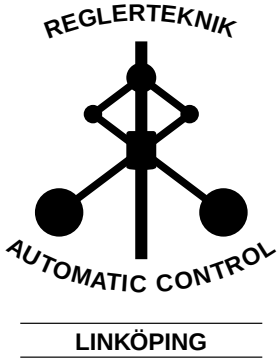
Linköping studies in science and technology. Thesis.
No. 1430

Structure exploitation in semidefinite programming for control

Rikard Falkeborn



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

Linköping 2010

This is a Swedish Licentiate's Thesis.

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

Linköping studies in science and technology. Thesis.
No. 1430

Structure exploitation in semidefinite programming for control

Rikard Falkeborn

falkeborn@isy.liu.se
www.control.isy.liu.se
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-7393-441-1 ISSN 0280-7971 LiU-TEK-LIC-2010:1

Printed by LiU-Tryck, Linköping, Sweden 2010

To Tina

Abstract

Many control problems can be cast as semidefinite programs. However, since the size of these problems grow quite quickly, the computational time to solve them can be quite substantial. In order to reduce the computational time, many proposals of how to tailor-make algorithms to various types of control problems can be found in the literature. In this thesis, two papers with similar ambitions are presented.

The first paper deals with the case where the constraints of the optimization problem are of the type that stems from the Kalman-Yakubovic-Popov lemma, and where some of these constraints are so called complicating constraints. This means the optimization problem will be greatly simplified if these constraints were not present. By the use of Lagrangian relaxation, the optimization problem is decomposed into smaller ones, which can be solved independently of each other. Computational results show that for some classes of problems, this algorithm can reduce the computational time compared to using a solver which does not take into account the nature of the complicating constraints.

In the second paper, the fact that many control-related semidefinite programs have matrix-valued variables is utilized to speed up computations. This implies that the corresponding basis matrices have a certain low-rank structure which can be exploited when formulating the equations for the search directions, something that was discovered in the 90s and is implemented in LMI Lab. However, much has happened in the area of semidefinite programming since the release of LMI Lab, and new, faster algorithms have been developed. However, the idea of using the lowrank structure in the basis matrices can still be used. We implement this, using the publicly available solver SDPT3 in combination with our code for formulating the system of equations for the search directions. In order to facilitate for potential users, we also describe how the modeling language YALMIP is changed so that this lowrank structure can be tracked, and how the code can be easily interfaced. Computational results show that the computational time is decreased.

Acknowledgments

First of all, I would like to thank Prof. Anders Hansson for his skillful guidance as my supervisor. Without him, this thesis would never have been finished. Also, Prof. Lennart Ljung who let me join the control group, and creates a very nice working atmosphere deserves my gratitude. I'd also like to thank Ulla Salaneck, who keeps track of everything and everyone at the control group.

Working at the Automatic Control group is a very pleasant experience. The discussions in the coffee room are always fun, and the fact that you can, at any given time, knock on any door in the corridor, and ask any question you like, is great, and certainly something that we should take advantage of more. Thank you all for your help.

When I started working here, there were several people who had just started or who started shortly after me. These people are Daniel Petersson, Lic. Christian Lyzell, Lic. Per Skoglar whom I shared a room with during my first time here, and Lic. Christian Lundkvist. I've really enjoyed taking graduate courses with you as well as discussing other important and unimportant subjects with you!

Dr. Johan Löfberg has great knowledge about optimization, and his will to share his knowledge is even greater. Dr. Christos Papageorgiou was post-doc here for two years, and helped me a lot with writing COFCLUO reports during that time. Thank you both.

Other people who has made working here less cumbersome are Dr. Gustaf Hendeby, who has written the LaTeX style files which were used in order to write this thesis and Dr. Henrik Tidefelt who always find time to help me as I never seem to make CVS, Emacs or Shapes behave the way I want them to.

For financial support, I would like to thank the European Commission under contract No. AST5-CT-2006-030768-COFCLUO and the Center for Industrial Information Technology (CENIIT).

Last, but not least, Tina. Thank you for your love, patience and support.

Rikard Falkeborn

Linköping, February 2010

Contents

Notational conventions

Abbreviations and Acronyms

Abbreviation	Meaning
KKT	Karush-Kuhn-Tucker
KYP	Kalman-Yakubovic-Popov
LFT	Linear fractional transformation
LMI	Linear matrix inequality
LTI	Linear time-invariant
SDP	Semidefinite programming

Symbols, Operators and Functions

Notation	Meaning
$A \prec (\preceq) \, 0$	A is negative (semi) definite
$A \succ (\succeq) \, 0$	A is positive (semi) definite
A^T	Transpose of matrix A
A^{-1}	Inverse of matrix A
$\mathrm{Tr}\, A$	Trace of matrix A
$\langle A, B \rangle_{\mathcal{Y}}$	Inner product of A and B over the space $\mathcal{Y}$
$A \otimes B$	Kronecker product of A and B
$A \otimes_s B$	Symmetric Kronecker product of $A, B \in \mathbf{S}^n$
vec(A)	Vectorization of A
svec(A)	Symmetric vectorization of A

1

Introduction

Optimization theory is about making things optimal, i.e. as good as possible. Before one can answer what is best, one need to define what one mean with ”best”. In some cases this is easy; in a running race, the optimal thing is to reach the goal line in as short period of time as possible. In other cases it is not that easy to define. What is the best way to drive between two cities for example? Is it better to drive fast to make the time to get there short, or is it better to drive slower and save fuel? The answer to that is usually ”it depends”. It depends on what is important to optimize over. In mathematical terms this corresponds to selecting your cost function, i.e. the function you want to minimize. One may also have restrictions, called *constraints*, which have to be fulfilled if we are going to be satisfied with the solution. In the car example, it may be that we do not want to go faster than a certain speed due to speed limits, or we need to be at the destination at a certain time in order for us not to be late.

Automatic control is about making systems behave the way we want. In most modern cars, the driver can activate a *cruise controller* which regulates the speed of the car, keeping it at the desired speed. There are much to consider when designing a control system like that and, many design methods have been proposed to design control systems. When the control system has been designed, one needs to evaluate it, and verify that the proposed controller fulfills the design specifications. It turns out that many of the design methods and analysis methods involve the solution of optimization problems. As control systems become more and more complex, the optimization problems needed to be solved also become more complex, which leads to long solution times and, in some cases, it may also be impossible to solve these problems due to the complexity. This motivates research which aims at tailor-making optimization algorithms which solves control problems.

1.1 Background

A certain class of optimization problems that has became a hot research area during the past two decades is semidefinite programming. An optimization problems is often refered to as a program, while semidefinite means that one or more variables in our problem is constrained to be symmetric and positive (or negative) semidefinite.

The fact that there exist several software packages which are publicly released makes it relatively easy for the non-expert to use semidefinite programming. In addition to this, many control related problems can be cast as semidefinite programs. Hence, since there are several publicly available solvers, one might think that solving semidefinite programs is an easy task. However, the number of variables in many problems quickly become prohibitive. In control problems this is often due to the introduction of a large Lyapunov variable which has the same size as the number of states in a possibly extended system. Therefore, there have been a number of efforts to tailor-make algorithms for various numbers of control problems (Wallin et al., 2008, 2009, Vandenberghe et al., 2004, Liu and Vandenberghe, 2007). In this thesis, two such tailor-made algorithms are presented.

1.2 Contributions

Paper A is an edited version of

> R. Falkeborn and A. Hansson. A decomposition algorithm for KYP-SDPs. In *Proceedings of the European Control Conference*, pages 3202–3207, August 2009.

The difference between these two papers are minor changes due to typographical reasons. This paper describes an algorithm which solves so called KYP-SDP-problems with a certain structure.

Paper B is an edited version of

> R. Falkeborn, J. Löfberg, and A. Hansson. Lowrank exploitation in semidefinite programming for control. Technical Report LiTH-ISY-R-2927, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, January 2010.

Just as in Paper A, the changes are due to typographical reasons. This paper presents an extension to the modeling language YALMIP (Löfberg, 2004) which automatically detects and exploits the fact that many control problems involve matrix variables, which leads to basis matrices having a certain low rank structure.

Work not included in this thesis where the author has made minor contributions has been published in

> C. Papageorgiou, R. Falkeborn, and A. Hansson. Formulation of the stability margins clearance criterion as a convex optimization problem. In *Proceedings of the 6th IFAC Symposium on Robust Control Design, 2009*, pages 325–330, June 2009.

1.3 Thesis Outline

The thesis is outlined as follows. Part I of the thesis contains some background material, where Chapter 2 introduces semidefinite programming. Chapter 3 is devoted to *Lagrangian Relaxation*, and describes some techniques that can be used when the constraints of the optimization problem have a specific structure. The last in Part I is Chapter 4 where some applications of semidefinite programming in systems and control theory are presented. Part II contains the publications mentioned in Section 1.2. Paper A presents an algorithm for a certain type of optimization problems, and Paper B presents an extension to the modeling language YALMIP (Löfberg, 2004) that utilizes the fact that in many semidefinite programming problems, the variables have a certain matrix structure.

Part I

Background Material

2

Semidefinite Programming

Optimization is the theory of finding the maximal or minimal value of a mathematical function, given that certain equality or inequality constraints hold. Semidefinite programming is a certain type of optimization where one or more variables are constrained to be positive semidefinite. This type of problems are convex optimization problems and this chapter will give a brief overview of some of the properties of these problems.

For a good introduction to convex optimization, see e.g. (Boyd and Vandenberghe, 2004). For a more detailed discussion of semidefinite programming, we refer to (Wolkowicz et al., 2000).

2.1 Definitions

Before we start by defining the problems we intend to tackle in the thesis, some preliminaries must be defined.

Definition 2.1 (Vectorization). The *vectorization* operator $\mathbf{vec}(U) : \mathbf{R}^{n\times n} \to \mathbf{R}^{n^2}$ is defined by

$$\mathbf{vec}(U) = \left[u_{11}, \ldots, u_{n1}, u_{21}, \ldots, u_{n2}, \ldots, u_{nn}\right]^T, \tag{2.1}$$

i.e. the vector that we get by stacking the columns of U.

Definition 2.2 (Symmetric vectorization). We define the *symmetric vectorization* operator $\mathbf{svec}(U) : \mathbf{S}^n \to \mathbf{R}^{n(n+1)/2}$ is by

$$\mathbf{svec}(U) = \left[u_{11}, \sqrt{2}u_{21}, \ldots, \sqrt{2}u_{n1}, u_{22}, \sqrt{2}u_{32}, \ldots, \sqrt{2}u_{n2}, \ldots, u_{nn}\right]^T, \tag{2.2}$$

i.e., $\mathbf{svec}()$ is an operator taking symmetric matrices to vectors.

The term $\sqrt{2}$ is included so that the inner products of U and V and $\mathbf{svec}(U)$ and $\mathbf{svec}(V)$ are equal, i.e.

$$\langle U, V\rangle = \langle \mathbf{svec}(U), \mathbf{svec}(V)\rangle\,. \tag{2.3}$$

Here we have abused the notation by not specifying the space in which the inner products are defined. We will continue to do this throughout the remainder of the thesis if the space is clear from the context.

Next, we define the *symmetric Kronecker product*.

Definition 2.3 (Symmetric Kronecker product). The *symmetric Kronecker product* between two matrices $A, B \in \mathbf{R}^{n\times n}$ is definted as

$$A \otimes_s B = \frac{1}{2} U \left(A \otimes B + B \otimes A\right) U^T, \tag{2.4}$$

where $U \in \mathbf{R}^{n(n+1)/2 \times n^2}$ is a matrix such that

$$U\mathbf{vec}(H) = \mathbf{svec}(H) \text{ and } U^T\mathbf{svec}(H) = \mathbf{vec}(H), \tag{2.5}$$

for all symmetric matrices H. One way to find the matrix U is the following. 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 the columns in the order $(1,1),(2,1),\ldots,(n,1),(1,2),\ldots,(n,2),(1,3),\ldots,(n,n)$, and then

$$U_{(i,j),(k,l)} = \begin{cases} 1 & \text{if } i=j=k=l, \\ \sqrt{2} & \text{if } i=k\neq j=l \text{ or } i=l\neq j=k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

2.2 Introduction

The basic optimization problem we will discuss in this chapter is

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & F_0 + \mathcal{F}(x) = S \\ & S \succeq 0, \end{aligned} \tag{2.7}$$

where $\mathcal{F}(x) = \sum_{i=1}^m x_i F_i$, $x \in \mathbf{R}^m$, $c \in \mathbf{R}^m$, $F_i \in \mathbf{S}^n$ and $S \in \mathbf{S}^n$.

The constraint $S \succeq 0$ means the matrix S is positive semidefinite. Much has been written on how to solve this optimization problem, see for example (Wolkowicz et al., 2000).

The semidefinite constraint in (2.7) is a convex constraint, which can be verified using the definition of semidefinite matrices and the definition of convex sets, see e.g. (Boyd and Vandenberghe, 2004) for details. Convex optimization problems are very attractive from a theoretical point of view since, if some technical conditions hold, there exist algorithms which solve the problems to a prespecified tolerance in a number of iterations that are polynomial in the number of variables, the size of the constraints and the specified tolerance. Since each iteration can be computed in polynomial time, the whole algorithm can be run in a time that is polynomial, and hence tractable, at least from a theoretical point of view.

