preprocessing algorithm can compute the optimal values of some, or all, of the optimization variables in polynomial time. For certain examples, it has been shown that the preprocessing algorithm can reduce the computational time significantly. In one case the computational time was reduced with a factor of 275. A suggestion to future work is to find more applications where the preprocessing algorithm can be applied. Also, extensions to the case of inequality constraints should be investigated.

A Appendix

In this appendix the definitions of the vectors and matrices used when the objective function for the MPC problem is cast in the form in (26) are presented. The system to be controlled is assumed to be in the form in (25).

$$
U_c = \begin{bmatrix} u_c(k) \\ u_c(k+1) \\ \vdots \\ u_c(k+H_p-1) \end{bmatrix}, \; U_b = \begin{bmatrix} u_b(k) \\ u_b(k+1) \\ \vdots \\ u_b(k+H_p-1) \end{bmatrix},
$$

$$
D = \begin{bmatrix} d(k) \\ d(k+1) \\ \vdots \\ d(k+H_p-1) \end{bmatrix}, R = \begin{bmatrix} r(k+1) \\ r(k+2) \\ \vdots \\ r(k+H_p) \end{bmatrix},
$$

$$
H = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{H_p} \end{bmatrix}, \; S_c = S(B_c), \; S_b = S(B_b), \; S_d = S(B_d),
$$

$$
S(B_*) = \begin{bmatrix} B_* & 0 & \dots & 0 \\ AB_* & B_* & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H_p-1}B_* & A^{H_p-2}B_* & \dots & B_* \end{bmatrix},
$$

$$
\tilde{M} = \begin{bmatrix} M & & 0 & 0 \\ & M & & 0 \\ 0 & & \ddots & \\ 0 & 0 & & M \end{bmatrix}, \tilde{Q}_r = \begin{bmatrix} Q_r & & 0 & 0 \\ & Q_r & & 0 \\ 0 & & \ddots & \\ 0 & 0 & & Q_r \end{bmatrix},
$$

$$
\tilde{Q}_c = \begin{bmatrix} Q_c & & 0 & 0 \\ & Q_c & & 0 \\ 0 & & \ddots & \\ 0 & 0 & & Q_c \end{bmatrix}, \tilde{Q}_b = \begin{bmatrix} Q_b & & 0 & 0 \\ & Q_b & & 0 \\ 0 & & \ddots & \\ 0 & 0 & & Q_b \end{bmatrix}
$$

References

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

[2] A. Bemporad and D. Mignone. *A Matlab function for solving Mixed Integer Quadratic Programs Version 1.02 User Guide*. Institut für Automatik, ETH, 2000.

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

[4] 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.

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

[6] 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.

[7] 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.

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

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

Paper B

A Low-Complexity High-Performance Preprocessing Algorithm for Multiuser Detection using Gold Sequences

Authors: Daniel Axehill, Fredrik Gunnarsson and Anders Hansson

Edited version of the paper: D. Axehill, F. Gunnarsson, and A. Hansson. A low-complexity high-performance preprocessing algorithm for multiuser detection using Gold sequences. Provisionally accepted for publication in *IEEE Transactions on Signal Processing*, Jan. 2008
which extends: 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

Preliminary version: Published as Technical Report LiTH-ISY-R-2840, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

A Low-Complexity High-Performance Preprocessing Algorithm for Multiuser Detection using Gold Sequences[†]

Daniel Axehill, Fredrik Gunnarsson and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,fred,hansson}@isy.liu.se

Abstract

The optimum multiuser detection problem can be formulated as a maximum likelihood problem, which yields a binary quadratic programming problem to be solved. Generally this problem is $\mathcal{NP}$-hard and is therefore hard to solve in real-time. In this article, a preprocessing algorithm is presented which makes it possible to detect some or all users optimally for a low computational cost if signature sequences with low cross-correlation, *e.g.*, Gold sequences, are used. The algorithm can be interpreted as, *e.g.*, an adaptive trade-off between parallel interference cancellation and successive interference cancellation. Simulations show that the preprocessing algorithm is able to optimally compute more than $94\,\%$ of the bits in the problem when the users are time-synchronous, even though the system is heavily loaded and affected by noise. Any remaining bits, not computed by the preprocessing algorithm, can either be computed by a suboptimal detector or an optimal detector. Simulations of the time-synchronous case show that if a suboptimal detector is chosen, the BER rate is significantly reduced compared to using the suboptimal detector alone.

1 Introduction

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

[†] The research has been supported by the Swedish Research Council for Engineering Sciences under contract Nr. 621-2002-3822.

the signal from other users as additive Gaussian noise, [23]. An improvement to this strategy is to use the known correlation between users in the demodulation process. The performance can be improved even more if the detector makes the most likely decision, which formally is achieved by solving a so-called Maximum Likelihood (ML) problem. When the optimum multiuser detection problem is cast in the form of an ML problem it requires the solution of a so-called Binary Quadratic Programming (BQP) problem. Unfortunately, these problems are generally known to be $\mathcal{NP}$-hard, [14]. If the signature waveform produces a cross-correlation matrix with some special structures, the problem can sometimes turn out to have lower complexity, [21, 22, 24].

Many contributions to the area of multiuser detection have already been published. The objective has been to find an algorithm which solves the multiuser detection problem in reasonable time in order to make a real-time implementation possible. Previously, this has been done either by restricting the class of possible cross-correlation matrices or by employing some suboptimal procedure. In [24], 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 [21], where the synchronous multiuser detection problem is solved with a polynomial complexity algorithm if the cross-correlations between the users are non-positive. It is also shown that Gold sequences satisfy this condition in the synchronous case. Another paper also dealing with a special class of cross-correlations is the one in [22], 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 [23]. Several different algorithms, optimal as well as suboptimal, are presented and evaluated in [12]. The suboptimal algorithm local search is evaluated in [13]. Branch and bound methods are investigated in [17]. Another near optimal approach is presented in [15]. Also the well-known Kalman filter has been applied to the problem. This approach is presented in [16]. Another interesting approach is to use so-called semidefinite relaxations to produce suboptimal solutions to the problem. This idea has been considered in, *e.g.*, [18] and [8].

A different detector approach is based on adaptive thresholding to provide a low-complexity detector for the ML problem, [20, 26]. The resulting algorithm is evaluated at low traffic loads for pseudo-random sequences with promising results. It is also concluded that the algorithm is not expected to perform well at high loads.

In this article, a preprocessing algorithm with polynomial complexity for the BQP problem is derived. A preprocessing algorithm is an algorithm which processes the optimization problem in the step previous to the one when the actual solver is applied. Because the preprocessing algorithm executes in polynomial time and the BQP solver, generally, executes in exponential time, the required CPU time can be reduced if bits can be computed optimally already in the preprocessing step. It became clear that when applied to the MUD problem, the resulting BQP preprocessing algorithm presented in [3] was found to be equivalent to the one presented in [20, 26]. The two major differences and enhancements made in this article compared to [20, 26] is that, first, the algorithm is derived using an optimization point of view of the problem, which results in a general algorithm for BQP problems and a framework which makes it easy to prove optimality for the formulation of the problem solved in the step after preprocessing, and second, that the evaluation is very thoroughly performed in the important case where chip sequences of

Gold type are used. Both the BER performance as well as the computational performance is evaluated. Note that the ideas presented in this article also work with other types of sequences with low cross-correlation, and the Gold sequences were chosen as one representative of such sequences. The algorithm is in principle also useful if the favorable cross-correlations from the time-synchronous case have been slightly distorted by, *e.g.*, time-asynchronous users, as long as the resulting cross-correlations between the users are low. Furthermore, it is shown in simulations that when signature sequences with low cross-correlations, like Gold sequences, are used in a synchronous system, the algorithm is useful also at high loads.

Section 2 provides the CDMA channel models upon which this article is based, and Section 3 describes preprocessing for the general BQP problem. The resulting preprocessing is applied to CDMA in Section 4, followed by extensive numerical evaluations in Section 5. Finally, Section 6 provides some conclusive remarks.

2 CDMA Channel Models

In this section, the synchronous and asynchronous CDMA channel models used in the article are presented. The notation has been taken from [25], where also a more thorough description of the models can be found. In the case when the users are time-synchronous, the received waveform $y(t)$ is modeled as a K-user channel consisting of the sum of K antipodally modulated synchronous signature waveforms embedded in additive white Gaussian noise, *i.e.*, as

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

where A_k is the received amplitude of the signal from user k, $b_k \in \{-1, +1\}$ is the data bit transmitted by user k, $s_k(t)$ is the deterministic signature waveform assigned to user k, normalized to have unit energy, and $n(t)$ is white Gaussian noise with unit power spectral density. Furthermore, σ is the channel noise variance and T is the inverse of the data rate.

The similarity of different signature waveforms $s_k(t)$ is expressed in terms of the cross-correlation defined by

$$\rho_{ij} = \int_{0}^{T} s_i(t) s_j(t) dt \tag{2}$$

The normalized cross-correlation matrix $R = \{\rho_{ij}\}$ has ones in the diagonal and is symmetric non-negative definite, [25]. If $\rho_{ij} = 0$ whenever $i \neq j$, the signature sequences are orthogonal. In this work, non-orthogonal sequences of Gold type have been used and these usually give low cross-correlation for all possible non-zero offsets. Another common choice of sequences with these properties is Kasami sequences.

In the case when the users are time-asynchronous, a frame of length $2M + 1$ symbols is considered. The signature sequences used in this work are user-specific, and reused repeatedly for each new bit sent by a specific user. The CDMA channel model is also in this case adopted from [25] and can be written as

$$y(t) = \sum_{k=1}^{K} \sum_{i=-M}^{M} A_k b_k[i] s_k(t - iT - \tau_k) + \sigma n(t) \tag{3}$$

where $b_k[i],\ i = -M, \dots, M$ represent the bits sent by user k during the frame under consideration, and τ_k is the user-specific time offset. Assume, without any loss of generality, that the users are labeled by their time if arrival, *i.e.*, that $\tau_1 \leq \tau_2 \leq \tau_3 \leq \ldots \leq \tau_k$. If $k < l$, the elements in the cross-correlation matrix can be written as

$$\begin{aligned} \rho_{kl} &= \int_{\tau_l - \tau_k}^{T} s_k(t) s_l(t - \tau_l + \tau_k) dt \\ \rho_{lk} &= \int_{0}^{\tau_l - \tau_k} s_k(t) s_l(t + T - \tau_l + \tau_k) dt \end{aligned} \tag{4}$$

where $0 \leq \tau_l - \tau_k \leq T$. If it is assumed that the intersymbol interference is limited to occur for symbols immediately adjacent in time, the complete cross-correlation matrix R is in this case built-up as a block-Toeplitz matrix with blocks $R[0]$ along the main diagonal and cross-correlations $R[1]^T$ and $R[1]$, repeated above and below the main block diagonal, respectively, where,

$$R_{jk}[0] = \begin{cases} 1, \text{ if } j = k \\ \rho_{jk}, \text{ if } j < k \\ \rho_{kj}, \text{ if } j > k \end{cases}, \quad R_{jk}[1] = \begin{cases} 0, \text{ if } j \geq k \\ \rho_{kj}, \text{ if } j < k \end{cases} \tag{5}$$

According to [25], the bit-sequence most likely sent by the users in the synchronous case are given by the solution $\hat{b} = \left[\hat{b}_1, \hat{b}_2, \dots, \hat{b}_n\right]^T$ to the optimization problem

$$\max_{\hat{b}} \exp\left(-\frac{1}{2\sigma^2} \int_0^T \left(y(t) - \sum_{k=1}^{K} \hat{b}_k A_k s_k(t)\right)^2 dt\right) \tag{6}$$

The asynchronous case follows analogously, but the notation becomes slightly more complex. Note that, the variables in the optimization problem in (6) are the bits to be estimated. This optimization problem can be rewritten as an equivalent minimization problem

$$\min_{\hat{b}} \frac{1}{2} \hat{b}^T H \hat{b} - y^T A^T \hat{b} \tag{7}$$

where $\hat{b} \in \{-1, +1\}^K$, $A = \text{diag}(A_1, A_2, \dots, A_k)$, $H = ARA$ and $y = [y_1, y_2, \dots, y_K]^T$, where y_i is the output from matched filter i. After a variable substitution, this problem can be identified as a $0/1$-BQP problem.

There exist a number of suboptimal detectors applicable to the multiuser detection problem. The simplest one is the conventional detector, [12],

$$\hat{b} = \text{sign}(y) \tag{8}$$

where disturbances from other users are treated as noise. This detector does not use any floating-point operations (flops). A simple detector using knowledge of the cross-correlation between different users is the decorrelating detector, [12],

$$\hat{b} = \text{sign}(H^{-1}Ay) \tag{9}$$

The number of flops for computing computing $H^{-1}Ay$ grows as $\mathcal{O}(K^3)$, [6].

For what follows, the channel models in (1) and in (3) are used for all computational results to be presented in this article. Furthermore, the received amplitudes from different users A_k are assumed equal to one for all users, *i.e.*, the uplink is subject to power control. The signature waveforms used are all of Gold type of length 127.

3 Preprocessing for the BQP Problem

Most algorithms used when solving BQP problems either focus on producing approximative solutions or only on handling various special cases of the general problem, [9]. The algorithm to be presented in this article belongs to the latter type of algorithms and it is applicable to BQP problems having dominating diagonal elements compared to the off-diagonal elements. In [3], the algorithm has previously been successfully applied to Model Predictive Control (MPC) for systems including binary variables. Some approximative heuristic algorithms can be found in, *e.g.*, [14], [4], [19], and [10].

In this article, a BQP problem in the form

$$\mathcal{P}: \begin{cases} \min\limits_{x} & \frac{1}{2}x^T H x + f^T x \\ \text{subject to} & x \in \{0,1\}^{n_b} \end{cases} \tag{10}$$

is studied, where $H \in \mathbb{R}^{n_b \times n_b}$ is symmetric. The algorithm presented in this section makes it possible to speed up the solution of BQP problems having large diagonal elements compared to the non-diagonal elements. 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. Some of the basic BQP optimization ideas used in this work can be found in, *e.g.*, [11]. In this reference, the ideas are not used to build a preprocessing algorithm, but to find all optimal solutions to the BQP problem. Before the main theorem is given, the following definition is introduced

$$H = H_d + H^+ + H^- \tag{11}$$

where

$$\begin{aligned} H_{d,ij} &= \begin{cases} H_{ij}, & i = j \\ 0, & i \neq j \end{cases} \\ H_{ij}^+ &= \max\left(0, H_{ij} - H_{d,ij}\right) \\ H_{ij}^- &= \min\left(0, H_{ij} - H_{d,ij}\right) \end{aligned} \tag{12}$$

The preprocessing algorithm is built upon the following theorem:

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

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

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: See [3]. A simple flop count shows that the computational complexity for each test (i) or (ii) grows as $\mathcal{O}(n_b)$. Hence, performing the tests for all variables gives a computational complexity of $\mathcal{O}({n_b}^2)$, *i.e.*, polynomial complexity in n_b. □

4 Application of the Results to CDMA

In this section, it is shown how to apply the preprocessing algorithm derived in Section 3 to the optimization problem in (7) and how the result can be interpreted.

4.1 Using Preprocessing

In order to be able to apply the preprocessing algorithm, the optimization problem in (7) has to be rewritten in the BQP form $\mathcal{P}$. Note especially the domain of the optimization variable x. In order to get an optimization problem with binary variables the following variable substitution is performed

$$\hat{b} = 2\bar{b} - \mathbf{1} \tag{13}$$

where $\bar{b} \in \{0, 1\}^K$, $\hat{b} \in \{-1, +1\}^K$ and $\mathbf{1}$ denotes a column vector with all elements equal to one. Using (13), neglecting constant terms and dividing by 4, the objective function in (7) can be rewritten as

$$\frac{1}{2}\bar{b}^T H \bar{b} + \tilde{f}^T \bar{b} \tag{14}$$

where

$$\tilde{f} = -\frac{1}{2}H\mathbf{1} - \frac{1}{2}Ay \tag{15}$$

The problem is now in the form $\mathcal{P}$, on which preprocessing can be performed.

4.2 Using the Algorithm in Multiuser Detection

In this section it is described how the algorithm can be used in multiuser detection. It is first shown how it works in its simplest form where the conditions in Theorem 1 are applied once for each user. Later it is shown how the performance of the algorithm can be significantly improved by applying the algorithm iteratively. Finally, it is described how any remaining bits not computed by the preprocessing algorithm can be computed by either an optimal or a suboptimal algorithm. Detailed algorithm descriptions can be found in [2], and in [26].

Non-Iterated Implementation

In its simplest form the algorithm consists of applying the conditions in Theorem 1 once for each user. As previously discussed in the proof of Theorem 1, the computational complexity of one such preprocessing iteration (test the conditions once for each user) grows as $\mathcal{O}(K^2)$. Using the notation in the multiuser detection problem, the two conditions (i) and (ii) in the corollary can after simplification be written as

$$\begin{cases} A_i y_i \leq -\sum_{j \neq i} |H_{ij}| & (i) \\ A_i y_i \geq \sum_{j \neq i} |H_{ij}| & (ii) \end{cases} \tag{16}$$

From (13) and (16), it follows that the optimal choice of $\hat{b}_i$ is given by

$$\hat{b}_i^* = \begin{cases} -1, & \text{if } (i) \text{ holds} \\ 1, & \text{if } (ii) \text{ holds} \end{cases} \tag{17}$$

Note that these conditions are equivalent to the ones given in [26]. Furthermore, note that the conditions in (16) for a certain bit i are independent of the value of any other undetected bits. Hence, if a decision can be made, it is made independently of the value of any so far undetected bits. If (16) is investigated, it can be realized that a necessary property of the cross-correlation matrix to be able to successfully use the algorithm, is that the signature sequences give low cross-correlations between different users. This is typically the case for, *e.g.*, Gold sequences.

In the non-iterated implementation, the detector can be interpreted as a threshold detector. Considering, *e.g.*, user 1, the distribution of the output from the matched filter for this user is illustrated in Figure 1. In the example, 60 users are sharing the channel and the SNR for user 1 is 6 dB. If the matched filter output gets outside the gray region **B**, the bit sent by the user considered can be detected *optimally* by the preprocessing algorithm in a *single* iteration. Note that the size of the threshold has been chosen by the algorithm in order to be able to *guarantee* optimality. The algorithm can also be related to the conventional detector in (8). In region **A** and **C**, the output from the preprocessing algorithm and the output from the conventional detector coincide. There is however an extremely important difference; the output from the preprocessing algorithm is guaranteed to be optimal. Note that the fact that the output from the two detectors often coincide is not a weakness of the preprocessing algorithm but rather a proof that the conventional detector often, but not always, make optimal decisions. Compared to previous optimal low complexity methods presented in [21, 24] and [22], the algorithm presented in this article does not introduce any requirements on the sign of the cross-correlations or that the cross-correlations between users are equal. On the other hand, the preprocessing algorithm cannot, in general, detect all users. In most cases the algorithm can detect some of the users in the first iteration, but there is also often some users which cannot be detected in the first iteration. In the iterated implementation of the algorithm, the algorithm

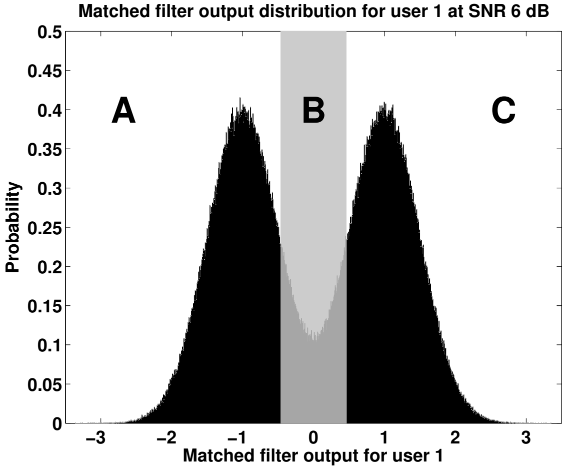


Figure 1: This plot shows the distribution for the matched filter output for user 1 *in an example with* 60*-users and SNR* 6 *dB for this user. The gray region **B** illustrates the region where user* 1 *cannot be detected in a single preprocessing iteration. The computational result shows that in this example, the probability of detection of user* 1 *in the first preprocessing iteration is* 0.856.

is applied repeatedly and bits detected optimally in previous iterations are used to relax the conditions for previously unsolved bits.

Iterated Implementation

When the algorithm is applied iteratively, the number of users possible to detect by the preprocessing algorithm is significantly increased. As long as at least one user could be detected in the previous run, it is possible to start over again and try to compute the remaining ones. In principle, this is done by inserting the already computed variables in the objective function of the BQP problem and formulating a new BQP problem in the remaining variables on which it is possible to run preprocessing. This is described in detail in the next section. This procedure is in this text referred to as the iterated implementation and is further described in [2]. Note that even in the worst case the number of iterations is limited to K since the algorithm is aborted if less than one user was detected in the last iteration. Hence, the worst case computational complexity for the iterated implementation of the preprocessing algorithm grows as $\mathcal{O}(K^3)$. Furthermore, note that it is possible, *e.g.*, to be able to satisfy time demands in a real-time system, to abort the algorithm before the K steps have completed and to solve any remaining variables suboptimally. Hence, it is easy to make a dynamic trade-off between BER performance and computational time. This has been further investigated in [20].

To gain performance compared to the conventional detector it is necessary that information from previous iterations contracts region **B** and either extends region **A** to the right of the origin or extends region **C** to the left of the origin. That this actually occurs in practice will be shown in Section 5 where the proposed algorithm outperforms the conventional detector.

When the iterated implementation is used, the algorithm can be interpreted as an adaptive trade-off between Parallel Interference Cancellation (PIC) and Successive Interference Cancellation (SIC). In SIC, only one user is detected and cancelled in each iteration, which means that the number of iterations equals the number of users to be detected. On the contrary, in PIC all users are detected in every iteration and subsequently cancelled. Uncertain detection reduces cancellation performance, and therefore, only a fraction of the regenerated signals are cancelled, known as partial PIC. Hence, in SIC only one user is detected per iteration even though more users can be detected with confidence, while in partial PIC all users are detected, also the ones with less confidence, in each iteration. In the proposed scheme, all previously undetected users that can be detected with confidence, but no more, are detected in each iteration. More information on PIC and SIC can be found in, *e.g.*, [1].

The Step After Preprocessing

When the preprocessing algorithm has terminated, another detector has to be applied to detect any remaining bits. Consider an optimization problem in the form in (7). Without any loss of generality, order the elements in $\hat{b}$ in two parts, and denote the parts $\hat{b}_c$ and $\hat{b}_r$ where the first part is possible to compute by preprocessing and the latter part is not. The problem can be rewritten in the new variables as

$$\min_{\hat{b}_c,\hat{b}_r} \frac{1}{2}\hat{b}_c^T H_{cc}\hat{b}_c - y_c^T A_{cc}^T\hat{b}_c + \hat{b}_c^T H_{cr}\hat{b}_r + \frac{1}{2}\hat{b}_r^T H_{rr}\hat{b}_r - y_c^T A_{rr}^T\hat{b}_r \qquad (18)$$

where subindices c and r have been used to denote the components in $\hat{b}$ that can and cannot be computed by preprocessing, respectively. These subindices are also used, analogously, to index out the corresponding parts in A, R, H and y.

Since it in an optimization problem always is possible to first minimize over some of the variables and then over the remaining ones, [6], it is here possible to first minimize over $\hat{b}_c$ and then over $\hat{b}_r$. The optimal solution $\hat{b}_c^*$ is then parameterized in $\hat{b}_r$, *i.e.*, the solution to the first optimization problem is in general a function $\hat{b}_c^*(\hat{b}_r)$, which can be hard to explicitly formulate in an integer optimization problem like this one. However, from the discussion below the equation in (17) it is clear that the preprocessing algorithm finds the optimal value of those variables that have an optimal value which is *independent* of the value of the remaining variables. Hence, $\hat{b}_c^*$ is not dependent on $\hat{b}_r$, *i.e.*, $\hat{b}_c^*(\hat{b}_r) = \hat{b}_c^*$. After the solution $\hat{b}_c^*$ from the preprocessing algorithm has been inserted into the problem in (18), the remaining optimization problem in $\hat{b}_r$ can be reformulated as

$$\min_{\hat{b}_r} \frac{1}{2}\hat{b}_r^T H_{rr}\hat{b}_r + \left(H_{cr}^T\hat{b}_c^* - A_{rr}y_r\right)^T \hat{b}_r \qquad (19)$$

This is a new optimization problem in BQP form. Depending on the time available, the algorithm used after preprocessing may be chosen to produce either optimal solutions or suboptimal solutions. If an optimal solution is desired, existing optimization software like the commercial state-of-the-art branch and cut solver CPLEX, [7], or the freely available branch and bound solver `miqp.m`, [5], can be used. If the dimension of $\hat{b}_r$ is low, which it often is after preprocessing, it is actually tractable to solve the problem in (19) by explicit enumeration of all solutions. If suboptimal solutions are considered sufficient, possible choices are to apply one of the detectors in (8) or (9). An important question to answer is to which MUD problem, *i.e.*, which $\tilde{y}$, $\tilde{A}$ and $\tilde{R}$ that defines the new problem, a detector should be applied in order to guarantee a jointly optimal decision in $\begin{bmatrix} \hat{b}_c^T & \hat{b}_r^T \end{bmatrix}^T$. This can be done by identifying the problem in (19) to be in the form in (7). The result after identification is

$$\tilde{y} = y_r - R_{cr}^T A_{cc} \hat{b}_c^*, \ \tilde{A} = A_{rr}, \ \tilde{R} = R_{rr} \tag{20}$$

Note that, this result also serves as a theoretical motivation for equation (15) in [26]. Consequently, an optimal detector working on the problem defined by the parameters in (20) will provide an optimal solution $\hat{b}_r^*$ to the optimization problem in (19), and hence the total solution $\begin{bmatrix} \hat{b}_c^T & \hat{b}_r^T \end{bmatrix}^T$ will be jointly optimal. This holds independently of if $\hat{b}_r^*$ is found as the solution to the optimization problem in (19) or if it is found as the optimally detected bit-sequence to an MUD problem with the parameters in (20).

Now, the choice of $\tilde{y}$ in (20) will be slightly more investigated. Using the CDMA models presented in Section 2, the matched filter output can be expressed as

$$y = \begin{bmatrix} y_c \\ y_r \end{bmatrix} = RAb + n = \begin{bmatrix} R_{cc}A_{cc}b_c + R_{cr}A_{rr}b_r + n_c \\ R_{cr}^T A_{cc}b_c + R_{rr}A_{rr}b_r + n_r \end{bmatrix} \tag{21}$$

which means that

$$\tilde{y} = y_r - R_{cr}^T A_{cc} \hat{b}_c^* = R_{rr}A_{rr}\hat{b}_r + R_{cr}^T A_{cc} \left(b_c - \hat{b}_c^*\right) + n_r \triangleq R_{rr}A_{rr}\hat{b}_r + \tilde{n}_r \tag{22}$$

where b_c and n_r denote the true values of the remaining bit-sequence and the noise in the output from the matched filters, respectively, and $\tilde{n}_r \in \mathcal{N}\left(R_{cr}^T A_{cc}\left(b_c - \hat{b}_c^*\right), \sigma^2 R_{rr}\right)$. The term $-R_{cr}^T A_{cc}\hat{b}_c^*$ has intuitively the function of cancelling the part of y_r resulting from bits that already have been detected in step one. However, it is not always true that $\hat{b}_c^* = b_c$ even though $\hat{b}_c^*$ is optimal. In [26], there is a discussion about that errors made in the first step will affect the detector in the second step and as a result there is a risk that the detector in the second step might make "additional erroneous decisions". However, note that this offset is *necessary* in order for the detector in the second step to make a decision which makes the entire decision $\begin{bmatrix} \hat{b}_c^T & \hat{b}_r^T \end{bmatrix}^T$ *jointly optimal*. It would be possible to assume that $\hat{b}_c^* = b_c$ and work on a reduced MUD system in the second step with measurements y_r and correlation R_{rr}, but this setup would result in a decision which might not be jointly optimal when considered as one large decision.

5 Numerical Experiments

In this section, the preprocessing algorithm is applied to the multiuser detection problem as described in Section 4 and its performance is evaluated using Monte Carlo simulations. The Signal to Noise Ratio (SNR) used in the simulations is computed after the matched filters. Hence, there are disturbances originating both from the noise $n(t)$ and from the cross-correlation from other users.

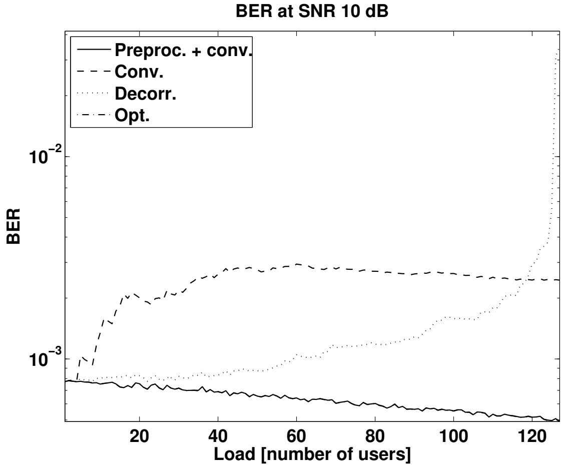


Figure 2: *The plot shows the BER of the different detectors as a function of the load when Gold sequences of length* 127 *are used. It can be seen that the multiuser detector implemented by the preprocessing algorithm combined with the conventional detector gives the lowest BER. Note that, the BER performance of the proposed detector is almost indistinguishable from the BER performance of the optimal detector.*

5.1 Synchronous Case

In the first simulation, the joint Bit Error Rate (BER) is compared for a multiuser detector implemented using a combination of the preprocessing algorithm and the conventional detector, the conventional detector itself in (8), the decorrelating detector in (9) and the optimal detector found by solving the optimization problem in (7) optimally. When the preprocessing algorithm is combined with a conventional detector, any variables not computed by the preprocessing algorithm is computed by the suboptimal detector (8) as described in Section 4.2. Hence, if the preprocessing algorithm does not solve all variables, some parts of the solution might be suboptimal. To be able to obtain an optimal solution, the preprocessing algorithm is combined with CPLEX, where CPLEX is used to solve any remaining variables not possible to compute by preprocessing. The algorithms were compared for the loads 1 to 127 users. In the simulation, Gold sequences of length 127 were used and the SNR for user 1 was chosen to be 10 dB. The SNRs for the other users are then given by the choice of signature sequences and the noise variance. However, in this case, it is close to 10 dB for all users. To get a sufficiently smooth plot in a reasonable time, the number of Monte Carlo simulations used was adjusted in several steps from 10^7 for low loads to 10^5 for high loads. In each Monte Carlo simulation, a new noise realization was used and a new random bit was assigned to each user. The result from this test can be found in Figure 2. The conclusion from this test is that the BER performance of the detector built on a combination of preprocessing and the conventional detector is better than the performance of the conventional detector used alone or the decorrelating detector. In this test, the BER performance of the proposed detector is almost indistinguishable from the BER performance of the optimal detector.

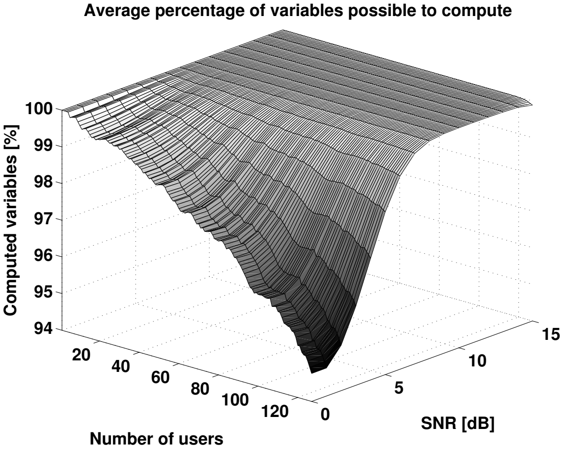


Figure 3: *This plot shows how many percents of the variables that were computed by preprocessing for loads between* 1 *and* 127 *users in combination of SNRs between* 0 *to* 15 *dB. The result shows that even at* 127 *users and at an SNR as low as* 0 *dB, the preprocessing algorithm only fails to compute about* $5\,\%$ *of the total number of variables.*

In Figure 3 it can be seen that for a wide span of loads and SNRs, the preprocessing algorithm computes nearly all variables in average. Therefore, for those combinations of load and SNR, the remaining variables not computed by preprocessing do not alter the BER performance significantly even though they are computed suboptimally. In Figure 3 it can be seen that the number of variables computed by preprocessing decreases for high loads in combination with low SNR. Somewhat counterintuitive, the number of computed variables also decreases slightly for high loads in combination with high SNR. The effect of this degradation in detection performance will be discussed in coming experiments. The conclusion drawn from the simulation is that the preprocessing algorithm can be expected to compute almost all variables for a large span of combinations of SNRs and loads. However, for large loads in combination with low SNRs, the number of variables computed by preprocessing decreases.

In Figure 4 the BER is evaluated for different values of SNR in the range from 0 dB to 15 dB. In the comparison, five different detectors are used; a detector formed by the combination of the preprocessing algorithm and the conventional detector, a combination of the preprocessing algorithm and the decorrelating detector, the conventional detector, the decorrelating detector and finally the optimal detector. Note that, this experiment is also an excellent practical example of the computational performance of the algorithm. Actually, without the computational performance of the preprocessing algorithm, combined with the computational performance of CPLEX, it would have been impossible to perform such rigorous comparisons (a huge number of Monte Carlo simulations are used even at high loads) in a reasonable time where different suboptimal detectors are compared with the *optimal* detector. As expected, the BER performances of the conventional detector in (8) and the decorrelating in (9) are significantly worse compared to the other

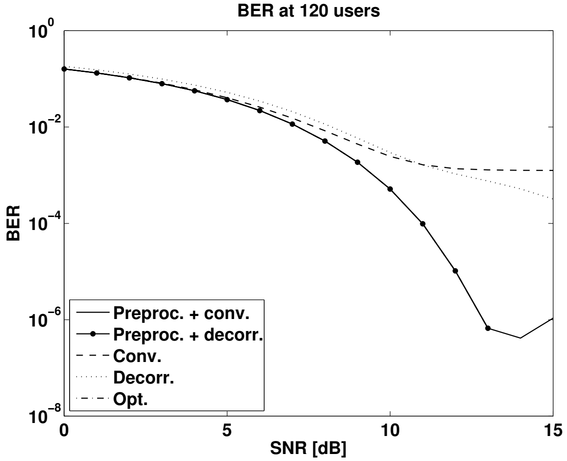


Figure 4: *In this plot the BER is examined when SNR varies from* 0 *dB to* 15 *dB. The result is that at higher SNRs, the conventional and decorrelating detectors cannot offer BERs as low as the preprocessing based detectors. For SNRs below* 13 *dB, the resulting BERs from the preprocessing based detectors are that close to the optimal one that the BER curves cannot be visually separated.*

detectors, specifically for high values of SNR. The BER performance of the detectors built on preprocessing is so close to the optimal one that they cannot be visually distinguished from each other in this plot when SNR is 14 dB or below. Above 14 dB, the detector built on a combination of the preprocessing algorithm and the conventional detector shows a slightly higher BER than the other detectors. The behavior seems to be a consequence of the decrease of variables computed by preprocessing for a combination of high loads and high SNRs shown in Figure 3. This does not occur for loads below 117 users, and can be avoided also for high loads by combining the preprocessing algorithm with the decorrelating detector instead of the conventional detector for high SNRs.

The computational complexities for the different detectors have been previously theoretically discussed. An example of computational times in practice are shown in Figure 5. The conventional detector in (8) is not shown in the plot because its computational time is negligible in comparison with the other two. The conclusion drawn is that the computational complexity for the preprocessing algorithm grows similarly, or slower, compared to the one for the decorrelating detector in (9). Since there are several different ways to implement the different detectors, the computational times presented in Figure 5 should only be considered as guidelines to relate the performance of the different algorithms. For example, the matrix inversion performed in (9) is in Matlab implemented much more efficiently than the m-code implementation of the preprocessing algorithm. By implementing the preprocessing algorithm in, *e.g.*, C, a significant reduction of the computational time is expected. If the result in this experiment is combined with the result from previous experiments, the conclusion for the synchronous case is that the preprocessing algorithm combined with the conventional detector can produce a near optimal BER at a computational time comparable to the one of the decorrelating detector. Furthermore, in this case it

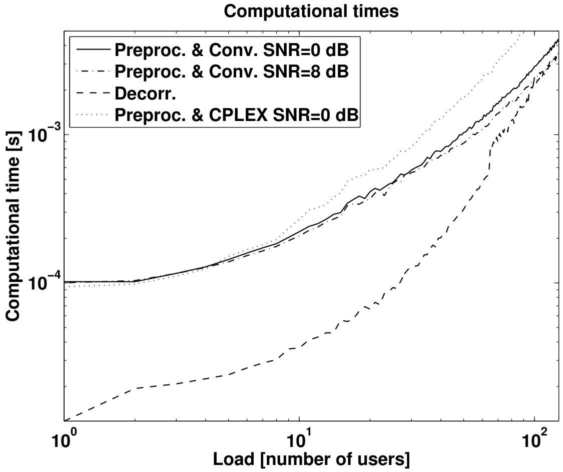


Figure 5: *In this plot, the computational times to detect* 127 *users by the detector using the preprocessing algorithm, first in combination with the conventional detector, and second in combination with* CPLEX*, are compared with the computational times for the decorrelating detector. The conventional detector has significantly lower computational time and it has therefore been excluded from this plot. The conclusion is that the computational complexity of the preprocessing algorithm grows as, or slower than, the computational complexity of the decorrelating detector.*

is probably not worth the extra amount of computational time needed to compute the optimal solution, since that even preprocessing in combination with the conventional detector often gives similar BER performance. Anyhow, it is a good example how computational time can be decreased with an order of magnitude if the structure of the problem is used. The tests of the computational times were performed on a computer with two processors of the type Dual Core AMD Opteron 270 sharing 4 GB RAM (the code was not written to utilize multiple cores) running CentOS release 4.6 (Final) Kernel 2.6.9-55.ELsmp and MATLAB 7.2.0.294.

From an optimization point of view it is interesting to investigate how far the suboptimal detectors are from being optimal. This is illustrated for 100 users in Figure 6, which shows the relative suboptimality here computed as $\frac{f_{\text{subopt}}-f^*}{|f*|}$, where f_{subopt} denotes the objective function value for a suboptimal detector and f^* denotes the optimal objective function value. From the figure it can be seen that up to 13 dB, the combination of the preprocessing algorithm and the conventional detector gives relatively a significantly lower objective function value compared to the conventional or decorrelating detectors used alone. Above 13 dB, the performance of this combination decreases slightly, which can be explained by the drop in the by preprocessing detected number of users for a combination of high load and SNR as shown in Figure 3. To get near optimal performance also in this region, the preprocessing algorithm can be combined with the decorrelating detector.

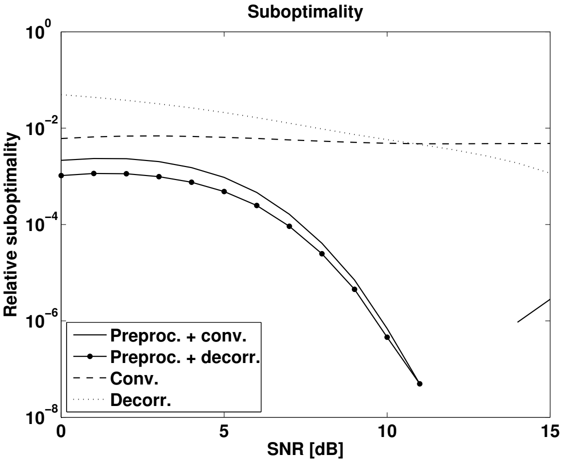


Figure 6: *This plot shows the relative suboptimality for the different detectors computed as $\frac{f_{subopt}-f^*}{|f^*|}$, where f_{subopt} denotes the objective function value for a suboptimal detector and f^* denotes the optimal objective function value. Note that the detector is in fact optimal if the line vanishes from the plot. The line fragment from the combination of preprocessing and the conventional detector visible in the figure between 14 dB and 15 dB originates from the slight drop in detection performance of the preprocessing algorithm for a combination of high loads and large SNRs as shown in Figure 3.*

5.2 Asynchronous Case

In the asynchronous case, the user are no longer assumed to be synchronized in time, *i.e.*, the channel model in (3) is used. The different offsets τ_i for different users are uniformly distributed between 0 and an upper value. Three experiments where this upper value have been chosen to 1, 5 and 64 chips have been considered. The result is presented in Figure 7 for the case when SNR is $5\,\mathrm{dB}$. The conclusion from this experiment is that the cross-correlation between the users is rather dependent on the offsets of the users, and the benefits of the preprocessing algorithm tends to decrease as the maximum possible offset is increased. This is a consequence of the fact that fewer and fewer bits are computed in the preprocessing step as the maximum offset is increased. For low values of this offset, the algorithm is still usefull, while for larger values, the algorithm is only useful for very low loads.

