Available online at www.sciencedirect.com
ScienceDirect
Automatica 42 (2006) 1875–1882
automatica
www.elsevier.com/locate/automatica

Brief paper

A Newton-like method for solving rank constrained linear matrix inequalities☆

Robert Orsi[a,*], Uwe Helmke[b], John B. Moore[a,c]

[a]*Research School of Information Sciences and Engineering, Australian National University, Canberra ACT 0200, Australia*
[b]*Department of Mathematics, University of Würzburg, AM Hubland 97074 Würzburg, Germany*
[c]*National ICT Australia Limited, Locked Bag 8001, Canberra ACT 2601, Australia*

Received 29 April 2005; received in revised form 7 April 2006; accepted 29 May 2006
Available online 22 August 2006

Abstract

This paper presents a Newton-like algorithm for solving systems of rank constrained linear matrix inequalities. Though local quadratic convergence of the algorithm is not a priori guaranteed or observed in all cases, numerical experiments, including application to an output feedback stabilization problem, show the effectiveness of the algorithm.

Keywords: Linear matrix inequalities; Rank constraints; Computational methods; Output feedback; Stabilization; Robust control; Polynomial constraints

1. Introduction

The *linear matrix inequality* (LMI) problem is a well known type of convex feasibility problem that has found many applications to controller analysis and design. The *rank constrained LMI* problem is a natural as well as important generalization of this problem. It is a non-convex feasibility problem defined by LMI constraints together with an additional matrix rank constraint.

Interest in rank constrained LMIs arises as many important output feedback and robust control problems, that cannot always be addressed in the standard LMI framework, can be formulated as special cases of this problem (El Ghaoui & Gahinet, 1993; Goh, Safonov, & Ly, 1996; Mesbahi, Safonov, & Papavassilopoulos, 1999; Skelton, Iwasaki, & Grigoriadis, 1998). Examples include *bilinear matrix inequality* (BMI) problems, see Goh et al. (1996) and Mesbahi et al. (1999), that are easily seen to be equivalent to rank one constrained LMI problems.

In addition to their importance for control, rank constrained LMIs also appear naturally in mathematical programming and combinatorial optimization tasks: all optimization problems with polynomial objective and polynomial constraints can be reformulated as LMI optimization problems with a rank one constraint (Boyd & Vandenberghe, 1997; Nesterov & Nemirovskii, 1994).

In general, if the set of points that satisfy an LMI is non-empty, then a numerical solution to the LMI problem can be efficiently found using well developed interior point algorithms, see for example Vandenberghe & Boyd (1996). Lack of convexity makes the rank constrained LMI problem much harder to solve. Currently available algorithms for the rank constrained LMI problem are largely heuristic in nature and are not guaranteed to converge to a solution even if one exists. Solution methods for this problem, or certain specializations of the problem, include those based on linearization (El Ghaoui, Oustry, & Ait Rami, 1997); alternating projections (Beran & Grigoriadis, 1996; Grigoriadis & Beran, 1999; Grigoriadis

☆ Due to space limitations, the paper does not include proofs. See Orsi, Helmke, and Moore (2006) for an extended version of the paper which includes proofs and other additional material. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Didier Henrion under the direction of Editor Roberto Tempo.

* Corresponding author. Tel.: +61 2 6125 8637; fax: +61 2 6125 8660.
E-mail addresses: robert.orsi@anu.edu.au (R. Orsi), helmke@mathematik.uni-wuerzburg.de (U. Helmke), john.moore@anu.edu.au (J.B. Moore).

0005-1098/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2006.05.026

& Skelton, 1996); trace minimization methods that try to solve the problem by solving a related convex problem (Pare, 2000); augmented Lagrangian methods (Fares, Apkarian, & Noll, 2001); and sequential semidefinite programming (Fares, Noll, & Apkarian, 2002). Aside from Fares et al. (2002), these methods do not have established superlinear convergence rates and the challenge remains to find numerical schemes with verifiable local quadratic convergence rates.