Many problems in systems and control theory involve semidefinite programs with matrix variables, i.e. for example find $P \in \mathbf{S}^n$ such that

$$\begin{array}{ll} \min\limits_{P} & \operatorname{Tr} P \\ \text{s.t.} & A^T P + PA \succeq I, \end{array} \tag{2.8}$$

with $A \in \mathbf{R}^{n\times n}$. This can easily be put on the form (2.7) as follows. Let $E_1, \ldots, E_m$ be a basis for symmetric $n \times n$ matrices, with $m = n(n+1)/2$. Now, by letting $F_0 = -I$ and $F_i = A^T E_i + E_i A$, the problem (2.8) can easily be put on the form (2.7), and a number of available solvers can be used in order to solve the problem. We remark that the tedious and error-prone task of transforming (2.8) into (2.7) is best left to well-tested software, such as YALMIP (Löfberg, 2004). As we will see later, in some cases keeping the structure of (2.8) may be beneficial.

2.3 Duality

The Lagrangian (Boyd and Vandenberghe, 2004) function to (2.7) is, if we eliminate S from (2.7)

$$L(x, Z) = c^T x + \langle F_0 + \mathcal{F}(x), Z\rangle\,, \tag{2.9}$$

where $Z \in \mathbf{S}^n$.

The *Lagrange dual function* is defined as the minimal value of the Lagrangian over x.

$$g(Z) = \inf_{x\in\mathcal{D}} L(x, Z) = \inf_{x\in\mathcal{D}} c^T x + \langle -F_0 - \mathcal{F}(x), Z\rangle\,. \tag{2.10}$$

Note that in our case, the domain $\mathcal{D} = \mathbf{R}^m$, but for future reference, we emphasize that the infimum should be taken over the domain of the objective function. We remark that when $g(Z)$ is unbounded from below, the value of $g(Z)$ is said to be $-\infty$. We also remark that $g(Z)$ is a concave function.

If we require $Z \succeq 0$, we have

$$g(Z) \leq c^T x^*, \tag{2.11}$$

where x^* denotes the optimal point of (2.7). To see this, pick any feasible point $\tilde{x}$, i.e. a point such that $F_0 + \mathcal{F}(\tilde{x}) \succeq 0$, then

$$L(\tilde{x}, Z) = c^T \tilde{x} - \underbrace{\langle F_0 + \mathcal{F}(\tilde{x}), Z\rangle}_{\geq 0} \leq c^T \tilde{x}. \tag{2.12}$$

Hence we have

$$g(Z) = \inf_{x\in\mathcal{D}} L(x, Z) \leq L(\tilde{x}, Z) \leq c^T \tilde{x}. \tag{2.13}$$

Since (2.13) holds for any feasible $\tilde{x}$, it also holds for the optimal value x^*.

The inequality (2.11) gives a lower bound on the optimal value of our problem. Naturally, we want to find a bound that is as good as possible. It is therefore natural to consider the optimization problem

$$\begin{array}{ll} \max\limits_{Z} & g(Z) \\ \text{s.t.} & Z \succeq 0, \end{array} \tag{2.14}$$

called the dual problem.

We have

$$L(x, Z) = c^T x + \langle -F_0 - \mathcal{F}(x), Z\rangle = \langle -F_0, Z\rangle + \langle c - \mathcal{F}^*(Z), x\rangle\,, \tag{2.15}$$

where $\mathcal{F}^*(Z)$ is the dual function of $\mathcal{F}(x)$, which is equal to

$$\mathcal{F}^*(Z) = \begin{bmatrix} \langle F_1, Z\rangle \\ \vdots \\ \langle F_m, Z\rangle \end{bmatrix}. \tag{2.16}$$

We note that $g(Z) = -\infty$ unless we require $c - \mathcal{F}^*(Z) = 0$. When solving (2.14), we do not need to consider those Z for which $g(Z) = -\infty$. Hence the Lagrange dual problem to (2.7) can be stated as

$$\begin{aligned} \max_{Z} \quad & \langle -F_0, Z\rangle \\ \text{s.t.} \quad & -\mathcal{F}^*(Z) + c = 0, \\ & Z \succeq 0. \end{aligned} \tag{2.17}$$

We remark that keeping the structure of the problem may be beneficial, since the dual of (2.8) is

$$\begin{aligned} \max_{Z} \quad & -\operatorname{Tr} Z \\ \text{s.t.} \quad & -ZA^T - AZ + I = 0 \\ & Z \succeq 0, \end{aligned} \tag{2.18}$$

where the equality constraint uniquely determines Z, assuming that there exist no two eigenvalues α_i and α_j of A such that $\alpha_i + \alpha_j = 0$ (Bartels and Stewart, 1972).

2.3.1 Duality Gap

Let us denote the value of the objective function of the dual problem (2.17) d and the value of the primal problem (2.7) p for *any* feasible dual and primal points. Let us also denote the optimal values of these problems d^* and p^*, respectively. Then the important inequality

$$d \leq d^* \leq p^* \leq p \tag{2.19}$$

holds.

The *duality gap* of an optimization problem is defined as $p - d$ and gives an upper bound of how far from the optimal value of the problem we are. We refer to $p^* - d^*$ as the *optimal duality gap*.

2.3.2 Strong and Weak Duality

An important property is that $d^* \leq p^*$. This holds for all optimization problems, not only convex problems. If $d^* < p^*$, we have what is called *weak duality*, i.e. the optimal value

to the dual problem gives us a lower bound of the primal problem. If $d^* = p^*$, we have *strong duality*. This implies that if we solve the dual problem, we have the optimal value of the primal problem too. We remark that convexity of the optimization problem is *not* sufficient for strong duality to hold.

One condition that is sufficient for strong duality to hold for convex optimization problems is *Slater's condition*. Slater's condition for our problem states that if there exist a point x such that

$$\begin{aligned} F_0 + \mathcal{F}(x) &= S \\ S &\succ 0, \end{aligned} \tag{2.20}$$

then strong duality holds.

2.4 Optimality Conditions

The Karush-Kuhn-Tucker conditions for problem (2.7) are

$$\begin{aligned} F_0 + \mathcal{F}(x) &= S \\ -\mathcal{F}^*(Z) + c &= 0 \\ ZS &= 0 \\ Z &\succeq 0 \\ S &\succeq 0. \end{aligned} \tag{2.21}$$

The solution of these conditions define the global optimum of problem (2.7), assuming e.g. that Slater's condition holds, and many algorithms for solving this problem can be interpreted as methods for iteratively solving these conditions. We remark that all these constraints are linear, and thus easy to cope with, except the constraint $ZS = 0$ which is nonlinear and complicate the solution procedure. A naive approach can be to use Newtons method and linearize the nonlinear constraint. Unfortunately, this will only allow for very short steps to be taken, which will make the solution time very long.

A remedy to this is to relax the constraint $ZS = 0$ to $ZS = \tau I$, $\tau \geq 0$ which will allow for longer steps and faster convergence. Points satisfying (2.21) with the relaxed nonlinear constraint are said to be on the *central path*. We remark that as τ tends to zero, the central path converges to a solution to the KKT conditions. Algorithms that find the optimum by finding points on the central path are called path following algorithms.

An important concept is the *duality measure* ν, which is defined as

$$\nu = \frac{\langle Z, S \rangle}{n}, \tag{2.22}$$

where the duality measure ν is the duality gap for feasible Z and S. We remark that for points on the central path we have

$$\nu = \frac{\langle Z, S \rangle}{n} = \frac{\operatorname{Tr} \tau I}{n} = \frac{\tau n}{n} = \tau. \tag{2.23}$$

2.5 Algorithm

This section will describe a simple algorithm for semidefinite programming. The algorithm is a so called *interior-point* method. First, we state the algorithm, then the rest of the section will explain the different steps involved.

Algorithm 2.1 Semidefinite programming algorithm

1: Preprocess
2: Form system of equations to solve for search directions
3: Solve for search directions $Hx = b$
4: Determine step size
5: **if** $\frac{\langle Z,S\rangle}{n} >$ tol **then**
6: Goto 2
7: **end if**
8: Postprocess

2.5.1 Linearization of the KKT Conditions

This section will describe how the equations for the search directions described in Algortihm 2.1 are found. Let us start by linearizing the KKT conditions (2.21) with the relaxed version of the condition $ZS = \tau I$. That yields, replacing x with $x + \Delta x$, S with $S + \Delta S$ and Z with $Z + \Delta Z$,

$$\begin{aligned} F_0 + \mathcal{F}(x + \Delta x) &= S + \Delta S \\ -\mathcal{F}^*(Z + \Delta Z) + c &= 0 \\ \Delta Z \Delta S + \Delta Z S + Z \Delta S + ZS &= \tau I \\ Z + \Delta Z &\succeq 0 \\ S + \Delta S &\succeq 0. \end{aligned} \tag{2.24}$$

Now, let us collect all the Δ-terms on one side, and let us also ignore the nonlinear term $\Delta Z \Delta S$, yielding

$$\begin{aligned} \mathcal{F}(\Delta x) - \Delta S &= S - F_0 - \mathcal{F}(x) \\ \mathcal{F}^*(\Delta Z) &= -\mathcal{F}^*(Z) + c \\ \Delta Z S + Z \Delta S &= \tau I - ZS \\ \Delta Z &\succeq -Z \\ \Delta S &\succeq -S. \end{aligned} \tag{2.25}$$

Unfortunately, a symmetric solution to this system of equations are not guaranteed to exist (Kojima et al., 1997). A remedy to this is to introduce a symmetry transformation

$$\mathcal{H}(X) = \frac{1}{2}\left(R^{-1} X R + \left(R^{-1} X R\right)^T\right). \tag{2.26}$$

In (Zhang, 1998), it is shown that

$$ZS = \tau I \iff \mathcal{H}(ZX) = \tau I. \tag{2.27}$$

Hence, we replace the last equality constraint with the symmetrized one, i.e.

$$\mathcal{H}(\Delta Z S + Z \Delta S) = \tau I - \mathcal{H}(ZS). \tag{2.28}$$

Let us now summarize the equations we have, namely

$$\begin{aligned} \mathcal{F}(\Delta x) - \Delta S &= D_1 \\ \mathcal{F}^*(\Delta Z) &= D_2 \\ \mathcal{H}(\Delta Z S + Z \Delta S) &= D_3, \end{aligned} \tag{2.29}$$

for suitable D_1, D_2 and D_3. In order for us to simplify this, we symmetric vectorize the equations (2.29), and thus get a system of equations like

$$\begin{bmatrix} A & -I & 0 \\ 0 & 0 & A^T \\ 0 & F & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \mathbf{svec}(\Delta S) \\ \mathbf{svec}(\Delta Z) \end{bmatrix} = \begin{bmatrix} \mathbf{svec}(D_1) \\ \mathbf{svec}(D_2) \\ \mathbf{svec}(D_3) \end{bmatrix}, \tag{2.30}$$

where the operator $\mathbf{svec}(X)$ is defined in Definition 2.2, and where $A \in \mathbf{R}^{n(n+1)m/2}$ is the vectorization of the operator $\mathcal{F}(x)$, i.e.

$$A\Delta x = \mathbf{svec}(\mathcal{F}(\Delta x)). \tag{2.31}$$

Furthermore, we have that $F = R^{-1}Z \otimes_s R$ and $E = R^{-1} \otimes_s SR$ are the vectorizations of the symmetric operator $\mathcal{H}(X)$. Using block elimination techniques, we can eliminate ΔS and ΔZ get the system of equations

$$-A^T E^{-1} F A \Delta x = D_4, \tag{2.32}$$

for a suitable choice of D_4. Obviously, one needs to specify what the symmetry transformation matrices R are. Different choices of R give different search directions. We remark that there are several possible search directions, for example the AHO direction (Alizadeh et al., 1998), the HKM direction (Helmberg et al., 1996, Kojima et al., 1997, Monteiro, 1997) and the Nestrov-Todd direction (Todd et al., 1998).

In the HKM-direction, the scaling matrix R is equal to $S^{\frac{1}{2}}$. This simplifies the expression, such that we can write the equation system (2.32) as $M\Delta x = D_4$ where each element of M can be written as

$$M_{ij} = \left\langle F_i, Z F_j S^{-1} \right\rangle. \tag{2.33}$$

2.5.2 Computation of steplength

Once the search direction has been computed, the steplength needs to be decided. Different approaches exist, such as exact line search and backtracking (Boyd and Vandenberghe, 2004). However, we present a third idea which utilizes the fact that the semidefinite constraints on S and Z can be computed, as described in the manual to SPDT3 (Tütüncü et al., 2003).

Let us denote the current iterate S^i and the next iterate S^{i+1}, and we have found the search direction ΔS^i. Denote the steplength α. We require S^{i+1} to be positive semidefinite, that is

$$S^{i+1} = S^i + \alpha \Delta S^i \succeq 0. \tag{2.34}$$

It is not hard to show that the maximum steplength α we can choose is

$$\alpha_{\max} = \begin{cases} \frac{-1}{\lambda_{\min}((S^i)^{-1}\Delta S^i)} & \text{if } \lambda_{\min} \text{ is negative,} \\ \infty & \text{otherwise.} \end{cases} \tag{2.35}$$

Here we let $\lambda_{\min}((S^i)^{-1}\Delta S^i)$ denote the minimal eigenvalue of $(S^i)^{-1}\Delta S^i$. Invertibility of S^i is guaranteed if S^i is strictly feasible, and hence positive definite. The stepsize can the be chosen as

$$\alpha = \min(1, \gamma\alpha_{\max}), \tag{2.36}$$

where γ is chosen to be a number slightly less than 1, say $\gamma = 0.98$. This steplength α is then used to update x^i and S^i. A similar idea can be used to determine the steplength β that is used to update Z^i. Note here that it is not necessary to update the primal and dual variables with the same steplength.

