Linköping studies in science and technology. Dissertations.
No. 1371

On design of low order H-infinity controllers

Daniel Ankelhed

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

Linköping 2011

Cover illustration: An illustration of the feasible set for an H_∞ synthesis problem when the system has one state. Any point in the interior corresponds to a controller with one state, i.e., a so-called full order controller. A point on the curved boundary corresponds to a controller with zero states, i.e., a so-called static controller or reduced order controller.

Linköping studies in science and technology. Dissertations.
No. 1371

On design of low order H-infinity controllers

Daniel Ankelhed

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

ISBN 978-91-7393-157-1 ISSN 0345-7524

Printed by LiU-Tryck, Linköping, Sweden 2011

To U. Ché

Abstract

When designing controllers with robust performance and stabilization requirements, H-infinity synthesis is a common tool to use. These controllers are often obtained by solving mathematical optimization problems. The controllers that result from these algorithms are typically of very high order, which complicates implementation. Low order controllers are usually desired, since they are considered more reliable than high order controllers. However, if a constraint on the maximum order of the controller is set that is lower than the order of the so-called augmented system, the optimization problem becomes nonconvex and it is relatively difficult to solve. This is true even when the order of the augmented system is low.

In this thesis, optimization methods for solving these problems are considered. In contrast to other methods in the literature, the approach used in this thesis is based on formulating the constraint on the maximum order of the controller as a rational function in an equality constraint. Three methods are then suggested for solving this smooth nonconvex optimization problem.

The first two methods use the fact that the rational function is nonnegative. The problem is then reformulated as an optimization problem where the rational function is to be minimized over a convex set defined by linear matrix inequalities (LMIs). This problem is then solved using two different interior point methods.

In the third method the problem is solved by using a partially augmented Lagrangian formulation where the equality constraint is relaxed and incorporated into the objective function, but where the LMIs are kept as constraints. Again, the feasible set is convex and the objective function is nonconvex.

The proposed methods are evaluated and compared with two well-known methods from the literature. The results indicate that the first two suggested methods perform well especially when the number of states in the augmented system is less than 10 and 20, respectively. The third method has comparable performance with two methods from literature when the number of states in the augmented system is less than 25.

Populärvetenskaplig sammanfattning

Robust reglering är ett verktyg som används för att konstruera regulatorer till system som har stora krav på tålighet mot parametervariationer och störningar. En metod för att konstruera dessa typer av robusta regulatorer är så kallad H-oändlighetssyntes. För att använda denna metod utgår man vanligen från en nominell modell av systemet som man utökar med olika viktfunktioner som motsvaras av de krav på robusthet för parameterosäkerheter och störningar som ställs på systemet. Detta får som följd att det utökade systemet ofta får mycket högre komplexitet än det nominella systemet. Regulatorns komplexitet blir i regel lika hög som det utökade systemets komplexitet. Inför man en övre gräns för den grad av komplexitet som tolereras, resulterar detta i mycket komplicerade optimeringsproblem som måste lösas för att en regulator ska kunna konstrueras, även i de fall då det nominella systemet har låg komplexitet. Att ha en regulator med låg komplexitet är önskvärt då det förenklar implementering.

I denna avhandling föreslås tre metoder för att lösa denna typen av optimeringsproblem. Dessa tre metoder jämförs med två andra metoder beskrivna i litteraturen. Slutsatsen är att de föreslagna metoderna uppnår bra resultat så länge som systemen inte har alldeles för hög komplexitet.

Acknowledgments

Wow, I have finally come to the end of this road! It has been a long trip and I would be lying if I said that I have enjoyed every part of it. However, I am so glad that I made it all the way.

First, I would like to greatly thank my supervisors Anders Hansson and Anders Helmersson. Without your guidance and encouragement this trip would have been truly impossible.

Thank you Lennart Ljung for letting me join the automatic control group and thank you Svante Gunnarsson, Ulla Salaneck, Åsa Karmelind and Ninna Stensgård for keeping track of everything here.

Extra thanks go to Anders Hansson, Anders Helmersson, Sina Khoshfetrat Pakazad, Daniel Petersson and Johanna Wallén for your thorough proofreading of the manuscript of this thesis. Without your comments, this thesis would certainly be painful to read. Writing the thesis would not have been so easy without the LaTeX support from Gustaf Hendeby, Henrik Tidefelt and David Törnqvist. Thank you!

A special thank you goes to Janne Harju Johansson with whom I shared office during my first three years here in the group. We started basically at the same time and had many enjoyable discussions about courses, research, rock music, fiction literature or just anything. I would also like to give special thanks to my colleagues André Carvalho Bittencourt, Sina Khoshfetrat Pakazad, Emre Özkan and Lubos Vaci. You always come up with fun ideas involving BBQ (regardless of the temperature being minus a two digit value or not), drinking beer, visit restaurants, watching great(?) movies or just hanging out. I would also like to thank the whole automatic control group for contributing to the nice atmosphere. The discussions in the coffee room can really turn a dull day into a great one!

I would also like to take the opportunity to thank my family. My parents Yvonne and Göte for your endless support, love and encouragement. Lars, always in a good mood and willing to help with anything. Joakim, Erik, Johanna and Jacob, when this book is finished I will hopefully have more time to spend time with you.

Last, I would like to thank the Swedish research council for financial support under contract no. 60519401.

Linköping, May 2011

Daniel Ankelhed

Contents

Notation

Abbreviations

Abbreviation	Meaning
BFGS	Broyden, Fletcher, Goldfarb, Shannon
BMI	Bilinear matrix inequality
KKT	Karush-Kuhn-Tucker
LMI	Linear matrix inequality
LQG	Linear quadratic Gaussian
LTI	Linear time-invariant
NLP	nonlinear problem/program
NT	Nesterov-Todd
SDP	Semidefinite program
SOCP	Second order cone program
SQP	Sequential quadratic programming
SSP	Sequential semidefinite programming
SVD	Singular value decomposition

Sets

Notation	Meaning
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of nonnegative real numbers
$\mathbb{R}_{++}$	Set of positive real numbers
$\mathbb{R}^n$	Set of real-valued vectors with n rows
$\mathbb{R}^{n\times m}$	Set of real-valued matrices with n rows and m columns
$\mathbb{S}^n$	Set of symmetric matrices of size $n \times n$
$\mathbb{S}^n_+$	The positive semidefinite cone of symmetric matrices
$\mathbb{S}^n_{++}$	The positive definite cone of symmetric matrices
$\emptyset$	The empty set

Symbols, operators and functions

Notation	Meaning
H_∞	H-infinity
$A \succ (\succeq) 0$	The matrix A is positive (semi)definite
$A \prec (\preceq) 0$	The matrix A is negative (semi)definite
$A \succeq_K (\succ_K) 0$	Generalized (strict) inequality with respect to the cone K
$x \geq (>) 0$	Element-wise (strict) inequality
A^T	Transpose of matrix A
A^{-1}	Inverse of matrix A
$A^\perp$	Orthogonal complement of matrix A
A_{ij}	Component (i, j) of matrix A
x_i	The ith element of vector x
$x^{(i)}$	Value of x at iteration i
$\det(A)$	Determinant of a matrix $A \in \mathbb{R}^{n \times n}$
I_n	Identity matrix of size $n \times n$
$\langle A, B \rangle$	Inner product of A and B
$\|\|x\|\|_p$	p-norm of vector x
$\|\|A\|\|_\mathrm{F}$	Frobenius norm of matrix A
$\|\|G(s)\|\|_\infty$	The H_∞ norm of the continuous LTI system $G(s)$
$\mathrm{trace}(A)$	Trace of a matrix A
$\mathrm{range}\, A$	Range of a matrix A
$\ker A$	Kernel of a matrix A
$A \otimes B$	Kronecker product of matrix A and B
$A \otimes_s B$	Symmetric Kronecker product of matrix A and B
$\mathrm{vec}(A)$	Vectorization of matrix A
$\mathrm{svec}(A)$	Symmetric vectorization of matrix $A \in \mathbb{S}$
$\mathrm{smat}(a)$	The inverse operator of svec for a vector a
$\mathrm{vech}(A)$	Half-vectorization of matrix $A \in \mathbb{R}^{n \times n}$
$\nabla_x f(x)$	The gradient of function $f(x)$ with respect to x
$\nabla^2_{xx} f(x)$	The hessian of function $f(x)$ with respect to x
$\min(a, b)$	The smallest of the scalars a and b
$\mathrm{argmin}_x\, f(x)$	The minimizing argument of $f(x)$
$\mathrm{diag}(A)$	The diagonal elements of A
$\mathrm{blkdiag}(A, B, \ldots)$	Block diagonal matrix with submatrices $A, B, \ldots$
$\mathrm{dom}\, f(x)$	The domain of function $f(x)$

Part I

Background

1 Introduction

In this thesis, the problem of designing low order H_∞ controllers for continuous linear time-invariant (LTI) systems is addressed. This chapter is structured as follows. In Section 1.1 we discuss the motivation for the thesis and present important work in the past that are related to this work. In Section 1.2 questions are listed which we aim to answer in this thesis. A list of publications is given in Section 1.3, where the author is the main contributor, and the contributions are summarized in Section 1.4. We conclude this chapter with Section 1.5 by presenting the outline of the remaining chapters of the thesis.

1.1 Background

We begin by a brief overview of the history of robust control. Then we present what has been done in the past that is related to design of low order H_∞ controllers, and we also very briefly discuss controller reduction methods. We conclude this section with the motivation for the thesis.

1.1.1 Robust control

The development of robust control theory emerged during the 1980s and and a contributory factor certainly was the fact that the robustness of linear quadratic Gaussian (LQG) controllers can get arbitrarily bad as reported in Doyle [1978]. A few years later an important step in the development towards a robust control theory was taken by Zames [1981], who introduced the concept of H_∞ theory.

The H_∞ synthesis, which is an important tool when solving robust control problems, was a cumbersome problem to solve until a technique was presented in Doyle et al. [1989], which is based on solving two Riccati equations. Using this method, the robust design tools became much easier to use and gained popularity. Quite soon thereafter, in Gahinet and Apkarian [1994] and Iwasaki and Skelton [1994], linear matrix inequalities (LMIs) were found to be a suitable tool for solving these kinds of problems. Also related problems, such as gain scheduling synthesis, see e.g. Packard [1994] and Helmersson [1995], fit into the LMI framework. In parallel to the theory for solving problems using LMIs, see e.g. the survey papers by Vandenberghe and Boyd [1996] and Todd [2001], numerical methods for solving LMIs were being developed.

Typical applications for robust control include systems that have high requirements for robustness to parameter variations and high requirements for disturbance rejection. The controllers that result from these algorithms are typically of very high order, which complicates implementation. However, if a constraint on the maximum order of the controller is set, that is lower than the order of the plant, the problem is no longer convex and is then relatively hard to solve. These problems become very complex, even when the order of the system to be controlled is low. This motivates the use of efficient special purpose algorithms that can solve these kinds of problems.

1.1.2 Previous work related to low order control design

In Hyland and Bernstein [1984] necessary conditions for a reduced order controller to stabilize a system were stated. In Gahinet and Apkarian [1994] it was shown that in order to find a reduced order controller using the LMI formulation, a rank constraint had to be satisfied. In Fu and Luo [1997] it was shown that this problem is so-called NP-hard. Many researchers were now focused on the issue of finding efficient methods to solve this hard problem.

The DK-iteration procedure became a popular method to solve robust control problems, see e.g. Doyle [1985]. In Iwasaki [1999] a similar method for low order H_∞ synthesis was presented, which solved the associated BMIs by fixing some variables and optimizing on others in an alternating manner. The problems to be solved in each step were LMIs. One problem with these methods is that convergence is not guaranteed.

In Grigoriadis and Skelton [1996] an alternating projections algorithm was presented. The algorithm seeks to locate an intersection of a convex set of LMIs and a rank constraint. However, only local convergence is guaranteed for the low order case. Similar methods are presented in Beran [1997]. Other algorithms that appeared around this time were e.g. a min/max algorithm in Geromel et al. [1998], the XY-centering algorithm in Iwasaki and Skelton [1995], the potential reduction algorithm in David [1994]. For a survey on static output feedback methods, see Syrmos et al. [1997].

Also some global methods began to appear. In Beran [1997] a branch and bound algorithm for solving bilinear matrix inequalities (BMIs) was presented, which divides the set into several smaller ones where upper and lower bounds on the optimal value are calculated. Similar approaches were presented in e.g. Goh et al. [1995], VanAntwerp et al. [1997] and Tuan and Apkarian [2000] and in the references therein. In Apkarian and Tuan [1999, 2000] an algorithm was presented for minimization of a nonconvex function over convex sets defined by LMIs. The problem was solved using a modified Frank-Wolfe method, see Frank and Wolfe [1956], combined with branch and bound methods.

Mesbahi and Papavassilopoulos [1997] showed how to compute lower and upper bound on the order of a dynamical output feedback controller that stabilizes a given system by solving two semi-definite programs. In El Ghaoui et al. [1997], a cone complementarity linearization method was presented. In each iteration it solved a problem involving a linearized nonconvex objective function subject to LMI constraints. Another method based on linearization is Leibfritz [2001].

A primal-dual method with a trust-region framework for solving nonconvex robust control problems was suggested in Apkarian and Noll [2001]. A modified version of the augmented Lagrangian method, the *partially* augmented Lagrangian method was used in Fares et al. [2001] to solve a robust control problem where the search direction was calculated using a modified Newton method and a trust-region method. In Fares et al. [2002] the same formulation was used but solved using a sequential semidefinite programming approach. In Apkarian et al. [2003] and Apkarian et al. [2004] the low order H_∞ problem was considered, where in each iteration a convex SDP was solved in order to find a search direction.

A barrier method for solving problems involving BMIs was presented in Kocvara et al. [2005] which is an extension of the so-called PBM method in Ben-Tal and Zibulevsky [1997] to semidefinite programming. It uses a modified Newton method to calculate the search direction. If ill-conditioning of the Hessian is detected, a slower but more robust trust-region method is used instead. The method was implemented in the software PENBMI. Other methods that attack the BMI problem using local methods are e.g. Leibfritz and Mostafa [2002], Hol et al. [2003] Kanev et al. [2004] and Thevenet et al. [2005]. For another overview of earlier work in robust control, the introduction of Kanev et al. [2004] is recommended.

In Orsi et al. [2006], a method similar to the alternating projection algorithm in Grigoriadis and Skelton [1996] for finding intersections of sets defined by rank constraints and LMIs is proposed. The method is implemented in the software LMIrank, see Orsi [2005].

In Apkarian and Noll [2006a] a nonsmooth, multi directional search approach was considered that did not use the LMI formulation in Gahinet and Apkarian [1994] but instead searched directly in the controller parameter space. However, the method they used for solving this nonsmooth problem seem to have been abandoned in favor for an approach using subgradient calculus, see e.g. Clarke [1990]. This approach was presented in Apkarian and Noll [2006b] and in 2010 it resulted in the code HINFSTRUCT, which is part of the ROBUST CONTROL TOOLBOX in MATLAB® version 7.11. In parallel, a code package for H-infinity fixed order optimization, HIFOO, was being developed that considered the same nonsmooth problem formulation as in Apkarian and Noll [2006b] but with another approach. It was presented in Burke et al. [2006] and parts of its code are based on gradient sampling, see Burke et al. [2005]. Further developments of HIFOO were presented in Gumussoy and Overton [2008a] and as presented in the paper by Arzelier et al. [2011] it now also includes methods for H_2 controller synthesis. An advantage with these kind of nonsmooth methods, compared to LMI-based methods, is that they seem better fitted to handle systems with large dimensions. Another approach that directly minimizes an appropriate nonsmooth function of the controller parameters is presented in Mammadov and Orsi [2005]. However, it uses a different objective function compared to Burke et al. [2006] and Apkarian and Noll [2006b].

New sufficient LMI conditions for low order controller design were presented in Trofino [2009], however some degree of conservatism was introduced due to the conditions only being sufficient. An algorithm that combines a randomization algorithm with a coordinate descent cross-decomposition algorithm is presented in Arzelier et al. [2011]. The first part of the algorithm randomizes a set of feasible points while the second part optimizes on a group of variables while keeping the other group of variables fixed and vice versa.

1.1.3 Controller reduction methods

A completely different approach worth mentioning is to compute the full order controller and then apply model reduction techniques to get a low order controller. Some references on this approach are e.g. Enns [1984], Zhou [1995], Goddard and Glover [1998], and Kavranoğlu and Al-Amer [2001]. The results in Gumussoy and Overton [2008a] indicate that controller reduction methods perform best when the controller order is close to the order of the system. However, when the controller order is low compared to the order of the system, the results is in clear favor of optimization based methods such as HIFOO.

1.1.4 Motivation

To conclude the overview of related published work the following can be said. Several approaches to low order H_∞ controller synthesis have been proposed in the past. All methods have their advantages and disadvantages. None of the global methods have polynomial time complexity due to the NP-hardness of the problem. As a result, these approaches require a large computational effort even for problems of modest size. Most of the local methods, on the other hand, are computationally fast but may not converge to the global optimum. The reason for this is the inherent nonconvexity of the problem. Some problem formulations are "less" nonconvex than others, e.g. Apkarian et al. [2003], in the sense that it is only the objective function that is nonconvex while the constraints are convex. However, a drawback with the approach in Apkarian et al. [2003] is that the system is augmented with extra states in the case that a dynamic controller, i.e., a controller with nonzero states, is searched for.

In Helmersson [2009] it was shown that the matrix rank constraint used in H_∞

controller synthesis can be formulated as a polynomial constraint. This allows new approaches for low order H_∞ controller design where a nonconvex function is to be minimized over a convex set defined by LMIs. The order of the controller in these approaches is decided by using a specific coefficient in a polynomial as objective function instead of augmenting the system as is done in e.g. Apkarian et al. [2003]. Hopefully, this results in lower computational complexity and better results. In this thesis three different optimization algorithms for low order H_∞ controller synthesis will be investigated that use the formulation in Helmersson [2009].

1.2 Goals

In this thesis we aim to answer the following questions.

1. Which are currently the state of the art methods for low order H_∞ controller synthesis?
2. When using the reformulation of the rank constraint of a matrix as a quotient of two polynomials, see Helmersson [2009], in tandem with the classical LMI formulation of the H_∞ controller synthesis problem in Gahinet and Apkarian [1994], what approaches can be used for solving the resulting optimization problems? How well do these methods perform?
3. What are the advantages and disadvantages of using the LMI formulation of the H_∞ controller synthesis problem compared to using nonsmooth methods, e.g. the methods in Apkarian and Noll [2006b] and Gumussoy and Overton [2008a]?

1.3 Publications

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

D. Ankelhed, A. Helmersson, and A. Hansson. A primal-dual method for low order H-infinity controller synthesis. In *Proceedings of the 2009 IEEE Conference on Decision and Control*, Shanghai, China, Dec 2009.

D. Ankelhed, A. Helmersson, and A. Hansson. Additional numerical results for the quasi-Newton interior point method for low order H-infinity controller synthesis. Technical Report LiTH-ISY-R-2964, Department of Automatic Control, Linköping university, Sweden, 2010. URL `http://www.control.isy.liu.se/publications/doc?id=2313`.

D. Ankelhed, A. Helmersson, and A. Hansson. A quasi-Newton interior point method for low order H-infinity controller synthesis. *Accepted for publication in IEEE Transactions on Automatic Control*, 2011a.

D. Ankelhed. An efficient implementation of gradient and Hessian calculations of the coefficients of the characteristic polynomial of I - XY. Technical Report LiTH-ISY-R-2997, Department of Automatic Control, Linköping university, Sweden, 2011. URL `http://www.control.isy.liu.se/publications/doc?id=2387`.

D. Ankelhed, A. Helmersson, and A. Hansson. A partially augmented Lagrangian algorithm for low order H-infinity controller synthesis using rational constraints. *Submitted to the 2011 IEEE Conference on Decision and Control*, December 2011b.

Some of the results in this thesis has previously been published in

D. Ankelhed. *On low order controller synthesis using rational constraints*. Licentiate thesis no. 1398, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Mar 2009.

Additionally, some work not directly related to this thesis has been published in

D. Ankelhed, A. Helmersson, and A. Hansson. Suboptimal model reduction using LMIs with convex constraints. Technical Report LiTH-ISY-R-2759, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Dec 2006. URL `http://www.control.isy.liu.se/publications/doc?id=1889`.

1.4 Contributions

Below the main contributions of the thesis are listed.

- A new algorithm for low order H_∞ controller design based on a primal-dual method is presented. The algorithm was introduced in Ankelhed [2009] and Ankelhed et al. [2009]. The algorithm was implemented and evaluated and the results are compared with a well-known method from the literature.

- A new algorithm for low order H_∞ controller design based on a primal log-barrier method is presented. The algorithm was introduced in Ankelhed [2009] and solves the same problem as the primal-dual algorithm above. The method was implemented and evaluated and the results are compared with the primal-dual method above and a well-known method from the literature.

- A modified version of the primal-dual algorithm is presented that clearly lowers the required computational time. This work was first presented in Ankelhed et al. [2011a], where exact Hessian calculations are replaced with BFGS updating formulae. The algorithm was implemented and the results from an extended numerical evaluation was published in Ankelhed et al. [2010], and a summary of the results is presented in the thesis.

- A computationally efficient implementation of the gradient and Hessian calculations of the coefficients in the characteristic polynomial of the matrix $I - XY$ is presented. This work was first published in Ankelhed [2011].

- A new algorithm for low order H_∞ controller design based on a partially augmented Lagrangian method is presented. The algorithm was implemented and evaluated and the results were compared with two methods from the literature. This work was first published in Ankelhed et al. [2011b] and uses the efficient implementation for gradient and Hessian calculations that was published in Ankelhed [2011] in order to lower the required computational time.

1.5 Thesis outline

The thesis is divided into three parts. The aim of the first part is to cover the background for the thesis.

- In Chapter 2, the focus lies on H_∞ synthesis for linear systems.
- In Chapter 3 optimization preliminaries are presented. The focus is on minimization of a nonconvex objective function subject to semidefinite constraints.
- In Chapter 4 it is described how a quotient of coefficients of the characteristic polynomial of a special matrix are connected to its rank. This is a common theme for all suggested methods in the thesis.

The second part of the thesis presents the suggested methods for design of low order H_∞ controllers.

- In Chapter 5 a primal logarithmic barrier method for H_∞ synthesis is presented.
- In Chapter 6 a primal-dual method for H_∞ synthesis is presented. A modification of this method is also described.
- In Chapter 7 a partially augmented Lagrangian method for H_∞ synthesis is presented.

The third part of the thesis presents the results and conclusions of the thesis.

- In Chapter 8 the numerical evaluations of the suggested methods are presented. The other methods used in the evaluation are also described.
- In Chapter 9 conclusions and suggested future research are presented.

2 Linear systems and H_∞ synthesis

In this chapter, basic theorems related to linear systems and H_∞ synthesis are presented. This includes defining the performance measure for a system and reformulating this to an LMI (linear matrix inequality) framework and how to recover the controller parameters once the problem involving the LMIs has been solved. We also briefly mention the concept of balanced realizations. The chapter is concluded by summarizing its contents in a general algorithm for H_∞ synthesis.

Some references in the field of robust control are e.g. Zhou et al. [1996], Skogestad and Postlethwaite [1996] and for the LMI formulations we refer to Gahinet and Apkarian [1994] and Dullerud and Paganini [2000]. More details on balanced realizations and gramians can be found in e.g. Glover [1984] and Skogestad and Postlethwaite [1996].

Denote with $\mathbb{S}^n$ the set of real symmetric $n \times n$ matrices and $\mathbb{R}^{m\times n}$ is the set of real $m \times n$ matrices, while $\mathbb{R}^n$ denotes a real vector of dimension $n \times 1$. The notation $A \succ 0$ $(A \succeq 0)$ and $A \prec 0$ $(A \preceq 0)$ means A is a positive (semi)definite matrix and negative (semi)definite matrix, respectively. If A is a symmetric matrix, the notation $A \in \mathbb{S}^n_{++}$ $(A \in \mathbb{S}^n_{+})$ and $A \succ 0$ $(A \succeq 0)$ are equivalent.

2.1 Linear system representation

In this section we will introduce notations for linear systems that are controlled by linear controllers and derive the equations that describe the closed loop system.

2.1.1 System description

Let H denote a linear system with state vector, $x \in \mathbb{R}^{n_x}$. The input vector contains the disturbance signal, $w \in \mathbb{R}^{n_w}$, and the control signal, $u \in \mathbb{R}^{n_u}$. The output vector contains the measurement signal, $y \in \mathbb{R}^{n_y}$, and the performance signal, $z \in \mathbb{R}^{n_z}$. The system is illustrated in Figure 2.1. In terms of its state-space matrices, we can represent the linear system as

$$H : \quad \left(\begin{array}{c} \dot{x} \\ \hline z \\ y \end{array}\right) = \left(\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right)\left(\begin{array}{c} x \\ \hline w \\ u \end{array}\right). \tag{2.1}$$

We assume that D_{22} is zero, i.e., the system is strictly proper from u to y. If this is not the case, we can find a controller $\tilde{K}$ for the system where D_{22} is set to zero, and then construct the controller for the system in (2.1) as

$$K = \tilde{K}(I + D_{22}\tilde{K})^{-1}.$$

Hence, there is no loss of generality in making this assumption. For simplicity, we assume that the system is on minimal form, i.e., it is both observable and controllable. However, in order to find a controller, it is enough to assume stabilizability of (A, B_2) and detectability of (A, C_2), i.e., the nonobservable and noncontrollable modes are stable.

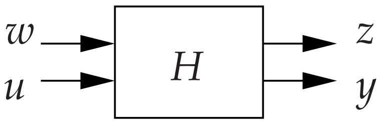

Figure 2.1: A system with two inputs and two outputs. The inputs are the disturbance signal, w, and the control signal, u. The outputs are the performance signal, z, and the output signal, y.

2.1.2 Controller description

The linear controller is denoted K. It takes the system measurement, y, as input and the output vector is the control signal, u. The state-space matrices for the controller are defined by the equation

$$K : \quad \begin{pmatrix} \dot{x}_K \\ u \end{pmatrix} = \begin{pmatrix} K_A & K_B \\ K_C & K_D \end{pmatrix}\begin{pmatrix} x_K \\ y \end{pmatrix}, \tag{2.2}$$

where $x_K \in \mathbb{R}^{n_k}$ is the state vector of the controller. How the controller is connected to the system is illustrated in Figure 2.2.

2.1.3 Closed loop system description

Next we will derive expressions for the closed loop system in terms of the state-space matrices of the system in (2.1) and the controller in (2.2). Let us denote the closed loop system by H_c. The state-space matrices of the closed loop system can

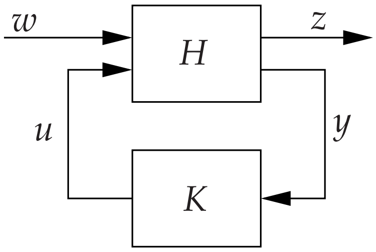

Figure 2.2: *A standard setup for H$_\infty$ controller synthesis, with the system H controlled through feedback by the controller K.*

be derived by combining (2.1) and (2.2) to obtain the following.

$$\begin{aligned} u &= K_C x_K + K_D y = K_C x_K + K_D C_2 x + K_D D_{21} w \\ \dot{x}_K &= K_A x_K + K_B y = K_A x_K + K_B C_2 x + K_B D_{21} w \\ \dot{x} &= Ax + B_1 w + B_2 u = (A + B_2 K_D C_2)x + B_2 K_C x_K + (B_1 + B_2 K_D D_{21})w \\ z &= C_1 x + D_{11} w + D_{12} u \\ &= (C_1 + D_{12} K_D C_2)x + D_{12} K_C x_K + (D_{11} + D_{12} K_D D_{21})w \end{aligned}$$

From the equations above we obtain the closed loop expression as

$$H_c : \quad \begin{pmatrix} \dot{x} \\ \dot{x}_K \\ \hline z \end{pmatrix} = \left(\begin{array}{cc|c} A + B_2 K_D C_2 & B_2 K_C & B_1 + B_2 K_D D_{21} \\ K_B C_2 & K_A & K_B D_{21} \\ \hline C_1 + D_{12} K_D C_2 & D_{12} K_C & D_{11} + D_{12} K_D D_{21} \end{array}\right) \begin{pmatrix} x \\ x_K \\ \hline w \end{pmatrix}. \tag{2.3}$$

Denoting the closed loop system states $x_C = \begin{pmatrix} x \\ x_K \end{pmatrix}$ and using the above matrix partitioning, we can write (2.3) as

$$H_c : \quad \begin{pmatrix} \dot{x}_C \\ z \end{pmatrix} = \begin{pmatrix} A_C & B_C \\ C_C & D_C \end{pmatrix} \begin{pmatrix} x_C \\ w \end{pmatrix}, \tag{2.4}$$

where $x_C \in \mathbb{R}^{n_x + n_k}$. Using the system matrices in (2.4) we can write the transfer function of H_c in terms of its system matrices A_C, B_C, C_C, D_C as

$$H_c(s) = C_C(sI - A_C)^{-1} B_C + D_C. \tag{2.5}$$

2.2 The H$_\infty$ norm

In this section we introduce the concept of H$_\infty$ norm and conditions in terms of linear matrix inequalities (LMIs) of its upper bound. First, we define the following.

Definition 2.1 (H$_\infty$ norm of a system). Let the transfer function of a stable linear system be given by $H(s) = C(sI - A)^{-1}B + D$, where A, B, C, D are its state-space matrices. The H$_\infty$ norm of this system is defined as

$$\|H(s)\|_\infty = \sup_\omega \bar{\sigma}\Big(H(i\omega)\Big),$$

where $\bar{\sigma}(\,\cdot\,)$ denotes the largest singular value.

Using the *Bounded real lemma*, which is introduced below, we can state an equivalent condition for a system to have H$_\infty$ norm less than γ.

Lemma 2.1 (Bounded real lemma, Scherer [1990]). *For any $\gamma > 0$ we have that all eigenvalues of A are in the left hand half-plane and $\|H(s)\|_\infty < \gamma$ hold if and only if there exists a matrix $P \in \mathbb{S}_{++}$ such that*

$$\begin{pmatrix} A^TP + PA + C^TC & PB + C^TD \\ B^TP + D^TC & D^TD - \gamma^2 I \end{pmatrix} \prec 0. \tag{2.6}$$

We can rewrite the inequality in (2.6) as

$$\begin{pmatrix} PA + A^TP & PB \\ B^TP & -\gamma^2 I \end{pmatrix} + \begin{pmatrix} C^T \\ D^T \end{pmatrix} I \begin{pmatrix} C & D \end{pmatrix} \prec 0. \tag{2.7}$$

Then multiply the inequality (2.7) by γ^{-1} and let $P_1 = \gamma^{-1}P$ to obtain

$$\begin{pmatrix} P_1A + A^TP_1 & P_1B \\ B^TP_1 & -\gamma I \end{pmatrix} + \begin{pmatrix} C^T \\ D^T \end{pmatrix} \gamma^{-1} I \begin{pmatrix} C & D \end{pmatrix} \prec 0. \tag{2.8}$$

From now on, we will drop the index on P for convenience. For later purposes it is useful to rewrite the inequality in (2.8) such that it becomes linear in the state-space matrices A, B, C, D. In order to do this, the following lemma is useful.

Lemma 2.2 (Schur complement formula, Boyd et al. [1994]). *Assume we have that $R \in \mathbb{S}^n$, $S \in \mathbb{S}^m$ and $G \in \mathbb{R}^{n\times m}$. Then the following conditions are equivalent.*

1. $$R \prec 0, \quad S - G^TR^{-1}G \prec 0 \tag{2.9}$$
2. $$\begin{pmatrix} S & G^T \\ G & R \end{pmatrix} \prec 0 \tag{2.10}$$

Now, by using Lemma 2.2, the inequality in (2.8) can be written as

$$\begin{pmatrix} PA + A^TP & PB & C^T \\ B^TP & -\gamma I & D^T \\ C & D & -\gamma I \end{pmatrix} \prec 0, \tag{2.11}$$

which is an LMI in the matrices A, B, C, D if the matrix P and γ are given. Now we have shown that finding a matrix P such that the inequality in (2.6) is satisfied is equivalent to finding a matrix P such that the inequality in (2.11) is satisfied.

2.3 H$_\infty$ controller synthesis

In this section we will derive the solvability conditions for finding a controller for the system in (2.1) such that the closed loop H$_\infty$ norm is less than γ, i.e., such that $\|H_c(s)\|_\infty < \gamma$.

Let the matrix $P \in \mathbb{S}^{n_x+n_k}$ and its inverse be partitioned as

$$P = \begin{pmatrix} X & X_2 \\ X_2^T & X_3 \end{pmatrix} \quad \text{and} \quad P^{-1} = \begin{pmatrix} Y & Y_2 \\ Y_2^T & Y_3 \end{pmatrix}, \tag{2.12}$$

where $X, Y \in \mathbb{S}^{n_x}_{++}$, $X_2, Y_2 \in \mathbb{R}^{n_x \times n_k}$ and $X_3, Y_3 \in \mathbb{S}^{n_k}_{++}$. Then insert P and the closed loop system matrices A_C, B_C, C_C, D_C into the inequality in (2.11). After some rearrangements we get the following matrix inequality.

$$\begin{aligned}
&\underbrace{\begin{pmatrix} XA + A^TX & A^TX_2 & XB_1 & C_1^T \\ X_2^TA & 0 & X_2^TB_1 & 0 \\ B_1^TX & B_1^TX_2 & -\gamma I & D_{11}^T \\ C_1 & 0 & D_{11} & -\gamma I \end{pmatrix}}_{Q} + \\
&+ \underbrace{\begin{pmatrix} XB_2 & X_2 \\ X_2^TB_2 & X_3 \\ 0 & 0 \\ D_{12} & 0 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} K_D & K_C \\ K_B & K_A \end{pmatrix}}_{K} \underbrace{\begin{pmatrix} C_2 & 0 & D_{21} & 0 \\ 0 & I & 0 & 0 \end{pmatrix}}_{V^T} \\
&+ \underbrace{\begin{pmatrix} C_2 & 0 & D_{21} & 0 \\ 0 & I & 0 & 0 \end{pmatrix}^T}_{V} \underbrace{\begin{pmatrix} K_D & K_C \\ K_B & K_A \end{pmatrix}^T}_{K^T} \underbrace{\begin{pmatrix} XB_2 & X_2 \\ X_2^TB_2 & X_3 \\ 0 & 0 \\ D_{12} & 0 \end{pmatrix}^T}_{U^T} \prec 0
\end{aligned} \tag{2.13}$$

The matrix inequality in (2.13) is bilinear in the controller variables, K_A, K_B, K_C, K_D and the matrices X, X_2, X_3. The introduced matrix aliases Q, U, K, V in (2.13) correspond to the matrices in Lemma 2.3 below, which states two conditions on the existence of a matrix K that satisfies the inequality in (2.13). However, first we need to define the concept of an orthogonal complement.