6 Conclusions

In this work, a preprocessing algorithm for BQP problems has been successfully applied to the multiuser detection problem when signature sequences of Gold type have been used. The preprocessing algorithm is able to detect some or all bits. These bits are detected optimally and any remaining bits can be detected either optimally or suboptimally by another algorithm. Numerical experiments have shown that if Gold sequences are used, more than $94.7\,\%$ of the bits are computed by the preprocessing algorithm in the synchronous case. Furthermore, simulations have shown that the preprocessing algorithm combined with a suboptimal algorithm outperforms the suboptimal algorithm used alone in terms of BER and the resulting BER is often very close to the optimal one. Moreover, the computational complexity of the preprocessing algorithm is similar to the one of the simple suboptimal decorrelating detector. The result in this work shows that it can be very advantageous to use the proposed preprocessing algorithm for multiuser detection problems where the cross-correlation between different users is low. Compared to several other suboptimal detectors, often a better BER performance can be expected. The benefits of using the proposed preprocessing algorithm decreases in the asynchronous case. In that case, the preprocessing algorithm is still able to compute a significant number of bits at low loads, but fewer and fewer with increased load. If users are almost synchronous, the preprocessing benefits are still evident, while in the worst case scenario, where users can be misaligned up to half a symbol, the benefits are minor already at moderate loads.

References

[1] J. G. Andrews. Interference cancellation for cellular systems: A contemporary overview. *IEEE Wireless Communications Magazine*, pages 19–29, Apr. 2005.

[2] D. Axehill. *Applications of Integer Quadratic Programming in Control and Communication.* Licentiate's thesis, Linköpings universitet, 2005. URL `http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263`.

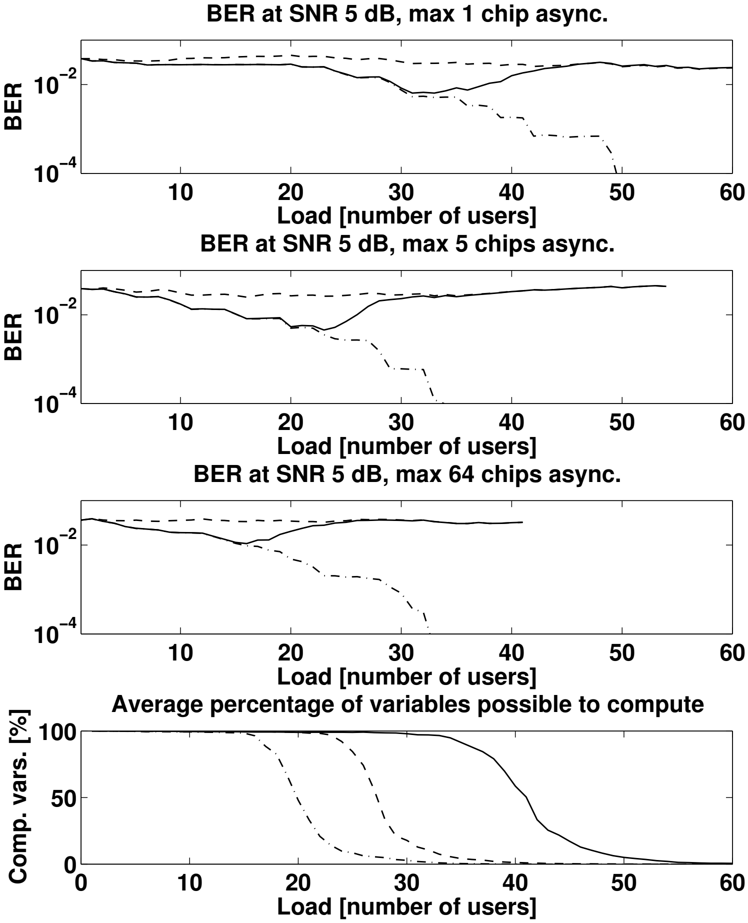


Figure 7: *The three uppermost plots show the BER in an asynchronous system. These plots show three cases where different possible worst-case offsets τ_i among the users have been considered. The offsets are uniformly distributed between zero and this value. The BER of the conventional detector is dashed, the conventional detector in combination with preprocessing is solid, and the optimal is dash-dotted. In the bottommost plot, the number of variables computed by preprocessing is shown for the three different cases of maximum offset. Dash-dotted, dashed and solid lines represent a maximum offset of* 64, 5 *and* 1 *chips, respectively. In the cases where the offset is* 5 *and* 64 *chips, the experiment is aborted at a load lower than* 60 *users because it is not possible in these cases to maintain an SNR of* 5 *dB at high loads.*

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

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

[5] 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.

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

[7] CPLEX's webpage. ILOG CPLEX. Website. URL: http://www.ilog.com/products/cplex, accessed June 10, 2007.

[8] J. Dahl, B. H. Fleury, and L. Vandenberghe. Approximate maximum-likelihood estimation using semidefinite programming. In *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003*, volume 6, pages VI – 721–724, Apr. 2003.

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

[10] 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.

[11] P. L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, 1968.

[12] F. Hasegawa, J. Luo, K. R. Pattipati, P. Willett, and D. Pham. Speed and accuracy comparison of techniques for multiuser detection in synchronous CDMA. *IEEE Transactions on Communications*, 52(4):540 – 545, Apr. 2004.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] W. K. Ma, T. N. Davidson, K. Wong, Z. Q. Luo, and P. Ching. Quasi-maximum-likelihood multiuser detection using semi-definite relaxation. *IEEE Transactions on Signal Processing*, 50(4):912–922, 2002.

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

[20] R. Nilsson, F. Sjöberg, O. Edfors, P. Ödling, H. Eriksson, S. K. Wilson, and P. O. Börjesson. A low complexity threshold detector making MLSD decisions in a multiuser environment. In *Proceedings of the 48th IEEE Vehicular Technology Conference*, pages 333 – 337, May 1998.

[21] 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.

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

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

[24] 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.

[25] S. Verdu. *Multiuser Detection.* Cambridge University Press, 1998.

[26] P. Ödling, H. B. Eriksson, and P. O. Börjesson. Making MLSD decisions by thresholding the matched filter output. *IEEE Transactions on Communications*, 48(2):324 – 332, Feb. 2000.

Paper C

A Mixed Integer Dual Quadratic Programming Algorithm Tailored for MPC

Authors: Daniel Axehill and Anders Hansson

Edited version of the paper: D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698, Manchester Grand Hyatt, San Diego, USA, Dec. 2006

Preliminary version: Published as Technical Report LiTH-ISY-R-2761, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

A Mixed Integer Dual Quadratic Programming Algorithm Tailored for MPC[†]

Daniel Axehill and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,hansson}@isy.liu.se

Abstract

The objective in this work is to develop an MIQP solver tailored for MPC. The MIQP solver is built on the branch and bound method, where QP relaxations of the original problem are solved in the nodes of a binary search tree. The difference between the subproblems is often small and therefore it is interesting to be able to use a previous solution as a starting point in a new subproblem. This is referred to as a warm start of the solver. Because of its good warm start properties, a dual active set QP method was chosen. The method is tailored for MPC by solving a part of the KKT system using a Riccati recursion, which makes the computational complexity of the QP iterations grow linearly with the prediction horizon. Simulation results are presented both for the QP solver itself and when it is incorporated as a part of the MIQP solver. In both cases the computational complexity is significantly reduced compared to if a primal active set solver not utilizing structure is used.

1 Introduction

From the beginning, Model Predictive Control (MPC) was only applicable to linearly constrained linear systems. Nowadays, MPC is applicable to nonlinear systems, and specifically to hybrid systems in Mixed Logical Dynamical (MLD) form, [6]. In the basic linear setup, the MPC problem can be cast in the form of a Quadratic Programming (QP) problem. When MPC is extended to more advanced systems, also the type of optimization problem to solve in each sample is changed. When a hybrid system is to be controlled,

[†] Research supported by the Swedish Research Council for Engineering Sciences under contract Nr. 621-2002-3822 and by the VINNOVA competence centre ISIS.

the corresponding optimization problem is changed from a QP problem to a Mixed Integer Quadratic Programming (MIQP) problem, hence the term Mixed Integer Predictive Control (MIPC) is sometimes used. Today there exist tailored optimization routines with the required performance for linear MPC. However, there is still a need for efficient optimization routines for MIPC. The MIQP problem is in general known to be $\mathcal{NP}$-hard, [22]. Therefore, to be able to use the algorithm in real-time, it is desirable to reduce the computational complexity. One way of doing that is by utilizing problem structure when solving the optimization problem.

A popular method for solving MIQP problems is branch and bound, where the original integer optimization problem is solved as a sequence of smaller QP subproblems. The subproblems are ordered in a tree structure, where one new integer variable is fixed at each level. Depending on the problem, sometimes a large number of QP subproblems have to be solved. Therefore it is important to be able to efficiently solve these subproblems. In this paper, the efficiency of this process is enhanced in two ways. First, because the difference between different subproblems in the tree is small, the solution from a previously solved subproblem is reused as a starting point in a new subproblem. This procedure is often called a warm start of the solver, and is most easily and effectively implemented if a dual active set QP solver is used, [1]. Early work on dual active set solvers can be found in [15] and [20]. A more recent method is found in [11], which has been refined in [17], [7] and [3]. Second, in this dual QP solver a large part of the Karush-Kuhn-Tucker system is solved using a Riccati recursion. This has previously been used in primal active set QP solvers, *e.g.*, [14], and in interior point solvers, *e.g.*, [19]. A more thorough description of the material presented in this paper can be found in [1].

For what follows, two notations will be introduced. First, a set with integers on an interval, $\mathbb{Z}_{i,j} = \{i, i+1, \ldots, j\}$, and second, $\mathbb{S}^n_{++}$ ($\mathbb{S}^n_{+}$) that denotes the set of symmetric positive (semi) definite matrices with n rows.

2 Problem Definition

When the MPC problem is cast in the form of a QP problem, it can either be written as a QP problem with only control signals as free variables or it can be written as a QP problem where control signals, states and control errors all are free variables. In this paper, it will be shown that the latter approach will result in a KKT system possible to solve using a Riccati recursion, and it will be seen that this is advantageous from a computational point of view.

Consider a linear system in state space form

$$\begin{aligned} x(t+1) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) \end{aligned} \tag{1}$$

and the resulting MPC optimization problem

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

where x, u and e denote vectors in which the states, control signals and control errors respectively from all time instants are stacked. Furthermore, $A(t) \in \mathbb{R}^{n\times n}$, $B(t) \in \mathbb{R}^{n\times m}$, $M(t) \in \mathbb{R}^{p\times n}$, $H_x(t) \in \mathbb{R}^{c(t)\times n}$, $H_u(t) \in \mathbb{R}^{c(t)\times m}$ and $h(t) \in \mathbb{R}^{c(t)}$ where $c(t)$ denotes the number of inequality constraints at time t. Also, the following assumptions are made

Assumption 1. $Q_e(t) \in \mathbb{S}^p_{++},\ t = 0,\ldots,N$

Assumption 2. $Q_u(t) \in \mathbb{S}^m_{++},\ t = 0,\ldots,N-1$

3 Optimization Preliminaries

In this paper, an optimization problem such as

$$
\begin{array}{ll}
\underset{x}{\text{minimize}} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0,\ i = 1,\ldots,m \\
 & h_i(x) = 0,\ i = 1,\ldots,p
\end{array}
\tag{3}
$$

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 in standard form. The optimal objective function value of (3) is denoted p^*. Associated with each optimization problem there is a Lagrange dual problem in the form

$$
\begin{array}{ll}
\underset{\lambda,\nu}{\text{maximize}} & g(\lambda,\nu) \\
\text{subject to} & \lambda \geq 0
\end{array}
\tag{4}
$$

where $g(\lambda,\nu)$ is called the Lagrange dual function. Duality is thoroughly discussed in, *e.g.*, [8]. An important property of the Lagrange dual function is that for any $\lambda \geq 0$, the following inequality holds

$$g(\lambda,\nu) \leq p^* \tag{5}$$

The optimal dual objective function value is denoted d^*. From (5) the following inequality can be derived

$$d^* \leq p^* \tag{6}$$

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. For some problems, the inequality in (6) can be shown to hold with equality, *i.e.*,

$$d^* = p^* \tag{7}$$

This important property is called strong duality.

In this paper the KKT conditions are used as optimality conditions. They are stated in the following theorem, which is based on the discussion in [8, pp. 243–244].

Theorem 1 (KKT)
Consider the optimization problem in (3)*. Assume that it is convex, that* $f_i(x),\ i = 0, \dots, m$ *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*

$$
\begin{align}
&f_i(x^*) \leq 0,\ i = 1, \dots, m \tag{8a}\\
&h_i(x^*) = 0,\ i = 1, \dots, p \tag{8b}\\
&\lambda_i^* \geq 0,\ i = 1, \dots, m \tag{8c}\\
&\lambda_i^* f_i(x^*) = 0,\ i = 1, \dots, m \tag{8d}\\
&\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0 \tag{8e}
\end{align}
$$

Proof: See [8, p. 244]. □

4 Quadratic Programming

In this paper the following special case of a general QP will be of interest

$$
\begin{array}{ll}
\underset{x_1, x_2}{\text{minimize}} & \frac{1}{2} \begin{bmatrix} x_1^T & x_2^T \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{H} & 0 \\ 0 & 0 \end{bmatrix}}_{\triangleq H} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \tilde{f}^T & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
\text{subject to} & \begin{bmatrix} A_{\mathcal{E},1} & A_{\mathcal{E},2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = b_{\mathcal{E}} \\
& \begin{bmatrix} A_{\mathcal{I},1} & A_{\mathcal{I},2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq b_{\mathcal{I}}
\end{array} \tag{9}
$$

where $x_1 \in \mathbb{R}^{n_1}$, $x_2 \in \mathbb{R}^{n_2}$, $n = n_1 + n_2$, $\tilde{H} \in \mathbb{S}_{++}^{n_1}$, $\tilde{f} \in \mathbb{R}^{n_1}$. This QP problem differs from what is normally considered in the literature in the context of QP duality, where the Hessian often is assumed positive definite. The dual problem is found by forming the Lagrange dual function and performing maximization over λ and ν. This is thoroughly described in [1]. A problem equivalent to the dual problem of (9) can then be found to be

$$\begin{aligned} \underset{\lambda,\nu}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}\lambda^T & \nu^T\end{bmatrix}\begin{bmatrix}A_{\mathcal{I},1}\\ A_{\mathcal{E},1}\end{bmatrix}\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix}\begin{bmatrix}\lambda\\ \nu\end{bmatrix}+ \\ & +\left(\tilde{f}^T\tilde{H}^{-1}\begin{bmatrix}A_{\mathcal{I},1}^T & A_{\mathcal{E},1}^T\end{bmatrix}+\begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda\\ \nu\end{bmatrix} \\ \text{subject to} \quad & A_{\mathcal{I},2}^T\lambda + A_{\mathcal{E},2}^T\nu = 0 \\ & \lambda \geq 0 \end{aligned} \tag{10}$$

By saying that (10) is an equivalent problem to the dual problem of (9) it is meant that given the solution to one of the equivalent problems the other one can easily be derived from the first one. According to [1], some important strong duality results can be stated for this problem. First, if the primal problem is feasible, then the primal and dual optimal objective function values coincide. Second, the primal is feasible if and only if the dual optimal objective function value is bounded from above. At large, these results also hold for the problem in (10) equivalent to the dual, but some extra care should be taken since the sign of the objective function has been changed and the objective function value has in general been off-set. For an extensive bibliography on QP, see [13].

4.1 Active Set Methods

A QP can either be solved by an active set method or an Interior Point (IP) method. Compared to active set methods, IP methods are often preferred for large problems since their computational performance is less sensitive to the number of constraints, [4]. On the other hand, active set methods can gain more from warm starts, [23], which makes them preferable for solving a sequence of slightly modified problems. Because of their good warm start properties, an active set method was chosen in this paper. For a thorough description of an active set method applicable to QP, see [16]. Briefly, an active set method solves an inequality constrained optimization problem by solving a sequence of equality constrained optimization problems. The set containing the equality constraints is called the working set, and it contains all original equality constraints and usually some of the original inequality constraints treated as equality constraints. Solving an equality constrained problem involves the solution of a KKT system in the form

$$\begin{bmatrix}H & A^T\\ A & 0\end{bmatrix}\begin{bmatrix}x\\ \nu\end{bmatrix} = \begin{bmatrix}-f\\ b\end{bmatrix} \tag{11}$$

Since one such system is solved in each QP iteration, it is very important to be able to solve systems of this type efficiently. For the MPC application, this has previously been done for primal active set methods in [14]. In this paper it will be shown how this can be performed for a dual active set algorithm as well.

A primal active set method starts in a primal feasible point and primal feasibility is maintained during all subsequent QP iterations. In a dual active set method, dual feasibility is instead maintained. One of the main advantages with a dual QP algorithm is that it is in many cases much easier to find an initial dual feasible point than an initial primal feasible point. Usually, an initial feasible point is found by a so-called Phase I algorithm.

According to [11], 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. Consider the dual problem in (10). It is easy to see that the origin, *i.e.*, $\lambda = 0$ and $\nu = 0$, is always a feasible point both with respect to the equality and the inequality constraints. Hence, an initial dual feasible point can easily be found.

The solution time for an active set algorithm is much dependent on how many QP iterations that have to be performed. If the optimal active set was known in advance, the process would terminate in a single QP iteration. This is the idea behind warm starts, where a slightly modified problem can be solved efficiently by utilizing information from a previously solved problem. Due to the simple structure of the inequality constraints in the dual problem, a dual active set QP algorithm is very easy to warm start. This property makes it suitable for, *e.g.*, Sequential Quadratic Programming (SQP), where several similar inequality constrained QPs are solved sequentially, [17].

The structure of the dual problem also makes it possible to use the gradient projection method efficiently, which allows for rapid changes to the working set, [16]. If this method is employed, the number of QP iterations can be reduced.

5 Mixed Integer Quadratic Programming

An MIQP problem is similar to the ordinary QP problem in (9), but the optimization is to be performed for $x \in \mathbb{R}^{n_c} \times \{0, 1\}^{n_b}$. This means that 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, [22]. In this paper, a common special case of MIQP, *i.e.*, when the integer variables are constrained to be 0 or 1, is considered.

There exist several methods for solving MIQP problems. Among them several authors consider the branch and bound method as the best choice for MIQP problems, [6, 9]. An integer program can in principle be solved by explicit enumeration, where the objective function value is evaluated for all integer combinations and the best combinations are chosen as the optimizers. Branch and bound cuts down the number of combinations necessary to evaluate. A thorough description of the branch and bound algorithm can be found in, *e.g.*, [22] and [10].

In branch and bound for MIQP, QP subproblems are solved in the tree. According to [9], solving the subproblems using a dual active set method offers the most straightforward way to exploit the structure introduced in the branching procedure. According to [22], active set methods (the reference considers the linear programming case) is preferable for solving the relaxed problems in branch and bound. However, for very large problems, IP algorithms can be used to solve the first subproblem, but in the subsequent subproblems an active set method should be used. After a branch, the solution to the parent primal problem is in general infeasible in the child primal problems. But, as later will become apparent, 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 a dual active set solver using information from the solution of the parent problem. 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 in the subproblems prematurely, [9].

An important step in an MIQP solver is the preprocessing step, where the problem is reduced if possible and the formulation is made as strong as possible. A preprocessing algorithm for unconstrained MIPC is presented in [2].

6 A Dual Quadratic Programming Algorithm

The idea in this paper is to solve a primal problem in the form in (9) by solving the associated Lagrange dual problem in the form in (10). In the first part of the derivation the following assumption is made

Assumption 3. $H_u(t)$ has full row rank for $t = 0, \dots, N-1$.

Following the general procedure in Section 3, a problem equivalent to the dual problem can be found to be

$$
\begin{aligned}
\underset{\tilde{\mathrm{x}},\tilde{\mathrm{u}}}{\text{minimize}} \quad & \frac{1}{2}\tilde{u}^T(-1)\tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) \\
& + \frac{1}{2}\sum_{\tau=0}^{N-1}\big(\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) + \tilde{u}^T(\tau)\tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) \\
& + 2\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau) + 2\tilde{q}_{\tilde{u}}^T(\tau)\tilde{u}(\tau)\big) + \tilde{q}_{\tilde{x}}^T(N)\tilde{x}(N) \qquad (12) \\
\text{subject to} \quad & \tilde{x}(0) = \tilde{B}(-1)\tilde{u}(-1) \\
& \tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}(\tau)\tilde{u}(\tau),\ \tau = 0, \dots, N-1 \\
& \begin{bmatrix} 0 & -I_{c(N-\tau-1)} \end{bmatrix}\tilde{u}(\tau) \leq 0,\ \tau = 0, \dots, N-1
\end{aligned}
$$

where the parameters and variables are defined as in [1, p. 75]. More on duality in control problems can be found in, *e.g.*, [18] and [12].

If the dual problem is feasible it follows from Slater's refined condition (see [1, p. 14]) that strong duality holds. Then the KKT conditions in Theorem 1 constitutes the following necessary and sufficient conditions for optimality

$$\begin{bmatrix} 0 & -I_{c(N-\tau-1)} \end{bmatrix}\tilde{u}(\tau) \leq 0,\ \tau = 0, \dots, N-1 \qquad (13a)$$

$$\tilde{x}(0) = \tilde{B}(-1)\tilde{u}(-1) \qquad (13b)$$

$$\tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}(\tau)\tilde{u}(\tau),\ \tau = 0, \dots, N-1 \qquad (13c)$$

From the dual feasibility condition in (8c), it holds that

$$\mu(\tau) \geq 0,\ \tau = 0, \dots, N-1 \qquad (14)$$

The complementary slackness condition in (8d) gives the equation

$$
\begin{aligned}
& \mu_i(\tau)\tilde{u}_{\tilde{m}(\tau)-c(N-\tau-1)+i}(\tau) = 0, \\
& i = 1, \dots, c(N-\tau-1),\ \tau = 0, \dots, N-1
\end{aligned} \qquad (15)
$$

Finally, from (8e) the following equations are obtained

$$
\begin{aligned}
\frac{\partial L_D}{\partial \tilde{x}(\tau)} &= \tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) + \tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau) + \tilde{A}^T(\tau)\lambda(\tau+1) - \lambda(\tau) = 0, && \text{(16a)}\\
&\tau = 0, \ldots, N-1 && \text{(16b)}\\
\frac{\partial L_D}{\partial \tilde{x}(N)} &= \tilde{q}_{\tilde{x}}(N) - \lambda(N) = 0 && \text{(16c)}\\
\frac{\partial L_D}{\partial \tilde{u}(\tau)} &= \tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) + \tilde{Q}^T_{\tilde{x}\tilde{u}}(\tau)\tilde{x}(\tau) + \tilde{q}_{\tilde{u}}(\tau) + \tilde{B}^T(\tau)\lambda(\tau+1) - \begin{bmatrix} 0 \\ I \end{bmatrix} \mu(\tau) = 0, &&\\
&\tau = 0, \ldots, N-1 && \text{(16d)}\\
\frac{\partial L_D}{\partial \tilde{u}(-1)} &= \tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) + \tilde{B}^T(-1)\lambda(0) = 0 && \text{(16e)}
\end{aligned}
$$

Using a general notation, for a specific QP iteration the KKT conditions for the dual problem, excluding the non-linear complementary slackness condition in (15), is given by a linear system in the form

$$
\left[\begin{array}{c:c} K_{11} & K_{12} \\ \hdashline K_{21} & K_{22} \end{array}\right] \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \left[\begin{array}{cc:c} H & A_{\mathcal{E}}^T & A_{\mathcal{I}\cap\mathcal{W}}^T \\ A_{\mathcal{E}} & 0 & 0 \\ \hdashline A_{\mathcal{I}\cap\mathcal{W}} & 0 & 0 \end{array}\right] \left[\begin{array}{c} \hat{x} \\ \hat{\nu} \\ \hdashline \hat{\lambda}_{\mathcal{I}\cap\mathcal{W}} \end{array}\right] = \left[\begin{array}{c} -f \\ b_{\mathcal{E}} \\ \hdashline b_{\mathcal{I}\cap\mathcal{W}} \end{array}\right] \tag{17}
$$

where $A_{\mathcal{I}\cap\mathcal{W}}$ contains the rows in $A_{\mathcal{I}}$ corresponding to the inequality constraints contained in the working set and $b_{\mathcal{I}\cap\mathcal{W}}$ is defined analogously. Furthermore, $\hat{\lambda}_{\mathcal{I}\cap\mathcal{W}}$ contains the dual variables corresponding to constraints in the working set. The matrix $A_{\mathcal{E}}$ and the vector $b_{\mathcal{E}}$ define the equality constraints in the dual 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 (17) is unchanged during the QP iterations it first seems possible to proceed as in [14], where it is proposed to use block elimination and Riccati recursions in order to solve (17). Trying to follow the same approach in the dual might cause 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 similarly as in [14] when solving the dual problem.

Since it is generally not possible to compute K_{11}^{-1}, an alternative efficient algorithm has to be developed. For what follows, it is 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 (12), let $\bar{x}_1 = \begin{bmatrix} \tilde{\mathsf{x}}^T & \mathsf{w}^T & \lambda^T & \mathsf{v}^T_{\mathcal{W}_{\mathsf{v}}^{\mathsf{c}}} \end{bmatrix}^T$ and $\bar{x}_2 = \begin{bmatrix} \mathsf{v}^T_{\mathcal{W}_{\mathsf{v}}} & \mu^T_{\mathcal{W}_{\mathsf{v}}} \end{bmatrix}^T$, 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 in the form

$$
\left[\begin{array}{c:c} \bar{K}_{11} & \bar{K}_{12} \\ \hdashline \bar{K}_{21} & \bar{K}_{22} \end{array}\right] \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \left[\begin{array}{c:cc} \bar{R}_{11} & \bar{R}_{12} & 0 \\ \hdashline \bar{R}_{21} & \bar{R}_{22} & -I \\ 0 & -I & 0 \end{array}\right] \left[\begin{array}{c} \bar{x}_1 \\ \hdashline \bar{x}_{2,1} \\ \bar{x}_{2,2} \end{array}\right] = \left[\begin{array}{c} \bar{b}_1 \\ \hdashline \bar{b}_{2,1} \\ \bar{b}_{2,2} \end{array}\right] \tag{18}
$$

Note that $\bar{K}_{22}$ is non-singular. In (18), it has been used that the inequality constraints are in the form $-v(\tau) \leq 0$, which implies that the matrix containing the coefficient for $\mathsf{v}_{\mathcal{W}_\mathsf{v}}$ is $-I$. Dual variables corresponding to constraints not in the working set, $\mu_{\mathcal{W}_\mathsf{v}^\complement}$, are *set* to zero. Also, note that $\bar{b}_{2,2} = 0$. This follows from the fact that the right hand side of the inequality in (13a) 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

$$\begin{aligned} \bar{x}_1 &= \left(\bar{K}_{11} - \bar{K}_{12}\bar{K}_{22}^{-1}\bar{K}_{21}\right)^{-1}\left(\bar{b}_1 - \bar{K}_{12}\bar{K}_{22}^{-1}\bar{b}_2\right) \\ \bar{x}_2 &= \bar{K}_{22}^{-1}\left(\bar{b}_2 - \bar{K}_{21}\bar{x}_1\right) \end{aligned} \tag{19}$$

Notice that

$$\bar{K}_{22}^{-1} = \begin{bmatrix} 0 & -I \\ -I & -\bar{R}_{22} \end{bmatrix} \tag{20}$$

Furthermore,

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

From this it immediately follows that

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

Using (21) and (22) in the expression for $\bar{x}_1$ in (19), the following simplified equation for $\bar{x}_1$ is obtained

$$\bar{R}_{11}\bar{x}_1 = \bar{b}_1 + \bar{R}_{12}\bar{b}_{2,2} = \bar{b}_1 \tag{23}$$

where the last equality follows from $\bar{b}_{2,2} = 0$. By using (20) and that $\bar{b}_{2,2} = 0$, the expression for $\bar{x}_2$ can be simplified to

$$\bar{x}_2 = \begin{bmatrix} 0 \\ \bar{R}_{21}\bar{x}_1 - \bar{b}_{2,1} \end{bmatrix} \tag{24}$$

where $\bar{R}_{21}$ is block diagonal. As a consequence, when $\bar{x}_1$ has been computed, $\mathsf{v}_{\mathcal{W}_\mathsf{v}}$ and $\mu_{\mathcal{W}_\mathsf{v}}$ can easily be computed as

$$\mathsf{v}_{\mathcal{W}_\mathsf{v}} = 0, \; \mu_{\mathcal{W}_\mathsf{v}} = \bar{R}_{21}\bar{x}_1 - \bar{b}_{2,1} \tag{25}$$

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 (13) and (16). From (18), it follows that (16) should be modified by setting components in $\mathsf{v}_{\mathcal{W}_\mathsf{v}}$ and $\mu_{\mathcal{W}_\mathsf{v}^\complement}$ to zero. After dropping time indices and a suitable reordering of the variables and equations, (23) can be written as

$$\begin{bmatrix} \tilde{Q}_{\tilde{u}} & \tilde{B}^T & 0 & 0 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ \tilde{B} & 0 & -I & 0 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\ 0 & -I & \tilde{Q}_{\tilde{x}} & \tilde{Q}_{\tilde{x}\tilde{u}} & \tilde{A}^T & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\ 0 & 0 & \tilde{Q}_{\tilde{x}\tilde{u}}^T & \tilde{Q}_{\tilde{u}} & \tilde{B}^T & 0 & 0 & \dots & \dots & \dots & \dots & \vdots \\ 0 & 0 & \tilde{A} & \tilde{B} & 0 & -I & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \tilde{A} & \tilde{B} & 0 & -I \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0 & -I & 0 \end{bmatrix} \bar{x}_1' = \bar{b}_1' \tag{26}$$

where $\bar{x}_1'$ and $\bar{b}_1'$ are defined as

$$\bar{x}_1' = \begin{bmatrix} \tilde{u}(-1) \\ \lambda(0) \\ \tilde{x}(0) \\ \tilde{u}(0) \\ \vdots \\ \tilde{x}(N-1) \\ \tilde{u}(N-1) \\ \lambda(N) \\ \tilde{x}(N) \end{bmatrix}, \quad \bar{b}_1' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\tilde{q}_{\tilde{u}}(0) \\ 0 \\ \vdots \\ -\tilde{q}_{\tilde{u}}(N-1) \\ -\tilde{q}_{\tilde{x}}(N) \end{bmatrix} \tag{27}$$

where the system in (26) includes all components of $w(\tau)$ but only the components in $v(\tau)$ not included in a constraint in the working set since these are directly set to zero.

It is shown in [1], that if Assumption 3 is satisfied, then $\tilde{Q}_{\tilde{u}}(\tau) \in \mathbb{S}_{++}^{\tilde{m}(\tau)}$. Furthermore, if $\tilde{Q}_{\tilde{u}}(\tau) \in \mathbb{S}_{++}^{\tilde{m}(\tau)}$, a system of equations in the form in (26) can be solved using Riccati recursions. This is shown in detail in, *e.g.*, [1], [24] and [21]. Since all computations necessary to solve the KKT system, including the Riccati recursion, [21, 24], can be performed with linear computational complexity in the time horizon N, the entire operation can be concluded to have linear computational complexity in N.

As already concluded in Section 4.1, the origin can always be used as a feasible initial point. Hence, the need for a Phase I algorithm has been eliminated. Also, it can be realized that the origin is always feasible independently of the working set chosen, *i.e.*, reuse of old working sets during warm starts can be easily performed.

Unfortunately, Assumption 3 makes it impossible to have upper and lower bound constraints on the optimization variables since then the constraint gradients become linearly dependent. To be able to use the QP algorithm in a branch and bound framework where binary constraints are relaxed to interval constraints, Assumption 3 has to be relaxed. Consider constraints in the form

$$\begin{bmatrix} \bar{H}_x(t) \\ \hat{H}_x(t) \\ -\hat{H}_x(t) \end{bmatrix} x(t) + \begin{bmatrix} \bar{H}_u(t) \\ \hat{H}_u(t) \\ -\hat{H}_u(t) \end{bmatrix} u(t) \leq \begin{bmatrix} \bar{h}(t) \\ \hat{h}_+(t) \\ -\hat{h}_-(t) \end{bmatrix} \tag{28}$$

where the top block contains general constraints, and the second and third block together contain upper and lower bound constraints defining a set with strictly feasible points, *i.e.*, $\hat{h}_+(t) > \hat{h}_-(t)$. A relaxed assumption can be formulated as

Assumption 4. $\begin{bmatrix} \bar{H}_u^T(t) & \hat{H}_u^T(t) \end{bmatrix}^T$ in (28) has full row rank and $\hat{h}_+(t) > \hat{h}_-(t)$ for all $t = 0, \dots, N-1$.

It is shown in [1], that if the initial working set is properly chosen, the QP algorithm still works under Assumption 4.

7 A Mixed Integer Quadratic Programming Algorithm

In this section it is described how the dual active set solver developed in Section 6 can be used to solve the subproblems in branch and bound. 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 in (9) and the problem equivalent to the dual in (10). The subproblems to be solved in the nodes of the branch and bound tree will be of the type in (10). A branch can be interpreted as if a primal inequality constraint is converted to a primal equality constraint. Note that ν and λ are the Lagrange multiplier vectors corresponding to the equality constraints and the inequality constraints respectively. During a branch, the number of elements in ν is increased by one and the number of elements in λ is decreased by one. Hence, a new initial working set can be found from the working set of the parent problem by simply removing any inequality constraint regarding the removed element in λ from the working set. The interpretation in the dual MPC problem in (12) is that a sign constrained input signal is replaced by an unconstrained input signal. The conclusion is that during a branch, the feasible set of the dual problem is enlarged and the old feasible set is a subset of the new one. Hence, a feasible dual solution is also a feasible dual solution after a branch.

A possible enhancement, not yet implemented, enabled by the use of a dual solver is to utilize the inequality in (5) to prematurely abort the solution process of a subproblem.

8 Examples

In this section, the performance of the presented algorithms is evaluated. A more thorough description of the setup and the results can be found in [1]. The tests have been performed on an 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 times are obtained using the MATLAB command `cputime`. In the tests, the main objective has been to compare the performance in terms of computational complexity instead of absolute computational time. The absolute values are highly dependent on the level of code optimization, which has been left as future work.

8.1 Dual Active Set QP

In the computations presented in this section, the dual active set QP algorithm is applied to the MPC problem of controlling a mass in one dimension with a single real control signal $u(t)$. The magnitude of $u(t)$ is limited according to $|u(t)| \leq 9$. By limiting the magnitude of $u(t)$ to 9, a reasonable amount of constraints are active at the optimum.

The problem has been solved for different prediction horizons and the computational times have been measured. The result is shown in Figure 1. The QP algorithm presented in this paper is implemented in the function `drqp`. In the tests, `drqp` is used to solve the problem first from scratch and, second, given the optimal active set. 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. 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 QP itera-

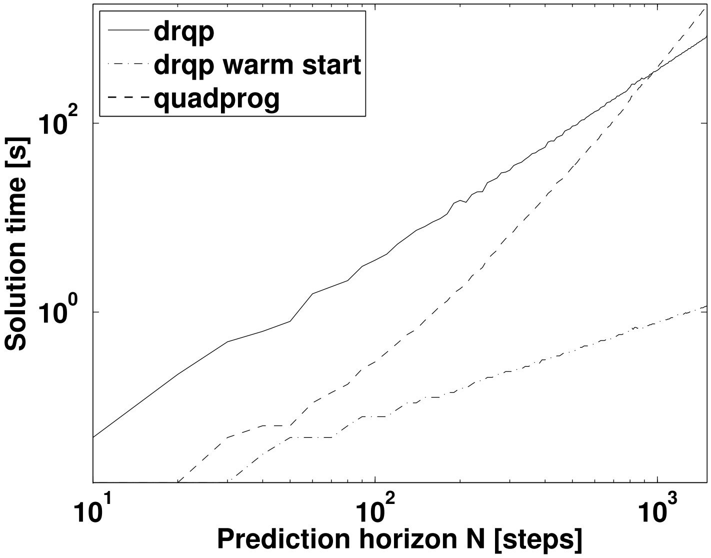


Figure 1: *This plot shows the computational times for two different QP solvers. The QP algorithm described in this paper is implemented in the function* `drqp`*. The solid line shows the computational time when this algorithm solves the problem from scratch. The dash-dotted line shows the computational time when a warm start is simulated when using* `drqp`*. The dashed line shows the computational time for the standard QP solver* `quadprog`*.*

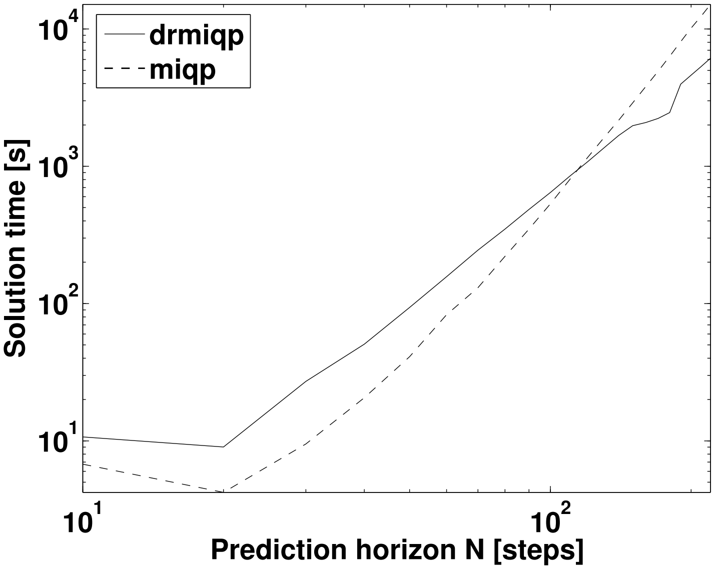


Figure 2: *This plot shows the computational times for two different MIQP solvers. The MIQP algorithm described in this paper is implemented in the function* `drmiqp`*. The solid line shows the computational time when this algorithm is used. The dashed line shows the computational time for the standard MIQP solver* `miqp`*.*

tion in the form described in Section 6 had to be performed and those are since previously concluded to have computational complexity $\mathcal{O}(N)$. 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 paper has a significantly lower computational complexity compared to the generic algorithm used in `quadprog`. Also, warm starts are found to be very efficient.

8.2 MIQP

In the current section, the MIQP algorithm presented in Section 7 is applied to the MPC problem to control a satellite with three actuators; two oppositely directed external thrusters and one internal reaction wheel. The thrusters are assumed to be controlled binary and the magnitude of the control signal to the reaction wheel is limited to less than or equal to one. The system contains three states, *i.e.*, the satellite attitude, the angular velocity and the internal wheel velocity. 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 2. The MIQP algorithm presented in this paper is implemented in the function `drmiqp`. In this example, for prediction horizons longer than 20 time steps, `drmiqp` has an approximate computational complexity of $\mathcal{O}(N^{2.7})$, while the standard function `miqp`, [5], using the QP solver `quadprog`, has an approximate computational complexity of $\mathcal{O}(N^{3.4})$. 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 drqp and to enable the use of warm starts. Hence, it is exactly the same branch and bound code used in the results for drmiqp and miqp, and therefore, the branch and bound tree has been explored in exactly the same way. The conclusion from the test is that the MIQP algorithm presented in this paper has a significantly lower computational complexity compared to the generic MIQP algorithm used in miqp.

9 Conclusions

In this paper a dual active set QP solver tailored for MPC has been derived. By utilizing problem structure, the QP iterations are performed with computational complexity $\mathcal{O}(N)$. The tailored solver has also been used in an MIQP solver where the warm start properties of the dual active set type of solver have been utilized. Simulation results indicate that both the QP solver itself and the MIQP solver get a significantly lower computational complexity compared to if a standard primal active set QP solver is used. Some suggestions to future work are to, first, try to apply the gradient projection method and, second, relax Assumption 4, in order to allow for pure state constraints and, third, to abort the QP solver in the MIQP solver prematurely.

References

[1] D. Axehill. *Applications of Integer Quadratic Programming in Control and Communication*. Licentiate's thesis, Linköpings universitet, 2005. URL http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263.

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

[3] 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.

[4] 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.

[5] 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.

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

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

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

[9] 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.

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

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

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

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

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

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

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

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

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

[19] 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.

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

[21] 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.

[22] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

[23] 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.

[24] M. Åkerblad. *A Second Order Cone Programming Algorithm for Model Predictive Control.* Licentiate's thesis, Royal Institute of Technology, 2002.