2.5.3 Pre- and postprocessing

The pre- and postprocessing steps can consist of different parts depending on the algorithm, and will not be described in detail. We mention that preprocessing can for example be detection of linear constraints and detecting that constraints are linearly dependent. Also, choosing a starting point for the algorithm is included in the preprocessing.

3

Lagrangian Relaxation

Even though there exist efficient implementations of many classes of optimization problems, sometimes the optimization problems might be too large to solve, or the structure of the problem allows for a faster solution time. This chapter discusses some methods for doing this when the variables and constraints are such that if one, or possible several, constraints contains the variables in a way such that if that constraint would not have been present, the problem would decompose into several smaller problems. These constraints are in the literature called *complicating constraints*.

3.1 Introduction

A *decomposition method* is a method that solves an optimization problem, not by iterating over the whole problem, but rather solving *sub-problems* instead. These ideas were pioneered in the 1960s for linear systems by Dantzig and Wolfe (Dantzig and Wolfe, 1960) and Benders (Benders, 1962).

3.2 Complicating Constraints

This section will describe what a complicating constraint is. In loose terms, it is a constraint such that without it, the solution of the optimization problem would have been much simpler to obtain. We will focus on the following optimization problem with complicating constraints.

$$
\begin{aligned}
\min_{x} \quad & \sum_{j=0}^{N} c_j^T x^j \\
\text{s.t.} \quad & F_{00} + \sum_{j=0}^{N} \sum_{i=1}^{m_j} F_{0ij} x_i^j = S_0 && \text{(3.1a)} \\
& F_{j0} + \sum_{i=0}^{m_j} F_{kij} x_i^j = S_k, \quad k = 1, \ldots, N && \text{(3.1b)} \\
& S_k \succeq 0, \quad k = 0, \ldots, N.
\end{aligned}
$$

In (3.1), the problem would have decomposed into N different optimization problems, if the constraint (3.1a) had not been present. Various methods for overcoming this exists. Most of them involve decomposition using the Lagrangian. In this chapter we will present what is known as *Lagrangian decomposition*, but first we will show an abstraction to (3.1), in order to streamline the presentation. Consider the following optimization problem.

$$
\begin{aligned}
\min_{X} \quad & \sum_{i=0}^{N} \langle f_i, X_i \rangle \\
\text{s.t.} \quad & g(X) = g_c + \sum_{i=0}^{N} g_i(X_i) \preceq 0 && \text{(3.2a)} \\
& h_i(X_i) \preceq 0, \quad i = 1, \ldots, N && \text{(3.2b)}
\end{aligned}
$$

where $X = (X_0, \ldots, X_N)$ and where $g_i(X_i)$ are linear matrix-valued functions and $h_i(X_i)$ are affine matrix-valued functions. Let us also define the sets

$$
S_i = \{X_i : h_i(X_i) \preceq 0\}. \tag{3.3}
$$

We see that the optimization problems in (3.1) are a subset of the problems in (3.2). From now on, we will discuss the optimization problem (3.2), but it is clear that everything we say about it will also hold for the optimization problem (3.1).

3.3 Lagrangian with Respect to some Constraints

In this section, we will take the Lagrangian of problem (3.2) with respect to the complicating constraint (3.2a) and let the domain of X_i be S_i, as defined in (3.3). Then, following (2.10), the Lagrange dual function to (3.2) is

$$
g(Z) = \inf_{X_0, X_i \in S_i} L(X, Z) = \inf_{X_0, X_i \in S_i} \sum_{i=0}^{N} \langle f_i, X_i \rangle + \left\langle g_c + \sum_{i=0}^{N} g_i(X_i), Z \right\rangle. \tag{3.4}
$$

For a fixed value of Z, the minimization of this can be formulated as

$$g(Z) = \inf_{X_0, X_i \in S_i} L(X, Z) = \inf_{X_0, X_i \in S_i} \left\{ \langle X_0, f_0 + g_0^*(Z) \rangle + \langle Z, g_c \rangle + \sum_{i=1}^{N} \langle f_i, X_i \rangle + \langle g_i^*(Z), X_i \rangle \right\}. \quad (3.5)$$

If we look at (3.5), we see that for fixed Z, the minimization of $L(X, Z)$ is *separable* in X_i. That is, we can do the minimization for each X_i independent of $X_j,\ j \neq i$. This fact is used in many decomposition algorithms.

From (3.5), we can also deduce that we have no constraints on X_0. Since $L(X, Z)$ linear in X_0, we have that

$$f_0 + g_0^*(Z) = 0, \quad (3.6)$$

is a necessary constraint in order for this minimum to be bounded. This is one of the constraints of the dual problem (3.2). It is also possible that the minimization over $X_i \in S_i$ will not be bounded, but this will be discussed more in detail later on.

Keep in mind that the optimization problem that we are really interested in solving is

$$\max_{Z \succeq 0} \min_{X_0, X_i \in \mathbf{S}^n} L(X, Z). \quad (3.7)$$

We remark here that there exist several techniques to tackle this problem. A good starting point is (Lemaréchal, 2007). We will now present some of the methods that are available in the literature in order for us to tackle the relaxed problem.

3.4 Subgradient method of Uzawa

Perhaps the first method that was suggested for this problem was the subgradient method from 1958 by Uzawa in (Arrow et al., 1958). First, let us denote the partial derivatives of $L(X, Z)$ with respect to X and Z by $L_X(X, Z)$ and $L_Z(X, Z)$ respectively. Then, if we allow Z and X to be updated according to

$$\begin{aligned} X_{k+1} &= \mathrm{Proj}\, X_k + \tau L_X(X_k, Z_k) \\ Z_{k+1} &= \mathrm{Proj}\, Z_k - \tau L_Z(X_k, Z_k), \end{aligned} \quad (3.8)$$

where Proj means a projection such that X_{k+1} and Z_{k+1} are their respective domains. If certain conditions on $L(X, Z)$ holds, then the system of equations will converge to the optimal values of X and Z. We remark that the original algorithm was developed for the case where the domain of X and Z was the positive real numbers, which made the projection easy. We remark that if we want to use this method for the problem in (3.4), we need to ensure that the constraint (3.6) holds.

In order for (3.8) to converge, the function $L(X, Z)$ must be *strictly* convex in X which rules out systems with linear dependence on X. Also, the stepsize $\tau > 0$ must satisfy certain technical conditions, i.e. it must be chosen "small" enough.

There exist various tricks in order for this algorithm to converge even when $L(X, Z)$ is not strictly convex in X, such as adding a term $\varepsilon \langle X, X \rangle$ or making small modifications

to the constraints. In the original book (Arrow et al., 1958), Arrow and Hurwitz suggest that the constraints should be transformed with a strictly increasing function $\rho(z)$ such that $\rho(0) = 0$. They suggest to use $\rho(z) = 1 - e^{-\nu}$ with $\nu > 0$. We remark that the original work was done for the case when the constraints where not semidefinite.

3.5 Kelley-Cheney-Goldstein

Another approach to solving this problem is by a method of Kelley (Kelley, 1960) and Cheney and Goldstein (Cheney and Goldstein, 1959). The idea is based on the fact that for every feasible value of Z, say Z^k, and corresponding minimizing X_i^k to (3.4), it holds that

$$g(Z^k)+\left\langle Z - Z^k, g_c + \sum_{i=1}^{N} g_i(X_i^k) \right\rangle = \sum_{i=1}^{N} \langle f_i, X_i^k \rangle + \left\langle Z, g_c + \sum_{i=1}^{N} g_i(X_i^k) \right\rangle, \quad (3.9)$$

is a *linear supporting function* to $h(Z)$ at Z^k. A linear supporting function to $h(Z)$ at Z^k is a linear function which never lies below $h(Z)$ and contacts it at Z^k. Hence, assuming we have r values Z^k and X_i^k, the function

$$v^r = \min_{1\leq k\leq r} \sum_{i=1}^{N} \langle f_i, X_i^k \rangle + \left\langle Z, g_c + \sum_{i=1}^{N} g_i(X_i^k) \right\rangle, \quad (3.10)$$

is an approximation to $h(Z)$. Hence, instead of maximizing $h(Z)$ directly, the Kelley-Cheney-Goldstein algorithm iterates between doing the minimizing with respect to X_i in (3.4) and maximizing (3.10) with respect to Z. We remark that the maximization of (3.10) is done by using the epigraph formulation, i.e. by solving the optimization problem

$$\begin{aligned} \max_{\sigma,Z} \quad & \sigma \\ \text{s.t.} \quad & \sum_{i=1}^{N} \langle f_i, X_i^k \rangle + \left\langle Z, g_c + \sum_{i=1}^{N} g_i(X_i^k) \right\rangle \geq \sigma \\ & f_0 + g_0^*(Z) = 0 \\ & Z \succeq 0. \end{aligned} \quad (3.11)$$

The convergence of this method is in some cases very slow (Hiriart-Urruty and Lemaréchal, 1993).

3.6 Bundle methods

The bundle method is similar to the Kelley-Cheney-Goldstein method, but differs in that an extra term is added in the objective function in order to penalize too big steps in Z. The method basically works like this. Let us define $\hat{Z}$ to be the previous value of Z that has

given us the lowest value of the dual function. The optimization problem in the bundle method is then

$$\begin{array}{ll} \max\limits_{\sigma,Z} & \sigma - \dfrac{\mu_k}{2}\left\langle Z-\hat{Z}, Z-\hat{Z}\right\rangle \\ \text{s.t.} & \displaystyle\sum_{i=1}^{N}\left\langle f_i, X_i^k\right\rangle + \left\langle Z, g_c + \sum_{i=1}^{N} g_i(X_i^k)\right\rangle \geq \sigma \\ & f_0 + g_0^*(Z) = 0 \\ & Z \succeq 0. \end{array} \tag{3.12}$$

That is, the method penalizes deviations from the previously best value of Z. If we get a better value of Z than $\hat{Z}$, that is, a value of Z that gives a higher value of $g(Z)$, we set $\hat{Z} = Z$.

4

Semidefinite Programming in Systems and Control Theory

In this chapter, some applications of semidefinite programs to control theory related problems will be reviewed. An early reference, which contains a large number of collected applications is (Boyd et al., 1994).

4.1 System considered

The system we consider in this chapter is an LTI system on state-space form. The system is

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bw(t) \\ z(t) &= Cx(t) + Dw(t), \end{aligned} \tag{4.1}$$

where $x(t) \in \mathbf{R}^{n_x}$, $w(t) \in \mathbf{R}^{n_w}$, $z(t) \in \mathbf{R}^{n_z}$ and all matrices have suitable dimensions for the multiplications to be defined. We assume that (A, B, C, D) is a minimal realization.

We remark that methods similar to the ones we demonstrate in this chapter exist for other types of models, such as linear fractional transformation (LFT), models with polytopic uncertainty and so on. Also, everything in this chapter is about continuous time systems, but similar methods exist for discrete time systems.

4.2 Stability

The internal stability of system (4.1) is equivalent to the existence of $P \in \mathbf{S}^n$ such that

$$\begin{aligned} A^T P + PA &\prec 0 \\ P &\succ 0. \end{aligned} \tag{4.2}$$

This was shown in the 1890s by Lyapunov for an autonomous system (Boyd et al., 1994). We remark that the stability of (4.1) can also be checked by computing the eigenvalues of A.

4.3 H_∞-norm

The H_∞-norm of system (4.1) can be computed by solving the optimization problem

$$\begin{aligned} \min_{\gamma^2, P} \quad & \gamma^2 \\ \text{s.t.} \quad & \begin{bmatrix} A^T P + PA + C^T C & PB + C^T D \\ B^T P + D^T C & D^T D - \gamma^2 I \end{bmatrix} \preceq 0 \\ & P \succeq 0, \end{aligned} \tag{4.3}$$

where $P \in \mathbf{S}^{n_x}$ and $\gamma^2 \geq 0$ by construction. Assuming $\gamma > 0$, this can, by means of the Schur complement, be reformulated as

$$\begin{aligned} \min_{\gamma, \tilde{P}} \quad & \gamma \\ \text{s.t.} \quad & \begin{bmatrix} A^T \tilde{P} + \tilde{P} A & \tilde{P} B & C^T \\ B^T \tilde{P} & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} \preceq 0 \\ & \tilde{P} \succeq 0, \end{aligned} \tag{4.4}$$

where $\tilde{P} = \frac{P}{\gamma}$. Whichever formulation that is preferable depends on the situation. In (4.4), the size of the basis matrices in (2.7) will be bigger than if (4.3) was used, but if a more general problem was to be solved, the fact that the products $C^T C$, $C^T D$, $D^T C$ and $D^T D$ don't appear in (4.4) might be advantageous. We remark that the H_∞-norm of (4.1) can be found in other ways, for example by using a bisection algorithm or checking the eigenvalues of a certain Hamiltonian matrix (Zhou et al., 1996, p. 115-116).