Definition 2.2 (Orthogonal complement). Let $V^\perp$ denote any full rank matrix such that $\ker V^\perp = \operatorname{range} V$. Then $V^\perp$ is an *orthogonal complement* of V.

Remark 2.1. Note that for $V^\perp$ to exist, V needs to have linearly dependent rows. We have that $V^\perp V = 0$. There exist infinitely many choices of $V^\perp$.

Lemma 2.3 (Elimination lemma, Gahinet and Apkarian [1994]). *Given matrices $Q \in \mathbb{S}^n$, $U \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{n\times p}$, there exists a $K \in \mathbb{R}^{m\times p}$ such that*

$$Q + UKV^T + VK^TU^T \prec 0, \tag{2.14}$$

if and only if

$$U^\perp Q U^{\perp T} \prec 0 \quad \text{and} \quad V^\perp Q V^{\perp T} \prec 0, \tag{2.15}$$

where $U^\perp$ is an orthogonal complement of U and $V^\perp$ is an orthogonal complement of V. If $U^\perp$ or $V^\perp$ does not exist, the corresponding inequality disappears.

In order to apply Lemma 2.3 to the inequality in (2.13), we need to derive the orthogonal complements $U^\perp$ and $V^\perp$. Note that U in (2.13) can be factorized as

$$U = \left(\begin{array}{cc} XB_2 & X_2 \\ X_2^T B_2 & X_3 \\ \hline 0 & 0 \\ D_{12} & 0 \end{array}\right) = \left(\begin{array}{c|c} P & 0 \\ \hline 0 & I \end{array}\right)\left(\begin{array}{cc} B_2 & 0 \\ 0 & I \\ \hline 0 & 0 \\ D_{12} & 0 \end{array}\right).$$

and an orthogonal complement $U^\perp$ can now be constructed as

$$U^\perp = \left(\begin{array}{cc} B_2 & 0 \\ 0 & I \\ \hline 0 & 0 \\ D_{21} & 0 \end{array}\right)^\perp \left(\begin{array}{c|c} P^{-1} & 0 \\ \hline 0 & I \end{array}\right).$$

By using Lemma 2.3 and performing some rearrangements, the inequality (2.13) is now equivalent to the two LMIs

$$\begin{aligned} &\begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} XA + A^TX & XB_1 & C_1^T \\ B_1^TX & -\gamma I & D_{11}^T \\ C_1 & D_{11} & -\gamma I \end{pmatrix}\begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix} \prec 0 \\ &\begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} AY + YA^T & YC_1^T & B_1 \\ C_1Y & -\gamma I & D_{11} \\ B_1^T & D_{11}^T & -\gamma I \end{pmatrix}\begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix} \prec 0, \end{aligned} \tag{2.16}$$

where $\mathcal{N}_X$ and $\mathcal{N}_Y$ denote any bases of the nullspaces of $\begin{pmatrix} C_2 & D_{21} \end{pmatrix}$ and $\begin{pmatrix} B_2^T & D_{12}^T \end{pmatrix}$ respectively. Now, the LMIs in (2.16) are coupled by the relation of X and Y through (2.12), which can be simplified after using the following lemma.

Lemma 2.4 (Packard [1994]). *Suppose $X, Y \in \mathbb{S}_{++}^{n_x}$ and n_k being a nonnegative integer. Then the following statements are equivalent.*

1. *There exist $X_2, Y_2 \in \mathbb{R}^{n_x\times n_k}$ and $X_3, Y_3 \in \mathbb{R}^{n_k}$ such that*

$$P = \begin{pmatrix} X & X_2 \\ X_2^T & X_3 \end{pmatrix} \succ 0 \quad \text{and} \quad P^{-1} = \begin{pmatrix} Y & Y_2 \\ Y_2^T & Y_3 \end{pmatrix} \succ 0. \tag{2.17}$$

2. *The following inequalities hold.*

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \geq 0 \quad \textit{and} \quad \operatorname{rank}\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \leq n_x + n_k. \tag{2.18}$$

Remark 2.2. Note that the first inequality in (2.18) implies that $Y > 0$. The factorization

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} = \begin{pmatrix} I & Y^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} X - Y^{-1} & 0 \\ 0 & Y \end{pmatrix} \begin{pmatrix} I & 0 \\ Y^{-1} & I \end{pmatrix} \tag{2.19}$$

implies that

$$\operatorname{rank}\begin{pmatrix} X & I \\ I & Y \end{pmatrix} = \operatorname{rank}(X - Y^{-1}) + \operatorname{rank}(Y) = \operatorname{rank}(XY - I) + n_x.$$

Using this fact, the second inequality in (2.18) is equivalent to

$$\operatorname{rank}(XY - I) \leq n_k. \tag{2.20}$$

The solvability conditions for the H$_\infty$ problem, which is essential for this thesis, will now be stated.

Theorem 2.1 (H$_\infty$ controllers for continuous plants). *The problem of finding a linear controller of order $n_k \leq n_x$ such that the closed loop system H_c is stable and such that $\|H_c(s)\|_\infty < \gamma$, is solvable if and only if there exist $X, Y \in \mathbb{S}^{n_x}_{++}$, which satisfy*

$$\begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} XA + A^T X & XB_1 & C_1^T \\ B_1^T X & -\gamma I & D_{11}^T \\ C_1 & D_{11} & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix} \prec 0, \tag{2.21a}$$

$$\begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} AY + YA^T & YC_1^T & B_1 \\ C_1 Y & -\gamma I & D_{11} \\ B_1^T & D_{11}^T & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix} \prec 0, \tag{2.21b}$$

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \geq 0, \tag{2.21c}$$

$$\operatorname{rank}(XY - I) \leq n_k, \tag{2.21d}$$

where $\mathcal{N}_X$ and $\mathcal{N}_Y$ denote any base of the null-spaces of $\begin{pmatrix} C_2 & D_{21} \end{pmatrix}$ and $\begin{pmatrix} B_2^T & D_{12}^T \end{pmatrix}$ respectively.

Proof: Combine Lemma 2.1–2.4 or see the LMI reformulation of Theorem 4.3 in Gahinet and Apkarian [1994]. □

Remark 2.3. If $n_k = n_x$, the rank constraint (2.21d) is trivially satisfied and the problem is convex.

2.4 Gramians and balanced realizations

It is well known that for a linear system, there exist infinitely many realizations and for two realizations of the same system, there exists a nonsingular transformation matrix T that connects the representations. Let A, B, C, D and $\bar{A}, \bar{B}, \bar{C}, \bar{D}$ be the system matrices for two realizations of the same system. Then there exist a nonsingular transformation matrix T such that

$$\bar{A} = TAT^{-1}, \quad \bar{B} = TB, \quad \bar{C} = CT^{-1}, \quad \bar{D} = D. \tag{2.22}$$

The two systems have the same input-output properties, but are represented differently. The state vectors are connected by the relation $\bar{x} = Tx$.

Definition 2.3 (Controllability and observability gramians). Let A, B, C, D be the system matrices for a stable linear system H of order n_x. Then there exist $X, Y \in \mathbb{S}_{++}^{n_x}$ that satisfy

$$XA + A^TX + BB^T = 0, \quad AY + YA^T + C^TC = 0, \tag{2.23}$$

where X and Y are called the *controllability* and *observability gramians*, respectively.

The gramians are often used in model reduction algorithms and can be interpreted as a measure of controllability and observability of a system. For more details, see e.g. Glover [1984] or Skogestad and Postlethwaite [1996].

Definition 2.4 (Balanced realization). A system is said to be in a *balanced realization* if the controllability and observability gramians are equal and diagonal matrices, i.e., $X = Y = \Sigma$, where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. The diagonal entries $\{\sigma_1, \dots, \sigma_n\}$ are called the *Hankel singular values*.

For any stable linear system, a transformation matrix T can be computed that brings the system to a balanced realization by using (2.22). The procedure is described in e.g. Glover [1984], and is stated in Algorithm 1 for convenience.

Actually, we can perform a more general type of a balanced realization around any $X_0, Y_0 \in \mathbb{S}_{++}^{n_x}$ which is described in Definition 2.5.

Definition 2.5 (Balanced realization around $\mathbf{X_0, Y_0}$). Given any $X_0, Y_0 \in \mathbb{S}_{++}^{n_x}$ which need not necessarily be solutions to (2.23), it is possible to calculate a transformation T such that (2.25) holds. Then the realization described by (2.24) is a *Balanced realization around* X_0, Y_0.

The balancing procedure around X_0, Y_0 is the same as normal balancing, thus Algorithm 1 can be used for the same purpose.

Algorithm 1 Transforming a system into a balanced realization, Glover [1984]

Assume that the matrices $X, Y \in \mathbb{S}^{n_x}_{++}$ and the system matrices A, B, C, D are given.

```
1: Perform a Cholesky factorization of $X$ and $Y$ such that
       $X = R_X^T R_X$ and $Y = R_Y^T R_Y$.
2: Then perform a singular value decomposition (SVD) such that
       $R_Y R_X^T = U \Sigma V^T$,
   where $\Sigma$ is a diagonal matrix with the singular values along the diagonal.
3: The state transformation can now be calculated as
       $T = \Sigma^{-1/2} V^T R_X$.
4: The system matrices for a balanced realization of the system are given by
       $\bar{A} = TAT^{-1}, \quad \bar{B} = TB, \quad \bar{C} = CT^{-1}, \quad \bar{D} = D.$   (2.24)
   The controllability gramian and observability gramian are given by
       $\bar{X} = T^{-T} X T^{-1} = \Sigma, \quad \bar{Y} = TYT^T = \Sigma.$   (2.25)
```

2.5 Recovering the matrix P from X and Y

Assume that we have found $X, Y \in \mathbb{S}^{n_x}_{++}$ that satisfy (2.21). We now wish to construct a P such that (2.17) holds. First note the equality

$$P^{-1} = \begin{pmatrix} Y & Y_2 \\ Y_2^T & Y_3 \end{pmatrix} = \begin{pmatrix} (X - X_2 X_3^{-1} X_2^T)^{-1} & -X^{-1} X_2 (X_3 - X_2^T X^{-1} X_2)^{-1} \\ -X_3^{-1} X_2^T (X - X_2 X_3^{-1} X_2^T)^{-1} & (X_3 - X_2^T X^{-1} X_2)^{-1} \end{pmatrix}, \tag{2.26}$$

which is verified by multiplying the expression in (2.26) by the matrix

$$P = \begin{pmatrix} X & X_2 \\ X_2^T & X_3 \end{pmatrix}$$

from the left. Using the fact that the $(1, 1)$ elements in (2.26) are equal, the following equality must hold.

$$X - Y^{-1} = X_2 X_3^{-1} X_2^T \tag{2.27}$$

Now we intend to find $X_2 \in \mathbb{R}^{n_x \times n_k}$ and $X_3 \in \mathbb{R}^{n_k \times n_k}$ that satisfy the equality in (2.27). Perform a Cholesky factorization of X and Y such that $X = R_X^T R_X$ and $Y = R_Y^T R_Y$. Then we have that

$$R_X^T R_X - R_Y^{-1} R_Y^{-T} = X_2 X_3^{-1} X_2^T,$$

which after multiplication by R_Y from the left and by R_Y^T from the right becomes

$$R_Y R_X^T R_X R_Y^T - I = R_Y X_2 X_3^{-1} X_2^T R_Y^T.$$

Then use a singular value decomposition $R_Y R_X^T = U \Sigma V^T$ to obtain

$$U(\Sigma^2 - I)U^T = U\Gamma^2 U^T = R_Y X_2 X_3^{-1} X_2^T R_Y^T, \tag{2.28}$$

where

$$\Sigma = \begin{pmatrix} \Sigma_{n_k} & 0 \\ 0 & I_{n_x - n_k} \end{pmatrix}, \quad \Gamma^2 = \Sigma^2 - I_{n_x} \quad \text{and} \quad \Gamma = \begin{pmatrix} \Gamma_{n_k} & 0 \\ 0 & 0 \end{pmatrix}.$$

Let the transformation matrix be $T = \Sigma^{-1/2} V^T R_X$ which balances the system, i.e., we have that $T^{-T} X T^{-1} = T Y T^T = \Sigma$. Now we can choose

$$X_3 = \Sigma_{n_k} \quad \text{and} \quad X_2 = T^T \begin{pmatrix} \Gamma_{n_k} \\ 0 \end{pmatrix},$$

which satisfy (2.17) and (2.28).

2.6 Obtaining the controller

In the previous section we recovered the matrix variable P. The controller state-space matrices K_A, K_B, K_C, K_D can be obtained by solving the following convex optimization problem, which was suggested in Beran [1997].

$$\begin{aligned} \underset{d, K_A, K_B, K_C, K_D}{\text{minimize}} \quad & d \\ \text{subject to} \quad & F(P) \prec dI \end{aligned} \tag{2.29}$$

Where $F(P)$ is defined as the left hand side of the matrix inequality in (2.13). Since we have used Theorem 2.1, we know that the optimal value d^* of the problem in (2.29) satisfies $d^* < 0$ and that the closed loop system has an H_∞ norm that is less than γ, i.e., we have that

$$\|H_c(s)\|_\infty < \gamma.$$

2.7 A general algorithm for H_∞ synthesis

Now we summarize the contents of this chapter in an algorithm for H_∞ synthesis using an LMI approach in Algorithm 2.

Algorithm 2 Algorithm for H_∞ synthesis using LMIs

```
Assume that $\gamma$, $n_k$ and system matrices $A, B, C, D$ are given.
1: Find $X, Y \in \mathbb{S}_{++}^{n_x}$ that satisfy (2.21).
2: Recover $P$ from $X$ and $Y$ as described in Section 2.5.
3: Solve (2.29) to get the controller system matrices $K_A, K_B, K_C, K_D$.
```

3 Optimization

An optimization problem is defined by an objective function and a set of constraints. The aim is to minimize the objective function while satisfying the constraints. In this chapter, optimization preliminaries are presented and some methods that can be used to solve optimization problems are outlined. The presentation closely follows relevant sections in Boyd and Vandenberghe [2004] and Nocedal and Wright [2006]. Those familiar with the subject may want to skip directly to the next chapter.

3.1 Nonconvex optimization

In a nonconvex optimization problem, the objective function or the feasible set or both are nonconvex. There is no efficient method to solve a general nonconvex optimization problem, so specialized methods are often used to solve them. It is not possible in general to predict how difficult it is to solve a nonconvex problem and even small problems with few variables may be hard to solve. Nonconvex optimization is a very active research topic today.

3.1.1 Local methods

One approach to use when solving nonconvex optimization problems is to use local methods. A local method searches among the feasible points in a local neighborhood for the optimal point. The drawback for this kind of methods is that a solution that is found may not be the global optimum, and that it may be very sensitive to the choice of the starting point, which in general must be provided or found using some heuristics. There are parameters that may have to be tuned for these algorithms to work. All methods that are presented in this thesis are local methods.

3.1.2 Global methods

In contrast to local methods, global methods find the global minimum of a non-convex minimization problem. It can be very hard to construct such methods and they tend to be very complex and time consuming in general, and thus not efficient in practice. This of course does not mean that global methods cannot be successful for certain subclasses of nonconvex optimization problems. However, we will not consider or analyze any global methods in this thesis.

3.2 Convex optimization

Convex optimization problems belong to a subgroup of nonlinear optimization problems, where both the objective function is convex and the feasible set is convex. This in turn gives some very useful properties. A locally optimal point for a convex optimization problem is the global optimum, see Boyd and Vandenberghe [2004]. Convex problems are in general easy to solve in comparison to nonconvex problems. However, it may not always be the case due to e.g. large scale issues, numerical issues or both.

3.3 Definitions

In this section we present some definitions and concepts that are commonly used in optimization and later on in this thesis.

3.3.1 Convex sets and functions

In this section, convex sets and functions are defined, which are important concepts in optimization since they imply some very useful properties.

Definition 3.1 (Convex set). A set $\mathcal{C} \subseteq \mathbb{R}^n$ is a *convex set* if the line segment between two arbitrary points $x_1, x_2 \in \mathcal{C}$ lies within $\mathcal{C}$, i.e.,

$$\theta x_1 + (1-\theta)x_2 \in \mathcal{C}, \quad \theta \in [0,1], \tag{3.1}$$

The concept is illustrated in Figure 3.1.

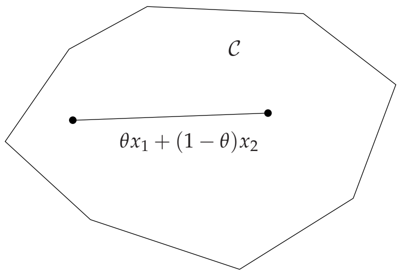

Figure 3.1: *Illustrations of a convex set as defined in Definition 3.1.*

With the definition of a convex set, we can continue with the definition of a convex function.

Definition 3.2 (Convex function). A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a *convex function* if $\operatorname{dom} f$ is a convex set and if for all $x_1, x_2 \in \operatorname{dom} f$ we have that

$$f\big(\theta x_1 + (1-\theta)x_2\big) \leq \theta f(x_1) + (1-\theta)f(x_2), \quad \theta \in [0,1]. \tag{3.2}$$

In plain words, this means that the line segment between the points $\big(x_1, f(x_1)\big)$ and $\big(x_2, f(x_2)\big)$ lies above the graph of f as illustrated in Figure 3.2.

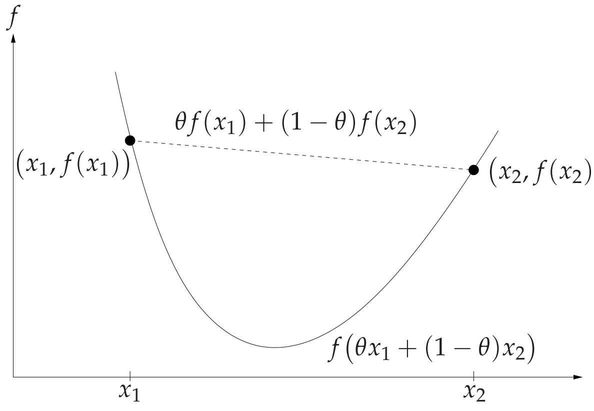
f
$\theta f(x_1) + (1-\theta)f(x_2)$
$(x_1, f(x_1))$
$(x_2, f(x_2))$
$f(\theta x_1 + (1-\theta)x_2)$
x_1
x_2

Figure 3.2: *Illustrations of a convex function as defined in Definition 3.2.*

3.3.2 Cones

In this section we define a special kind of sets, referred to as cones.

Definition 3.3 (Cone). A set $\mathcal{K} \subseteq \mathbb{R}^n$ is a *cone*, if for every $x \in \mathcal{K}$ we have that

$$\theta x \in \mathcal{K}, \quad \theta \geq 0. \tag{3.3}$$

Definition 3.4 (Convex cone). A cone $\mathcal{K} \subseteq \mathbb{R}^n$ is a *convex cone*, if it is convex or equivalently, if for arbitrary points $x_1, x_2 \in \mathcal{K}$ we have

$$\theta_1 x_1 + \theta_2 x_2 \in \mathcal{K}, \quad \theta_1, \theta_2 \geq 0. \tag{3.4}$$

Cones provide the foundation for defining *generalized inequalities*, but before we explain this concept, we first need to define a *proper cone*.

Definition 3.5 (Proper cone). A convex cone $\mathcal{K}$ is a *proper cone*, if the following properties are satisfied:

- $\mathcal{K}$ is closed.
- $\mathcal{K}$ is solid, i.e., it has nonempty interior.
- $\mathcal{K}$ is pointed, i.e., $x \in \mathcal{K}$ and $-x \in \mathcal{K}$ implies $x = 0$.

3.3.3 Generalized inequalities

Definition 3.6 (Generalized inequality). A *generalized inequality* $\succeq_{\mathcal{K}}$ with respect to a proper cone $\mathcal{K}$ is defined as

$$x_1 \succeq_{\mathcal{K}} x_2 \Leftrightarrow x_1 - x_2 \in \mathcal{K}. \tag{3.5}$$

The strict generalized inequality ($\succ_{\mathcal{K}}$) is defined analogously. From now on, the index $\mathcal{K}$ is dropped when the cone is implied from context. We remark that the set of positive semidefinite matrices is a proper cone.

Example 3.1

Let A be a real symmetric matrix of dimension $n \times n$, i.e., $A \in \mathbb{S}^n$. If the proper cone $\mathcal{K}$ is defined as the set of positive semidefinite matrices, then $A \succ 0$ means that A lies strictly in the cone $\mathcal{K}$ which is equivalent to that all eigenvalues of A are strictly positive.

Now we define an optimization problem with matrix inequality constraints using the concepts described previously in this chapter.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \succeq 0, \quad i \in \mathcal{I}, \\ & h_i(x) = 0, \quad i \in \mathcal{E}, \end{aligned} \tag{3.6}$$

where $f_0 : \mathbb{R}^n \to \mathbb{R}$, $f_i : \mathbb{R}^n \to \mathbb{S}^{m_i}$, $h_i : \mathbb{R}^n \to \mathbb{R}^{m_i}$, and where $\succeq$ denotes positive semidefiniteness. The functions are smooth and real-valued and $\mathcal{E}$ and $\mathcal{I}$ are two finite sets of indices.

3.3.4 Logarithmic barrier function and the central path

Definition 3.7 (Generalized logarithm for $\mathbb{S}_+^m$). The function

$$\psi(X) = \log\det X \tag{3.7}$$

is a *Generalized logarithm* for the cone $\mathbb{S}_+^m$ with degree m.

The *Central path* is a concept that emerged as the barrier methods became popular, mostly used in the context of convex optimization. It is also referred to as *The barrier trajectory*.

Definition 3.8 (Central path, logarithmic barrier formulation). The *Central path* of the problem (3.6) is the set of points $x^*(\mu)$, $\mu \geq 0$, that solves the following optimization problems.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) - \mu \sum_{i \in \mathcal{I}} \psi(f_i(x)). \\ \text{subject to} \quad & h_i(x) = 0, \quad i \in \mathcal{E}. \end{aligned} \tag{3.8}$$

This proposes a way to solve (3.6) by iteratively solving (3.8) for a sequence of values μ_k that approaches zero. The optimal point in iteration k can be used as an initial point in iteration $k+1$, see e.g. Boyd and Vandenberghe [2004].

Note that the central path might not be unique in case that the problem is not convex.

3.4 First order optimality conditions

In this section, the first order necessary conditions for x^* to be a local minimizer are stated. When presenting these conditions, the following definition is useful.

Definition 3.9 (Lagrangian function). The *Lagrangian function* (or just *Lagrangian* for short) for the problem in (3.6) is

$$\mathcal{L}(x, Z, v) = f_0(x) - \sum_{i \in \mathcal{I}} \langle Z_i, f_i(x) \rangle - \sum_{i \in \mathcal{E}} \langle v_i, h_i(x) \rangle \tag{3.9}$$

where $Z_i \in \mathbb{S}_+^{m_i}$, $v_i \in \mathbb{R}^{m_i}$ and $\langle A, B \rangle = \operatorname{trace}(A^T B)$ denotes the inner product between A and B.

Assume that the point x^* satisfy assumptions about regularity, see e.g. Forsgren et al. [2002] or Forsgren [2000] for the semidefinite case. We are now ready to state the first-order necessary conditions for (3.6) that must hold at x^* for it to be an optimum.

Theorem 3.1 (First order necessary conditions for optimality, Boyd and Vandenberghe [2004]). *Suppose $x^* \in \mathbb{R}^n$ is any local solution of (3.6), and that the functions f_i, h_i in (3.6) are continuously differentiable. Then there exists Lagrange multipliers $Z_i^* \in \mathbb{S}^{m_i}, i \in \mathcal{I}$ and $v_i^* \in \mathbb{R}^{m_i}, i \in \mathcal{E}$, such that the following conditions are satisfied at (x^*, Z^*, v^*)*

$$\nabla_x \mathcal{L}(x^*, Z^*, v^*) = 0, \tag{3.10a}$$

$$h_i(x^*) = 0, \quad i \in \mathcal{E}, \tag{3.10b}$$

$$f_i(x^*) \succeq 0, \quad i \in \mathcal{I}, \tag{3.10c}$$

$$Z_i^* f_i(x^*) = 0, \quad i \in \mathcal{I}, \tag{3.10d}$$

$$Z_i^* \succeq 0, \quad i \in \mathcal{I}. \tag{3.10e}$$

The conditions (3.10) are sometimes referred to as the Karush-Kuhn-Tucker conditions or the KKT conditions for short. See Karush [1939] or Kuhn and Tucker [1951] for early references.

3.5 Unconstrained optimization

In this section we will present useful background theory for solving the unconstrained optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x), \tag{3.11}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. We assume that the problem has at least one local optimum x^*.

There exist several kinds of methods to solve problems like (3.11), e.g. trust-region methods, line search methods and derivative-free optimization methods, see e.g. Nocedal and Wright [2006], Bertsekas [1995]. We will describe *Newton's method*, which is a line search method. Newton's method is thoroughly described in e.g. Nocedal and Wright [2006], Boyd and Vandenberghe [2004], Bertsekas [1995].

The following theorem defines a necessary characteristic for a locally optimal point of the unconstrained problem in (3.11).

Theorem 3.2 (First order necessary conditions, Nocedal and Wright [2006]). *If x^* is a local minimizer and $f(x)$ is continuously differentiable in an open neighborhood of x^*, then $\nabla f(x^*) = 0$.*

3.5.1 Newton's method

Assume a point $x_k \in \operatorname{dom} f(x)$ is given. Then the second order Taylor approximation (or model) $M_k(p)$ of $f(x)$ at x_k is defined as

$$M_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p. \tag{3.12}$$

Definition 3.10. The *Newton direction* p_K^N is defined by

$$\nabla^2 f(x_k) p_k^N = -\nabla f(x_k). \tag{3.13}$$

The Newton direction has some interesting properties. If $\nabla^2 f(x_k) \succ 0$:

- p_k^N minimizes $M_k(p)$ in (3.12), as illustrated in Figure 3.3.
- $\nabla f(x_k)^T p_k^N = -\nabla f(x_k)^T \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) < 0$ unless $\nabla f(x_k) = 0$, i.e., the Newton step is a descent direction unless x_k is a local optimum.

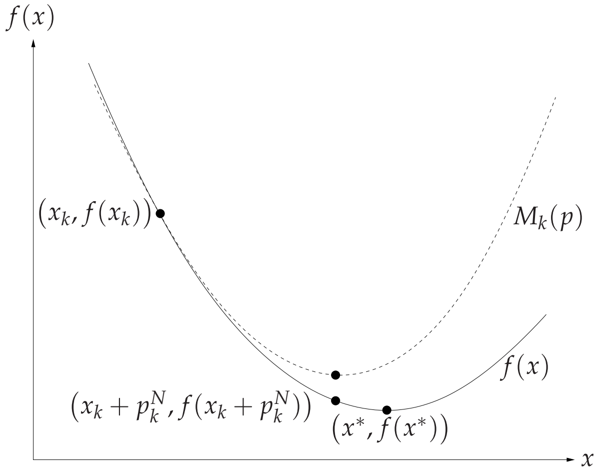

Figure 3.3: *The function $f(x)$ and its second order Taylor approximation $M_k(p)$. The Newton step p_k^N is the minimizer of $M_k(p)$. The optimum x^* minimizes $f(x)$.*

If $\nabla^2 f(x_k) \nsucc 0$, the Newton step does not minimize $M_k(p)$ and is not guaranteed be a descent direction. If that is the case, let B_k be a positive definite approximation of $\nabla^2 f(x)$, i.e. $B_k \in \mathbb{S}_{++}$, and choose the search direction by solving $B_k p_k^M = -\nabla f(x_k)$. Then p_k^M is guaranteed to be a descent direction and minimizes the convex quadratic model

$$M_k^B(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T B_k p. \tag{3.14}$$

Some methods to obtain approximations $B_k \in \mathbb{S}_{++}$ of the Hessian will be presented in Section 3.5.3.

Definition 3.11 (Newton decrement). The *Newton decrement* $\Lambda(x_k)$ is defined as

$$\Lambda(x_k) = \left(\nabla f(x_k)^T B_k^{-1} \nabla f(x_k)\right)^{1/2} = (-\nabla f(x_k)^T p_k^M)^{1/2}. \tag{3.15}$$

The Newton decrement is useful as a stopping criterion. Let $M_k^B(p)$ be the convex quadratic model of f at x_k. Then

$$f(x_k) - \inf_p M_k^B(p) = f(x_k) - M_k^B(p_k^M) = \frac{1}{2}\Lambda(x_k)^2, \tag{3.16}$$

and we see that $\Lambda(x_k)^2/2$ can be interpreted as an approximation of $f(x_k) - p^*$, i.e., a measure of distance from optimality based on the modified quadratic convex approximation of f at x_k.

3.5.2 Line search

When a descent direction p_k^M is determined, we want to find the minimum of the function along that direction. Since $M_k^B(p)$ is only an approximation, p_k^M will not necessarily minimize $f(x_k + p)$. The problem of minimizing the objective function along the search direction can be stated as

$$\underset{\alpha \in (0,1]}{\text{minimize}} \, f(x_k + \alpha p_k^M), \tag{3.17}$$

which is called *exact line search*. An approximate method, but cheaper in computation than exact line search, is *Backtracking line search*. The idea is not to solve (3.17) exactly, but just to reduce f enough, according to some criterion. The procedure is outlined in Algorithm 3 and illustrated in Figure 3.4.

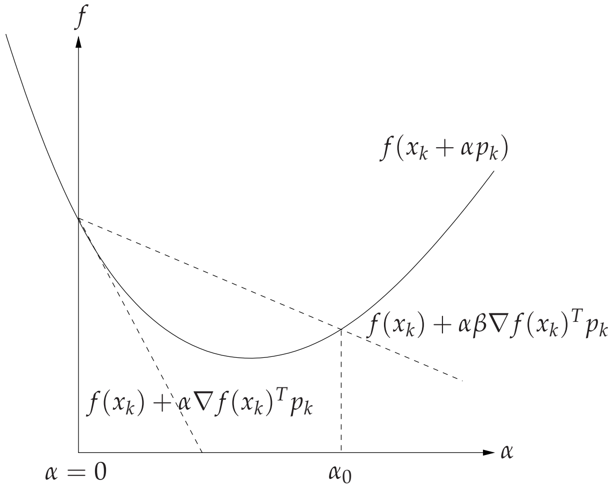
f
$f(x_k + \alpha p_k)$
$f(x_k) + \alpha\beta\nabla f(x_k)^T p_k$
$f(x_k) + \alpha\nabla f(x_k)^T p_k$
α
$\alpha = 0$
α_0

Figure 3.4: *The figure illustrates* Backtracking line search. *The curve shows the function f along the descent direction. The lower dashed line shows the linear extrapolation of f and the upper dashed line has a slope of factor β smaller. The condition for accepting a point is that $\alpha \le \alpha_0$.*

Algorithm 3 Backtracking line search

```
Given a descent direction $p_k$ for $f(x_k)$, $\beta \in (0, 0.5)$, $\gamma \in (0, 1)$.
$\alpha := 1$
while $f(x_k + \alpha p_k) > f(x_k) + \beta\alpha\nabla f(x_k)^T$ do
    $\alpha := \gamma\alpha$
end while
```

3.5.3 Hessian modifications

As mentioned in Section 3.5.1, it is important that the Hessian approximation B_k in (3.14) is positive definite for the calculated step to be a descent direction. There are several ways to modify the Hessian in order to achieve this. The main idea is to make the smallest modification possible in order not to change the curvature more than necessary. We will describe some of these methods in this section.

Eigenvalue computation

A straight-forward approach would be to calculate the eigenvalues of the Hessian $\nabla^2 f(x)$ and choose the approximation to be

$$B = \nabla^2 f(x_k) + dI \succ 0, \quad d \geq 0, \tag{3.18}$$

where $d > \lambda_{\min}\big(\nabla^2 f(x_k)\big)$.

Cholesky factorization

Another suggestion would be to try and calculate the Cholesky factors R of (3.18),

$$R^T R = \nabla^2 f(x_k) + dI \succ 0, \tag{3.19}$$

starting with $d = 0$ and successively trying greater values until we succeed. This will guarantee (with quite high accuracy) that $R^T R \succ 0$ since Cholesky factorization can only be carried out on a positive definite matrix. The disadvantage of this method is that we do not beforehand know which d that is appropriate, and therefore we might have to make many tries before we succeed. A similar but somewhat more elaborate procedure is described in [Nocedal and Wright, 2006, Appendix B].