In this paper we present a new heuristic method for solving the rank constrained LMI problem. The method is closely related to existing alternating projection methods but is expected to have improved convergence properties due to a built-in Newton-type step. In Grigoriadis and Skelton (1996) and Grigoriadis and Beran (1999) alternating projection algorithms are proposed that involve tangent-like ideas, similar to our approach. However, the implementation details are different and the connection to the Newton method is neither mentioned nor obvious. In fact, it is this established connection to Newton's method that distinguishes our approach from earlier ones.

Our method is based on the "tangent and lift" methodology (Chu, 1992), a generalization of Newton's method. While the classical Newton algorithm can be used to find zeroes of functions, the tangent and lift method is more general and can be used to find a point in the intersection of an affine subspace and a manifold. We show that the rank constrained LMI problem can be formulated as a problem of finding a point in the intersection of an affine subspace and another set which, though not a manifold, is a finite union of manifolds. Part of the contribution of this paper is a demonstration that tangent and lift methods can be extended to this more general setting and we present an algorithm for solving the rank constrained LMI problem based on such an extension. Numerical experiments show the effectiveness of this approach.

Since our method is based on a generalization of the Newton method, local quadratic convergence to isolated solutions is expected. However, complications arise due to the non-smoothness of the constraints as well as the possibility of continua of solutions. This makes a rigorous convergence theory difficult to develop and in fact, as some of our experiments show, local quadratic convergence cannot be expected in all cases. The challenge therefore is to single out a class of problems for which local quadratic convergence can be rigorously established.

The rest of the paper is structured as follows. Section 2 contains a statement of the rank constrained LMI problem and a reformulation of this problem into an equivalent form. Section 3 contains a discussion of the tangent and lift method and details of how we extend this methodology so that it can be applied to the rank constrained LMI problem. Section 4 discusses important geometric properties of rank constrained positive semidefinite matrices. Our algorithm for solving the rank constrained LMI problem is given in Section 5. Section 6 considers various numerical implementation issues. It includes a discussion of how the basic approach extends to enable the solution of more general problems such as those with multiple rank constraints. Section 7 reports on some numerical experiments and includes an application of the algorithm to an output feedback problem. The paper ends with some concluding remarks.

2. Problem formulation

Let $\mathbb{R}$ denote the set of real numbers and $\mathcal{S}^n$ denote the set of real symmetric $n \times n$ matrices. For $A \in \mathcal{S}^n$, let $A \succcurlyeq 0$ denote the property that A is positive semidefinite. The rank constrained LMI problem is the following:

Problem 1. Find $x \in \mathbb{R}^m$ such that

$$F(x) := F_0 + \sum_{i=1}^{m} x_i F_i \succcurlyeq 0, \tag{1}$$

$$G(x) := G_0 + \sum_{i=1}^{m} x_i G_i \succcurlyeq 0, \tag{2}$$

$$\operatorname{rank} G(x) \leqslant r. \tag{3}$$

The problem data are the real symmetric matrices $F_i \in \mathcal{S}^{n_F}$ and $G_i \in \mathcal{S}^{n_G}$, and the rank bound r. Problem 1 consists of two LMI constraints, (1) and (2), and a rank constraint, (3). When $r = n_G$, constraint (3) is always satisfied and the problem reduces to a standard LMI feasibility problem. The more interesting case is when $r < n_G$. In this case the problem is non-convex.

Let $\mathcal{S}^n_+ = \{X \in \mathcal{S}^n \mid X \succcurlyeq 0\}$ and, for each integer s, let $\mathcal{S}^n_+(s) = \{X \in \mathcal{S}^n \mid X \succcurlyeq 0, \operatorname{rank}(X) = s\}$. Define $\mathcal{M}_r = \mathcal{S}^{n_F}_+ \times \cup_{s=0}^{r} \mathcal{S}^{n_G}_+(s) = \{(X, Y) \in \mathcal{S}^{n_F} \times \mathcal{S}^{n_G} \mid X \succcurlyeq 0,\ Y \succcurlyeq 0,\ \operatorname{rank}(Y) \leqslant r\}$ and $\mathcal{L} = \{(X, Y) \in \mathcal{S}^{n_F} \times \mathcal{S}^{n_G} \mid (X, Y) = (F(x), G(x)) \text{ for some } x \in \mathbb{R}^m\}$. Problem 1 can be stated in the following equivalent form.