4.4 H_2-norm

For the same system as in (4.1) with $D = 0$, the H_2-norm can be found be solving the following SDP.

$$\begin{aligned} \min_{X} \quad & \operatorname{Tr} X \\ \text{s.t.} \quad & \begin{bmatrix} X & C \\ C^T & W^{-1} \end{bmatrix} \succeq 0 \\ & AW + WA^T + BB^T = 0, \end{aligned} \tag{4.5}$$

where W is the controllability Gramian of (4.1). We remark that one would solve the Lyapunov equation in (4.5) first, and then solve the semidefinite program. After solving the optimization problem, the squared H_2-norm is then equal to $\operatorname{Tr} X$. We also remark that the H_2-norm can be directly computed from $\sqrt{CWC^T}$ (Zhou et al., 1996, p. 113).

Bibliography

F. Alizadeh, J.P.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.

K.J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and non-linear programming*. Stanford University Press, 1958.

R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$ [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

J.F. Benders. Partitioning methods for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, 1994. ISBN 0-89871-334-X.

S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

E.W. Cheney and A.A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1(1):253–268, 1959.

G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

R. Falkeborn and A. Hansson. A decomposition algorithm for KYP-SDPs. In *Proceedings of the European Control Conference*, pages 3202–3207, August 2009.

R. Falkeborn, J. Löfberg, and A. Hansson. Lowrank exploitation in semidefinite programming for control. Technical Report LiTH-ISY-R-2927, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, January 2010.

C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. Springer, 1993.

J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8(4):703–712, 1960.

M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7:86, 1997.

C. Lemaréchal. The omnipresence of Lagrange. *Annals of Operations Research*, 153(1): 9–27, 2007.

Z. Liu and L. Vandenberghe. Low-rank structure in semidefinite programs derived from the KYP lemma. In *Proceedings of the 46th IEEE Conference on Decision and Control*. Citeseer, 2007.

J. Löfberg. YALMIP: a toolbox for modeling and optimization in MATLAB. *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289, 2004.

R.D.C. Monteiro. Primal–dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization*, 7(3):663–678, 1997.

C. Papageorgiou, R. Falkeborn, and A. Hansson. Formulation of the stability margins clearance criterion as a convex optimization problem. In *Proceedings of the 6th IFAC Symposium on Robust Control Design, 2009*, pages 325–330, June 2009.

M.J. Todd, K.C. Toh, and R.H. Tutuncu. On the Nesterov-Todd direction in semidefinite programming. *SIAM Journal on Optimization*, 8(3):769–796, 1998.

R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95(2):189–217, 2003.

L. Vandenberghe, V.R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. Interior-point algorithms for semidefinite programming problems derived from the KYP lemma. *Positive Polynomials in Control, Lectures Notes in Control and Information Science. Springer*, 2004.

R. Wallin, C.-Y. Kao, and A. Hansson. A cutting plane method for solving KYP-SDPs. *Automatica*, 44(2):418 – 429, 2008. ISSN 0005-1098.

R. Wallin, A. Hansson, and J. H. Johansson. A structure exploiting preprocessor for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. *IEEE Transactions on Automatic Control*, 54(4):697–704, April 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2014922.

H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, 2000.

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.

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.

Part II

Publications

Paper A

A Decomposition Algorithm for KYP-SDPs

Authors: Rikard Falkeborn and Anders Hansson

Published as

R. Falkeborn and A. Hansson. A decomposition algorithm for KYP-SDPs. In *Proceedings of the European Control Conference*, pages 3202–3207, August 2009

A Decomposition Algorithm for KYP-SDPs

Rikard Falkeborn and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden

Abstract

In this paper, a structure exploiting algorithm for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma, where some of the constraints appear as complicating constraints is presented. A decomposition algorithm is proposed, where the structure of the problem can be utilized. In a numerical example, where a controller that minimizes the sum of the H_2-norm and the H_∞-norm is designed, the algorithm is shown to be faster than SeDuMi and the special purpose solver KYPD.

1 Introduction

Semidefinite programs (SDPs) arise in many applications in control and signal processing (Boyd et al., 1994). In many cases, the programs that need to be solved involve large matrix variables that can make the computational burden very large. In some cases, the structure of the problem can be utilized to reduce the computational demands.

SDPs derived from the Kalman-Yakubovich-Popov lemma (KYP-SDPs) (Rantzer, 1996) is one such example, where tailor-made solvers have been successfully developed, see for example (Wallin et al., 2009, 2008, Kao et al., 2004). In this paper an algorithm for solving KYP-SDPs where some of the constraints appear as *complicating constraints*, i.e. constraints which are such that the optimization program would have been much easier to solve, if they were not present. More specifically, we treat optimization programs with the following structure

$$\begin{aligned} \min_{x_i,P_i} \ & \sum_{i=0}^{N} \langle C_i, P_i \rangle + c_i^T x_i \\ \text{s.t. } & \mathcal{F}_0(P_0) + M_0 + \mathcal{G}(x) \preceq 0 \\ & \mathcal{F}_i(P_i) + M_{i_0} + \mathcal{H}_i(x_i) \preceq 0, \quad i = 1, \dots, N, \end{aligned} \tag{1}$$

where $C_i, P_i \in \mathbf{S}^{n_i}$, and where $\mathbf{S}^n$ is the space of symmetric matrices of dimension $n \times n$. The inner product $\langle C_i, P_i \rangle$ is defined as $\operatorname{Tr} C_i P_i$. We define the operators

$$\mathcal{F}_i(P_i) = \begin{bmatrix} A_i^T P_i + P_i A_i & P_i B_i \\ B_i^T P_i & 0 \end{bmatrix}, i = 0, \ldots, N,$$

where $A_i \in \mathbf{R}^{n_i \times n_i}$, $B_i \in \mathbf{R}^{n_i \times m_i}$,

$$\mathcal{H}_i(x_i) = \sum_{j=1}^{p_i} M_{i_j} x_{i_j}, i = 1, \ldots, N,$$

where $M_{i_j} \in \mathbf{S}^{n_i+m_i}$, $x_i \in \mathbf{R}^{p_i}$, and where x_{i_j} denotes the jth component of the vector x_i. Let us also define the operators $\mathcal{G}_i(x_i) = \sum_{j=1}^{p_i} x_{i_j} M_{0_{i_j}}$, and $\mathcal{G}(x) = \sum_{i=0}^{N} \mathcal{G}_i(x_i)$ where $x = (x_0, x_1, \ldots, x_N)$.

We assume, that the pairs (A_i, B_i) are controllable. This implies that the operators $\mathcal{F}_i(P_i)$ have full rank. This can be relaxed to stabilizability of the pair (A_i, B_i) provided that the range of C_i is in the controllable subspace of (A_i, B_i), see (Wallin et al., 2009) for details. We also assume the optimal value of (1) exists and is finite.

The constraint involving $\mathcal{G}(x)$ is a complicating constraint, since without it, the problem would decompose into several smaller problems, which all could be solved separately. Hence, a decomposition algorithm would be suitable to use. We remark that the algorithm we propose can easily be generalized to have several complicating constraints.

This type of programs appear for example in robust control analysis using integral quadratic constraints (Megretski and Rantzer, 1997), and linear system design and analysis (Hindi et al., 1998, Oishi and Balakrishnan, 1999).

We remark that a standard linear matrix inequality (LMI)

$$M = M_0 + \sum_{k=1}^{n} x_k M_k \preceq 0,$$

is a special case of a KYP-LMI with the size of A being 0×0. Hence we can handle a mixture of KYP constraints and standard LMIs. In fact, as we will see in Section 3, in some cases, where the complicating constraint is a regular LMI, the ability to solve the regular LMI using a standard solver and use tailormade solvers for the KYP-constraints can, as we will see, reduce the computational time.

1.1 Eliminating variables from KYP-SDPs

We here show how the structure of the KYP-SDP can be utilized to eliminate dual variables and thus formulate a smaller problem which can be solved in less time. The details can be found in (Wallin et al., 2009). For simplicity, we only show how the elimination is done for the case with one KYP-constraint, but a generalization is straightforward. Consider the problem

$$\begin{aligned} \min_{x,P} \ & \langle C, P \rangle + c^T x \\ \text{s.t. } & \mathcal{F}(P) + M_0 + \mathcal{G}(x) \preceq 0. \end{aligned} \tag{2}$$

The dual formulation of this problem is (Wallin et al., 2009)

$$\begin{aligned} \max_{Z} \ & \langle M_0, Z \rangle \\ \text{s.t. } & \mathcal{F}^*(Z) + C = 0 \\ & \mathcal{G}^*(Z) + c = 0 \\ & Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{12}^T & Z_{22} \end{bmatrix} \succeq 0 \end{aligned} \tag{3}$$

where the adjoint operators $\mathcal{F}^*(Z)$ and $\mathcal{G}^*(Z)$ are defined as

$$\begin{aligned} \mathcal{F}^*(Z) &= AZ_{11} + Z_{11}A^T + BZ_{12}^T + Z_{12}B^T \\ \mathcal{G}^*(Z) &= \begin{bmatrix} \langle M_1, Z \rangle \\ \vdots \\ \langle M_p, Z \rangle \end{bmatrix}. \end{aligned} \tag{4}$$

By computing a basis for the nullspace of the adjoint operator $\mathcal{F}^*(Z)$, it is possible to reduce the number of variables. In (Wallin et al., 2009) it is shown how such a basis can be easily found by solving Lyapunov equations which can be done in an efficient and numerically stable way (Bartels and Stewart, 1972).

When the reduction of variables is done, the reduced dual problem can be solved, and when a solution is found, the original optimal variables can be computed, see (Wallin et al., 2009) for details.

2 Lagrangian relaxation

Optimization programs with complicating constraints have been extensively studied within the field of optimization and operations research and is usually tackled with different decomposition algorithms. We will employ one of these, *Lagrangian relaxation* (Lemaréchal, 2001) pioneered in (Kelley, 1960, Cheney and Goldstein, 1959), and show how the specific structure of (1) can be used to speed up the computational time.

2.1 Forming the Lagrangian decomposes the problem

In order to make the presentation more streamlined, we derive the algorithm in a slightly more general form than (1). Consider the problem

$$\min_{X} \sum_{i=0}^{N} \langle f_i, X_i \rangle$$

$$\text{s.t. } g(X) = g_c + \sum_{i=0}^{N} g_i(X_i) \preceq 0 \tag{5a}$$

$$h_i(X_i) \preceq 0, \quad i = 1, \ldots, N, \tag{5b}$$

where g_i and h_i are assumed to be symmetric linear matrix functions of X, and where f_i is assumed to be a symmetric matrix. We let $X_i = (P_i, x_i)$ and $X = (X_0, \ldots, X_N)$.

The Lagrangian function to (5) with respect to the complicating constraint (5a) is (Boyd and Vandenberghe, 2004)

$$L(X,Z) = \sum_{i=0}^{N} \langle f_i, X_i \rangle + \left\langle Z, g_c + \sum_{i=0}^{N} g_i(X_i) \right\rangle. \tag{6}$$

Hence, if we let $S_i = \{X_i : h_i(X_i) \preceq 0\}$, the dual function to (5) is

$$h(Z) = \min_{X_0, X_i \in S_i} L(X,Z). \tag{7}$$

For fixed Z, the minimization can be formulated as

$$\min_{X_0, X_i \in S_i} \left\{ \langle X_0, f_0 + g_0^*(Z) \rangle + \langle Z, g_c \rangle + \sum_{i=1}^{N} \langle f_i, X_i \rangle + \langle g_i^*(Z), X_i \rangle \right\}. \tag{8}$$

where $g_i^*(Z)$ denotes the dual of $g_i(X_i)$. In order for the minimal value to be bounded from below when minimizing $L(X,Z)$ with respect to X_0, we have to require that Z fulfills the constraint

$$g_0^*(Z) + f_0 = 0. \tag{9}$$

For the problem studied in this paper, this corresponds to

$$\begin{aligned} \mathcal{F}_0^*(Z) + C_0 &= 0 \\ \mathcal{G}_0^*(Z) + c_0 &= 0, \end{aligned} \tag{10}$$

where the adjoint operators $\mathcal{F}_0^*(Z)$ and $\mathcal{G}_0^*(Z)$ are defined as in (4). After minimizing with respect to X_0, we obtain, since we have required Z to fulfill the constraint (9),

$$h(Z) = \min_{X_0, X_i \in S_i} L(X_0, \ldots, X_N, Z) = \sum_{i=1}^{N} \min_{X_i \in S_i} \langle f_i + g_i^*(Z), X_i \rangle. \tag{11}$$

Hence, for fixed Z, the problem of minimizing the Lagrangian under the constraints (5b) is a separable problem in $X_i = (P_i, x_i)$ for $i = 1, \ldots, N$. In our problem, each minimization is equal to

$$\begin{aligned} &\min_{x_i, P_i} \langle C_i, P_i \rangle + \bar{c}_i^T x_i \\ &\text{s.t. } \mathcal{F}_i(P_i) + M_{i_0} + \mathcal{H}_i(x_i) \preceq 0, \end{aligned} \tag{12}$$

where $\bar{c} = c_i + \mathcal{G}_i^*(Z)$.

We remark that $h(Z)$ is less than or equal to $\sum_{i=0}^{N} \langle f_i, Z \rangle$ for $Z \succeq 0$ which fulfills (9).

One should not solve the separable optimization programs as they stand, but use a tailor-made solver for KYP-SDPs. We take the same approach as in (Wallin et al., 2009) and eliminate variables in the dual problem as described in Section 1.1. Note that the reduction of the dual variables for the ith subproblem can be reused if we need to solve the same subproblem but for a different Z. This will be used in Section 2.2.