Modified symmetric indefinite factorization

For any symmetric matrix B it is possible to calculate the factorization

$$P^T BP = LDL^T, \tag{3.20}$$

which is called the *symmetric indefinite factorization* or `LDL` factorization, see e.g. Golub and Van Loan [1996]. The matrix L is a lower triangular matrix, P is a permutation matrix and D is a block diagonal matrix with block sizes of 1×1 and 2×2, which makes it tridiagonal. A procedure that is presented in Cheng and Higham [1998] is to then calculate a modification matrix F such that $L(D + F)L^T$ is positive definite. In order to calculate this modification matrix, one first computes the eigenvalue factorization

$$D = Q\bar{D}Q^T, \tag{3.21}$$

and then calculates the modification matrix

$$F = QEQ^T,$$

where the diagonal matrix E is defined by

$$E_{ii} = \begin{cases} 0, & \text{if } \bar{D}_{ii} \geq \delta, \\ \delta - \bar{D}_{ii}, & \text{if } \bar{D}_{ii} < \delta, \end{cases} \quad i = 1, 2, \ldots \tag{3.22}$$

The matrix F is now the minimal matrix in Frobenius norm such that $D + F \succeq \delta I$. Note that since D is tridiagonal, calculating the eigenvalue factorization in (3.21) is computationally cheap.

3.5.4 Newton's method with Hessian modification

Now we outline a method in Algorithm 4, which is based on Algorithm 9.5 in Boyd and Vandenberghe [2004]. The stopping criterion used is the Newton decrement, defined in Definition 3.11.

Algorithm 4 Newton's method with Hessian modification

```
Given a starting point $x_0 \in \text{dom } f$, tolerance $\epsilon > 0$
loop
    Compute the Newton step by solving $B_k p_k^M = -\nabla f(x_k)$ with approximate Hessian $B_k \succ 0$.
    Compute the Newton decrement $\Lambda(x_k)$.
    if $\Lambda(x_k) < \Lambda_{\text{tol}}$ then
        Exit loop.
    end if
    Choose step size $\alpha_k$ by using Backtracking line search.
    Update iterate. $x_{k+1} := x_k + \alpha_k p_k^M$.
    Set $k := k + 1$.
end loop
```

3.5.5 Quasi-Newton methods

Quasi-Newton methods only require the gradient of the objective function to be supplied at each iteration. By measuring the change in gradient they update the Hessian estimate from the previous iterate in a way that is good enough to produce superlinear convergence.

Especially in cases where the Hessian is unavailable or computationally expensive to calculate, quasi-Newton methods are efficient to use. Among the quasi-Newton methods, the BFGS method (named after its discoverers Broyden [1970], Fletcher [1970], Goldfarb [1970], Shanno [1970]) is perhaps the most well-known and most popular. According to Nocedal and Wright [2006] it is presently considered to be the most effective of all quasi-Newton updating formulae.

BFGS updating

The BFGS updating formulae is derived in a quite intuitive way in Nocedal and Wright [2006], where also additional details are given. Below we will briefly summarize the main ideas in the derivation.

Suppose we have formed the convex quadratic approximate model of the objective function as in (3.14). The minimizer of the convex quadratic model $M_k^B(p)$ can be written explicitly as

$$p_k = -B_k^{-1}\nabla f_k$$

and is used as the search direction to update the iterate as

$$x_{k+1} = x_k + \alpha p_k.$$

where the step length α is chosen in a way such that it satisfies the Wolfe conditions, see Nocedal and Wright [2006]. Define the vectors

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k. \tag{3.23}$$

and let the inverse of B_k be denoted H_k. By requiring that $\nabla_p M_k^B(s_k) = \nabla f_{k+1}$ in (3.14) we get that

$$H_{k+1} y_k = s_k, \tag{3.24}$$

which is usually referred to as the *secant equation*. Since we require that $H_{k+1} \in \mathbb{S}_{++}$, the equation (3.24) is feasible only if y_k and s_k satisfy the *curvature condition*,

$$y_k^T s_k > 0, \tag{3.25}$$

which follows from multiplying (3.24) by y_k^T. The inverse Hessian update H_{k+1} is now obtained as the minimizer to the following optimization problem.

$$\begin{aligned} \underset{H}{\text{minimize}} \quad & \|H - H_k\|_W \\ \text{subject to} \quad & H y_k = s_k, \quad H \in \mathbb{S}_{++} \end{aligned} \tag{3.26}$$

where W is any matrix satisfying $W s_k = y_k$ and $\|A\|_W = \|W^{1/2} A W^{1/2}\|_F$ is the weighted Frobenius norm. The objective function in (3.26) reflects the desire that the updated (inverse) Hessian should in some sense be close to the (inverse) Hessian in the previous iteration. The unique solution to (3.26) is given by

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{3.27}$$

and by applying the Sherman-Morrison-Woodbury formula (see e.g. [Nocedal and Wright, 2006, Appendix A]) on (3.27) we obtain

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{3.28}$$

which is the formula that probably is the most known and used for BFGS updating. Other references on BFGS are e.g. the books by Fletcher [1987] and Dennis, Jr. and Schnabel [1983], where the second book of the two also describes an update of the Cholesky factor of B_k.

Damped BFGS updating

The curvature condition (3.25) will not always hold for nonconvex functions $f(x)$ or if we do not choose step sizes according to the Wolfe conditions. In those cases one can modify the BFGS update by modifying the definition of y_k. Below we will briefly describe this modified procedure which is known as *damped BFGS updating*, which is described in [Nocedal and Wright, 2006, Procedure 18.2].

Given the symmetric and positive definite matrix B_k, define s_k and y_k as in (3.23) and let

$$r_k = \theta_k y_k + (1 - \theta_k) B_k s_k, \tag{3.29}$$

where θ is defined as follows.

$$\theta_k = \begin{cases} 1 & \text{if } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ (0.8 s_k^T B_k s_k)/(s_k^T B_k s_k - s_k^T y_k) & \text{if } s_k^T y_k < 0.2 s_k^T B_k s_k \end{cases} \tag{3.30}$$

Then obtain the updated Hessian estimate B_{k+1} using the formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{r_k r_k^T}{r_k^T s_k}, \tag{3.31}$$

which is similar to (3.28) but with y_k replaced by r_k. This guarantees that B_{k+1} is positive definite and one can verify, using the equations above, that when $\theta_k \neq 1$ we have

$$s_k^T r_k = 0.2 s_k^T B_k s^k > 0. \tag{3.32}$$

3.6 Constrained optimization

We now turn to constrained optimization. The approach we will present here is the primal-dual approach. As the name suggests, the approach uses both the original primal variables as well as dual variables. One way to derive this approach is to start from the KKT conditions (3.10) and add slack variables S_i to the inequalities. We get the following equations

$$\nabla_x \mathcal{L}(x^*, Z^*, \nu^*) = 0, \tag{3.33a}$$
$$h_i(x^*) = 0, \quad i \in \mathcal{E}, \tag{3.33b}$$
$$f_i(x^*) - S_i^* = 0, \quad i \in \mathcal{I}, \tag{3.33c}$$
$$Z_i^* S_i^* = 0, \quad i \in \mathcal{I}. \tag{3.33d}$$
$$Z_i^* \geq 0, \quad S_i^* \geq 0, \quad i \in \mathcal{I}, \tag{3.33e}$$

and we note that this is now a collection of equality constraints, except for the inequality constraints (3.33e). Below we will describe a method for finding a solution to a system of equations.

3.6.1 Newton's method for nonlinear equations

Recall from Section 3.5 that Newton's method is based on the principle of minimizing a second order model, which is created by taking the first three terms of the Taylor series approximation, see (3.12), of the function around the current iterate x_k. The Newton step is the vector that minimizes that model. When solving nonlinear equations, Newton's method is derived in a similar way, the difference is that a linear model is used.

Assume that $r : \mathbb{R}^n \to \mathbb{R}^n$ and that we want to find x^* such that $r(x^*) = 0$. Assume we have a point x_k such that $r(x_k) \approx 0$. Denote the Jacobian of r as $J(x)$ and we can create a linear model $M_k(p)$ of $r(x_k + p)$ as

$$M_k(p) = r(x_k) + J(x_k)p. \tag{3.34}$$

If we choose $p_k = -J(x_k)^{-1} r(x_k)$ we see that $M_k(p_k) = 0$ and this choice of p_k is Newton's method in its pure form. A structured description is given in Algorithm 5.

Algorithm 5 Newton's method for nonlinear equations

```
Given $x_0$
**for** $k = 0, 1, 2, \ldots$ **do**
    Calculate a solution $p_k$ to the Newton equations $J(x_k)p_k = -r(x_k)$.
    $x_{k+1} := x_k + p_k$
**end for**
```

If any positivity constraints are present, such as (3.33e), we can use line search techniques to find a feasible point along the search direction.

Remark 3.1. Note that Newton's method for nonlinear equations can be extended to functions $r : \mathbb{S}^n \to \mathbb{S}^n$ by applying standard differential calculus for matrices.

Remark 3.2. Note that the domain and range of the function defined by the left hand side of (3.33) are not the same spaces, since the product in (3.33d) is not symmetric in general. This means that Newton's method is not directly applicable. A solution is to apply a symmetry transformation to the left hand side in (3.33d) before Newton's method is applied. We will come back to this issue in Section 6.4.2.

3.6.2 Central path

With Remark 3.2 in mind, the KKT system in (3.33) can be solved by applying Newton's method for equations while ignoring the inequality in (3.33e). However, the inequality in (3.33e) should be taken into consideration when computing the step length. Unfortunately, due to (3.33d), only very short steps can be taken. As a result the convergence will be very slow. In order to be able to take longer steps, we can relax the condition in (3.33d). A way to do this relaxation is to use the central path.

Definition 3.12 (Central path, primal-dual formulation). The *central path* is defined as the set of solution points, parameterized by μ, for which

$$\nabla_x \mathcal{L}(x, Z, \nu) = 0, \tag{3.35a}$$
$$h_i(x) = 0, \quad i \in \mathcal{E}, \tag{3.35b}$$
$$f_i(x) - S_i = 0, \quad i \in \mathcal{I}, \tag{3.35c}$$
$$Z_i S_i = \mu I_{n_i}, \quad i \in \mathcal{I}. \tag{3.35d}$$
$$Z_i \succeq 0, \quad S_i \succeq 0, \quad i \in \mathcal{I}, \tag{3.35e}$$

where $\mu \geq 0$ and where $\mathcal{L}(x, Z, \nu)$ is defined in Definition 3.9.

The idea is to start with a $\mu = \mu_0$ and let μ tend to zero, in order to make the limit solution satisfy the KKT conditions. Methods following this principle are called *path-following* methods.

Example 3.2

Consider the convex optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \succeq 0 \end{aligned} \tag{3.36}$$

where $c, x \in \mathbb{R}^n$ and $F_i \in \mathbb{S}^n_+, \forall i$. Assume that (3.36) is strictly feasible. Using Definition 3.12, the central path for (3.36) is the set of solution points, parameterized by μ, for which

$$\begin{aligned} \nabla_x \mathcal{L}(x, Z)_i = c_i - \text{trace}(Z F_i) &= 0, \quad \forall i \\ F(x) - S &= 0, \\ ZS &= \mu I, \\ Z \succeq 0, \quad S &\succeq 0, \end{aligned} \tag{3.37}$$

where $Z, S \in \mathbb{S}^n$. If we now instead use Definition 3.8, the central path of (3.36) is the set of solution points $x^*(\mu)$ that solves

$$\underset{x}{\text{minimize}} \quad c^T x - \mu \log\det F(x). \tag{3.38}$$

Since (3.38) is a convex problem, by using Theorem 3.1 or Theorem 3.2 we know that global minimizers satisfy

$$\begin{aligned} c_i - \mu\, \text{trace}(F^{-1}(x) F_i) &= 0, \quad \forall i, \\ F(x) &\succeq 0. \end{aligned} \tag{3.39}$$

By introducing a slack variable $S \succeq 0$ and defining the dual variable $Z = \mu F^{-1}(x)$, we actually obtain exactly the same equations as in (3.37) which shows that the two definitions coincide for the problem in (3.36).

3.6.3 A generic primal-dual interior point method

Using what has been presented in recent sections, we can outline a simple algorithm that produces iterates that tend towards a solution of the KKT conditions.

Write the equations (3.35a)–(3.35d) on the form $r(x, Z, \nu, S, \mu) = 0$. Assume that an initial point (x, Z, ν, S) is available that satisfies the condition (3.35e). Compute the next iterate as

$$(x^+, Z^+, \nu^+, S^+) = (x, Z, \nu, S) + \alpha(\Delta x, \Delta Z, \Delta \nu, \Delta S), \tag{3.40}$$

where $(\Delta x, \Delta Z, \Delta \nu, \Delta S)$ is the search direction and α is a step length such that the next iterate S^+, Z^+ satisfies (3.35e). We outline what is just described in Algorithm 6.

Algorithm 6 A generic interior point method

```
Choose an initial point $z = (x, Z, v, S)$ that satisfies (3.35e).
Choose error tolerance $\epsilon$.
Choose a starting value for $\mu$.
while $\|r(x, Z, v, S, \nu)\|_2 > \epsilon$ do
    Choose a search direction $(\Delta x, \Delta Z, \Delta \nu, \Delta S)$
    Find $\alpha > 0$ such that $Z^+, S^+$ are feasible in (3.40).
    Update iterate using (3.40).
    Choose a new $\mu$.
end while
```

The choice of the parameter μ is a tricky issue, which has no obvious solution. Different choices have given rise to different methods, e.g. predictor-corrector method and the method by Mehrotra [1992].

3.7 Penalty methods and augmented Lagrangian methods

In this section we will discuss some smooth methods that are useful when considering the problem of minimizing a function subject to equality constraints. We follow the presentation in [Nocedal and Wright, 2006, Chapter 17.1 and 17.3] quite closely, where only penalty methods for equality constraints are presented.

3.7.1 The quadratic penalty method

The quadric penalty function was first proposed by Courant [1943]. Consider the equality constrained problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0. \quad i \in \mathcal{E} \end{aligned} \tag{3.41}$$

We can formulate a related problem by using a quadratic penalty function as in

$$\underset{x}{\text{minimize}} \quad f(x) + \frac{\mu}{2}\sum_{i\in\mathcal{E}} c_i^2(x), \tag{3.42}$$

where μ is a the *penalty parameter*. By squaring the equality constraint violations and scaling them by $\mu/2$, the minimizer of (3.42) will tend towards being a feasible solution of (3.41) as $\mu \to \infty$. An approach would be to consider a sequence of increasing values $\{\mu_k\}$ as $k \to \infty$ and for each value of μ_k, solve (3.42) using unconstrained optimization. Each solution x_k obtained using μ_k would serve as a good starting point for the following minimization problem when searching for the minimizer μ_{k+1}. If $\mu_k \to \infty$ as $k \to \infty$ it can be proved that every limit point x^* of the sequence $\{x_k\}$ is a global solution of the problem (3.41), see [Nocedal and Wright, 2006, Theorem 17.1].

3.7.2 The augmented Lagrangian method

Sometimes also referred to as the *method of multipliers*, the augmented Lagrangian method was proposed independently by Hestenes [1969] and Powell [1969]. They were considering an equality constrained optimization problem and it was shown that the numerical properties were improved compared to using regular penalty methods, e.g. the one presented in Section 3.7.1. More details regarding multiplier and penalty methods can be found in the books by Bertsekas [1982, 1995], Fletcher [1987], Nocedal and Wright [2006] and the survey paper by Rockafellar [1993].

Motivation and definition

When using quadratic penalty methods, the minimizers of the problem (3.42) do not quite satisfy the equality constraints of (3.41) unless in the limit where $\mu_k \to \infty$. The augmented Lagrangian method was shown to produce approximate minimizers that more closely satisfy the equality constraints in (3.41), even for moderate values of the penalty parameters μ_k.

The augmented Lagrangian for the optimization problem (3.41) is defined as

$$\mathcal{L}_A(x, \lambda, \mu) = f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2}\sum_{i\in\mathcal{E}} c_i^2(x), \tag{3.43}$$

where λ_i is an estimate the of Lagrange multiplier. We can see that the augmented Lagrangian (3.43) differs from the quadratic penalty function by including a sum of $\lambda_i c_i(x)$.

Algorithmic framework

If x_k is the approximate minimizer of $\mathcal{L}_A(x, \lambda^k, \mu_k)$ then by the optimality conditions for unconstrained minimization, Theorem 3.2, we have that

$$\nabla_x \mathcal{L}_A(x, \lambda) = \nabla_x f(x) + \sum_{i\in\mathcal{E}} \left(\lambda_i^k + \mu_k c_i(x)\right)\nabla c_i(x) \approx 0, \tag{3.44}$$

which we can compare to the first optimality condition for constrained optimization, Theorem 3.1, for (3.41) which is

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla_x f(x) + \sum_{i \in \mathcal{E}} \lambda_i \nabla c_i(x) \approx 0. \tag{3.45}$$

where we can draw the conclusion that at the optimum x_k of (3.43), we have that

$$\lambda_i \approx \lambda_i^k + \mu_k c_i(x_k), \quad \forall i \in \mathcal{E}. \tag{3.46}$$

This suggests an updating rule for the Lagrange multiplier estimates

$$\lambda_i^{k+1} = \lambda_i^k + \mu_k c_i(x_k), \quad \forall i \in \mathcal{E}. \tag{3.47}$$

We will now summarize the ideas from this section in Algorithm 7, see also [Nocedal and Wright, 2006, Framework 17.3].

Algorithm 7 An augmented Lagrangian method for equality constraints

```
Choose $\mu_0 > 0$, tolerance $\tau_0 > 0$, starting point $x_0^s$ and $\lambda^0$.
for $k = 0, 1, 2, \dots$ do
  Start at $x_k^s$ and minimize $\mathcal{L}_A(x, \lambda^k, \mu_k)$ approximately to find $x_k$ such that
  $\|\nabla_x \mathcal{L}_A(x_k, \lambda^k, \mu_k)\| \le \tau_k$.
  if some convergence test for (3.41) is satisfied then
    Stop with approximate solution $x_k$.
  end if
  Calculate the updated Lagrange multipliers $\lambda^{k+1}$ using (3.47).
  Choose new penalty parameter $\mu_{k+1} \ge \mu_k$.
  Let the starting point for the next iteration be $x_{k+1}^s = x_k$.
  Set tolerance for next iteration, $\tau_{k+1}$.
end for
```

This completes the chapter of optimization preliminaries.

4 The characteristic polynomial and rank constraints

The approach to low order H_∞ control in this thesis involves the characteristic polynomial of the matrix $I - XY$ in (2.21d). In this chapter we will explain how the rank constraint of a matrix can be expressed as a quotient of coefficients of its characteristic polynomial. As a consequence this enables gradient methods to be applied to the resulting smooth optimization problem. The reference on this topic is Helmersson [2009].

The chapter is structured as follows. We begin in Section 4.1 by presenting two lemmas regarding properties of the coefficients of the characteristic polynomial. In Section 4.2 we present a modified version of Theorem 2.1, and in Section 4.3 we will reformulate the solvability conditions of Theorem 2.1 as an optimization problem. In Section 4.4 we will describe how to compute the coefficients of the characteristic polynomial of $I - XY$ and their derivatives. The chapter is concluded by showing how the problem of finding a dynamic controller (i.e., a controller with nonzero number of states) can be equivalently rewritten as a problem of finding a static controller (i.e., a controller with no states). Additionally, some advantages and disadvantages of using such an approach in the context of low order H_∞ controller synthesis will be discussed.

4.1 A polynomial criterion

The characteristic polynomial of a matrix $Z \in \mathbb{R}^{n_x \times n_x}$ is defined by

$$\det(\lambda I - Z) = \sum_{i=0}^{n_x} c_i(Z)\lambda^i, \tag{4.1}$$

where the coefficients $c_i(Z)$ are polynomial functions of the elements in the matrix Z. Then we have that e.g.

$$c_{n_x}(Z) = 1, \quad c_{n_x-1}(Z) = -\operatorname{trace}(Z) \quad \text{and} \quad c_0 = (-1)^{n_x} \det(Z).$$

Some other nontrivial properties of the coefficients of the characteristic polynomial in the case where the matrix Z is positive semidefinite are presented in Lemma 4.1 and Lemma 4.2 below.

Lemma 4.1 (Helmersson [2009]). *Let $Z \in \mathbb{R}^{n_x \times n_x}$ be a matrix with real nonnegative eigenvalues, $\lambda_i(Z) \geq 0$, and let $c_i(-Z), i = 1, 2, \ldots, n_x$, be the coefficients of the characteristic polynomial of $-Z$ as defined in* (4.1). *Then, the following statements are equivalent if $n_k < n_x$:*

1. $c_{n_x-n_k-1}(-Z) = 0$,
2. $\operatorname{rank} Z \leq n_k$.

Lemma 4.2 (Helmersson [2009]). *Let $Z \in \mathbb{R}^{n_x \times n_x}$ be a matrix with real nonnegative eigenvalues ordered by $\lambda_1(Z) \geq \lambda_2(Z) \geq \ldots \geq \lambda_{n_x}(Z) \geq 0$ and assume that $c_{n_x-n_k}(-Z) \neq 0$. Then, for $n_k < n_x$, the following relations hold:*

$$\frac{1}{n_k + 1} \sum_{i=n_k+1}^{n_x} \lambda_i(Z) \leq \frac{c_{n_x-n_k-1}(-Z)}{c_{n_x-n_k}(-Z)} \leq \sum_{i=n_k+1}^{n_x} \lambda_i(Z), \tag{4.2}$$

or equivalently

$$\frac{c_{n_x-n_k-1}(-Z)}{c_{n_x-n_k}(-Z)} \leq \sum_{i=n_k+1}^{n_x} \lambda_i(Z) \leq (n_k + 1)\frac{c_{n_x-n_k-1}(-Z)}{c_{n_x-n_k}(-Z)}. \tag{4.3}$$

Remark 4.1. For future reference, note that the quotient in (4.2) is nonnegative if Z is positive semidefinite.

From experience, better numeric properties are obtained by using the quotient in (4.2) instead of only the nominator when searching for matrices Z with low rank. Insights on this can be gained from noting that the quotient in (4.2) is bounded both from above and from below by the sum of eigenvalues of Z that should be zero for $\operatorname{rank} Z \leq n_k$ to hold.

4.2 Replacing the rank constraint

Let us define $Z = XY - I$, where $X, Y \in \mathbb{S}^{n_x}$. The following corollary can be proved.

Corollary 4.1. *Assume that $X, Y \in \mathbb{S}^{n_x}$ satisfy*

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \succeq 0, \quad \textit{or equivalently,} \quad XY - I \succeq 0.$$

Then the following statements are equivalent:

1. $\operatorname{rank}(XY - I) = n_k$,

2. $\dfrac{c_{n_x - n_k - 1}(I - XY)}{c_{n_x - n_k}(I - XY)} = 0$, *and* $c_{n_x - n_k}(I - XY) \neq 0$.

Proof: Note that $Z = XY - I$ has real eigenvalues, since it is similar to a symmetric matrix. From Lemma 4.1 we have that

$$\operatorname{rank}(XY - I) = n_k \Leftrightarrow \begin{cases} \operatorname{rank}(XY - I) \leq n_k \\ \operatorname{rank}(XY - I) > n_k - 1 \end{cases} \Leftrightarrow \begin{cases} c_{n_x - n_k - 1}(I - XY) = 0 \\ c_{n_x - n_k}(I - XY) \neq 0 \end{cases}$$

which completes the proof. □

By combining Theorem 2.1 and Corollary 4.1, we can now formulate a theorem for synthesis of H$_\infty$ controllers for continuous plants that only involves smooth functions.

Theorem 4.1 (H$_\infty$ controllers for continuous plants). *The problem of finding a linear controller of order $n_k < n_x$ such that the closed loop system H_c is stable and such that $\|H_c(s)\|_\infty < \gamma$, is solvable if there exist $X, Y \in \mathbb{S}_{++}^{n_x}$, which satisfy*

$$\begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} XA + A^T X & XB_1 & C_1^T \\ B_1^T X & -\gamma I & D_{11}^T \\ C_1 & D_{11} & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix} \prec 0, \tag{4.4a}$$

$$\begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} AY + YA^T & YC_1^T & B_1 \\ C_1 Y & -\gamma I & D_{11} \\ B_1^T & D_{11}^T & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix} \prec 0, \tag{4.4b}$$

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \succeq 0, \tag{4.4c}$$

$$\frac{c_{n_x - n_k - 1}(I - XY)}{c_{n_x - n_k}(I - XY)} = 0, \tag{4.4d}$$

$$c_{n_x - n_k}(I - XY) \neq 0, \tag{4.4e}$$

where $\mathcal{N}_X$ and $\mathcal{N}_Y$ denote any base of the null-spaces of $\begin{pmatrix} C_2 & D_{21} \end{pmatrix}$ and $\begin{pmatrix} B_2^T & D_{12}^T \end{pmatrix}$ respectively.

Proof: Combine Theorem 2.1 and Corollary 4.1. □

Remark 4.2. In practice we do not consider the constraint in (4.4e) when looking for a solution, but instead we just check afterwards that this condition is satisfied. In case not, Lemma 4.1 implies that $\operatorname{rank}(XY - I) \leq n_k - 1$. Thus a controller with order less than n_k can be found.

4.3 Problem reformulations

In this section we will formulate the problem of finding X, Y that satisfy (4.4) as a minimization problem. We will also show how to reformulate the LMIs in (4.4a)–(4.4c) in a more compact way.

4.3.1 An optimization problem formulation

Recall from Remark 4.1 that the quotient in (4.4d) is nonnegative as long as we have $Z = XY - I \succeq 0$. This implies that the problem of finding X, Y such that the conditions in (4.4) are satisfied can be formulated as the following optimization problem, where n_k and γ is given.

$$\begin{aligned} \underset{X,Y}{\text{minimize}} \quad & \frac{c_{n_x-n_k-1}(I - XY)}{c_{n_x-n_k}(I - XY)} \\ \text{subject to} \quad & (X, Y) \in \mathbb{X}, \end{aligned} \tag{4.5}$$

where $\mathbb{X}$ denotes the convex set defined by the LMIs in (4.4a)–(4.4c). If the solution X^*, Y^* to the problem in (4.5) is such that the conditions in (4.4d) and (4.4e) hold, we have that X^*, Y^* satisfy (4.4).

4.3.2 A reformulation of the LMI constraints

Let us define the half-vectorization operator as follows.

Definition 4.1 (Half-vectorization). Let

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Then

$$\operatorname{vech}(A) = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{n1} & a_{22} & \dots & a_{n2} & a_{33} & \dots & a_{nn} \end{pmatrix}^T,$$

i.e., the operator vech stacks the columns of A from the diagonal elements downwards in a column vector. See Lütkepohl [1996] for properties and details.

Remark 4.3. Note that $\operatorname{vech}(A)$ does not require A to be symmetric. However, if it is not, some information will be lost.

For $X, Y \in \mathbb{S}^{n_x}$ let us make the variable substitution

$$x = \begin{pmatrix} \operatorname{vech}(X) \\ \operatorname{vech}(Y) \end{pmatrix} \in \mathbb{R}^m, \tag{4.6}$$

where $m = n_x(n_x + 1)$. By choosing appropriate symmetric matrices $A_i^{(j)}$ and $C^{(j)}$

we can rewrite the problem in (4.5) equivalently as

$$\underset{x}{\text{minimize}} \quad \frac{c_{n_x - n_k - 1}(x)}{c_{n_x - n_k}(x)} \tag{4.7a}$$

$$\text{subject to} \quad C^{(j)} - \sum_{i=1}^{m} A_i^{(j)} x_i \succ 0, \quad j = 1, 2, 3, \tag{4.7b}$$

where $c_i(x) = c_i(I - XY)$ is an abuse of notation. Note that (4.7b) replaces the LMIs (4.4a)–(4.4c). The index j corresponds to each of the three LMIs. This problem formulation will be used in Chapter 5 and Chapter 6 when we assume that a value of γ is given.

Remark 4.4. Note that the nonstrict matrix inequality in (4.4c) is replaced by a strict matrix inequality in (4.7b). The LMI in (4.4c) is strictly feasible. To see this, let e.g. $X = Y = \alpha I, \alpha > 1$. Thus by solving the optimization problem in (4.7) with strict inequalities and interpreting minimize as seeking an infimum we will obtain a solution to problem in (4.5) involving a nonstrict inequality, see Section 2.5 in Boyd et al. [1994].

4.4 Computing the characteristic polynomial of I − XY and its derivatives

To determine the characteristic polynomial of a square matrix Z symbolically, one calculates $\det(\lambda I - Z)$ using basic linear algebra. However, when Z is of higher dimension, it might not be efficient to use the symbolic derivation of the characteristic polynomial. Instead numeric approaches can be used, e.g. the MATLAB command `poly` or the implementation suggested in Berkowitz [1984]. However, the calculations of the gradient and Hessian of the coefficients in the characteristic polynomial are not as straight-forward to calculate and will be explained in the next few sections.

4.4.1 Derivative calculations

In this section, we will present how to compute the derivatives of a coefficient of the characteristic polynomial with respect to X_{ij} and Y_{ij} for the special case when $Z = I - XY$.

Gradient calculations

The gradient expressions are derived in [Helmersson, 2009, Section 3.1]. A simplified presentation is provided below.

Let $C_i \in \mathbb{R}^{n_x \times n_x}$ and define

$$\begin{aligned}
C_{n_x} &= 0, \\
C_{n_x-1} &= I, \\
C_{n_x-2} &= Z + c_{n_x-1}(Z)I, \\
&\vdots \\
C_k &= C_{k+1}Z + c_{k+1}(Z)I, \\
&\vdots \\
C_0 &= C_1 Z + C_1(Z)I.
\end{aligned}$$

Then the first-order derivatives of a coefficient $c_k(Z)$ in (4.1) are given by

$$\frac{\partial c_k}{\partial X_{ij}} = \operatorname{trace} C_k E_{ij} Y, \tag{4.8a}$$

$$\frac{\partial c_k}{\partial Y_{ij}} = \operatorname{trace} C_k X E_{ij}, \tag{4.8b}$$

with the matrix E_{ij} defined as

$$E_{ij} = \begin{cases} e_i e_j^T, & \text{if } i = j, \\ e_i e_j^T + e_j e_i^T, & \text{if } i \neq j, \end{cases} \tag{4.9}$$

where $e_i, e_j \in \mathbb{R}^{n_x}$ are the ith and jth unit vectors respectively. The expressions in (4.8) can be simplified using the fact that $\operatorname{trace} ABC = \operatorname{trace} CAB = \operatorname{trace} BCA$ for matrices A, B, C with compatible dimensions. For instance if $i \neq j$ we get that

$$\frac{\partial c_k}{\partial X_{ij}} = e_j^T Y C_k e_i + e_i^T Y C_k e_j, \tag{4.10}$$

which is the same as extracting two elements from the product of Y and C_k and summing them.

Hessian calculations

The Hessian expressions are derived in [Helmersson, 2009, Section 3.2]. It holds that

$$\begin{aligned}
\frac{\partial^2 c_k}{\partial X_{ij} \partial X_{pq}} = {} & (\operatorname{trace} C_{n_x-1} E_{pq} Y)(\operatorname{trace} Z^{n_x-k-2} E_{ij} Y) - \operatorname{trace} C_{n_x-1} E_{pq} Y Z^{n_x-k-2} E_{ij} Y \\
& + \dots \\
& + (\operatorname{trace} C_{k+2} E_{pq} Y)(\operatorname{trace} Z E_{ij} Y) - \operatorname{trace} C_{k+2} E_{pq} Y Z E_{ij} Y \\
& + (\operatorname{trace} C_{k+1} E_{pq} Y)(\operatorname{trace} E_{ij} Y) - \operatorname{trace} C_{k+1} E_{pq} Y E_{ij} Y
\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 c_k}{\partial Y_{ij}\partial Y_{pq}} &= (\operatorname{trace} C_{n_x-1}XE_{pq})(\operatorname{trace} Z^{n_x-k-2}XE_{ij}) - \operatorname{trace} C_{n_x-1}XE_{pq}Z^{n_x-k-2}XE_{ij}\\ &+ \dots\\ &+ (\operatorname{trace} C_{k+2}XE_{pq})(\operatorname{trace} ZXE_{ij}) - \operatorname{trace} C_{k+2}XE_{pq}ZXE_{ij}\\ &+ (\operatorname{trace} C_{k+1}XE_{pq})(\operatorname{trace} XE_{ij}) - \operatorname{trace} C_{k+1}XE_{pq}XE_{ij}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 c_k}{\partial X_{ij}\partial Y_{pq}} &= (\operatorname{trace} C_{n_x-1}XE_{pq})(\operatorname{trace} Z^{n_x-k-2}E_{ij}Y) - \operatorname{trace} C_{n_x-1}XE_{pq}Z^{n_x-k-2}E_{ij}Y\\ &+ \dots\\ &+ (\operatorname{trace} C_{k+2}XE_{pq})(\operatorname{trace} ZE_{ij}Y) - \operatorname{trace} C_{k+2}XE_{pq}ZE_{ij}Y\\ &+ (\operatorname{trace} C_{k+1}XE_{pq})(\operatorname{trace} E_{ij}Y) - \operatorname{trace} C_{k+1}XE_{pq}E_{ij}Y\\ &+ \operatorname{trace} C_kE_{ij}E_{pq}\end{aligned}$$

The number of terms in each expression is dependent on k. For $k = n_x - 1$ the first two expressions contain no terms at all, while the third expression will contain only one term, which corresponds to searching for a zeroth-order controller ($n_k = 0$) while $k = n_x - 2$ corresponds to a first-order controller ($n_k = 1$). The smaller k is, the more terms will enter the expressions.