Problem 2. Find $(X, Y) \in \mathcal{M}_r \cap \mathcal{L}$.

We will see that, for each s, $\mathcal{S}^n_+(s)$ is a manifold and hence that the rank constrained LMI problem is equivalent to finding a point in the intersection of an affine subspace and another set which is a finite union of manifolds. This structure will enable us to use the tangent and lift ideas that are discussed in the next section.

3. Tangent and lift

In this section we discuss the tangent and lift methodology and present an extension that can be applied to the rank constrained LMI problem.

Before proceeding with the main discussion, a brief note on projections is required. Let x be an element in a Hilbert space H and let C be a closed (possibly non-convex) subset of H. Any $c_0 \in C$ such that $\|x - c_0\| \leqslant \|x - c\|$ for all $c \in C$ will be called a *projection* of x onto C. In the cases of interest here, namely that H is a finite dimensional Hilbert space, there is always at least one such point for each x. If C is convex as well as closed then each x has exactly one such minimum distance point (Luenberger, 1969).

The tangent and lift method is a generalization of Newton's method and can be used to find a point in the intersection of an

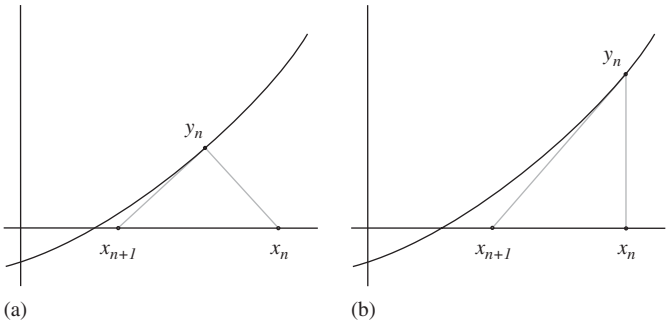


Fig. 1. Two different methods for finding a zero of a function: (a) Tangent and lift; (b) Newton's method.

affine subspace and a manifold. It originated in Chu (1992) and is based on a geometric interpretation of an algorithm appearing in Friedland, Nocedal, and Overton (1987).

Recall that Newton's method for finding a zero of a function $f : \mathbb{R} \to \mathbb{R}$ is iterative in nature and is given by the recursion $x_{n+1} = x_n - f(x_n)/f'(x_n)$. Geometrically speaking, x_{n+1} is the x-axis intercept of the line which is tangent to the graph of f at $(x_n, f(x_n))$. In tangent and lift, the role of the x-axis is replaced by an affine subspace and the role of the graph of f is replaced by a manifold. More precisely, the method works as follows. Let H be a real finite dimensional Hilbert space and suppose $\mathscr{L}$ is an affine subspace of H and that $\mathscr{M}$ is a submanifold of H. Given $x_n \in \mathscr{L}$, and assuming it is possible to calculate projections onto $\mathscr{M}$, let y_n be a projection of x_n onto $\mathscr{M}$. As $\mathscr{M}$ is a manifold, it has a tangent space T at the point y_n. T has a canonical representation as a linear subspace of H and $y_n + T$ can be thought of as an affine subspace of H that is tangent to the manifold at y_n. Assuming $y_n + T$ and $\mathscr{L}$ intersect uniquely, x_{n+1} is taken to be the intersection point of $y_n + T$ and $\mathscr{L}$. As $x_{n+1} \in \mathscr{L}$, the scheme can be iterated.