The boundedness of the subproblems (12) can be an issue and is ensured by bounding the optimal value of (12), see Appendix. This is done in the numerical example we present in Section 3.

2.2 Updating Z

The dual function $h(Z)$ is a lower bound on the optimal value of (5). We know that if the problem is convex and Slaters condition (Boyd and Vandenberghe, 2004) holds, the maximum of the dual function is equal to the optimal objective value of the optimization problem. Hence, we want to maximize the dual function to get a good lower bound on the optimal value. That is, we want to solve the optimization problem

$$\max_Z h(Z) = \max_Z \min_X L(X, Z). \tag{13}$$

A problem is that we do not have an explicit expression for the dual function in (11) since it depends on X_i.

Dual formulation

To be able to compute a lower bound on the optimal objective function, (Lasdon, 1970) proposes a *tangential approximation* method, first outlined in (Geoffrion, 1970). We note that, if we have solved the Lagrangian problem r times for r different fixed Z^k satisfying (9), and then have r different solutions X_i^k, the functions

$$h(Z^k) + \left\langle Z - Z^k, g_c + \sum_{i=1}^N g_i(X_i^k) \right\rangle = \sum_{i=1}^N \left\langle f_i, X_i^k \right\rangle + \left\langle Z, g_c + \sum_{i=1}^N g_i(X_i^k) \right\rangle, \tag{14}$$

are *linear supporting functions* to $h(Z)$ at Z^k. A linear supporting function to $h(Z)$ at Z^k is a linear function which never lies below $h(Z)$ and contacts it at Z^k. We can therefore use the piecewise linear function

$$v^r(Z) = \min_{1 \le k \le r} \sum_{i=1}^N \left\langle f_i, X_i^k \right\rangle + \left\langle Z, g_c + \sum_{i=1}^N g_i(X_i^k) \right\rangle, \tag{15}$$

as an approximation to $h(Z)$. Instead of maximizing $h(Z)$, we maximise the approximation v^r. By using the epigraph formulation of (15), an equivalent problem is

$$\begin{aligned} \max_{\sigma, Z} \ & \sigma \\ \text{s.t.} \ & \sum_{i=1}^N \left\langle f_i, X_i^k \right\rangle + \left\langle Z, g_c + \sum_{i=1}^N g_i(X_i^k) \right\rangle \ge \sigma, \quad k = 1, \dots, r \\ & f_0^* + g_0^*(Z) = 0 \\ & Z \succeq 0. \end{aligned} \tag{16}$$

Since it is not certain that the optimal value of (16) is bounded, it is necessary to add the constraint

$$\sigma < \sigma_{\max}, \tag{17}$$

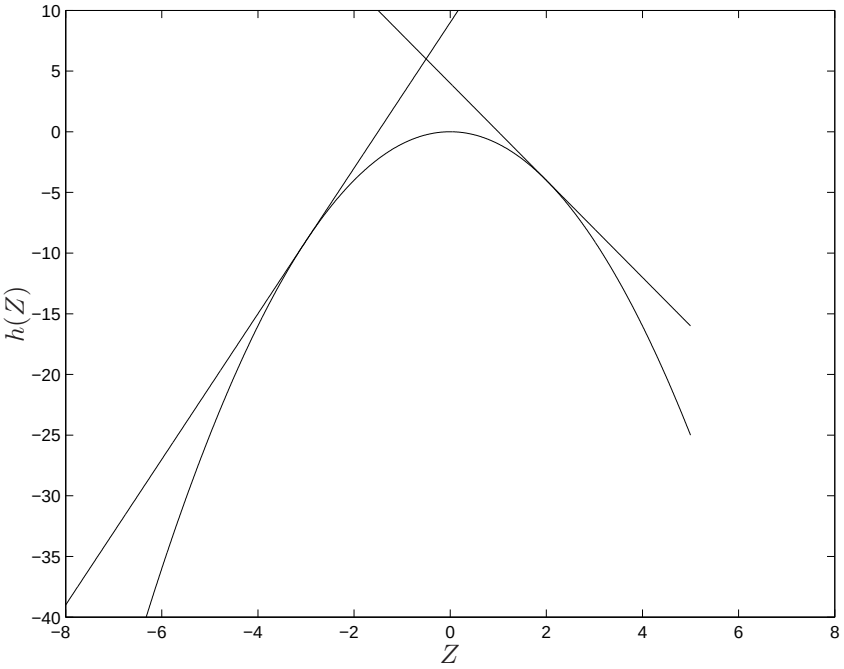


Figure 1: *Tangential approximation of* $h\,(Z)$.

in order to get a bounded optimal value. Here $\sigma_{\max}$ is chosen such that $\sigma_{\max}$ is larger than the optimal value of the original problem. We remark that this value is unknown, and one should pick a large value of $\sigma_{\max}$.

In Figure 1, tangential approximation of the concave function $h(Z)$ is shown.

If $h(Z^r) = L(X_1^r, \ldots, X_N^r)$, we know that we have found the optimum. If not, we can use the "new" Z^{r+1} to solve the subproblems again and obtain a new X_i^{r+1}. In practice, it is not possible to find the exact optimum due to numerics, and we have to settle for when $h(Z^r) - L(X_1^r, \ldots, X_N^r)$ is less than ϵ. This implies that the duality gap is ϵ or less.

An iterative procedure to solve the original problem (1) can be outlined as follows, starting from iteration r.

1. Solve the problem (16), and obtain an optimal solution Z^{r+1}.
2. Solve the subproblems (12) to obtain an optimal solution $\left(X_0^{r+1}, \ldots, X_N^{r+1}\right)$.
3. If $|h(Z^{r+1}) - L(X_0^{r+1}, \ldots, X_N^{r+1})| < \epsilon$, terminate.
4. Add a linear constraint like in (16) and return to step 1.

Here we can note that σ is a non-increasing function in iterations r, but no such guarantees can be given for L (Lasdon, 1970, p. 433).

Utilizing the structure of Z

Solving the problem (16) as it stands is not a good idea. Instead, it is possible to reformulate the optimization problem as an equivalent problem with fewer variables. We take same approach as described in Section 1.1. However, note that the computation of the basis can be reused in all iterations.

Primal formulation

It is also possible to formulate the dual of (16) and solve that problem instead. The dual problem of (16) is, note that P_i^k and x_i^k are fixed,

$$\begin{aligned}
&\min_{y,Q} c_0^T y_0 + \langle C_0, Q\rangle + \sum_{k=1}^{r}\sum_{i=1}^{N} \left(c_i^T x_i^k + \left\langle C_i, P_i^k \right\rangle\right) y_k \\
&\text{s.t. } \mathcal{F}_0(Q) + \mathcal{G}_0(y_0) + M_0 \underbrace{\sum_{k=1}^{r} y_k}_{=1} + \sum_{k=1}^{r}\sum_{i=1}^{N} \mathcal{G}_i(x_i^k) y_k \preceq 0 \\
&\quad \sum_{k=1}^{r} y_k = 1 \\
&\quad y_k \geq 0, \quad k = 1, \ldots, r,
\end{aligned} \tag{18}$$

where $y_0 \in \mathbf{R}^{p_0}$, $Q \in \mathbf{S}^{n_0}$, and where $y_i \in \mathbf{R}, i = 1, \ldots, r$. The dual variable Z needed in the next iteration can be returned by primal-dual solvers. In order to avoid the equality constraint we can eliminate the "last" variable y_r and replace it with $1 - \sum_{k=1}^{r-1} y_k$. Thus, tailor-made solvers for KYP-problems (Wallin et al., 2009, 2008) can be used to solve the problem.

Note that if a feasible point for iteration r is known, then a feasible point for $r+1$ is also known (by letting $y_{r+1} = 0$). Hence, if the underlying solver uses a method which requires a strictly feasibly point, the first phase where such a point is found can be skipped. This might save some computational time.

In the first iterations of the algorithm, it is not certain that the LMI (18) is feasible since there are only a few x_i^k. This corresponds to the case where σ is unbounded from above in (16). A remedy to this is to include the constraint (17) when formulating the dual of (16). The dual is then

$$\begin{aligned}
&\min_{y,Q,w} \; w\sigma_{\max} + c_o^T y_0 + \langle C_0, Q\rangle + \sum_{k=1}^{r}\sum_{i=1}^{N} \left(c_i^T x_i^k + \left\langle C_i, P_i^k \right\rangle\right) y_k \\
&\text{s.t. } \mathcal{F}_0(Q) + \mathcal{G}_0(y_0) + M_0 \sum_{k=1}^{r} y_k + \sum_{k=1}^{r}\sum_{i=1}^{N} \mathcal{G}_i(x_i^k) y_k \preceq 0 \\
&\quad \sum_{k=1}^{r} y_k + w = 1 \\
&\quad y_k \geq 0, \quad k = 1, \ldots, r \\
&\quad w \geq 0,
\end{aligned} \tag{19}$$

which, if $w = 0$ is the same problem as (18). If $w = 0$ in the solution to (19), the problem (18) is feasible. For numerical reasons, it is better to switch to solving (18) when the optimal value of w is zero. We remark that for problems where n_0 is small compared to m_0, that is, when the number of columns in A_0 is small compared to the number of columns in B_0, it may be beneficial to solve the problem in (19) instead of its dual (16). We also remark that one should bound the optimal value of the problem in (19) in the same fashion as we show in the Appendix.

3 Numerical example

The efficiency of the algorithm is investigated in a numerical example borrowed from (Oishi and Balakrishnan, 1999), see also (Hindi et al., 1998, Farhoodi and Beheshti, 2007), and it is compared to the generic solver SeDuMi 1.21, (Sturm, 1999), and the tailor-made solver for KYP-problems, KYPD, version 1.2 (Wallin et al., 2009). All solvers were interfaced via YALMIP version 3, release 20090505, (Löfberg, 2004).

Using the Youla parametrization (Boyd and Barratt, 1991), the set of all stable closed-loop plants of a system can be written as

$$G_{cl} = T_1 + T_2 Q T_3, \tag{20}$$

where T_i are stable and depends on the system matrices and Q is an arbitrary stable function transfer matrix. The corresponding controller is then, assuming positive feedback

$$K = Q(I + GQ)^{-1}. \tag{21}$$

By restricting Q to lie in a finite-dimensional subspace in such a way that the parameters enter linearly, i.e. $Q = Q(\theta) = \sum_{i=1}^{n} Q_i \theta_i$, convex constraints on the closed loop system result in convex optimization problems where the optimum can be easily found by polynomial time methods (Boyd and Vandenberghe, 2004).

In order to formulate constraints on the closed loop system as LMIs, it is necessary to write (20) in state space form. It is also necessary that this state space realization has all θ_i in the C and D matrix in order for the constraints to be convex. The realization of (20) is typically obtained using system Kronecker products (Khargonekar and Rotea, 1991, Hindi et al., 1998) and yields a system of a much higher order than the original plant order. The resulting closed loop system matrices can then be written as A, B, $C(\theta)$ and $D(\theta)$ where $C(\theta)$ and $D(\theta)$ depends affinely on the parameters chosen in the parametrization of $Q(\theta)$.

Many design specifications can be cast as LMIs. We can mention for example specifications and constraints on the H_2-norm, H_∞-norm, dispassivity constraints and the location of the closed loop poles. Some of these are in the form of KYP-SPDs. As an example, we will solve the joint minimization of the H_2-norm and the H_∞-norm of a system (Hindi et al., 1998, Oishi and Balakrishnan, 1999). The design problem results in the optimization problem

$$\min_{\gamma^2, X, \theta, P} \operatorname{Tr} X + \gamma^2 \tag{22a}$$

$$\text{s.t.} \begin{bmatrix} X & C(\theta) W^{\frac{1}{2}} \\ W^{\frac{1}{2}} C(\theta)^T & I \end{bmatrix} \succeq 0 \tag{22b}$$

$$\begin{bmatrix} A^TP + PA & PB & 0 \\ B^TP & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & C(\theta)^T \\ 0 & -\gamma^2 I & D(\theta)^T \\ C(\theta) & D(\theta) & -I \end{bmatrix} \preceq 0, \tag{22c}$$

where X and P are symmetric positive definite matrices, A, B, $C(\theta)$ and $D(\theta)$ are the state space matrices of the closed loop system, W is the controllability gramian of the system, i.e. W solves

$$AW + WA^T + BB^T = 0. \tag{23}$$

This problem can be transformed to the form (1) where the complicating constraint is (22b) since without it, tailor-made solvers for KYP-problems that use the dual formulation could solve the problem. However, formulating the dual of the entire problem would create a very large matrix variable corresponding to the LMI (22b). Hence, using a decomposition algorithm that alternates between solving the KYP-SDP in it's dual form, after eliminating variables and solving the H_2-LMI (22b) in it's primal form with much fewer variables than the dual form will lower the computational burden.

To test the algorithm, we create random systems using `rss` in Matlab with one input and one output. We chose the Youla parameter $Q(\theta)$ to be

$$Q(\theta) = \sum_{i=0}^{n_Q} \frac{\theta_i}{(s+0.5)^i}, \tag{24}$$

as suggested in (Boyd and Barratt, 1991). We choose to have $n_Q = 10$ in all examples. The reason for letting $n_Q = 10$ is seen in Figure 2, where the normalized objective value is plotted for 10 different systems of state dimension 20 with increasing n_Q. The computations were done using SeDuMi, i.e. no decomposition was used. We stopped the increase in n_Q when the objective function was not improving more than 0.1% or if SeDuMi ran into numerical problems that were too severe.