Paper D

A Dual Gradient Projection Quadratic Programming Algorithm Tailored for Mixed Integer Predictive Control

Authors: Daniel Axehill and Anders Hansson

Published as Technical Report LiTH-ISY-R-2833, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

A Dual Gradient Projection Quadratic Programming Algorithm Tailored for Mixed Integer Predictive Control

Daniel Axehill and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,hansson}@isy.liu.se

Abstract

The objective of this work is to derive a Mixed Integer Quadratic Programming algorithm tailored for Model Predictive Control for hybrid systems. The Mixed Integer Quadratic Programming algorithm is built on the branch and bound method, where Quadratic Programming relaxations of the original problem are solved in the nodes of a binary search tree. The difference between these subproblems is often small and therefore it is interesting to be able to use a previous solution as a starting point in a new subproblem. This is referred to as a warm start of the solver. Because of its warm start properties, an algorithm that works similar to an active set method is desired. A drawback with classical active set methods is that they often require many iterations in order to find the active set in optimum. So-called gradient projection methods are known to be able to identify this active set very fast. In the algorithm presented in this report, an algorithm built on gradient projection and projection of a Newton search direction onto the feasible set is used. It is a variant of a previously presented algorithm by the authors and makes it straightforward to utilize the previous result, where it is shown how the Newton search direction for the dual MPC problem can be computed very efficiently using Riccati recursions. As in the previous work, this operation can be performed with linear computational complexity in the prediction horizon. Moreover, the gradient computation used in the gradient projection part of the algorithm is also tailored for the problem in order to decrease the computational complexity. Furthermore, it is shown how a Riccati recursion still can be useful in the case when the system of equations for the ordinary search direction is inconsistent. In numerical experiments, the algorithm shows good performance, and it seems like the gradient projection strategy efficiently cuts down the number of Newton steps necessary to compute in order to reach the solution. When the algorithm is used as a part of an MIQP solver for hybrid MPC, the performance is still very good for small problems. However, for more difficult problems, there still seems to be some more work to do in order to get the performance of the commercial state-of-the-art solver CPLEX.

1 Introduction

In recent years, the field of application of the popular control strategy Model Predictive Control (MPC) has been broadened in several steps. From the beginning, MPC was only applicable to linearly constrained linear systems. Today, MPC is applicable to nonlinear systems as well as to hybrid systems in Mixed Logical Dynamical (MLD) form, [6]. The focus in this work will be on control of hybrid systems. In the basic linear setup, the MPC problem can be cast in the form of a Quadratic Programming (QP) problem. When MPC is extended to more advanced systems, also the type of optimization problem to solve in each sample is changed. When a hybrid system is to be controlled, the corresponding optimization problem is changed from a QP problem into a Mixed Integer Quadratic Programming (MIQP) problem, and hence, the term Mixed Integer Predictive Control (MIPC) is sometimes used. Today there exist tailored optimization routines with the required performance for linear MPC. However, there is still a need for efficient optimization routines for MIPC. The MIQP problem is in general known to be $\mathcal{NP}$-hard, [40]. Therefore, to be able to use the algorithm in real-time, it is desirable to reduce the computational complexity. One way of doing that is by utilizing problem structure when solving the optimization problem.

A popular method for solving MIQP problems is branch and bound, where the original integer optimization problem is solved as a sequence of QP problems. Depending on the problem, sometimes a large number of QP problems have to be solved. Therefore, it is important to be able to solve these subproblems efficiently. In this work, the efficiency of this process is enhanced in two ways. First, since the difference between different subproblems is small, the solution from a previously solved subproblem is reused as a starting point in a new subproblem. This procedure is often called a warm start of the solver, and it is most easily and efficiently implemented if a dual active set QP solver is used, [1]. Early work on dual active set solvers can be found in [25] and [37]. A more recent method is found in [19], which has been refined in [33], [9] and [4]. Second, in the dual QP solver to be presented, a large part of the KKT system is solved using a Riccati recursion. This has previously been done in primal active set QP solvers, *e.g.*, [24], and in interior point solvers, *e.g.*, [35]. The material presented in this report is based on an extension of the work presented in [1] and in [3], where a dual active set QP solver tailored for MIPC built on a classical active set method is presented. In this report, the classical active set method has been replaced by a gradient projection method which has the potential to give better performance, especially for problems with many constraints that are active at the optimum, [7]. Early work on gradient projection methods can be found in [20] and in [26]. Many articles have been written about gradient projection. In [29], a method is presented which is very similar to the one used in this report. The difference is that in [29] basic gradient projection iterations are combined with conjugated gradient iterations, while in this report, the gradient projection iterations are combined with projected Newton steps which can be very efficiently computed using Riccati recursions.

In this report, $\mathbb{S}^n_{++}$ ($\mathbb{S}^n_{+}$) denotes the set of symmetric positive (semi) definite matrices with n rows. Further, let $\mathbb{Z}$ be the set of integers and $\mathbb{Z}_{++}$ be the set of positive (non-zero) integers.

All computational performance tests have been performed on a computer with two processors of the type Dual Core AMD Opteron 270 sharing 4 GB RAM (the code was not written to utilize multiple cores) running CentOS release 4.6 (Final) Kernel 2.6.9-55.ELsmp and MATLAB 7.2.0.294. Computational times have been calculated using the MATLAB command `cputime`. CPLEX has been called using the freely available MATLAB interface CPLEXINT, which was slightly modified in order to be able extract more information about the solution process from CPLEX. In all computational times presented for CPLEX, the time spent in CPLEXINT is also included. The latter is assumed to be negligible in comparison with the time spent in CPLEX itself. Since all ideas presented in this work can be illustrated without solving the MPC optimization problem repeatedly as done in practice, the problem is only solved in *one* time instant. Therefore, the word *time instant* is from now on used to refer to a certain time in the *prediction* of the future behavior of the system.

This report is organized as follows. In the remainder of this section, the MPC problem studied in this work is presented. In Section 2, an overview of MIQP is given. In Section 3, the QP background to be used later in the report is presented. Furthermore, some different solution methods for QPs are discussed. In Section 4, the general ideas in the dual gradient projection QP solver presented in this report are discussed. In Section 5, efficient algorithms for computation of search directions are presented. In Section 6, the solver is incorporated into a branch and bound method. In Section 7, numerical experiments are used to evaluate the performance of the algorithm. In the Appendix, definitions and different optimization related results can be found.

1.1 Problem Definition

As for linear MPC, there are at least two different equivalent optimization problem formulations of the MIPC problem. First, it can be written as an MIQP problem with only control signals as free variables, and second, it can be written as an MIQP where control signals, states and control errors all are free variables. The derivations of both formulations for a general linear MPC problem can be found in [27] (and in Appendix B in Part I of this thesis). 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 report, it will be seen that the latter system is in a form which is advantageous from a computational point of view.

Consider an MIPC problem in the form

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

where

$$
\begin{aligned}
&\mathsf{x} = \left[x^T(0),\dots,x^T(N)\right]^T, \ \mathsf{u} = \left[u^T(0),\dots,u^T(N-1)\right]^T, \\
&\mathsf{e} = \left[e^T(0),\dots,e^T(N)\right]^T
\end{aligned}
\tag{2}
$$

and where the system matrices are given by $A(t) \in \mathbb{R}^{n\times n}$, $B(t) \in \mathbb{R}^{n\times m}$ and $M(t) \in \mathbb{R}^{p\times n}$. Furthermore, for $t \in \mathbb{Z}$

$$
x(t) = \begin{bmatrix} x_c^T(t) & x_b^T(t) \end{bmatrix}^T, \ x_c(t) \in \mathbb{R}^{n_c}, \ x_b(t) \in \{0,1\}^{n_b}, \ n = n_c + n_b \tag{3}
$$

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

$$
e(t) = \begin{bmatrix} e_c^T(t) & e_b^T(t) \end{bmatrix}^T, \ e_c(t) \in \mathbb{R}^{p_c}, \ e_b(t) \in \{0,1\}^{p_b}, \ p = p_c + p_b \tag{4}
$$

The control input is partitioned according to

$$
u(t) = \begin{bmatrix} u_c^T(t) & u_b^T(t) \end{bmatrix}^T, \ u_c(t) \in \mathbb{R}^{m_c}, \ u_b(t) \in \{0,1\}^{m_b}, \ m = m_c + m_b \tag{5}
$$

where $u_c(t)$ denotes the continuous inputs and $u_b(t)$ the logical inputs. Moreover, $H_x(t) \in \mathbb{R}^{c(t)\times n}$, $H_u(t) \in \mathbb{R}^{c(t)\times m}$ 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 1. $Q_e(t) \in \mathbb{S}^p_{++}, \ t = 0,\dots,N$

Assumption 2. $Q_u(t) \in \mathbb{S}^m_{++}, \ t = 0,\dots,N-1$

The objective with this work is to design an optimization algorithm for control of systems in the MLD form presented in [6]. Except for notational differences, there are four differences between the optimal control problem for MLD systems presented in [6] and the optimization problem in (1). First, the problem in [6] has reference signals on the control signals, states and outputs. Second, the MLD system has a direct term in $e(t)$ from the input $u(t)$. Third, the control problem in [6] has an equality constraint on the final state. Fourth, the weight matrix for the “auxiliary variables” in the control problem defined in [6] is positive semidefinite. In this work, it is assumed positive definite for all variables. The results presented in this work can be generalized to the case when the three first differences have been eliminated. However, so far, the fourth difference has not been possible to overcome.

Remark 1. Note that (1) becomes a linear MPC problem if $n_b = p_b = m_b = 0$.

2 Mixed Integer Quadratic Programming

In a QP problem, the optimization variables are real-valued, but in an MIQP problem, some variables are restricted to be integer-valued. In this text, the common special case of MIQP when the integer variables are constrained to be 0 or 1 is studied. A survey of Quadratic Integer Programming (QIP) can be found in [39].

The mathematical definition of an MIQP problem is

$$\begin{aligned} \underset{x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_b}}{\text{minimize}} \quad & \frac{1}{2} x^T H x + f^T x \\ \text{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}}, \quad A_{\mathcal{I}} x \le b_{\mathcal{I}} \end{aligned} \tag{6}$$

where $n = n_c + n_b$, $x \in \mathbb{R}^n$, $H \in \mathbb{S}_+^n$, $f \in \mathbb{R}^n$, $A \in \mathbb{R}^{p+m \times n}$ and $b_{\mathcal{I}} \in \mathbb{R}^{p+m}$. Moreover, $\mathcal{E} \in \mathbb{Z}_{++}^p$ and $\mathcal{I} \in \mathbb{Z}_{++}^m$ denote sets of indices to rows representing equality constraints and inequality constraints, respectively, in A and b.

The four most commonly used methods for solving MIQP problems are, [6], cutting plane methods, decomposition methods, logic-based methods and branch and bound methods. According to [16], branch and bound is the best method for mixed integer quadratic programs. An important explanation to why branch and bound is so fast for MIQP problems is that the QP subproblems are very cheap to solve, [16].

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.

2.1 Branch and Bound

In worst case, the solution process of an MIQP problem requires explicit enumeration of 2^{n_b} combinations of binary variables. The branch and bound method is a method that can significantly cut down the number of combinations necessary to investigate. Unfortunately, the worst case complexity is still exponential and the actual computational burden is problem dependent. The key idea to reduce the computational effort needed is to, in a computationally cheap way, 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 anymore since the optimal solution will not be found in any of them. A thorough derivation of and motivation for the branch and bound method can be found in [40] and in [17], where also a formal algorithm description is presented.

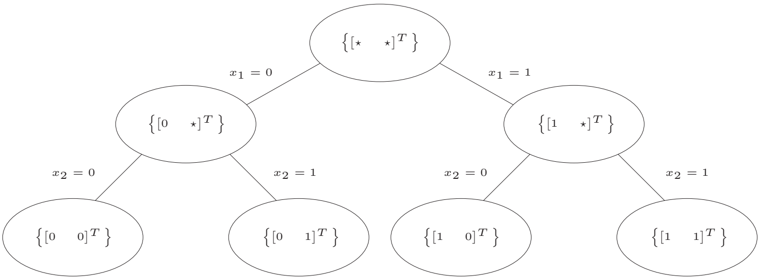


Figure 1: *This figure shows an example of a binary search tree for two binary variables, x_1 and x_2. In each node, represented as an ellipse, the corresponding feasible set $\mathcal{S}_i$ is shown. The symbol $\star$ is used to denote that this variable is free to be either 0 or 1.*

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

$$\mathcal{S} = \bigcup_{i=1}^{K} \mathcal{S}_i \tag{7}$$

The partitioning can be represented using a tree structure and is at first coarse, but is in later steps more and more refined. An example of a tree is given in Figure 1. The tree in Figure 1 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.

For what follows, let P_i denote the optimization subproblem over the set $\mathcal{S}_i$, N_i the node containing P_i, z^* the optimal objective function value over $\mathcal{S}$, and z^{i*} the optimal objective function value for subproblem P_i.

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

$$\begin{aligned} z^{i*} &= \underset{x \in \mathcal{S}_i}{\text{minimize}}\; f_0(x),\; i \in \{1, \ldots, K\} \\ z^* &= \min_{i \in \{1,\ldots,K\}} \left\{ z^{i*} \right\} \end{aligned} \tag{8}$$

where $f_0(x)$ is the objective function in the original integer problem. The optimal solution over $\mathcal{S}$ is found as the optimal solution to the subproblem with the lowest optimal objective function value. The idea in branch and bound is to use bounds on z^{i*} in order to reduce the number of partitions that have to be explicitly explored. In an MIQP solver built on branch and bound, QP relaxations of the original integer problem provide lower bounds and feasible integer suboptimal solutions provide upper bounds. QP relaxations are found by relaxing integer constraints into bound constraints. That is, if the binary variable indexed by j is relaxed, the constraint

$$x_j \in \{0, 1\} \tag{9}$$

is replaced by

$$x_j \in [0, 1] \tag{10}$$

In this report, 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 $\mathcal{S}_i^R$.

According to [16], 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 primal 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. Also, since a dual active set method is an ascent method generating dual feasible points, it can use an upper bound as a cut-off value for terminating the QP solver prematurely, [16]. According to [40], 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.

A branch and bound algorithm description can be found in, *e.g.*, [40] and [17].

3 Quadratic Programming

In this section, the QP problem is introduced and some basic properties are discussed. Furthermore, the gradient projection algorithm used in this work is presented. For an extensive bibliography on QP, see [22]. The optimization notation used in this report is chosen similar to the one in [10]. For what follows, a Quadratic Programming (QP) problem with n variables, p equality constraints and m inequality constraints in the following form will be of importance

$$\begin{aligned}
\underset{x_1,x_2}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}x_1^T & x_2^T\end{bmatrix}\underbrace{\begin{bmatrix}\tilde{H} & 0\\ 0 & 0\end{bmatrix}}_{H}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x}+\begin{bmatrix}\tilde{f}^T & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\\
\text{subject to} \quad & \underbrace{\begin{bmatrix}A_{1\mathcal{E}} & A_{2\mathcal{E}}\end{bmatrix}}_{A_{\mathcal{E}}}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=b_{\mathcal{E}},\quad \underbrace{\begin{bmatrix}A_{1\mathcal{I}} & A_{2\mathcal{I}}\end{bmatrix}}_{A_{\mathcal{I}}}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\leq b_{\mathcal{I}}
\end{aligned}\tag{11}$$

where $n = n_1 + n_2$, $x_1 \in \mathbb{R}^{n_1}$, $x_2 \in \mathbb{R}^{n_2}$, $\tilde{H} \in \mathbb{S}_{++}^{n_1}$, $\tilde{f} \in \mathbb{R}^{n_1}$, $A_1 \in \mathbb{R}^{p+m\times n_1}$, $A_2 \in \mathbb{R}^{p+m\times n_2}$ and $b \in \mathbb{R}^{p+m}$. Furthermore, $\mathcal{E} \in \mathbb{Z}_{++}^{p}$ and $\mathcal{I} \in \mathbb{Z}_{++}^{m}$ denote sets of indices to rows representing equality constraints and inequality constraints respectively in $A \in \mathbb{R}^{p+m\times n}$ and $b \in \mathbb{R}^{p+m}$. The dual problem to the problem in (11) can be found as described in Appendix A.2. By reformulating the resulting dual maximization problem in (74) as a minimization problem and by removing a constant in the objective function, the result is a new equivalent QP problem in the form

$$\begin{aligned}
\underset{\lambda,\nu}{\text{minimize}} \quad & \frac{1}{2}Q_D(\lambda,\nu)\\
\text{subject to} \quad & A_{2\mathcal{I}}^T\lambda + A_{2\mathcal{E}}^T\nu = 0,\quad \lambda \geq 0
\end{aligned}\tag{12}$$

where $\lambda \in \mathbb{R}^m$, $\nu \in \mathbb{R}^p$ and

$$\begin{aligned} Q_D(\lambda, \nu) = \begin{bmatrix} \lambda^T & \nu^T \end{bmatrix} \begin{bmatrix} A_{1\mathcal{I}} \\ A_{1\mathcal{E}} \end{bmatrix} \tilde{H}^{-1} \begin{bmatrix} A_{1\mathcal{I}}^T & A_{1\mathcal{E}}^T \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} + \\ + 2 \left(\tilde{f}^T \tilde{H}^{-1} \begin{bmatrix} A_{1\mathcal{I}}^T & A_{1\mathcal{E}}^T \end{bmatrix} + \begin{bmatrix} b_{\mathcal{I}}^T & b_{\mathcal{E}}^T \end{bmatrix} \right) \begin{bmatrix} \lambda \\ \nu \end{bmatrix} \end{aligned} \tag{13}$$

By Theorem 3, strong duality holds for the primal problem in (11) and the dual problem in (74). Note that the solution to the equivalent problems in (12) and in (74) are equal, but that the optimal objective function values do not coincide since the sign of the objective function has been changed and a constant has been removed.

The idea in this work is to solve a primal QP problem in the form in (11) by solving a dual problem in the form in (12) and then compute the primal optimal solution from the dual optimal solution. According to Theorem 3, the primal problem has a feasible solution if and only if the solution to the dual problem is bounded. Given such a bounded dual optimal solution λ^* and ν^*, a primal optimal solution x^* can be found from the necessary and sufficient (by Theorem 2) KKT conditions for the primal problem.

Note that, there is only one difference between how primal equality constraints and primal inequality constraints appear in the dual QP problem. The difference is that the dual variables corresponding to primal inequality constraints get a lower bound inequality constraint (*cf.*, $\lambda \geq 0$) in the dual, while the dual variables corresponding to primal equality constraints are not subject to any inequality constraints in the dual. Note especially, if the dual inequality constraints are removed, the inequality constraints in the corresponding primal QP problem has been "converted" into equality constraints. This observation turns out to be useful later on in this report.

3.1 A Short Introduction to Dual Active Set QP Methods

A quick verbal description of the algorithm presented in this work is that it can be interpreted as an active set method, where large changes of the active set is possible in each iteration. Hence, most properties of such methods, like the possibility of efficient warm start also hold for this improved algorithm. For an introduction of active set methods, see, *e.g.*, [31]. A more detailed description of the algorithm presented in this work will be given in Section 3.3.

A primal feasible active set QP method starts in a primal feasible point and primal feasibility is thereafter maintained in all subsequent QP iterations. One important drawback with such a method is that it can be hard to find a primal feasible starting point for a general QP. If a feasible starting point cannot be obtained using, for example, practical knowledge of the problem, a so-called Phase I algorithm can be applied to find such a point. According to [19], 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. A great structural advantage with the dual problem in (12), compared to the primal problem in (11), is that the latter only has constraints with *linear* (*not* affine) constraint functions.

Briefly, a dual active set method solves an inequality constrained optimization problem of the type in (12) by solving a sequence of equality constrained problems in the form

$$\begin{array}{ll}\underset{\lambda,\nu}{\text{minimize}} & Q_D(\lambda,\nu) \\ \text{subject to} & A_{2\mathcal{I}}^T\lambda + A_{2\mathcal{E}}^T\nu = 0, \quad [I_{n_a}\ 0]\,\lambda = 0\end{array} \tag{14}$$

where it has been assumed that n_a inequality constraints are chosen active and, without any loss of generality, that the components in λ have been ordered such that the components that are constrained to zero are placed first in λ. The set of indices to the original equality constraints in the problem, plus the n_a inequality constraints that currently have been converted into equality constraints are referred to as the working set and is in iteration k denoted $\mathcal{W}_k$. The word active set is used to denote the set of indices to the constraints that hold with equality in a point and is denoted $\mathcal{A}(x)$ for the point x. Basically, an active set method tries to find the optimal active set by solving several problems with different choices of working sets. Once the optimal working set has been found, the KKT conditions for the original problem are satisfied by the solution to the subproblem, and the algorithm terminates.

A consequence of the simple constraint structure in the dual QP problem is that the origin is always a feasible solution (the trivial solution will always satisfy the constraints, *cf.*, the trivial solution always satisfies a homogeneous linear system of equations). Also note that this is true independently of the working set chosen, *i.e.*, the working set cannot be chosen in a way such that a QP subproblem becomes infeasible (using the same argument).

The aim in this work is to derive an algorithm that can solve the subproblems of QP type in a branch and bound algorithm efficiently. These subproblems are in the form given in (1) with $n_b = p_b = m_b = 0$, *i.e.*, Remark 1 applies, and where some of the inequality constraints may have been changed into similar equality constraints (the constraint function is unchanged, the only difference is that "$\leq$" has been replaced by "$=$") during the branch and bound process. From the previous discussion in this section, it can be concluded that because of its good warm start abilities, a dual active set QP solver would probably be the best choice for solving the node problems. However, ordinary active set solvers are rather inefficient since they often need many iterations to find the optimal active set. Therefore, the solver presented in this work combines ideas from dual active set methods with ideas from gradient projections methods.

3.2 Solving the Relaxations Using a Dual QP Method

It will now be described how the QP relaxations in a branch and bound algorithm for MIQP can be solved using a dual active set QP solver. In the discussion that follows, the notation introduced in Section 2.1 is used. Consider a branching procedure for an arbitrary node N_k. When the node is branched on variable j, two new nodes N_{k0} and N_{k1} are created with feasible sets

$$\mathcal{S}_{k0} = \mathcal{S}_k \cap B_{k0},\ \mathcal{S}_{k1} = \mathcal{S}_k \cap B_{k1} \tag{15}$$

where

$$B_{k0} = \{x|x_j = 0\}\,,\ B_{k1} = \{x|x_j = 1\} \tag{16}$$

In the branch and bound method used in this work, the lower bounds computed by the QP relaxations in the nodes are found by relaxing some of the binary constraints to interval constraints. Let $\underline{x}^{k*}$ denote the optimizer to the relaxed problem P_k^R. 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} are not relaxed. To understand what happens when the equality constraints after a branch are introduced, three different relaxed problems are considered.

$$\begin{aligned} &\underline{x}^{k*} = \underset{x \in \mathcal{S}_k^R}{\operatorname{argmin}}\, f_0(x) \\ &\underline{x}^{k0*} = \underset{x \in \mathcal{S}_{k0}^R}{\operatorname{argmin}}\, f_0(x),\ \underline{x}^{k1*} = \underset{x \in \mathcal{S}_{k1}^R}{\operatorname{argmin}}\, f_0(x) \end{aligned} \tag{17}$$

Since the constraint differing $\mathcal{S}_{k0}^R$ from $\mathcal{S}_{k1}^R$ is not relaxed, $\mathcal{S}_{k0}^R \cap \mathcal{S}_{k1}^R = \emptyset$. 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, *i.e.*, $\underline{x}_j^{k*} \in \{0, 1\}$, would be feasible in one of the child nodes. 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.

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 in (11) and the dual one in (74). The subproblems to be solved in the nodes of the branch and bound tree will be of the type in (74) (or the equivalent one in (76)). Note that, when a new constraint is added to the primal problem, the dimension of the dual problem increases. A feasible solution to the new problem is readily available by keeping the old solution and, for example, choosing the new dual variable equal to zero. If the added constraint is an inequality constraint, the new dual variable has to be chosen non-negative.

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 due to the relaxation, an alternative interpretation is that, *e.g.*, 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. Hence, if the primal problem is constrained, the dual problem is relaxed. It should be stressed, that in most implementations, the locked variable is eliminated from the problem, which implies that the problems will become smaller and smaller further down in the tree since one more variable is locked for each level down in the tree. When the locked variable is removed from the problem, the corresponding constraint can also be removed, and hence, the dimension of the dual problem is decreased.

To summarize, using a dual solver for the relaxed problems in the nodes will make warm starts more easy. Given the old optimal working set, hopefully only a few QP iterations have to be performed until the optimal working set of the new problem is found. However, this is problem dependent and the dual algorithm presented in this report will not change this fact.

3.3 Gradient Projection for QP

In this section, the gradient projection algorithm used in this work is presented. It is presented for a general problem, and the efficient computations that utilizes problem structure will be presented later in the report.

Introduction

A drawback with a classical active set method is that the working set is changing very slowly. For each change in the working set, a system of equations for a Newton step has to be solved. If the initial working set is far from the optimal active set, it will take a lot of effort to reach this set. The idea in a gradient projection method is to allow a more rapid change of the working set, which in turn implies that often less Newton systems have to be solved before the optimal active set is found. However, when this method is applied to a QP problem with general inequality constraints, the projection operation performed in each iteration can become very computationally expensive. An important exception is when the inequality constraints only consist of upper and lower bounds on variables. An important problem that have constraints of this type is the dual QP problem, [31].

The fundamental ideas behind the gradient projection algorithm presented in this report was first presented, independently, in [20] and in [26]. Gradient projection methods have properties that make them suitable for problems with many active constraints in the optimum. This property is especially important in optimal control applications where often many control inputs are at their boundaries at the optimum, [7]. Furthermore, the Newton step in [7] is computed by solving a Riccati equation. The first gradient projection ideas are refined in [28], where the step length is found by a line search instead of using a fixed length. In [7], the gradient projection method is combined with a Newton method in order to improve the convergence rate. It is also proposed that the Newton direction can be projected in a similar way as the steepest descent direction. However, as discussed in [8], some care must be taken when using such a direction. A combination of a standard active set method and gradient projection method is presented in [30]. In [29], the gradient projection method is combined with a conjugated gradient method. The presented algorithm is similar to the one presented in this, except for that the conjugated gradient part of the algorithm are replaced by a Newton step found using a Riccati recursion. Convergence properties of gradient projection methods are discussed in, *e.g.*, [11], [12], and in [18].

In principle, a gradient projection method can be applied to a general nonlinear optimization problem (that satisfies certain conditions). In this work, the discussion is limited to the QP case with simple lower bound constraints on some of the variables. To minimize the notational complexity, a generic QP problem in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q(x) = \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & x \geq 0 \end{aligned} \tag{18}$$

is considered, where $H \in \mathbb{S}_+^n$ and $f \in \mathbb{R}^n$. In the algorithm presented in this work, analogous ideas are applied to a problem in the form in (12). Note that, linear equality constraints can always be eliminated and an equivalent unconstrained optimization problem can be formulated. For more details, see [31]. In the QP case, this equivalent problem is another QP. The Hessian of this new QP is called the reduced Hessian.

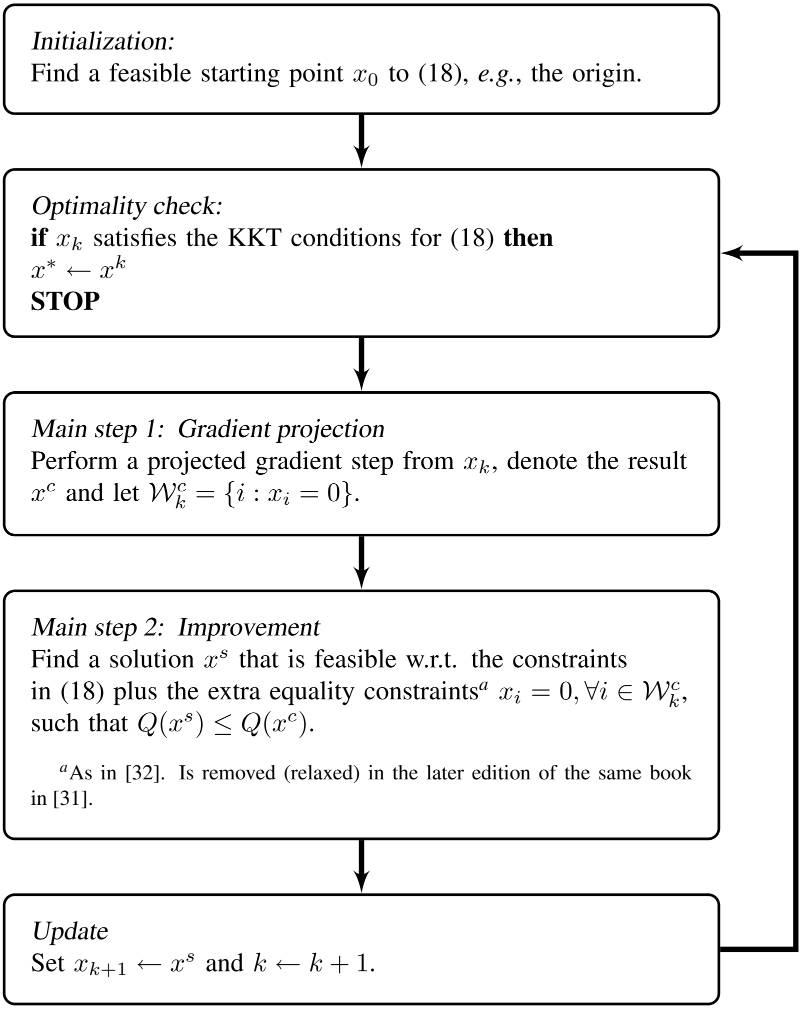
Each iteration of the method used in this work can be considered to consist of two steps. The procedure is outlined in Figure 2, when it is applied to a problem in the form in (18). The description here is based on the one in [31] and has been adopted to the dual QP problem.

The method described in [31] is a variant of the one presented in [12], which is a method built on trust region ideas. However, when it is used to solve a convex QP problem, the trust-region part of the algorithm is not used since the "model" is exact. It also performs, in contrary to many gradient projection algorithms, an *exact* line search in the quadratic model during the gradient projection step. In the first step ("Main step 1" in Figure 2), the gradient is computed at the current point. After the gradient has been computed, a line search optimization along the negative gradient (*i.e.*, steepest descent) direction is performed. If an inequality constraint is encountered before a minimizer is found along the line, the search direction is bent-off such that the constraint remains satisfied. Hence, the search line is projected onto the constraints and the search is continued until a local optimal solution is found along the piecewise linear path, the search is stopped by constraints in sufficiently many directions to make it impossible to continue anymore in any of the initial directions, or the step size tends to infinity without any constraints in its way. In the latter case, an eigenvector corresponding to a zero eigenvalue has been found and the problem is unbounded. In the second step ("Main step 2" in Figure 2), a smaller optimization problem is defined from the original problem but where all constraints that were activated during the gradient projection step are *locked.* The smaller optimization problem can be solved in various ways. As will be discussed later, the point found in the first step is good enough to obtain global convergence, [31], and the purpose with "Main step 2" is to improve the convergence rate. A common choice is to run a conjugated gradient method on the subspace, [29, 31], and either stop the search when a constraint is hit, or to temporarily ignore the constraints and project the iterates back onto the feasible set, [31]. In this work, an alternative approach has been used. During the second step, the Newton step is computed for the subproblem and the resulting *direction* is projected onto the constraints. Hence, this step in the algorithm can be interpreted as a projected Newton step, which previously has been discussed in, *e.g.*, [7]. "Main step 2" will be thoroughly described later in this section.

If the gradient projection algorithm is applied to a problem for which strict complementarity holds, *i.e.*, the Lagrangian multipliers associated with all active constraints in optimum are non-zero, the algorithm will eventually produce active sets $\mathcal{A}(x^c)$ that are equal to the optimal active set for all iterations k that are sufficiently large, [31]. This means that constraint indices will not enter and leave the working set repeatedly on successive iterations. However, to prevent this from happening, different solutions have been proposed, [31].

Projected Line Search

An important part of the gradient projection algorithm presented in this report is the projected line search operation. During this operation, the objective function is minimized along a piecewise linear path, which is the result of a search started in a certain point in



Initialization:
Find a feasible starting point x_0 to (18), *e.g.*, the origin.

Optimality check:
if x_k satisfies the KKT conditions for (18) **then**
$x^* \leftarrow x^k$
STOP

Main step 1: Gradient projection
Perform a projected gradient step from x_k, denote the result x^c and let $\mathcal{W}_k^c = \{i : x_i = 0\}$.

Main step 2: Improvement
Find a solution x^s that is feasible w.r.t. the constraints in (18) plus the extra equality constraints[a] $x_i = 0, \forall i \in \mathcal{W}_k^c$, such that $Q(x^s) \leq Q(x^c)$.

[a] As in [32]. Is removed (relaxed) in the later edition of the same book in [31].

Update
Set $x_{k+1} \leftarrow x^s$ and $k \leftarrow k + 1$.

Figure 2: *An overview of a gradient projection algorithm. The figure is inspired by the algorithm presented in [31, 32].*

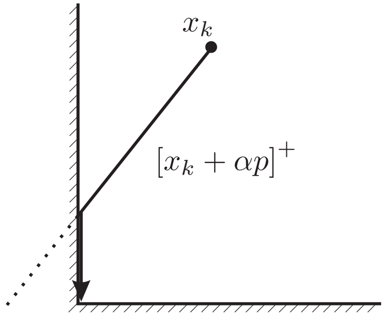


***Figure 3:** The figure illustrates how the points along the line starting in the point x_k in the direction p are projected back onto the positive orthant during the operation $[x_k + \alpha p]^+$. Another interpretation is that the* search direction *(cf., gradient projection) is projected onto the positive orthant. The resulting path is piecewise linear.*

a given search direction, and where the search direction is bent-off during the search in order to maintain feasibility. This procedure is illustrated in Figure 3. In the algorithm presented in this report, the operation is used for different kinds of search directions, *i.e.*, the steepest descent direction, a Newton direction, and a projection of the steepest descent direction onto the nullspace of the Hessian. Note that, when the search line is bent-off, the optimization problem to find the minimizer along this piecewise linear path is, in general, no longer a convex optimization problem, [8]. However, in gradient projection methods, the global minimizer along this piecewise linear path is not used. Instead, either a step size given by a modified version of the Armijo condition (see [31]), or the first *local* minimizer found using exact line search, is used. In this work, the latter is chosen in accordance with [31]. The point found during the line search is called the Cauchy point, and its purpose here is analogous to the one in unconstrained optimization, where it is the minimizer of the objective function along the steepest descent direction subject to a trust-region constraint, [31]. Basically, an algorithm that produces steps at least as good as the one to the Cauchy point will be globally convergent. For further details, see [31].

The search along the piecewise linear path to find the Cauchy point is now described in more detail. This part of the algorithm is standard, and hence, the presentation here is basically a summary of the one in [31]. See the cited reference for details. For an optimization problem with positivity constraints on the variables, like the one in (18), the piecewise linear path from the point x_0 in the direction p is defined by

$$x(s) = [x_0 + sp]^+ \tag{19}$$

where the ith component in the function $[x]^+$ is defined as

$$[x]_i^+ = \begin{cases} x_i, \; x_i \geq 0 \\ 0, \; x_i < 0 \end{cases} \tag{20}$$

i.e., $[\cdot]^+$ denotes projection onto the positive orthant, and where $s \geq 0$ is the scalar that parameterizes the path.

In order to find the Cauchy point, the aim is now to find the first local minimizer s^* of $Q([x(s)]^+)$ for $s \geq 0$. Since $Q([x(s)]^+)$ is *piecewise* quadratic in s, it is not necessarily continuously differentiable in the breakpoints between the different quadratic sections. Therefore, it is hard to give an expression for x^c in closed form. In order to find the first local minimizer, the breakpoints are computed, and then each quadratic section is considered separately until a minimizer is found either at a breakpoint or in the interior of a quadratic section. The breakpoints can be found explicitly from the expression, [31],

$$\bar{s}_i = \begin{cases} \frac{x_i}{p_i}, \ p_i < 0 \\ \infty, \ \text{otherwise} \end{cases} \tag{21}$$

Hence, the components of the parameterized path can be expressed as

$$x_i(s) = \begin{cases} x_i + sp_i, \ s \leq \bar{s}_i \\ 0, \ \text{otherwise} \end{cases} \tag{22}$$

Once the breakpoints have been computed, they have to be sorted, and zero values and duplicates have to be removed. The latter occur if more than one constraint is reached for the same value of s. Hence, the unsorted set of original breakpoints $\{\bar{s}_1, \ldots, \bar{s}_n\}$ have been sorted and reduced into a new set consisting of l sorted breakpoints $\{s_1, \ldots, s_l\}$, where $0 < s_1 < s_2 < \ldots < s_l$. On each interval $[0, s_1]$, $[s_1, s_2]$, $[s_2, s_3]$,. . .,$[s_{l-1}, s_l]$, the objective function is quadratic and can be optimized analytically if the upper and lower bounds on s are temporarily disregarded. After such an unconstrained minimizer has been found to one of these quadratic subproblems, it has to belong to the current interval in s to be a valid local minimizer. Otherwise, the minimizer of the piecewise quadratic function is found either at one of the boundaries of the interval or in another segment. On each interval, $x(s)$ can be written as

$$x(s) = x(s_{j-1}) + \Delta s \hat{p}^{j-1} \tag{23}$$

where $0 \leq \Delta s \leq s_j - s_{j-1}$ and

$$\hat{p}_i^{j-1} = \begin{cases} p_i, \ s_{j-1} < \bar{s}_i \\ 0, \text{otherwise} \end{cases} \tag{24}$$

On each interval, (23) can be inserted into the objective function, and the result is a new one-dimensional QP problem in the variable s which can be solved analytically. In Section 5.1, a tailored version of this operation is presented. The general case is thoroughly considered in, *e.g.*, [31].

Main Step 2 in Detail

In "Main step 1", a standard gradient projection step is performed as described in [31]. Even though several computations in that part of the algorithm have been tailored in this work, the main idea is standard. Hence, the focus will now be on "Main step 2", which will be studied in detail. In "Main step 2", several iterations where a Newton direction is projected onto the feasible set is performed. For what follows, these iterations are called inner iterations. The procedure to perform one loop in Figure 2 is here referred to as an outer iteration.

In this section, the problem in (18) is used to represent the problem in (12). The problems are not identical, but the presented ideas can be applied in an analogous way to the problem in (12).