A graphical representation of the algorithm is given in Fig. 1(a). Here the Hilbert space H is $\mathbb{R}^2$, $\mathscr{L}$ is the x-axis, and $\mathscr{M}$ is the graph of a function $f : \mathbb{R} \to \mathbb{R}$. In this case, finding a point in $\mathscr{M} \cap \mathscr{L}$ is equivalent to finding a zero of f. Newton's method can also be employed to solve this problem and for purposes of comparison is also illustrated in Fig. 1.

For tangent and lift to work it must be possible to calculate projections onto $\mathscr{M}$. This step replaces the process of 'lifting' x to $(x, f(x))$ in Newton's method. In addition, at least for all points near a solution, each $y_n + T$ must intersect $\mathscr{L}$ uniquely. This essentially places a rather strong requirement on the dimensions of $\mathscr{L}$ and $\mathscr{M}$:

$$\dim \mathscr{L} + \dim \mathscr{M} = \dim H. \tag{4}$$

In Problem 2, $\mathscr{M}_r$ is not a manifold but rather a finite union of pairwise disjoint manifolds, see Section 4. This means that each point in $\mathscr{M}_r$ lies in a manifold with a well defined tangent space. However, these manifolds are of varying dimensions. Depending on $y \in \mathscr{M}_r$, it may therefore happen that $y + T$ does not intersect $\mathscr{L}$ uniquely. The intersection may be empty or it may contain more than one point. This may happen even arbitrarily close to a solution point.

In order to apply tangent and lift ideas to Problem 2, the approach must be extended to deal with these intersection issues. Our method of doing this is as follows. We consider all points in $\mathscr{L}$ that are of minimum distance to $y_n + T$ and from these points choose x_{n+1} to be the point closest to y_n. As we will see in Section 5, x_{n+1} can be found by solving a linearly constrained least squares problem. Numerical experiments demonstrate this methodology leads to a locally convergent algorithm which, though it not always the case, often exhibits local quadratic convergence.

Regarding rigorous convergence results, though our method is based on a generalization of the Newton method, analysis of convergence is complicated due to the non-smoothness of the constraints as well as the possibility of continua of solutions. We currently have only partial results and the development of a rigorous convergence theory presents a challenge for the future.

4. The geometry of rank constrained positive semidefinite matrices

Before proceeding to describe our algorithm for solving the rank constrained LMI problem in greater detail, in this section we collect together some geometric properties of rank constrained positive semidefinite matrices.

Theorem 3. *$\mathscr{S}^n_+(s)$ is a connected smooth manifold of dimension $\frac{1}{2}s(2n-s+1)$. The tangent space of $\mathscr{S}^n_+(s)$ at an element X is $T_X\mathscr{S}^n_+(s) = \{\Omega X + X\Omega^{\mathrm{T}} \mid \Omega \in \mathbb{R}^{n\times n}\}$.*

Corollary 4. *$\mathscr{M}_r$ is a finite union of manifolds.*

As the next theorem shows, after applying an appropriate transformation, $T_X\mathscr{S}^n_+(s)$ has a rather simple form.

Theorem 5. *Given $X \in \mathscr{S}^n_+(s)$, let $X = \Theta\bar{X}\Theta^{\mathrm{T}}$, $\bar{X} = \Lambda \oplus 0$, where $\Theta \in \mathbb{R}^{n\times n}$ is orthogonal and $\Lambda \in \mathscr{S}^s$ is a positive definite diagonal matrix. Then $\Theta^{\mathrm{T}} T_X\mathscr{S}^n_+(s)\Theta = T_{\bar{X}}\mathscr{S}^n_+(s)$ $= \left\{ \begin{bmatrix} \Omega_1 & \Omega_2^{\mathrm{T}} \\ \Omega_2 & 0 \end{bmatrix} \mid \Omega_1 \in \mathscr{S}^s,\ \Omega_2 \in \mathbb{R}^{(n-s)\times s} \right\}$.*

The following useful fact is obvious from Theorem 3.