We solve the resulting LMIs using our algorithm, the tailor-made solver KYPD calling SeDuMi and the generic solver SeDuMi. All computations were terminated when the absolute error or the relative error was less than 10^{-3}. We create 10 different systems for each n, where n is the number of states, and let n vary. The number n_Q was set to 10 for all computations. The computational times were averaged and the averaged times can be seen in Figure 3. We remark that in practice, n would include both the states of the original plant as well as states from various weighting filters that is commonly used in H_∞-synthesis. We also remark that the number n in Figure 3 is the number of states in the original plant, not the dimension of the A-matrix in (1). As an example, for $n = 20$, the A-matrix is has 90 rows and columns.

In Figure 3 we see that the decomposition algorithm in this case outperforms both the tailormade solver KYPD and the generic solver SeDuMi. It can also be seen that the tailormade solver for KYP-problems actually performs worse than the generic solver SeDuMi. This can be explained as follows. SeDuMi solves an optimization problem where the majority of variables come from P which is $n \times n$ and therefore yield $(n^2 + n)/2$ variables assuming the order of $Q(\theta)$ and the number of variables in X, which is determined by the number of outputs, can be neglected. KYPD formulates the dual problem and eliminates variables. For a constraint of the type

$$\mathcal{F}(P) + M_0 + \mathcal{G}(x) \preceq 0, \tag{25}$$

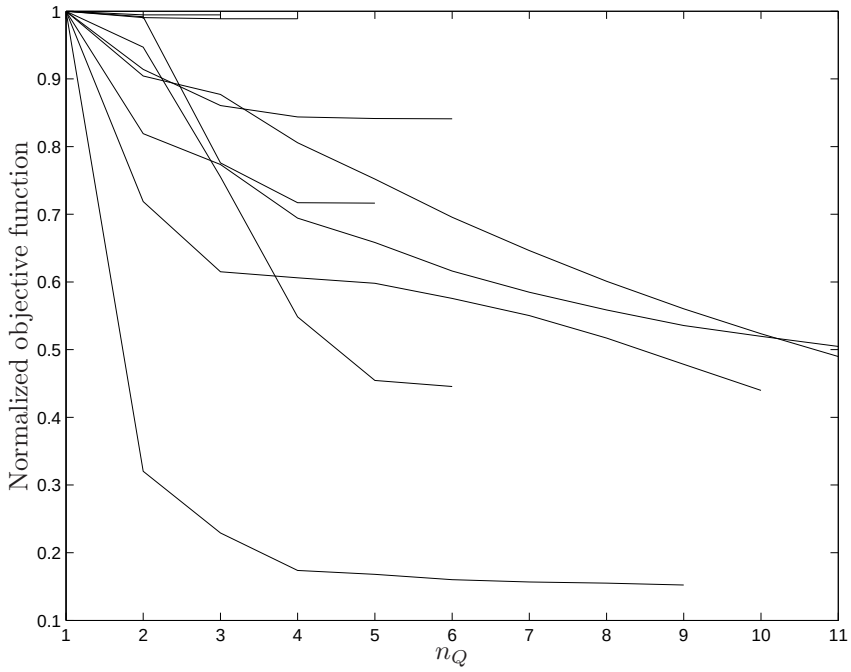


Figure 2: *Normalized objective values for different sizes of $Q(\theta)$.*

the number of remaining dual variables, after the reduction, is $mn + m\left(m+1\right)/2$ where m is the number of rows in the B matrix. This is usually a lot lower than $\left(n^2+n\right)/2$. However, for this numerical example, we also have the constraint (22b). When we formulate the dual problem of the numerical example, there will be a dual variable Z_2 corresponding the LMI (22b). This variable has no special structure that is used in KYPD, and no variables will be reduced, adding an extra matrix variable with $\left(\left(n+p\right)^2+n+p\right)/2$ scalar variables, where p is the number of outputs to the system. This is even more variables than the original primal formulation, and hence KYPD will usually be slower than SeDuMi for this specific example.

4 Conclusions

In this paper, a structure exploiting algorithm for KYP-SDPs where some of the constraints appear as complicating constraints is proposed. The structure of the KYP-SDP is utilized to reduce the computational complexity. The algorithm is basically the Kelley-Cheney-Goldstein cutting plane method (Kelley, 1960, Cheney and Goldstein, 1959). The convergence of the method is established if the technical but important condition that Z_k is bounded holds, see for example (Lemaréchal, 2001). A sufficient condition for this is that there exist an interior point for the problem (19) (Wolkowicz et al., 2000, Thm. 4.1.3).

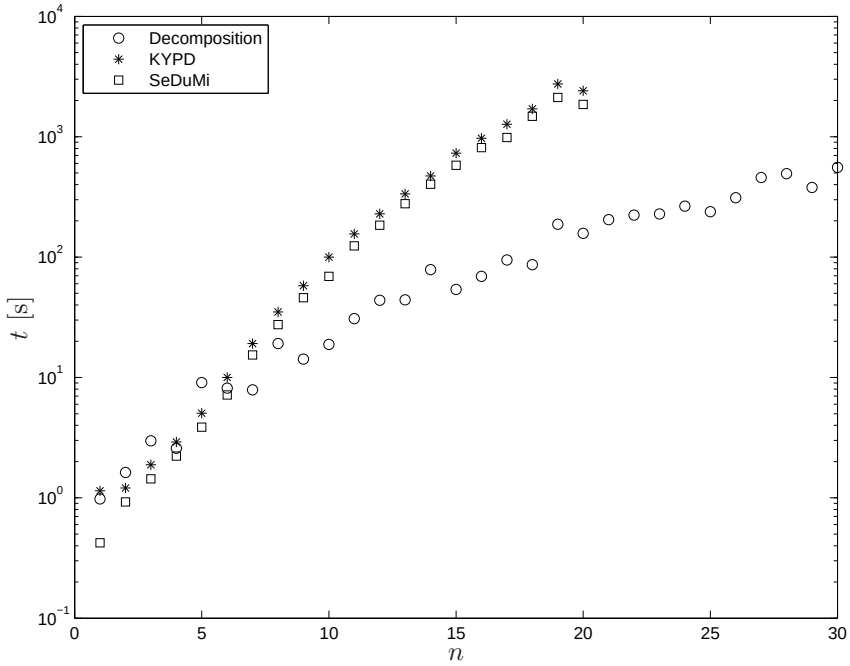


Figure 3: *Averaged computational times for the controller synthesis problem of SISO systems. Note that n denotes the number of states in the original plant and not the size of the matrices that are involved in the actual computations.*

That this is indeed the case for all possible problems is still an open question. However, our experience is that we have had no problems with convergence using the algorithm.

We remark that the worst case complexity of the number of iterates is proportional to $\mathcal{O}(1/\epsilon^m)$ where m is the dimension of the dual variable, which is very poor.

In a numerical example, it is shown that it is beneficial to use the proposed algorithm in some cases. It is advantageous to use the algorithm in cases where one or more constraints is associated with a large dual variable that has no specific structure that can be exploited.

References

R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$ [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, 1994. ISBN 0-89871-334-X.

S.P. Boyd and C.H. Barratt. *Linear controller design: limits of performance*. Englewood Cliffs, NJ: Prentice Hall, 1991.

S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

E.W. Cheney and A.A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1(1):253–268, 1959.

M. Farhoodi and M.T.H. Beheshti. A case study of the multiobjective H_2/H_∞ control via finite dimensional youla parameterization and LMI optimization. *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 493–497, 2007.

A. M. Geoffrion. Primal resource-directive approaches for optimizing nonlinear decomposable systems. *Operations Research*, 18(3):375–403, may 1970. ISSN 0030-364X.

H.A. Hindi, B. Hassibi, and S.P. Boyd. Multiobjective H_2/H_∞-optimal controlvia finite dimensional Q-parametrization and linear matrix inequalities. *American Control Conference, 1998. Proceedings of the 1998*, 5, 1998.

C.-Y. Kao, A. Megretski, and U. Jönsson. Specialized fast algorithms for IQC feasibility and optimization problems. *Automatica*, 40(2):239 – 252, 2004. ISSN 0005-1098.

J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8(4):703–712, 1960.

P. Khargonekar and M Rotea. Multiple objective optimal control of linear systems: the quadratic norm case. *Automatic Control, IEEE Transactions on*, 36(1):14–24, 1991.

L. S. Lasdon. *Optimization Theory for Large Systems*. MacMillan Series in Operations Research. MacMillan Publishing, 1970.

C. Lemaréchal. Lagrangian relaxation. In M. Jünger and D. Nadded, editors, *Computational Combinatorial Optimization*, pages 112 – 156. Springer Verlag, Heidelberg, 2001.

J. Löfberg. YALMIP: a toolbox for modeling and optimization in MATLAB. *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289, 2004.

A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819 – 830, 1997. ISSN 0018-9286.

J. Oishi and V. Balakrishnan. Linear controller design for the NEC laser bonder via LMI optimization. *Advances in Linear Matrix Inequality Methods in Control, Advances in Control and Design. SIAM*, 1999.

A. Rantzer. On the Kalman-Yakubovich-Popov lemma. *Systems and Control Letters*, 28 (1):7 – 10, 1996. ISSN 0167-6911.

J.F. Sturm. Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1):625–653, 1999.

R. Wallin, C.-Y. Kao, and A. Hansson. A cutting plane method for solving KYP-SDPs. *Automatica*, 44(2):418 – 429, 2008. ISSN 0005-1098.

R. Wallin, A. Hansson, and J. H. Johansson. A structure exploiting preprocessor for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. *IEEE Transactions on Automatic Control*, 54(4):697–704, April 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2014922.

H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, 2000.

A Appendix

Each of the subproblems in (12) are on the form

$$\begin{aligned} &\min_{x,P} \ \langle C, P\rangle + c^T x \\ &\text{s.t. } \mathcal{F}(P) + M_0 + \mathcal{G}(x) \preceq 0. \end{aligned} \tag{26}$$

One way to make this optimization problem bounded is to add the constraint

$$\langle C, P\rangle + c^T x - f_{\min} \geq 0. \tag{27}$$

The dual of (26) with the extra constraint (27) is

$$\begin{aligned} &\max_{\lambda,z} \ \langle M_o, Z\rangle - f_{\min}\lambda \\ &\text{s.t. } \mathcal{F}^*(Z) + C(1+\lambda) = 0 \\ &\quad \mathcal{G}^*(Z) + c(1+\lambda) = 0 \\ &\quad Z \succeq 0 \\ &\quad \lambda \leq 0. \end{aligned} \tag{28}$$

We can eliminate some of the dual variables Z using the constraint $\mathcal{F}^*(Z)+C(1+\lambda) = 0$ in a very similar way as was done in (Wallin et al., 2009). In (Wallin et al., 2009), the number of variables in Z is reduced by finding a basis for the nullspace of $\mathcal{F}^*(Z)+C = 0$. The only difference is that we have the term $C(1+\lambda)$ instead of C. Hence the only difference compared to (Wallin et al., 2009) is that we need to find a solution to the Lyapunov equation

$$AF_0 + F_0A^T + C(1+\lambda) = 0, \tag{29}$$

where λ is a variable. A solution to this is $F_0 = F_c(1+\lambda)$ where F_c solves the Lyapunov equation

$$AF_c + F_cA^T + C = 0. \tag{30}$$

To see this, just insert F_0 in (29) and we get

$$\begin{aligned} AF_0 + F_0A^T + C(1+\lambda) = \\ AF_c(1+\lambda) + (1+\lambda)F_cA^T + C(1+\lambda) = \\ \underbrace{AF_c + F_cA^T + C}_{=0} + \lambda\left(\underbrace{AF_c + F_cA^T + C}_{=0}\right) = 0. \end{aligned} \tag{31}$$

Having this basis we can solve the dual problem using the reduced dual variables. The reconstruction of P and x is not affected by this and can be found in the exact same fashion as in (Wallin et al., 2009).

Paper B

Lowrank exploitation in semidefinite programming for control

Authors: Rikard Falkeborn, Johan Löfberg and Anders Hansson

Published as

R. Falkeborn, J. Löfberg, and A. Hansson. Lowrank exploitation in semidefinite programming for control. Technical Report LiTH-ISY-R-2927, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, January 2010

Lowrank exploitation in semidefinite programming for control

Rikard Falkeborn, Johan Löfberg and Anders Hansson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden

Abstract

Many control related problems can be cast as semidefinite programs but, even though there exist polynomial time algorithms and good publicly available solvers, the time it takes to solve these problems can be long. Something many of these problems have in common, is that some of the variables enter as matrix valued variables. This leads to a low-rank structure in the basis matrices which can be exploited when forming the Newton equations. In this paper, we describe the how this can be done, and show how our code can be used when using SDPT3. The idea behind this is old and is implemented in LMI Lab, but we show that when using a modern algorithm, the computational time can be reduced. Finally, we describe how the modeling language YALMIP is changed in such a way that our code can be interfaced using standard YALMIP commands, which greatly simplifies for the user.