The motivation for the existence of “Main step 2” is that an algorithm that always takes a step to the next Cauchy point basically implements a steepest descent algorithm, which unfortunately converges very slowly. However, once the Cauchy point has been found, it is possible to improve the convergence rate by employing a method with better convergence properties, and this is what is performed in “Main step 2”. During this part of the algorithm, subproblems in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q_s(x) = \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & x_i = x_i^c,\ i \in \mathcal{A}(x^c) \\ & x_i \geq 0,\ i \notin \mathcal{A}(x^c) \end{aligned} \tag{25}$$

are solved approximately, where $\mathcal{A}(x^c)$ denotes the active set in the last Cauchy point. Even though it is in principle possible to solve it exactly since the subproblems also are QPs, it is not desirable because it might be almost as difficult to solve as the original problem in (18). To obtain global convergence of the algorithm outlined in Figure 2, it is only necessary that the approximate solution is feasible with respect to the constraints of the original problem in (18) and that the objective function value of the approximate solution x^+ is not worse than the already found Cauchy point x^c, [31]. However, the algorithm presented in this work, satisfies the slightly harder constraint, taken from the *first* edition [32] of the book [31], that x^+ should be feasible in the problem in (25) instead of in the problem in (18). The approach used in this report is a variant of the one that uses the conjugated gradient method in “Main step 2” as proposed in [31]. In the approach chosen here, the search direction is taken as the vector from the current point toward the minimizer of a problem in the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q_s(x) = \frac{1}{2}x^T H x + f^T x \\ \text{subject to} \quad & x_i = x_i^c,\ i \in \mathcal{A}(x^c) \end{aligned} \tag{26}$$

In this problem, the inequality constraints in (25) have been disregarded. The motivation to this choice is that, if there were no inequality constraints in the problem, this search direction would be the Newton step on the subspace defined by the equality constraints. Therefore, this step is sometimes called the projected Newton step in this report. Here, this problem is solved directly using Riccati recursions. This is explained in detail in Section 5.3. A special case is when the objective function is unbounded. That case is handled in a different way, which is described in detail in Section 3.3. In both cases, the resulting search direction is used in a projected line search. Since the optimization problem in (26) is a QP problem (the equality constraints can in principle be eliminated and the resulting problem is unconstrained), a single Newton step is enough to find the minimizer. Furthermore, this Newton step, without projection, is used as the search direction in an ordinary active set solver. When an ordinary active set solver searches in this direction and encounters a constraint, the constraint is added to the working set and a new problem in the

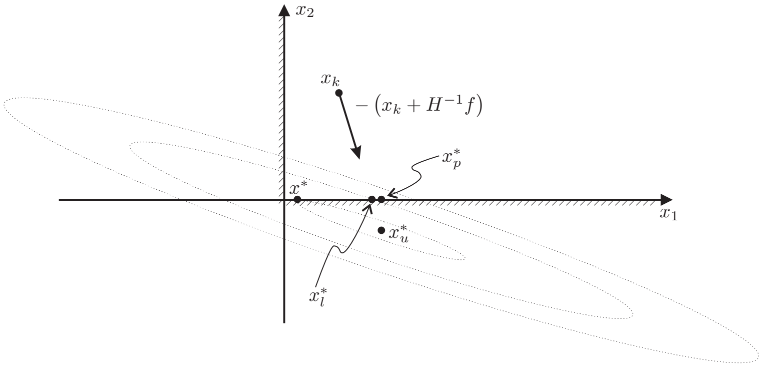
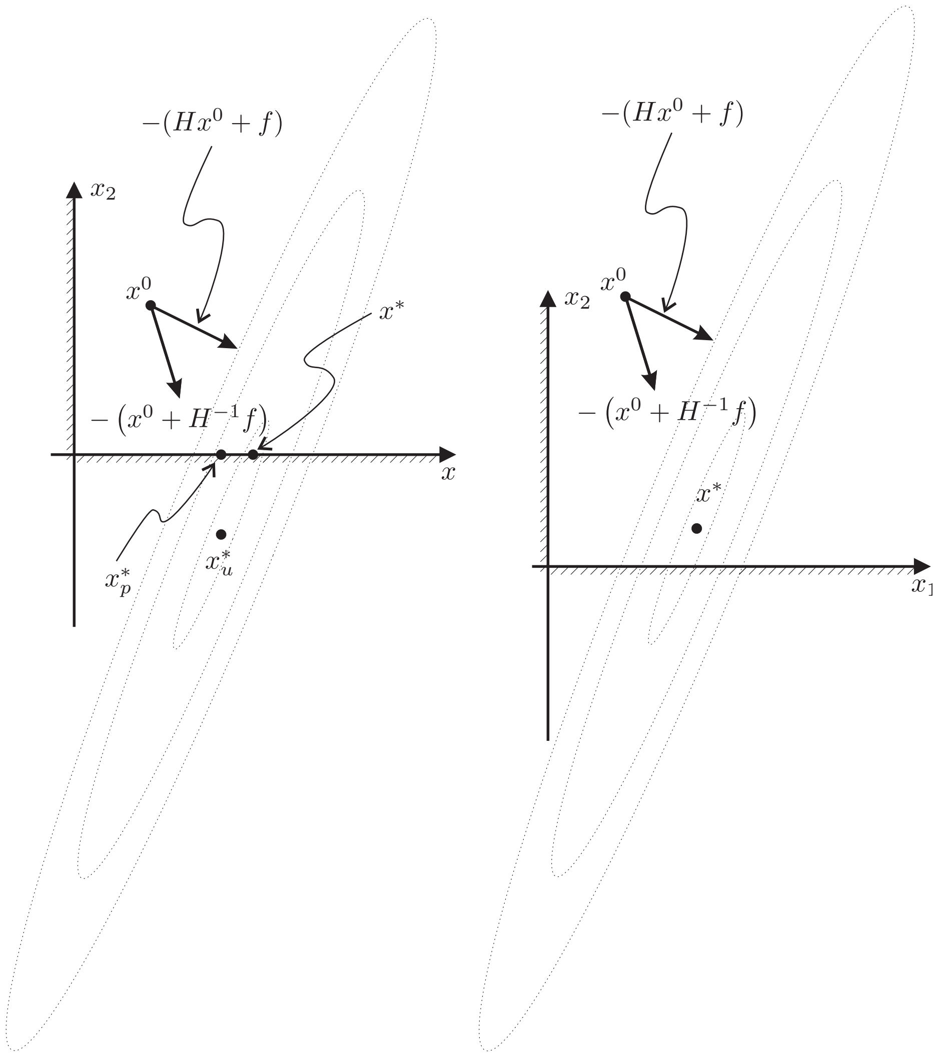


Figure 4: The figure illustrates that it might not be straightforward to utilize the result from a projection of the unconstrained minimizer of a quadratic objective function onto the feasible set in order to solve a constrained version of the problem. The dotted ellipses illustrate the level curves of the objective function $\frac{1}{2}x^T Hx + f^T x$, and the feasible set is the positive orthant. The unconstrained minimizer is denoted x_u^*. The projection of the unconstrained minimizer onto the feasible set is denoted x_p^*. If a line search is performed along the projection of the unconstrained search direction, that is, the piecewise linear path from x_k to x_l^* to x_p^*, the minimizer x_l^* is found. The true minimizer to the constrained problem is denoted x^*.

form in (26) is solved. The enhancement made to this strategy in this report, is that an attempt is made to make further progress by bending the search direction along the constraint boundary and continue along that path until a local minimizer is found. The idea is to use the information in the search direction as much as possible. However, note that, it is not in general true that the projection of the Newton step from (26) onto the feasible set of the problem in (25) will lead to the true minimizer of (25) in one iteration. This is illustrated in Figure 4. Furthermore, if several steps of this type are performed, without taking this property into account, the algorithm might get stuck in a suboptimal point. This undesirable behavior is also shown in Figure 4, where an algorithm using projected Newton steps would get stuck in the point x_l^*. This potential problem is discussed in [8], where also a remedy is presented. In this report, this property of the projected Newton search direction is handled in several ways. First, a line search is performed along the path and the final step chosen is to the first local minimizer along this path. Hence, the objective function value cannot be made worse during this part of the algorithm and the feasibility is ensured by the projection. This makes it possible to gain from the strategy in situations as the one in Figure 5a, and still have a reasonable behavior in situations as the one in Figure 4. Second, as described above, iterations where this step is used are *alternated* with gradient projection iterations (projection of the steepest descent direction). The steepest descent direction does not suffer from this problem and ensures convergence (under certain assumptions) as described in [31]. Basically, when the optimal active set has been identified, the solution to the original QP problem is found by solving *one* problem in the form in (26). This case is illustrated in Figure 5b. If only gradient projection steps were used, in general, many iterations had been necessary even though it is only



(a) *An example of a successful Newton step projection. The true minimizer is found after only one Newton step computation.*

(b) *Another example of a successful Newton step projection. This is an extra important case since this is basically the case when the optimal working set has been identified. Also in this case, the true minimizer is found after only one Newton step computation.*

Figure 5: *Two examples that motivate the use of the projection of the direction toward the unconstrained minimizer of the quadratic objective function. The notation used is the same as the one in Figure 4. Note that the gradient projection method would have converged rather slowly in these cases because the eigenvalues of the Hessian of the quadratic objective function are not equal (the level curves are not circles) and a zigzagging behavior would have occurred, [31].*

a QP with inequality constraints. Note that, the convergence properties of the algorithm presented in this report follow from the properties of the gradient projection iteration as described in [8]. Hence, for the *convergence* of the algorithm, there is actually no need to find the true minimizer during the projected Newton step iterations. The projected Newton steps are mainly used for performance reasons. Third, an active set strategy is used if *several* projected Newton steps are performed without any gradient projection step in between. More specifically, in the beginning of "Main step 2", the working set is initialized to $\mathcal{W}_k^c$, and every constraint encountered during consecutive inner iteration are accumulated to this set. That is, no constraints will be dropped from the working set during inner iterations. The inner iterations are aborted when either the maximum number of inner iterations is reached, or when no new constraints were encountered during the last iteration. Constraints are dropped after "Main step 2" has terminated and before "Main step 1" has started, since "Main step 1" starts with an empty working set. The difference compared to an ordinary active set method is that the update of the working set is performed first when a suboptimal solution along the piecewise linear path has been found, instead of as soon as the first constraint has been encountered. Hence, in contrast to an ordinary active set method, several constraints can be added after *one* Newton step computation. Illustrated to the problem in Figure 4 this means that after a full projected Newton step from x_k, the point x_p^* would have been found. However, since a line search is used along the piecewise linear path, the minimizer x_l^* is found. Note that, if a new projected Newton step is performed directly, without an update of the working set, no progress from this point will be made. To be able to make progress, the working set is updated and, in this example, the constraint $x_2 = 0$ is added to the working set, and a new projected Newton step subject to the new working set is performed. In the example in Figure 4, the second projected Newton step iteration will be in the negative direction of the x_1-axis. Hence, the true minimizer x^* is found in the second iteration.

Gradient Projected onto the Nullspace of the Hessian

In this section, the case when the problem in (26) is unbounded is discussed. The case is illustrated in a two-dimensional example in Figure 6. When this occurs, there does not exist any point where the KKT conditions are satisfied (there is no stationary point). For simplicity, the equality constraints in (26) are now eliminated and an equivalent problem in the form

$$\underset{x}{\text{minimize}} \quad \tilde{Q}_s(x) = \frac{1}{2}x^T\tilde{H}x + \tilde{f}^Tx \tag{27}$$

will be considered. The KKT system for this reduced problem is

$$\tilde{H}x + \tilde{f} = 0 \tag{28}$$

If $\tilde{f} \notin \operatorname{range}\tilde{H}$ there is no solution to the system of equations in (28). In such a case, an alternative search direction has to be chosen. One possibility is to choose the steepest descent direction, but then the convergence rate will be rather slow. This is illustrated in Figure 6b, where the steepest descent direction $-g$ would lead to rather short steps since it points in a direction where the objective function is bounded. In this work, the search direction is chosen as the projection of the steepest descent direction onto the nullspace of the Hessian $\tilde{H}$. This choice is motivated in Appendix A.6 and the result can be written as

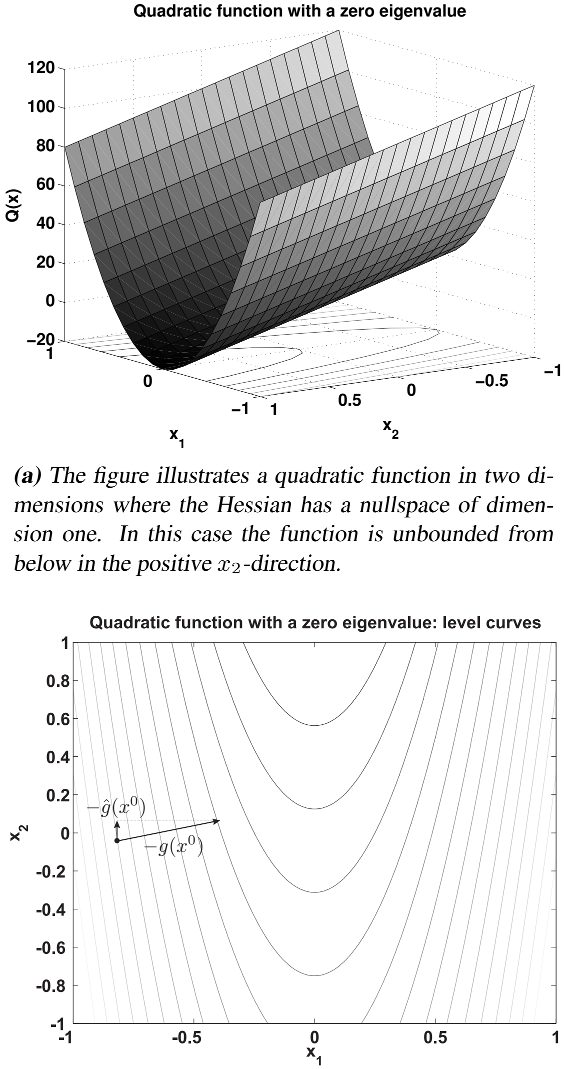
$$-\hat{g}(x^0) = -\frac{\mathcal{P}\left(\tilde{H}x^0 + f\right)}{\left\|\mathcal{P}\left(\tilde{H}x^0 + f\right)\right\|} \tag{29}$$

where $\mathcal{P} = Z\left(Z^T Z\right)^{-1} Z^T$ and the columns in Z form a basis for the nullspace of $\tilde{H}$. The interpretation of this situation is that the quadratic objective function has one or more unbounded rays along which the curvature is zero. Such unbounded directions are found in the nullspace of the Hessian of the quadratic objective function. Since the inequality constraints in the original problem in (18) are disregarded in (26), every descent direction in this nullspace results in an unbounded objective function value. After the search direction has been computed, it is used during the projection process and the search direction is, as usually, bent if any constraint is encountered along the search path.

3.4 Convergence and an Algorithm Description

In this section, the presentation of the algorithm will be concluded with a formal description of the algorithm. Furthermore, the convergence properties of the algorithm will be discussed. The formal algorithm is presented as Algorithm 1. Note that, a "projected step" includes a projected line search as previously described.

Now, finite termination of the algorithm will be discussed. Assume that strict complementarity holds and that "Main step 2" satisfies the conditions shown in Figure 2. Then, according to [31], the optimal active set will be identified in a finite number of iterations. It is straightforward to show that the conditions for "Main step 2" are satisfied. First, feasibility with respect to the inequality constraints in (18) are guaranteed by the projected line search used in "Main step 2". Second, feasibility with respect to the working set $\mathcal{W}_k^c$ computed in "Main step 1" follows from that none of those constraints can be dropped during "Main step 2". Third, since the projected line search operation is used, the objective function value cannot be made worse during "Main step 2". Hence, the conditions for "Main step 2" in Figure 2 are satisfied. Consequently, if strict complementarity holds, the *entire* algorithm ("Main step 1" and "Main step 2") will terminate in a finite number of iterations, if "Main step 2" terminates in a finite number of iterations. This will now be shown. First, assume that $m_{max} = \infty$ (see Algorithm 1). Finite termination of "Main step 2" follows basically by standard active set arguments. There is a limited number of constraints in the problem. By construction, either at least one constraint is added during each iteration of "Main step 2", or an optimal point has been found subject to the constraints indexed by the current working set. Furthermore, no constraints are dropped during "Main step 2", since they are accumulated between iterations. Note that, such a point will be found after a maximum of r iterations, where r is the number of constraints in the problem, and when that occurs $\mathcal{W}_m^N = \mathcal{W}_{m+1}^N$ (no constraints added during the last iteration). Hence, "Main step 2" will terminate after a maximum of r iterations. Now, consider the case when $m_{max} < \infty$. In this case, finite termination of "Main step 2" follows trivially. There is however, one important difference compared to the previous case. If "Main step 2" terminates because the maximum number of iterations has been reached, a point which is optimal subject to the current working has not necessarily been found. However, since the conditions for "Main step 2" in Figure 2 still are satisfied, the



(a) *The figure illustrates a quadratic function in two dimensions where the Hessian has a nullspace of dimension one. In this case the function is unbounded from below in the positive x_2-direction.*

(b) *The figure illustrates that the steepest descent direction $-g(x^0)$ does in general not point in a direction where the quadratic objective function is unbounded. The vector $-\hat{g}(x^0)$ is the steepest descent direction projected onto the nullspace of the Hessian. If the gradient is not orthogonal to this nullspace, it will point in a direction where the objective function is unbounded from below.*

Figure 6: *The figure shows an example of the situation when the Hessian of a quadratic function has a zero eigenvalue. This means that the quadratic function behaves as a linear one in one or more directions.*

composed algorithm (“Main step 1” and “Main step 2”) will find the optimal active set in a finite number of iterations, and when that occurs, “Main step 2” will terminate in one iteration with an optimal solution to the original inequality constrained problem (the situation is the one shown in Figure 5b).

4 The Dual MPC Problem

The gradient projection algorithm previously presented in this work is used to solve the QP relaxations in the branch and bound tree. These are linear MPC problems almost in the form in (1) with $n_b = p_b = m_b = 0$, *i.e.*, with only real-valued variables. The difference compared to the problem in (1) is that some of the inequality constraints are converted into similar equality constraints. The dual of the MPC problem is presented in Section 4.1 and the relation to the primal variables is presented in Section 4.2. How branching affects the problem is discussed in Section 4.3.

4.1 Derivation of the Dual Problem

In order to design a solver working on the dual problem to a problem in the form in (1) with $n_b = p_b = m_b = 0$, the dual optimization problem is derived. Note that, it is the dual of a *relaxation* of the problem in (1) that is sought for, not the dual problem of the original non-convex optimization problem for which strong duality does not hold. This is the reason why the case when $n_b = p_b = m_b = 0$ is studied. The optimization problem in (1) is in the form in (11). Hence, the dual optimization problem is in the form in (12). After a suitable selection of dual variables, the dual problem to the problem in (1) can be written in the form

$$
\begin{aligned}
\underset{\tilde{\mathsf{x}},\tilde{\mathsf{u}}}{\text{minimize}} \quad & J_D(\tilde{\mathsf{x}},\tilde{\mathsf{u}}) = \frac{1}{2}\tilde{u}^T(-1)\tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) + \tilde{q}_{\tilde{u}}^T(-1)\tilde{u}(-1) \\
& + \frac{1}{2}\sum_{\tau=0}^{N-1}\Big(\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) + \tilde{u}^T(\tau)\tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) \\
& + 2\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)\tilde{u}(\tau) + 2\tilde{q}_{\tilde{u}}^T(\tau)\tilde{u}(\tau)\Big) + \tilde{q}_{\tilde{x}}^T(N)\tilde{x}(N) \\
\text{subject to} \quad & \tilde{x}(0) = \tilde{B}(-1)\tilde{u}(-1) \\
& \tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}(\tau)\tilde{u}(\tau), \qquad \tau = 0,\ldots,N-1 \\
& \begin{bmatrix} 0 & -I_{c(N-\tau-1)} \end{bmatrix}\tilde{u}(\tau) \leq 0, \qquad \tau = -1,\ldots,N-1
\end{aligned}
\tag{30}
$$

where

$$
\tilde{\mathsf{x}} = \left[\tilde{x}^T(0),\ldots,\tilde{x}^T(N)\right]^T, \ \tilde{\mathsf{u}} = \left[\tilde{u}^T(-1),\ldots,\tilde{u}^T(N-1)\right]^T \tag{31}
$$

Algorithm 1 Gradient Projection for QP Dual

```
Compute a feasible starting point $(\lambda_0, \nu_0)$ (trivial).
Define the maximum number of iterations as $k_{max}$.
Define the maximum number of Newton iterations per gradient projection as $m_{max} \geq 1$.
$\mathcal{W}_0 \leftarrow \emptyset$
$k \leftarrow 0$
while $k < k_{max}$ do
    Perform a projected gradient step from $(\lambda_k, \nu_k)$, denote the result $(\lambda_k^c, \nu_k^c)$ and $\mathcal{W}_k^c$.
    $\left(\lambda_0^N, \nu_0^N\right) \leftarrow (\lambda_k^c, \nu_k^c)$ and $\mathcal{W}_0^N \leftarrow \mathcal{W}_k^c$.
    if Unbounded step length then
        // Dual unbounded ⇒ Primal infeasible
        STOP
    end if
    $m \leftarrow 0$
    while $m < m_{max}$ do
        if Newton system inconsistent then
            Perform a projected step from $\left(\lambda_m^N, \nu_m^N\right)$ in the direction of the steepest descent direction projected onto the nullspace of the Newton system.
        else
            Perform a projected Newton step from $\left(\lambda_m^N, \nu_m^N\right)$ subject to the constraints indexed by $\mathcal{W}_m^N$, denote the result $\left(\lambda_{m+1}^N, \nu_{m+1}^N\right)$ and $\mathcal{W}_{m+1}^N$ (where $\mathcal{W}_{m+1}^N \supseteq \mathcal{W}_m^N$).
        end if
        if Unbounded step length then
            // Dual unbounded ⇒ Primal infeasible
            STOP
        end if
        if $\mathcal{W}_m^N = \mathcal{W}_{m+1}^N$ then
            // Optimizer on a subspace found
            Compute dual variables
            if Dual feasible then
                // KKT conditions satisfied
                STOP
            end if
            Exit while
        end if
        $m \leftarrow m + 1$
    end while
    $(\lambda_k, \nu_k) \leftarrow \left(\lambda_{m+1}^N, \nu_{m+1}^N\right)$
    $k \leftarrow k + 1$
end while
No solution was found in $k_{max}$ iterations.
```

and where $\tilde{x}(\tau) \in \mathbb{R}^{\tilde{n}}$ and $\tilde{u}(\tau) \in \mathbb{R}^{\tilde{m}(\tau)}$. For a detailed derivation of the dual problem in (30), see [1]. The relations to the primal quantities are given by

$$\begin{aligned}
\tilde{Q}_{\tilde{u}}(-1) &= \operatorname{diag}(Q_e^{-1}(N), 0) \\
\tilde{q}_{\tilde{u}}(-1) &= \begin{bmatrix} 0 & -h^T(N) \end{bmatrix}^T \\
\tilde{Q}_{\tilde{x}}(\tau) &= B(N-\tau-1)Q_u^{-1}(N-\tau-1)B^T(N-\tau-1) \\
\tilde{Q}_{\tilde{u}}(\tau) &= \operatorname{diag}(Q_e^{-1}(N-\tau-1), H_u(N-\tau-1)Q_u^{-1}(N-\tau-1)H_u^T(N-\tau-1)) \\
\tilde{Q}_{\tilde{x}\tilde{u}}(\tau) &= \begin{bmatrix} 0 & B(N-\tau-1)Q_u^{-1}(N-\tau-1)H_u^T(N-\tau-1) \end{bmatrix} \\
\tilde{q}_{\tilde{u}}(\tau) &= \begin{bmatrix} 0 & -h^T(N-\tau-1) \end{bmatrix}^T \\
\tilde{A}(\tau) &= A^T(N-\tau-1) \\
\tilde{B}(\tau) &= \begin{bmatrix} M^T(N-\tau-1) & H_x^T(N-\tau-1) \end{bmatrix}, \; \tau = -1, \ldots, N-2 \\
\tilde{q}_{\tilde{x}}(N) &= -x_0 \\
\tilde{B}(N-1) &= \begin{bmatrix} M^T(0) & 0 \end{bmatrix}
\end{aligned} \tag{32}$$

where the relations hold for $\tau = 0, \ldots, N-1$ unless stated differently. The dual state dimension and the dual control signal dimension are related to the dimensions of the primal variables by the equations $\tilde{n} = n$ and $\tilde{m}(\tau) = p + c(N-\tau-1)$. More on duality for optimal control problems related to the one in (1), and the interpretation, can be found in [34] and [21].

Note that, by Assumption 1, Assumption 2, and the equations in (32), the following inequality holds

$$\begin{bmatrix} \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}\tilde{u}}(\tau) \\ \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau) & \tilde{Q}_{\tilde{u}}(\tau) \end{bmatrix} \succeq 0, \; \tau = -1, \ldots, N-1 \tag{33}$$

A slight abuse of notation is used when (30) is referred to as the dual problem of (1), while the correct relation is that the latter problem is *equivalent* to the dual problem of the former.

Remark 2. 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 making the definition $\tilde{q}_{\tilde{u}}(\tau) = \begin{bmatrix} r^T(N-\tau-1) & -h^T(N-\tau-1) \end{bmatrix}^T$.

4.2 Connection Between Primal and Dual Variables

As previously mentioned, the idea is to compute the solution to the primal problem from the solution to the dual problem. In the primal problem, the primal variables are x, e and u. In the dual problem, the primal variables are $\tilde{\mathsf{x}}$ and $\tilde{\mathsf{u}}$, and the dual variables are λ and μ.

Start with an optimal solution to the dual problem consisting of $\tilde{\mathsf{x}}$, $\tilde{\mathsf{u}}$, λ, and μ. Then, from the equations corresponding to (68), the following expression for u is obtained

$$u(t) = -Q_u^{-1}(t)\big(B^T(t)\tilde{x}(N-t-1) + H_u^T(t)\begin{bmatrix} 0 & I_{c(t)} \end{bmatrix}\tilde{u}(N-t-1)\big) \tag{34}$$

Furthermore, it can be shown that

$$x(t) = -\lambda(N - t) \tag{35}$$

For a detailed derivation of this equation, see [1]. Note that, the definitions of $w(\tau)$ and $v(\tau)$ have been slightly changed in this report compared to [1].

4.3 The Consequence of Branching

As previously discussed in Section 3.2, during a branch in branch and bound, a subproblem is split into two new subproblems where one of the previously relaxed binary variables are locked to either 0 or to 1. This means that in one of the new subproblems, an inequality constraint in the form $u(t) \geq 0$ has been converted into an equality constraint in the form $u(t) = 0$, and in the other new subproblem, an inequality constraint in the form $u(t) \leq 1$ has been converted into an equality constraint in the form $u(t) = 1$. Now, the observation from Section 3 is useful; converting a primal inequality constraint into a similar equality constraint changes the dual problem only in the way that a dual inequality constrained is removed. The interpretation in the dual MPC problem in (30) is that a lower bound constraint on one of the dual control signals is dropped and the dual control signal becomes unconstrained. Hence, all problems in the branch and bound tree will be in the form in (30), the difference is only the ratio between the number of inequality constrained and unconstrained dual control signals.

5 Efficient Computation of Search Directions

In this section, the most computationally demanding parts of the algorithm presented in Section 3.3 are tailored for the specific application MPC. In Section 5.1, an efficient algorithm for the projected line search operation is presented. In this work, three different search directions are used as inputs to this algorithm. First, the steepest descent direction is always used in “Main step 1”. How these computations can be tailored is described in Section 5.2. Second, if the KKT system is non-singular, the Newton step is used in “Main step 2”. A tailored algorithm is presented in Section 5.3. Third, if it is singular, an alternative search direction is used in “Main step 2”. The singularity means that either there exist several optimal solutions to the subproblem or that there does not exist any stationary point. Usually, the reason for singularity is that the dual problem does not have any stationary points, *i.e.*, it is unbounded. In this case it is interesting to take a step in an unbounded direction in order to find out if there are any inequality constraints that can stop the unbounded ray. In this work, this direction is chosen as the projection of the steepest descent direction at the current point onto the nullspace of the KKT system. An efficient algorithm for computation of this search direction is presented in Section 5.4.

5.1 Tailored Projected Line Search

The projected line search has been previously described in Section 3.3. The projected line search operation is performed by first finding breakpoints where the search direction is bent and second performing one-dimensional optimizations along some of the segments between some of the breakpoints. The one-dimensional objective function on each of those segments is a convex quadratic function, which makes it easy to find the exact optimizer for a segment. The procedure to find breakpoints is not tailored in this work, *i.e.*, it is performed as previously presented in Section 3.3. However, the one-dimensional line search procedure given a certain segment is tailored, and those ideas are now described in detail.

The algorithm is derived for a search in the dual control signal space in an arbitrary search direction $\Delta\tilde{u}$ parameterized by the parameter s, *i.e.*, $\tilde{\mathsf{u}} = \tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s$. The corresponding step in the state $\tilde{\mathsf{x}}$ is written as $\tilde{\mathsf{x}} = \tilde{\mathsf{x}}^0 + \mathsf{K}_1 s$, where

$$\begin{aligned}\tilde{\mathsf{x}}^0 &= \begin{bmatrix}\tilde{x}^{0T}(0) & \dots & \tilde{x}^{0T}(N)\end{bmatrix}^T, \; \tilde{x}^0(\tau) = \sum_{k=0}^{\tau-1} \tilde{A}(\tau-1)\cdot\ldots\cdot\tilde{A}(0)\tilde{B}(k)\tilde{u}^0(k)\\ \mathsf{K}_1 &= \begin{bmatrix}K_1^T(1) & \dots & K_1^T(N+1)\end{bmatrix}^T, \; K_1(\tau) = \sum_{k=0}^{\tau-1} \tilde{A}(\tau-1)\cdot\ldots\cdot\tilde{A}(0)\tilde{B}(k)\Delta\tilde{u}(k)\end{aligned} \tag{36}$$

The constants $\tilde{\mathsf{x}}_0$ and $\tilde{\mathsf{u}}_0$ represent the starting point from which the line search is initialized and $\Delta\tilde{x}$ and $\Delta\tilde{u}$ are components of the line search direction p^{j-1} used in the presentation in Section 3.3. The new objective function in the single variable s can be written as

$$\begin{aligned}&J_D(\tilde{\mathsf{x}}^0 + \mathsf{K}_1 s, \tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s) = \frac{1}{2}\left(\tilde{\mathsf{x}}^0 + \mathsf{K}_1 s\right)^T \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}}\left(\tilde{\mathsf{x}}^0 + \mathsf{K}_1 s\right)\\ &+ \frac{1}{2}\left(\tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s\right)^T \tilde{\mathsf{Q}}_{\tilde{\mathsf{u}}}\left(\tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s\right) + \left(\tilde{\mathsf{x}}^0 + \mathsf{K}_1 s\right)^T \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\tilde{\mathsf{u}}}\left(\tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s\right) + \tilde{\mathsf{q}}_{\tilde{\mathsf{x}}}^T\left(\tilde{\mathsf{x}}^0 + \mathsf{K}_1 s\right)\\ &+ \tilde{\mathsf{q}}_{\tilde{\mathsf{u}}}^T\left(\tilde{\mathsf{u}}^0 + \Delta\tilde{\mathsf{u}}s\right)\end{aligned} \tag{37}$$

This a new QP in one variable. Since the objective function of the original QP is a convex function, and it is well-known that a convex function is still convex when it is restricted to any line that intersects its domain, [10], it is clear that the function in (37) is convex and, hence, the optimization problem of minimizing the function in (37) with respect to s is a convex optimization problem. From the first order necessary and sufficient conditions of optimality, the optimal s is found as

$$s^* = -\frac{\mathsf{K}_1^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}}\tilde{\mathsf{x}}^0 + \Delta\tilde{\mathsf{u}}^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{u}}}\tilde{\mathsf{u}}^0 + \tilde{\mathsf{x}}^{0T}\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\tilde{\mathsf{u}}}\Delta\tilde{\mathsf{u}} + \mathsf{K}_1^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\tilde{\mathsf{u}}}\tilde{\mathsf{u}}_0 + \tilde{\mathsf{q}}_{\tilde{\mathsf{x}}}\mathsf{K}_1 + \tilde{\mathsf{q}}_{\tilde{\mathsf{u}}}\Delta\tilde{\mathsf{u}}}{\mathsf{K}_1^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}}\mathsf{K}_1 + \Delta\tilde{\mathsf{u}}^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{u}}}\Delta\tilde{\mathsf{u}} + 2\mathsf{K}_1^T\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\tilde{\mathsf{u}}}\Delta\tilde{\mathsf{u}}} \tag{38}$$

By studying the structure of the equation in (38), s^* can be found by a recursive algorithm with computational complexity $\mathcal{O}(N)$. One such algorithm is presented in Algorithm 2. An important special case that has to be monitored is if $K_7 = 0$ (K_7 is basically the Hessian in the scalar QP problem). If simultaneously $K_6 \neq 0$, the objective function is unbounded in the direction $\Delta\tilde{u}$, and if $K_6 = 0$, there are multiple solutions and any choice of s^* (in the interval in s under consideration) will be optimal.

The efficiency of Algorithm 2 can be even further improved by utilizing that the search direction is only slightly changed between the different intervals, *i.e.*, previous computations can be reused during coming line optimizations.

Algorithm 2 Line search tailored for MPC

```
$K_1(0) = \tilde{B}(-1)\Delta\tilde{u}(-1)$
$K_3(-1) = \Delta\tilde{u}^T(-1)\tilde{Q}_{\tilde{u}}(-1)$
$K_5(-1) = \tilde{q}_{\tilde{u}}^T(-1)$
for $\tau = 0,\dots,N-1$ do
    $K_1(\tau+1) = \tilde{A}(\tau)K_1(\tau) + \tilde{B}(\tau)\Delta\tilde{u}(\tau)$
    $K_2(\tau) = K_1^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)$
    $K_3(\tau) = \Delta\tilde{u}^T(\tau)\tilde{Q}_{\tilde{u}}(\tau)$
    $K_4(\tau) = K_1^T(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau)$
    $K_5(\tau) = \tilde{x}^{0T}(\tau)\tilde{Q}_{\tilde{x}\tilde{u}}(\tau) + \tilde{q}_{\tilde{u}}^T(\tau)$
end for
$K_6 = K_3(-1)\tilde{u}^0(-1) + K_5(-1)\Delta\tilde{u}(-1)$
$K_7 = K_3(-1)\Delta\tilde{u}(-1)$
for $\tau = 0,\dots,N-1$ do
    $K_6 = K_6 + K_2(\tau)\tilde{x}^0(\tau) + \big(K_3(\tau) + K_4(\tau)\big)\tilde{u}^0(\tau) + K_5(\tau)\Delta\tilde{u}(\tau)$
    $K_7 = K_7 + K_2(\tau)K_1(\tau) + \big(K_3(\tau) + 2K_4(\tau)\big)\Delta\tilde{u}(\tau)$
end for
$K_7 = K_7 + \tilde{q}_{\tilde{x}}^T(N)K_1(N)$
$s^* = -\frac{K_6}{K_7}$
```

5.2 Computation of Steepest Descent Direction

The steepest descent direction is the direction of the negative gradient. In this section, it will be shown how the gradient of the objective function with respect to $\tilde{\mathsf{u}}$ can be computed efficiently for this dual MPC problem. The inequality constraints are not considered in this computation, since they are handled by the projection operation in the line search operation. Note that, because of the dynamics, also $\tilde{\mathsf{x}}$ depends on $\tilde{\mathsf{u}}$. Consider a quadratic objective function in the form in (30). If the objective function is differentiated with respect to $\tilde{\mathsf{u}}(\tau)$, and the fact that $\tilde{x}(\tau)$ is a function of $\tilde{\mathsf{u}}(s),\ s < \tau$ is utilized, the result is

$$\frac{\partial J_D(\tilde{\mathsf{x}}(\tilde{\mathsf{u}}),\tilde{\mathsf{u}})}{\partial\tilde{u}(\tau)} = \sum_{k=\tau+1}^{N} \frac{\partial J_D(\tilde{\mathsf{x}},\tilde{\mathsf{u}})}{\partial\tilde{x}(k)}\frac{\partial\tilde{x}(k)}{\partial\tilde{u}(\tau)} + \frac{\partial J_D(\tilde{\mathsf{x}},\tilde{\mathsf{u}})}{\partial\tilde{u}(\tau)} \tag{39}$$

for $\tau = -1,\dots,N-1$, where

$$\frac{\partial J_D(\tilde{\mathsf{x}},\tilde{\mathsf{u}})}{\partial\tilde{x}(k)} = \begin{cases}\tilde{Q}_{\tilde{x}}(k)\tilde{x}(k) + \tilde{Q}_{\tilde{x}\tilde{u}}(k)\tilde{u}(k),\ 0 \le k \le N-1\\ \tilde{q}_{\tilde{x}}(N),\ k = N\end{cases} \tag{40}$$

and

$$\frac{\partial\tilde{x}(k)}{\partial\tilde{u}(\tau)} = \begin{cases}0,\ k \le \tau\\ \tilde{B}(\tau),\ k = \tau+1\\ \left(\prod_{j=\tau+1}^{k-1}\tilde{A}(j)\right)\tilde{B}(\tau),\ k > \tau+1\end{cases} \tag{41}$$

and

$$\frac{\partial J_D(\tilde{\mathsf{x}},\tilde{\mathsf{u}})}{\partial\tilde{u}(\tau)} = \tilde{Q}_{\tilde{x}\tilde{u}}^T(\tau)\tilde{x}(\tau) + \tilde{Q}_{\tilde{u}}(\tau)\tilde{u}(\tau) + \tilde{q}_{\tilde{u}}(\tau) \tag{42}$$

Algorithm 3 Steepest descent direction computation tailored for (dual) MPC

```
$K = \tilde{q}_{\tilde{x}}(N)$
$\Delta\tilde{u}_d(N-1) = -\Big(\tilde{B}^T(N-1)K + \tilde{Q}^T_{\tilde{x}\tilde{u}}(N-1)\tilde{x}(N) + \tilde{Q}_{\tilde{u}}(N-1)\tilde{u}(N-1)$
$+\tilde{q}_{\tilde{u}}(N-1)\Big)$
for $i = N-1, \ldots, 1$ do
  $K = \tilde{A}^T(i)K + \tilde{Q}_{\tilde{x}}(i)\tilde{x}(i) + \tilde{Q}_{\tilde{x}\tilde{u}}(i)\tilde{u}(i)$
  $\Delta\tilde{u}_d(i-1) = -\Big(\tilde{B}^T(i-1)K + \tilde{Q}^T_{\tilde{x}\tilde{u}}(i-1)\tilde{x}(i-1) + \tilde{Q}_{\tilde{u}}(i-1)\tilde{u}(i-1)$
  $+\tilde{q}_{\tilde{u}}(i-1)\Big)$
end for
$K = \tilde{A}^T(0)K + \tilde{Q}_{\tilde{x}}(0)\tilde{x}(0) + \tilde{Q}_{\tilde{x}\tilde{u}}(0)\tilde{u}(0)$
$\Delta\tilde{u}_d(-1) = -\Big(\tilde{B}^T(0)K + \tilde{Q}_{\tilde{u}}(-1)\tilde{u}(-1) + \tilde{q}_{\tilde{u}}(-1)\Big)$
```

An efficient algorithm for computation of the steepest descent direction is presented in Algorithm 3, where $\Delta\tilde{u}_d(\tau), \tau = -1, \ldots, N-1$ represents the steepest descent direction at the point $\tilde{x}(\tau), \tau = 0, \ldots, N$ and $\tilde{u}(\tau), \tau = -1, \ldots, N-1$. Since it works blockwise, its computational complexity is $\mathcal{O}(N)$.

It should be noted that in most cases, the gradient does not have to be explicitly computed. In Appendix A.2 it is shown that the gradient can be found from the Newton step computation as the the dual variables associated with some of the equality constraints. Hence, the steepest descent direction is found by flipping the sign of these variables.

5.3 Newton Step Computation

The Newton step computation is one of the most important parts of the algorithm presented in this report. In each iteration in “Main step 2”, the solution to an equality constrained QP in the form in (26) has to be computed. This means that for a subset of the inequality constrained components in $\tilde{u}(\tau)$, the inequality constraints are restricted to equality constraints and for other components the inequality constraints are relaxed by ignoring them.

In order to get a problem in the form in (26), rewrite the optimization problem in a form such that $\tilde{u}(\tau)$ are split into two parts, where the first part, which is denoted $w(\tau)$, is not subject to any inequality constraints and the second part, which is denoted $v(\tau)$, is constrained to be *equal* to zero. Then the equality constrained problem can be written in the form in (43).

$$
\begin{aligned}
\underset{\tilde{\mathsf{x}},\mathsf{w},\mathsf{v}}{\text{minimize}} \quad & \bar{J}_D(\tilde{\mathsf{x}},\mathsf{w},\mathsf{v}) \\
\text{subject to} \quad & \tilde{x}(0) = \begin{bmatrix} \tilde{B}_w(-1) & \tilde{B}_v(-1) \end{bmatrix} \begin{bmatrix} w(-1) \\ v(-1) \end{bmatrix} \\
& \tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \begin{bmatrix} \tilde{B}_w(\tau) & \tilde{B}_v(\tau) \end{bmatrix} \begin{bmatrix} w(\tau) \\ v(\tau) \end{bmatrix}, \qquad \tau = 0,\ldots,N-1 \\
& v(\tau) = 0, \qquad \tau = -1,\ldots,N-1
\end{aligned}
\tag{43}
$$

where

$$
\begin{aligned}
\bar{J}_D(\tilde{\mathsf{x}},\mathsf{w},\mathsf{v}) &= \frac{1}{2}\begin{bmatrix} w^T(-1) & v^T(-1) \end{bmatrix} \begin{bmatrix} \tilde{Q}_w(-1) & \tilde{Q}_{wv}(-1) \\ \tilde{Q}_{wv}^T(-1) & \tilde{Q}_v(-1) \end{bmatrix} \begin{bmatrix} w(-1) \\ v(-1) \end{bmatrix} \\
&+ \begin{bmatrix} \tilde{q}_w^T(-1) & \tilde{q}_v^T(-1) \end{bmatrix} \begin{bmatrix} w(-1) \\ v(-1) \end{bmatrix} + \frac{1}{2}\sum_{\tau=0}^{N-1}\Bigg(\tilde{x}^T(\tau)\tilde{Q}_{\tilde{x}}(\tau)\tilde{x}(\tau) \\
&+ \begin{bmatrix} w^T(\tau) & v^T(\tau) \end{bmatrix} \begin{bmatrix} \tilde{Q}_w(\tau) & \tilde{Q}_{wv}(\tau) \\ \tilde{Q}_{wv}^T(\tau) & \tilde{Q}_v(\tau) \end{bmatrix} \begin{bmatrix} w(\tau) \\ v(\tau) \end{bmatrix} + 2\tilde{x}^T(\tau)\begin{bmatrix} \tilde{Q}_{\tilde{x}w}(\tau) & \tilde{Q}_{\tilde{x}v}(\tau) \end{bmatrix} \begin{bmatrix} w(\tau) \\ v(\tau) \end{bmatrix} \\
&+ 2\begin{bmatrix} \tilde{q}_w^T(\tau) & \tilde{q}_v^T(\tau) \end{bmatrix} \begin{bmatrix} w(\tau) \\ v(\tau) \end{bmatrix} \Bigg) + \tilde{q}_{\tilde{x}}^T(N)\tilde{x}(N)
\end{aligned}
\tag{44}
$$

and $\mathsf{w} = \begin{bmatrix} w^T(-1),\ldots,w^T(N-1) \end{bmatrix}^T$, $\mathsf{v} = \begin{bmatrix} v^T(-1),\ldots,v^T(N-1) \end{bmatrix}^T$. Furthermore, $w(\tau) \in \mathbb{R}^{\tilde{m}_w(\tau)}$ and $v(\tau) \in \mathbb{R}^{\tilde{m}_v(\tau)}$, where $\tilde{m}_w(\tau) = m(\tau) - n_a(\tau)$ and $\tilde{m}_v(\tau) = n_a(\tau)$. The constants $n_a(\tau)$ denote the number of inequality constraints that are converted to equality constraints at time instant τ. The Newton step at a point $(\tilde{\mathsf{x}},\mathsf{w},\mathsf{v})$ is found as the vector from $(\tilde{\mathsf{x}},\mathsf{w},\mathsf{v})$ to the optimal solution to the problem in (43).