Lemma 6. *$X \in T_X\mathscr{S}^n_+(s)$ for each $X \in \mathscr{S}^n_+(s)$.*

5. Algorithm

This section presents our algorithm for solving the rank constrained LMI problem. It contains a description of the algorithm at a conceptual level followed by details of the various components of the algorithm, including the required projections and initialization. The algorithm described here has been made into a freely available Matlab toolbox titled LMIRank (Orsi, 2005). LMIRank can be used either directly or via YALMIP (Löfberg, 2004).

In order to do projections, we need to define an appropriate Hilbert space. From now on $\mathscr{S}^n$ will be regarded as a Hilbert space with inner product $\langle A, B\rangle = \mathrm{tr}(AB) = \sum_{i,j} A_{ij}B_{ij}$.

The associated norm is the Frobenius norm $\|A\| = \langle A, A\rangle^{\frac{1}{2}}$. In addition, $\mathcal{S}^{n_F} \times \mathcal{S}^{n_G}$ will be regarded as a Hilbert space with the usual inner product for a space that is a product of Hilbert spaces.

We will have need to refer to the tangent space of $\mathcal{S}_+^{n_F}(s) \times \mathcal{S}_+^{n_G}(t)$ at a point (X, Y) as an affine subspace of $\mathcal{S}^{n_F} \times \mathcal{S}^{n_G}$: for $(X, Y) \in \mathcal{S}_+^{n_F}(s) \times \mathcal{S}_+^{n_G}(t)$, define $\mathcal{A}_{(X,Y)} = (X, Y) + T_{(X,Y)}(\mathcal{S}_+^{n_F}(s) \times \mathcal{S}_+^{n_G}(t))$. Lemma 6 implies that $(X, Y) \in T_{(X,Y)}(\mathcal{S}_+^{n_F}(s) \times \mathcal{S}_+^{n_G}(t))$. Hence, $\mathcal{A}_{(X,Y)} = T_{(X,Y)}(\mathcal{S}_+^{n_F}(s) \times \mathcal{S}_+^{n_G}(t))$ and $\mathcal{A}_{(X,Y)}$ is in fact a linear subspace and not just an affine subspace.

At a conceptual level, the algorithm is as follows.

Algorithm.

Problem Data. $F_0, \ldots, F_m \in \mathcal{S}^{n_F}$, $G_0, \ldots, G_m \in \mathcal{S}^{n_G}$, and $0 \leqslant r \leqslant n_G$.

Initialization. Either choose any $(X_1, Y_1) \in \mathcal{S}^{n_F} \times \mathcal{S}^{n_G}$, or use $(X_1, Y_1) = (F(x), G(x))$ where x is the solution of the semidefinite definite program (5).

repeat

(1) Project (X_1, Y_1) onto $\mathcal{M}_r$ to give a new point (X_2, Y_2).

(2) Define $\mathcal{B} = \{(X, Y) \in \mathcal{L} \mid \text{dist}((X, Y), \mathcal{A}_{(X_2,Y_2)}) = \text{dist}(\mathcal{L}, \mathcal{A}_{(X_2,Y_2)})\}$.

(3) $(X_3, Y_3) = \arg\min_{(X,Y)\in\mathcal{B}} \|(X, Y) - (X_2, Y_2)\|$.

(4) Set $(X_1, Y_1) = (X_3, Y_3)$.

until (X_1, Y_1) converges to a solution of Problem 2. (See Section 6.1 for a precise termination criteria.)