1 Introduction

Semidefinite programming (SDP) (Wolkowicz et al., 2000) has gained a lot of interest within the control community, and is now a well established mathematical tool in the field. Many fundamental control problems can be cast as semidefinite programs (Boyd et al., 1994), for example proving robust stability (Gahinet et al., 1996, Iwasaki and Shibata, 2001, Megretski and Rantzer, 1997).

Moreover, since the beginning of the 90s, there exist efficient algorithms to solve these SDPs in a time which is polynomial in the number of variables and constraints (Nesterov and Nemirovsky, 1994). Today, there are several solvers freely available, such as SeDuMi (Sturm, 1999), SDPT3 (Tütüncü et al., 2003) and SDPA (Yamashita et al., 2003) just to mention a few.

Although SDPs can be solved in polynomial time, the number of variables in the problems often grow rapidly and the time needed to solve the problems can be substantial, even though the plant size is modest. Because of this fact, tailor-made solvers for various types of problems have been developed, for example for programs derived from the Kalman-Yakubovic-Popov lemma (Wallin et al., 2008, 2009, Kao et al., 2004, Liu and Vandenberghe, 2007, Vandenberghe et al., 2004).

However, a problem with these tailor-made solvers is that they often are limited to very particular control problems, thus making them non-applicable for more complex problems where only some part of the problem specification happens to have the structure that can be exploited. Additionally, these efficient solvers are typically hard to hook up to easily used modeling languages, such as YALMIP (Löfberg, 2004), which ultimately means that few users actually will benefit from them.

This paper describes the implementation of a structure exploiting assembler for the Schur complement matrix that can be used in for example SDPT3. The assembler utilizes the fact that many problems in systems and control theory have matrix valued variables which lead to low rank structure in the basis matrices. Additionally, we present a new version of YALMIP which will allow the user to describe control problems in a very natural standard YALMIP format, and take care of the intricate communication between SDPT3 and the structure exploiting code.

To be fair, it should be pointed out that the theoretical idea exploited in this paper was incorporated already in LMI Lab (Gahinet et al., 1995), which was one of the first public implementations of an SDP solver, tailored specifically for control problems. However, many years of active research in the SDP field has led to much more efficient algorithms, and it is our goal to leverage on this. A major reason for pursuing the idea in this paper is that it is slightly embarrassing for people coming from the SDP field, that the now over 15 year old LMI Lab solver actually outperforms state-of-the-art general purpose SDP solvers on small- to medium-scale control problems.

The notation is standard. We let $A \succ B (A \succeq B)$ denote that $A - B$ is positive (semi)definite, $\mathbf{R}^n$ denotes the set of real vectors of dimension n and $\mathbf{S}^m$ denotes the set of real symmetric matrices of dimension m. The inner product is denoted by $\langle A, B \rangle$ and is for matrices defined as $\mathrm{Tr}\left(A^T B\right)$

2 Semidefinite programming

A semidefinite program can be written as

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t} \quad & F_0 + \sum_{i=1}^{n} F_i x_i = X \\ & X \succeq 0, \end{aligned} \tag{1}$$

where $c, x \in \mathbf{R}^n$ and $X, F_i \in \mathbf{S}^m$. For future reference, let us also state the dual of this problem. The dual of (1) is

$$\begin{aligned} \max_{Z} \quad & \langle F_0, Z \rangle \\ \text{s.t} \quad & c_i + \langle F_i, Z \rangle = 0, \quad i = 1, \ldots, n \\ & Z \succeq 0, \end{aligned} \tag{2}$$

where $Z \in \mathbf{S}^m$.

Almost all modern SDP-solvers today are using interior-point methods. That means, in loose terms, we are solving both the primal and the dual problem at the same time. The main parts of these algorithms consist of forming a system of equations to solve for the search directions needed, solve that system of equations and then do a line search in order to find out the appropriate step size. This procedure is then repeated until a stopping criterion is fulfilled. This stopping criterion may for example be that we are sufficiently close to the optimum.

We remark that forming the system of equations can be computationally expensive, and is in some cases by far the most time-consuming part of the algorithm.

If we let the system of equations to be solved in order to get the search directions be

$$H \Delta x = b, \tag{3}$$

where H is the coefficient matrix, which is usually symmetric and Δx is the search direction, the elements are given by $H_{ij} = \langle F_i, U F_j V \rangle$ when using the Helmberg-Kojima-Monteiro (HKM) direction (Helmberg et al., 1996, Kojima et al., 1997, Monteiro, 1997), and by $H_{ij} = \langle F_i, W F_j W \rangle$ when using the Nesterov-Todd direction (Nesterov and Todd, 1997). Here, H_{ij} denotes the ijth element of the matrix H, and U, V and W are scaling matrices.

In SDP programs encountered in systems and control, the majority of variables x_i in (1) do not come from scalar entities but rather as parts of a matrix variable. However, there is no way to inform any of the public solvers about this fact. Instead, we will use an approach where we supply the solver with the code to assemble the Hessian.

To illustrate this, let us take the Lyapunov inequality as an example. The Lyapunov inequality is

$$A^T P + P A + Q \preceq 0, \tag{4}$$

with $Q \succeq 0$. In order to put this inequality on the form (1), which is used by most solvers today, we let $F_0 = -Q$, $F_i = -A^T E_i - E_i A$ where $E_1, \ldots, E_m$ is a basis for $\mathbf{S}^n$, $m = n\,(n+1)\,/2$. However, by doing so, we lose a lot of information that can be used in the formulation of the Schur matrix.

The fact that we can choose E_i as any basis for $\mathbf{S}^n$ can be exploited. This means we can choose $E_i = e_k e_l^T + e_l e_k^T$, if the variable x_i is an off-diagonal element and $E_i = e_k e_k^T$ if x_i is an element on the diagonal, where e_i is a unit vector in $\mathbf{R}^n$. Now, let us compute one element of the resulting Schur matrix for the Lyapunov inequality (4), assuming the

HKM direction is used. This can be done by

$$\begin{aligned} H_{op} = \big\langle & A^T \left(e_i e_j^T + e_j e_i^T\right) + \left(e_i e_j^T + e_j e_i^T\right) A, \\ & U \left(A^T \left(e_k e_l^T + e_l e_k^T\right) + \left(e_k e_l^T + e_l e_k^T\right) A\right) V \big\rangle = \\ & e_k^T AUA^T e_i e_j^T V e_l + e_k^T AUA^T e_j e_i^T V e_l + \\ & e_l^T AUA^T e_i e_j^T V e_k + e_l^T AUA^T e_j e_i^T V e_k + \\ & e_k^T AUe_i e_j^T AV e_l + e_k^T AUe_j e_i^T AV e_l + \\ & e_l^T AUe_i e_j^T AV e_k + e_l^T AUe_j e_i^T AV e_k + \\ & e_i^T AUe_l e_k^T AV e_j + e_j^T AUe_l e_k^T AV e_i + \\ & e_i^T AUe_k e_l^T AV e_j + e_j^T AUe_k e_l^T AV e_i + \\ & e_l^T Ue_i e_j^T AVA^T e_j + e_l^T Ue_j e_i^T AVA^T e_k + \\ & e_k^T Ue_i e_j^T AVA^T e_l + e_k^T Ue_j e_i^T AVA^T e_l. \end{aligned} \tag{5}$$

Since $e_i^T B e_j$ is just the ijth element of B, the element H_{op} is just a sum of products of elements from the matrices AUA^T, AU, AVA^T, AV, U and V. Moreover, the matrices involved are the same for all the positions in the Schur matrix. Hence they can be precomputed once in each iteration. This is the structure exploiting idea that was incorporated already in LMI Lab for a projective method. The reason they could exploit it, while modern general purpose solvers fail to, is that the user has to specify the matrices and their position in the constraints in a very detailed, by many regarded cumbersome, fashion.

In this paper we present an extension to the modeling language YALMIP which allows the user to utilize this structure for interior-point methods using the semidefinite programming solver SDPT3 (Tütüncü et al., 2003).

We remark that this is not limited to symmetric matrix variables and a single Lyapunov inequality, but more general constraints and variables can be used, as will be described in the following section.

3 SDPs considered

In the paper, we consider SDPs where the constraints are on the following form.

$$\sum_{i=1}^{N_{im}} \sum_{j=1}^{N_{jm}} L_{ijm} P_j R_{ijm}^T + R_{ijm} P_j^T L_{ijm}^T + \sum_{i=1}^{N_{im}} \sum_{j=1}^{N_{jm}} A_{ijm}^T P_j A_{ijm} + M_{0m} + \sum_{k=1}^{p} M_{km} x_k \preceq 0, \quad m = 1, \ldots, N \tag{6}$$

where P_j and x_k are the optimization variables, and where all the matrices are assumed to have suitable dimensions.

We assume the basis matrices for P_j can be written as

$$E_j = \sum_{h=1}^{\alpha_j} \varepsilon_{hj} \delta_{hj}^T, \tag{7}$$

where ε_{hj} and δ_{hj} are assumed to be unit vectors in appropriate vector spaces.

This implies that P_j can be a symmetric matrix, rectangular matrix, matrix with block diagonal structure, tridiagonal structure, skew-symmetric and many more. We assume M_{km} has no exploitable structure.

For easier presentation, we will drop the indeces m, i.e. the indeces that indicate which constraint the matrices belong to.

We now show what the elements in the Schur matrix with respect to the j_1th and j_2th elements in P_j are, for the first term in (6). The corresponding element in the Schur matrix is

$$H_{j_1 j_2} = \left\langle L_{j_1} \sum_{h=1}^{\alpha_{j_1}} \varepsilon_{hj_1} \delta_{hj_1}^T R_{j_1}^T, U L_{j_2} \sum_{h=1}^{\alpha_{j_2}} \varepsilon_{hj_2} \delta_{hj_2}^T R_{j_2}^T V \right\rangle = \sum_{h=1}^{\alpha_{j_1}} \sum_{h=1}^{\alpha_{j_2}} \delta_{hj_1}^T R_{j_1}^T U L_{j_2} \varepsilon_{hj_2} \delta_{hj_2}^T R_{j_2}^T V L_{j_1} \varepsilon_{hj_1}. \quad (8)$$

It is clear that the for the other terms in (6), the expression will be similar. As an example, for the second term in (6), just interchange L_j and R_j, and ε_{hj} and δ_{hj}. We remark that the entry in the Schur matrix for the jth element in P_j, with respect to the first term in (6) and x_k can be written as

$$H_{jk} = \left\langle L_j \sum_{h=1}^{a_j} \varepsilon_{hj} \delta_{hj}^T R_j^T, U M_k V \right\rangle = \sum_{h=1}^{a_j} \left\langle \delta_j^T R_j^T U M_k V L_i \varepsilon_j \right\rangle. \quad (9)$$

Also in this case, the contribution from the other terms in (6) is very similar. It is obvious from the discussion in the previous section what matrices to precompute and that this will speed up the computations. Finally, we remark that for the unstructured matrices, M_k, the entry in the Schur matrix will be

$$H_{k_1 k_2} = \langle M_{k_1}, U M_{k_2} V \rangle, \quad (10)$$

just as it is implemented in SDPT3.

As a last remark in this section, we mention that since sparsity in the basis matrices in (10) is exploited by solvers, the more sparsity in the basis matrices, the faster will the computations in (10) be. We also see that the computations in (8) will not be affected by sparsity in the basis matrices. Hence, the more full the basis matrices are, the better it will be to use (8) in order to assemble the Schur matrix. We also mention that a continuous time Lyapunov inequality, where the basis matrices have the form $A^T E_i + E_i A$ will be relatively sparse, will have roughly $4n$ out of n^2 elements that are non-zero, while a discrete time Lyapunov inequality, where the basis matrices are on the form $A^T E_i A - E_i$ will have all n^2 elements full, unless of there is some sparsity in A, which indicates that the proposed method will be relatively better for discrete time systems than continuous time systems.

4 Implementation

The solver SDPT3 allows for the user to provide the solver with a function that handles the assembly of the Schur matrix. We have written a function that computes the Schur

matrix as described in Section 3. As input, the function takes the matrices R_{ijk}, L_{ijk}, A_{ijk}, M_{0k}, M_{ijk} from (6) and information about the basis matrices in (7). The computations of the elements in the matrix H in (8) are done using mex-files for increased performance. We remark that the case where we compute the element in H where we have two unstructured matrices as in (10), we use the built in function in SDPT3. To specify all these arguments can be cumbersome, but if YALMIP is used, the user does not have to care about specifying any of the low-level input arguments, since this is done automatically by YALMIP, as will be described in the next section.

5 Improvement of the YALMIP language

One of the most important steps is to make the whole framework easily accessible to the casual user. An efficient solver with a cumbersome interface will have little impact in practice. A first step towards incorporation of an efficient structure-exploiting solver for control was the YALMIP interface to the solver KYPD (Wallin et al., 2008). Although this interface allowed users to describe problems to KYPD in a fairly straightforward fashion, it still required special-purpose commands specific to this solver. The reason for this is that the core idea in YALMIP is that all expressions are evaluated on the fly, and only the basis-matrices and decision variables are kept. In other words, all expressions are immediately disaggregated and knowledge of underlying matrix variables is lost.

To circumvent this, a new version of YALMIP has been developed. To be able to use the efficient solver described in this paper, it is essential that YALMIP keeps track of aggregated matrix variables. Hence, when an expression is evaluated, YALMIP saves information internally about the factors that constitute the constraints, essentially corresponding to the matrices L and R in (6). These factors are also tracked when some basic operations are performed, such as concatenation, addition, subtraction and multiplication. The factors are however not guaranteed to be kept in highly complex manipulations. If we use an operator for which the factor-tracking not is supported, the expression will be disaggregated, and constraints involving this expression will be handled as a standard SDP constraint by the solver.