Efficient Factorization of the KKT System Coefficient Matrix

After a straightforward elimination of the variable v, the optimality condition for the optimization problem in (43) is

$$
\begin{bmatrix} 0 & \tilde{\mathsf{A}} & \tilde{\mathsf{B}}_w \\ \tilde{\mathsf{A}}^T & \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} & \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}} \\ \tilde{\mathsf{B}}_w^T & \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}^T & \tilde{\mathsf{Q}}_{\mathsf{w}} \end{bmatrix} \begin{bmatrix} \lambda \\ \tilde{\mathsf{x}} \\ \mathsf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ -\tilde{\mathsf{q}}_{\tilde{\mathsf{x}}} \\ -\tilde{\mathsf{q}}_{\mathsf{w}} \end{bmatrix}
\tag{45}
$$

where $\lambda = \begin{bmatrix} \lambda^T(0),\ldots,\lambda^T(N) \end{bmatrix}^T$, and where $\tilde{\mathsf{A}}$, $\tilde{\mathsf{B}}_w$, $\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}}$, $\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}$, $\tilde{\mathsf{Q}}_{\mathsf{w}}$, $\tilde{\mathsf{q}}_{\tilde{\mathsf{x}}}$ and $\tilde{\mathsf{q}}_{\mathsf{w}}$ are defined in Appendix A.3. This system of equations is in this text referred to as the KKT system. In order to solve this system of equations efficiently, the coefficient matrix is factorized. If there exist matrices $\tilde{\mathsf{P}} \in \mathbb{S}^{(N+1)\tilde{n}}$, $\tilde{\mathsf{G}} \in \mathbb{S}^{\sum_{\tau=-1}^{N-1}\tilde{m}_w(\tau)}$ and $\tilde{\mathsf{K}} \in \mathbb{R}^{\sum_{\tau=-1}^{N-1}\tilde{m}_w(\tau)\times(N+1)\tilde{n}}$ such that

$$
\begin{aligned}
\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} &= -\tilde{\mathsf{A}}^T\tilde{\mathsf{P}}\tilde{\mathsf{A}} - \tilde{\mathsf{P}}\tilde{\mathsf{A}} - \tilde{\mathsf{A}}^T\tilde{\mathsf{P}} + \tilde{\mathsf{K}}^T\tilde{\mathsf{G}}\tilde{\mathsf{K}} \\
\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}} &= -\tilde{\mathsf{A}}^T\tilde{\mathsf{P}}\tilde{\mathsf{B}}_w - \tilde{\mathsf{P}}\tilde{\mathsf{B}}_w - \tilde{\mathsf{K}}^T\tilde{\mathsf{G}} \\
\tilde{\mathsf{Q}}_{\mathsf{w}} &= -\tilde{\mathsf{B}}_w^T\tilde{\mathsf{P}}\tilde{\mathsf{B}}_w + \tilde{\mathsf{G}}
\end{aligned}
\tag{46}
$$

then the following factorization holds

$$
\begin{aligned}
&\begin{bmatrix} \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} & \tilde{\mathsf{A}}^T & \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}} \\ \tilde{\mathsf{A}} & 0 & \tilde{\mathsf{B}}_w \\ \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}^T & \tilde{\mathsf{B}}_w^T & \tilde{\mathsf{Q}}_{\mathsf{w}} \end{bmatrix} \\
&\qquad = \begin{bmatrix} \left(\tilde{\mathsf{A}} + \tilde{\mathsf{B}}_w \tilde{\mathsf{K}}\right)^T & -\tilde{\mathsf{P}} & -\tilde{\mathsf{K}}^T \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \tilde{\mathsf{B}}_w^T & 0 & I \end{bmatrix} \begin{bmatrix} -\tilde{\mathsf{P}} & I & 0 \\ I & 0 & 0 \\ 0 & 0 & \tilde{\mathsf{G}} \end{bmatrix} \cdot \\
&\qquad\qquad \begin{bmatrix} I & 0 & \tilde{\mathsf{B}}_w \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \tilde{\mathsf{A}} + \tilde{\mathsf{B}}_w \tilde{\mathsf{K}} & 0 & 0 \\ -\tilde{\mathsf{P}} & I & 0 \\ -\tilde{\mathsf{K}} & 0 & I \end{bmatrix}
\end{aligned} \tag{47}
$$

The existence and uniqueness of such matrices has already been considered in [38] under the assumption that $\tilde{\mathsf{Q}}_{\mathsf{w}} \succ 0$ and it was shown that $\tilde{\mathsf{P}} \succeq 0$ and $\tilde{\mathsf{G}} \succ 0$. In this work, that result is generalized to the singular case, *i.e.*, when $\tilde{\mathsf{Q}}_{\mathsf{w}} \succeq 0$ is singular.

Note that the outer four matrices in (47) are non-singular, and specifically that $\tilde{\mathsf{A}} + \tilde{\mathsf{B}}_w \tilde{\mathsf{K}}$ is invertible for any choice of $\tilde{\mathsf{A}}$, $\tilde{\mathsf{B}}_w$ and $\tilde{\mathsf{K}}$, since it is always a lower triangular matrix with diagonal elements equal to -1. For further details, see Appendix A.4. This implies that the KKT system is non-singular if and only if the center matrix is non-singular. The upper left block $\begin{bmatrix} -\tilde{\mathsf{P}} & I \\ I & 0 \end{bmatrix}$ is non-singular since it can easily be transformed into an upper triangular matrix with elements equal to 1 in the diagonal by changing places of the two block columns. Hence, the KKT system is non-singular if and only $\tilde{\mathsf{G}}$ is non-singular. Furthermore, if the KKT coefficient matrix is singular, the nullspace of the entire matrix stems from the nullspace of $\tilde{\mathsf{G}}$.

Notice that the equation in (46) can be equivalently written in the form

$$
\begin{aligned}
\tilde{\mathsf{P}} &= \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} + \left(I + \mathsf{A}\right)^T \tilde{\mathsf{P}} \left(I + \mathsf{A}\right) - \tilde{\mathsf{K}}^T \tilde{\mathsf{G}} \tilde{\mathsf{K}} \\
\tilde{\mathsf{G}} \tilde{\mathsf{K}} &= -\left(\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}^T + \tilde{\mathsf{B}}_w^T \tilde{\mathsf{P}} \left(I + \tilde{\mathsf{A}}\right)\right) \\
\tilde{\mathsf{G}} &= \tilde{\mathsf{Q}}_{\mathsf{w}} + \tilde{\mathsf{B}}_w^T \tilde{\mathsf{P}} \tilde{\mathsf{B}}_w
\end{aligned} \tag{48}
$$

Note that, it follows directly from the equation in (48) that $\tilde{\mathsf{G}}$ is uniquely determined by $\tilde{\mathsf{P}}$, and that $\tilde{\mathsf{G}} \succeq 0$ if $\tilde{\mathsf{P}} \succeq 0$ since $\tilde{\mathsf{Q}}_{\mathsf{w}} \succeq 0$. It will now be shown that there exist matrices $\tilde{\mathsf{P}} \succeq 0$ and $\tilde{\mathsf{K}}$ such that the equations in (48) hold. These equations can be expressed in the block matrices of which the involved matrices consist. The result is

$$
\begin{aligned}
\tilde{P}(N) &= 0 \\
\tilde{G}(0) &= \tilde{Q}_w(-1) + \tilde{B}_w^T(-1) \tilde{P}(0) \tilde{B}_w(-1) \\
\tilde{F}(\tau+1) &= \tilde{Q}_{\tilde{x}}(\tau) + \tilde{A}^T(\tau) \tilde{P}(\tau+1) \tilde{A}(\tau) \\
\tilde{G}(\tau+1) &= \tilde{Q}_w(\tau) + \tilde{B}_w^T(\tau) \tilde{P}(\tau+1) \tilde{B}_w(\tau) \\
\tilde{H}(\tau+1) &= \tilde{Q}_{\tilde{x}w}(\tau) + \tilde{A}^T(\tau) \tilde{P}(\tau+1) \tilde{B}_w(\tau) \\
\tilde{G}(\tau+1) \tilde{K}(\tau+1) &= -\tilde{H}^T(\tau+1) \\
\tilde{P}(\tau) &= \tilde{F}(\tau+1) - \tilde{K}^T(\tau+1) \tilde{G}(\tau+1) \tilde{K}(\tau+1)
\end{aligned} \tag{49}
$$

where all equations hold for $\tau = 0, \dots, N-1$ unless anything else is stated.

It will now be shown that there exist matrices $\tilde{P}(\tau) \succeq 0$, and $\tilde{K}(\tau+1)$ such that $\tilde{G}(\tau+1)\tilde{K}(\tau+1) = -\tilde{H}^T(\tau+1)$. Furthermore, it will be shown that $\tilde{P}(\tau)$ is unique, also in the case when the KKT system is singular. It follows directly from (49) that $\tilde{P}(N) \succeq 0$. Now, assume that $\tilde{P}(\tau+1) \succeq 0$ for an arbitrary $\tau \in \{0, \dots, N-1\}$. Then, it follows from the equations in (49) and in (33) that $\tilde{F}(\tau+1) \succeq 0$. Furthermore,

$$\begin{aligned} &\begin{bmatrix} \tilde{F}(\tau+1) & \tilde{H}(\tau+1) \\ \tilde{H}^T(\tau+1) & \tilde{G}(\tau+1) \end{bmatrix} \\ &= \begin{bmatrix} \tilde{Q}_{\tilde{x}}(\tau) & \tilde{Q}_{\tilde{x}w}(\tau) \\ \tilde{Q}_{\tilde{x}w}^T(\tau) & \tilde{Q}_w(\tau) \end{bmatrix} + \begin{bmatrix} \tilde{A}^T(\tau) \\ \tilde{B}_w^T(\tau) \end{bmatrix} \tilde{P}(\tau+1) \begin{bmatrix} \tilde{A}(\tau) & \tilde{B}_w(\tau) \end{bmatrix} \succeq 0 \end{aligned} \tag{50}$$

by the equation in (33), the assumption that $\tilde{P}(\tau+1) \succeq 0$, and the fact that the last term in the expression is quadratic. By the Schur complement formula for positive semidefinite matrices the following holds

$$\begin{aligned} &\begin{bmatrix} \tilde{F}(\tau+1) & \tilde{H}(\tau+1) \\ \tilde{H}^T(\tau+1) & \tilde{G}(\tau+1) \end{bmatrix} \succeq 0 \Leftrightarrow \\ &\tilde{G}(\tau+1) \succeq 0,\ \left(I - \tilde{G}(\tau+1)\tilde{G}^\dagger(\tau+1)\right)\tilde{H}^T(\tau+1) = 0, \\ &\tilde{F}(\tau+1) - \tilde{H}(\tau+1)\tilde{G}^\dagger(\tau+1)\tilde{H}^T(\tau+1) \succeq 0 \end{aligned} \tag{51}$$

where † denotes the pseudoinverse. Furthermore, notice that

$$\tilde{P}(\tau) = \tilde{F}(\tau{+}1) - \tilde{K}^T(\tau{+}1)\tilde{G}(\tau{+}1)\tilde{K}(\tau{+}1) = \tilde{F}(\tau{+}1) - \tilde{H}(\tau{+}1)\tilde{G}^\dagger(\tau{+}1)\tilde{H}^T(\tau{+}1) \tag{52}$$

where the second equality follows from a basic property of the pseudoinverse (*i.e.*, $\tilde{G} = \tilde{G}\tilde{G}^\dagger\tilde{G}$), the symmetry of $\tilde{G}(\tau+1)$, and the definitions of $\tilde{K}(\tau+1)$ in (49). By combining (51) and (52), it directly follows that $\tilde{P}(\tau) \succeq 0$, and by induction it follows that this is true for all $\tau = N, \dots, 0$. Furthermore, since there exists a solution $\tilde{K}(\tau+1)$ to a system of equations in the form $\tilde{G}(\tau+1)\tilde{K}(\tau+1) = -\tilde{H}^T(\tau+1)$ if

$$\left(I - \tilde{G}(\tau+1)\tilde{G}^\dagger(\tau+1)\right)\tilde{H}^T(\tau+1) = 0 \tag{53}$$

(*i.e.*, the columns in $\tilde{H}^T(\tau+1)$ are orthogonal to the nullspace of $\tilde{G}^T(\tau+1)$ which is equivalent to that the columns in $\tilde{H}^T(\tau+1)$ are in the range of $\tilde{G}(\tau+1)$), it is possible to conclude that there exist matrices $\tilde{K}(\tau+1)$ for all $\tau = N-1, \dots, 0$. Note, however, that $\tilde{K}(\tau+1)$ is not unique in the case when the KKT system is singular since $\tilde{G}(\tau)$ is singular for at least one τ in that case. Finally, it follows from (52) that $\tilde{P}(\tau)$ is independent of the choice of $\tilde{K}(\tau+1)$, and is hence unique.

Summarizing, the Riccati factorization exists also in the case when the KKT system is singular. However, the factorization is not unique in that case because there is a freedom in the choice of $\tilde{K}(\tau+1)$. Despite this, $\tilde{P}(\tau)$ and $\tilde{G}(\tau)$ are unique.

The factorization can be performed very efficiently as a Riccati recursion. This is described in Algorithm 4, which can be shown to have a computational complexity of approximately $N\left(\frac{\tilde{m}_w^3}{3} + 3\tilde{n}^3 + 5\tilde{n}^2\tilde{m}_w + 4\tilde{n}\tilde{m}_w^2\right)$ flops if $\tilde{A}^T(\tau)\tilde{P}(\tau+1)$ on line 5 is

Algorithm 4 Factorization (Riccati recursion)

```
 1: $\tilde{P}(N) = 0$
 2: $\tilde{G}(0) = \tilde{Q}_w(-1) + \tilde{B}_w^T(-1)\tilde{P}(0)\tilde{B}_w(-1)$
 3: Compute and store a factorization of $\tilde{G}(0)$.
 4: for $\tau = N-1, \ldots, 0$ do
 5:     $\tilde{F}(\tau+1) = \tilde{Q}_{\tilde{x}}(\tau) + \tilde{A}^T(\tau)\tilde{P}(\tau+1)\tilde{A}(\tau)$
 6:     $\tilde{G}(\tau+1) = \tilde{Q}_w(\tau) + \tilde{B}_w^T(\tau)\tilde{P}(\tau+1)\tilde{B}_w(\tau)$
 7:     $\tilde{H}(\tau+1) = \tilde{Q}_{\tilde{x}w}(\tau) + \tilde{A}^T(\tau)\tilde{P}(\tau+1)\tilde{B}_w(\tau)$
 8:     Compute and store a factorization of $\tilde{G}(\tau+1)$.
 9:     Compute one solution $\tilde{K}(\tau+1)$ to $\tilde{G}(\tau+1)\tilde{K}(\tau+1) = -\tilde{H}^T(\tau+1)$.
10:     $\tilde{P}(\tau) = \tilde{F}(\tau+1) - \tilde{K}^T(\tau+1)\tilde{G}(\tau+1)\tilde{K}(\tau+1)$
11: end for
```

reused on line 7 and a factorization of Cholesky type is used on line 8. The algorithm is basically a repetition of the equations in (49).

If the KKT system is singular, this will be found during the factorization on line 8 in Algorithm 4. Generally, a factorization as, *e.g.*, the QR factorization or the SVD can be chosen. They work both in the non-singular case as well as the singular case, and they can be used to compute the nullspace of $\tilde{G}(\tau+1)$, which is needed in the computation of the nullspace of the entire KKT system. This will be considered in detail in Section 5.4. Another alternative (which is used during the complexity calculations earlier) is to use the Cholesky factorization as in [38] and monitor if it breaks down. As long as $\tilde{G}(\tau+1) \succ 0$, the Cholesky factorization can be computed. If it fails, an alternative factorization can be used to compute the nullspace required in the alternative procedure described in Section 5.4.

Solving the KKT System Using a Riccati Recursion

In this section, it will be shown how the KKT system in (45) can be solved efficiently. The solution process is begun by factorizing the coefficient matrix in the KKT system using Algorithm 4. The Newton step computation discussed in this section is only performed if the factorization step terminates without any detection of singularity. If singularity is detected, the search direction presented in the next section is used.

The factorization of $\tilde{G}(\tau+1)$ performed in Algorithm 4 is reused in Algorithm 5 and 6. The factorization of the KKT system coefficient matrix is followed by a backward recursion as described in Algorithm 5. This recursion has a computational complexity of about $2N\left(\tilde{n}+\tilde{m}_w\right)^2$ flops. Thereafter, the forward recursion in Algorithm 6 is performed. It has a computational complexity of about $4N\left(\tilde{n}^2+\tilde{n}\tilde{m}_w\right)$. Finally, the dual variables associated with the equality constraints $v(\tau)=0$ in the optimization problem in (43) are found via the forward recursion in Algorithm 7, which has a computational complexity of about $2N\left(2\tilde{n}+\tilde{m}_w\right)\tilde{m}_v$ flops. Summarizing, the computational for solving the KKT system, including the factorization, grows linearly in N and $\tilde{m}_v$, and cubically in $\tilde{n}$ and $\tilde{m}_w$.

Hence, expressed in primal constants, the computational complexity grows cubically in n, p and $c(N-\tau-1)-n_a(\tau)$.

Algorithm 5 Backward recursion

```
$\tilde{\Psi}(N) = -\tilde{q}_{\tilde{x}}(N)$
for $\tau = N-1, \ldots, 0$ do
    $u_0(\tau+1) = \tilde{G}^{-1}(\tau+1)\left(\tilde{B}_w^T(\tau)\tilde{\Psi}(\tau+1) - \tilde{q}_w(\tau)\right)$
    $\tilde{\Psi}(\tau) = \tilde{A}^T(\tau)\tilde{\Psi}(\tau+1) - \tilde{H}(\tau+1)u_0(\tau+1)$
end for
$u_0(0) = \tilde{G}^{-1}(0)\left(\tilde{B}_w^T(-1)\tilde{\Psi}(0) - \tilde{q}_w(-1)\right)$
```

Algorithm 6 Forward recursion

```
$w(-1) = u_0(0)$
$\tilde{x}(0) = \tilde{B}_w(-1)w(-1)$
for $\tau = 0, \ldots, N-1$ do
    $w(\tau) = u_0(\tau+1) + \tilde{K}(\tau+1)\tilde{x}(\tau)$
    $\tilde{x}(\tau+1) = \tilde{A}(\tau)\tilde{x}(\tau) + \tilde{B}_w(\tau)w(\tau)$
    $\lambda(\tau) = \tilde{P}(\tau)\tilde{x}(\tau) - \tilde{\Psi}(\tau)$
end for
$\lambda(N) = \tilde{q}_1(N)$
```

Algorithm 7 Forward recursion (Dual variables)

```
$\mu(-1) = \tilde{Q}_{wv}(-1)w(-1) + \tilde{B}_v^T(-1)\lambda(0) + \tilde{q}_v(-1)$
for $\tau = 0, \ldots, N-1$ do
    $\mu(\tau) = \tilde{Q}_{\tilde{x}v}^T(\tau)\tilde{x}(\tau) + \tilde{B}_v^T(\tau)\lambda(\tau+1) + \tilde{Q}_{wv}(\tau)w(\tau) + \tilde{q}_v(\tau)$
end for
```

5.4 Projecting the Gradient Onto the Nullspace of the Hessian

If the KKT system is inconsistent, an alternative search direction has to be found as discussed in Section 3.3. In this section, it is described how such a direction can be computed efficiently.

Nullspace Computation

Consider the KKT system coefficient matrix in (47) and multiply out the two outer matrices on each side of the center one. The result is

$$\begin{bmatrix} \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} & \tilde{\mathsf{A}}^T & \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}} \\ \tilde{\mathsf{A}} & 0 & \tilde{\mathsf{B}}_w \\ \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}^T & \tilde{\mathsf{B}}_w^T & \tilde{\mathsf{Q}}_\mathsf{w} \end{bmatrix} = \begin{bmatrix} \tilde{\mathsf{A}}^T & -\tilde{\mathsf{P}}^T & -\tilde{\mathsf{K}}^T \\ 0 & I & 0 \\ \tilde{\mathsf{B}}_w^T & 0 & I \end{bmatrix} \begin{bmatrix} -\tilde{\mathsf{P}} & I & 0 \\ I & 0 & 0 \\ 0 & 0 & \tilde{\mathsf{G}} \end{bmatrix} \begin{bmatrix} \tilde{\mathsf{A}} & 0 & \tilde{\mathsf{B}}_w \\ -\tilde{\mathsf{P}} & I & 0 \\ -\tilde{\mathsf{K}} & 0 & I \end{bmatrix} \triangleq \Pi^T \Sigma \Pi \tag{54}$$

Note that

$$\xi \in \text{null}(\Pi^T \Sigma \Pi) \Leftrightarrow \Pi^T \Sigma \Pi \xi = 0 \Leftrightarrow \Sigma \Pi \xi = 0 \tag{55}$$

since Π is non-singular. Furthermore,

$$\Sigma \Pi \xi = 0 \Leftrightarrow \Sigma \eta = 0 \text{ and } \xi = \Pi^{-1}\eta \tag{56}$$

Let the columns of the matrix N_Σ be a basis for the nullspace of Σ, and the columns of $N_{\Sigma\Pi}$ a basis for the nullspace of $\Pi^T \Sigma \Pi$. It is straightforward to generalize the ideas in (55) and in (56) in order to compute a basis for the nullspace of $\Pi^T \Sigma \Pi$ using a basis of the nullspace of Σ. That is, $N_{\Sigma\Pi}$ can be computed as

$$N_{\Sigma\Pi} = \Pi^{-1} N_\Sigma \tag{57}$$

Note that, Π is non-singular independently of which solution $K(\tau + 1)$ that is used. Furthermore, note that since Π depends on $K(\tau + 1)$, Π is non-unique. However, for every choice of $K(\tau + 1)$, the columns of $N_{\Sigma\Pi}$ forms *one* basis for the nullspace of $\Pi^T \Sigma \Pi$ since Π is always non-singular. The computations necessary to perform the calculation in (57) can be performed very efficiently thanks to the structure of N_Σ and Π. This computation can be expressed as a recursion similar to the one presented in Algorithm 6. Because the upper left block $\begin{bmatrix} -\tilde{\mathsf{P}} & I \\ I & 0 \end{bmatrix}$ in Σ is non-singular, N_Σ has the structure $\begin{bmatrix} 0 \\ N_{\tilde{\mathsf{G}}} \end{bmatrix}$ where the columns of $N_{\tilde{\mathsf{G}}}$ forms a basis for the nullspace of $\tilde{\mathsf{G}}$. Note that, since $\tilde{\mathsf{G}}$ is block diagonal, it is possible to choose a basis for the nullspace such that $N_{\tilde{\mathsf{G}}}$ is also block diagonal.

Projection of the Gradient Onto the Nullspace of the KKT System Coefficient Matrix

In this section, it will be shown how the search direction discussed in Section 3.3 can be computed efficiently for the dual QP problem in the MPC application. The desired search direction is the projection of the gradient in the current point onto the nullspace of the reduced Hessian of the dual problem in (43). Since $\tilde{\mathsf{A}}$ is non-singular, λ and $\tilde{\mathsf{x}}$ can be eliminated from the KKT system in (45). The resulting equation for w is

$$\left(\tilde{\mathsf{Q}}_{\mathsf{w}} - \tilde{\mathsf{B}}_{\mathsf{w}}^T \tilde{\mathsf{A}}^{-T} \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}} - \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\mathsf{w}}^T \tilde{\mathsf{A}}^{-1} \tilde{\mathsf{B}}_{\mathsf{w}} + \tilde{\mathsf{B}}_{\mathsf{w}}^T \tilde{\mathsf{A}}^{-T} \tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} \tilde{\mathsf{A}}^{-1} \tilde{\mathsf{B}}_{\mathsf{w}}\right) \mathsf{w} = 0 \tag{58}$$

Furthermore, if the dual states are eliminated in the problem in (43), and if $\mathsf{v} = 0$ is inserted, it can be seen that the resulting Hessian in the new QP problem will be exactly the matrix in front of w in (58). Hence, the equation for the nullspace of the reduced Hessian (the states are eliminated) in problem (43) is identical to the equation in (58). Therefore, the nullspace of the reduced Hessian can be computed by computing the nullspace of the KKT system coefficient matrix in (45). How this can be computed efficiently has already been shown in the previous section.

Mathematically, the projection of the gradient onto the nullspace spanned by the columns in $N_{\Sigma\Pi}$ can be found as

$$\hat{g} = N_{\Sigma\Pi} \left(N_{\Sigma\Pi}^T N_{\Sigma\Pi}\right)^{-1} N_{\Sigma\Pi}^T g \tag{59}$$

where g is the gradient, and where $N_{\Sigma\Pi}^T N_{\Sigma\Pi}$ is non-singular since the columns of $N_{\Sigma\Pi}$ are linearly independent because they are basis vectors. Note that,

$$-g^T N_{\Sigma\Pi} \left(N_{\Sigma\Pi}^T N_{\Sigma\Pi}\right)^{-1} N_{\Sigma\Pi}^T g \leq 0 \tag{60}$$

in general, but also that

$$-g^T N_{\Sigma\Pi} \left(N_{\Sigma\Pi}^T N_{\Sigma\Pi}\right)^{-1} N_{\Sigma\Pi}^T g < 0 \tag{61}$$

when $N_{\Sigma\Pi}^T g \neq 0$, *i.e.*, $\hat{g}$ is a descent direction if the gradient is not orthogonal to the nullspace. In that case, there are infinitely many optimal solutions to the problem. Due to the structure in $N_{\Sigma\Pi}$, the matrix $N_{\Sigma\Pi}^T N_{\Sigma\Pi}$ is often sparse and computations in (59) can in those cases be performed very efficiently. A truly tailored version of this projection operation is left as future work.

5.5 Increasing Performance

In active set algorithms, similar KKT systems are solved in subsequent iterations. Usually, block elimination is used to take advantage of the fact that some parts of the equation system remain the same during all iterations. Unfortunately, this has not been found as straightforward in the dual case as in the primal case, since the reduced Hessian for the QP subproblems is not in general positive definite. Furthermore, the usefulness of factorization updates decreases in a gradient projection method, since the change in the working set in many iterations is rather large compared to a classical active set method. Hence, an update would often be rather complex. For the method described in this report, there are other ways to reuse computations from previous iterations. Even though the problem is changed in several time instants (seen from an MPC perspective) if several constraints are changed in a single iteration, the Riccati factorization is still unchanged in all time instants after the last modification of the working set. For example, if constraints are added or removed in time instants $\tau = 2$ and $\tau = 5$, $P(\tau)$ is unchanged for $\tau > 5$. Because the Riccati recursion runs backward in time, only time instants before the time step when the last constraint is added or removed have to be recomputed. 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 recomputed.

6 Using the Algorithm in Branch and Bound

The QP algorithm presented in this work is now incorporated as a part of a branch and bound algorithm, where it is used to solve the relaxed integer problems in the nodes of the search tree. Branch and bound has been generally described in Section 2, and in this section some details and possible future work is discussed.

Ideas similar to those presented in Section 5.5 regarding how to reuse previous computations have also been 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 recomputed for the time step the constraint is added and backward. A disadvantage of using this idea at the branch and bound level is that if a branch can be “suspended” before it is pruned, it might occur that several non-pruned subtrees exist and therefore a considerable amount of data might have to be stored for the recomputations. If the storage space is small, a compromise between storing data in all suspended nodes and storing no data at all is needed.

As in [16], in the current implementation of the algorithm, the inequality in (71) is used to prematurely abort the solution of a relaxation as soon as the dual objective function value becomes higher than the currently best known upper bound in the tree.

There are also some possible future enhancements that could increase the performance. Probably, the most effective step would be to design some kind of preprocessing algorithm similar to the one presented in [2]. 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.

7 Numerical Experiments

In this section, the algorithm presented in this report is applied to linear MPC problems and MIPC problems. In all results where random examples are used (*i.e.*, not the satellite example), 10 random problems are solved for each prediction horizon length, and the average result from these problems are presented in the figures.

7.1 Linear MPC

In this section, the algorithm presented in this work is applied to random linear MPC problems in the form in (1) with $n_c = 10, n_b = 0, p_c = 10, p_b = 0, m_c = 5, m_b = 0$ and $c(t) = 10, t = 0, \ldots, N-1$ for different values of N in the range 50 to 450. The random

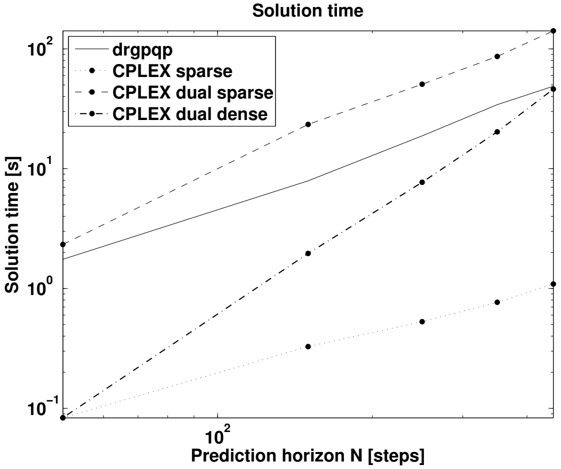


Figure 7: *This plot shows the computational times for different QP solvers. The QP algorithm described in this section is* `drgpqp`*. The performance of this solver is compared with those of* CPLEX*'s primal and dual solvers. The result is that the dual solver presented in this report has better computational complexity compared to* CPLEX*'s dual solvers. Especially, the computational complexity grows significantly slower compared to the dense dual solver in* CPLEX*.*

systems are found using the MATLAB function `drss`. The constraints are bound constraints on the control signals and are chosen such that both feasible as well as infeasible problems are present among the test problems. Out of the 150 problems, 114 are feasible. The reference signal is chosen as a vector of sinus signals with random phases; one for each output of the system. Furthermore, the cost matrices in the objective function have also been chosen randomly, but in a way that they are symmetric and positive definite.

In the examples, dense and sparse formulations of the MPC problem have been solved. The sparse formulation is a formulation in the form in (1). The objective function's Hessian and the equality constraints in this problem are sparse. After eliminating x and e, the result is a new equivalent QP problem without equality constraints. The objective function Hessian of this problem is dense, but of smaller size since the only free variables are the control signals in u.

The problem has been solved for different lengths N of the prediction horizon and the computational time has been evaluated. The result from these tests are shown in Figure 7. In all plots in this section, the following notation is used to identify the different solvers. The algorithm presented in this report is implemented in the function `drgpqp`. "CPLEX sparse" denotes CPLEX given a sparse representation of the MPC problem, and the default settings in CPLEX are used. This means the a primal IP solver will be used to solve the problem. "CPLEX dual sparse" is the sparse problem solved by a dual active set solver in CPLEX. "CPLEX dual dense" is the dense version of the MPC problem solved by a dual solver in CPLEX. As can be seen in Figure 7, `drgpqp` has lower computational complexity compared to CPLEX's dual solvers for large values of N. Since CPLEX uses its dual solver as the default solver for the node problems

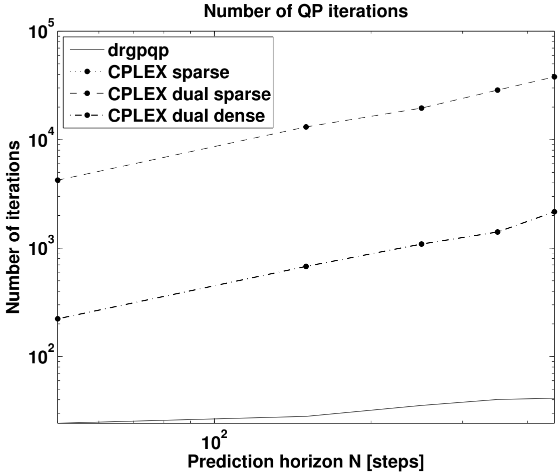


Figure 8: *This plot shows how the number of QP iterations grows as the prediction horizon length grows. Depending on algorithm and implementation, there might be several possible definitions of a "QP iteration". In this experiment it is assumed that the number of reported "simplex iterations" by* CPLEX *is at least proportional to the number of Newton systems solved.*

in branch and cut, the comparisons between drgpqp and CPLEX's dual solvers are the most important ones. The primal solver is only shown as a reference. Note that drgpqp is implemented entirely in m-code, while CPLEX is running in compiled code. Hence, the *trends* are most interesting in this experiment, and the absolute times are of minor interest. According to the result in this experiment, the m-coded solver built on the ideas in this work has lower absolute computational time compared to the dual solver in CPLEX. It is apparently so that CPLEX utilizes the fact that it is a sparse problem, since the computational complexity *grows* significantly slower for this formulation. However, for some reason, the absolute performance is rather bad for this configuration. Even though the solution returned is correct, it cannot be excluded that some error occurs, or some significant overhead is added, *e.g.*, in CPLEXINT during the call to CPLEX. Another, possible explanation is that CPLEX does not solve this problem formulation efficiently. The solution of the dense problem formulation can be performed much more efficiently for the prediction horizon lengths investigated. The computational complexities in this experiment grow approximately as $\mathcal{O}(N^{1.5})$ for drgpqp, $\mathcal{O}(N^{1.2})$ for "CPLEX sparse", $\mathcal{O}(N^{1.9})$ for "CPLEX dual sparse", and $\mathcal{O}(N^{2.9})$ for "CPLEX dual dense".

In Figure 8, the number of QP iterations are compared. Since CPLEX is not open source software, it is hard to make a fare comparison. The reported number is the number denoted "simplex iterations", which is assumed to be proportional to the number of Newton steps computed. Note however, that even though CPLEX reports a larger number of such iterations, each of them is most likely less computationally demanding since CPLEX is probably not making projected line searches after each Newton step computed. Furthermore, CPLEX also most certainly makes updates of factorizations between Newton steps, which also makes the iterations cheaper. If the computational time spent in different parts of the code of `drgpqp` is studied, it can be seen that a significant part of the computational time is spent in the line search code. An idea for future research is, hence, to make this code more efficient. One possible solution could be to use *inexact* line searches. This is commonly used in gradient projection algorithms. However, it seemed reasonable in this case to utilize the simple structure of the QP problem, as suggested in [31], in order to perform exact line searches. An idea worth to test in the future is if inexact line searches would do well enough to keep the number of Newton step computations low, while the cost for the line searches is significantly reduced.

During experiments, different types of problems have been investigated. The presented result is for control signal-constrained problems. Also state-constrained problems have been investigated. The performance seems to be similar. However, for the problem sizes that are investigated here, the solver ran into numerical problems (for $N = 250$), *e.g.*, the Hessian was reported negative during a line search. In problems with constraints on the control signals, only 11 of the 36 infeasible problems were reported feasible. This problem seems to be a result of numerical problems affecting the computed value of the dual variables. When these are tested for positivity, it can occur that numerical errors perturb the value enough to make the solver to make a wrong decision whether an optimal point has been found or not. This in turn could lead to cycling. To avoid that, a threshold has been introduced to make sure that numerical error does not make the algorithm cycle. Unfortunately, this could in turn make it possible that a non-optimal point is taken as the optimum. A better solution would probably to introduce some kind of anti-cycle mechanism which can found in the literature. That is, incorporating such a mechanism is a good suggestion for future research. It would also be necessary to investigate the numerical properties further.

7.2 Mixed Integer Predictive Control

In this section, the algorithm is applied to the MPC problem to control the attitude of a satellite. The system is assumed to be controlled by a reaction wheel, represented by the real-valued control signal u_c, and two binary controlled thrusters, represented by the control signals $u_{b,1}$ and $u_{b,2}$ respectively. 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

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 2.5 & 1 & -1 \\ -10 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_c \\ u_{b,1} \\ u_{b,2} \end{bmatrix}, \; y = I \cdot x$$

To obtain a discrete-time system in the form in (1), zero order hold sampling with sampling time $0.1\,\text{s}$ is used. The magnitude of the real-valued control signal is limited to be less than or equal to 1. The cost function used in this example is of the type in (1), with

$$\begin{aligned} Q_e &= \text{diag}\left(0.5 \cdot 10^4, 1.0 \cdot 10^{-2}, 1.0 \cdot 10^{-1}\right), \\ Q_u &= \text{diag}(Q_{u_c}, Q_{u_b}),\ Q_{u_c} = 10,\ Q_{u_b} = 10 \cdot I_2 \end{aligned} \tag{62}$$

The initial state is

$$x_1(0) = 0,\ x_2(0) = 0,\ x_3(0) = 0 \tag{63}$$

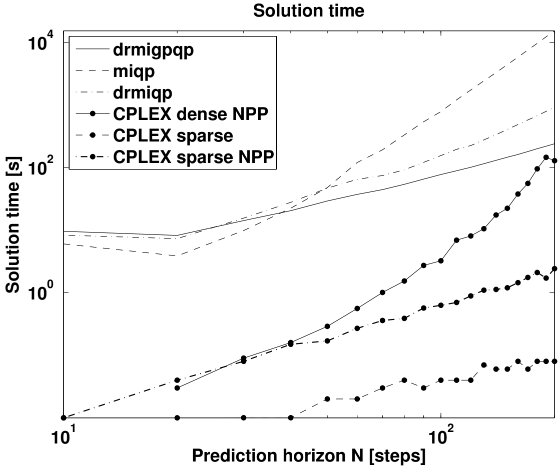


Figure 9: *This plot shows the computational times for six different MIQP solvers. The result is that the algorithm presented in this report is the m-code implementation with the best performance for large values of N. Even though the implementation of the algorithm in this report is far behind* CPLEX *when absolute computational times are compared, its computational time* grows *similar compared to the one of* CPLEX, *also when sparsity is utilized by the latter. It is also obvious that the implementation of* CPLEX *is very good.*

The reference signal is chosen as a step in the satellite attitude of magnitude 0.5 radians given after one fourth of the prediction horizon. The problem has been solved for several different prediction horizons in the range $N = 10$ to $N = 200$ and the corresponding computational times are presented in Figure 9. In all plots in this section, the following notation is used to identify the different solvers. The MIQP algorithm presented in this report is referred to as `drmigpqp`. Furthermore, `miqp` is the freely available solver described in [5] solving a dense (states and control error eliminated) formulation of the problem (fastest choice for this solver since `quadprog` called by `miqp` to solve the relaxations in the nodes cannot utilize sparsity), `drmiqp` is an implementation of the algorithm presented in [3], "CPLEX dense NPP" is CPLEX when given a dense version of the problem and preprocessing has been turned off, "CPLEX sparse" is CPLEX when given a sparse version (states kept) of the problem and default settings are used (*i.e.*, basically all features in CPLEX are enabled), and finally, "CPLEX sparse NPP" is CPLEX when given a sparse version of the problem and preprocessing has been turned off.

In this example, for prediction horizons longer than 20 time steps, the computational complexity for the algorithm presented in this report grows approximately as $\mathcal{O}(N^{1.5})$, while for the generic solver `miqp`, using the QP solver `quadprog`, it grows approximately as $\mathcal{O}(N^{3.6})$. The computational complexity for CPLEX grows in the default configuration approximately as $\mathcal{O}(N^{0.9})$. Since the branch and bound (branch and cut to be precise) part of CPLEX is much more advanced, *i.e.*, highly developed preprocessing and heuristics, *etc.*, than the one in `drmigpqp`, `drmiqp` and `miqp`, CPLEX has been "detuned" in order to make the comparison more fare. This is indicated by the letters "NPP"

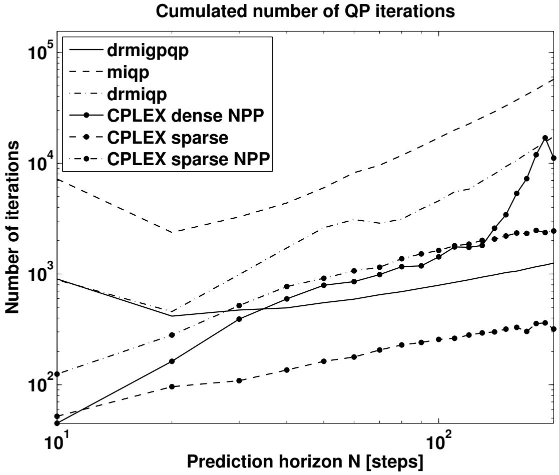


Figure 10: *This plot shows the number of cumulated QP iterations summed from all explored nodes in the branch and bound tree. The purpose is to give an* indication *of how many Newton steps that have to be computed during the solution process of an entire MIQP problem. Note that, however, exactly what is meant by the word "iteration" might be dependent of the algorithm, and the implementation of it. However, it gives an indication of that the algorithm presented in this algorithm is rather efficient.*

(No PreProcessing) in the plots, and the parameters modified are listed in Appendix A.5. After this detuning, CPLEX computational time grows approximately as $\mathcal{O}(N^{1.8})$ using the sparse formulation and $\mathcal{O}(N^{3.6})$ using the dense formulation. The implementation of the branch and bound algorithm used in this work is the one originally implemented in `miqp`, which has been modified in order to be able to use the dual gradient projection algorithm previously presented and to enable the use of warm starts. A similar procedure is also used in `drmiqp`. Hence, it is exactly the same branch and bound code used in the results for `drmigpqp`, `drmiqp` and `miqp`, and therefore, the branch and bound tree has been explored in exactly (except for $N = 10$ where `miqp` explores 529 instead 533 nodes as the other two solvers) the same way. In the result for `miqp`, the MIPC problem has been formulated as an MIQP problem using a dense QP formulation.