Here are some comments regarding the above algorithm. Step 1 is readily calculated via eigenvalue–eigenvector decompositions of X_1 and Y_1. This will be shown in Section 5.2 below. In Step 2, $\mathcal{B}$ is the set of points in $\mathcal{L}$ that are of minimum distance to $\mathcal{A}_{(X_2,Y_2)}$. Step 3 is the projection of (X_2, Y_2) onto $\mathcal{B}$. Note that as $\mathcal{L}$ and $\mathcal{A}_{(X_2,Y_2)}$ are closed affine subspaces, the distance between them is zero if and only if they intersect. Whether the sets intersect or not, $\mathcal{B}$ itself will always be either a single point or an affine subspace. In the case that $\mathcal{B}$ is a single point, Step 3 is trivial. In the case that $\mathcal{B}$ is an affine subspace, Step 3 is equivalent to solving a linearly constrained least squares problem. Details of how to solve this step are given in Section 5.3 below. Finally, note that each new (X_1, Y_1) is in $\mathcal{L}$ as $(X_1, Y_1) = (X_3, Y_3) \in \mathcal{B} \subset \mathcal{L}$. Hence, the termination criterion of the algorithm can be replaced by 'until $(X_1, Y_1) \in \mathcal{M}_r$'.

5.1. Initialization

There is no guarantee that the algorithm will converge from an arbitrary initial condition (X_1, Y_1). While a random choice for the initial condition does often work, an alternative choice is to use $(X_1, Y_1) = (F(x), G(x))$ where x is the solution of the following semidefinite programming (SDP) problem:

$$\min_{x\in\mathbb{R}^m} \text{tr}(G(x)) \quad \text{s.t. } F(x) \succcurlyeq 0, \ G(x) \succcurlyeq 0. \tag{5}$$

This is based on the heuristic that minimizing the trace of a matrix subject to LMI constraints often leads to a low rank solution. Nice insights into why trace minimization might be effective can be found in Fazel (2002).

As we will see in the results section, in some cases the solution of (5) will satisfy $\text{rank } G(x) \leqslant r$, in which case the overall problem is solved. In general, however, the solution of the SDP gives a singular matrix $G(x)$ which does not satisfy this rank constraint.

5.2. Projecting onto $\mathcal{M}_r$

Step 1 of the algorithm is the projection of a point (X_1, Y_1) onto the set $\mathcal{M}_r$. This projection is equivalent to componentwise projection of X_1 onto $\mathcal{S}_+^{n_F} = \cup_{s=0}^{n_F} \mathcal{S}_+^{n_F}(s)$ and Y_1 onto $\cup_{s=0}^{r} \mathcal{S}_+^{n_G}(s)$.

The projection of $X \in \mathcal{S}^n$ onto $\cup_{s=0}^{r} \mathcal{S}_+^n(s)$ is given by Theorem 7 below. More precisely, Theorem 7 gives *a* projection of X onto $\cup_{s=0}^{r} \mathcal{S}_+^n(s)$ as, for r strictly less than n, the set $\cup_{s=0}^{r} \mathcal{S}_+^n(s)$ is nonconvex and projections onto this set are not always guaranteed to be unique. While Theorem 7 is not new (see Mardia, 1978) our proof (see Orsi, Helmke, & Moore, 2006) is new.

Theorem 7. *Given $X \in \mathcal{S}^n$ and $0 \leqslant r \leqslant n$, let $X = \Theta \,\text{diag}(\lambda_1, \cdots, \lambda_n)\Theta^{\mathrm{T}}$ with $\lambda_1 \geqslant \cdots \geqslant \lambda_n$ and Θ a real orthogonal matrix. Define $P_r : \mathcal{S}^n \to \mathcal{S}^n$ by $P_r(X) = \Theta \,\text{diag}(\max\{\lambda_1, 0\}, \ldots, \max\{\lambda_r, 0\}, 0, \ldots, 0)\Theta^{\mathrm{T}}$. Then $P_r(X)$ is a best approximant in $\cup_{s=0}^{r} \mathcal{S}_+^n(s)$ to X in the Frobenius norm.*

5.3. Projecting onto $\mathcal{B}$

In this section, we will make use of the following notation. Given $X \in \mathbb{R}^{n\times n}$ and $0 \leqslant s \leqslant n$, let $X_s \in \mathbb{R}^{(n-s)\times(n-s)}$ denote the matrix consisting of the last $n - s$ rows and columns of X.