To summarize, for the user, standard YALMIP code applies and nothing has changed, the only difference is that in some problems structure will automatically be detected and exploited. As an example, the example described in the following section would be coded as

```
P = sdpvar(n);
x = sdpvar(nx,1);
M = M0+x(1)*M1+x(2)*M2+x(3)*M3
F = [A'*P+P*A P*B;B'*P zeros(m)]+M > 0
O = trace(C*P)+c'*x
```

Knowledge about the way the matrix variable P enters the problem will be tracked by YALMIP and exploited.

6 Computational Results

In this section, we give some computational results that demonstrates that in some cases, it is advantageous to use this way of computing the Schur matrix. The first example is on the following form and is taken from (Johansson and Hansson, To appear). The SDPs we solve have the following structure

$$\begin{aligned} \min_{x,P} \quad & \langle C, P\rangle + c^T x \\ \text{s.t} \quad & \begin{bmatrix} A_i^T P + P A_i & P B_i \\ B_i^T P & 0 \end{bmatrix} + \\ & M_{i,0} + \sum_{k=1}^{n_x} x_k M_{i,k} \succeq 0, \quad i = 1, \ldots, n_i \end{aligned} \tag{11}$$

where the decision variables are $P \in \mathbf{S}^n$ and $x \in \mathbf{R}^{n_x}$. All data matrices were generated randomly, but certain care was taken in order for the optimization problems to be feasible. See (Johansson and Hansson, To appear) for the exact details on how the matrices were generated. This type of LMIs appear in a vast number of analysis problems for linear differential inclusions (Boyd et al., 1994).

The optimization problem (11) can easily be put on the form (6) with

$$L_i = \begin{bmatrix} A_i^T \\ B_i^T \end{bmatrix}, \qquad R_i = \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{12}$$

We solve the problem (11) for increasing numbers of states n. We keep $n_i = 3$ and $n_x = 3$ constant for all the problems. The problem was solved for 10 times for each n and the average solution times are plotted in Figure 1.

As can be seen in Figure 1, the solution times decrease if we use the tailor made code for the Schur compilation.

We also give a second, slightly more complicated example. The example is a model reduction algorithm from (Helmersson, 1994) where semidefinite programming is used to reduce the order of a linear time-invariant (LTI) system. A short description of the algorithm now follows.

It is well known that the H_∞-norm γ of an LTI system can be computed as

$$\min_{\gamma,P} \gamma \tag{13}$$

$$\begin{bmatrix} A^T P + P A & P B & C \\ B^T P & -\gamma I & D \\ C^T & D^T & -\gamma I \end{bmatrix} \prec 0, \tag{14}$$

We also know that the difference of two systems $\tilde{G} = G - \hat{G}$ can be written on state space form with the matrices, where $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ are the state-space matrices of a realization of $\tilde{G}$, and analogously with G and $\hat{G}$,

$$\left[\begin{array}{c|c} \tilde{A} & \tilde{B} \\ \hline \tilde{C} & \tilde{D} \end{array}\right] = \left[\begin{array}{cc|c} A & 0 & B \\ 0 & \hat{A} & \hat{B} \\ \hline C & -\hat{C} & D - \hat{D} \end{array}\right]. \tag{15}$$

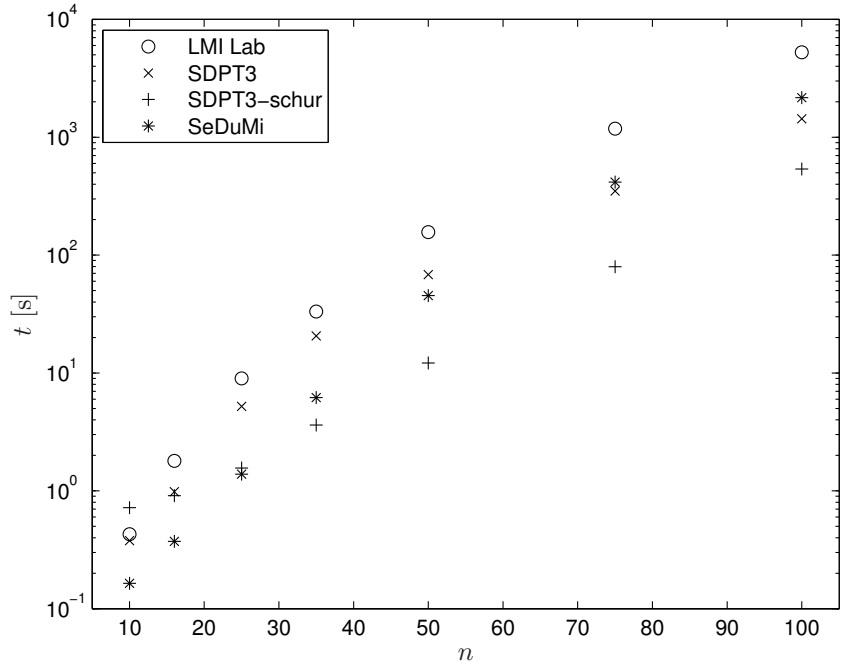


Figure 1: *Averaged computational times.*

Now, using $\tilde{G}$ in (13), and by partitioning the matrix P into

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{bmatrix} \succ 0, \tag{16}$$

and using this in (13), we get,

$$\min_{\hat{A},\hat{B},\hat{C},\hat{D},P_{11},P_{12},P_{22},\gamma} \gamma$$
$$\begin{bmatrix} A^TP + PA & A^TP_{12} + P_{12}\hat{A} & P_{11}B - P_{12}\hat{B} & C^T \\ \hat{A}^TP_{12}^T + P_{12}^TA & \hat{A}^TP_{22} + P_{22}\hat{A} & P_{12}^TB - P_{22}\hat{B} & \hat{C}^T \\ B^TP_{11} - \hat{B}^TP_{12}^T & B^TP_{12} - \hat{B}^TP_{22} & -\gamma I & D^T - \hat{D}^T \\ C & \hat{C} & D - \hat{D} & -\gamma I \end{bmatrix} \prec 0. \tag{17}$$

Since this is a bilinear matrix inequality (BMI), the approach taken in (Helmersson, 1994) is to fix some matrices to be constant to make it an LMI, solve that optimization problem, and then fix other matrices. This is best described in the following algorithm from (Helmersson, 1994).

i Start with a $\hat{G}$ obtained from, for example a truncated balanced realization

ii Solve (17) subject to (16) with respect to $(P, \hat{C}, \hat{D})$, keeping $(\hat{A}, \hat{B})$ constant.

iii Solve (17) subject to (16) with respect to $(P_{11}, \hat{A}, \hat{B}, \hat{C}, \hat{D})$, keeping (P_{12}, P_{22}) constant.

iv Repeat *ii* and *iii* until some given convergence criteria is met.

We remark that this procedure does not guarantee global convergence.

Our numerical experience with this algorithm indicates that the numerical properties of the LMIs we need to solve is improved if we let (A, B, C, D) be a balanced realization of G.

We test the algorithm using using SDPT3 both with and without our special purpose Schur-compiler. The systems we test it on are from the Compleib library (Leibfritz, 2003). We remark that for the H_∞-norm to be well defined, the systems must be stable, i.e. all eigenvalues must have strictly negative real parts. Unfortunately, this is not the case for most of the models in the Compleib library. In an attempt to increase the number of models we can use, we shift the spectrum of the A-matrices in some models such that no eigenvalue have larger real part than -1. Results of this is summarized in Table 1. As can be seen in the table, the computational times can be reduced by the use of our code.

Table 1: *Comparisson of times. Here, n_x is the number of states in the original model, n_{red} is the number of states in the reduced system, n_u is the number of inputs, n_y is the number of outputs, t_{STRUL} and t_{SDPT3} is the time used by SDPT3 with and without the Schur-compiler. The time is for doing one round of iterations in the algorithm outlined above. The models LAH and JE1 are used without doing any shifting of the spectrum, while the other models where first shifted in order to get stable models.*

Name	n_x	n_{red}	n_u	n_y	t_{STRUL}	t_{SDPT3}
LAH	48	18	1	1	211.01	496.17
JE1	24	4	3	5	12.13	26.54
AC10	48	10	2	2	116.51	315.82
AC13	24	8	3	4	15.64	32.33
JE2	21	4	3	3	10.13	19.41
JE3	21	4	3	6	9.12	18.43
IH	20	10	11	10	15.72	30.64
CSE1	19	4	2	10	6.09	10.41
EB5	38	9	1	1	55.09	125.26

7 Conclusions

In this paper we have presented a dedicated assembler for the Schur matrix for SDPT3. The Schur matrix is the coefficient matrix that defines the system of equations used in order to solve for the search directions. The assembler utilizes the fact that many semidefinite programs in systems and control theory involve large matrix variables which implies that the basis matrices have a special low rank structure that can be exploited in order reduce the computational burden. We also presented a related extension to the modelling

language YALMIP which allows us to keep track of aggregated matrix variables, and exploit these in a solver. In two examples, it was demonstrated that using this method can be beneficial. The first example was an academic example where the SDP has the so called KYP structure for increasing sizes of the problem. It this example, the speedup using our code was about five times faster than using SDPT3, SeDuMi and LMI Lab for some sizes of the problem. In the second example, we tested the code on a model reduction algorithm on models from the COMPLIB library. The speed up here was not as good as in the previous example, but still most of the examples are at least twice as fast as SDPT3. There are several contributing factors to this. The major one is that in the first example, we have multiple constraints which include the same variables. This means the time spent on assembling the Schur matrix is three (since we had three constraints) times as large as if we had only had one constraint, but the time to solve for the search directions is only dependent on the number of variables. In the second example we do not have this situation. Finally, we remark that since sparsity in the basis matrices is beneficial for SDPT3, our code would be even better on discrete time problems since these types of problems often have full basis matrices.

References

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, 1994. ISBN 0-89871-334-X.

P. Gahinet, A. Nemirovski, AJ Laub, and M. Chilali. LMI control toolbox. *The MathWorks Inc*, 1995.

P. Gahinet, P. Apkarian, and M. Chilali. Affine parameter-dependent Lyapunov functions and real parametric uncertainty. *IEEE Transactions on Automatic Control*, 41(3):436 – 442, 1996. ISSN 0018-9286.

C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

A. Helmersson. Model reduction using LMIs. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 3217–3222, Orlando, Florida, February 1994.

T. Iwasaki and G. Shibata. LPV system analysis via quadratic separator for uncertain implicit systems. *IEEE Transactions on Automatic Control*, 46(8):1195 – 1208, 2001. ISSN 0018-9286.

J. H. Johansson and A. Hansson. An inexact interior-point method for system analysis. *International Journal of Control*, To appear.

C.-Y. Kao, A. Megretski, and U. Jönsson. Specialized fast algorithms for IQC feasibility and optimization problems. *Automatica*, 40(2):239 – 252, 2004. ISSN 0005-1098.

M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7:86, 1997.

F. Leibfritz. COMPLeIB, COnstraint Matrix-optimization Problem LIbrary-a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Technical report, Technical report, Universität Trier, 2003, 2003.

Z. Liu and L. Vandenberghe. Low-rank structure in semidefinite programs derived from the KYP lemma. In *Proceedings of the 46th IEEE Conference on Decision and Control*. Citeseer, 2007.

J. Löfberg. YALMIP: a toolbox for modeling and optimization in MATLAB. *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289, 2004.

A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819 – 830, 1997. ISSN 0018-9286.

R.D.C. Monteiro. Primal–dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization*, 7(3):663–678, 1997.

Y. Nesterov and A. Nemirovsky. Interior point polynomial methods in convex programming. *Studies in applied mathematics*, 13, 1994.

Y. 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. ISSN 0364765X. URL `http://www.jstor.org/stable/3690138`.

J.F. Sturm. Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1):625–653, 1999.

R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95(2):189–217, 2003.

L. Vandenberghe, V.R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. Interior-point algorithms for semidefinite programming problems derived from the KYP lemma. *Positive Polynomials in Control, Lectures Notes in Control and Information Science. Springer*, 2004.

R. Wallin, C.-Y. Kao, and A. Hansson. A cutting plane method for solving KYP-SDPs. *Automatica*, 44(2):418 – 429, 2008. ISSN 0005-1098.

R. Wallin, A. Hansson, and J. H. Johansson. A structure exploiting preprocessor for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. *IEEE Transactions on Automatic Control*, 54(4):697–704, April 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2014922.

H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, 2000.

M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). *Optimization Methods and Software*, 18 (4):491–505, 2003.

Licentiate Theses
Division of Automatic Control
Linköping University

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.
B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.
A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No. 75, 1986.
G. Malmberg: A Study of Adaptive Control Missiles. Thesis No. 76, 1986.
S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.
M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.
K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.
A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.
P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.
K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.
I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.
F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.
H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.
S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.
K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.
J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.
P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.
H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.
T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.
T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.
P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.
J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.
J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.
A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.
M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.
K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.
J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.
A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.
M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.
U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

S. Björklund: A Survey and Comparison of Time-Delay Estimation Methods in Linear Systems. Thesis No. 1061, 2003.

M. Gerdin: Parameter Estimation in Linear Descriptor Systems. Thesis No. 1085, 2004.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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