The algorithm has also been applied to *random* problems similar to those used in Section 7.1, but with integer variables included in the problem. The results from the tests with random problems are presented in Figure 12, Figure 13 and Figure 14. In this experiment, the computational times grows roughly as $\mathcal{O}(N^{4.9})$ for `drmigpqp`, $\mathcal{O}(N^{1.3})$ for "CPLEX dense", $\mathcal{O}(N^{2.4})$ for "CPLEX sparse", $\mathcal{O}(N^{2.6})$ for "CPLEX sparse NPP", and $\mathcal{O}(N^{5.2})$ for `miqp`. This means that the performance is not as good as in the satellite example. A reasonable explanation is that when more advanced problems are solved, the number of explored nodes explodes and the depth of the tree is expected to grow. This is confirmed by comparing Figure 11 with Figure 14. Hence, a large amount of small relaxations have to be solved. The strategy used in the current implementation is to add equality constraints to lock variables as the the branch and bound search dives down in

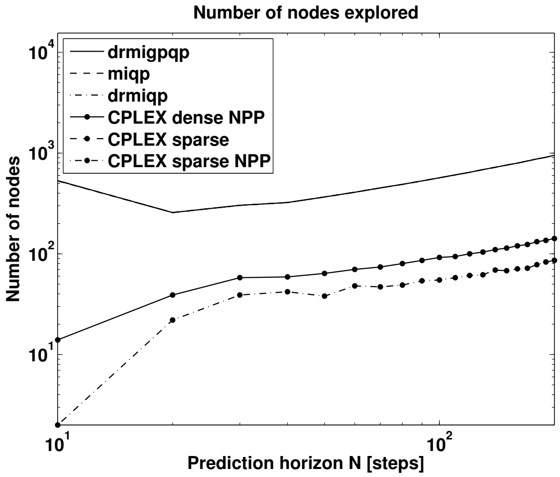


Figure 11: *This plot shows the number of nodes that are explored in branch and bound by the different solvers. The first three solvers explore the same amount of nodes (except in the first problem, when* `miqp` *explores 529 instead 533 nodes. The reason is probably numerical problems, but the solutions returned coincide) since they are all based on the branch and bound part implemented in the solver described in [5].* CPLEX *explores a significantly smaller amount of nodes, even though it has been "detuned" in order to be more comparable with the basic branch and bound method implemented in [5].*

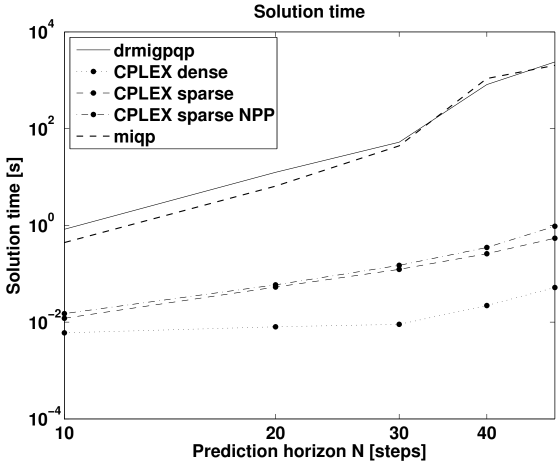


Figure 12: *This plot shows the average computational times required to solve the random MIPC problem. The result for* `drmigpqp` *is not as good as the one in Figure 9.*

the tree. Often, also in `miqp`, the locked variables are eliminated from the problem, and the number of free variables are reduced by one for each level down in the tree. In the implementation of the algorithm in this work, the problem size is kept constant in the branch and bound tree. The reason for this was mainly to decrease the programming effort, which mainly has been concentrated on the implementation of `drgpqp`. Hence, this part of the algorithm, and the corresponding part of the implementation, is an interesting target for future research. Note however, that the increased computational effort for `drmigpqp` *compared to CPLEX* can largely be explained by the difference in the number of nodes that are explored in the branch and bound tree. Hence, the comparison is not all fair, because independently of how efficiently the QP solver for the relaxation is, it cannot compensate for the difference in the number of problems necessary to solve.

7.3 Discussion

The result presented in this report should be interpreted more like a “proof of concept”, rather than as *the* QP solver for MIPC. To be useful in practice, there are some features that remain to be added. For example, an efficient preprocessing algorithm has to be used in and before branch and bound. Another example is that the concept of updating factorizations should be further investigated. It is partly performed in the solver presented in this work, but it might be possible to improve. Furthermore, variables that are locked during the branch and bound process should be removed from the problem, such that the dimensions of the relaxations are reduced when it is possible. Methods that detects and prevents cycling should be incorporated. It would also be interesting to investigate the numerical properties further, and see if it is possible to incorporate ideas from [19], where a dual QP solver with good numerical properties is presented. Finally, the algorithm should be efficiently coded in C-code in order to be able to evaluate its true performance.

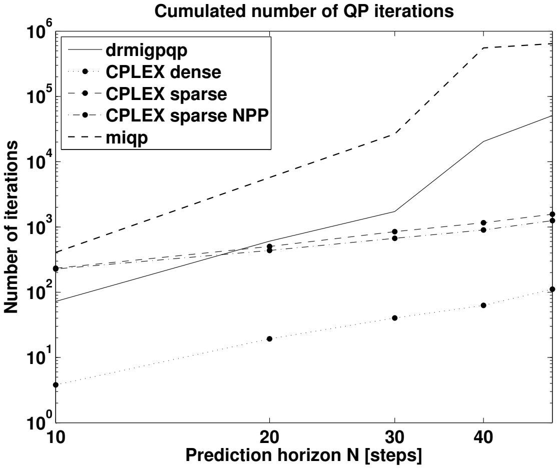


Figure 13: *This plot shows the average number of cumulated QP iterations summed from all explored nodes in the branch and bound tree in the case when random MIPC problems are investigated.*

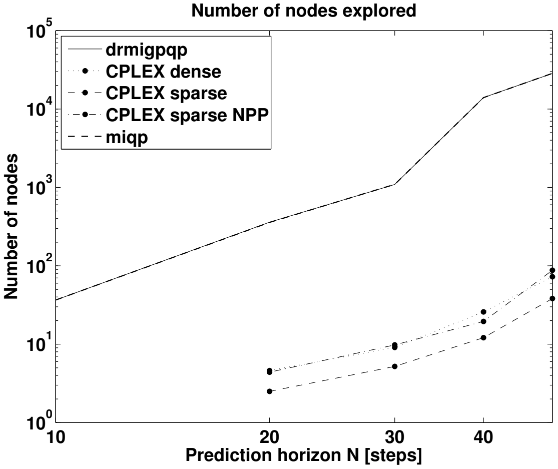


Figure 14: *This plot shows average number of explored nodes in the branch and bound tree in the case when random MIPC problems are investigated. This plot shows one possible explanation to why the performance of* `drmigpqp` *is not as good as in the satellite example case. If this plot is compared with the one in Figure 11, it can be seen that the number of nodes grows much faster with the prediction horizon in the random problem case. Hence, a rather large amount of smaller nodes down in the tree has to be solved, which is unfortunately disadvantageously for the current implementation of the branch and bound algorithm in* `drmigpqp`*, where the sizes of the QP relaxations are not explicitly reduced as the algorithm proceeds down in the branch and bound tree.*

Despite that there are several improvements that could be made, the goal has in large been reached. The algorithm is tailored for MPC, where most computations can be performed with linear complexity in the length of the prediction horizon. The fundamental limitation of classical active set methods, *i.e.*, to require a potentially large number of iterations before the active set in optimum has been found, seems to be overcome by incorporating ideas from gradient projection. The algorithm can easily gain from knowledge of a solution to a previously solved similar problem.

It is of course not realistic to expect that an experimental m-code implementation of a solver, mostly implemented to show “proof of concept”, performs better than a commercial state-of-the-art solver. However, the result shown in this section indicates that the ideas presented in this report are promising and that they potentially could be useful in a high-performance implementation of an MIQP solver for MIPC. It is also interesting to see that the number of QP iterations is reduced by an order of magnitude compared to the solver presented in [3], which was also tailored for this specific application. Basically, the difference between the algorithm in that reference compared to the algorithm presented in this report is that the active set strategy has been “updated” from a classical active set strategy to a strategy based on ideas from gradient projection. Furthermore, the algorithm in this report also handles the case when there does not exist any solution to the Newton step equations in a better way. In [3], the Newton step is in these situations computed by perturbing the Hessian slightly such that it becomes positive definite, while the algorithm presented here explicitly computes a step in the nullspace to the Hessian without any regularizations. In the cases when the Newton steps are defined, they are computed using a similar Riccati solver in both algorithms. Hence, the ideas presented in this report tends to identify the optimal active set using a smaller amount of Newton step computations. However, it should be stressed that each iteration is in general more expensive in the new algorithm since a large number of line searches are performed.

8 Conclusions

The main result in this work is that several important concepts have been combined into an algorithm that has the potential of being a strong alternative when solving MIPC problems. The algorithm works in the dual space to facilitate fast warm starts. Furthermore, it has warm start properties similar to a classical active set method. All computations of major complexity, except for a projection operation, have been tailored for the MPC problem. This operation is not frequently used, but it would be interesting in the future to fill in this last gap to an all tailored algorithm.

In numerical experiments, the performance is good; for QP problems the computational complexity grows similar to, or slower than, the one of CPLEX’s dual solvers as the length of the prediction horizon grows. Furthermore, the results indicate that the algorithm solves the problem using fewer QP iterations than CPLEX. It is also clear that the gradient projection ideas incorporated help to cut down the number of QP iterations, compared to a previously presented tailored dual QP solver. However, the iterations in the algorithm presented in this report are, most likely, more expensive than those in CPLEX. One possible remedy to this problem might be to replace the exact line search used in the current version with an inexact line search. It is future research to see how the allover performance is changed after such a change in the algorithm. When the algorithm is applied to MIPC problems, the performance is still good on simple problems, but it seems like the way in which the branch process changes the subproblems has to be changed in order to maintain high performance also in more difficult problems.

Finally, it should be mentioned that the algorithm is not only interesting for MIPC. Its properties are also useful in ordinary linear MPC where it is useful to be able to warm start given the optimal solution from the previous sample. It might also be useful in non-linear MPC where it can form a part of an SQP solver. Also in that case, good warm start properties are necessary.

Examples of future work are to test inexact line search methods, to test if more advanced factorization updates are necessary and how they can be efficiently implemented, and finally, to change the size of the relaxations as the algorithm gets deeper down in the branch and bound tree.

A Appendix

A.1 Optimality and Strong Duality

The following theorem is given without proof and is based on the discussion in [10, p. 226].

Theorem 1 (Slater's theorem for QP)
For a QP problem in the form in (11)*, strong duality holds if the problem is feasible.*

In some cases, the so-called Karush-Kuhn-Tucker (KKT) conditions provide necessary and sufficient conditions for optimality. In Theorem 2, which is based on the discussion in [10, pp. 243–244], the conditions are presented for the special case of a convex QP problem.

Theorem 2 (KKT for convex QP)
Consider a convex QP problem in the form

$$\begin{aligned} \underset{x}{\textit{minimize}} \quad & \frac{1}{2}x^T H x + f^T x \\ \textit{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}} \\ & A_{\mathcal{I}} x \leq b_{\mathcal{I}} \end{aligned} \tag{64}$$

where $x \in \mathbb{R}^n$, $H \in \mathbb{S}_+^n$, $f \in \mathbb{R}^n$, $A_{\mathcal{E}} \in \mathbb{R}^{p \times n}$, $A_{\mathcal{I}} \in \mathbb{R}^{m \times n}$, $b_{\mathcal{E}} \in \mathbb{R}^p$ and $b_{\mathcal{I}} \in \mathbb{R}^m$. Then the following so-called Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions for x^* and (λ^*, ν^*) to be primal and dual optimal points, respectively

$$A_{\mathcal{I}} x \leq b_{\mathcal{I}} \tag{65a}$$

$$A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{65b}$$

$$\lambda^* \geq 0 \tag{65c}$$

$$\lambda^{*T} \left(A_{\mathcal{I}} x^* - b_{\mathcal{I}} \right) = 0 \tag{65d}$$

$$H x^* + f + A_{\mathcal{I}}^T \lambda^* + A_{\mathcal{E}}^T \nu^* = 0 \tag{65e}$$

Proof: See [10, p. 244], insert the objective function and constraint functions for the QP problem into the KKT conditions and use that strong duality holds for this problem by Theorem 1. □

A.2 Quadratic Programming

Deriving the QP Dual

The Lagrangian L for the problem in (11) is

$$\begin{aligned} L(x_1, x_2, \lambda, \nu) = \frac{1}{2} x_1^T \tilde{H} x_1 + \tilde{f}^T x_1 + \lambda^T A_{1\mathcal{I}} x_1 \\ - \lambda^T b_{\mathcal{I}} + \nu^T A_{1\mathcal{E}} x_1 - \nu^T b_{\mathcal{E}} + \left(\lambda^T A_{2\mathcal{I}} + \nu^T A_{2\mathcal{E}}\right) x_2 \end{aligned} \tag{66}$$

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

$$\begin{aligned} g(\lambda, \nu) &= \inf_{x_1, x_2} L(x_1, x_2, \lambda, \nu) \\ &= \begin{cases} \inf\limits_{x_1 \in \mathbb{R}^{n_1}} \frac{1}{2} x_1^T \tilde{H} x_1 + \left(\tilde{f}^T + \lambda^T A_{1\mathcal{I}} + \nu^T A_{1\mathcal{E}}\right) x_1 - \lambda^T b_{\mathcal{I}} - \nu^T b_{\mathcal{E}}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } \lambda^T A_{2\mathcal{I}} + \nu^T A_{2\mathcal{E}} = 0 \\ -\infty, \text{ otherwise} \end{cases} \end{aligned} \tag{67}$$

According to (67), if g is to be bounded from below, the following condition has to be satisfied

$$\lambda^T A_{2\mathcal{I}} + \nu^T A_{2\mathcal{E}} = 0 \tag{68}$$

If (68) is inserted into (66), the Lagrangian becomes a strictly convex function of x_1 for fixed λ and ν and can easily be minimized using the first order necessary and sufficient conditions for optimality with respect to x_1, which give

$$x_1 = -\tilde{H}^{-1} \left(\tilde{f} + A_{1\mathcal{I}}^T \lambda + A_{1\mathcal{E}}^T \nu\right) \tag{69}$$

The Lagrange dual problem is defined as

$$\begin{aligned} \underset{\lambda, \nu}{\text{maximize}} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{70}$$

Denote the optimal dual objective function value d^*. An important property of the Lagrange dual function is that for any $\lambda \geq 0$, the following inequality holds

$$g(\lambda, \nu) \leq p^* \tag{71}$$

where p^* is the optimal primal objective function value. The inequality in (71) holds specifically for the dual optimal pair (λ^*, ν^*) and thus

$$d^* \leq p^* \tag{72}$$

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. For some problems the inequality in (72) holds with equality, *i.e.*,

$$d^* = p^* \tag{73}$$

This important property is called strong duality. Conditions guaranteeing strong duality are called constraint qualifications. One well-known constraint qualification is Slater's condition which can be found in Theorem 1.

When formulating the dual problem, the implicit constraint in (68) is made explicit by adding it to the list of constraints. After inserting (69) into (67), the dual problem is concluded to be

$$\begin{aligned} \underset{\lambda,\nu}{\text{maximize}} \quad & -\frac{1}{2}Q_D(\lambda,\nu) - C_D \\ \text{subject to} \quad & A_{2\mathcal{I}}^T\lambda + A_{2\mathcal{E}}^T\nu = 0, \quad \lambda \geq 0 \end{aligned} \tag{74}$$

where $\nu \in \mathbb{R}^p$ and $\lambda \in \mathbb{R}^m$ and where

$$\begin{aligned} Q_D(\lambda,\nu) &= \begin{bmatrix}\lambda^T & \nu^T\end{bmatrix} \begin{bmatrix}A_{1\mathcal{I}} \\ A_{1\mathcal{E}}\end{bmatrix} \tilde{H}^{-1} \begin{bmatrix}A_{1\mathcal{I}}^T & A_{1\mathcal{E}}^T\end{bmatrix} \begin{bmatrix}\lambda \\ \nu\end{bmatrix} + \\ &\quad + 2\left(\tilde{f}^T\tilde{H}^{-1}\begin{bmatrix}A_{1\mathcal{I}}^T & A_{1\mathcal{E}}^T\end{bmatrix} + \begin{bmatrix}b_{\mathcal{I}}^T & b_{\mathcal{E}}^T\end{bmatrix}\right)\begin{bmatrix}\lambda \\ \nu\end{bmatrix} \\ C_D &= \frac{1}{2}\tilde{f}^T\tilde{H}^{-1}\tilde{f} \end{aligned} \tag{75}$$

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

$$\begin{aligned} \underset{\lambda,\nu}{\text{minimize}} \quad & \frac{1}{2}Q_D(\lambda,\nu) \\ \text{subject to} \quad & A_{2\mathcal{I}}^T\lambda + A_{2\mathcal{E}}^T\nu = 0, \quad \lambda \geq 0 \end{aligned} \tag{76}$$

For the sake of simplicity, in this report also (76) is denoted the dual problem to (11), even though it is only equivalent to the dual problem.

Remark 3. If the objective function of the primal problem in (11) is purely quadratic ($\tilde{f} = 0$), there will be no constant term C_D in the dual objective. Then, the magnitude of the optimal objective function value in (74) and in (76) coincide.

Strong Duality for Convex Quadratic Programming

Early work on duality for QPs can be found in [14], [15] and [13]. It follows from Theorem 1 in the appendix 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 in (11) and the dual problem in (74) are considered. Note that the primal problem in (11) has an objective function that is bounded from below since $\tilde{H} \in \mathbb{S}^{n_1}_{++}$. It will now be shown that the dual problem has a solution if and only if the primal problem has a solution.

First, a strong alternative result from [10] is needed.

Lemma 1
Given $A \in \mathbb{R}^{p\times n}$ and $b \in \mathbb{R}^p$. The following two systems of inequalities are strong alternatives

1. $\exists x \in \mathbb{R}^n : Ax \leq b$
2. $\exists \lambda \in \mathbb{R}^p : \lambda \geq 0,\ A^T\lambda = 0,\ b^T\lambda < 0$

that is, exactly one of the alternatives holds.

Proof: See [36]. □

In order to reach the final result, the following intermediate result is needed.

Lemma 2
If the primal problem in (11) *is infeasible, then the dual problem in* (74) *is unbounded from above.*

Proof: Consider a QP problem of the type in (11) with only inequality constraints and define $J_D(\lambda)$ to be the dual objective function. The dual problem is always feasible since the origin is always a feasible point. Then $\exists\, \bar{\lambda} : A_{2\mathcal{I}}^T\bar{\lambda} = 0,\ \bar{\lambda} \geq 0$. Further, assume the primal infeasible. Then, from Lemma 1, it follows that $\exists\, \lambda' : \lambda' \geq 0,\ A_{\mathcal{I}}^T\lambda' = 0,\ b_{\mathcal{I}}^T\lambda' < 0$. Note that

$$A_{\mathcal{I}}^T\lambda' = \begin{bmatrix} A_{1\mathcal{I}} & A_{2\mathcal{I}} \end{bmatrix}^T \lambda' = 0 \tag{77}$$

Since $A_{2\mathcal{I}}^T\left(\bar{\lambda} + \alpha\lambda'\right) = 0$, the sum $\bar{\lambda} + \alpha\lambda'$ is dual feasible for every $\alpha \geq 0$. Using that $A_{1\mathcal{I}}^T\lambda' = 0$, it holds that

$$\begin{aligned} J_D(\bar{\lambda} + \alpha\lambda') &= -\frac{1}{2}\bar{\lambda}^T A_{1\mathcal{I}}\tilde{H}^{-1}A_{1\mathcal{I}}^T\bar{\lambda} - \left(\tilde{f}^T\tilde{H}^{-1}A_{1\mathcal{I}}^T + b_{\mathcal{I}}^T\right)\bar{\lambda} - \frac{1}{2}\tilde{f}^T\tilde{H}^{-1}\tilde{f} - \alpha b_{\mathcal{I}}^T\lambda' \\ &= J_D(\bar{\lambda}) - \alpha b_{\mathcal{I}}^T\lambda' \to +\infty,\ \alpha \to +\infty \end{aligned} \tag{78}$$

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

It is now possible to state the main result in this section.

Theorem 3
If the primal problem is feasible, then the primal and dual objective function values coincide. Furthermore, the dual problem in (74) *has a bounded solution if and only if the primal problem in* (11) *is feasible.*

Proof: The first statement follows directly from strong duality for QP, *i.e.*, Theorem 1 in the appendix.

Furthermore, strong duality shows that the dual problem has a bounded solution if the primal problem is feasible, since in that case the primal and dual objective function values coincide and the primal objective function value is bounded from below since $\tilde{H} \in \mathbb{S}^{n_1}_{++}$.

From Lemma 2 it follows that an infeasible primal implies an unbounded dual, which is equivalent to that a bounded dual implies a feasible primal. □

Computing the Gradient From the Dual Variables

Consider the subproblem in (14). If m_{max} in Algorithm 1 is set high enough (or to infinity), then the inner loop breaks only if an optimal point $\left(\lambda^N_{m+1}, \nu^N_{m+1}\right)$ in a working set $\mathcal{W}^N_{m+1}$ has been found, or if an unbounded direction has been found. The latter case is not of interest here, since then the algorithm is terminated. After possibly a reordering of the variables such that $\lambda^N_{1;m+1}$ contains variables in the current working set and such that $\lambda^N_{2;m+1}$ does not, the KKT conditions are in the form

$$\left[\begin{array}{cc:c} H_{11} & H_{12} & -I \\ H_{12}^T & H_{22} & 0 \\ \hdashline -I & 0 & 0 \end{array}\right] \left[\begin{array}{c} \lambda^N_{1;m+1} \\ \lambda^N_{2;m+1} \\ \hdashline \nu^N_{m+1} \end{array}\right] = \left[\begin{array}{c} -f_1 \\ -f_2 \\ \hdashline 0 \end{array}\right] \tag{79}$$

which can be equivalently rewritten in the form

$$\begin{aligned} H_{12}\lambda^N_{2;m+1} - \nu^N_{m+1} &= -f_1 \Leftrightarrow \nu^N_{m+1} = H_{12}\lambda^N_{2;m+1} + f_1 \\ H_{22}\lambda^N_{2;m+1} &= -f_2 \end{aligned} \tag{80}$$

Now, differentiate the objective function with respect to λ^N_{m+1} in a point where the KKT conditions in (80) are satisfied.

$$\left.\frac{\partial Q}{\partial \lambda^N_{m+1}}\right|_{\substack{\lambda^N_{1;m+1}=0, \\ H_{22}\lambda^N_{2;m+1}=-f_2}} = \left.\begin{bmatrix} H_{11}\lambda^N_{1;m+1} + H_{12}\lambda^N_{2;m+1} + f_1 \\ H_{12}^T\lambda^N_{1;m+1} + H_{22}\lambda^N_{2;m+1} + f_2 \end{bmatrix}\right|_{\substack{\lambda^N_{1;m+1}=0, \\ H_{22}\lambda^N_{2;m+1}=-f_2}} = \begin{bmatrix} \nu^N_{m+1} \\ 0 \end{bmatrix} \tag{81}$$

As a consequence, if the dual variables ν^N_{m+1} are computed from the equation in (79), the gradient in that point is also already known. This is the case in the algorithm outlined in Figure 2, where the dual variables always are computed in the end of “Main step 2” (if $m_{max} = \infty$) in order to be able to check dual feasibility . It is therefore straightforward to reuse these computations in the next coming “Main step 1” where a projected line search is performed in the steepest descent direction. However, there are situations where the dual variables are not up-to-date and it is necessary to compute the gradient explicitly. For example, the first iteration is one such situation.

A.3 Definitions of Stacked Matrices

$$
\begin{aligned}
&\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}} = \operatorname{diag}\left(\tilde{Q}_{\tilde{x}}(0),\dots,\tilde{Q}_{\tilde{x}}(N-1)\right),\ \tilde{\mathsf{Q}}_{\mathsf{w}} = \operatorname{diag}\left(\tilde{Q}_{w}(-1),\dots,\tilde{Q}_{w}(N-1)\right)\\
&\tilde{\mathsf{Q}}_{\tilde{\mathsf{x}}\tilde{\mathsf{w}}} = \operatorname{diag}\left(\tilde{Q}_{\tilde{x}w}(0),\dots,\tilde{Q}_{\tilde{x}w}(N-1)\right)\\
&\tilde{\mathsf{q}}_{\tilde{\mathsf{x}}} = \left[0,\dots,\tilde{q}^{T}_{\tilde{x}}(N)\right]^{T},\ \tilde{\mathsf{q}}_{\mathsf{w}} = \left[\tilde{q}^{T}_{w}(-1),\dots,\tilde{q}^{T}_{w}(N-1)\right]^{T}\\
&\tilde{\mathsf{A}} = \begin{bmatrix} -I & 0 & \dots & 0 & 0\\ \tilde{A}(0) & -I & \dots & 0 & 0\\ 0 & \tilde{A}(1) & \ddots & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{A}(N-1) & -I \end{bmatrix},\ \tilde{\mathsf{B}}_{\mathsf{w}} = \begin{bmatrix} \tilde{B}(-1) & 0 & \dots & 0\\ 0 & \tilde{B}(0) & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{B}(N-1) \end{bmatrix},\\
&\tilde{\mathsf{K}} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0\\ \tilde{K}(1) & 0 & \dots & 0 & 0\\ 0 & \tilde{K}(2) & \ddots & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{K}(N) & 0 \end{bmatrix},\ \tilde{\mathsf{G}} = \begin{bmatrix} \tilde{G}(0) & 0 & \dots & 0\\ 0 & \tilde{G}(1) & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{G}(N) \end{bmatrix},\\
&\tilde{\mathsf{P}} = \begin{bmatrix} \tilde{P}(0) & 0 & \dots & 0\\ 0 & \tilde{P}(1) & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{P}(N) \end{bmatrix}
\end{aligned}
\tag{82}
$$

A.4 Invertibility of $\tilde{\mathsf{A}} + \tilde{\mathsf{B}}_{w}\tilde{\mathsf{K}}$

The matrix

$$
\tilde{\mathsf{A}} + \tilde{\mathsf{B}}_{w}\tilde{\mathsf{K}} = \begin{bmatrix} -I & 0 & \dots & 0 & 0\\ \tilde{A}(0)+\tilde{B}(0)\tilde{K}(1) & -I & \dots & 0 & 0\\ 0 & \tilde{A}(1)+\tilde{B}(1)\tilde{K}(2) & \ddots & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots\\ 0 & 0 & \dots & \tilde{A}(N-1)+\tilde{B}(N-1)\tilde{K}(N) & -I \end{bmatrix} \tag{83}
$$

is non-singular. This follows from the fact that it is lower triangular with all diagonal elements nonzero, [23], and it holds for any choice of $\tilde{A}(\tau)$, $\tilde{B}(\tau)$ and $\tilde{K}(\tau)$.

A.5 Detuning of CPLEX

The following integer parameters were modified during the attempt to detune CPLEX in order to get it to work more similar to the *basic* branch and bound algorithm used for the implementation of the algorithm presented in this report. For further description, see the CPLEX manual.

Table 1: *Integer parameters*

CPLEX Integer parameter	Value
1030	0
2037	-1
2064	0
4010	0
2018	0
2028	1
2034	0
2029	0
2042	-1

A.6 Motivation for the Choice of Search Direction in the Inconsistent Case

In this appendix, the case when the Hessian in the quadratic objective function is singular is studied. The purpose with the analysis is to find a search direction that is useful in that case. The optimization problem considered is in the form in (27), which is here repeated for convenience,

$$\underset{x}{\text{minimize}} \quad \tilde{Q}_s(x) = \tfrac{1}{2}x^T\tilde{H}x + \tilde{f}^Tx$$

where $\tilde{H} \in \mathbb{S}_+^n$. According to [31], the minimum of the problem in (27) is attained if and only if $\tilde{H} \succeq 0$ and $\tilde{f} \in \operatorname{range}\tilde{H}$. Since $\tilde{H} \succeq 0$ by assumption, the situation when $\tilde{f} \notin \operatorname{range}\tilde{H}$ is the interesting one and that is the situation that will be studied in this appendix. In this situation, there is no solution to the KKT conditions and the objective function is unbounded from below. The objective with this analysis is to find the direction p along which the objective function decreases most rapidly for stepsizes tending to infinity.

Partition p as $p = p_Z + p_{Z^\perp}$, where $p_Z = \mathcal{P}p$ and $p_{Z^\perp} = (I - \mathcal{P})\,p$, and where $\mathcal{P} = Z\left(Z^TZ\right)^{-1}Z^T$. Z is a matrix which columns form a basis for the nullspace of $\tilde{H}$. Now, consider a step αp starting from the point x_0. The objective function can be written as

$$\begin{aligned}\tilde{Q}_s(x(\alpha)) &= \frac{1}{2}\left(x_0 + \alpha p\right)^T\tilde{H}\left(x_0 + \alpha p\right) + \tilde{f}^T\left(x_0 + \alpha p\right)\\ &= \frac{1}{2}\alpha^2 p_{Z^\perp}^T\tilde{H}p_{Z^\perp} + \alpha f^Tp_Z + \alpha x_0^THp_{Z^\perp} + \alpha f^Tp_{Z^\perp} + K_1\end{aligned} \tag{84}$$

along the ray $\{x | x = x_0 + \alpha p, \alpha \geq 0\}$. K_1 is a constant. Note that, if $p_{Z^\perp}^T\tilde{H}p_{Z^\perp} > 0$ then $\tilde{Q}_s(x(\alpha)) \to +\infty, \alpha \to +\infty$ for any choice of f and K_1. Furthermore, it holds that $p_{Z^\perp}^T\tilde{H}p_{Z^\perp} \succeq 0$ since $\tilde{H} \succeq 0$. Hence, since the objective with this appendix is to find a search direction such that $\tilde{Q}_s(x(\alpha)) \to -\infty, \alpha \to +\infty$, the case when $p_{Z^\perp}^T\tilde{H}p_{Z^\perp} > 0$ is not of further interest. Note that $p_{Z^\perp}^T\tilde{H}p_{Z^\perp} = 0 \Leftrightarrow p_{Z^\perp} = 0$, since $p_{Z^\perp} \in \operatorname{null}(H)^\perp$. Hence, for what follows, $p = p_Z$. The choice of p_Z that gives the best descent direction in the nullspace of the Hessian can be found using a trust region approach by solving the problem

$$\begin{array}{ll} \underset{p_Z}{\text{minimize}} & \tilde{Q}_Z(p_Z) = \frac{1}{2}\left(x_0 + p_Z\right)^T \tilde{H}\left(x_0 + p_Z\right) + f^T\left(x_0 + p_Z\right) \\ \text{subject to} & \|p_Z\| \leq \Delta \\ & p_Z \in \operatorname{null} \tilde{H} \end{array} \tag{85}$$

Note that, since $\tilde{H}\mathcal{P} = 0$, and $\mathcal{P} = \mathcal{P}^2$, it holds that $Q_Z(p_Z) = \left(\mathcal{P}\left(Hx_0 + f\right)\right)^T p_Z + K_2$. For what follows, it is assumed that $\mathcal{P}\left(\tilde{H}x_0 + f\right) \neq 0$, otherwise the objective function is "flat" on the nullspace of $\tilde{H}$ and there will not be possible to find any directions where the original problem is unbounded. Under this assumption, the optimal solution will be on the boundary of the feasible set in (85) (it is an LP). If the constraint $p_Z \in \operatorname{null} \tilde{H}$ is temporarily disregarded, then the optimal solution with norm Δ can be found to be

$$\bar{p}_Z = -\frac{\mathcal{P}\left(\tilde{H}x_0 + f\right)\Delta}{\left\|\mathcal{P}\left(\tilde{H}x_0 + f\right)\right\|} \tag{86}$$

by inspection. From (86), it is also clear that $\bar{p}_Z$ satisfies the last constraint $p_Z \in \operatorname{null} \tilde{H}$ in (85), since $\mathcal{P} = Z\left(Z^TZ\right)^{-1}Z^T$ and the columns in Z form a basis for the nullspace of $\tilde{H}$. Hence, the optimal solution p_Z^* to the problem in (85) is given by (86). When $\Delta \to \infty$, the solutions to the problem in (85) approaches the desirable steps of infinite length for the problem in (27), and hence, it follows that the search direction

$$p = -\frac{\mathcal{P}\left(\tilde{H}x_0 + f\right)}{\left\|\mathcal{P}\left(\tilde{H}x_0 + f\right)\right\|} \tag{87}$$

is the one that provides the fastest descent on the nullspace of the Hessian in the problem in (84).

The search direction presented in (87) can be simplified to

$$\bar{p}_Z = -\frac{\mathcal{P}f}{\|\mathcal{P}f\|} \tag{88}$$

since $\mathcal{P}H = 0$. In this report, the algorithm computes the search direction according to the equation in (87). It would, however, be possible to simplify the computations slightly by instead implementing it as in (88).

References

[1] D. Axehill. *Applications of Integer Quadratic Programming in Control and Communication.* Licentiate's thesis, Linköpings universitet, 2005. URL `http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263`.

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

[3] D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698, Manchester Grand Hyatt, San Diego, USA, Dec. 2006.

[4] 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.

[5] 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.

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

[7] D. P. Bertsekas. On the Goldstein – Levitin – Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174 – 184, Apr. 1976.

[8] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221 – 246, Mar. 1982.

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

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

[11] J. V. Burke and J. J. Moré. Exposing constraints. *SIAM Journal on Optimization*, 4 (3):573 – 595, Aug. 1994.

[12] A. R. Conn, N. I. M. Gould, and P. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433 – 460, Apr. 1988.

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

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

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

[16] 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.

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

[18] E. M. Gafni and D. P. Bertsekas. Convergence of a gradient projection method. Technical Report LIDS-P-1201, Laboratory for Information and Decision Systems Massachusetts Institute of Technology, 1982.

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

[20] A. A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, May 1964.

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

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

[23] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, 1985.

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

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

[26] E. S. Levitin and B. T. Polyak. Constrained minimization problems. *Computational Mathematics and Mathematical Physics*, 6:1 – 50, 1966.

[27] 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.

[28] G. P. McCormick. Anti-zig-zagging by bending. *Management Science*, 15(5):315 – 320, Jan. 1969.

[29] J. J. Moré. Gradient projection techniques for large-scale optimization problems. In *Proceedings of the 28th Conference on Decision and Control*, pages 378 – 381, Dec. 1989.

[30] J. J. Moré and G. Toraldo. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55:377 – 400, 1989.

[31] J. Nocedal and S. J. Wright. *Numerical Optimization – Second Edition.* Springer Verlag, 2006.

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

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

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

[35] 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.

[36] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

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

[38] 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.

[39] 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.

[40] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Paper E

On Relaxations Applicable to Model Predictive Control for Systems with Binary Control Signals

Authors: Daniel Axehill, Lieven Vandenberghe and Anders Hansson

Edited version of the paper: D. Axehill, L. Vandenberghe, and A. Hansson. On relaxations applicable to model predictive control for systems with binary control signals. In *Preprints of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 200–205, Pretoria, South Africa, Aug. 2007

Preliminary version: Published as Technical Report LiTH-ISY-R-2771, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

On Relaxations Applicable to Model Predictive Control for Systems with Binary Control Signals

Daniel Axehill*, Lieven Vandenberghe** and Anders Hansson*

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,hansson}@isy.liu.se

**Dept. of Electrical Engineering
University of California, Los Angeles
Los Angeles, California 90095-1594, USA
vandenbe@ee.ucla.edu

Abstract

In this work, different relaxations applicable to an MPC problem with binary control signals are compared. The relaxations considered are the QP relaxation, the standard SDP relaxation and an alternative equality constrained SDP relaxation. The relaxations are related theoretically, and both the tightness of the bounds and the computational complexities are compared in numerical experiments. The result is that for long prediction horizons, the equality constrained SDP relaxation proposed in this paper provides a good trade-off between the quality of the relaxation and the computational time.

1 Introduction

In recent years the field of application for the popular control strategy Model Predictive Control (MPC) has been broadened in several steps. From the beginning, MPC was only applicable to linearly constrained linear systems. Today, it is possible to use MPC for control of nonlinear systems and hybrid systems. In this work, the focus is on control of hybrid systems. In the basic linear setup, the MPC problem can be cast in the form of a Quadratic Programming (QP) problem. When a hybrid system is to be controlled, binary variables are introduced and the optimization problem is changed from a QP to a Mixed Integer Quadratic Programming (MIQP) problem, which is in general known to be $\mathcal{NP}$-hard, [19]. MPC for hybrid systems is sometimes called Mixed Integer Predictive Control (MIPC). Today, there exist tailored optimization routines with low computational complexity for linear MPC. However, there is still a need for efficient optimization routines for MIPC.

A popular method for solving MIQP problems is branch and bound, where the original integer optimization problem is solved as a sequence of smaller QP subproblems ordered in a tree. For some problems, a large number of QP subproblems have to be solved and the worst case complexity is known to be exponential. The efficiency of the branch and bound method highly depends on the possibility to efficiently compute tight bounds on the optimal objective function value. For MIQP-problems, usually QP relaxations, where integer constraints are relaxed to interval constraints, are solved in the nodes to produce these bounds. Recent research results have shown that a more advanced relaxation built on Semidefinite Programming (SDP) is useful when solving QPs involving binary variables, [4, 9]. For certain instances, this relaxation gives a lower bound with a guaranteed accuracy, [7], and it is expected to give a tighter lower bound than the QP relaxation. A drawback with the SDP relaxation, compared to the QP relaxation, is that the former is more computationally demanding. The SDP relaxations have previously been considered in several contexts and they have successfully been applied to, *e.g.*, the Max Cut problem [7] and the Multiuser Detection problem [6, 14].

In this paper, several relaxations for MIPC are considered, and their corresponding tightness and computational complexity are compared. By considering an alternative equivalent formulation of the original integer problem, a computationally efficient SDP relaxation of the original MIPC problem is found, where the inherent sparseness of the problem is maintained. Similar sparseness has previously been used for large SDP relaxations in, *e.g.*, [11], but it has so far not been used for relaxations of control problems involving binary variables. For a special case, it is shown in numerical experiments that the computational time often can be reduced if this alternative SDP relaxation is used in the nodes in branch and bound. Furthermore, it is investigated how to use the SDP solution to compute suboptimal solutions to the problem at a low computational cost. The results in this paper are shown for unconstrained systems containing only binary control signals. For this case, the optimization problem becomes a special case of an MIQP, *i.e.*, a Binary Quadratic Programming (BQP) problem. However, the results are in principle also applicable in the case of mixed binary valued and real valued control signals.

2 Problem Formulation

In this paper, an MIPC problem for a system in the form

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= Ax(t) + Bu(t),\, t = 0, \dots, N-1 \\ e(t) &= Cx(t) - r(t),\, t = 0, \dots, N \end{aligned} \tag{1}$$

is considered, where $t \in \mathbb{Z}$ is the discrete time, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \{0,1\}^m$ is the control input and $e(t) \in \mathbb{R}^p$ is the control error. The quadratic objective function to be minimized is

$$J = \frac{1}{2}\sum_{t=0}^{N-1} \left(\|e(t)\|_{Q_e}^2 + \|u(t)\|_{Q_u}^2 \right) + \frac{1}{2}\|e(N)\|_{Q_e}^2 \tag{2}$$

where $\|v\|_Q^2 = v^T Q v$, $Q_e \in \mathbb{S}_{++}^p$ and $Q_u \in \mathbb{S}_{++}^m$. For what follows, (1) and (2) are written more compactly using block matrices $\mathsf{Q_e}$, $\mathsf{Q_u}$, A, B, C and block vectors x, u,

e and r, where A is invertible. The Sans Serif font is used to indicate that a matrix or a vector is, in some way, composed by stacked matrices or vectors from different time instants. The stacked matrices or vectors have a similar symbol as the composed matrix, but in an ordinary font. For example, $\mathsf{Q}_\mathsf{u} = \mathrm{diag}(Q_u, \ldots, Q_u)$. Furthermore, a column vector b is introduced, in which the top element is $-x_0$ and all other elements are zero. For details, see [1] (or Appendix B in Part I of this thesis).