4.4.2 Computational complexity

When implementing the gradient and Hessian calculations, it is important to utilize the structure of the problem. A careless, straightforward implementation that carries out all the matrix multiplications might end up using $\mathcal{O}(n_x^5)$ operations for the gradient calculations and $\mathcal{O}(n_x^7)$ operations for the Hessian calculations, see Ankelhed [2011]. By utilizing the structure in the problem, e.g. by using the trace manipulations as in (4.10), it is possible to reduce this complexity at least down to $\mathcal{O}(n_x^3)$ and $\mathcal{O}(n_x^4)$ for the gradient and Hessian calculations, respectively. For more details on implementational issues, refer to Helmersson [2009] and Ankelhed [2011].

4.4.3 Derivatives of quotient

In the previous section we showed how to calculate the gradient and Hessian of one coefficient. However the gradient and Hessian of the quotient of two coefficients are needed. These are obtained by applying differential calculus as follows. First define the quotient $\hat{c}(x)$ as

$$\hat{c}(x) = \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)}.$$

Then the gradient of $\hat{c}(x)$ is given by

$$\nabla_x \hat{c}(x) = \frac{1}{c_{n_x-n_k}(x)} \nabla_x c_{n_x-n_k-1}(x) - \frac{c_{n_x-n_k-1}(x)}{c^2_{n_x-n_k}(x)} \nabla_x c_{n_x-n_k}(x)$$

and the Hessian of $\hat{c}(x)$ is given by

$$\begin{aligned}\nabla^2_{xx} \hat{c}(x) = &\frac{1}{c_{n_x-n_k}(x)} \nabla^2_{xx} c_{n_x-n_k-1}(x) - \frac{c_{n_x-n_k-1}(x)}{c^2_{n_x-n_k}(x)} \nabla^2_{xx} c_{n_x-n_k}(x) \\ &+ \frac{2c_{n_x-n_k-1}(x)}{c^3_{n_x-n_k}(x)} \left(\nabla_x c_{n_x-n_k}(x) \nabla^T_x c_{n_x-n_k}(x)\right) \\ &- \frac{1}{c^2_{n_x-n_k}(x)} \left(\nabla_x c_{n_x-n_k-1}(x) \nabla^T_x c_{n_x-n_k}(x) + \nabla_x c_{n_x-n_k}(x) \nabla^T_x c_{n_x-n_k-1}(x)\right).\end{aligned}$$

4.5 Formulating a dynamic controller problem as a static controller problem

As a comparison we will now present another approach to reduced order control synthesis. It has been shown, e.g. in El Ghaoui et al. [1997], Syrmos et al. [1997], and Nett et al. [1989] that a dynamic controller problem ($n_k > 0$) can be written as an equivalent problem of finding a static controller ($n_k = 0$) by augmenting the system. This procedure is used by several other methods for H_∞ controller synthesis, e.g. Apkarian et al. [2003], Apkarian and Noll [2006b] and Gumussoy and Overton [2008a] to name a few. This procedure is described below.

Let the system H be defined by

$$H: \quad \left(\begin{array}{c} \dot{x} \\ \hline z \\ y \end{array}\right) = \left(\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right) \left(\begin{array}{c} x \\ \hline w \\ u \end{array}\right),$$

which is a restatement of (2.1). Assume we augment the state vector with $x_K \in \mathbb{R}^{n_k}$, the control signal vector with $u_K \in \mathbb{R}^{n_k}$ and the measurement signal vector with $y_K \in \mathbb{R}^{n_k}$ in the following way.

$$\tilde{x} = \begin{pmatrix} x \\ x_K \end{pmatrix}, \qquad \tilde{u} = \begin{pmatrix} u_K \\ u \end{pmatrix}, \qquad \tilde{y} = \begin{pmatrix} y_K \\ y \end{pmatrix} \tag{4.11}$$

In plain words, we add n_k states, n_k control signals and n_k outputs to the system. The system matrices for the augmented system are chosen as follows.

$$\begin{aligned} &\tilde{A} = \begin{pmatrix} A & 0_{n_x\times n_k} \\ 0_{n_k\times n_x} & 0_{n_k\times n_k} \end{pmatrix}, \quad \tilde{B}_1 = \begin{pmatrix} B_1 \\ 0_{n_k\times n_w} \end{pmatrix}, \quad \tilde{B}_2 = \begin{pmatrix} 0_{n_x\times n_k} & B_2 \\ I_{n_k\times n_k} & 0_{n_k\times n_u} \end{pmatrix} \\ &\tilde{C}_1 = \begin{pmatrix} C_1 & 0_{n_z\times n_k} \end{pmatrix}, \quad \tilde{D}_{11} = D_{11}, \quad \tilde{D}_{12} = \begin{pmatrix} 0_{n_z\times n_k} & D_{12} \end{pmatrix} \\ &\tilde{C}_2 = \begin{pmatrix} 0_{n_k\times n_x} & I_{n_k\times n_k} \\ C_2 & 0_{n_y\times n_k} \end{pmatrix}, \quad \tilde{D}_{21} = \begin{pmatrix} 0_{n_k\times n_w} \\ D_{21} \end{pmatrix}, \quad \tilde{D}_{22} = 0_{(n_k+n_y)\times(n_k+n_u)}. \end{aligned} \tag{4.12}$$

Then assume that we have synthesized a static output feedback controller $\tilde{K}$ for the system in (4.12). The control signal $\tilde{u}$ is then computed as

$$\tilde{u} = \tilde{K}\tilde{y} = \begin{pmatrix} K_A & K_B \\ K_C & K_D \end{pmatrix} \tilde{y}, \tag{4.13}$$

where the way the submatrices in $\tilde{K}$ are labeled will be apparent later. This results in the closed loop system equations

$$\begin{aligned} \dot{\tilde{x}} &= \tilde{A}\tilde{x} + \tilde{B}_1 w + \tilde{B}_2 \tilde{u} \\ &= \begin{pmatrix} A + B_2 K_D C_2 & B_2 K_C \\ K_B C_2 & K_A \end{pmatrix} \tilde{x} + \begin{pmatrix} B_1 + B_2 K_D D_{21} \\ K_B D_{21} \end{pmatrix} w, \\ z &= \tilde{C}_1 \tilde{x} + \tilde{D}_{11} w + \tilde{D}_{12} \tilde{u} \\ &= \begin{pmatrix} C_1 + D_{12} K_D C_2 & D_{12} K_C \end{pmatrix} \tilde{x} + \begin{pmatrix} D_{11} + D_{12} K_D D_{21} \end{pmatrix} w. \end{aligned}$$

The above expressions are equal to the expressions in (2.3). This means that by augmenting the system and synthesizing a static controller $\tilde{u} = \tilde{K}\tilde{y}$, we can obtain a dynamic controller for the nonaugmented system using the exact same matrices in the controller, but arranged as suggested by the partitioning in (4.13).

The augmented system contains more states than the nonaugmented system. This is not a desired property if we use Theorem 4.1 to synthesize a controller since the dimensions of the variables X and Y increase, which in turn increases the computational burden. However, the dimensions of the matrix $\tilde{K}$ do not increase, which makes this procedure suitable for methods that do not involve matrix variables of the kind that are used in Theorem 4.1. Some references on such methods are e.g. Gumussoy and Overton [2008a] and Apkarian and Noll [2006a], Apkarian and Noll [2006b] and Mammadov and Orsi [2005].

This concludes the background part of this thesis. In the next part we will present the suggested methods for low order H_∞ controller synthesis.

Part II

Methods

5
A primal logarithmic barrier method

In this chapter we will describe a primal barrier method that is adapted to the problem of finding a feasible x for the problem in (4.7). For convex problems, better convergence is obtained from methods using the primal-dual framework according to Potra and Wright [2000]. However, we consider a primal log-barrier method mostly because of its straight-forward implementation, see e.g. Boyd and Vandenberghe [2004]. The method presented in this chapter consists of both solving linear and nonlinear SDPs in order to find a solution. The approach is somewhat ad hoc and some parts are based on empirical results more than a solid theoretical foundation. However, the algorithm has shown to perform quite well.

This chapter is structured as follows. We begin in Section 5.1 by presenting published work related to logarithmic barrier methods. Then we present the problem formulation in Section 5.2. We continue by adapting a barrier method to the problem in Section 5.3–Section 5.5. We summarize all parts of the method in Algorithm 8 in Section 5.6 and conclude the chapter by a summary in Section 5.7.

5.1 Introduction

The barrier method emerged as a popular interior point algorithm for convex optimization as early as in the 1960s, and its early history is described in detail in [Fiacco and McCormick, 1968, Chapter 1.2]. Other related techniques are the method of centers, see Liêũ and Huard [1966], and penalty (or exterior point) methods, see [Fiacco and McCormick, 1968, Chapter 4]. The interest for barrier methods decreased in the 1970s due to concerns about ill-conditioning for high values of the barrier parameter.

When the paper on a polynomial-time projective algorithm in Karmarkar [1984] was published, researchers really began to focus on interior point methods. The barrier method regained interest when it was pointed out in Gill et al. [1986] that it had close connection with Karmarkar's algorithm. Nesterov and Nemirovski [1994] developed the theory of self-concordance, which was shown to be the key property of a barrier function for the method to achieve polynomial-time complexity. They extended the framework for primal logarithmic barrier algorithms to also include semidefinite programming (SDP) problems and second-order cone programming (SOCP) problems. A survey paper on interior-point methods for convex, conic and general nonlinear optimization can be found in Nemirovski and Todd [2008] and a survey paper on numerical methods for large-scale nonlinear optimization in Gould et al. [2005].

As mentioned in the introduction for the thesis, barrier methods have been used in several algorithms for design of low order H_∞ controllers, e.g. Kocvara et al. [2005], Leibfritz and Mostafa [2002], Hol et al. [2003], Thevenet et al. [2005] and Stingl [2006]. However they considered the BMI formulation of the problem in (2.13) as a starting point for their method instead of the LMI formulation in (2.21). The algorithm in Kocvara et al. [2005] was implemented in the software PENBMI which is a special version of PENNON. PENNON can be used for general nonlinear and semidefinite problems, see Kocvara and Stingl [2003].

5.2 Problem formulation

As described in Chapter 4, the problem of finding an H_∞ controller of order n_k such that the closed loop system satisfy $\|H_c(s)\|_\infty \leq \gamma$ can be formulated as solving the following nonconvex semidefinite program, which is a restatement of (4.7).

$$\underset{x}{\text{minimize}} \quad \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} \tag{5.1a}$$

$$\text{subject to} \quad F_j(x) = C^{(j)} - \sum_{i=1}^{m} A_i^{(j)} x_i \succ 0, \quad j = 1, 2, 3 \tag{5.1b}$$

If the solution x^* to (5.1) is such that $c_{n_x-n_k-1}(x)/c_{n_x-n_k}(x) = 0$ and $c_{n_x-n_k}(x) \neq 0$, then x^* satisfies (4.4) and we can obtain the controller by continuing from step 2 in Algorithm 2 from Chapter 2.

5.3 Using the logarithmic barrier formulation

Using Definition 3.8, we can state that the central path for the problem in (5.1) is the solutions $x^*(t)$, $t \geq 0$ to the following minimization problems,

$$\underset{x}{\text{minimize}} \quad t\frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} - \sum_{j=1,2,3} \log\det F_j(x), \tag{5.2}$$

where we have made the variable substitution $t = 1/\mu$ in (3.8) and multiplied by t. The idea for obtaining an approximate solution to (5.1) is to solve (5.2) iteratively for a sequence of increasing values of t_k. The point $x^*(t_{k-1})$ can be used in the next iteration as a starting point for the problem of finding $x^*(t_k)$.

It is important to note that since the objective function in (5.1) is a nonconvex function, a solution to (5.2) might not be a global solution and thus as $t \to \infty$, the sequence $x^*(t)$ might not approach a global solution of the problem (5.1).

5.4 An extra barrier function

In some cases, the problem in (5.2) is unbounded from below, even if the optimal point is attainable for the problem in (5.1). This can happen if the set defined by the inequalities in (5.1b) is not bounded. We can introduce an extra constraint such that the set becomes bounded and such that the optimum of the original problem in (5.1) is not affected. Consider the following lemma.

Lemma 5.1. *For $X, Y \in \mathbb{S}^{n_x}$, the following constraints define a bounded set.*

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \succeq 0, \tag{5.3}$$

$$\operatorname{trace}(X + Y) < M. \tag{5.4}$$

where $0 < M < \infty$ is an arbitrary constant.

Proof: From the inequality in (5.3) and the nonstrict Schur complement formula in Boyd et al. [1994] follows that $X \succ 0$. This and the inequality in (5.4) imply that

$$0 < X_{ii} < M, \quad i = 1, \ldots, n_x,$$

which means that each diagonal element is bounded. Since $X \succ 0$, we have for each submatrix $\hat{X} \in \mathbb{S}^2$ of X defined by

$$\hat{X} = \begin{pmatrix} X_{ii} & X_{ji} \\ X_{ji} & X_{jj} \end{pmatrix}, \quad 1 \le i < j \le n_x,$$

that $\hat{X} \succ 0$. By using Lemma 2.2, this implies that

$$X_{ii} - X_{ij}^2/X_{jj} > 0,$$

which is equivalent to the requirement that

$$-\sqrt{X_{ii}X_{jj}} < X_{ij} < \sqrt{X_{ii}X_{jj}},$$

and we have that

$$-M < X_{ij} < M.$$

This shows that each element in X is bounded. A similar argument proves that each element in Y is bounded. □

To make sure that the set defined by the inequalities in (4.4a)–(4.4c) is bounded, we can use Lemma 5.1 and add the inequality in (5.4) with M chosen as a large number. This corresponds to a fourth LMI for the problem in (5.1). The constant M should be chosen large enough such that the solution to the modified problem is the same as the solution to the original problem.

To derive the fourth LMI for the problem in (5.1), write the inequality in (5.4) as

$$M - \operatorname{trace}(X + Y) > 0. \tag{5.5}$$

Then by using the half-vectorization operator vech, see Definition 4.1, we make the variable substitution

$$x = \begin{pmatrix} \operatorname{vech}(X) \\ \operatorname{vech}(Y) \end{pmatrix}.$$

By defining

$$I_{XY} = \begin{pmatrix} \operatorname{vech}(I_{n_x}) \\ \operatorname{vech}(I_{n_x}) \end{pmatrix}, \tag{5.6}$$

we can write the inequality in (5.5) as

$$F_4(x) = M - I_{XY}^T x > 0.$$

Now we can construct an extra logarithmic barrier function as

$$\psi_4(x) = -\log F_4(x), \tag{5.7}$$

which is added to (5.2) and results in the following modified problem.

$$\underset{x}{\text{minimize}} \quad B(x,t) = t\frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} - \sum_{j=1,2,3} \log\det F_j(x) - \log F_4(x). \tag{5.8}$$

One must make sure that the extra barrier function does not create any false optimum. This may happen if the true optimum lies outside the extra barrier. In case that happens, we can enlarge the extra barrier, i.e., increase M, and solve again. However, $M = 10^4$ seems to work well in numerical evaluations.

5.5 Outlining a barrier based method

We will now focus on the problem in (5.8) and outline a barrier based method for solving it iteratively as $t \to \infty$. In the limit, the solution x^* to the problem in (5.8) will approach a local optimum of the problem in (5.1). The method interchangeably uses Algorithm 4 in Section 3.5.4 and a standard linear SDP solver for finding the initial point. The decision of when to switch between the methods is motivated by a mix of theory, intuition and experimental results. The impact of balancing the system, as described in Section 2.4, has appeared to be of great importance for the algorithm to work efficiently, and therefore has a central role in the algorithm.

5.5.1 Initial point calculation

Since Newton's method (Algorithm 4) is applied to the problem in (5.8), it needs to be initialized with a feasible starting point. An initial point x_0 can be calculated by solving the following problem, which is also a heuristic method for obtaining low rank solutions, as mentioned in e.g. Iwasaki [1999], Orsi et al. [2006], and Fazel et al. [2001].

$$x_0 = \underset{x}{\operatorname{argmin}} \quad I_{XY}^T x \quad \big(= \operatorname{trace}(X+Y) \big) \tag{5.9a}$$

$$\text{subject to} \quad F_j(x) = C^{(j)} - \sum_{i=1}^{m} A_i^{(j)} x_i \succ 0, \quad j = 1,2,3. \tag{5.9b}$$

This is the same optimization problem as in (5.1) but with another objective function. The problem in (5.9) is a standard linear SDP, and it can be solved using the software packages SDPT3, Toh et al. [2006], and YALMIP, Löfberg [2004].

After solving the problem in (5.9), the system is balanced around X_0, Y_0, as described in Section 2.4, in order to make the problem better conditioned. Then, the problem is reformulated using the new system matrices and the initial point calculation is done again.

5.5.2 Solving the main, nonconvex problem

Now we have a feasible starting point, and we can apply Newton's method (Algorithm 4), to the, now numerically balanced, problem in (5.8). However, we still need to discuss the choice of t, and especially how to initialize t.

In order to simplify notation in later sections, we introduce the following definitions. Let

$$f(x) = \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} \quad \text{and} \quad \phi(x) = -\sum_{j=1,2,3} \log\det F_j(x) - \log F_4(x). \tag{5.10}$$

Choosing the barrier parameter

When starting the algorithm, we need to choose an initial value of the barrier parameter, t_0. In [Boyd and Vandenberghe, 2004, page 570], a method is suggested which is based on a minimization of a measure of fulfilling the optimality conditions for being on the central path with respect to t. The result is

$$t_0 = \underset{t}{\operatorname{argmin}} \quad \big(t\nabla_x f(x_0) + \nabla_x \phi(x_0)\big)^T H^{-1} \big(t\nabla_x f(x_0) + \nabla_x \phi(x_0)\big) \tag{5.11}$$

where x_0 is the initial point and $H \succ 0$ is some matrix that defines a norm, for instance the Hessian

$$\nabla_{xx}^2 B(x,t) = t\nabla_{xx}^2 f(x_0) + \nabla_{xx}^2 \phi(x_0)$$

could be used, where $B(x,t)$ is defined in (5.8). The problem is, $\nabla_{xx}^2 f(x_0)$ need not be positive definite, which may cause problems. One may assume that in the beginning, t is usually small, and could be approximated as zero. This will also guarantee that the minimization problem is convex, since we then have that

$$\nabla^2_{xx} B(x,0) = \nabla^2_{xx}\phi(x_0) \succ 0.$$

Here, the extra barrier in (5.7) need not be included, because its contribution can be approximated as zero, if M is chosen appropriately big. If we simplify notation by writing $\nabla f_0 = \nabla_x f(x_0)$ and $\nabla\phi_0 = \nabla_x \phi(x_0)$, the problem in (5.11) can then be written as

$$t_0 = \underset{t}{\operatorname{argmin}} \quad t^2(\nabla f_0)^T H^{-1}\nabla f_0 + 2t\nabla f_0^T H^{-1}\nabla\phi_0 + (\nabla\phi_0)^T H^{-1}\nabla\phi_0, \tag{5.12}$$

which is a scalar quadratic problem. The solution can be written explicitly as

$$t_0 = -\frac{\nabla f_0 H^{-1}\nabla\phi_0}{\nabla f_0^T H^{-1}\nabla f_0}. \tag{5.13}$$

For $k \geq 1$, the barrier parameter is updated in a standard way as

$$t_k = \mu t_{k-1}, \tag{5.14}$$

where $\mu = 10$ has shown to work quite well.

Interference of bad scaling

We must make sure that the extra barrier, which was described in Section 5.4, will not interfere with the convergence of the algorithm to an optimal point of the original problem in (5.1) that we want to solve. Consequently, we test if

$$I_{XY}^T x > k_M M,$$

where $0 < k_M < 1$. When this condition is satisfied, it is assumed that this is due to bad scaling, since M is chosen to be big enough. Then we balance the problem around X_k, Y_k and restart the algorithm once again by choosing an initial point as described in Section 5.5.1, and proceed from there.

Dealing with the nonconvexity

When using a Newton method, to get a descent direction, the Hessian must be positive definite, as described in Section 3.5.1. If that is not the case, one may use any of the methods presented in Section 3.5.3 to create an approximate Hessian that is positive definite. An alternative to these methods is to use some special properties of the problem. Since the constraints define a convex set, the barrier function is convex, and hence the Hessian of the barrier function will be positive definite. The total Hessian will be

$$\nabla^2_{xx} B(x,t) = t\nabla^2_{xx} f(x) + \nabla^2_{xx}\phi(x), \tag{5.15}$$

where the first term on the right hand side might be indefinite but the second one is always positive definite. Therefore, we can modify the Hessian as

$$H(x,t,a) = t_H^a t\nabla^2_{xx} f(x) + \nabla^2_{xx}\phi(x), \tag{5.16}$$

where $0 < t_H < 1$ and a is a non-negative integer chosen such that $H(x, t, a) \succ 0$. We know that at least in the limit, $a \to \infty$, this will be true.

5.5.3 Stopping criteria

The presented algorithm consists of inner and outer iterations. The inner iterations are described by Algorithm 4, where the stopping criterion is some function of the Newton decrement, $\Lambda(x)$. For each outer iteration, the solution of (5.8) gives a point on the central path. The stopping criteria are as follows.

1. The algorithm is exceeding the maximum number of inner or outer iterations. This is the case if the algorithm has slow or no progress, thus the algorithm terminates. A solution to this problem could be to reformulate the problem, e.g. increase the performance measure γ, and restart the algorithm. However, this needs to be done manually.

2. The extra barrier is interfering as discussed in the previous section, i.e.,
$$I_{XY}^T x > k_M M, \quad 0 < k_M < 1.$$
Re-balance the problem around the current iterate (X_k, Y_k) and restart the algorithm by following the procedure described in Section 5.5.1, and proceed from there.

3. The objective function value is close to zero,
$$\frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} < \epsilon,$$
i.e., we have found an optimum to the problem (5.1) with accuracy ϵ.

5.6 The logarithmic barrier based algorithm

Now all parts of the algorithm have been described, and we can put them together into a complete algorithm, which is presented as Algorithm 8.

5.6.1 Optimality of the solution

Note that the optimality conditions are not necessarily satisfied when this algorithm finishes. This is due to the inner loop stopping criterion, which is a directional derivative. The only KKT condition for the unconstrained problem is that the gradient should be zero, which need not be true just because a directional derivative is.

However, the point x^* need not be optimal, just sufficiently good. When the algorithm finishes with success, the achieved solution is always feasible for (5.1) and the objective function is sufficiently small, i.e., near zero, which means that we have found a solution (with accuracy ϵ), to the original feasibility problem (4.4).

5.6.2 Finding the controller

When we have found a local optimal point x^* or equivalently, (X^*, Y^*), we now need to find a controller, which is a convex problem. To do so we proceed from Step 2 in Algorithm 2, i.e., recover the variable P as described in Section 2.5.

5.6.3 Algorithm performance

The performance of the algorithm will be investigated in Chapter 8, where we will apply Algorithm 8 to a set of test problems. Comparisons will be made with a primal-dual method, which will be described in Chapter 6, as well as with a method from literature, HIFOO, which is briefly presented in Section 8.1.2.

5.7 Chapter summary

In this chapter a primal logarithmic barrier method for design of low order H_∞ controllers is suggested. The algorithm is initialized by solving a semidefinite programming problem corresponding to a heuristic for finding low rank solutions. In each outer iteration the main algorithm attempts so find a local minimum to an unconstrained problem using Newton's method with a line search. As the barrier parameter tends to infinity, the aim is to find a feasible solution that corresponds to a low order controller with performance measure γ. A numerical evaluation of the algorithm is presented in Chapter 8.

Algorithm 8 The barrier algorithm

```
Given performance criteria $\gamma$ and system matrices $A, B, C, D$, formulate the problem.
Choose parameter $M$.
Calculate an initial point $x_0$, as in Section 5.5.1.
Calculate an initial value $t_0$, of the barrier parameter as in Section 5.5.2.
$k_{\text{inner}} := 0$, $k_{\text{outer}} := 0$
**while** $c_{n_x-n_k-1}(x_k)/c_{n_x-n_k}(x_k) < \epsilon$ **do**
  $k_{\text{outer}} := k_{\text{outer}} + 1$
  **while** 1 **do**
    Calculate gradient, $g_k = \nabla_x B(x_k, t_k)$ and Hessian $H_k = H(x_k, t_k, a_k)$.
    Find integer $a_k \geq 0$ such that $H_k \succ 0$, according to (5.16).
    Solve for the Newton step: $H_k p_k = -g_k$.
    Perform backtracking line search to find $\alpha$.
    $x_{k+1} := x_k + \alpha p_k$
    Calculate Newton decrement: $\Lambda = \sqrt{-g_k p_k}$
    $k_{\text{inner}} := k_{\text{inner}} + 1$
    **if** $\Lambda < \Lambda_{\text{tol}}$ **then**
      Satisfying tolerance for the Newton decrement, exit inner loop.
    **end if**
    **if** $k_{\text{inner}} \geq k_{\text{inner, max}}$ **then**
      **Abort**, too many inner iterations.
    **end if**
  **end while**   (Inner loop)
  **if** $k_{\text{outer}} \geq k_{\text{outer, max}}$ **then**
    **Abort**, too many outer iterations.
  **end if**
  Check for bad scaling, see Section 5.5.2. Re-balance and restart if necessary.
  $t := \mu t$
**end while**   (Outer loop)
**Success!**
```

6 A primal-dual method

In this chapter we will describe a primal-dual interior point method for low order H_∞ controller synthesis. The method appears in two versions that are named the *primal-dual method* and the *quasi-Newton method*. The second method is based on the first one and the main difference is that the Hessian calculations are carried out using damped BFGS updating.

The presentation in this chapter is based on relevant chapters in Nocedal and Wright [2006] and Todd et al. [1998].

This chapter is structured as follows. We begin in Section 6.1 by presenting published work for solving semidefinite programming problems with focus on primal-dual methods. Then we present the problem formulation in Section 6.2 and the optimality conditions in Section 6.3. In Section 6.4 and Section 6.5 we describe the different parts of a Mehrotra primal-dual method. We summarize our method in Algorithm 9 in Section 6.6 and conclude the chapter by a summary in Section 6.7.

6.1 Introduction

Since primal-dual methods are also interior point methods, they share their early history with primal logarithmic barrier methods, which is briefly presented in Section 5.1. In Meggido [1988] a framework for primal-dual algorithms was described, which was originally published in 1987. Mehrotra [1992] presented a practical algorithm for linear programming that remains the basis for most current software. This work was extended to semidefinite programming (SDP) and second-order cone programming (SOCP) in the work by Nesterov and Todd [1997, 1998].

Forsgren and Gill [1998] and Vanderbei and Shanno [1999] present methods that extend primal-dual interior theory for linear and quadratic programming to nonconvex nonlinear programming. For a survey on interior point methods for nonlinear optimization, see Forsgren et al. [2002].

A method for solving nonconvex semidefinite programming was proposed in Jarre [2000]. It combines features of trust-region methods, sequential quadratic programming (SQP) methods and predictor-corrector interior point methods. In Freund et al. [2007] a sequential semidefinite programming (SSP) method was presented, which generalizes the SQP method for standard nonlinear programs. This method have some parts in common with primal-dual methods for semidefinite programming.

One approach for solving nonconvex semidefinite programming problems is Kanzow et al. [2005], which is a successive linearization method for nonlinear semidefinite programs with a trust-region framework. Another approach is presented in Correa and Hector Ramirez [2005], which is a method inspired by the SQP method. It solves a quadratic SDP in each iteration to get the search direction and then uses a nondifferentiable merit function combined with a line search strategy.

Algorithms for solving optimization problems defined on a subset of the cone of symmetric positive matrices are presented in Burer and Monteiro [2003] and Journée et al. [2010]. Those algorithms rely on the factorization $X = YY^T$, where $Y \in \mathbb{R}^{n \times r}$ and $r < n$, hence resulting in low rank solutions of the problem.

In Yamashita et al. [2011], a primal-dual interior point method is presented that minimizes a nonconvex function subject to linear semidefinite constraints. They suggest two versions of their method, where one uses an exact Hessian while the other uses a BFGS update of the Hessian. The approach in this chapter is similar, however we were unaware of this work until very recently.

For a reference on primal-dual interior point methods for linear programming, see e.g. the book by Wright [1997]. For a survey of sequential quadratic programming, see Boggs and Tolle [1995], for a survey on semidefinite optimization, see Todd [2001] and for a more general survey on interior point methods for optimization, see Nemirovski and Todd [2008].

6.2 Problem formulation

As described in Chapter 4, the problem of finding an H_∞ controller of order n_k such that the closed loop system satisfy $\|H_c(s)\|_\infty \leq \gamma$ can be formulated as solving the following nonconvex semidefinite program, which is a restatement of

(4.7) or (5.1).

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)} && \text{(6.1a)} \\ \text{subject to} \quad & C^{(j)} - \sum_{i=1}^{m} A_i^{(j)} x_i \succ 0, \quad j = 1, 2, 3. && \text{(6.1b)} \end{aligned}$$

If the solution x^* to (6.1) is such that $c_{n_x-n_k-1}(x)/c_{n_x-n_k}(x) = 0$ and $c_{n_x-n_k}(x) \neq 0$, this means that x^* satisfies (4.4) and we can obtain the controller by continuing from step 2 in Algorithm 2 from Chapter 2. To keep the notation simple, define

$$f(x) = \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)},$$

and merge the three semidefinite constraints in (6.1b) into one constraint by placing the matrices $C^{(j)}$ and $A_i^{(j)}$ in block diagonal matrices consisting of three blocks, one for each value of j. This results in

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & C - \sum_{i=1}^{m} A_i x_i \succeq 0. \end{aligned} \tag{6.2}$$

Note that the constraints are merged to simplify notation. It is advisable to utilize the block structure in order not to increase the number of variables at a later stage when dual and slack variables are introduced.

6.3 Optimality conditions

The first-order necessary conditions for optimality, see Theorem 3.1, have a very central role in primal-dual methods. The Karush-Kuhn-Tucker conditions for (6.2) are given by

$$\begin{aligned} \frac{df(x)}{dx_i} + \langle A_i, Z \rangle = 0, \quad i = 1, \ldots, m && \text{(6.3a)} \\ C - \sum_{i=1}^{m} A_i x_i - S = 0, && \text{(6.3b)} \\ ZS = 0, && \text{(6.3c)} \\ Z \succeq 0, \quad S \succeq 0, && \text{(6.3d)} \end{aligned}$$

where $Z \in \mathbb{S}^p$ is a dual variable and $S \in \mathbb{S}^p$ is a slack variable of appropriate dimension p. Note that (6.3a) consists of m scalar equations, while (6.3b)–(6.3c) are matrix equations. Since $f(x)$ is a nonconvex function, these are only necessary and not sufficient conditions for an optimal solution of (6.2).

6.4 Solving the KKT conditions

As mentioned previously, the main idea of primal-dual methods is to find a solution to the KKT conditions. The way it is usually done is to search for solutions along the central path.

6.4.1 Central path

The central path (as defined in Definition 3.12) for the problem in (6.2) is the solution points for

$$\frac{df(x)}{dx_i} + \langle A_i, Z \rangle = 0, \quad i = 1, \dots, m \tag{6.4a}$$

$$C - \sum_{i=1}^{m} A_i x_i - S = 0, \tag{6.4b}$$

$$ZS = \nu I, \tag{6.4c}$$

$$Z \succeq 0, \quad S \succeq 0, \tag{6.4d}$$

where $\nu \geq 0$. The difference from (6.3) is the third equation. When solving the equations for the KKT conditions in (6.3), we generate a sequence of iterates

$$\bar{x}^{(k)} = (x^{(k)}, Z^{(k)}, S^{(k)}),$$

that solve (6.4a)–(6.4c) for values of ν that tend to zero. The iterates need not necessarily be feasible, except for (6.4d), which always must hold.

6.4.2 Symmetry transformation

Since the domain and range of the function defined by the left hand side of the equation in (6.4c) is not the same space, we cannot directly apply Newton's method, as described in Section 3.6.1, to the system of equations defined by (6.4a)–(6.4c). A solution for this is to use the symmetry transformation in Zhang [1998] which is defined as

$$\mathcal{H}_P(M) = \frac{1}{2}(PMP^{-1} + (PMP^{-1})^T) \tag{6.5}$$

for a given invertible matrix P. It is shown that if M is similar to a symmetric positive definite matrix, then

$$\mathcal{H}_P(M) = \nu I \Leftrightarrow M = \nu I.$$

This means that (6.4c) can be replaced by

$$\mathcal{H}_P(ZS) = \nu I \tag{6.6}$$

without affecting the definition of the central path, since ZS is similar to the positive definite matrix $S^{1/2}ZS^{1/2}$. The equations in (6.4a), (6.4b) and (6.6) can be written as $F(\bar{x}) = 0$ for some F and $\bar{x}$. Now we can apply Newton's method for nonlinear equations to $F(\bar{x})$. We denote the step that minimizes the linear model as $(\Delta x, \Delta Z, \Delta S)$ and obtain the follow equations that define the step.

$$\frac{df(x)}{dx_i} + \langle A_i, Z\rangle + \frac{d}{dx_i}\nabla_x^T f(x)\Delta x + \langle A_i, \Delta Z\rangle = 0, \quad i = 1,\ldots,m \tag{6.7a}$$

$$C - \sum_{i=1}^{m} A_i x_i - S - \sum_{i=1}^{m} A_i \Delta x_i - \Delta S = 0, \tag{6.7b}$$

$$\mathcal{H}_P(ZS) + \mathcal{H}_P(\Delta Z S + Z\Delta S) - \nu I = 0. \tag{6.7c}$$

The scaling matrix P which is used in the symmetrization operator, can be chosen in several ways which has given rise to different methods, e.g. the Nesterov-Todd (NT) method, given in Nesterov and Todd [1997, 1998], the AHO method in Alizadeh et al. [1998] and the H..K..M method, see Helmberg et al. [1996], Kojima et al. [1997], Monteiro [1997]. For a presentation of how to calculate these scaling matrices and some numerical experiments using them, see Todd et al. [1998].