Remark 1. Even though a time-invariant description is chosen in the presentation of this work, all results also hold for the case with time-varying system matrices in (1) and time-varying cost matrices in (2).

From an optimization point of view, this problem can be formulated as a BQP problem in, at least, two equivalent ways. First, the equality constraints representing the dynamics of the system, can be kept and the result is a BQP (since x and e are real-valued this formulation is strictly speaking an MIQP, but it is here called a BQP to simplify notation) in the form

$$\begin{array}{ll} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} & J_{\mathrm{BQP}_1}(\mathsf{u}, \mathsf{e}) \\ \text{subject to} & \begin{bmatrix} \mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I \end{bmatrix} \begin{bmatrix} \mathsf{x} \\ \mathsf{u} \\ \mathsf{e} \end{bmatrix} = \begin{bmatrix} b \\ \mathsf{r} \end{bmatrix} \\ & \mathsf{u} \in \{0,1\}^{Nm} \end{array} \tag{3}$$

where $J_{\mathrm{BQP}_1}(\mathsf{u}, \mathsf{e}) = \frac{1}{2}\mathsf{e}^T \mathsf{Q}_\mathsf{e} \mathsf{e} + \frac{1}{2}\mathsf{u}^T \mathsf{Q}_\mathsf{u} \mathsf{u}$. Second, the constraints in (1) can be used to eliminate the states and control errors and the resulting optimization problem can be expressed as a BQP problem equivalent to the problem in (3) in the form

$$\begin{array}{ll} \underset{\mathsf{u}}{\text{minimize}} & J_{\mathrm{BQP}_2}(\mathsf{u}) \\ \text{subject to} & \mathsf{u} \in \{0,1\}^{Nm} \end{array} \tag{4}$$

where $J_{\mathrm{BQP}_2}(\mathsf{u}) = \frac{1}{2}\mathsf{u}^T \left(\mathsf{E}^T \mathsf{Q}_\mathsf{e} \mathsf{E} + \mathsf{Q}_\mathsf{u}\right) \mathsf{u} + \mathsf{e}_0{}^T \mathsf{Q}_\mathsf{e} \mathsf{E} \mathsf{u} + \frac{1}{2}\mathsf{e}_0{}^T \mathsf{Q}_\mathsf{e} \mathsf{e}_0$, $\mathsf{E} = -\mathsf{C}\mathsf{A}^{-1}\mathsf{B}$ and $\mathsf{e}_0 = \mathsf{C}\mathsf{A}^{-1} b - \mathsf{r}$. For what follows, it is important to note that $\mathsf{E}^T \mathsf{Q}_\mathsf{e} \mathsf{E} + \mathsf{Q}_\mathsf{u}$ is dense while Q_e and Q_u are block diagonal. Since the optimal objective function values of the problems in (3) and in (4) coincide, the optimal objective function value is defined as $J^* \triangleq J^*_{\mathrm{BQP}_1} = J^*_{\mathrm{BQP}_2}$.

3 Relaxations

An optimization problem is said to be a relaxation of another optimization problem if the feasible set is larger than the feasible set of the original problem and the objective functions are equivalent in the two problems. In this work, the integer constraints are the difficult constraints and therefore the aim is to replace them by constraints that are easier to work with. As will be shown, this can be done in several different ways, *i.e.*, there are several different possible relaxations of these constraints.

3.1 QP Relaxation

The QP relaxations of the problems in (4) and in (3), can easily be derived by replacing the binary constraints with interval constraints. This means that the QP relaxation of the problem in (3) is

$$\begin{aligned} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & J_{\mathrm{QP}_1}(\mathsf{u},\mathsf{e}) \\ \text{subject to} \quad & \begin{bmatrix} \mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I \end{bmatrix} \begin{bmatrix} \mathsf{x} \\ \mathsf{u} \\ \mathsf{e} \end{bmatrix} = \begin{bmatrix} b \\ \mathsf{r} \end{bmatrix} \\ & 0 \leq \mathsf{u} \leq 1 \end{aligned} \tag{5}$$

where $J_{\mathrm{QP}_1}(\mathsf{u},\mathsf{e}) = J_{\mathrm{BQP}_1}(\mathsf{u},\mathsf{e})$ and the QP relaxation of the problem in (4) is

$$\begin{aligned} \underset{\mathsf{u}}{\text{minimize}} \quad & J_{\mathrm{QP}_2}(\mathsf{u}) \\ \text{subject to} \quad & 0 \leq \mathsf{u} \leq 1 \end{aligned} \tag{6}$$

where $J_{\mathrm{QP}_2}(\mathsf{u}) = J_{\mathrm{BQP}_2}(\mathsf{u})$. These are QP problems corresponding to linear MPC problems. For QP problems of the type in (5), efficient algorithms already exist, [2, 10, 17].

3.2 Standard SDP Relaxation

In this section the SDP relaxation, also known as the moment relaxation, [12, 18], of the problem in (4) is investigated. It can also be found as the dual problem of the dual of the problem in (4). This relaxation is the SDP relaxation which is most commonly used for a BQP problem and can be formulated as

$$\begin{aligned} \underset{U,\mathsf{u}}{\text{minimize}} \quad & J_{\mathrm{SDP}_2}(U,\mathsf{u}) \\ \text{subject to} \quad & U_{ii} = \mathsf{u}_i,\ i = 1,\dots,Nm \\ & \begin{bmatrix} U & \mathsf{u} \\ \mathsf{u}^T & 1 \end{bmatrix} \succeq 0 \end{aligned} \tag{7}$$

where $J_{\mathrm{SDP}_2}(U,\mathsf{u}) = \frac{1}{2}\operatorname{tr}\left(\left(\mathsf{E}^T\mathsf{Q}_\mathsf{e}\mathsf{E} + \mathsf{Q}_\mathsf{u}\right)U\right) + \mathsf{e}_0{}^T\mathsf{Q}_e\mathsf{E}\mathsf{u} + \frac{1}{2}\mathsf{e}_0{}^T\mathsf{Q}_e\mathsf{e}_0$, $\mathsf{u} \in \mathbb{R}^{Nm}$ and $U \in \mathbb{S}_+^{Nm}$. This relaxation has been extensively studied in the literature and there exist bounds on the tightness of the relaxation. The first work was presented for the Max Cut problem in [7], and this result has been refined in several papers, *e.g.*, [16] and [20]. The BQP problem is considered in [15]. Furthermore, there exist randomization methods to derive sub-optimal solutions from the solution to the relaxed problem, [7]. However, a drawback with the standard SDP relaxation is that the number of variables grows fast and, as a consequence, it soon becomes rather computationally demanding.

3.3 Equality Constrained SDP Relaxation

The equality constrained SDP relaxation can be found as the moment relaxation, [12, 18], of (3), or as the dual of the dual problem of (3). The equality constrained SDP relaxation can be written as

$$\begin{aligned}
\underset{U,\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & J_{\mathrm{SDP}_1}(U,\mathsf{x},\mathsf{u},\mathsf{e}) \\
\text{subject to} \quad & U_{ii} = \mathsf{u}_i,\ i = 1,\dots,Nm \\
& \begin{bmatrix} U & \mathsf{u} \\ \mathsf{u}^T & 1 \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} \mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I \end{bmatrix} \begin{bmatrix} \mathsf{x} \\ \mathsf{u} \\ \mathsf{e} \end{bmatrix} = \begin{bmatrix} b \\ \mathsf{r} \end{bmatrix}
\end{aligned} \tag{8}$$

where $J_{\mathrm{SDP}_1}(U,\mathsf{x},\mathsf{u},\mathsf{e}) = \frac{1}{2}\mathsf{e}^T\mathsf{Q}_\mathsf{e}\mathsf{e}+\frac{1}{2}\operatorname{tr}(\mathsf{Q}_\mathsf{u}U)$, $U \in \mathbb{S}_+^{Nm}$, $\mathsf{x} \in \mathbb{R}^{(N+1)n}$ $\mathsf{u} \in \mathbb{R}^{Nm}$ and $\mathsf{e} \in \mathbb{R}^{(N+1)n}$. A very important question is if there is a difference in the bounds provided by the optimal objective function values of the problems in (7) and in (8). This question will be answered in the next section.

3.4 Relations Between the Relaxations

In this section, the relaxations in (5), (6), (7) and in (8) are related. It is rather straightforward to show that the problems in (5) and in (6) are equivalent by eliminating the equality constraints in (5). Hence, their optimal objective function values coincide.

For what follows, two results are needed.

Lemma 1
The following statements are equivalent

$$(i):\ \exists Y_{ij}, i,j = 1,\dots,n, i \neq j: \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1n} & y_1 \\ Y_{12}^T & Y_{22} & \ddots & Y_{2n} & y_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ Y_{1n}^T & Y_{2n}^T & \dots & Y_{nn} & y_n \\ y_1^T & y_2^T & \dots & y_n^T & 1 \end{bmatrix} \succeq 0$$

$$(ii):\ \begin{bmatrix} Y_{11} & y_1 \\ y_1^T & 1 \end{bmatrix} \succeq 0,\ \begin{bmatrix} Y_{22} & y_2 \\ y_2^T & 1 \end{bmatrix} \succeq 0,\ \dots,\ \begin{bmatrix} Y_{nn} & y_n \\ y_n^T & 1 \end{bmatrix} \succeq 0$$

Proof: The lemma is shown for the case $n = 2$ and the result for the general case follows by using the result for that special case repeatedly.

Define $Y = \begin{bmatrix} Y_{11} & Y_{12} & y_1 \\ Y_{12}^T & Y_{22} & y_2 \\ y_1^T & y_2^T & 1 \end{bmatrix}$ and $\tilde{Y} = \begin{bmatrix} Y_{11} - y_1y_1^T & Y_{12} - y_1y_2^T \\ Y_{12}^T - y_2y_1^T & Y_{22} - y_2y_2^T \end{bmatrix}$. Using the Schur complement formula, [5], the following holds

$$Y \succeq 0 \Leftrightarrow \tilde{Y} \succeq 0 \tag{9}$$

Assume that (i) holds. Since the blocks along the diagonal of a positive semidefinite matrix have to be positive semidefinite, $Y \succeq 0$ implies that $Y_{11} - y_1y_1^T \succeq 0$ and $Y_{22} - y_2y_2^T \succeq 0$ by (9). From the Schur complement formula it follows that (ii) holds.

Now, assume that (ii) holds. Then,

$$Y_{11} - y_1y_1^T \succeq 0 \text{ and } Y_{22} - y_2y_2^T \succeq 0 \tag{10}$$

by the Schur complement formula. Let $Y_{12} = y_1 y_2^T$. Then $\tilde{Y}$ is positive semidefinite. Then, by (9), it follows that (i) holds. □

Theorem 1
There exists a $U \in \mathbb{S}_+^n$ such that $u \in \mathbb{R}^n$ satisfies

$$\begin{cases} U_{ii} = u_i, \; i = 1, \ldots, n \\ \begin{bmatrix} U & u \\ u^T & 1 \end{bmatrix} \succeq 0 \end{cases}$$

if and only if u satisfies

$$0 \leq u \leq 1$$

Proof: Obviously, u satisfies $0 \leq u \leq 1$ if and only if $u_i \geq u_i^2$. By the Schur complement formula, this can be equivalently formulated as

$$\begin{bmatrix} u_i & u_i \\ u_i & 1 \end{bmatrix} \succeq 0 \tag{11}$$

or as

$$\begin{cases} U_{ii} = u_i \\ \begin{bmatrix} U_{ii} & u_i \\ u_i & 1 \end{bmatrix} \succeq 0 \end{cases} \tag{12}$$

where a new matrix U has been introduced. By Lemma 1, this statement is equivalent to

$$\exists U : \begin{cases} U_{ii} = u_i \\ \begin{bmatrix} U & u \\ u^T & 1 \end{bmatrix} \succeq 0 \end{cases}$$

and the desired result follows. □

Now, the QP relaxation in (6) is related to the SDP relaxation in (7). Consider the feasible set of the problem in (7). By Theorem 1, all u that are feasible in the SDP relaxation are also feasible in the QP relaxation. Furthermore, the objective functions in (6) and in (7) are not the same. The difference is that $\mathsf{u}^T \left(\mathsf{E}^T \mathsf{Q}_\mathsf{e} \mathsf{E} + \mathsf{Q}_\mathsf{u}\right) \mathsf{u} = \operatorname{tr}\left(\left(\mathsf{E}^T \mathsf{Q}_\mathsf{e} \mathsf{E} + \mathsf{Q}_\mathsf{u}\right) \mathsf{u}\mathsf{u}^T\right)$ in the problem in (6) has been replaced by $\operatorname{tr}\left(\left(\mathsf{E}^T \mathsf{Q}_\mathsf{e} \mathsf{E} + \mathsf{Q}_\mathsf{u}\right) U\right)$ in the problem in (7). However, the positive semidefinite constraint can be written as $U \succeq \mathsf{u}\mathsf{u}^T$, and hence the following relation between the problems in (6) and (7) holds

$$J^*_{\mathrm{QP}_2} \leq J^*_{\mathrm{SDP}_2} \tag{13}$$

Using similar arguments, it can be also be shown that

$$J^*_{\mathrm{QP}_1} \leq J^*_{\mathrm{SDP}_1} \tag{14}$$

The conclusion that follows is that the considered SDP relaxations are as least as tight as the considered QP relaxations. This result is general and holds also for non-MPC BQP problems.

The next step is to relate the optimal objective function values of the problem in (7) and the problem in (8). If the equality constraints for the dynamics and the control error in (8) are eliminated, the result is an equivalent problem in the form

$$\begin{array}{ll} \underset{U,\mathsf{u}}{\text{minimize}} & J_{\mathrm{SDP}_{12}}(U,\mathsf{u}) \\ \text{subject to} & U_{ii} = \mathsf{u}_i,\ i = 1,\dots,Nm \\ & \begin{bmatrix} U & \mathsf{u} \\ \mathsf{u}^T & 1 \end{bmatrix} \succeq 0 \end{array} \tag{15}$$

where $J_{\mathrm{SDP}_{12}}(U,\mathsf{u}) = \frac{1}{2}\mathsf{u}^T\mathsf{E}^T\mathsf{Q}_\mathsf{e}\mathsf{E}\mathsf{u} + \frac{1}{2}\operatorname{tr}(\mathsf{Q}_\mathsf{u}U) + \mathsf{e}_0{}^T\mathsf{Q}_\mathsf{e}\mathsf{E}\mathsf{u} + \frac{1}{2}\mathsf{e}_0{}^T\mathsf{Q}_\mathsf{e}\mathsf{e}_0$. The difference between the problem in (7) and the problem (15) is that the term $\operatorname{tr}\left(\mathsf{E}^T\mathsf{Q}_\mathsf{e}\mathsf{E}U\right)$ in (7) has been replaced by the term $\mathsf{u}^T\mathsf{E}^T\mathsf{Q}_\mathsf{e}\mathsf{E}\mathsf{u} = \operatorname{tr}\left(\mathsf{E}^T\mathsf{Q}_\mathsf{e}\mathsf{E}\mathsf{u}\mathsf{u}^T\right)$ in (15). Since $U - \mathsf{u}\mathsf{u}^T \succeq 0$, and $\mathsf{Q}_\mathsf{e} \succeq 0$, the conclusion is that

$$J^*_{\mathrm{SDP}_1} \leq J^*_{\mathrm{SDP}_2} \tag{16}$$

i.e., the equality constrained SDP relaxation cannot in general be expected to be as tight as the standard SDP relaxation. The complete relation between the different problems is therefore

$$J^*_{\mathrm{QP}_1} = J^*_{\mathrm{QP}_2} \leq J^*_{\mathrm{SDP}_1} \leq J^*_{\mathrm{SDP}_2} \leq J^* \tag{17}$$

4 Reducing Computational Complexity

4.1 Dense Cost Matrix Q_u

The main advantage with the problem in (8) compared to the one in (7) is that the objective function Hessian for the problem in (8) is block diagonal. Specifically, Q_u is block diagonal. Hence, the off-diagonal blocks of U are not used in the objective function. Furthermore, they are not used in any constraint other than the positive semidefinite constraint involving U and u. Therefore, the problem becomes a feasibility problem in the off-diagonal blocks in U, *i.e.*, given the diagonal blocks in U, is it possible to find off-diagonal blocks such that the constraint is satisfied. Then, according to Lemma 1, such off-diagonal blocks making the entire matrix U positive semidefinite can be found if and only if (ii) in Lemma 1 is satisfied. Hence, the feasible set for u and the diagonal blocks in U can equivalently be written as several smaller semidefinite constraints. Therefore, by using Lemma 1, and that Q_x also is block diagonal, the problem in (8) can be written as

$$\begin{array}{ll} \underset{U(t),x(t),u(t),e(t)}{\text{minimize}} & J_{\mathrm{SDP}_1'}(U(t),x(t),u(t),e(t)) \\ \text{subject to} & U_{ii}(t) = u_i(t),\ i \in 1,\dots,m,\ t \in \mathcal{T} \\ & \begin{bmatrix} U(t) & u(t) \\ u^T(t) & 1 \end{bmatrix} \succeq 0,\ t \in \mathcal{T} \\ & x(0) = x_0 \\ & x(t+1) = Ax(t) + Bu(t),\ t \in \mathcal{T} \\ & e(t) = Cx(t) - r(t),\ t \in \mathcal{T} \end{array} \tag{18}$$

where $J_{\mathrm{SDP}_{1'}}(U(t), x(t), u(t), e(t)) = \frac{1}{2}\sum_{t=0}^{N-1}\left(e^T(t)Q_e e(t) + \mathrm{tr}\left(Q_u U(t)\right)\right)$ $+\frac{1}{2}e^T(N)Q_e(N)e(N)$ and $\mathcal{T} = \{0,\ldots,N-1\}$. Here, also the equality constraints have been written out using the corresponding partitioning of u. The dynamics now becomes clearly visible. Note that the number of variables grows linearly in the prediction horizon N.

4.2 Diagonal Cost Matrix Q_u

If Q_u is diagonal, then the SDP relaxation in (18) can be shown to be equivalent to a QP. First, the objective function can be written as $\frac{1}{2}\sum_{t=0}^{N-1}\left(e^T(t)Q_e e(t) + \mathrm{diag}(Q_u)u(t)\right) + \frac{1}{2}e^T(N)Q_e(N)e(N)$ since $U_{ii}(t) = u_i(t)$. Then, by Theorem 1, the positive semidefinite constraint and the constraint $U_{ii}(t) = u_i(t)$ can be replaced by a constraint in the form $0 \leq u \leq 1$. The resulting problem is a QP in the form

$$\begin{array}{cr}\underset{U(t),x(t),u(t),e(t)}{\text{minimize}} & J_{\mathrm{QP}_3}(U(t), x(t), u(t), e(t)) \\ \text{subject to} & 0 \leq u_i(t) \leq 1,\ i \in 1,\ldots,m \\ & x(0) = x_0 \\ & x(t+1) = Ax(t) + Bu(t),\ t \in \mathcal{T} \\ & e(t) = Cx(t) - r(t),\ t \in \mathcal{T}\end{array} \tag{19}$$

where $J_{\mathrm{QP}_3}(U(t), x(t), u(t), e(t)) = \frac{1}{2}\sum_{t=0}^{N-1}\left(e^T(t)Q_e e(t) + \mathrm{diag}(Q_u)u(t)\right)$ $+\frac{1}{2}e^T(N)Q_e(N)e(N)$ and $\mathcal{T} = \{0,\ldots,N-1\}$. Hence, in the diagonal case, the equality constrained SDP relaxation can be computed with similar computational complexity as the QP relaxation.

4.3 Efficient Generation of Suboptimal Solutions

In this section, it is shown how the optimal solution to the problem in (18) can be used to generate suboptimal integer solutions to the problems in (3) and (4). The heuristic method applied is to generate variables from a Gaussian distribution with mean $u(t)$ and variance $U(t) - u(t)u^T(t)$ and thereafter round the result either to 0 or to 1. This is similar to the one presented in [8] for $-1/1$-variables. The procedure is defined by the equations in (20), where $\bar{u}(t)$ is the generated random variable and $\hat{u}(t)$ is the rounded integer solution, where $\mathrm{round}_{0/1}$ represents a function that rounds to 0 or to 1.

$$\begin{aligned}\bar{u}(t) &\in \mathcal{N}\left(u(t), U(t) - u(t)u^T(t)\right) \\ \hat{u}(t) &= \mathrm{round}_{0/1}\left(\bar{u}(t)\right)\end{aligned} \tag{20}$$

Usually, several variables are generated from the distribution and the sample that gives the lowest objective function value is kept as the suboptimal solution. Since the randomization procedure is possible to perform block wise, and there are $N-1$ blocks, the computational complexity grows linearly in the prediction horizon. Note that this also includes the objective function value computation.

5 Numerical Experiments

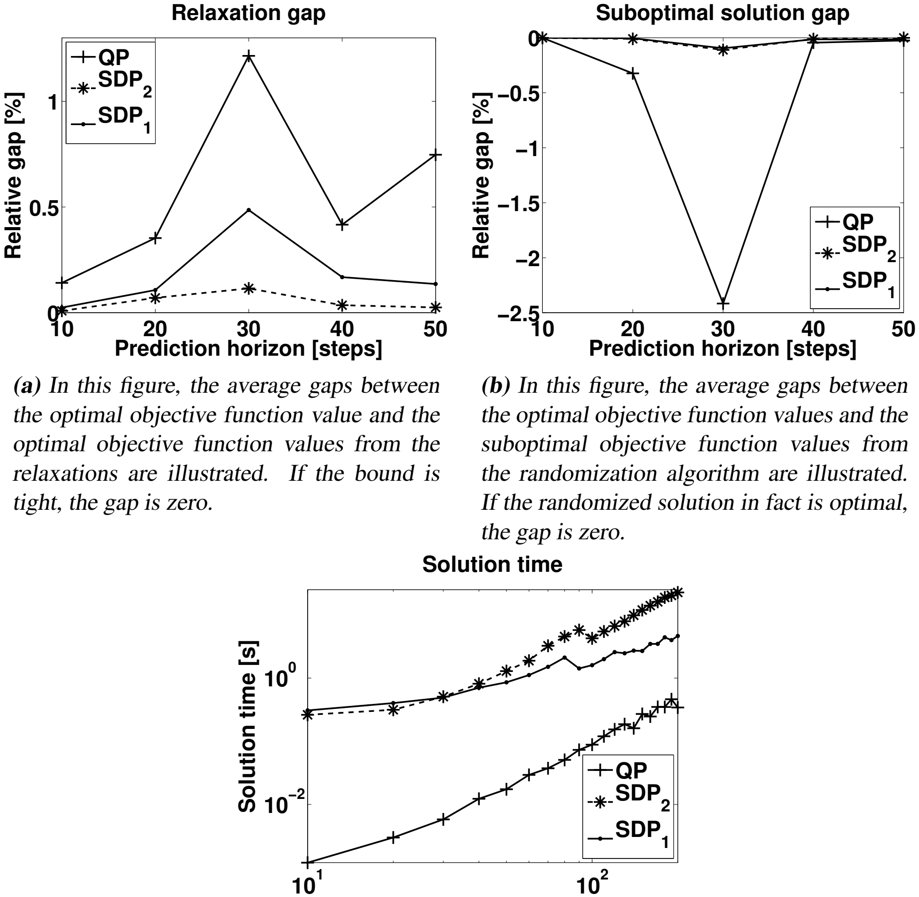
In this section, the relaxations in (6), (7) and (18) are compared in numerical experiments. Furthermore, the relaxation in (19) is compared to the ordinary QP relaxation when used in branch and bound. All tests of the computational times were performed on a computer with two processors of the type Dual Core AMD Opteron 270 sharing 4 GB RAM (the code was not written to utilize multiple cores) running CentOS release 4.4 Kernel 2.6.9-42.0.3.ELsmp and MATLAB 7.2. In the first three experiments, the SDP problems were formulated using YALMIP, [13], and solved by SDPT3, version 4.0. The optimal solution and the QP relaxation were computed by using CPLEX version 10.010. In the fourth experiment, the MIQP problems were solved by *miqp.m*, [3], and the relaxations in the nodes by `quadprog`.

In the first experiment, the respective tightness of the bounds are compared for different prediction horizons. The results are presented in Figure 1a and are found as the average over 10 MPC problems. The systems used in these problems are randomly generated by the MATLAB function `drss` and they contain 2 states, 2 control signals and dense cost matrices Q_x and Q_u. The result from the experiment clearly confirms the theoretical result in (17). It should be mentioned that the tightness is problem dependent, and the quality of the bounds may vary. The result indicates that there are practical control problems where the equality constrained relaxation is useful, *i.e.*, the bound is tighter compared to the QP relaxation. Similar results have been found in the case with diagonal cost matrix Q_u. The practical value in branch and bound of the improved bounds is investigated in experiment four.

In the second experiment, the quality of the randomized suboptimal solutions are compared for the same problems as in the first experiment. The gaps are illustrated in Figure 1b, and is the result after 100 randomizations for each example. The standard SDP relaxation and the equality constrained SDP relaxation generate the best suboptimal solutions. Their quality are similar. The rounded QP relaxation is also considered in the comparison. This approach seems in general to generate worse suboptimal solutions compared to the previous two.

In the third experiment, their respective computational times are compared. The result is presented in Figure 1c and it was found by using the MATLAB command `cputime` and it does not include the time spent in YALMIP. The conclusion is that the SDP relaxations are significantly more demanding to compute compared to the QP relaxation. Furthermore, for short prediction horizons, the standard SDP relaxation and the equality constrained SDP relaxation have similar computational complexities. When the prediction horizons grows, the computational complexity of the standard SDP relaxation grows significantly faster compared to the equality constrained SDP relaxation.

Ideally, it would have been desirable to use the SDP relaxation in all explored nodes in the branch and bound process. In practice this is generally not possible, due to the computational complexity. However, if the cost matrix Q_u is diagonal, the result in Section 4.2 can be used and the equality constrained SDP relaxation can be solved as a QP. In that case, it is actually possible to use the equality constrained SDP relaxation in all nodes and this has been done in the fourth experiment. The result is illustrated in Figure 2. In this experiment, the MIQP solver *miqp.m* was chosen instead of CPLEX, in order to use a “clean” branch and bound code without any preprocessing *etc.*. In Figure 2a, it can be

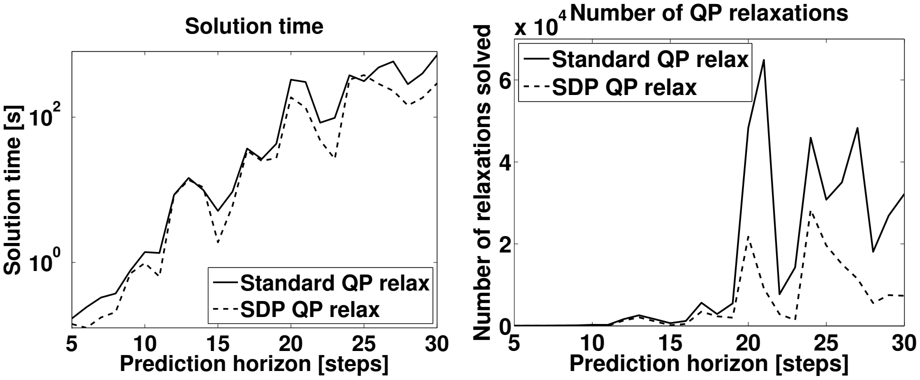


(a) In this figure, the average gaps between the optimal objective function value and the optimal objective function values from the relaxations are illustrated. If the bound is tight, the gap is zero.

(b) In this figure, the average gaps between the optimal objective function values and the suboptimal objective function values from the randomization algorithm are illustrated. If the randomized solution in fact is optimal, the gap is zero.

(c) In this figure, the average computational times for the relaxations are shown.

Figure 1: *Numerical results when the cost matrix Q_u is a dense matrix. For each prediction horizon length, 10 random MPC problems were solved and the presented result is an average over the result from those problems. SDP$_1$ refers to the relaxation in (18), SDP$_2$ to the relaxation in (7), and QP to the relaxation in (6).*



(a) *Comparison of average computational times for solving the BQP problem using branch and bound with different types of relaxations solved in the nodes.*

(b) *Average number of relaxed problems solved before the optimal solution is found by branch and bound.*

Figure 2: *Numerical results when the cost matrix Q_u is a diagonal matrix. For each prediction horizon length,* 10 *random MPC problems were solved and the presented result is an average over the result from those problems. The relaxations used in branch and bound in the comparison are the standard QP relaxation in* (6) *and the equality constrained SDP relaxation computed as a QP from* (19)*. Note that the global integer optimum is computed in this experiment.*

seen that the computational time is usually reduced when the equality constrained SDP relaxation is used. Figure 2b confirms that the number of nodes necessary to solve in branch and bound is significantly reduced if the equality constrained SDP relaxation is used in the nodes. This result indicates that even though the improvement of the bounds seems small in the first experiment, it is actually useful in branch and bound, and can be used to speed up the solution process.

6 Conclusions

In this paper, the QP relaxation, the standard SDP relaxation and an alternative equality constrained SDP relaxation have been applied to an MPC problem with binary control signals and their respective tightness and computational complexity have been compared. The conclusion is that the best lower bound is achieved by the standard SDP relaxation, which is also the most computationally demanding relaxation for longer prediction horizons. Furthermore, the QP relaxation gives the worst lower bound, but is also significantly faster to compute. The equality constrained SDP relaxation gives a bound at least as good as the bound from the QP relaxation and its computational complexity does not grow as fast as the one for the standard SDP relaxation as the prediction horizons grows. Furthermore, if the cost matrix for the control signal Q_u is diagonal, the equality constrained SDP relaxation can be computed as a QP. For this case, the ordinary QP relaxation usually used in branch and bound was successfully replaced by the equality constrained SDP relaxation. As a result, the overall computational time for branch and bound was in average reduced. Finally, it is also shown how suboptimal integer solutions can be generated in a computationally efficient way by using the optimal solution from the equality constrained SDP relaxation. Problems with constraints and the search for an efficient solution to the Newton equations for the relaxations are left as future work.

References

[1] D. Axehill. *Applications of Integer Quadratic Programming in Control and Communication*. Licentiate's thesis, Linköpings universitet, 2005. URL `http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263`.

[2] D. Axehill and A. Hansson. A mixed integer dual quadratic programming algorithm tailored for MPC. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5693–5698, Manchester Grand Hyatt, San Diego, USA, Dec. 2006.

[3] 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.

[4] A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1): 55 – 68, 2007.

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

[6] J. Dahl, B. H. Fleury, and L. Vandenberghe. Approximate maximum-likelihood estimation using semidefinite programming. In *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003*, volume 6, pages VI – 721–724, Apr. 2003.

[7] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing, STOC'94*, pages 422–431, Montréal, Québec, Canada, May 1994.

[8] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115 – 1145, Nov. 1995.

[9] C. Helmberg and F. Rendl. Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291 – 315, Aug. 1998.

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

[11] S. Kim, M. Kojima, and H. Waki. Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, 15(3):697 – 719, 2005.

[12] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796 – 817, 2001.

[13] J. Löfberg. Yalmip: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. URL `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[14] W. K. Ma, T. N. Davidson, K. Wong, Z. Q. Luo, and P. Ching. Quasi-maximum-likelihood multiuser detection using semi-definite relaxation. *IEEE Transactions on Signal Processing*, 50(4):912–922, 2002.

[15] Y. Nesterov. Quality of semidefinite relaxation for nonconvex quadratic optimization. Technical report, CORE, Universite Catholique de Louvain, Belgium, 1997.

[16] Y. Nesterov. Global quadratic optimization via conic relaxation. Technical report, CORE, Universite Catholique de Louvain, Belgium, 1998.

[17] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99 (3):723 – 757, Dec. 1998.

[18] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming – Theory, Algorithms and Applications.* Kluwer, 2000.

[19] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

[20] Y. Ye. Approximating quadratic programming with bound constraints. *Mathematical Programming*, 84:219 – 226, 1999.

Paper F

Relaxations Applicable to Mixed Integer Predictive Control – Comparisons and Efficient Computations

Authors: Daniel Axehill, Anders Hansson and Lieven Vandenberghe

Edited version of the paper: D. Axehill, A. Hansson, and L. Vandenberghe. Relaxations applicable to mixed integer predictive control – comparisons and efficient computations. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4103–4109, Hilton New Orleans Riverside, New Orleans, USA, Dec. 2007

Preliminary version: Published as Technical Report LiTH-ISY-R-2839, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Relaxations Applicable to Mixed Integer Predictive Control – Comparisons and Efficient Computations

Daniel Axehill[*], Anders Hansson[*] and Lieven Vandenberghe[**]

[*]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{daniel,hansson}@isy.liu.se

[**]Dept. of Electrical Engineering
University of California, Los Angeles
Los Angeles, California 90095-1594, USA
vandenbe@ee.ucla.edu

Abstract

In this work, different relaxations applicable to an MPC problem with a mix of real valued and binary valued control signals are compared. In the problem description considered, there are linear inequality constraints on states and control signals. The relaxations are related theoretically and both the tightness of the bounds and the computational complexities are compared in numerical experiments. The relaxations considered are the Quadratic Programming (QP) relaxation, the standard Semidefinite Programming (SDP) relaxation and an equality constrained SDP relaxation. The result is that the standard SDP relaxation is the one that usually gives the best bound and is most computationally demanding, while the QP relaxation is the one that gives the worst bound and is least computationally demanding. The equality constrained relaxation presented in this paper often gives a better bound than the QP relaxation and is less computationally demanding compared to the standard SDP relaxation. Furthermore, it is also shown how the equality constrained SDP relaxation can be efficiently computed by solving the Newton system in an Interior Point algorithm using a Riccati recursion. This makes it possible to compute the equality constrained relaxation with approximately linear computational complexity in the prediction horizon.

1 Introduction

In recent years the field of application of the popular control strategy Model Predictive Control (MPC) has been broadened in several steps. From the beginning, MPC was only applicable to linearly constrained linear systems. Today, it is possible to use MPC for control of nonlinear systems and hybrid systems. In this work, the focus is on control of hybrid systems. In the basic linear setup, the MPC problem can be cast in the form of a Quadratic Programming (QP) problem. When a hybrid system is to be controlled, binary variables are introduced and the optimization problem is changed from a QP to a Mixed Integer Quadratic Programming (MIQP) problem, which is in general known to be $\mathcal{NP}$-hard, [13]. MPC for hybrid systems is sometimes called Mixed Integer Predictive Control (MIPC). Today, there exist tailored optimization routines with low computational complexity for linear MPC. However, there is still a need for efficient optimization routines for MIPC.

A popular method for solving MIQP problems is branch and bound, where the original integer optimization problem is solved as a sequence of smaller QP subproblems. The subproblems are ordered in a tree structure, where one new integer variable is fixed at each level. Depending on the problem, sometimes a large number of QP subproblems have to be solved and the worst case complexity is known to be exponential. The efficiency of the branch and bound method highly relies on the possibility to efficiently compute good bounds on the optimal objective function value. For MIQP-problems, usually QP relaxations (which are often called linear relaxations), where integer constraints are relaxed to interval constraints, are solved in the nodes to produce these bounds. However, recent research has shown that it is possible to use Semidefinite Programming (SDP) in order to compute tighter bounds for the problem. Unfortunately, solving the SDP relaxation is generally much more time consuming than solving the corresponding QP relaxation. Therefore, it is of greatest interest to investigate if it is possible to decrease the computational complexity. In this work this is done by exploiting problem structure. The SDP relaxations have previously been considered in several contexts and they have successfully been applied to, *e.g.*, the Max Cut problem [4] and the Multiuser Detection problem [3, 8].

In a previous work by the authors, *i.e.*, in [2], the different relaxations have been compared for the special case without inequality constraints. In this work, the results are extended to the general case with inequality constraints on states and control signals. Computational results are shown for the general case with mixed real valued and binary valued control signals. Furthermore, it is shown how the proposed equality constrained SDP relaxation can be efficiently solved by an Interior Point (IP) method where the Newton system is solved using a Riccati recursion. The computational performance of the different relaxations are compared and the expected performance gain from the use of Riccati recursions in the SDP solver are illustrated in numerical experiments.

In this paper, $\mathbb{S}^n_{++}$ ($\mathbb{S}^n_{+}$) denotes symmetric positive (semi) definite matrices with n columns, and $\mathbb{R}^n_{++}$ denotes positive real n-vectors. Superscript $*$ is used to denote values of variables and functions at optimum. The set $\mathcal{T} = \{0, \ldots, N-1\}$ is also frequently used. A Sans Serif font is used to indicate that a matrix or a vector is, in some way, composed by stacked matrices or vectors from different time instants. The stacked matrices or vectors have a similar symbol as the composed matrix, but in an ordinary font. For example, $\mathsf{Q}_\mathsf{u} = \operatorname{diag}(Q_u, \ldots, Q_u)$.

2 Introduction to the Control Problem

In this paper, an MIPC problem with a quadratic objective function, or performance measure, in the form

$$J_{\mathrm{MPC}} = \frac{1}{2} \sum_{t=0}^{N-1} \left(\|e(t)\|_{Q_e}^2 + \|u(t)\|_{Q_u}^2 \right) + \frac{1}{2} \|e(N)\|_{Q_e}^2 \tag{1}$$

is considered, where $\|v\|_Q^2 = v^T Q v$, $Q_e \in \mathbb{S}_{++}^p$ and $Q_u \in \mathbb{S}_{++}^m$. The dynamical system to be controlled is in the form

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= Ax(t) + Bu(t) \\ e(t) &= Cx(t) - r(t) \end{aligned} \tag{2}$$

where $t \in \mathbb{Z}$ is the discrete time, $x(t) \in \mathbb{R}^n$ is the state, x_0 the initial state, $u(t)^T = \begin{bmatrix} u_c(t)^T & u_b(t)^T \end{bmatrix}$ is the control input and $e(t) \in \mathbb{R}^p$ is the control error with reference signal $r(t) \in \mathbb{R}^p$. Furthermore, $u_c(t) \in \mathbb{R}^{m_c}$, $u_b(t) \in \{0,1\}^{m_b}$ and $m = m_c + m_b$. Note that the choice of partitioning of the control signal into real valued and binary valued components can be made without any loss of generality. The system is also in each time instant subject to c linear inequality constraints in the form

$$\begin{aligned} H_x x(t) + H_u u(t) + h \leq 0,\ t \in \mathcal{T} \\ H_x x(N) + h \leq 0 \end{aligned} \tag{3}$$

where $H_x \in \mathbb{R}^{c \times n}$, $H_u \in \mathbb{R}^{c \times m}$ and $h \in \mathbb{R}^c$.

Remark 1. Even though a time-invariant description is chosen in the presentation of this work, all results also hold for the case with a time-varying problem description $A(t)$, $B(t)$, $C(t)$, $H_x(t)$, $H_u(t)$, $h(t)$ and $c(t)$.

It is known from, *e.g.*, [2], that this MIPC problem can be written in at least two different, but equivalent, forms. First, the equality constraints representing the dynamics of the system, can be kept and the result is an MIQP in the form

$$\begin{aligned} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} \quad & J_{\mathrm{MIQP}_1}(\mathsf{u},\mathsf{e}) \\ \text{subject to} \quad & \begin{bmatrix} \mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I \end{bmatrix} \begin{bmatrix} \mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T \end{bmatrix}^T = \begin{bmatrix} b \\ \mathsf{r} \end{bmatrix} \\ & \begin{bmatrix} \mathsf{H}_x & \mathsf{H}_u & 0 \end{bmatrix} \begin{bmatrix} \mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T \end{bmatrix}^T + \mathsf{h} \leq 0 \\ & \mathsf{u}_i \in \{0,1\},\ i \in \mathcal{B} \end{aligned} \tag{4}$$

where $\mathsf{x} \in \mathbb{R}^{(N+1)n}$, $\mathsf{e} \in \mathbb{R}^{(N+1)p}$, $\mathsf{u} \in \mathbb{R}^{Nm}$, $J_{\mathrm{MIQP}_1}(\mathsf{u},\mathsf{e}) = \frac{1}{2}\mathsf{e}^T \mathsf{Q}_\mathsf{e} \mathsf{e} + \frac{1}{2}\mathsf{u}^T \mathsf{Q}_\mathsf{u} \mathsf{u}$ and the set $\mathcal{B}$ contains the indices to the binary components in u. Q_e and Q_u are block diagonal matrices with Q_e and Q_u, respectively, along the diagonal. A detailed explanation of the notation can be found in [1, pp. 113–115] (or in Appendix B in Part I of this thesis). Second, the equality constraints in (2) can be used to eliminate the states and control errors and the resulting optimization problem can be expressed as an MIQP problem in the form

$$\begin{array}{ll} \underset{\mathsf{u}}{\text{minimize}} & J_{\mathrm{MIQP}_2}(\mathsf{u}) \\ \text{subject to} & (\mathsf{H}_x S_u + \mathsf{H}_u)\,\mathsf{u} + \mathsf{h} + \mathsf{H}_x S_x x_0 \leq 0 \\ & \mathsf{u}_i \in \{0,1\},\ i \in \mathcal{B} \end{array} \tag{5}$$

which is equivalent to the problem in (4), and where $\mathsf{u} \in \mathbb{R}^{Nm}$, $J_{\mathrm{MIQP}_2}(\mathsf{u}) = \frac{1}{2}\mathsf{u}^T \left(E^T \mathsf{Q}_\mathsf{e} E + \mathsf{Q}_\mathsf{u}\right)\mathsf{u} + e_0^T \mathsf{Q}_\mathsf{e} E \mathsf{u} + \frac{1}{2} e_0^T \mathsf{Q}_\mathsf{e} e_0$, $E = -\mathsf{C}\mathsf{A}^{-1}\mathsf{B}$ and $e_0 = \mathsf{C}\mathsf{A}^{-1} b - \mathsf{r}$, and the notation is similar to the one used in [1]. The main idea used in this paper is the fact that $E^T \mathsf{Q}_\mathsf{e} E + \mathsf{Q}_\mathsf{u}$ is dense while Q_e and Q_u are block diagonal. The optimal objective function values of the problems in (4) and in (5) coincide and the optimal objective function value is defined as $J^* \triangleq J^*_{\mathrm{MIQP}_1} = J^*_{\mathrm{MIQP}_2} = J^*_{\mathrm{MPC}}$.

3 Relaxations

An optimization problem is said to be a relaxation of another optimization problem if the feasible set is larger than the feasible set of the original problem and the objective functions are equivalent in the two problems. In this work, the integer constraints are relaxed in different ways and the result is new convex optimization problems that are much easier to solve.

The QP relaxations of the problems in (4) and (5), can easily be derived by replacing the binary constraints with interval constraints. This means that the QP relaxation of the problem in (4) is

$$\begin{array}{ll} \underset{\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} & J_{\mathrm{QP}_1}(\mathsf{u},\mathsf{e}) \\ \text{subject to} & \begin{bmatrix} \mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I \end{bmatrix} \begin{bmatrix} \mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T \end{bmatrix} = \begin{bmatrix} b \\ \mathsf{r} \end{bmatrix} \\ & \begin{bmatrix} \mathsf{H}_x & \mathsf{H}_u & 0 \end{bmatrix} \begin{bmatrix} \mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T \end{bmatrix} + \mathsf{h} \leq 0 \\ & 0 \leq \mathsf{u}_i \leq 1,\ i \in \mathcal{B} \end{array} \tag{6}$$

where $J_{\mathrm{QP}_1}(\mathsf{u},\mathsf{e}) = J_{\mathrm{MIQP}_1}(\mathsf{u},\mathsf{e})$ and the QP relaxation of the problem in (5) is

$$\begin{array}{ll} \underset{\mathsf{u}}{\text{minimize}} & J_{\mathrm{QP}_2}(\mathsf{u}) \\ \text{subject to} & (\mathsf{H}_x S_u + \mathsf{H}_u)\,\mathsf{u} + \mathsf{h} + \mathsf{H}_x S_x x_0 \leq 0 \\ & 0 \leq \mathsf{u}_i \leq 1,\ i \in \mathcal{B} \end{array} \tag{7}$$

where $J_{\mathrm{QP}_2}(\mathsf{u}) = J_{\mathrm{MIQP}_2}(\mathsf{u})$.

In recent years, the moment relaxation [6, 12] of problems with binary variables has been extensively studied. The moment relaxation of the problem formulation in (4) is

$$\begin{array}{ll}\underset{U,\mathsf{x},\mathsf{u},\mathsf{e}}{\text{minimize}} & J_{\mathrm{SDP}_1}(U,\mathsf{x},\mathsf{u},\mathsf{e}) \\ \text{subject to} & \begin{bmatrix}\mathsf{A} & \mathsf{B} & 0 \\ \mathsf{C} & 0 & -I\end{bmatrix}\begin{bmatrix}\mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T\end{bmatrix} = \begin{bmatrix} b \\ \mathsf{r}\end{bmatrix} \\ & \begin{bmatrix}\mathsf{H}_x & \mathsf{H}_u & 0\end{bmatrix}\begin{bmatrix}\mathsf{x}^T & \mathsf{u}^T & \mathsf{e}^T\end{bmatrix} + \mathsf{h} \leq 0 \\ & U_{ii} = \mathsf{u}_i,\ i \in \mathcal{B} \\ & \begin{bmatrix} U & \mathsf{u} \\ \mathsf{u}^T & 1\end{bmatrix} \succeq 0 \end{array} \tag{8}$$

where $J_{\mathrm{SDP}_1}(U,\mathsf{x},\mathsf{u},\mathsf{e}) = \frac{1}{2}\mathsf{e}^T\mathsf{Q}_\mathsf{e}\mathsf{e} + \frac{1}{2}\operatorname{tr}(\mathsf{Q}_\mathsf{u}U)$, $U \in \mathbb{S}_+^{Nm}$, $\mathsf{x} \in \mathbb{R}^{(N+1)n}$, $\mathsf{u} \in \mathbb{R}^{Nm}$, $\mathsf{e} \in \mathbb{R}^{(N+1)p}$ and where U_{ii} denotes diagonal element i of the matrix U. The relaxation in (8) is in this work referred to as the equality constrained SDP relaxation.

Similarly, the moment relaxation of the problem in (5) can be found to be

$$\begin{array}{ll}\underset{U,\mathsf{u}}{\text{minimize}} & J_{\mathrm{SDP}_2}(U,\mathsf{u}) \\ \text{subject to} & \left(\mathsf{H}_x S_u + \mathsf{H}_u\right)\mathsf{u} + \mathsf{h} + \mathsf{H}_x S_x x_0 \leq 0 \\ & U_{ii} = \mathsf{u}_i,\ i \in \mathcal{B} \\ & \begin{bmatrix} U & \mathsf{u} \\ \mathsf{u}^T & 1\end{bmatrix} \succeq 0 \end{array} \tag{9}$$

where $J_{\mathrm{SDP}_2}(U,\mathsf{u}) = \frac{1}{2}\operatorname{tr}\left(\left(E^T\mathsf{Q}_\mathsf{e}E + \mathsf{Q}_\mathsf{u}\right)U\right) + e_0^T\mathsf{Q}_e E\mathsf{u} + \frac{1}{2}e_0^T\mathsf{Q}_e e_0$, $\mathsf{u} \in \mathbb{R}^{Nm}$ and $U \in \mathbb{S}_+^{Nm}$.

The relation between the optimization problems in (4)–(9) has previously been thoroughly discussed in [2]. The result is

$$J^*_{\mathrm{QP}_1} = J^*_{\mathrm{QP}_2} \leq J^*_{\mathrm{SDP}_1} \leq J^*_{\mathrm{SDP}_2} \leq J^* \tag{10}$$

which holds similarly also in the case with inequality constraints. One important conclusion is that the lower bounds on the optimal objective function value from the SDP relaxation does in general depend on if the dynamics is expressed as equality constraints or if it is eliminated and included in the objective function.

4 Reducing Computational Complexity

4.1 Utilizing Block Structure

The main advantage with the problem formulation in (8) compared to the one in (9) is that the objective function Hessian for the optimization problem in (8) is block diagonal. Specifically, Q_u is block diagonal. This property can be used as described in [2] to rewrite the problem in the equivalent form

$$\begin{aligned}
\underset{U(t),x(t),u(t),e(t)}{\text{minimize}} \quad & J_{\text{SDP}_{1'}}(U(t),x(t),u(t),e(t)) \\
\text{subject to} \quad & U_{ii}(t) = u_i(t),\ i \in \{m_c+1,\ldots,m\} \\
& 0 \preceq \begin{bmatrix} U(t) & u(t) \\ u(t)^T & 1 \end{bmatrix} \\
& x(0) = x_0 \\
& x(t+1) = Ax(t) + Bu(t) \\
& e(t) = Cx(t) + Du(t) - r(t) \\
& e(N) = Cx(N) - r(N) \\
& 0 \geq H_x x(t) + H_u u(t) + h \\
& 0 \geq H_x x(N) + h
\end{aligned} \tag{11}$$

where $t \in \mathcal{T}$ unless stated differently and

$$\begin{aligned}
& J_{\text{SDP}_{1'}}(U(t),x(t),u(t),e(t)) \\
&= \frac{1}{2}\sum_{t=0}^{N-1}\left(e(t)^T Q_e e(t) + \operatorname{tr}\left(Q_u U(t)\right)\right) + \frac{1}{2}e(N)^T Q_e(N) e(N)
\end{aligned}$$

For completeness, a direct term from $u(t)$ to $e(t)$, with coefficient matrix D, has been introduced in the dynamics of the system in (11). With the problem in the form in (11), the dynamics is clearly visible and the large matrix variable U has been replaced by several smaller matrix variables $U(t) \in \mathbb{S}_+^m$. The number of variables (matrix elements) now grows linearly in the prediction horizon N.

Remark 2. If Q_u is diagonal, then the problem in (11) can be formulated as a QP and this makes it tractable to replace the ordinary QP relaxation used in branch and bound with the SDP relaxation in (11) (computed as a QP). This is further described in [2] for the case without inequality constraints.

4.2 Efficient Computation of Search Directions

In this section, it will be shown how the search directions used in an IP solver applied to the problem in (11) can be computed efficiently using Riccati recursions. For notational simplicity, the derivation is performed for the special case with only binary valued control signals, *i.e.*, $m = m_b$. Using the log-determinant barrier function, the associated centering problem to the SDP problem in (11) can be written as

$$\begin{aligned} \underset{U(t),e(t),x(t),u(t)}{\text{minimize}} \quad & J_{\mathrm{c}}\left(U(t), e(t), x(t), u(t), s\right) \\ \text{subject to} \quad & U_{ii}(t) = u_i(t),\ i \in \{1, \dots, m\} \\ & x(0) = x_0 \\ & x(t+1) = Ax(t) + Bu(t) \\ & e(t) = Cx(t) + Du(t) - r(t) \\ & e(N) = Cx(N) - r(N) \\ & w(t) = 1 \end{aligned} \tag{12}$$

where $t \in \mathcal{T}$ unless stated differently and

$$\begin{aligned} & J_{\mathrm{c}}\left(U(t), e(t), x(t), u(t), s\right) = sJ_{\mathrm{SDP}_1{}'}\left(U(t), x(t), u(t), e(t)\right) \\ & + \sum_{t=0}^{N-1} \Big(-\log\det\big(\tilde{U}(t)\big) - \sum_{i=1}^{c} \log\big(-h_i - H_{x,i}x(t) - H_{u,i}u(t)\big)\Big) \\ & - \sum_{i=1}^{c} \log\big(-h_i - H_{x,i}x(N)\big) \end{aligned}$$

and where $s \in \mathbb{R}_{++}$ is the barrier parameter, $U(t) \in \mathbb{S}_+^m$, $u(t) \in \mathbb{R}^m$ and $\tilde{U}(t) = \begin{bmatrix} U(t) & u(t) \\ u(t)^T & w(t) \end{bmatrix}$. After introducing the Lagrangian multipliers $y(t) \in \mathbb{R}^n$, $p(t) \in \mathbb{R}^p$, $\lambda(t) \in \mathbb{R}^m$ and $\nu(t) \in \mathbb{R}$, and after forming the Lagrangian, the KKT conditions are found to be

$$\frac{\partial L}{\partial e(t)} = sQ_e e(t) - p(t) = 0,\ t = 0, \dots, N \tag{13a}$$

$$\frac{\partial L}{\partial x(t)} = -y(t) + A^T y(t+1) + C^T p(t) - \sum_{i=1}^{c} V_{x,i}(x(t), u(t)) = 0 \tag{13b}$$

$$\frac{\partial L}{\partial x(N)} = -y(N) + C^T p(N) - \sum_{i=1}^{c} V_{x,i}(x(N), u(N)) = 0 \tag{13c}$$

$$\frac{\partial L}{\partial U(t)} = \frac{s}{2} Q_u - \left(\tilde{U}(t)^{-1}\right)_{11} + \operatorname{diag}\left(\lambda(t)\right) = 0 \tag{13d}$$

$$\begin{aligned} \frac{\partial L}{\partial u(t)} = & -2\left(\tilde{U}(t)^{-1}\right)_{12} + B^T y(t+1) + D^T p(t) \\ & - \lambda(t) - \sum_{i=1}^{c} V_{u,i}(x(t), u(t)) = 0 \end{aligned} \tag{13e}$$

$$\frac{\partial L}{\partial w(t)} = -\left(\tilde{U}(t)^{-1}\right)_{22} + \nu(t) = 0 \tag{13f}$$

$$x(0) = x_0 \tag{13g}$$

$$x(t+1) = Ax(t) + Bu(t) \tag{13h}$$

$$e(t) = Cx(t) + Du(t) - r(t) \tag{13i}$$

$$e(N) = Cx(N) - r(N) \tag{13j}$$
$$U_{ii}(t) = u_i(t),\ i \in \{1, \dots, m\} \tag{13k}$$
$$w(t) = 1 \tag{13l}$$

where $t \in \mathcal{T}$ unless stated differently and the functions $V_{x,i}$ and $V_{u,i}$ are defined in (30). $(\cdot)_{ij}$ denotes sub block (i, j). In the KKT system in (13), the equations in (13b)–(13f) are nonlinear. The Newton system is found by linearizing the KKT system with respect to $U(t)$, $x(t)$ and $u(t)$ around $U^0(t)$, $x^0(t)$ and $u^0(t)$ respectively. After linearization according to (28), (13d) can approximately be written as

$$\begin{aligned}&\frac{s}{2}Q_u - Z^0(t) + Z^0(t)\Delta U(t)Z^0(t) + z^0(t)\Delta u(t)^T Z^0(t)\\ &+ Z^0(t)\Delta u(t)z^0(t)^T + \operatorname{diag}(\lambda(t)) = 0,\ t \in \mathcal{T}\end{aligned} \tag{14}$$

where $Z^0(t) \in \mathbb{S}^m_{++}$ and $z^0(t) \in \mathbb{R}^m$ are defined in (29). By multiplying the equations in (14) from left and right by $Z^0(t)^{-1}$, $\Delta U(t)$ can be found as

$$\begin{aligned}\Delta U(t) &= Z^0(t)^{-1} - \frac{s}{2}Z^0(t)^{-1}Q_u Z^0(t)^{-1} - Z^0(t)^{-1}z^0(t)\Delta u(t)^T\\ &- \Delta u(t)z^0(t)^T Z^0(t)^{-1} - Z^0(t)^{-1}\operatorname{diag}(\lambda(t))Z^0(t)^{-1},\ t \in \mathcal{T}\end{aligned} \tag{15}$$

By considering the diagonal of (15), and using the equations in (13k), (31) and (32), the following result is obtained

$$\begin{aligned}\Delta u(t) &= -2\operatorname{diag}\left(Z^0(t)^{-1}z^0(t)\right)\Delta u(t) - \bar{Z}^0(t)\lambda(t)\\ &+ \operatorname{diag}\left(Z^0(t)^{-1} - \frac{s}{2}Z^0(t)^{-1}Q_u Z^0(t)^{-1}\right)\end{aligned} \tag{16}$$

By using (16) and that $\bar{Z}^0(t) \succ 0$ (see Appendix), $\lambda(t)$ can be found as

$$\begin{aligned}\lambda(t) &= \bar{Z}^0(t)^{-1}\Big\{ - \Big(I + 2\operatorname{diag}\left(Z^0(t)^{-1}z^0(t)\right)\Big)\Delta u(t)\\ &+ \operatorname{diag}\left(Z^0(t)^{-1} - \frac{s}{2}Z^0(t)^{-1}Q_u Z^0(t)^{-1}\right)\Big\}\end{aligned} \tag{17}$$

After linearization according to (28) of the equation in (13e), the result is

$$\begin{aligned}&2\left(z^0(t)z^0(t)^T + v^0(t)Z^0(t)\right)\Delta u(t) + B^T y(t+1) + D^T p(t) - \lambda(t)\\ &+ 2Z^0(t)\Delta U(t)z^0(t) - 2z^0(t) + \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t))V_{u,i}(x^0(t), u^0(t))^T\Delta u(t)\\ &= \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t)),\ t \in \mathcal{T}\end{aligned} \tag{18}$$

where the equations in (29), (33) and (34) have been used, and where $v^0(t) \in \mathbb{R}_{++}$ is defined in (29). By using the expression for $\Delta U(t)$ in (15), the equation in (18) can be written as

$$\begin{aligned}
&2\left(v^0(t) - z^0(t)^T Z^0(t)^{-1} z^0(t)\right) Z^0(t)\Delta u(t) + B^T y(t+1) + D^T p(t) - \lambda(t) \\
&- 2\,\mathrm{diag}(\lambda(t)) Z^0(t)^{-1} z^0(t) + \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t)) V_{u,i}(x^0(t), u^0(t))^T \Delta u(t) \\
&= s Q_u Z^0(t)^{-1} z^0(t) + \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t))
\end{aligned} \tag{19}$$

By inserting (17) into (19), an expression in the following form will be obtained

$$R(t)\Delta u(t) + B^T y(t+1) + D^T p(t) = l(t),\ t \in \mathcal{T} \tag{20}$$

where $R(t)$ and $l(t)$ are defined as

$$\begin{aligned}
R(t) &= 2\left(v^0(t) - z^0(t)^T Z^0(t)^{-1} z^0(t)\right) Z^0(t) + \hat{Z}^0(t)\bar{Z}^0(t)^{-1}\hat{Z}^0(t) \\
&\quad + \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t)) V_{u,i}(x^0(t), u^0(t))^T \\
l(t) &= s Q_u Z^0(t)^{-1} z^0(t) + \hat{Z}^0(t) \cdot \bar{Z}^0(t)^{-1} \cdot \mathrm{diag}\left(Z^0(t)^{-1} - \frac{s}{2} Z^0(t)^{-1} Q_u Z^0(t)^{-1}\right) \\
&\quad + \sum_{i=1}^{c} V_{u,i}(x^0(t), u^0(t))
\end{aligned} \tag{21}$$

where $\hat{Z}^0(t) = \left(I + 2\,\mathrm{diag}\left(Z^0(t)^{-1} z^0(t)\right)\right)$. For what follows, it is important to show that $R(t) \succ 0$. Since $U^0(t)$ and $u^0(t)$ are interior points that satisfy the positive semidefinite constraints in the problem strictly, it follows from the definition in (29) that $Z^0(t) \succ 0$ and $\tilde{Z}^0(t) \succ 0$. Therefore, it follows from the Schur complement formula that

$$\tilde{Z}^0(t) \succ 0 \Leftrightarrow v^0(t) - z^0(t)^T Z^0(t)^{-1} z^0(t) \succ 0 \tag{22}$$

which implies that the first term in $R(t)$ in (21) is positive definite. Since the other terms are quadratic and $\bar{Z}^0(t) \succ 0$, the sum is positive definite. Hence, $R(t)$ is positive definite.

Now, by using $x(t) = x^0(t) + \Delta x(t)$ and $u(t) = u^0(t) + \Delta u(t)$, the equations in (13h) and (13i) can be expressed in $\Delta x(t)$ and $\Delta u(t)$ as follows

$$\begin{aligned}
\Delta x(t+1) &= A\Delta x(t) + B\Delta u(t) - x^0(t+1) + Ax^0(t) + Bu^0(t) \\
e(t) &= C\Delta x(t) + D\Delta u(t) - r(t) + Cx^0(t) + Du^0(t)
\end{aligned} \tag{23}$$

If the equations in (13a), (13b), (20) and (23) are collected blockwise, the result is

$$\begin{aligned}
&\begin{bmatrix} 0 & -I & D & C & 0 \\ -I & sQ_e & 0 & 0 & 0 \\ D^T & 0 & R(t) & 0 & 0 \\ C^T & 0 & 0 & \sum_i V_{x,i}(x^0(t),u^0(t))V_{x,i}(x^0(t),u^0(t))^T & -I \\ 0 & 0 & B & A & 0 \end{bmatrix} \begin{bmatrix} p(t) \\ e(t) \\ \Delta u(t) \\ \Delta x(t) \\ y(t) \end{bmatrix} \\
&+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B^T & 0 \\ A^T & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} y(t+1) \\ \Delta x(t+1) \end{bmatrix} = \begin{bmatrix} r(t) - Cx^0(t) - Du^0(t) \\ 0 \\ l(t) \\ \sum_i V_{x,i}(x^0(t),u^0(t)) \\ x^0(t+1) - Ax^0(t) - Bu^0(t) \end{bmatrix}, \; t \in \mathcal{T}
\end{aligned} \tag{24}$$

where the sums over i range from 1 to c. Eliminating $e(t)$ and $p(t)$ results in

$$\begin{bmatrix} -I & Q_1(t) & K(t) \\ 0 & K(t)^T & Q_2(t) \\ 0 & A & B \end{bmatrix} \begin{bmatrix} y(t) \\ \Delta x(t) \\ \Delta u(t) \end{bmatrix} + \begin{bmatrix} A^T & 0 \\ B^T & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} y(t+1) \\ \Delta x(t+1) \end{bmatrix} = \begin{bmatrix} \bar{q}(t) \\ \bar{r}(t) \\ \bar{b}(t+1) \end{bmatrix}, \; t \in \mathcal{T} \tag{25}$$

where $Q_1(t)$, $Q_2(t)$, $K(t)$, $\bar{q}(t)$, $\bar{r}(t)$ and $\bar{b}(t)$ are defined in (35). Furthermore, at time instants $t = 0$ and $t = N$ the following equations hold

$$\begin{aligned} x(0) &= x_0, \\ -y(N) + Q_1(N)\Delta x(N) &= \bar{q}(N) \end{aligned} \tag{26}$$

Since $R(t) \succ 0$, the system defined by the equations in (25) and (26) are in a form which can be solved very efficiently using Riccati recursions. A derivation of the Riccati recursions can be found in, *e.g.*, [10, 14], and the resulting recursions can be found in Algorithm 1.

Finally, by using (28), the linearized version of the equation in (13f) is found to be

$$-v^0(t) + z^0(t)^T \Delta U(t) z^0(t) + v^0(t)\Delta u(t)^T z^0(t) + z^0(t)^T \Delta u(t) v^0(t) + \nu(t) = 0 \tag{27}$$

As a summary, all computations necessary to solve the Newton system efficiently are presented as Algorithm 2. All computations, including the Riccati recursion, can be performed with linear computational complexity in the prediction horizon. In each iteration in an IP method, usually one or two Newton (like) systems of equations have to be solved. Since in practice the number of iterations is roughly independent of problem size, the overall cost of solving the SDP is roughly proportional to the cost of solving the Newton equations, [11]. Consequently, by using the method presented in this section, the optimization problem in (11) can be solved with approximately linear computational complexity in the prediction horizon. It should be mentioned that even though the proposed Riccati method for simplicity is illustrated on a basic barrier method, it is also applicable to a primal-dual IP method as the one described in [11].

5 Numerical Experiments

In this section, the relaxations in (6), (9) and (11) are compared in numerical experiments. Furthermore, it is shown how large performance gain, compared to a generic solver, that

Algorithm 1 Riccati recursion

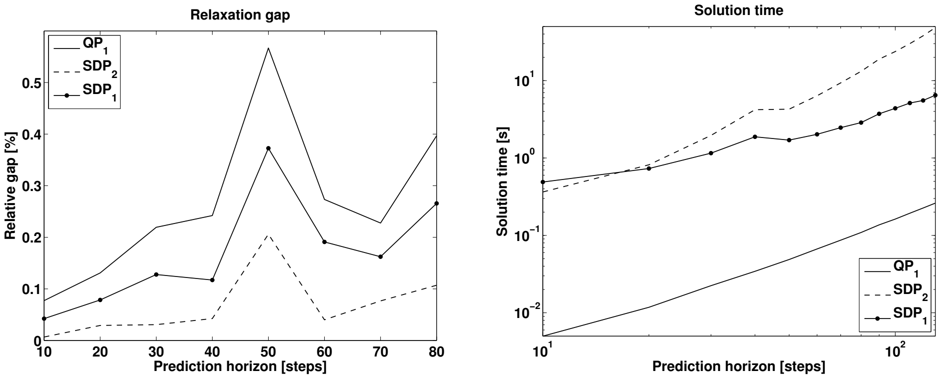
```
$P(N) = Q_1(N)$
$\Psi(N) = \tilde{q}(N)$
for $t = 1$ to $N-1$ do
    $G(t+1) = Q_2(t) + B^T P(t+1) B$
    $H(t+1) = K(t) + A^T P(t+1) B$
    $L(t+1) = G(t+1)^{-1} H(t+1)^T$
    $P(t) = Q_1(t) + A^T P(t+1) A - H(t+1) L(t+1)$
    $\Psi(t) = \tilde{q}(t) - L(t+1)^T \left(B(t)^T \Psi(t+1) + \tilde{r}(t)\right) + A^T \Psi(t+1)$
end for
$\tilde{x}(0) = -\tilde{b}(0)$
for $t = 1$ to $N-1$ do
    $\tilde{u}(t) = -L(t+1)\tilde{x}(t) + G(t+1)^{-1} \left(B^T \Psi(t+1) + \tilde{r}(t)\right)$
    $\tilde{x}(t+1) = A\tilde{x}(t) + B\tilde{u}(t) - \tilde{b}(t+1)$
    $y(t) = -\Psi(t) + P(t)\tilde{x}(t)$
end for
$y(N) = -\Psi(N) + Q_1(N)\tilde{x}(N)$
```

Algorithm 2 Efficient solution of the Newton system

```
Precompute $Z^0$, $\left(Z^0\right)^{-1}$, $\bar{Z}^0$, $\left(\bar{Z}^0\right)^{-1}$, $z^0$, $v^0$, $V_x$, $V_u$, $Q_1$, $Q_2$, $K$, $\tilde{q}$, $\tilde{r}$ and $\tilde{b}$ for all
    time instants necessary.
Compute $\Delta x(t)$, $\Delta u(t)$ and $y(t)$ using Algorithm 1 with the definitions in (35).
Compute $e(t)$ according to (23) and $p(t)$ according to (13a).
Compute $\lambda(t)$ according to (17).
Compute $\Delta U(t)$ according to (15).
Compute $w(t)$ according to (13l).
Compute $\nu(t)$ according to (27).
```

can be expected when the problem in (11) is solved efficiently with an IP solver that solves the Newton system using Riccati recursions. All tests of the computational times were performed on a computer with two processors of the type Dual Core AMD Opteron 270 sharing 4 GB RAM (the code was not written to utilize multiple cores) running CentOS release 4.4 Kernel 2.6.9-42.0.3.ELsmp and MATLAB 7.2. In all experiments with SDP problems, the dual SDP problems were derived manually and given to YALMIP, [7], and they were finally solved by SDPT3 version 4.0, [9]. The optimal solution (MIQP) and the QP relaxation were computed by using CPLEX version 10.010.

In the first experiment, the relative gaps of the different relaxations are compared for different prediction horizons. The relative gap is here defined as $\frac{J^* - J_R^*}{J^*}$, where J_R^* is replaced by the optimal objective function value of the relaxation of interest. The results are presented in Figure 1a and are for each prediction horizon found as the average of 5 problems generated by the MATLAB function `drss` with 4 states, 2 real valued control signals, 2 binary valued control signals and full cost matrices Q_x and Q_u. The two real valued control signals were constrained by a random upper and random lower bound,



(a) This figure shows the average relative gaps between the optimal objective function value and the optimal objective function values from the relaxations.

(b) In this figure, the average computational times for the relaxations are shown.

Figure 1: *The figures present the numerical results illustrating the relative tightness of the relaxations in* (6), (9) *and* (11) *and the corresponding computational time.*

chosen such that the problems are feasible and such that the constraints are "reasonable active" along the prediction horizon. The result from the experiment clearly confirms the theoretical result in (10). It should be mentioned that the tightness is problem dependent, and the quality of the bounds may vary. The result shows that there are practical control problems where the equality constrained relaxation is useful, *i.e.*, the bound is tighter compared to the QP relaxation and the computational time is less than for the ordinary SDP relaxation.

Note that all approaches actually seem to produce rather good bounds, and that the practical difference seems fairly small in the examples considered. In [2], it has been shown in simulations that rather small improvements in the bounds can actually cut down the branch and bound tree significantly and that this, at least in the case with diagonal Q_u, can decrease the computational time.

In the second experiment, the respective computational times for the different relaxations are compared. The result is presented in Figure 1b and it was found by using the MATLAB command `cputime` and it does not include the time spent in YALMIP. The conclusion is that the standard SDP relaxation is rather slow to compute. The equality constrained SDP relaxation is much less computationally demanding for large values of N and the complexity grows significantly slower as N grows. The QP relaxation is the least computationally demanding relaxation to compute. Note that, if Q_u is diagonal, the result in Remark 2 applies and it is then possible to compute the equality constrained SDP relaxation to a computational cost similar to the one of the QP relaxation.

In the third experiment, the computational times for solving the Newton system are investigated. As previously mentioned, a system of equations of this type has to be solved in each iteration in an IP method and the solution of this system is the major cost in the solution process. In the results in Figure 1b, the generic state-of-the-art SDP solver

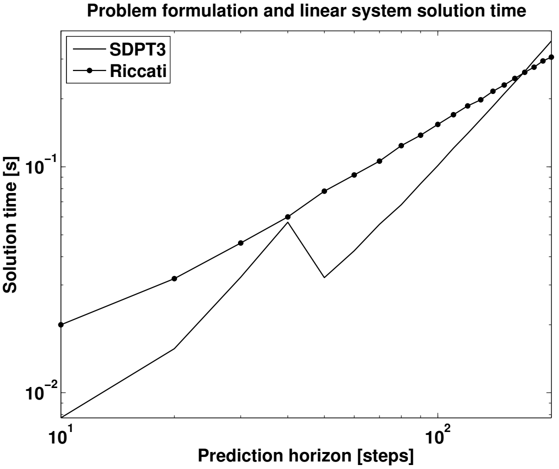


Figure 2: *The result in this figure compares the computational time for solving the Newton (like) system in the predictor step in* SDPT3 *for a problem of the type in* (11) *and the computational time for solving a similar Newton system using the Riccati approach presented in this paper for a problem with similar properties and size. The computational complexity for large values of N grows as $\mathcal{O}(N^{1.9})$ and $\mathcal{O}(N)$ respectively.*

SDPT3 is used to solve the equality constrained SDP relaxation. The purpose with this third experiment is to show that it would be possible to improve the computational times presented in Figure 1b for the equality constrained SDP relaxation by solving the Newton like equation systems in the IP solver using the Riccati approach in Algorithm 2. The expected performance is evaluated by comparing the computational time of certain parts of the code in the primal-dual IP solver SDPT3 with the computational time to perform a similar operation using the Riccati approach in Algorithm 2. The time presented for SDPT3 is the time it takes to form the Schur complement system, [9], and to solve it (similar to solving a Newton system). This is compared to the time it takes to (implicitly) form and solve a similar Newton system using the Riccati approach for a problem of the same size. Each Newton system is solved 10 times and the presented computational time is an average from these 10 experiments. From Figure 2, it can be found that for large values of N, SDPT3 and the Riccati approach solve the system with computational complexities of about $\mathcal{O}(N^{1.9})$ and $\mathcal{O}(N)$ respectively. The absolute computational time is not of major interest here since the Riccati approach is implemented in m-code, while the corresponding operations in SDPT3 are implemented in fast C-code. Our numerical experiments have shown that the number of iterations used in an IP solver applied to the problem in (11) is, as expected, roughly independent of the prediction horizon. In our tests, around 15 iterations were needed to solve the problem. Hence, the overall computational complexity is approximately $\mathcal{O}(N^{1.9})$ for the state-of-the-art generic solver SDPT3 but can be reduced to approximately $\mathcal{O}(N)$ if the Riccati approach presented in this paper is applied.

Acknowledgments

Johan Löfberg is acknowledged for help with YALMIP. The research has been supported by the Swedish Research Council for Engineering Sciences under contract Nr. 621-2002-3822.

6 Conclusions

In this paper, the QP relaxation, the standard moment relaxation and an equality constrained moment relaxation have been applied to an MPC problem with mixed real valued and binary valued control signals subject to linear inequality constraints on states and control signals. The respective tightness and computational complexity of the relaxations have been compared. The conclusion is that the best lower bound is achieved by the standard moment relaxation, which is also the most computationally demanding relaxation. Furthermore, the QP relaxation gives the worst lower bound, but is also the least computationally demanding relaxation. The equality constrained moment relaxation presented in this paper gives a bound at least as good as the bound from the QP relaxation and it is less computationally demanding compared to the standard moment relaxation. This is mainly because of the fact that the number of variables is less and it grows linearly in the prediction horizon. In the special case that the cost matrix for the control signal Q_u is diagonal, the equality constrained SDP relaxation can be formulated and solved as a QP. Furthermore, the computational complexity for computing the solution to the equality constrained SDP relaxation is expected to be possible to reduce further by solving the Newton equations in an IP solver by using Riccati recursions. This approach has been described in detail and promising results from numerical experiments have been presented. A complete implementation of a solver using the proposed Riccati approach and efficient computation of the solution to the standard SDP relaxation are left as future work.

A Appendix

In this appendix some constants are defined and some help equations are presented. Recall that $\tilde{U}(t) = \begin{bmatrix} U(t) & u(t) \\ u(t)^T & w(t) \end{bmatrix}$ and let $U(t) = U^0(t) + \Delta U(t)$, $u(t) = u^0(t) + \Delta u(t)$ and $w(t) = 1$. For small deviations $\Delta U(t)$ and $\Delta u(t)$ from $U^0(t)$ and $u^0(t)$, respectively, and if time indices are neglected, the following result can be found from the Taylor series expansion

$$\begin{aligned} \left(\tilde{U}^{-1}\right)_{11} &\approx Z^0 - Z^0 \Delta U Z^0 - z^0 \Delta u^T Z^0 - Z^0 \Delta u {z^0}^T \\ \left(\tilde{U}^{-1}\right)_{12} &\approx z^0 - Z^0 \Delta U z^0 - z^0 \Delta u^T z^0 - Z^0 \Delta u v^0 \\ \left(\tilde{U}^{-1}\right)_{22} &\approx v^0 - {z^0}^T \Delta U z^0 - v^0 \Delta u^T z^0 - {z^0}^T \Delta u v^0 \end{aligned} \tag{28}$$

where the constants $Z^0(t)$, $z^0(t)$ and $v^0(t)$ are defined as

$$\tilde{Z}^0(t) = \begin{bmatrix} Z^0(t) & z^0(t) \\ z^0(t)^T & v^0(t) \end{bmatrix} = \begin{bmatrix} U^0(t) & u^0(t) \\ u^0(t)^T & 1 \end{bmatrix}^{-1} \tag{29}$$

Note that the inverse in (29) always exists since the IP solver generates a sequence of $U^0(t)$ and $u^0(t)$ that always satisfies the positive semidefinite constraint in the problem strictly. Component i of the vector valued functions V_x and V_u respectively are defined as

$$\begin{aligned} V_{x,i}(x(t), u(t)) &= \frac{H_{x,i}^T}{h_i + H_{x,i}x(t) + H_{u,i}u(t)}, \ t \in \mathcal{T} \\ V_{x,i}(x(N), u(N)) &= \frac{H_{x,i}^T}{h_i + H_{x,i}x(N)} \\ V_{u,i}(x(t), u(t)) &= \frac{H_{u,i}^T}{h_i + H_{x,i}x(t) + H_{u,i}u(t)}, \ t \in \mathcal{T} \end{aligned} \tag{30}$$

where $H_{x,i}$ and $H_{u,i}$ denote row i in H_x and H_u, respectively. The following equalities hold

$$\begin{aligned} &\operatorname{diag}\left(Z^0(t)^{-1}z^0(t)\Delta u(t)^T + \Delta u(t)z^0(t)^T Z^0(t)^{-1})\right) \\ &= 2\operatorname{diag}\left(Z^0(t)^{-1}z^0(t)\right)\Delta u(t) \end{aligned} \tag{31}$$

$$\operatorname{diag}\left(Z^0(t)^{-1}\operatorname{diag}(\lambda(t))Z^0(t)^{-1}\right) = Z^0(t)^{-1} \circ Z^0(t)^{-1}\lambda(t) \triangleq \bar{Z}^0(t)\lambda(t) \tag{32}$$

where the constant $\bar{Z}^0(t)$ has been defined and $\circ$ is the Hadamard (elementwise) product. Note that $\bar{Z}^0(t) \succ 0$ since $Z^0(t)^{-1} \succ 0$, [5, p. 458]. Furthermore,

$$z^0(t)\Delta u(t)^T z^0(t) = z^0(t)z^0(t)^T \Delta u(t) \tag{33}$$

since $\Delta u(t)^T z^0(t)$ is a scalar.

$$Z^0(t)\Delta u(t)v^0(t) = v^0(t)Z^0(t)\Delta u(t) \tag{34}$$

since $v^0(t)$ is a scalar.

The following definitions are used in order to use Algorithm 2 to solve the Newton system.

$$\begin{aligned} &\bar{b}(0) = x_0, \quad \bar{b}(t) = x^0(t+1) - Ax^0(t) - Bu^0(t) \\ &\tilde{b}(0) = \bar{b}(0), \quad \tilde{b}(t) = \bar{b}(t) \\ &\bar{q}(t) = sC^T Q_e\left(r(t) - Cx^0(t) - Du^0(t)\right) + \sum_{i=1}^{c} V_{x,i}(x^0(t), u^0(t)) \\ &\tilde{q}(t) = \bar{q}(t) + A^T(t)P(t+1)\bar{b}(t+1), \quad \tilde{q}(N) = \bar{q}(N) \\ &\bar{r}(t) = l(t) + sD^T Q_e\left(r(t) - Cx^0(t) - Du^0(t)\right) \\ &\tilde{r}(t) = \bar{r}(t) + B^T(t)P(t+1)\bar{b}(t+1) \\ &\tilde{x}(t) = \Delta x(t), \quad \tilde{u}(t) = \Delta u(t) \\ Q_1(t) &= sC^T Q_e C + \sum_{i=1}^{c} V_{x,i}(x^0(t), u^0(t))V_{x,i}(x^0(t), u^0(t))^T \\ Q_2(t) &= R(t) + sD^T Q_e D, \quad K(t) = sC^T Q_e D \end{aligned} \tag{35}$$

References

[1] D. Axehill. *Applications of Integer Quadratic Programming in Control and Communication.* Licentiate's thesis, Linköpings universitet, 2005. URL `http://www.diva-portal.org/liu/theses/abstract.xsql?dbid=5263`.

[2] D. Axehill, L. Vandenberghe, and A. Hansson. On relaxations applicable to model predictive control for systems with binary control signals. Technical Report LiTH-ISY-R-2771, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Jan. 2007. URL `http://www.rt.isy.liu.se/publications/doc?id=1928`.

[3] J. Dahl, B. H. Fleury, and L. Vandenberghe. Approximate maximum-likelihood estimation using semidefinite programming. In *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003*, volume 6, pages VI – 721–724, Apr. 2003.

[4] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing, STOC'94*, pages 422–431, Montréal, Québec, Canada, May 1994.

[5] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, 1985.

[6] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796 – 817, 2001.

[7] J. Löfberg. Yalmip: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. URL `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[8] W. K. Ma, T. N. Davidson, K. Wong, Z. Q. Luo, and P. Ching. Quasi-maximum-likelihood multiuser detection using semi-definite relaxation. *IEEE Transactions on Signal Processing*, 50(4):912–922, 2002.

[9] K. C. Toh, R. Tütüncü, and M. J. Todd. On the implementation and usage of SDPT3 – a MATLAB software package for semidefinite-quadratic-linear programming, version 4.0. Technical report, Department of Mathematics, National University of Singapore, July 2006.

[10] 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.

[11] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. *Interior-point algorithms for semidefinite programming problems derived from the KYP lemma*, volume 312 of *Lecture notes in control and information sciences*, pages 195 – 238. Springer, Feb. 2005.

[12] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming – Theory, Algorithms and Applications.* Kluwer, 2000.

[13] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

[14] M. Åkerblad. *A Second Order Cone Programming Algorithm for Model Predictive Control.* Licentiate's thesis, Royal Institute of Technology, 2002.

PhD Dissertations
Division of Automatic Control
Linköping University

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.

H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.

A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.

P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.

J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.

M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.

U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.

A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.

N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.

K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.

M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.

F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.

V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.

M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.

F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.

J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.

J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.

J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.

O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.

R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.

D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.

R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.

J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.

E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.

M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.

T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.