In this work, we chose the Nesterov-Todd (NT) scaling matrix since it has shown to perform well in applications, see e.g. Todd et al. [1998]. To compute the NT scaling matrix, begin by Cholesky factorizing Z and S as

$$Z = LL^T, \quad S = RR^T, \tag{6.8}$$

and let $UDV^T = R^T L$ be the SVD of $R^T L$. Then the NT scaling matrix is given by

$$P = LVD^{-1/2}. \tag{6.9}$$

As can be seen here, the scaling matrix depends on Z and S and must therefore be recalculated in each iteration.

6.4.3 Definitions

Before we move on, we define the symmetric vectorization operator svec and the symmetric Kronecker product.

Definition 6.1 (svec). The operator svec : $\mathbb{S}^n \to \mathbb{R}^{n(n+1)/2}$ is defined by

$$\text{svec}(U) = (u_{11}, \sqrt{2}u_{21}, \ldots, \sqrt{2}u_{n1}, u_{22}, \sqrt{2}u_{32}, \ldots, \sqrt{2}u_{n2}, \ldots, u_{nn})^T, \tag{6.10}$$

i.e., svec is an operator that maps symmetric matrices into vectors. The factor $\sqrt{2}$ is introduced so that svec is an isometry between $\mathbb{S}^{n\times n}$ and $\mathbb{R}^{n(n+1)/2}$ with their respective standard inner products. Let the inverse of svec be denoted smat.

Note that the operator svec is different from the operator vech in Definition 4.1, since svec scales the nondiagonal entries by $\sqrt{2}$.

Example 6.1

Let us illustrate the operator svec by a simple example. If we let

$$U = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix},$$

then

$$\operatorname{svec}(U) = \left(1,\ 2\sqrt{2},\ 3\sqrt{2},\ 4,\ 5\sqrt{2},\ 6\right)^T,$$

and the inner products

$$\langle U, U\rangle = \operatorname{trace}(U^T U) = 129,$$
$$\langle \operatorname{svec}(U), \operatorname{svec}(U)\rangle = \operatorname{svec}(U)^T \operatorname{svec}(U) = 129,$$

are the same.

Definition 6.2 (Symmetric Kronecker product). Let A and B be two real matrices of dimension $n \times n$. The *symmetric Kronecker product* of A and B is then defined as

$$A \otimes_S B = \frac{1}{2} U(A \otimes B + B \otimes A)U^T, \tag{6.11}$$

where $U \in \mathbb{R}^{n(n+1)\times n^2}$ is a matrix created in the following way. If we label the rows of U in the order $(1,1),(2,1),\ldots,(n,1),(2,2),(3,2),\ldots,(n,2),(3,3)\ldots,(n,n)$ and its columns in the order $(1,1),(2,1),\ldots,(n,1),(1,2),\ldots,(n,2),(1,3),\ldots,(n,n)$, then

$$U_{(i,j),(k,l)} = \begin{cases} 1 & \text{if } i = j = k = l, \\ 1/\sqrt{2} & \text{if } i = k \neq j = l, \text{or } i = l \neq j = k, \\ 0 & \text{otherwise.} \end{cases}$$

See e.g. the appendix of Todd et al. [1998] for properties and details regarding the symmetric Kronecker product.

Example 6.2

When $n = 3$, the matrix U in (6.11) is given by

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

6.4.4 Computing the search direction

To compute the Newton step $(\Delta x, \Delta Z, \Delta S)$, it is convenient to express the linear systems of equations in the standard matrix vector form. Using the previously defined symmetric vectorization operator and symmetric Kronecker product, we can now write the Newton step equations (6.7) as a 3×3 block equation

$$\begin{pmatrix} H & \mathcal{A}^T & 0 \\ \mathcal{A} & 0 & \mathcal{I} \\ 0 & E & F \end{pmatrix} \begin{pmatrix} \Delta x \\ \text{svec}(\Delta Z) \\ \text{svec}(\Delta S) \end{pmatrix} = \begin{pmatrix} r_p \\ \text{svec}(R_d) \\ \text{svec}(R_c) \end{pmatrix}, \tag{6.12}$$

where H is a positive definite approximation of the Hessian of $f(x)$,

$$\mathcal{A} = [\text{svec}(A_1), \ldots, \text{svec}(A_m)], \tag{6.13}$$

$$E = P \otimes_s P^{-T} S, \quad F = PZ \otimes_s P^{-T}, \tag{6.14}$$

and $\mathcal{I}$ is the identity matrix of appropriate dimension. Also we have the residuals

$$r_p = \nabla_x f(x) + \mathcal{A}^T Z, \quad R_d = C - S - \sum_{i=1}^{m} A_i x_i, \quad R_c = \nu I - \mathcal{H}_P(ZS). \tag{6.15}$$

By solving (6.12), we get the search direction $(\Delta x, \Delta Z, \Delta S)$.

6.4.5 Choosing a positive definite approximation of the Hessian

Since the objective function $f(x)$ is a nonconvex function, its Hessian needs not be positive definite. Two ways of convexifying the Hessians have been implemented for the primal-dual method. These are briefly described next.

Adding a multiple of the identity

An approximate Hessian can be chosen as in (3.18), i.e.

$$H = \nabla^2_{xx} f(x) + dI, \tag{6.16}$$

where $d \geq 0$ such that H becomes positive definite. If $\nabla^2_{xx} f(x) \nsucc 0$, we can choose

$$d = -(1 + \delta_\lambda)\lambda_{\min}\big(\nabla^2_{xx} f(x)\big), \tag{6.17}$$

where $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue and δ_λ is a small positive constant. The version of the algorithm using this way of approximating the Hessian is referred to as the *primal-dual method* or *PD* later in the thesis.

Damped BFGS updating

Instead of calculating the exact Hessian and convexifying it as done in the previous section, an approximation of the Hessian can be calculated using damped BFGS updating. This procedure is described in Section 3.5.5. The approximation of the Hessian that results from damped BFGS updating is positive definite, thus suitable here. The version of the algorithm using this way of approximating the Hessian is referred to as the *quasi-Newton method* or *QN* later in the thesis.

6.4.6 Uniqueness of symmetric search directions

To prove that the symmetric search directions that we obtain from solving (6.12) are unique, we show that the only solution to the 3×3 block equation

$$\begin{pmatrix} H & \mathcal{A}^T & 0 \\ \mathcal{A} & 0 & \mathcal{I} \\ 0 & E & F \end{pmatrix} \begin{pmatrix} \Delta x \\ \mathrm{svec}(\Delta Z) \\ \mathrm{svec}(\Delta S) \end{pmatrix} = 0 \tag{6.18}$$

is the trivial zero solution. Since P is an invertible matrix and $Z, S \succ 0$, by Theorem 3.2 in Todd et al. [1998], we have that $E^{-1}F \succ 0$. Further, we can assume that E and F are invertible, by using properties of the symmetric Kronecker product (see Appendix of Todd et al. [1998] for details). Using block Gaussian elimination we can reduce (6.18) to a Schur complement equation

$$(H + \mathcal{A}^T E^{-1} F \mathcal{A})\Delta x = 0. \tag{6.19}$$

Since $H \succ 0$ and $\mathcal{A}^T E^{-1} F \mathcal{A} \succeq 0$ regardless of the rank of $\mathcal{A}$, $(H + \mathcal{A}^T E^{-1} F \mathcal{A}) \succ 0$ is invertible and the only solution to (6.19) is the trivial solution. Now the second block equation of (6.18) gives that $\Delta S = -\mathrm{smat}(\mathcal{A}\Delta x) = 0$, and the third one gives that $\Delta Z = -E^{-1}F\Delta S = 0$. This shows that the solution to (6.12) exists and is unique.

6.4.7 Computing step lengths

Once a search direction $(\Delta x, \Delta Z, \Delta S)$ is found, the next step is to determine how far to go in that direction. Step lengths $0 < \alpha \leq 1$ and $0 < \beta \leq 1$ are chosen such that

$$Z^{(k+1)} = Z^{(k)} + \alpha \Delta Z \succeq 0, \qquad S^{(k+1)} = S^{(k)} + \beta \Delta S \succeq 0, \tag{6.20}$$

i.e., such that the positive definiteness is maintained for the symmetric variables Z and S. The variable x is updated as

$$x^{(k+1)} = x^{(k)} + \beta \Delta x.$$

We use the *fraction to boundary rule*, see e.g. Nocedal and Wright [2006], which means that we do not let the next iterate end up at the boundary of semidefiniteness, but instead only near it. One way to do this is to choose the step lengths as

$$\alpha^{(k)} = \min\Big(1, \frac{-\tau^{(k)}}{\lambda^{-}_{\min}(L^{-1}\Delta Z L^{-T})}\Big), \quad \beta^{(k)} = \min\Big(1, \frac{-\tau^{(k)}}{\lambda^{-}_{\min}(R^{-1}\Delta S R^{-T})}\Big), \tag{6.21}$$

with L and R chosen as in (6.8) and where $\lambda^{-}_{\min}(A)$ denotes the minimum eigenvalue of A. If all eigenvalues are positive the term is ignored, i.e., the step length is chosen to be 1, regardless of the value of the $\min(\cdot,\cdot)$ expressions. The parameter τ is chosen based on the step lengths taken in the previous iteration, i.e., by setting

$$\tau^{(k+1)} = 0.9 + 0.09 \min(\alpha^{(k)}, \beta^{(k)}), \tag{6.22}$$

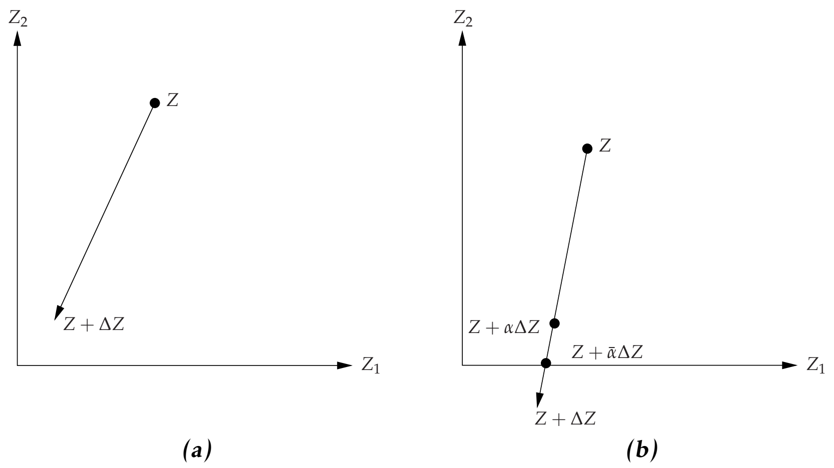

Figure 6.1: *The two figures above illustrate the case where the cone is $\mathbb{R}^2_+$. To the left, the full step does not take us outside of the cone. In the figure to the right we apply the fraction to boundary rule. If $\tau = 1$, the resulting step length parameter is $\bar{\alpha}$ and we end up at the boundary. The number α is calculated using $\tau < 1$ and we end up at some distance from the boundary instead.*

where $\alpha^{(k)}$ and $\beta^{(k)}$ are the step lengths in the kth iteration. An initial value $\tau^{(0)} = 0.98$ has shown to work well. The fraction to boundary rule is illustrated in Figure 6.1.

6.5 A Mehrotra predictor-corrector method

In this section we will describe the steps of an algorithm based on Mehrotra's algorithm, see Mehrotra [1992], however we follow the presentation in the papers by Todd et al. [1998] and Toh et al. [1999] which extend Mehrotra's algorithm to semidefinite programming.

6.5.1 Initial point calculation

Always when solving optimization problems, a good initial point is important. For convex problems it will have an impact on the number of iterations needed, and for nonconvex problems it also determines which KKT-point the algorithm will converge to.

The initial point calculation used for the primal-dual method is based on what is suggested in Toh et al. [1999] and will be described below.

Assume the matrices A_i and C are block-diagonal of the same structure, each consisting of L blocks of square matrices of dimensions $n_1, n_2, \dots, n_L$. Let $A_i^{(j)}$

and $C^{(j)}$ denote the jth block of A_i and C, respectively. Then the initial point can be chosen as

$$\begin{aligned}x^{(0)} &= \bar{\mathbf{1}},\\ Z^{(0)} &= \mathrm{blkdiag}(\xi_1 I_{n_1},\ \xi_2 I_{n_2},\ \dots,\ \xi_L I_{n_L}),\\ S^{(0)} &= \mathrm{blkdiag}(\eta_1 I_{n_1},\ \eta_2 I_{n_2},\ \dots,\ \eta_L I_{n_L}),\end{aligned} \tag{6.23}$$

where $\bar{\mathbf{1}}$ is a column vector containing ones, I_{n_j} is the identity matrix with dimension n_j and

$$\xi_j = n_j \max_{1\le i\le m} \frac{1+|b_i|}{1+\|A_i^{(j)}\|_F}, \quad \eta_j = \frac{1+\max[\max_i\{\|A_i^{(j)}\|_F\}, \|C^{(j)}\|_F]}{\sqrt{n_j}},$$

where b is referring to the linear objective function which in our case is

$$b = \begin{pmatrix}\mathrm{vech}(I_n)\\ \mathrm{vech}(I_n)\end{pmatrix} \tag{6.24}$$

so that

$$b^T x = \mathrm{trace}(X+Y), \tag{6.25}$$

i.e., we follow the heuristics for minimizing rank as mentioned in e.g. Iwasaki [1999], Orsi et al. [2006], and Fazel et al. [2001]. The only difference from Toh et al. [1999] is that the authors suggest $x^{(0)} = 0$, but our choice has shown to work better in our applications. By multiplying the identity matrix I_{n_j} by the factors ξ_j and η_j for each j, the initial point has a better chance of having the same order of magnitude as an optimal solution of the SDP according to Toh et al. [1999].

Note that the assumption that A_i and C are block-diagonal is valid since we merged three constraints into one, as explained in Section 6.2.

6.5.2 The predictor step

Set $\nu = 0$ in (6.15), solve the system of equations in (6.12) and denote the solution $(\Delta x^{\mathrm{aff}}, \Delta Z^{\mathrm{aff}}, \Delta S^{\mathrm{aff}})$. This step is sometimes called the *predictor step* or *affine scaling step*. Then calculate step lengths $\alpha^{\mathrm{aff}}, \beta^{\mathrm{aff}}$ as done in (6.21). Define μ and μ^{aff} as

$$\mu = \langle Z, S\rangle / p, \tag{6.26}$$

$$\mu^{\mathrm{aff}} = \langle Z + \alpha^{\mathrm{aff}}\Delta Z^{\mathrm{aff}}, S + \beta^{\mathrm{aff}}\Delta S^{\mathrm{aff}}\rangle / p, \tag{6.27}$$

and the centering parameter σ as

$$\sigma = \Big(\frac{\mu^{\mathrm{aff}}}{\mu}\Big)^e, \tag{6.28}$$

The exponent e is chosen according to

$$e = \begin{cases}\max\Big(1, 3\ \min(\alpha^{\mathrm{aff}}, \beta^{\mathrm{aff}})^2\Big) & \text{if } \mu > 10^{-6},\\ 1 & \text{if } \mu \le 10^{-6},\end{cases} \tag{6.29}$$

which is a heuristic suggested in Toh et al. [1999] that does not have a solid analytical justification, but appears to work well in practice.

Note that the algorithm does not update the iterate with the predictor step. It is only used to calculate the centering parameter, σ, which is needed for the computation of the corrector step, which will be described next.

6.5.3 The corrector step

The search direction is now calculated by solving (6.12) again but replacing R_c with

$$R_s = \sigma \mu I - \mathcal{H}_P(ZS) - \mathcal{H}_P(\Delta Z^{\text{aff}} \Delta S^{\text{aff}}), \tag{6.30}$$

where μ and σ are calculated as in (6.26) and (6.28) respectively. The previously calculated predictor step $(\Delta Z^{\text{aff}}, \Delta S^{\text{aff}})$ is used in a *second-order correction*, see Todd et al. [1998], where it is assumed that the predictor step is a decent approximation of the corrector step. This will result in

$$\begin{pmatrix} H & \mathcal{A}^T & 0 \\ \mathcal{A} & 0 & \mathcal{I} \\ 0 & E & F \end{pmatrix} \begin{pmatrix} \Delta x \\ \text{svec}(\Delta Z) \\ \text{svec}(\Delta S) \end{pmatrix} = \begin{pmatrix} r_p \\ \text{svec}(R_d) \\ \text{svec}(R_s) \end{pmatrix}, \tag{6.31}$$

with the submatrices and vectors defined in (6.13)–(6.15). The coefficient matrix in (6.31) is the same as in the calculation of the predictor step in (6.12), and hence the cost of solving the second system can be made relatively small since the same matrix factorization can be used in both cases.

6.6 The primal-dual algorithm

By summarizing the last few sections we can now state an algorithm as Algorithm 9. Define the residual

$$r(x, Z, S, \nu) = \begin{pmatrix} r_p(x, Z) \\ \text{svec}\left(R_d(S, x)\right) \\ \text{svec}\left(R_c(Z, S, \nu)\right) \end{pmatrix} \tag{6.32}$$

with $\nu = 0$, where r_p, R_d and R_c are calculated as in (6.15). This expression is used in a stopping criterion in the algorithm.

6.6.1 Finding the controller

If the algorithm finishes with $f(x) = 0$, we can recover the controller parameters by solving a convex problem. To do so we proceed from Step 2 in Algorithm 2 in Chapter 2, i.e., recover the variable P as described in Section 2.5.

6.6.2 Algorithm performance

The performance of the algorithm is evaluated in Chapter 8 where two evaluations are made. In the first evaluation, the *primal-dual method* is compared with the primal logarithmic barrier method from Chapter 5 and with a method

Algorithm 9 The primal-dual algorithm

```
Calculate the initial point $(x^{(0)}, Z^{(0)}, S^{(0)})$ as explained in Section 6.5.1.
Set $\tau^{(0)} = 0.98$, $k := 0$.
  while $\|r(x^{(k)}, Z^{(k)}, S^{(k)}, 0)\|_2 < r_{\text{tol}}$ do
    if $k > k_{\max}$ then
      Abort, too many iterations.
    end if

    Predictor step:
    Set $(x, Z, S) = (x^{(k)}, Z^{(k)}, S^{(k)})$.
    Solve (6.12) for $(\Delta x^{\text{aff}}, \Delta Z^{\text{aff}}, \Delta S^{\text{aff}})$ with $\nu = 0$.
    Calculate step lengths $\alpha^{\text{aff}}, \beta^{\text{aff}}$ using (6.21).
    Calculate $\mu$ and $\mu^{\text{aff}}$ using (6.26) and (6.27).
    Set centering parameter to $\sigma = (\mu^{\text{aff}}/\mu)^e$ where $e$ is calculated as in (6.29).

    Corrector step:
    Solve (6.31) for $(\Delta x, \Delta Z, \Delta S)$.
    Calculate step lengths $\alpha, \beta$ using (6.21).
    Set $(x^{(k+1)}, Z^{(k+1)}, S^{(k+1)}) = (x + \beta\Delta x, Z + \alpha\Delta Z, S + \beta\Delta S)$.
    Calculate $\tau^{(k+1)}$ using (6.22).
    Set $k := k + 1$.
  end while
```

from literature, HIFOO, which is briefly presented in Section 8.1.2. In the second evaluation, the *quasi-Newton method* is compared with HIFOO.

6.7 Chapter summary

I this chapter a primal-dual method for design of low order H_∞ controllers is suggested. The starting point of the algorithm is chosen based on heuristics for finding low rank solutions. When the starting point has been chosen, a Mehrotra predictor-corrector is applied to the problem. When the algorithm finishes with success, the KKT conditions are satisfied to a specified accuracy, and a controller of low order with performance measure γ is found. A numerical evaluation of the algorithm is presented in Chapter 8.

7 A partially augmented Lagrangian method

In this chapter we will present a partially augmented Lagrangian method for low order H_∞ synthesis. It is a similar method to the augmented Lagrangian method that was presented in Section 3.7.2, but *partially* refers to the fact that some, and in our case only one of the constraints is relaxed while the others are kept as constraints. References on the standard augmented Lagrangian method can be found in Section 3.7.2.

One main difference in the approach as compared to the algorithms that were presented in Chapter 5 and Chapter 6 is that this algorithm explicitly tries to minimize the performance measure γ instead of finding a controller that satisfies a pre-specified value. Parts of this chapter was also presented in Ankelhed et al. [2011b].

This chapter is structured as follows. In Section 7.1 we present published work related to partially augmented Lagrangian methods. Then the problem formulation is presented in Section 7.2, and how the partially augmented Lagrangian method is applied is described in Section 7.3. In Section 7.4 we present how the search direction is calculated, and in Section 7.5 the suggested algorithm is outlined. In Section 7.6 the chapter is summarized.

7.1 Introduction

In Conn et al. [1991] the problem of minimizing an augmented Lagrangian function subject to simple bound constraints was considered. In Conn et al. [1996] the work was extended to include linear inequality constraints, where also local and global convergence of the algorithm were proved.

In Fares et al. [2001] a partially augmented Lagrangian method for a class of LMI-constrained problems in robust control was considered. A modified Newton method and a trust-region method were considered for solving the semidefinite optimization problem in each iteration. The trust-region method was considered inferior to the modified Newton method for this problem formulation. In Fares et al. [2002] a sequential semidefinite programming algorithm was considered for a similar problem formulation.

In Apkarian et al. [2003] a partially augmented Lagrangian method for low order H_∞ controller synthesis was described, which relaxed the constraint $\|XY - I\|_F = 0$. This corresponds to the case where a static controller ($n_k = 0$) is searched for. In each iteration, a semidefinite problem was solved where a modified Newton method was used. A spectral quadratic-SDP method for solving this semidefinite problem was developed in Apkarian et al. [2004] using the same framework. However, it was indicated that a good implementation was critical for it to work properly. In Noll et al. [2004] the framework was generalized to optimization problems with general matrix inequality constraints, where also convergence proofs are provided.

The method described in this chapter is similar to the one in Apkarian et al. [2003], but here the equality constraint is the quotient of the two coefficients of the characteristic polynomial in (4.4d). Compared to the approach in Apkarian et al. [2003], our approach does not lead to a problem with more variables in the case when a controller with more states than zero is desired. In that case we simply choose another equality constraint, while the approach in Apkarian et al. [2003] is to augment the system with extra states as explained in Section 4.5.

7.2 Problem formulation

As mentioned in the beginning of this chapter, the problem we wish to solve here is slightly different compared to previous chapters, but it is related. Instead of choosing a γ beforehand, we now wish to minimize γ subject to the constraints. Formally this can be stated as the following optimization problem.

$$\begin{aligned} \underset{\gamma,X,Y}{\text{minimize}} \quad & \gamma \\ \text{subject to} \quad & \frac{c_{n_x-n_k-1}(I-XY)}{c_{n_x-n_k}(I-XY)} = 0, \\ & (\gamma, X, Y) \in \mathbb{X}, \end{aligned} \tag{7.1}$$

where $\mathbb{X}$ is a convex set defined by the three LMIs in (4.4a)–(4.4c). A change of variables is made in order to simplify notation as follows. Let x include the half-vectorizations of X and Y as well as γ stacked on top of each other as in

$$x = \begin{pmatrix} \text{vech}(X) \\ \text{vech}(Y) \\ \gamma \end{pmatrix} \in \mathbb{R}^{n_x(n_x+1)+1},$$

where vech is defined in Definition 4.1. This is similar to what is done in Section 4.3.2 but the difference is that we include γ in x, since γ is now an optimization variable. This results in the following equivalent problem formulation.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & b^T x \\ \text{subject to} \quad & \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k-1}(x)} = 0, \\ & x \in \mathbb{X}, \end{aligned} \tag{7.2}$$

where b is the last unit vector, i.e.

$$b^T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix},$$

such that $b^T x = \gamma$. We realize that stating the equivalence of $(X, Y, \gamma) \in \mathbb{X}$ to $x \in \mathbb{X}$ is abuse of notation.

7.3 The partially augmented Lagrangian

After defining $\hat{c}(x)$ as

$$\hat{c}(x) = \frac{c_{n_x-n_k-1}(x)}{c_{n_x-n_k}(x)},$$

the problem in (7.2) is approached by using the partially augmented Lagrangian method, where the equality constraint is relaxed and added to the objective function in the following way.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \Phi(x, \lambda, \mu) = b^T x + \lambda \hat{c}(x) + \frac{\mu}{2} \hat{c}^2(x) \\ \text{subject to} \quad & x \in \mathbb{X}, \end{aligned} \tag{7.3}$$

where λ is a Lagrangian multiplier and μ is a penalty multiplier. As mentioned earlier in the chapter, the word *partially* refers to the fact that only the equality constraint is used in the augmentation while the LMIs are kept as constraints. This is a nonconvex problem, since $\hat{c}(x)$ is a nonconvex function. However $\mathbb{X}$ is a convex set which makes the problem somewhat less difficult to solve than a general nonconvex problem.

Similarly to the augmented Lagrangian method in Section 3.7.2, the idea is to approximately solve (7.3) for a sequence of increasing values of μ_k using Newton's method. However, since the optimization problem in (7.3) still includes constraints, we must make sure that the next point also satisfies the constraints. How to calculate a search direction that accomplishes this is the topic of next section.

7.4 Calculating the search direction

To calculate the search direction, approximate $\Phi(x + p, \lambda, \mu)$ by a quadratic function related to the first three terms in the Taylor series expansion around the point x. Similarly to what is done in regular Newton methods, we intend to find a step direction p that minimizes this second order model, but the difference is that we also require that $x + p \in \mathbb{X}$, i.e. that the next point also lies in the feasible set. This problem can be formulated as

$$\begin{aligned} \underset{p}{\text{argmin}} \quad & \nabla_x \Phi(x, \lambda, \mu)^T p + \frac{1}{2} p^T H(x, \lambda, \mu) p \\ \text{subject to} \quad & x + p \in \mathbb{X}, \end{aligned} \tag{7.4}$$

which can be reformulated as a conic programming problem that can be solved using e.g. YALMIP, Löfberg [2004] with SDPT3, Toh et al. [1999]. The symmetric matrix $H(x, \lambda, \mu, \delta)$ is a positive definite approximation of the Hessian of $\Phi(x, \lambda, \mu)$.

7.4.1 Calculating the derivatives

Differentiating $\Phi(x, \lambda, \mu)$ with respect to x we get

$$\begin{aligned} \nabla_x \Phi(x, \lambda, \mu) &= b + \lambda \nabla_x \hat{c}(x) + \mu \hat{c}(x) \nabla_x \hat{c}(x), \\ \nabla_{xx} \Phi(x, \lambda, \mu) &= \big(\lambda + \mu \hat{c}(x)\big) \nabla^2_{xx} \hat{c}(x) + \mu \nabla_x \hat{c}(x) \nabla_x^T \hat{c}(x), \end{aligned}$$

where the gradient and Hessian of $\hat{c}(x)$, are derived in Section 4.4.3.

Since the constraint function $\hat{c}(x)$ is nonconvex, the Hessian $\nabla^2_{xx} \hat{c}(x)$ is not always positive definite which in turn might lead to that $\nabla^2_{xx} \Phi_c(x, \lambda, \mu)$ is not necessarily positive definite, which has to be dealt with. Two common ways are to either use Newton methods in which the Hessian is convexified or to use trust-region methods where the nonconvexity is dealt with by optimizing over a limited region in each iteration. The authors of Apkarian et al. [2003] advice against using trust-region methods since the complexity of such a method is too big in their case. Since we use a similar approach to theirs, we proceed by suggesting two different ways to calculate and convexify the Hessian $\nabla_{xx} \Phi(x, \lambda, \mu)$.

7.4.2 Hessian modifications

Two options for convexifying the Hessian are to

1. Use BFGS to approximate the Hessian, see Section 3.5.5. There is no need to convexify, since the BFGS approximation is positive definite.

2. Calculate the exact Hessian and then use the modified symmetric indefinite factorization so that it becomes positive definite as explained in Section 3.5.3. This was done in Apkarian et al. [2003], however they calculated the Gauss-Newton approximation instead of the exact Hessian.

We have tried both options listed above and found that the second alternative was the one that performed best. The parameter δ in (3.22) was chosen as

$$\delta = 10^{-4}\|\nabla^2_{xx}\Phi\|_\infty,$$

where the matrix norm $\|A\|_\infty$ denotes the largest row sum of A.

7.5 An outline of the algorithm

The algorithm can be outlined as follows.

1. **Initial phase.**

   (a) Find a good point to balance the system around, see Definition 2.5. With I_{XY} defined as in (5.6), solve the convex SDP

   $$\begin{aligned} \underset{x}{\text{minimize}} \quad & I_{XY}^T x \quad \Big(= \operatorname{trace}(X+Y)\Big) \\ \text{subject to} \quad & x \in \mathbb{X}, \end{aligned}$$

   i.e., use the heuristics for minimizing rank as mentioned in e.g. Orsi et al. [2006], Iwasaki [1999] and Fazel et al. [2001].

   (b) Balance the plant system around this solution to get good numeric properties, as explained in Section 2.4.

   (c) Find a starting point by solving the convex SDP

   $$\begin{aligned} \underset{x}{\text{minimize}} \quad & b^T x + I_{XY}^T x \\ \text{subject to} \quad & x \in \mathbb{X} \end{aligned} \tag{7.5}$$

   and denote the solution $(X^{(0)}, Y^{(0)})$. The objective function in (7.5) reflects our desire to find a low rank solution that, at the same time, has a low value of γ.

   (d) Set $k := 0$. Choose starting values for $\lambda^{(0)}$ and $\mu^{(0)}$, the parameters $\rho > 1$ and $0 < \rho_0 < 1$ and the tolerance ϵ.

2. **Optimization phase.**
   Set $k := k + 1$ and let $p_X, p_Y \in \mathbb{S}^{n_x}, p_\gamma \in \mathbb{R}$.

   (a) Using $\lambda = \lambda^{(k-1)}$ and $\mu = \mu^{(k-1)})$, solve (7.4) for the solution

   $$p = \begin{pmatrix} \operatorname{vech}(p_X) \\ \operatorname{vech}(p_Y) \\ p_\gamma \end{pmatrix},$$

   which is the step direction.

   (b) Update the variables as

   $$X^{(k)} = X^{(k-1)} + \alpha p_X, \quad Y^{(k)} = Y^{(k-1)} + \alpha p_Y, \quad \gamma^{(k)} = \gamma^{(k-1)} + \alpha p_\gamma,$$

or equivalently

$$x^{(k)} = x^{(k-1)} + \alpha p,$$

where $\alpha = 0.98$.

3. **Update phase**.
Update the Lagrangian multiplier λ using the following update rule.

$$\lambda^{(k)} = \lambda^{(k-1)} + \mu^{(k-1)}\hat{c}(x^{(k)}) \tag{7.6}$$

If $\hat{c}(x^{(k)}) > \epsilon$, update μ as follows.

$$\mu^{(k)} = \begin{cases} \rho\mu^{(k-1)} & \text{if } \hat{c}(x^{(k)}) > \rho_0\hat{c}(x^{(k-1)}) \\ \mu^{(k-1)} & \text{if } \hat{c}(x^{(k)}) \leq \rho_0\hat{c}(x^{(k-1)}) \end{cases} \tag{7.7}$$

The first option in (7.7) reflects our thought that the decrease in the equality constraint function value was not enough. Therefore we increase the penalty parameter. The second option reflects our content with the value of the constraint function, and we leave the penalty parameter at its current value.

4. **Terminating phase.**
If $\hat{c}(x^{(k)}) > \epsilon$, go to phase 2, otherwise we check the following.
   - if $\gamma^{(k)} < 0.99\gamma^{(k-1)}$ for three consequent iterates, it is likely we are close enough to a local optimum. Proceed to phase 5.
   - Otherwise, the objective function value is still decreasing, hence we continue the optimization, i.e., go back to phase 2.

5. **Recover controller phase**.
Recover the matrix P from X and Y as explained in Section 2.5 and construct an n_kth order H_∞ controller as explained in Section 2.6.

Remark 7.1. Note that in the optimization phase, one normally choose α in the interval $0 < \alpha \leq 1$ by performing a line search. However, we noticed that very small step-lengths α were taken which resulted in bad performance that might be caused by the *Maratos effect*. A solution could be to use a *watchdog* strategy to remedy this, but we have chosen to simply use $\alpha = 0.98$ which seem to work well. For more details on the Maratos effect and watchdog strategies, see Nocedal and Wright [2006].

The performance of the algorithm will be investigated in Chapter 8, where we will apply the algorithm to a set of problems. Comparisons will be made with the methods HIFOO and HINFSTRUCT, which are briefly presented in Section 8.1.2 and Section 8.1.3, respectively.

7.6 Chapter summary

In this chapter a partially augmented Lagrangian algorithm for design of low order H_∞ controllers is suggested. The algorithm calculates the search direction in each iteration by solving a convexified conic problem. A numerical evaluation is presented in Chapter 8.

Part III

Results and Conclusions

8 Numeric evaluations

In this chapter we will present the numeric evaluations of the suggested methods in chapters 5–7. This chapter consists of the following sections.

- In Section 8.1 we describe the benchmark problem library COMPl$_e$ib and two methods from the literature, HIFOO and HINFSTRUCT, which will be compared with our suggested methods.

- In Section 8.2 we evaluate the barrier method from Chapter 5 and the primal-dual method from Chapter 6. This version of the primal-dual method calculates a positive definite approximation of the Hessian by adding a multiple of the identity, as described in Section 6.4.5.

- In Section 8.3 we evaluate the primal-dual method from Chapter 6 that uses damped BFGS updating to approximate the Hessian as described in Section 6.4.5. This method is labeled the *quasi-Newton method* or *QN* in order to distinguish it from the above mentioned version of the primal-dual method.

- In Section 8.4 we evaluate the partially augmented Lagrangian method from Chapter 7.

The reason why the evaluations are presented separately, instead of all together, is that they were carried out at different points in time. Also the software that we use in the different sections of the evaluation are different versions.

8.1 Evaluation setup

Before the evaluation of the methods, we will present the evaluation setup. A core element of the evaluation is the problem library COMPl$_e$ib, which will be presented next. After that follows a brief presentation of two methods from the literature, HIFOO and HINFSTRUCT, that are used in the evaluation for comparison.

8.1.1 COMPl$_e$ib

Evaluation of the methods will be done using the benchmark problem library COMPl$_e$ib (Constrained matrix-optimization problem library), which is presented in Leibfritz [2004, 2006]. This library contains problems collected from the engineering literature and also pure academic problems. The set of problems include e.g. models of aircraft, helicopters, jet engines and reactors. The difficulty of solving these problems varies, as well as the dimensions of the problems. The library is easily obtainable from a web page, which makes it a natural choice when choosing benchmark problems.

8.1.2 HIFOO

HIFOO[1] (H-infinity fixed-order optimization) is a software package that can be run in MATLAB. HIFOO is described in Gumussoy and Overton [2008a], Burke et al. [2006] Gumussoy et al. [2009]. The supporting package HANSO, which is used for nonconvex, nonsmooth optimization, uses a hybrid algorithm that combines a quasi-Newton algorithm (BFGS) for the initial phase and a local bundle phase which tries to verify local optimality for the best points that BFGS finds. If the bundle phase does not succeed, a gradient sampling phase, see Burke et al. [2005], is used to improve the approximation of the local minimizer, and it returns a rough local optimality measure. One of the options available is to run HIFOO in fast mode, where the output is the best controller found after the BFGS phase of the algorithm.

HIFOO first chooses three random starting points with which the algorithm is initialized. Then it tries to find stabilizing controllers before it optimizes locally with respect to the closed loop H_∞ norm. Both optimization problems are nonsmooth and nonconvex. The latter optimization problem can be written as

$$\underset{K_A,K_B,K_C,K_D}{\text{minimize}} \|H_c(s)\|_\infty$$

where K_A, K_B, K_C, K_D and $H_c(s)$ are defined by (2.2)–(2.5). When HIFOO has finished the optimization, the controller with the best H_∞ norm of the three candidates is the output. Due to randomization of the initial points and the randomization in the gradient sampling phase, see Gumussoy and Overton [2008a], the experiments are not repeatable with the same result each time. In Gumussoy and Overton [2008a,b] the authors suggest that HIFOO is evaluated by running it ten times on each problem and choosing the best result. The results from applying

[1]Available from: `http://www.cs.nyu.edu/overton/software/hifoo/`.

HIFOO 2.0 on the system AC6 are found in Table B.2 in Appendix B, which is an extract from Ankelhed et al. [2010].

In Gumussoy and Overton [2008a], HIFOO is shown to perform very well compared to several other methods, and it has also been used several times in different applications, e.g. Robu et al. [2010], Dotta et al. [2009] and Wildschek et al. [2009].

The current version of the software is HIFOO 3.0, and it has been extended to also include H_2 controller synthesis, see Arzelier et al. [2011]. Since some parts of the evaluation here were carried out at earlier points in time, the version of HIFOO differs between the different evaluations.

8.1.3 HINFSTRUCT

HINFSTRUCT is included in the ROBUST CONTROL TOOLBOX in MATLAB, version 7.11 (R2010b), and based on the paper by Apkarian and Noll [2006b]. The method uses subgradient calculus, see Clarke [1990], to solve the problem by first minimizing the spectral abscissa of the closed loop system to find parameters for a stable controller. These parameters are then used as a starting point when optimizing locally to minimize the H_∞ norm. Thus HINFSTRUCT uses the same problem formulations as HIFOO but solves them in a different way. Since this software was released in the fall of 2010, it is only used in the evaluation of our latest contribution, the partially augmented Lagrangian method in Section 8.4.

HINFSTRUCT is a deterministic technique which does not involve any random elements, however extra starting points can be randomized upon request. According to Apkarian [2011], HINFSTRUCT should be initialized with two extra starting points when comparing its performance with HIFOO, since HIFOO uses three randomized starting points. However it is claimed that running HINFSTRUCT ten times is not needed. An extensive comparison of HINFSTRUCT and HIFOO can be found in Apkarian [2010]. More details on HINFSTRUCT can be found on the internet[2].

8.2 The barrier method and the primal-dual method

In this section we present the evaluation of the barrier method from Chapter 5 and the primal-dual method from Chapter 6 that calculates a positive definite approximation of the Hessian by adding a multiple of the identity, as described in Section 6.4.5.

All experiments in this section were performed on a DELL OPTIPLEX GX620 with 2GB RAM, INTEL P4 640 (3.2 GHz) CPU running under WINDOWS XP using MATLAB, version 7.4 (R2007a).

[2]See: `http://pierre.apkarian.free.fr/Software.html`.

8.2.1 Benchmarking problems

We chose to focus on two sets of problems in the benchmark library COMPl$_e$ib. The first set is the aircraft (AC) problems. The AC problems were chosen because they are often used in articles for benchmarking purposes. This kind of problems can be stabilized by a static controller, i.e., a controller with no states. The problems AC10, AC13 and AC14 have quite high number of states, and therefore are very complex to solve. The methods of this thesis were applied to these problems, but they failed due to large scale issues. However, HIFOO manages to find a zeroth and first order controller for AC10, if initialized by a stable controller, as presented in Gumussoy and Overton [2008a]. The optimal H_∞ norm of AC1 is almost zero, regardless of the order of the controller. For this reason it was not included in the comparisons in this section, even though the barrier method, the primal-dual method and HIFOO manage to find controllers for this system.

The second set is the reduced order controller (ROC) problems. The ROC problems were chosen because they are interesting in the sense that they cannot be stabilized by static output feedback controllers. Therefore a dynamic controller is required, i.e., a controller with at least n_c states, where n_c can be found in Table 8.8.

8.2.2 Evaluated methods

In this evaluation the following methods are compared.

- The barrier method from Chapter 5. The following parameter values were used.

$$\epsilon = 10^{-4}, \quad \Lambda_{\text{tol}} = 10^{-4}, \quad M = 10^4, \quad t_H = 0.1,$$
$$k_m = 0.1, \quad k_{\text{outer, max}} = 15, \quad k_{\text{inner, max}} = 10^3$$

- The primal-dual method from Chapter 6 that calculates a positive definite approximation of the Hessian by adding a multiple of the identity, as described in Section 6.4.5. The following parameter values were used.

$$r_{\text{tol}} = 10^{-5}, \quad \delta_\lambda = 10^{-4}, \quad k_{\max} = 1000$$

- HIFOO 1.0 with HANSO 1.0 using default values of the parameters. For a brief description of HIFOO see Section 8.1.2.

8.2.3 The tests

First, the full order controller (nominal controller) was computed. In these computations γ was minimized, which is a convex problem since there is no rank constraint. Note that the controller found this way may not always be stable, even though the closed loop system is. The nominal controller was computed using the Control System Toolbox in MATLAB, using the `hinfsyn` command, with the `'lmi'` option. The minimized upper bound on the performance measure obtained using the nominal controller is denoted γ^* and the achieved closed loop performance is denoted $||H_c^*||_\infty$.

Secondly, a reduced order controller was searched for, by first trying the same γ as the full order controller, and then decreasing the order of the controller by one each time. Then the performance measure was relaxed, i.e., by increasing γ. This was done in four steps, 5 %, 10 %, 20 % and 50 % increase of γ. For each of these steps, controllers were searched for with decreasing order. An increase of the upper bound of the H_∞ norm, γ, by more than 50% was considered not to be relevant, since the sacrificed performance is too big.

The evaluation of HIFOO was performed as follows. For each system, controller of orders from n_x–1 down to n_c was searched for. For each order, HIFOO was applied ten times, see Section 8.1.2 for a motivation. The median closed loop performance $||H_c||_\infty$ was calculated, sorted and placed in the appropriate group depending on its deviation from the nominal performance measure, γ^*. The different groups were 0%, 5%, 10%, 20% and 50%, analogously to what was done when evaluating the other two methods. The reason for choosing the median value is that for the benchmarking problem sets used, it requires as much time or more to run HIFOO once, as required to run the barrier method or the primal-dual method, as can be seen in Tables A.1–A.6 in Appendix A.

The required time for the algorithms to run was computed using the command `cputime`. By using the command `norm(H,inf,1e-6)` the H_∞ norm was computed, where `H` is the plant and the third argument is the tolerance used in the norm calculations (`1e-2` by default).

None of the methods that were investigated found a lower order controller for AC12 or ROC4 within +50 % of γ^*, which is why we have excluded those systems from the tables of results.

Remark 8.1. We are aware of that this way of comparing the methods may favor the barrier method and the primal-dual method. However it is difficult to design objective tests when the methods have different approaches to finding low order H_∞ controllers as is the case here.

8.2.4 An in-depth study of AC8 and ROC8

We will now take a deeper look into how the evaluation was done in two cases. The first system in the study is AC8, followed by ROC8.

AC8

The system AC8 has nine states, four inputs and five outputs. The system can, as we will see, be stabilized by a zeroth order controller but the closed loop H_∞ norm will increase dramatically compared to a full order controller.

The barrier method

The barrier method was applied to the problem AC8. The results are summarized in Table 8.1, which is an extract from Table A.1. The barrier method starts by using heuristics to find a good starting point, as described in Section 5.5.1. The starting point evaluated this way has shown to solve the problem completely for several cases in this study, however, this is not the case for this system. The barrier method is able to find a third order controller with the same value of γ as the nominal controller, i.e. $\gamma = 1.6220$. This is a drop in controller order of six states without any impact on the performance. If we are satisfied with a 10 % performance loss, we can get a controller of first order. Not until γ is increased by 50 %, a zeroth order controller can be found.

Table 8.1: *The table shows the results from running the barrier method on AC8. In the leftmost column the nominal performance data is listed. The second column from the right shows the required time to find the controller, and the rightmost column the number of iterations (#It) and restarts (#Rs) required.*

AC8, $n_x = 9$	γ	n_k	$\|H_c\|_\infty$	$t(s)$	#It(#Rs)
$\gamma^* = 1.6220$	+0 % (1.6220)	3	1.6216	573.5	432(1)
$\|H_c^*\|_\infty = 1.6194$	+5 % (1.7031)	2	1.6923	618.5	737(1)
	+10 % (1.7842)	1	1.7456	107.6	254(1)
	+50 % (2.4330)	0	2.1088	73.6	331(1)

The primal-dual method

For AC8, the primal-dual method did not perform as good as the barrier method. The results are summarized in Table 8.2, which is an extract from Table A.3. If we accept an increase in γ by 5 %, the method found a controller of order five, and an increase of 20 % resulted in a controller of order one. Similar to the barrier method, an increase in γ by 50 % resulted in that the primal-dual method found a zeroth order controller. The conclusion for AC8 is that the barrier method performed better, even though the primal-dual method found the zeroth order faster than the barrier method.

Table 8.2: *The table shows the results from running the primal-dual method on AC8. In the leftmost column the nominal performance data is listed. The second column from the right shows the required time to find the controller, and the rightmost column the number of iterations (#It) required.*

AC8, $n_x = 9$	γ	n_k	$\|H_c\|_\infty$	$t(s)$	#It
$\gamma^* = 1.6220$	+5 % (1.7031)	5	1.6787	354.3	130
$\|H_c^*\|_\infty = 1.6194$	+20 % (1.9464)	1	1.8654	99.2	139
	+50 % (2.4330)	0	2.2806	18.4	34

Hifoo

Hifoo was also applied to AC8. The results are summarized in Table 8.3, which is an extract from Table A.5. Without any performance loss the method was able to find a fourth order controller, i.e., one order more was required compared to the barrier method. When the required performance was relaxed by 5 %, Hifoo found a controller of order one, which is one order lower than what the barrier method managed for this system. When the performance requirement was relaxed by 50 %, the order of the controller dropped down to zero.

Table 8.3: *The table shows the results from running HIFOO on AC8. In the leftmost column the nominal performance data is listed. The rightmost column shows the required time to find the controller.*

AC8, $n_x = 9$	γ	n_k	$\|\|H_c\|\|_\infty$	$t(s)$
$\gamma^* = 1.6220$	+0 % (1.6220)	4	1.6202	429.3
$\|\|H_c^*\|\|_\infty = 1.6194$	+5 % (1.7031)	1	1.6516	130.2
	+50 % (2.4330)	0	2.0050	30.4

ROC8

We now turn to another system, ROC8, which is a sixth order system with four inputs and four outputs that requires a controller of order at least three to be stabilized.

The barrier method
The efficiency of the heuristics that the barrier method uses becomes evident here as it found a fifth order controller for ROC8 after 2.5 *s* using zero iterations of the main algorithm. It was needed to relax the performance requirement by 50 % in order to drop the order down to four. The results are shown in Table 8.4, which is an extract from Table A.2.

Table 8.4: *The table shows the results from running the barrier method on ROC8. In the leftmost column the nominal performance data is listed. The second column from the right shows the required time to find the controller, and the rightmost column the number of iterations (#It) and restarts (#Rs) that were required.*

ROC8, $n_x = 6$	γ	n_k	$\|\|H_c\|\|_\infty$	$t(s)$	#It(#Rs)
$\gamma^* = 3.4876$	+0 % (3.4876)	5	3.4870	1.7	0(0)
$\|\|H_c^*\|\|_\infty = 3.4870$	+50 % (5.2314)	4	5.2291	1.6	0(0)

The primal-dual method
For ROC8, the primal-dual method found a fifth order controller after 8.8 *s* using only 14 iterations without any performance loss. By relaxing the performance by 10 % the order dropped down to four, and to order three at 20 %. The results are summarized in Table 8.5, which is an extract from Table A.4.

HIFOO
When HIFOO was run on ROC8, the fifth order controller with no performance loss was not found. (It was not found consistently, since we are looking at median values.) A fourth order controller with 5 % performance loss and a third order one with 10 % respectively were found. The results are summarized in Table 8.6, which is an extract from Table A.6.

Table 8.5: *The table shows the results from running the primal-dual method on ROC8. In the leftmost column the nominal performance data is listed. The second column from the right shows the required time to find the controller, and the rightmost column the number of iterations (#It) that were required.*

ROC8, $n_x = 6$	γ	n_k	$\|H_c\|_\infty$	$t(s)$	#It
$\gamma^* = 3.4876$	+0 % (3.4876)	5	3.4870	8.8	14
$\|H_c^*\|_\infty = 3.4870$	+10 % (3.8363)	4	3.8343	127.9	303
	+20 % (4.1851)	3	4.1594	18.9	54

Table 8.6: *The table shows the results from running HIFOO on ROC8. In the leftmost column the nominal performance data is listed. The rightmost column shows the required time to find the controller.*

ROC8, $n_x = 6$	γ	n_k	$\|H_c\|_\infty$	$t(s)$
$\gamma^* = 3.4876$	+5 % (3.6613)	4	3.5638	255.5
$\|H_c^*\|_\infty = 3.4870$	+10 % (3.8356)	3	3.7595	144.7

8.2.5 Concluding the results

In this evaluation, the barrier method, the primal-dual method and HIFOO were applied to a total of 22 different systems. All the numeric values can be found in Appendix A and in this section we summarize the results.

The results for the AC systems and ROC systems can be seen in Table 8.7 and Table 8.8, respectively. The third, fourth and fifth columns in the tables show the minimum order of the controller that was found using the method listed in the first row of the table. The γ values are listed in the second column. A dash (–) means that no controller was found. An empty spot means that no controller with lower order was found, hence the controller listed above that spot can be used (since it satisfies stricter requirements). The notation (u) indicates that the found controller is unstable (but the closed loop system is still stable).

Next, some conclusions are made from the results. We begin with the AC problems and proceed with the ROC problems.

AC problems

Some points of interest concerning the AC problems are listed below.

- In general, all three methods find a controller with less than full order with the same performance as the nominal controller, or at least with a slightly relaxed performance requirement.
- All nominal controllers are stable, except the ones for AC4, AC11 and AC18.
- All the lower order controllers listed in Table 8.7 are stable, except for a few cases. The controllers for AC18 (found by the barrier method and HIFOO)

Table 8.7: *The table summarizes the evaluations of all three tested methods on the AC problems. See Section 8.2.5 for details.*

System	γ	Barr.(n_k)	PD(n_k)	HIFOO (n_k)
AC2, $n_x = 5$	+0 % (0.1115)	0	0	0
AC3, $n_x = 5$	+0 % (2.9701)	1	1	-
	+5 % (3.1186)			3
	+10 % (3.2671)			2
	+20 % (3.564)	0	0	1
	+50 % (4.4551)			0
AC4, $n_x = 4$	+0 % (0.5579)	1	2	2 (u)
(u)	+5 % (0.5858)		1	1 (u)
AC5, $n_x = 4$	+0 % (658.8393)	1	1	-
	+5 % (691.7813)	0	0	0
AC6, $n_x = 7$	+0 % (3.4328)	2	-	-
	+5 % (3.6045)		1	3
	+10 % (3.7761)			1
	+20 % (4.1194)	0	0	0
AC7, $n_x = 9$	+0 % (0.0384)	6	-	-
	+5 % (0.0403)	5	2	2
	+10 % (0.0422)	3		
	+20 % (0.0461)	2		
	+50 % (0.0576)		1	1
AC8, $n_x = 9$	+0 % (1.6220)	3	-	4
	+5 % (1.7131)	2	5	1
	+10 % (1.7842)	1		
	+20 % (1.9464)		1	
	+50 % (2.4330)	0	0	0
AC9, $n_x = 10$	+0 % (1.0004)	3	-	-
	+5 % (1.0504)	2	3	0
	+10 % (1.1004)		0	
AC11, $n_x = 5$	+0 % (2.8121)	1	1	-
(u)	+5 % (2.9527)		0	1 (u)
	+50 % (4.2181)	0		0
AC15, $n_x = 4$	+0 % (14.8759)	1	1	-
	+5 % (15.6197)	0	0	0
AC16, $n_x = 4$	+0 % (14.8851)	0	0	0
AC17, $n_x = 4$	+0 % (6.6125)	0	0	0
AC18, $n_x = 10$	+0 % (5.3967)	8 (u)	-	-
(u)	+10 % (5.9364)	7 (u)		
	+20 % (6.4760)			5 (u)
	+50 % (8.0950)	6 (u)		1 (u)

Table 8.8: *The table summarizes the evaluations of all three tested methods on the ROC problems. See Section 8.2.5 for details.*

System	γ	Barr.(n_k)	PD(n_k)	HIFOO (n_k)
ROC1, $n_x = 8$	+0 % (1.1311)	7	7	-
($n_c = 1$)	+5 % (1.1877)	6	5	-
	+10 % (1.2442)	2		2
	+20 % (1.3574)		3	
	+50 % (1.6967)	1	1	1
ROC2, $n_x = 9$	+0 % (0.0414)	5 (u)	-	-
($n_c = 1$), (u)	+5 % (0.0435)	3 (u)	3 (u)	-
	+10 % (0.0454)			3 (u)
	+50 % (0.0621)		2 (u)	2 (u)
ROC3, $n_c = 9$	+0 % (46.4986)	-	-	-
($n_c = 2$), (u)	+5 % (48.8235)	6 (u)	-	-
ROC5, $n_x = 6$	+0 % ($4.98 \cdot 10^{-5}$)	-	2	-
($n_c = 1$)	+5 % ($5.22 \cdot 10^{-5}$)	5		-
	+10 % ($5.47 \cdot 10^{-5}$)		1	-
	+50 % ($7.46 \cdot 10^{-5}$)	2		-
ROC6, $n_x = 3$	+0 % (21.6040)	2	2	2
($n_c = 2$)				
ROC7, $n_x = 4$	+0 % (1.1247)	1	1	1
($n_c = 1$)				
ROC8, $n_x = 6$	+0 % (3.4876)	5	5	-
($n_c = 3$)	+5 % (3.6619)			4
	+10 % (3.8363)		4	3
	+20 % (4.1851)		3	
	+50 % (5.2314)	4		
ROC9, $n_x = 4$	+0 % (2.2409)	3	3	-
($n_c = 2$)	+10 % (2.4612)			3
ROC10, $n_x = 5$	+0 % (0.0756)	-	-	-
($n_c = 1$), (u)	+5 % (0.0794)	1 (u)	4 (u)	-
	+10 % (0.0829)			3 (u)
	+20 % (0.0905)			1 (u)
	+50 % (0.1134)		1 (u)	

and some of the controllers of first order for AC11 found by HIFOO are not stable.

- For the system AC12, none of the methods can find a controller with a performance measure lower than +50 % of the nominal one. That system is such that even a controller of order $n_x - 1$ cannot be found that satisfies the constraints.

- For almost all controllers found by the barrier method, the heuristics for finding the initial point actually finds a solution to the problem. This means that Algorithm 8 does not need to run, hence the problem is solved very fast.
- According to Gumussoy and Overton [2008a], HIFOO is able to find controllers for AC10. The barrier method and the primal-dual method cannot find controllers for AC10, due to large-scale issues.

ROC problems

For the ROC problems, it is worthwhile to point out the following.

- The nominal controllers for ROC2, ROC3 and ROC10 are unstable.
- Neither of the three methods, that are investigated here, manage to find a controller for ROC2 that is stable. The closed loop system is of course stable, though.
- The barrier method is the only method that finds a controller for ROC3, even though the controller is unstable.
- For ROC5, the barrier method and the primal-dual method find controllers that are sufficiently good, while HIFOO does not.
- The heuristics used in the barrier method manage to find solutions for many of the ROC problems, similar as for the AC problems, and hence solves these problems very fast.

Quantifying the results

In this section we will attempt to quantify the results by grading the different methods. We will take two aspects into account: the ability to find low order controllers with no or little performance loss and the ability to find the lowest order controller with at most 50 % performance loss. If a method has the best result for a system, with respect to one of the criteria above, it will get one point. If several methods have equal result, these methods get one point each. We will base the grading on Table 8.7 and Table 8.8.

Prioritizing performance
The grading with respect to performance is done as follows. We look at the +0 % row for each system and give a point to the method that found the lowest order controller. If several methods have the same order, each method gets a point. If no method found a controller with +0 % performance loss, continue to +5 %, and so on. For example, all methods get a point for AC2, the barrier and the primal-dual method get points for AC3, but only the barrier method gets a point for AC4. The results from this grading procedure can be seen in Table 8.9. The results are clearly in favor of the barrier method, followed by the primal-dual method and last, HIFOO.

Table 8.9: *Grading with respect to closed loop performance. See Section 8.2.5 for details on the grading procedure.*

Method	Grade
The barrier method	21
The primal-dual method	13
HIFOO	5

Prioritizing lowest order

The grading with performance with respect to lowest order is performed as follows. For each system, we look at the controller with the lowest order, but at the same time having the least drop in performance. If several methods have found controllers with the same order and performance, each of them get a point. For example, for AC7 the primal-dual method and HIFOO get a point each. For AC9, HIFOO gets a point. The results from this grading procedure can be seen in Table 8.10. The results indicate that all three methods are equally good at finding the lowest order controllers within +50 % of the nominal peformance.

Table 8.10: *Grading with respect to lowest order of the controller. See Section 8.2.5 for details on the grading procedure.*

Method	Grade
The barrier method	15
The primal-dual method	16
HIFOO	14

8.2.6 Concluding remarks on the evaluation

- For some systems, ROC3 and ROC5, HIFOO cannot find any controller within at least 50 % of the nominal performance, while the barrier method succeeds in both cases, and the primal-dual method succeeds in the ROC5 case.

- HIFOO has shown to be able to solve problems with large dimensions as reported in Gumussoy and Overton [2008a]. This certainly favors HIFOO when systems of this kind are addressed.

- The interface for HIFOO is more developed than the interfaces for the barrier method and the primal-dual method. Since it is not needed to supply a value of γ when running, HIFOO certainly has an edge in terms of simplicity for the user.

- The random elements in HIFOO can cause unexpected and unreliable results. A solution for that is to calculate a large batch of controllers for the system to be controlled so that the random elements have less impact on the results. This, however, requires more computational time.

- The approach in the barrier method is somewhat ad hoc, but to the author it is quite remarkable that the performance is as good as it is, compared to the primal-dual method and HIFOO.

- The framework for the primal-dual method is thoroughly studied in the literature, so the fact that it performs quite well is not a big surprise.

- The barrier method is the best method at finding lower order controllers when performance is prioritized. The primal-dual method is second best followed by HIFOO. This statement is supported by the results in Table 8.9.

- When the aim is to find the lowest order controller that has a closed loop H_∞ norm no worse than +50 % of the nominal value, then the conclusion is that the barrier method, the primal-dual method and HIFOO perform equally well. This statement is supported by the results in Table 8.10.

This concludes the evaluation of the barrier method and the primal-dual method.

8.3 The quasi-Newton method

In this section we present the evaluation of the quasi-Newton method from Chapter 6 that calculates a positive definite approximation of the Hessian by using damped BFGS updating.

All experiments in this section were performed on a DELL OPTIPLEX GX620 with 2GB RAM, INTEL P4 640 (3.2 GHz) CPU running under WINDOWS XP using MATLAB, version 7.4 (R2007a).

8.3.1 Benchmarking problems

We have chosen to evaluate this method on a total of 48 different systems from the benchmarking problem library COMPl$_e$ib with the number of states ranging from 4 to 24. These systems are AC1-18, ROC1-10, NN11, REA3, CM1, EB1-4, HE6, HE7, JE2-3, AGS, BDT1, IH, CSE1, TG1, WEC1-3 and DLR1. Some systems resulted in too complex matrix computations for the quasi-Newton method to handle and therefore no controller was calculated for these systems. These systems are AC10, AC13-14, and JE2. This results in 44 systems being part of the evaluation.

8.3.2 Evaluated methods

In this evaluation the following methods are compared.

- The quasi-Newton method (QN for short).

- HIFOO 2.0 with HANSO 1.01. For a brief description of HIFOO, see Section 8.1.2. Both default mode and fast mode are evaluated.

8.3.3 The tests

The tests were carried out as follows. First, the full order controller (nominal controller) was computed. In these computations γ was minimized, which is a convex problem since there is no rank constraint involved. This controller was computed using the Control System Toolbox in MATLAB, using the `hinfsyn` command, with the '`lmi`' option. The minimized upper bound on the performance measure obtained using the nominal controller is denoted γ^* and the achieved closed loop performance is denoted $\|H_c^*\|_\infty$.

Define a vector of multipliers

$$\bar{\gamma} = \begin{pmatrix} 1 & 1.05 & 1.1 & 1.2 & 1.35 & 1.5 & 2 & 3 & 5 & 10 \end{pmatrix},$$

where each element in the vector $\bar{\gamma}$ refers to different degrees of relaxations of the performance requirement, i.e. +5 %, +10 %, etc. An increase of the upper bound of the H_∞ norm, γ, by more than a factor 10 is considered not to be of any interest here because the performance we sacrifice then is too much.

The quasi-Newton primal-dual method was applied in order to find reduced order controllers with $n_c \leq n_k \leq n_{k,\max}$ states, where n_c is a lower bound on the number of states of the controller required to stabilize the system and where $n_{k,\max} = \min(10, n_x - 1)$. Let $\gamma = \gamma^* \bar{\gamma}_i$, where $1 \leq i \leq 10$ is the index of the vector $\bar{\gamma}$ and $i = 1$ at start. This procedure is described by Algorithm 10.

Algorithm 10 An algorithm for iterating through orders and performance

```
Calculate nominal controller and calculate $\gamma^* = \|H_c^*\|_\infty$.
Set $i := 1$, $n_k := \min(10, n_x - 1)$.
while $n_k \geq n_c$ and $i \leq 10$ do
    Apply the QN algorithm using $\gamma = \gamma^* \bar{\gamma}_n$
    if success, save controller, set $n_k := n_k - 1$
    else, set $i := i + 1$
    end if
end while
```

If the QN algorithm is successful, a controller of a lower order is found. If not, the performance requirement is relaxed (i is increased). Note however that the above algorithm of iterating through orders and performance is just one way of evaluating the QN algorithm, and that no initial controller is needed to run the algorithm, just a value of γ.

The evaluation of HIFOO was performed as follows. For each system, controllers of orders from $n_{k,\max}$ down to n_c were searched for. For each order, HIFOO was applied ten times and the closed loop H_∞ norm and the computational time was saved for each run. The reason for applying it ten times is explained in Section 8.1.2.

The minimum and median H_∞ norm that was achieved by HIFOO for each system and controller order (n_k) was calculated. These are denoted *min* and *med*. The associated required computational time is sum of the required time for all ten runs, while for the median H_∞ norm the mean time of these ten runs are calculated.

The required time for the algorithms to run was computed using the command `cputime`. Using the command `norm(Gc,inf,1e-6)` the H_∞ norm was computed, where `Gc` is the closed loop system and the third argument is the tolerance used in the calculations. Note that we take all the time needed into account, even QN runs that fail to find a controller.

Remark 8.2. In contrast to the evaluation of the barrier method and the primal-dual method in Section 8.2.3, this way of evaluating the methods may favor HIFOO. The reason is that the quasi-Newton method is now applied without any knowledge of the closed loop H_∞ norm that may be achieved. In order to even out the odds somewhat, we tolerate a higher level of relaxation of the performance requirement in this evaluation compared to the evaluation in Section 8.2. However, in cases where the closed loop H_∞ norm when using a reduced order controller is close to the closed loop H_∞ norm when using a full-order controller it may instead favor the quasi-Newton method. Though, without any knowledge of the system it is impossible to draw any conclusions regarding this issue.

8.3.4 A case study, AC6

For the system AC6 with seven states ($n_x = 7$), the closed loop H_∞ norms are listed in Table 8.11. The time required to compute the results was 218 s using the QN algorithm and 50033 s using HIFOO in default mode, i.e., almost a factor of 230. When using HIFOO in fast mode, the required time is 8221 s, i.e. almost a factor of 38 compared to QN. Figure 8.1 and Figure 8.2 illustrate the achieved norms for the system AC6 using HIFOO default mode and fast mode, respectively.

In Table 8.11 we can see that the minimum values from HIFOO are lower than those from QN in 3 out of 7 cases, while ending in a draw in 3 cases resulting in a win ratio of 64 % for HIFOO and 36 % for QN. However, if we use the median values, the win ratio is only 43 % for HIFOO, but then the time required should be divided by 10, resulting in a factor ≈ 4 in time compared to QN. For HIFOO fast mode the H_∞ norms are higher in general as expected, but for $n_k = 2$ the norm is actually lower than what is achieved when using default mode, which is the result of the nonconvexity of the problem combined with the nondeterministic behavior of HIFOO.

In Appendix B an extract from Ankelhed et al. [2010] is presented. It shows detailed results from the evaluation of the quasi-Newton method and HIFOO, normal mode for the system AC6.

8.3.5 Extensive study

The quasi-Newton algorithm, HIFOO default and fast modes were applied to a total of 44 systems, as mentioned in Section 8.3.1. Note that the same tuning parameters in the quasi-Newton algorithm were used for all systems in the study, i.e., no individual tuning for different systems was done.

For each system in this study the win percent values were calculated and aver-

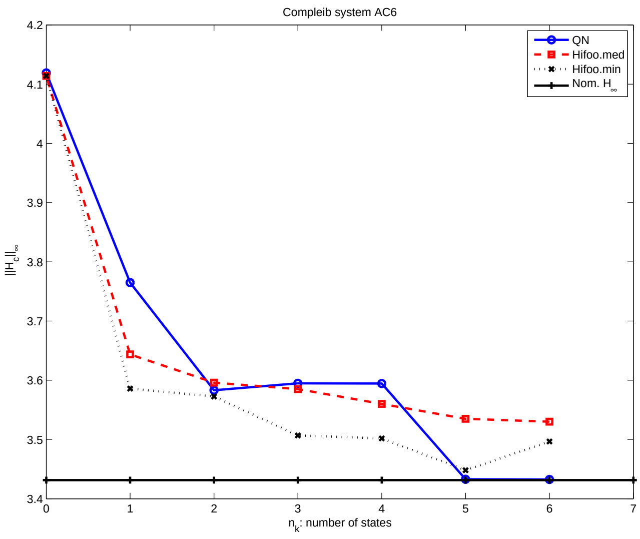

Figure 8.1: *The plot shows the closed loop H_∞ norm for the quasi-Newton method (QN) algorithm and HIFOO default mode, when applied to COMPl$_e$ib system AC6.*

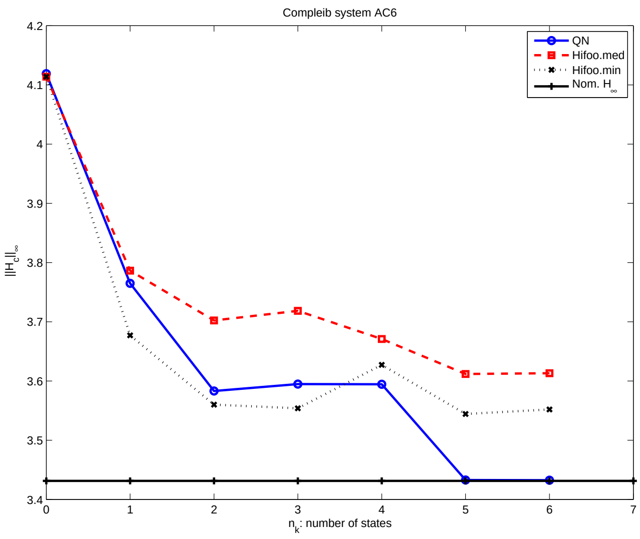

Figure 8.2: *The plot shows the closed loop H_∞ norm for the quasi-Newton method (QN) algorithm and HIFOO fast mode, when applied to COMPl$_e$ib system AC6.*

aged over all systems. In cases where the quasi-Newton algorithm cannot find a controller but HIFOO can, it was assumed a win for the latter. If the difference in norm obtained by the different methods was less than 1% it was declared a draw, so that minor numerical differences are not a big factor in the results. Also, any calculated controller that resulted in an unstable closed loop system, regardless of the used method, was counted as a failure.

The results from applying the algorithms on all 44 systems used in the evaluation can be seen in Ankelhed et al. [2010], but are summarized in Table 8.12 and Table 8.13. We can see that the default mode of HIFOO is better than the quasi-Newton algorithm in 61 % of the cases when the minimum H_∞ norm of the ten runs are chosen, but the required time is more than a factor of 17 of what is required by the quasi-Newton algorithm. However, if we compare the median H_∞ norm of the ten runs instead, the number is only 55 % of the cases but the required time is now approximately 1.7 times more compared to the quasi-Newton algorithm.

As for the fast mode option of HIFOO, the required time is about a quarter needed compared to the default mode, but the difference in the results is quite small. The best achieved norm is approximately the same but the median values are better.

Table 8.11: *Results from computing controllers for the COMPl$_e$ib system AC6 with 7 states. The best results (within an interval of 1 % of the best value if there are several close to each other) for each controller order are marked with bold font. Required computational time is 218 s for the quasi-Newton method (QN), 50033 s for HIFOO default mode and 8221 s for HIFOO fast mode.*

		HIFOO, Default mode		HIFOO, Fast mode	
n_k	QN, $\|\|H\|\|_\infty$	$\|\|H\|\|_\infty^{med}$	$\|\|H\|\|_\infty^{min}$	$\|\|H\|\|_\infty^{med}$	$\|\|H\|\|_\infty^{min}$
6	**3.4325**	3.5301	3.4967	3.6132	3.5519
5	**3.4328**	3.5349	**3.4481**	3.6119	3.5444
4	3.5944	3.5602	**3.5019**	3.6709	3.6273
3	3.5948	3.5851	**3.5069**	3.7186	3.5539
2	**3.5831**	3.5960	**3.5725**	3.7023	**3.5601**
1	3.7649	3.6438	**3.5859**	3.7863	3.6772
0	**4.1189**	**4.1140**	**4.1140**	**4.1140**	**4.1140**

Table 8.12: *Summary for all systems when comparing the quasi-Newton method (QN) and HIFOO, default mode. More details can be found in Section 8.3.5.*

	QN	HIFOO, min	HIFOO, med
Average win, min	39 %	**61** %	-
Average win, med	45 %	-	**55** %
Time required	$\mathbf{8.25 \cdot 10^4}$ **s**	$1.44 \cdot 10^6$ s	$1.44 \cdot 10^5$ s

Table 8.13: Summary for all systems when comparing the quasi-Newton method (QN) and HIFOO, fast mode. More details can be found in Section 8.3.5.

	QN	HIFOO, min	HIFOO, med
Average win, min	38 %	**62 %**	-
Average win, med	**50 %**	-	**50 %**
Time required	$8.25 \cdot 10^4\ s$	$3.45 \cdot 10^5$ s	$\mathbf{3.45 \cdot 10^4}$ **s**

To summarize the evaluation, we have drawn the following conclusions. The quasi-Newton algorithm is very fast for low order systems, as seen in the example (AC6) in Section 8.3.4. Though for higher order systems with more than 10 states, HIFOO is faster for most systems. The difference between the default mode and the fast mode options of HIFOO is not very significant, but the variance in the results is less when using the default mode.

8.3.6 Concluding remarks on the evaluation

The quasi-Newton algorithm has been evaluated and the results have been compared with HIFOO. The conclusion is that the quasi-Newton algorithm has comparable performance and speed, but HIFOO has an edge for higher order systems. For lower order systems (≤ 10 states), e.g. AC6, the proposed algorithm is much faster than HIFOO. When trying to synthesize controllers for some high order systems (> 20 states), it resulted in too big matrices for the quasi-Newton method to handle, while according to Gumussoy and Overton [2008a], HIFOO is able to find controllers for AC10 (55 states), which is one of these systems. How to handle systems with higher dimensions is something we are going to look into when developing the proposed method further. It would also be interesting to investigate if the calculation of the initial point can be done in a better way.

8.3.7 A comparison with the other methods

A similar comparison as was done in Section 8.2.5 was carried out but with the difference that HIFOO was replaced with the quasi-Newton method. In other words, the quasi-Newton method was compared with the barrier method and the primal-dual method. This comparison is presented in Appendix C. Also a grading when prioritizing the performance and when prioritizing the lowest order is presented in Table C.3 and Table C.4, respectively. The outcome is that the quasi-Newton is equally good as the barrier method but better than the primal-dual method when prioritizing the performance, while all three methods are equal when the lowest order is prioritized.

8.4 The partially augmented Lagrangian method

In this section we present the evaluation of the augmented Lagrangian method from Chapter 7.

All experiments were performed on a DELL OPTIPLEX GX620 with 2GB RAM, INTEL P4 640 (3.2 GHz) CPU running under WINDOWS XP using MATLAB, version 7.11 (R2010b).

8.4.1 Benchmarking problems

A collection of systems with different number of states ranging from 4 to 24 were chosen from the benchmarking library COMPl$_e$ib. These are AC2, AC5, AC18, CM1, EB4, JE3 and IH.

8.4.2 Evaluated methods

In this evaluation the following methods are compared.

- The augmented Lagrangian method.
- HIFOO 3.0 with HANSO 2.0.
- HINFSTRUCT (from MATLAB version 7.11).

For a brief description of HIFOO and HINFSTRUCT, see Section 8.1.

8.4.3 The tests

Since all methods in this evaluation attempts to minimize the closed loop H_∞ norm in the optimization while searching for a stabilizing controller, the evaluation procedure in this section is quite straight-forward. Next, the obtained closed loop H_∞ norms and required computational times are compared.

The results from the evaluation is presented in Table 8.14, where the H_∞ norms and required computational times for the respective methods are displayed. Note that the same settings were used throughout the whole evaluation for the augmented Lagrangian method. Cases where the augmented Lagrangian method had numerical problems are marked by *. HIFOO was run ten times for every combination of system and controller order using the default settings. The reason for applying it ten times is explained in Section 8.1.2. The best H_∞ norm from these ten runs is displayed in Table 8.14 while the required time is the sum of all ten runs. The options for HINFSTRUCT were modified in order to add two extra randomized starting points for reasons that are explained in Section 8.1.3.

The upper part of Table 8.14 shows the results from when controllers of either order zero or three were synthesized in order to evaluate both static output feedback controllers and reduced order feedback controllers. In cases where only the static output feedback controller is shown it is due to the fact that the higher order controllers turned out to have the same performance, thus there is no gain in using these results.

Since the computational complexity of HIFOO and HINFSTRUCT depend on the number of parameters in the controller while the augmented Lagrangian method does not, we chose to also include a system (IH) which has 11 input signals and 10 output signals in order to check if the results would differ. The number of decision variables for HINFSTRUCT and HIFOO is $n_k^2 + n_k n_y + n_u n_k + n_u n_y$ while for the augmented Lagrangian method it is $n_x(n_x + 1) + 1$, which means that the number of decision variables in our method is not affected by the number of states of the controller (n_k), inputs (n_u) or outputs (n_y), while the other methods are. The results of this evaluation are shown in the lower part of Table 8.14. For this example we also synthesized controllers of higher order than for the other examples.

As comparison we also included the results from the quasi-Newton method in Table 8.14. These results are taken from Ankelhed et al. [2010].

Table 8.14: *Results from evaluation on a collection of systems from COMPl$_e$ib. The first column displays the system name, the order of the system, the number of inputs and outputs and the order of the controller that was synthesized. The second, third, forth and fifth columns show the H_∞ norm and required time for the augmented Lagrangian method (AL), HINFSTRUCT (HS), HIFOO (HF) and the quasi-Newton method (QN) respectively. Cases where the augmented Lagrangian method had numerical problems are marked by *.*

Sys, (n_x,n_u,n_y,n_k)	$\|\|\cdot\|\|_\infty^{AL}$, t^{AL}	$\|\|\cdot\|\|_\infty^{HS}$, t^{HS}	$\|\|\cdot\|\|_\infty^{HF}$, t^{HF}	$\|\|\cdot\|\|_\infty^{QN}$, t^{QN}
AC2 (5,3,3,0)	0.11, 19.1 s	0.11, 3.47 s	0.11, 168 s	0.11, 2.89 s
AC5 (4,2,2,0)	670, 20.8 s	665, 1.80 s	669, 24.8 s	691, 3.81 s
AC5 (4,2,2,3)	660*, 10.3 s	658, 3.88 s	643, 1100 s	664, 3.70 s
AC18 (10,2,2,0)	14.8, 37.4 s	10.7, 2.97 s	12.6, 124 s	Fail, -
AC18 (10,2,2,3)	8.09, 36.9 s	6.51, 8.22 s	6.54, 3860 s	18.4, 13.9 s
CM1 (20,1,2,0)	0.84, 278 s	0.82, 1.91 s	0.82, 125 s	0.82, 142 s
EB4 (20,1,1,0)	2.46*, 460 s	2.06, 3.94 s	2.06, 10.5 s	2.32, 647 s
EB4 (20,1,1,3)	2.14, 370 s	1.82, 7.78 s	1.82, 1160 s	2.05, 192 s
JE3 (24,3,6,0)	8.74, 645 s	5.10, 5.31 s	5.10, 4880 s	9.62, 1080 s
JE3 (24,3,6,3)	2.89*, 1403 s	2.90, 11.6 s	2.89, 5910 s	3.11, 1210 s
IH (21,11,10,0)	1.88, 367 s	1.59, 38.0 s	1.90, 2450 s	0.00045, 255 s
IH (21,11,10,1)	1.86, 523 s	1.80, 43.0 s	1.80, 2410 s	0.00058, 288 s
IH (21,11,10,3)	1.49, 373 s	1.57, 51.0 s	1.74, 2170 s	0.00018, 386 s
IH (21,11,10,5)	1.39*, 868 s	1.15, 65.3 s	1.69, 2620 s	0.00021, 374 s
IH (21,11,10,7)	1.61*, 169 s	0.79, 86.2 s	1.72, 2450 s	0.00023, 1030 s

8.4.4 Results and conclusions

The results in the upper part of Table 8.14 indicate that the augmented Lagrangian method achieves comparable results in most cases. However HINFSTRUCT obtains the best results overall and is by far the fastest algorithm. However it does not always find the best result of the three methods. HIFOO achieves good results, but it requires a lot of computational time.

The results in the lower part of Table 8.14 shows that even if the number of parameters in the controller are many, HINFSTRUCT achieves better results than the augmented Lagrangian method in all cases but one. For these problems HIFOO does not perform as well as for the problems in the upper part of the table and the required time is far more than required by the other methods. However, if time is an issue, either using fast mode or just running it once would reduce the required computational time.

The remarkable point for the results regarding the system IH, is that the quasi-Newton method finds controllers in all cases that obtains a closed loop H_∞ norm that is almost zero, e.g. 0.00045 for the controller with zero states. In this case this can be explained by the fact that the full order controller has an H_∞ norm that is almost zero. Since the quasi-Newton method starts from the performance measure obtained using the full order controller and then relaxes it, it tries to find lower order controllers that, to begin with, has the same H_∞ norm. In this case there is even a controller with zero states that obtains this H_∞ norm. This controller seem to be very hard to find for the other three methods that attempts to minimize the H_∞ norm during the optimization. This indicates that the quasi-Newton method has an edge in cases where the obtainable H_∞ norm for a low order controller is close to the full order controller.

9 Conclusions and further work

In this chapter we present conclusions regarding evaluation of the suggested methods together with some concluding remarks. Some directions for future research are also suggested.

9.1 Summary

We have developed and implemented three different methods for solving the nonconvex problem related to low order H_∞ controller synthesis using linear matrix inequalities with a rank constraint.

The approach used in the thesis is based on formulating the rank constraint in the classical LMI formulation as a rational equation. By using the fact that the quotient in this equation is nonnegative on the feasible set, we can reformulate this as an optimization problem where a nonconvex rational function is to be minimized over a convex set defined by linear matrix inequalities.

To solve this optimization problem, we have proposed three different methods, namely a barrier method, a primal-dual method and a partially augmented Lagrangian method. A slight modification of the primal-dual method resulted in an additional method called the quasi-Newton method. These methods have been evaluated on a set of different examples and compared with two methods in the literature, HIFOO and HINFSTRUCT.

9.2 Conclusions

The following conclusions can be drawn. These conclusions answer the questions in Section 1.2.

- The barrier method performs well, at least for systems with 10 states or less as is seen in the evaluation in Section 8.2.
- The quasi-Newton method has shown to perform well for systems with around 20 states or less as seen in the evaluation in Section 8.3.
- The quasi-Newton method achieves slightly better performance than the primal-dual method and has similar performance as the barrier method, as indicated by the comparison in Section 8.3.7. However, the quasi-Newton method requires less computational time than both the other methods.
- The partially augmented Lagrangian method achieves good results for systems of around 25 states or less as seen in the evaluation in Section 8.4. For most systems this method performs better than the quasi-Newton method. Thus the partially augmented Lagrangian method obtains the best results overall among the suggested methods.
- Under certain circumstances, as for the system IH in Section 8.4, the quasi-Newton method achieves better results than the partially augmented Lagrangian method, HINFSTRUCT and HIFOO.
- The barrier method, the primal-dual method and the quasi-Newton method require an upper bound on the performance, γ, before the algorithms can start. This is in general a disadvantage compared to the partially augmented Lagrangian method, HIFOO and HINFSTRUCT. To some extent this can be dealt with by using e.g. Algorithm 10 (An algorithm for iterating through orders and performance).
- All the suggested methods in this thesis use symmetric matrix variables since their approach is based on Theorem 2.1. The size of these variables grows with the square of the order of the system. This fact makes these methods less suitable for systems of higher order.
- Methods that do not use approaches involving symmetric matrix variables, like HIFOO and HINFSTRUCT, seem better fitted for synthesizing low order controllers for systems of very high order. The reason for this is that the number of variables in these methods are equal to the number of free parameters in the controller. However, the associated optimization problems are not only nonconvex, but also nonsmooth and therefore difficult to solve.
- The evaluation in this thesis indicate that HINFSTRUCT obtains the best results overall, but not in all cases. HIFOO is a good alternative, but may require considerable computational time if it needs to be run 10 times. These are two of the best software packages for low order H_∞ controller synthesis that are available.

9.3 Future work

Since all suggested methods are local methods, the initial point is important for the performance of the algorithms. It would be interesting to look further into how to choose the initial point in a better way.

In cases where the barrier method does not find a local minimum and γ needs to be increased, warmstarting procedures could be implemented. Here we could use the point at which the algorithm terminated as a starting point in the new problem since the new problem is merely a relaxation of the previous one.

The impact of using other scaling matrices P for the primal-dual algorithm would be interesting to investigate.

A more efficient algorithm for calculating the derivatives of the coefficients of the characteristic polynomial was presented in Ankelhed [2011]. This algorithm was used together with a modified symmetric indefinite factorization in the partially augmented Lagrangian algorithm. It would be interesting to use this combination in the barrier method and in the primal-dual method to investigate what results that can be achieved.

The primal-dual method, the quasi-Newton method and the partially augmented Lagrangian method do not use a merit function combined with a line search. By using a proper merit function and implementing a line search with step lengths that satisfy the Wolfe conditions, better convergence properties of the algorithms could be obtained, see Nocedal and Wright [2006].

The termination criterion that is used in the partially augmented Lagrangian method is ad hoc and could be replaced with a more elaborate one, e.g. the one in Noll et al. [2004].

It would be interesting to apply a trust-region method to the problem formulations in e.g. (4.7) and (7.2) and investigate what performance that can be achieved.

Part IV

Tables with additional results

A
Barrier, primal-dual and HIFOO

A.1 Contents

In this appendix we present the tables with results from the evaluation described in Section 8.2. Those are as follows.

Table A.1:	The barrier method: AC systems.
Table A.2:	The barrier method: ROC systems.
Table A.3:	The primal-dual method: AC systems.
Table A.4:	The primal-dual method: ROC systems.
Table A.5:	HIFOO: AC systems.
Table A.6:	HIFOO: ROC systems.

A.2 Notation

In the first column, data about the system and the nominal controller is listed. The order of the system is denoted n_x, γ^* is the nominal value of γ and $||H_c^*||_\infty$ is the closed loop infinity norm.

In the second column, the γ-values for the lower order controllers are listed, with percent values describing how much bigger the γ-values are compared to the nominal values. In the third column, the order of the controllers are listed, and in the fourth column, the closed loop infinity norms are listed. In the fifth column the required time to find the controller is listed, and in the sixth column, the number of iterations used (and the number of restarts for the barrier method, see Section 5.5.2) are listed. The sixth column is not included in the tables for HIFOO.

Table A.1: *Results from running the barrier method on the AC problems.*

AC2, $n_x = 5$	γ	n_k	$\|\|H_c\|\|_\infty$	$t(s)$	#It(#Rs)
$\gamma^* = 0.1115$	+0 % (0.1115)	0	0.1115	1.8	0(0)
$\|\|H_c^*\|\|_\infty = 0.1115$					
AC3, $n_x = 5$					
$\gamma^* = 2.9701$	+0 % (2.9701)	1	2.9700	2.2	0(0)
$\|\|H_c^*\|\|_\infty = 2.9601$	+20 % (3.5641)	0	3.4783	16.8	338(0)
AC4, $n_x = 4$					
$\gamma^* = 0.5579$, (u)	+0 % (0.5579)	1	0.5577	6.2	69(2)
$\|\|H_c^*\|\|_\infty = 0.5575$					
AC5, $n_x = 4$					
$\gamma^* = 658.8$	+0 % (658.8393)	1	658.8390	2.0	0(0)
$\|\|H_c^*\|\|_\infty = 658.2$	+5 % (691.7813)	0	666.8880	10.2	277(0)
AC6, $n_x = 7$					
$\gamma^* = 3.4328$	+0 % (3.4328)	2	3.4328	2.7	0(0)
$\|\|H_c^*\|\|_\infty = 3.4314$	+20 % (4.1194)	0	4.1155	68.8	648(0)
AC7, $n_x = 9$					
$\gamma^* = 0.0384$	+0 % (0.0384)	6	0.0383	3.0	0(0)
$\|\|H_c^*\|\|_\infty = 0.0380$	+5 % (0.0403)	5	0.0400	3.0	0(0)
	+10 % (0.0422)	3	0.0413	2.6	0(0)
	+20 % (0.0461)	2	0.437	2.7	0(0)
AC8, $n_x = 9$					
$\gamma^* = 1.6220$	+0 % (1.6220)	3	1.6216	573.5	432(1)
$\|\|H_c^*\|\|_\infty = 1.6194$	+5 % (1.7031)	2	1.6923	618.5	737(1)
	+10 % (1.7842)	1	1.7456	107.6	254(1)
	+50 % (2.4330)	0	2.1088	73.6	331(1)
AC9, $n_x = 10$					
$\gamma^* = 1.0004$	+0 % (1.0004)	3	1.0004	4.0	0(0)
$\|\|H_c^*\|\|_\infty = 1.0003$	+5 % (1.0504)	2	1.0312	3.5	0(0)
AC11, $n_x = 5$					
$\gamma^* = 2.8121$, (u)	+0 % (2.8121)	1	2.8120	2.1	0(0)
$\|\|H_c^*\|\|_\infty = 2.8111$	+50 % (4.2181)	0	4.1211	1.6	0(0)
AC15, $n_x = 4$					
$\gamma^* = 14.8759$	+0 % (14.8759)	1	14.8759	2.0	0(0)
$\|\|H_c^*\|\|_\infty = 14.8714$	+5 % (15.6197)	0	15.5977	1.6	0(0)
AC16, $n_x = 4$					
$\gamma^* = 14.8851$	+0 % (14.8851)	0	14.8849	1.6	0(0)
$\|\|H_c^*\|\|_\infty = 14.8666$					
AC17, $n_x = 4$					
$\gamma^* = 6.6125$	+0 % (6.6125)	0	6.6124	1.5	0(0)
$\|\|H_c^*\|\|_\infty = 6.6124$					
AC18, $n_x = 10$					
$\gamma^* = 5.3967$, (u)	+0 % (5.3967)	8	5.4004, (u)	4.4	0(0)
$\|\|H_c^*\|\|_\infty = 5.3938$	+10 % (5.9364)	7	5.9121, (u)	3.6	0(0)
	+50 % (8.0950)	6	7.8395, (u)	3.5	0(0)

Table A.2: *Results from running the barrier method on the ROC problems.*

ROC1, $n_x = 8$	γ	n_k	$\|H_c\|_\infty$	$t(s)$	#It(#Rs)
$\gamma^* = 1.1311$	+0 % (1.1311)	7	1.1311	5.6	0(0)
$\|H_c^*\|_\infty = 1.1304$	+5 % (1.1877)	6	1.1686	1.9	0(0)
	+10 % (1.2442)	2	1.2440	134.4	276(0)
	+50 % (1.6967)	1	1.5716	66.9	271(0)
ROC2, $n_x = 9$					
$\gamma^* = 0.0414$, (u)	+0 % (0.0414)	5	0.0413, (u)	2.7	0(0)
$\|H_c^*\|_\infty = 0.0413$	+5 % (0.0435)	3	0.0427, (u)	2.3	0(0)
ROC3, $n_x = 9$					
$\gamma^* = 46.4986$, (u)	+5 % (48.8235)	6	48.7953, (u)	4.5	0(0)
$\|H_c^*\|_\infty = 46.4906$					
ROC5, $n_x = 6$					
$\gamma^* = 4.98 \cdot 10^{-5}$	+5 % ($5.22 \cdot 10^{-5}$)	5	$1.85 \cdot 10^{-5}$	20.4	37(1)
$\|H_c^*\|_\infty = 4.05 \cdot 10^{-5}$	+50 % ($7.46 \cdot 10^{-5}$)	2	$6.72 \cdot 10^{-5}$	2.5	0(0)
ROC6, $n_x = 3$					
$\gamma^* = 21.604$	+0 % (21.6040)	2	21.5550	1.4	0(0)
$\|H_c^*\|_\infty = 21.557$					
ROC7, $n_x = 4$					
$\gamma^* = 1.1247$	+0 % (1.1247)	1	1.1247	1.5	0(0)
$\|H_c^*\|_\infty = 1.1233$					
ROC8, $n_x = 6$					
$\gamma^* = 3.4876$	+0 % (3.4876)	5	3.4870	1.7	0(0)
$\|H_c^*\|_\infty = 3.4870$	+50 % (5.2314)	4	5.2291	1.6	0(0)
ROC9, $n_x = 4$					
$\gamma^* = 2.2409$	+0 % (2.2409)	3	2.2409	1.7	0(0)
$\|H_c^*\|_\infty = 2.2375$					
ROC10, $n_x = 5$					
$\gamma^* = 0.0756$, (u)	+5 % (0.0794)	1	0.0791, (u)	2.2	0(0)
$\|H_c^*\|_\infty = 0.0754$					

Table A.3: *Results from running the primal-dual method on the AC problems.*

AC2, $n_x = 5$	γ	n_k	$\|H_c\|_\infty$	$t(s)$	#It
$\gamma^* = 0.1115$	+0 % (0.1115)	0	0.1115	2.5	11
$\|H_c^*\|_\infty = 0.1115$					
AC3, $n_x = 5$					
$\gamma^* = 2.9701$	+0 % (2.9701)	1	2.9700	7.8	44
$\|H_c^*\|_\infty = 2.9601$	+20 % (3.5641)	0	3.5637	30.7	306
AC4, $n_x = 4$					
$\gamma^* = 0.5579$, (u)	+0 % (0.5579)	2	0.5575	5.5	41
$\|H_c^*\|_\infty = 0.5575$	+5 % (0.5858)	1	0.5844	9.0	108
AC5, $n_x = 4$					
$\gamma^* = 658.8$	+0 % (658.8393)	1	658.7662	21.7	251
$\|H_c^*\|_\infty = 658.2$	+5 % (691.7813)	0	670.7818	3.1	18
AC6, $n_x = 7$					
$\gamma^* = 3.4328$	+5 % (3.6045)	1	3.6038	110.3	279
$\|H_c^*\|_\infty = 3.4314$	+20 % (4.1194)	0	4.1161	12.6	33
AC7, $n_x = 9$					
$\gamma^* = 0.0384$	+5 % (0.0403)	2	0.0400	254.8	245
$\|H_c^*\|_\infty = 0.0380$	+50 % (0.0576)	1	0.0568	87.5	149
AC8, $n_x = 9$					
$\gamma^* = 1.6220$	+5 % (1.7031)	5	1.6787	354.3	130
$\|H_c^*\|_\infty = 1.6194$	+20 % (1.9464)	1	1.8654	99.2	139
	+50 % (2.4330)	0	2.2806	18.4	34
AC9, $n_x = 10$					
$\gamma^* = 1.0004$	+5 % (1.0504)	3	1.0298	454.9	200
$\|H_c^*\|_\infty = 1.0003$	+10 % (1.1004)	0	1.0954	120.2	185
AC11, $n_x = 5$					
$\gamma^* = 2.8121$, (u)	+0 % (2.8121)	1	2.8120	7.4	56
$\|H_c^*\|_\infty = 2.8111$	+5 % (2.9527)	0	2.9478	3.7	30
AC15, $n_x = 4$					
$\gamma^* = 14.8759$	+0 % (14.8759)	1	14.8758	3.9	34
$\|H_c^*\|_\infty = 14.8714$	+5 % (15.6197)	0	15.5450	2.8	26
AC16, $n_x = 4$					
$\gamma^* = 14.8851$	+0 % (14.8851)	0	14.8849	2.5	23
$\|H_c^*\|_\infty = 14.8666$					
AC17, $n_x = 4$					
$\gamma^* = 6.6125$	+0 % (6.6125)	0	6.6124	2.2	13
$\|H_c^*\|_\infty = 6.6124$					
AC18, $n_x = 10$					
$\gamma^* = 5.3967$, (u)	-	-	-	-	-
$\|H_c^*\|_\infty = 5.3938$					

Table A.4: *Results from running the primal-dual method on the ROC problems.*

ROC1, $n_x = 8$	γ	n_k	$\|\|H_c\|\|_\infty$	$t(s)$	#It
$\gamma^* = 1.1311$	+0 % (1.1311)	7	1.1305	81.0	36
$\|\|H_c^*\|\|_\infty = 1.1304$	+5 % (1.1877)	5	1.1537	34.3	20
	+20 % (1.3574)	3	1.3549	399.4	465
	+50 % (1.6967)	1	1.6717	92.4	262
ROC2, $n_x = 9$					
$\gamma^* = 0.0414$, (u)	+5 % (0.0435)	3	0.0425, (u)	136.5	87
$\|\|H_c^*\|\|_\infty = 0.0413$	+50 % (0.0621)	2	0.0595, (u)	771.1	668
ROC3, $n_x = 9$					
$\gamma^* = 46.4986$, (u)	-	-	-	-	-
$\|\|H_c^*\|\|_\infty = 46.4906$					
ROC5, $n_x = 6$					
$\gamma^* = 4.98 \cdot 10^{-5}$	+0 % ($4.98 \cdot 10^{-5}$)	2	$8.85 \cdot 10^{-6}$	34.1	159
$\|\|H_c^*\|\|_\infty = 4.05 \cdot 10^{-5}$	+10 % ($5.47 \cdot 10^{-5}$)	1	$8.89 \cdot 10^{-6}$	114.0	929
ROC6, $n_x = 3$					
$\gamma^* = 21.604$	+0 % (21.6040)	2	21.5523	2.0	18
$\|\|H_c^*\|\|_\infty = 21.557$					
ROC7, $n_x = 4$					
$\gamma^* = 1.1247$	+0 % (1.1247)	1	1.1246	2.5	23
$\|\|H_c^*\|\|_\infty = 1.1233$					
ROC8, $n_x = 6$					
$\gamma^* = 3.4876$	+0 % (3.4876)	5	3.4870	8.8	14
$\|\|H_c^*\|\|_\infty = 3.4870$	+10 % (3.8363)	4	3.8343	127.9	303
	+20 % (4.1851)	3	4.1594	18.9	54
ROC9, $n_x = 4$					
$\gamma^* = 2.2409$	+0 % (2.2409)	3	2.2408	3.2	19
$\|\|H_c^*\|\|_\infty = 2.2375$					
ROC10, $n_x = 5$					
$\gamma^* = 0.0756$, (u)	+5 % (0.0794)	4	0.0775, (u)	11.5	46
$\|\|H_c^*\|\|_\infty = 0.0754$	+50 % (0.1134)	1	0.1071, (u)	10.9	134

***Table A.5:** Results from running HIFOO on the AC examples.*

AC2, $n_x = 5$	γ	n_k	$\|H_c\|_\infty$	$t(s)$
$\gamma^* = 0.1115$	+0 % (0.1115)	0	0.1115	5.2
$\|H_c^*\|_\infty = 0.1115$				
AC3, $n_x = 5$				
$\gamma^* = 2.9701$	+5 % (3.1186)	3	3.0373	197.3
$\|H_c^*\|_\infty = 2.9601$	+10 % (3.2671)	2	3.2199	141.1
	+20 % (3.5641)	1	3.4331	38.7
	+50 % (4.4551)	0	3.6535	22.8
AC4, $n_x = 4$				
$\gamma^* = 0.5579$, (u)	+0 % (0.5579)	2	0.5573, (u)	37.8
$\|H_c^*\|_\infty = 0.5575$	+5 % (0.5858)	1	0.5589, (u)	33.0
AC5, $n_x = 4$				
$\gamma^* = 658.8393$	+5 % (691.7813)	0	689.7123	1.2
$\|H_c^*\|_\infty = 658.2496$				
AC6, $n_x = 7$				
$\gamma^* = 3.4328$	+5 % (3.6045)	3	3.5783	441.1
$\|H_c^*\|_\infty = 3.4314$	+10 % (3.7761)	1	3.6610	265.3
	+20 % (4.1194)	0	4.1140	67.9
AC7, $n_x = 9$				
$\gamma^* = 0.0384$	+5 % (0.0403)	2	0.0400	71.0
$\|H_c^*\|_\infty = 0.0380$	+50 % (0.0576)	1	0.0525	18.5
AC8, $n_x = 9$				
$\gamma^* = 1.6220$	+0 % (1.6220)	4	1.6202	429.3
$\|H_c^*\|_\infty = 1.6194$	+5 % (1.7031)	1	1.6516	130.2
	+50 % (2.4330)	0	2.0050	30.4
AC9, $n_x = 10$				
$\gamma^* = 1.0004$	+5 % (1.0504)	0	1.0054	150.8
$\|H_c^*\|_\infty = 1.0003$				
AC11, $n_x = 5$				
$\gamma^* = 2.8121$, (u)	+5 % (2.9527)	1	2.8281, (u)	89.1
$\|H_c^*\|_\infty = 2.8111$	+50 % (4.2181)	0	3.5283	14.2
AC15, $n_x = 4$				
$\gamma^* = 14.8759$	+5 % (15.6197)	0	15.1702	35.9
$\|H_c^*\|_\infty = 14.8714$				
AC16, $n_x = 4$				
$\gamma^* = 14.8851$	+0 % (14.8851)	0	14.8728	15.4
$\|H_c^*\|_\infty = 14.8666$				
AC17, $n_x = 4$				
$\gamma^* = 6.6125$	+0 % (6.6125)	0	6.6124	0.8
$\|H_c^*\|_\infty = 6.6124$				
AC18, $n_x = 10$				
$\gamma^* = 5.3967$, (u)	+20 % (6.4760)	5	6.4722, (u)	385.2
$\|H_c^*\|_\infty = 5.3938$	+50 % (8.0950)	1	7.7691, (u)	48.2

Table A.6: *Results from running HIFOO on the ROC examples.*

ROC1, $n_x = 8$	γ	n_k	$\|H_c\|_\infty$	$t(s)$
$\gamma^* = 1.1311$	+10 % (1.2442)	2	1.2438	56.1
$\|H_c^*\|_\infty = 1.1304$	+50 % (1.6967)	1	1.4256	21.3
ROC2, $n_x = 9$				
$\gamma^* = 0.0414$, (u)	+10 % (0.0455)	3	0.0448, (u)	310.4
$\|H_c^*\|_\infty = 0.0413$	+50 % (0.0621)	2	0.0506, (u)	147.2
ROC3, $n_x = 9$				
$\gamma^* = 46.4986$, (u)	-	-	-	-
$\|H_c^*\|_\infty = 46.4906$				
ROC5, $n_x = 6$				
$\gamma^* = 4.98 \cdot 10^{-5}$	-	-	-	-
$\|H_c^*\|_\infty = 4.05 \cdot 10^{-5}$				
ROC6, $n_x = 3$				
$\gamma^* = 21.604$	+0 % (21.6040)	2	21.5437	62.0
$\|H_c^*\|_\infty = 21.557$				
ROC7, $n_x = 4$				
$\gamma^* = 1.1247$	+0 % (1.1247)	1	1.1224	18.0
$\|H_c^*\|_\infty = 1.1233$				
ROC8, $n_x = 6$				
$\gamma^* = 3.4876$	+5 % (3.6619)	4	3.5638	255.5
$\|H_c^*\|_\infty = 3.4870$	+10 % (3.8363)	3	3.7595	144.7
ROC9, $n_x = 4$				
$\gamma^* = 2.2409$	+10 % (2.4649)	3	2.4175	162.9
$\|H_c^*\|_\infty = 2.2375$				
ROC10, $n_x = 5$				
$\gamma^* = 0.0756$, (u)	+10 % (0.0832)	3	0.0826, (u)	62.7
$\|H_c^*\|_\infty = 0.0754$	+20 % (0.0907)	1	0.0878, (u)	31.0

B The quasi-Newton method, AC6

B.1 Contents

This appendix is an extract from Ankelhed et al. [2010] showing the results of the evaluation of the quasi-Newton method and HIFOO 2.0 with HANSO 1.01 on the system AC6 from the benchmark library COMPl$_e$ib.

B.2 Notation

In Table B.1 the first column lists the γ-values, starting with the nominal value for the full order controller, and then the percentage increase of that value further down the first column. The other columns list the values obtained for the controllers of the order displayed at the top of the column. A dash (-) means that the corresponding controller was not calculated and an underlined value indicates that a controller of that order was calculated but failed to satisfy the requirements regarding stability, feasibility etc. These controllers are *not* counted as successful controllers. However all the time spent on these attempts are still summed up and counted. The bottom part of the tables shows the required time to calculate the controllers corresponding to the values on the upper part of the tables. At the bottom row, the total amount of time is summed up for the whole table.

In Tables B.2 the first column of the upper part lists the number of each of the ten runs. The other columns list the achieved H_∞ norms. The middle part lists the statistics regarding the achieved H_∞ norm, such as the minimum value, the maximum value, the mean value, the median value and the standard deviation. The lower part lists the same statistics, but for the required time.

Table B.1: *The table shows the results obtained when applying the quasi-Newton method on COMPl$_e$ib system AC6, 7 states.*

γ / k	6	5	4	3	2	1	0
3.433	3.432	3.433	$2.795{\cdot}10^1$	-	-	-	-
+5%	-	-	3.594	3.595	3.583	3.687	-
+10%	-	-	-	-	-	3.765	4.891
+20%	-	-	-	-	-	-	4.119
t [s]							
+0%	$3.652{\cdot}10^1$	$3.497{\cdot}10^1$	$3.456{\cdot}10^1$				
+5%			8.422	$1.019{\cdot}10^1$	9.078	$3.217{\cdot}10^1$	
+10%						8.594	$3.208{\cdot}10^1$
+20%							$1.163{\cdot}10^1$
Total:	$2.182{\cdot}10^2 s$						

Table B.2: *The table shows the results obtained when applying* H<span>IFOO</span>*, normal mode on COMPl$_e$ib system AC6, 7 states.*

Run / k	6	5	4	3	2	1	0
1	3.525	3.533	3.545	3.567	3.579	3.600	4.114
2	3.516	3.556	3.544	3.604	3.585	3.658	4.114
3	3.549	3.563	3.568	3.507	3.759	3.653	4.114
4	3.528	3.597	3.564	3.541	3.573	3.586	4.114
5	3.533	3.555	3.502	3.812	3.601	3.649	4.114
6	3.577	3.519	3.685	3.596	3.699	3.753	4.114
7	3.507	3.479	3.568	3.575	3.582	3.638	4.114
8	3.497	3.537	3.529	3.596	3.591	3.804	4.114
9	3.551	3.448	3.556	3.599	3.658	3.612	4.114
10	3.544	3.528	3.633	3.546	3.630	3.616	4.114
min	3.497	3.448	3.502	3.507	3.573	3.586	4.114
max	3.577	3.597	3.685	3.812	3.759	3.804	4.114
mean	3.533	3.532	3.569	3.594	3.626	3.657	4.114
med	3.530	3.535	3.560	3.585	3.596	3.644	4.114
std	$2.371{\cdot}10^{-2}$	$4.260{\cdot}10^{-2}$	$5.271{\cdot}10^{-2}$	$8.275{\cdot}10^{-2}$	$6.186{\cdot}10^{-2}$	$6.934{\cdot}10^{-2}$	$4.296{\cdot}10^{-5}$
t [s]							
tmin	$1.249{\cdot}10^3$	$8.141{\cdot}10^2$	$8.086{\cdot}10^2$	$6.695{\cdot}10^2$	$3.620{\cdot}10^2$	$3.883{\cdot}10^2$	$5.470{\cdot}10^1$
tmax	$1.408{\cdot}10^3$	$1.143{\cdot}10^3$	$9.928{\cdot}10^2$	$7.649{\cdot}10^2$	$5.868{\cdot}10^2$	$4.810{\cdot}10^2$	$1.662{\cdot}10^2$
tmean	$1.337{\cdot}10^3$	$1.030{\cdot}10^3$	$8.704{\cdot}10^2$	$7.095{\cdot}10^2$	$5.198{\cdot}10^2$	$4.350{\cdot}10^2$	$1.018{\cdot}10^2$
tmed	$1.331{\cdot}10^3$	$1.049{\cdot}10^3$	$8.599{\cdot}10^2$	$7.020{\cdot}10^2$	$5.336{\cdot}10^2$	$4.358{\cdot}10^2$	$9.108{\cdot}10^1$
tstd	$4.916{\cdot}10^1$	$8.688{\cdot}10^1$	$4.849{\cdot}10^1$	$3.042{\cdot}10^1$	$6.350{\cdot}10^1$	$2.893{\cdot}10^1$	$3.691{\cdot}10^1$

C
Barrier, primal-dual and quasi-Newton

C.1 Contents

In this appendix we present a comparison of the following methods.

- The barrier method from Chapter 5.
- The primal-dual method from Chapter 6 that calculates a positive definite approximation of the Hessian by adding a multiple of the identity, as described in Section 6.4.5.
- The quasi-Newton method from Chapter 6 that uses damped BFGS updating to approximate the Hessian as described in Section 6.4.5.

This evaluation is built on the evaluation of the barrier method and the primal-dual method in Section 8.2, but the results for HIFOO has been replaced with the results from the quasi-Newton method in order to able to compare these three methods. In other words, Table C.1 is similar to Table 8.7 and Table C.2 is similar to Table 8.8. The results for the quasi-Newton method is extracted from Ankelhed et al. [2010].

C.2 Quantifying the results

In this section the grading procedure that is explained in Section 8.2.5 are us to grade the barrier method, the primal-dual method and the quasi-Newton method. The grading is based on the results in Tables C.1 and Table C.2 and are presented in Table C.3 and Table C.4.

Table C.1: *The table summarizes the evaluations of all three tested methods on the AC problems. See Section 8.2.5 for details.*

System	γ	Barr.(n_k)	PD(n_k)	QN(n_k)
AC2, $n_x = 5$	+0 % (0.1115)	0	0	0
AC3, $n_x = 5$	+0 % (2.9701)	1	1	1
	+20 % (3.564)	0	0	
	+50 % (4.4551)			0
AC4, $n_x = 4$	+0 % (0.5579)	1	2	2
(u)	+5 % (0.5858)		1	
	+10 % (0.6137)			1
AC5, $n_x = 4$	+0 % (658.8393)	1	1	-
	+5 % (691.7813)	0	0	0
AC6, $n_x = 7$	+0 % (3.4328)	2	-	5
	+5 % (3.6045)		1	2
	+10 % (3.7761)			1
	+20 % (4.1194)	0	0	0
AC7, $n_x = 9$	+0 % (0.0384)	6	-	7
	+5 % (0.0403)	5	2	2
	+10 % (0.0422)	3		
	+20 % (0.0461)	2		
	+50 % (0.0576)		1	1
AC8, $n_x = 9$	+0 % (1.6220)	3	-	4
	+5 % (1.7131)	2	5	1
	+10 % (1.7842)	1		
	+20 % (1.9464)		1	
	+50 % (2.4330)	0	0	0
AC9, $n_x = 10$	+0 % (1.0004)	3	-	3 (u)
	+5 % (1.0504)	2	3	1
	+10 % (1.1004)		0	
	+20 % (1.2005)			0
AC11, $n_x = 5$	+0 % (2.8121)	1	1	1
(u)	+5 % (2.9527)		0	0
	+50 % (4.2181)	0		
AC15, $n_x = 4$	+0 % (14.8759)	1	1	1
	+5 % (15.6197)	0	0	0
AC16, $n_x = 4$	+0 % (14.8851)	0	0	0
AC17, $n_x = 4$	+0 % (6.6125)	0	0	0
AC18, $n_x = 10$	+0 % (5.3967)	8 (u)	-	-
(u)	+5 % (5.6665)			8 (u)
	+10 % (5.9364)	7 (u)		7 (u)
	+50 % (8.0950)	6 (u)		

Table C.2: *The table summarizes the evaluations of all three tested methods on the ROC problems. See Section 8.2.5 for details.*

System	γ	Barr.(n_k)	PD(n_k)	QN(n_k)
ROC1, $n_x = 8$	+0 % (1.1311)	7	7	6
($n_c = 1$)	+5 % (1.1877)	6	5	5
	+10 % (1.2442)	2		4
	+20 % (1.3574)		3	2
	+50 % (1.6967)	1	1	
ROC2, $n_x = 9$	+0 % (0.0414)	5 (u)	-	4 (u)
($n_c = 1$), (u)	+5 % (0.0435)	3 (u)	3 (u)	3 (u)
	+50 % (0.0621)		2 (u)	2 (u)
ROC3, $n_c = 9$	+0 % (46.4986)	-	-	-
($n_c = 2$), (u)	+5 % (48.8235)	6 (u)	-	6 (u)
ROC5, $n_x = 6$	+0 % ($4.98 \cdot 10^{-5}$)	-	2	1
($n_c = 1$)	+5 % ($5.22 \cdot 10^{-5}$)	5		
	+10 % ($5.47 \cdot 10^{-5}$)		1	
	+50 % ($7.46 \cdot 10^{-5}$)	2		
ROC6, $n_x = 3$	+0 % (21.6040)	2	2	2
($n_c = 2$)				
ROC7, $n_x = 4$	+0 % (1.1247)	1	1	1
($n_c = 1$)				
ROC8, $n_x = 6$	+0 % (3.4876)	5	5	5
($n_c = 3$)	+10 % (3.8363)		4	
	+20 % (4.1851)		3	3
	+50 % (5.2314)	4		
ROC9, $n_x = 4$	+0 % (2.2409)	3	3	3
($n_c = 2$)				
ROC10, $n_x = 5$	+0 % (0.0756)	-	-	1 (u)
($n_c = 1$), (u)	+5 % (0.0794)	1 (u)	4 (u)	
	+50 % (0.1134)		1 (u)	

Table C.3: *Grading with respect to closed loop performance. See Section 8.2.5 for details on the grading procedure.*

Method	Grade
The barrier method	18
The primal-dual method	11
The quasi-Newton method	16

Table C.4: *Grading with respect to lowest order of the controller. See Section 8.2.5 for details on the grading procedure.*

Method	Grade
The barrier method	15
The primal-dual method	17
The quasi-Newton method	17

Bibliography

F. Alizadeh, J.A. Haeberly, and M.L. Overton. Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results. *SIAM Journal on Optimization*, 8(3):746–768, 1998. ISSN 1052-6234. Cited on page 65.

D. Ankelhed. *On low order controller synthesis using rational constraints*. Licentiate thesis no. 1398, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Mar 2009. Cited on page 8.

D. Ankelhed. An efficient implementation of gradient and Hessian calculations of the coefficients of the characteristic polynomial of I - XY. Technical Report LiTH-ISY-R-2997, Department of Automatic Control, Linköping university, Sweden, 2011. URL `http://www.control.isy.liu.se/publications/doc?id=2387`. Cited on pages 8, 9, 45, and 107.

D. Ankelhed, A. Helmersson, and A. Hansson. Suboptimal model reduction using LMIs with convex constraints. Technical Report LiTH-ISY-R-2759, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Dec 2006. URL `http://www.control.isy.liu.se/publications/doc?id=1889`. Not cited.

D. Ankelhed, A. Helmersson, and A. Hansson. A primal-dual method for low order H-infinity controller synthesis. In *Proceedings of the 2009 IEEE Conference on Decision and Control*, Shanghai, China, Dec 2009. Cited on page 8.

D. Ankelhed, A. Helmersson, and A. Hansson. Additional numerical results for the quasi-Newton interior point method for low order H-infinity controller synthesis. Technical Report LiTH-ISY-R-2964, Department of Automatic Control, Linköping university, Sweden, 2010. URL `http://www.control.isy.liu.se/publications/doc?id=2313`. Cited on pages 8, 85, 97, 99, 102, 119, and 121.

D. Ankelhed, A. Helmersson, and A. Hansson. A quasi-Newton interior point method for low order H-infinity controller synthesis. *Accepted for publication in IEEE Transactions on Automatic Control*, 2011a. Cited on page 8.

D. Ankelhed, A. Helmersson, and A. Hansson. A partially augmented Lagrangian algorithm for low order H-infinity controller synthesis using rational constraints. *Submitted to the 2011 IEEE Conference on Decision and Control*, December 2011b. Cited on pages 9 and 73.

P. Apkarian. Comparison of HINFSTRUCT Matlab Robust Control Toolbox R2010b and HIFOO 3.0 with HANSO 2.0, 2010. URL `http://pierre.apkarian.free.fr/Benchv3.pdf`. Cited on page 85.

P. Apkarian. E-mail correspondance, 2011. Cited on page 85.

P. Apkarian and D. Noll. A prototype primal-dual LMI-interior algorithm for nonconvex robust control problems. Technical Report 01-08, Toulouse, 2001. Cited on page 5.

P. Apkarian and D. Noll. Controller design via nonsmooth multidirectional search. *SIAM Journal on Control and Optimization*, 44(6):1923–1949, 2006a. ISSN 0363-0129. Cited on pages 5 and 47.

P. Apkarian and D. Noll. Nonsmooth H_∞ synthesis. *IEEE Transactions on Automatic Control*, 51(1):71–86, 2006b. Cited on pages 5, 6, 7, 46, 47, and 85.

P. Apkarian and H.D. Tuan. Concave programming in control theory. *Journal of Global Optimization*, 15(4):343–370, 1999. Cited on page 5.

P. Apkarian and H.D. Tuan. Robust control via concave minimization local and global algorithms. *IEEE Transactions on Automatic Control*, 45(2):299–305, 2000. ISSN 0018-9286. Cited on page 5.

P. Apkarian, D. Noll, and H.D. Tuan. Fixed-order H-infinity control design via a partially augmented Lagrangian method. *International Journal of Robust and Nonlinear Control*, 13(12):1137–1148, 2003. ISSN 1049-8923. Cited on pages 5, 6, 7, 46, 74, and 76.

P. Apkarian, D. Noll, J.-B. Thevenet, and H.D. Tuan. A spectral quadratic-SDP method with applications to fixed-order H_2 and H_∞ synthesis. In *Proceedings of 2004 5th Asian Control Conference*, volume 2, pages 1337–1345, 2004. Cited on pages 5 and 74.

D. Arzelier, G. Deaconu, S. Gumussoy, and D. Henrion. H2 for Hifoo. *Submitted to the IFAC World Congress on Automatic control*, August 2011. Cited on pages 6 and 85.

A. Ben-Tal and M. Zibulevsky. Penalty/barrier multiplier methods for convex programming problems. *SIAM Journal on Optimization*, 7(2):347–366, 1997. Cited on page 5.

E.B. Beran. *Methods for optimization-based fixed-order control design*. Ph.D. Dissertation, Technical University of Denmark, Denmark, September 1997. Cited on pages 4 and 20.

S.J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147 – 150, 1984. ISSN 0020-0190. Cited on page 43.

D.P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic Press, Inc., New York, USA, 1982. ISBN 0-12-093480-9. Cited on page 36.

D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, Massachusetts, USA, 1995. ISBN 1-886529-14-0. Cited on pages 26 and 36.

P.T. Boggs and J.W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4: 1–51, 1995. Cited on page 62.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. Cited on pages 21, 22, 25, 26, 30, 51, and 55.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM Studies in Applied Mathematics. SIAM, 1994. ISBN 0-89871-334-X. Cited on pages 14, 43, and 53.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms, parts 1 and 2. *IMA Journal of Applied Mathematics*, 6(1):76–90 and 222–231, 1970. Cited on page 30.

S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming, Series B*, 95(2):329–357, 2003. Cited on page 62.

J.V. Burke, A.S. Lewis, and M.L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15 (3):751–779, 2005. Cited on pages 6 and 84.

J.V. Burke, D. Henrion, A.S. Lewis, and M.L. Overton. HIFOO - A MATLAB package for fixed-order controller design and H-infinity optimization. In *IFAC Proceedings of the 5th Symposium on Robust Control Design*, Toulouse, France, July 2006. Cited on pages 6 and 84.

S.H. Cheng and N.J. Higham. A modified cholesky algorithm based on a symmetric indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 19(4):1097–1110, 1998. Cited on page 29.

F.H. Clarke. *Optimization and nonsmooth analysis*. SIAM Class. Appl. Math. 5. SIAM, Philadelphia, 1990. Cited on pages 5 and 85.

A.R. Conn, N. Gould, and Ph.L. Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991. Cited on page 73.

A.R. Conn, N. Gould, A. Sartenaer, and Ph.L. Toint. Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6(3): 674–703, 1996. Cited on page 73.

R. Correa and C. Hector Ramirez. A global algorithm for nonlinear semidefinite programming. *SIAM Journal on Optimization*, 15(1):303–318, 2005. Cited on page 62.

R. Courant. Variational methods for the solution of problems with equilibrium and vibration. *Bulletin of the American Mathematical Society*, 49:1–23, 1943. Cited on page 35.

J. David. *Algorithms for analysis and design of robust controllers*. Ph.D. Dissertation, K.U. Leuven, March 1994. Cited on page 4.

J.E. Dennis, Jr. and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983. Cited on page 31.

D. Dotta, A.S. e Silva, and I.C. Decker. Design of power system controllers by nonsmooth, nonconvex optimization. In *2009 IEEE Power and Energy Society General Meeting, PES '09*, 2009. Cited on page 85.

J.C. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978. Cited on page 3.

J.C. Doyle. Structured uncertainty in control system design. In *IEEE Proceedings of the 24th Conference on Decision and Control*, pages 260–265, Fort Lauderdale, Florida, December 1985. Cited on page 4.

J.C. Doyle, K. Glover, P.P. Khargonekar, and B.A. Francis. State-space solutions to standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–47, 1989. ISSN 0018-9286. Cited on page 3.

G.E. Dullerud and F.G. Paganini. *A course in robust control theory: A convex approach*. Springer-Verlag, 2000. ISBN 0387989455. Cited on page 11.

L. El Ghaoui, F. Oustry, and M. AitRami. A cone complementarity linearization algorithm for static output-feedback and related problems. *IEEE Transactions on Automatic Control*, 42(8):1171–1176, 1997. ISSN 0018-9286. Cited on pages 5 and 46.

D.F. Enns. *Model reduction for control system design*. Ph.D. Dissertation, Stanford University, 1984. Cited on page 6.

B. Fares, P. Apkarian, and D. Noll. An augmented Lagrangian method for a class of LMI-constrained problems in robust control theory. *International Journal of Control*, 74(4):248–360, 2001. Cited on pages 5 and 74.

B. Fares, D. Noll, and P. Apkarian. Robust control via sequential semidefinite programming. *SIAM Journal on Control and Optimization*, 40(6):1791–1820, 2002. ISSN 0363-0129. Cited on pages 5 and 74.

M. Fazel, H. Hindi, and S.P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739, Viginia, June 2001. Cited on pages 55, 70, and 77.

A.V. Fiacco and G.P. McCormick. *Nonlinear programming: Sequential unconstrained minimization techniques*. Research Analysis Corporation, 1968. ISBN 0471258105. Cited on page 51.

R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970. Cited on page 30.

R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, New York, 1987. ISBN 0471915475. Cited on pages 31 and 36.

A. Forsgren. Optimality conditions for nonconvex semidefinite programming. *Mathematical Programming*, 88:105–128, 2000. ISSN 0025-5610. Cited on page 25.

A. Forsgren and P.E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4):1132–1152, 1998. Cited on page 61.

A. Forsgren, P.E. Gill, and M.H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002. Cited on pages 25 and 62.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. ISSN 1931-9193. Cited on page 5.

R.W. Freund, F. Jarre, and C.H. Vogelbusch. Nonlinear semidefinite programming: Sensitivity, convergence, and an application in passive reduced-order modeling. *Mathematical Programming*, 109(2-3):581–611, 2007. Cited on page 62.

M. Fu and Z.-Q. Luo. Computational complexity of a problem arising in fixed order output feedback design. *Systems and Control Letters*, 30(5):209–215, 1997. Cited on page 4.

P. Gahinet and P. Apkarian. A linear matrix inequality approach to H_∞ control. *International Journal of Robust and Nonlinear Control*, 4(4):421–48, 1994. ISSN 1049-8923. Cited on pages 4, 5, 7, 11, 16, and 17.

J.C. Geromel, C.C. de Souza, and R.E. Skelton. Static output feedback controllers: stability and convexity. *IEEE Transactions on Automatic Control*, 43(1):120–125, Jan 1998. Cited on page 4.

P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin, and M.H. Wright. On projected newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 36(2):183–209, 1986. Cited on page 52.

K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. *International Journal of Control*, 39(6):1115–1193, 1984. Cited on pages 11, 18, and 19.

P.J. Goddard and K. Glover. Controller approximation: Approaches for preserving H_∞ performance. *IEEE Transactions on Automatic Control*, 36:858–871, 1998. Cited on page 6.

K.-C. Goh, M.G. Safonov, and G.P. Papavassilopoulos. Global optimization for the biaffine matrix inequality problem. *Journal of Global Optimization*, 7:365–380, 1995. ISSN 0925-5001. Cited on page 4.

D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computations*, 24:23–26, 1970. Cited on page 30.

G.H. Golub and C.F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996. ISBN 0801854148. Cited on page 29.

N. Gould, D. Orban, and P. Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, 14:299–361, 2005. Cited on page 52.

K.M. Grigoriadis and R.E. Skelton. Low-order control design for LMI problems using alternating projection methods. *Automatica*, 32(8):1117–25, 1996. ISSN 0005-1098. Cited on pages 4 and 5.

S. Gumussoy and M.L. Overton. Fixed-order H^∞ controller design via HIFOO, a specialized nonsmooth optimization package. In *Proceedings of the 2008 American Control Conference*, pages 2750–2754, Seattle, 2008a. Cited on pages 6, 7, 46, 47, 84, 85, 86, 93, 94, and 100.

S. Gumussoy and M.L. Overton. Timings for numerical experiments on benchmark examples for fixed order H-infinity controller design, 2008b. URL `http://www.cs.nyu.edu/overton/papers/pdffiles/acc08times.pdf`. Cited on page 84.

S. Gumussoy, D. Henrion, M. Millstone, and M.L. Overton. Multiobjective robust control with HIFOO 2.0. In *Proceedings of the IFAC Symposium on Robust Control Design*, pages 144–149, Haifa, Israel, June 2009. Cited on page 84.

C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2): 342–361, 1996. Cited on page 65.

A. Helmersson. *Methods for robust gain scheduling*. PhD thesis, Linköping University, SE-581 83 Linköping, Sweden, Dec 1995. Cited on page 4.

A. Helmersson. On polynomial coefficients and rank constraints. Technical Report LiTH-ISY-R-2878, Department of Automatic Control, Linköping university, Sweden, 2009. URL `http://www.control.isy.liu.se/publications/doc?id=2119`. Cited on pages 6, 7, 39, 40, 43, 44, and 45.

M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. ISSN 0022-3239. Cited on page 36.

C.W.J. Hol, C.W. Scherer, E.G. Van Der Meché, and O.H. Bosgra. A nonlinear SDP approach to fixed-order controller synthesis and comparison with two other methods applied to an active suspension system. *European Journal of Control*, 9(1):13–28, 2003. Cited on pages 5 and 52.

D.C. Hyland and D.S. Bernstein. Optimal projection equations for fixed-order dynamic compensation. *IEEE Transactions on Automatic Control*, AC-29(11): 1034–1037, 1984. Cited on page 4.

T. Iwasaki. The dual iteration for fixed-order control. *IEEE Transactions on Automatic Control*, 44(4):783–788, April 1999. Cited on pages 4, 55, 70, and 77.

T. Iwasaki and R.E. Skelton. All controllers for the general H-infinity control problem: LMI existence conditions and state space formulas. *Automatica*, 30 (8):1307–1317, 1994. Cited on page 4.

T. Iwasaki and R.E. Skelton. XY-centring algorithm for the dual LMI problem: a new approach to fixed-order control design. *International Journal of Control*, 62(6):1257–1272, 1995. Cited on page 4.

F. Jarre. An interior method for nonconvex semidefinite programs. *Optimization and Engineering*, 1(4):347–372, 2000. ISSN 1573-2924. Cited on page 62.

M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5): 2327–2351, 2010. Cited on page 62.

S. Kanev, C. Scherer, M. Verhaegen, and B. De Schutter. Robust output-feedback controller design via local BMI optimization. *Automatica*, 40(7):1115–1127, 2004. Cited on page 5.

C. Kanzow, C. Nagel, H. Kato, and M. Fukushima. Successive linearization methods for nonlinear semidefinite programs. *Computational optimization and application*, 31(3):251–273, 2005. ISSN 1573-2894. Cited on page 62.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. Cited on page 52.

W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939. Cited on page 26.

D. Kavranoğlu and S.H. Al-Amer. New efficient frequency domain algorithm for H_∞ approximation with applications to controller reduction. *IEEE Proceedings: Control Theory and Applications*, 148(5):383–390, 2001. Cited on page 6.

M. Kocvara and M. Stingl. PENNON: a code for convex nonlinear and semidefinite programming. *Optimization methods and software*, 18(3):317–333, 2003. Cited on page 52.

M. Kocvara, F. Leibfritz, M. Stingl, and D. Henrion. A nonlinear SDP algorithm for static output feedback problems in COMPl$_e$ib. In *Proceedings of the IFAC World Congress on Automatic Control, Prague, Czech Republic*, July 2005. Cited on pages 5 and 52.

M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7(1):86–125, 1997. Cited on page 65.

H.W. Kuhn and A.W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, CA, 1951. University of California Press. Cited on page 26.

F. Leibfritz. An LMI-based algorithm for designing suboptimal static H_2/H_∞ output feedback controllers. *SIAM Journal on Control and Optimization*, 39 (6):1711–35, 2001. ISSN 0363-0129. Cited on page 5.

F. Leibfritz. COMPl$_e$ib: COnstraint Matrix optimization Problem library - a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Technical report, Department of Mathematics, Univ. Trier, Germany, 2004. Cited on page 84.

F. Leibfritz. COMPl$_e$ib: COnstrained Matrix optimization Problem library, 2006. Available from `http://www.complib.de/`. Cited on page 84.

F. Leibfritz and E.M.E. Mostafa. An interior point constrained trust region method for a special class of nonlinear semidefinite programming problems. *SIAM Journal on Optimization*, 12(4):1048–1074, 2002. Cited on pages 5 and 52.

B. Liêũ and P. Huard. La méthode des centres dans un espace topologique. *Numerische Mathematik*, 8(1):56–67, 1966. Cited on page 51.

J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. Available from `http://users.isy.liu.se/johanl/yalmip/`. Cited on pages 55 and 76.

H. Lütkepohl. *Handbook of matrices.* John Wiley & Sons, Ltd, 1996. ISBN 0471966886. Cited on page 42.

M.A. Mammadov and R. Orsi. A nonsmooth optimization approach to H_∞ synthesis. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference, CDC-ECC'05*, pages 6893–6898, 2005. Cited on pages 6 and 47.

N. Meggido. *Pathways to the optimal set in linear programming, in: N. Megiddo (Eds.), Progress in mathematical programming: Interior-point and related* *methods*, pages 131–158 (Chapter 8). Springer-Verlag New York, Inc., NY, USA, 1988. ISBN 0-387-96847-4. Cited on page 61.

S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992. Cited on pages 35, 61, and 69.

M. Mesbahi and G.P. Papavassilopoulos. Solving a class of rank minimization problems via semi-definite programs, with applications to the fixed order output feedback synthesis. In *Proceedings of the American Control Conference*, Albuqeurque, New Mexico, 1997. Cited on page 5.

R.D.C. Monteiro. Primal-dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization*, 7(3):663–678, 1997. Cited on page 65.

A.S. Nemirovski and M.J. Todd. Interior-point methods for optimization. *Acta Numerica*, 17:191–234, 2008. Cited on pages 52 and 62.

Yu.E. Nesterov and A.S. Nemirovski. *Interior point polynomial methods in convex programming: Theory and applications*. SIAM, Philadelphia, PA, 1994. Cited on page 52.

Yu.E. Nesterov and M.J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, 1997. Cited on pages 61 and 65.

Yu.E. Nesterov and M.J. Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on Optimization*, 8(2):324–364, 1998. Cited on pages 61 and 65.

C.N. Nett, D.S. Bernstein, and W.M. Haddad. Minimal complexity control law synthesis, Part 1: Problem formulation and reduction to optimal static output feedback. In *American Control Conference, 1989*, pages 2056–2064, Pittsburgh, PA, USA, June 1989. Cited on page 46.

J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 2nd edition, 2006. ISBN 0-387-30303-0. Cited on pages 21, 26, 29, 30, 31, 32, 35, 36, 37, 61, 68, 78, and 107.

D. Noll, M. Torki, and P. Apkarian. Partially augmented lagrangian method for matrix inequality constraints. *SIAM Journal on Optimization*, 15(1):161–184, 2004. Cited on pages 74 and 107.

R. Orsi. LMIrank: software for rank constrained LMI problems, 2005. Available from `http://users.rsise.anu.edu.au/~robert/lmirank/`. Cited on page 5.

R. Orsi, U. Helmke, and J.B. Moore. A Newton-like method for solving rank constrained linear matrix inequalities. *Automatica*, 42(11):1875–82, 2006. ISSN 0005-1098. Cited on pages 5, 55, 70, and 77.

A. Packard. Gain scheduling via linear fractional transformations. *Systems & Control letters*, 22(2):79–92, 1994. Cited on pages 4 and 16.

F.A. Potra and S.J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000. Cited on page 51.

M.J.D. Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969. Cited on page 36.

B. Robu, V. Budinger, L. Baudouin, C. Prieur, and D. Arzelier. Simultaneous H_∞ vibration control of fluid/plate system via reduced-order controller. In *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, pages 3146–3151, Dec. 2010. Cited on page 85.

R.T. Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, 35(2):183–238, 1993. Cited on page 36.

C. Scherer. *The Riccati inequality and state-space H_∞-optimal control*. Ph.D. Dissertation, Universität Würzburg, Germany, 1990. Cited on page 14.

D.F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computations*, 24:647–656, 1970. Cited on page 30.

S. Skogestad and I. Postlethwaite. *Multivariable feedback control: Analysis and design*. John Wiley & Sons, Inc., New York, NY, USA, 1996. ISBN 0471943304. Cited on pages 11 and 18.

M. Stingl. *On the solution of nonlinear semidefinite programs by augmented Lagrangian methods*. PhD thesis, Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen, Germany, 2006. Cited on page 52.

V.L. Syrmos, C.T. Abdallah, P. Dorato, and K. Grigoriadis. Survey paper: Static output feedback-a survey. *Automatica*, 33:125–137, February 1997. ISSN 0005-1098. Cited on pages 4 and 46.

J.-B. Thevenet, P. Apkarian, and D. Noll. Reduced-order output feedback control design with specSDP, a code for linear / nonlinear SDP problems. In *Proceedings of the 5th International Conference on Control and Automation, ICCA'05*, pages 465–470, 2005. Cited on pages 5 and 52.

M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001. Cited on pages 4 and 62.

M.J. Todd, K.C. Toh, and R.H. Tütüncü. On the Nesterov–Todd direction in semidefinite programming. *SIAM Journal on Optimization*, 8(3):769–796, 1998. Cited on pages 61, 65, 66, 68, 69, and 71.

K.C. Toh, M.J. Todd, and R.H. Tütüncü. SDPT3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(1-4): 545–581, 1999. Cited on pages 69, 70, 71, and 76.

K.C. Toh, R.H Tütüncü, and M.J Todd. SDPT3, 2006. Available from `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`. Cited on page 55.

A. Trofino. Sufficient LMI conditions for the design of static and reduced order controllers. In *Proceedings of the IEEE Conference on Decision and Control*, pages 6668–6673, 2009. Cited on page 6.

H.D. Tuan and P. Apkarian. Low nonconvexity-rank bilinear matrix inequalities: Algorithms and applications in robust controller and structure designs. *IEEE Transactions on Automatic Control*, 45(11):2111–2117, 2000. Cited on page 4.

J.G. VanAntwerp, R.D. Braatz, and N.V. Sahinidis. Globally optimal robust control for systems with time-varying nonlinear perturbations. *Computers and Chemical Engineering*, 21(Supplement 1):S125 – S130, 1997. Cited on page 4.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1): 49–95, 1996. Cited on page 4.

R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13(1-3): 231–252, 1999. Cited on page 62.

A. Wildschek, R. Maier, M. Hromcik, T. Hanis, A. Schirrer, M. Kozek, C. Westermayer, and M. Hemedi. Hybrid controller for gust load alleviation and ride comfort improvement using direct lift control flaps. In *In 3rd European Conference for Aerospace Sciences (EUCASS) 2009*, Versailles, France, 2009. Cited on page 85.

S.J. Wright. *Primal-dual interior-point methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-382-X. Cited on page 62.

H. Yamashita, H. Yabe, and K. Harada. A primal-dual interior point method for nonlinear semidefinite programming. *Mathematical Programming*, pages 1–33, 2011. ISSN 0025-5610. Cited on page 62.

G. Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Transactions on Automatic Control*, 26(2):301–320, Apr 1981. ISSN 0018-9286. Cited on page 3.

Y. Zhang. On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming. *SIAM Journal on Optimization*, 8 (2):365–386, 1998. Cited on page 64.

K. Zhou. A comparative study of H_∞ controller reduction methods. In *Proceedings of the American Control Conference 1995*, volume 6, pages 4015–4019, June 1995. Cited on page 6.

K. Zhou, J.C. Doyle, and K. Glover. *Robust and optimal control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. ISBN 0-13-456567-3. Cited on page 11.

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.