Theorem 8. *Suppose $(X, Y) \in \mathcal{M}_r$ and define $s = \text{rank}(X)$ and $t = \text{rank}(Y)$. Then X and Y have eigenvalue–eigenvector decompositions $X = V(\Lambda_X \oplus 0)V^{\mathrm{T}}$, $Y = W(\Lambda_Y \oplus 0)W^{\mathrm{T}}$, where $V \in \mathbb{R}^{n_F\times n_F}$ and $W \in \mathbb{R}^{n_G\times n_G}$ are orthogonal, and $\Lambda_X \in \mathcal{S}^s$ and $\Lambda_Y \in \mathcal{S}^t$ are positive definite diagonal matrices.*

Using the F_i's and G_i's of (1) *and* (2)*, and V and W, define $b \in \mathbb{R}^{(n_F-s)^2+(n_G-t)^2}$ and $B \in \mathbb{R}^{((n_F-s)^2+(n_G-t)^2)\times m}$ by*

$$b = \begin{bmatrix} \text{vec}((V^{\mathrm{T}}F_0V)_s) \\ \text{vec}((W^{\mathrm{T}}G_0W)_t) \end{bmatrix},$$

$$B = \begin{bmatrix} \text{vec}((V^{\mathrm{T}}F_1V)_s) & \ldots & \text{vec}((V^{\mathrm{T}}F_mV)_s) \\ \text{vec}((W^{\mathrm{T}}G_1W)_t) & \ldots & \text{vec}((W^{\mathrm{T}}G_mW)_t) \end{bmatrix}.$$

If $F(\cdot)$ and $G(\cdot)$ are the functions defined in (1) *and* (2)*, and $\|\cdot\|_2$ denotes the standard vector 2-norm, then the projection of (X, Y) onto $\mathcal{B}$ equals $(F(x), G(x))$ where x is a minimizing solution of*

$$\min_{x\in\mathbb{R}^m} \left\| \begin{bmatrix} \text{vec}(F_1) & \ldots & \text{vec}(F_m) \\ \text{vec}(G_1) & \ldots & \text{vec}(G_m) \end{bmatrix} x + \begin{bmatrix} \text{vec}(F_0 - X) \\ \text{vec}(G_0 - Y) \end{bmatrix} \right\|_2 \tag{6}$$

$$\text{s.t.} \quad B^{\mathrm{T}}Bx = -B^{\mathrm{T}}b. \tag{7}$$

Hence projecting onto $\mathscr{B}$ is equivalent to solving the linearly constrained least squares problem (6), (7). Such problems can be solved in a number of ways, see for example Lawson and Hanson (1995).

6. Numerical implementation issues

6.1. Convergence criteria

Let $\varepsilon_{\text{alg}} > 0$ be a user chosen algorithm tolerance. The convergence criteria is that the constraints (1)–(3) are 'satisfied to a tolerance of ε_{alg}', by which we mean the following conditions are met: $F(x) \succcurlyeq -\varepsilon_{\text{alg}} I$, $G(x) \succcurlyeq -\varepsilon_{\text{alg}} I$ and $G(x)$ has $n_G - r$ eigenvalues of absolute value $\leqslant \varepsilon_{\text{alg}}$. While choosing ε_{alg} small guarantees that the constraints (1)–(3) will be almost exactly satisfied, such a choice will lead to longer convergence times. The choice of tolerance ε_{alg} will be dictated by the problem being considered and it may be possible to choose a relatively large value. An example of this will be given in Section 7 when considering output feedback stabilization problems.

6.2. Additional LMI constraints and multiple rank constraints

The methods described in this paper can be readily modified to deal with additional LMI constraints. Indeed, it is even possible to have multiple rank constraints. In the rest of this section we briefly outline how the algorithm can be modified to incorporate these additions. These extensions are incorporated into our software package LMIRank (Orsi, 2005).

If there are q LMI constraints in total, when projecting onto '$\mathscr{M}_r$', q (rather than 2) individual projections must be carried out. As before, each projection is given by Theorem 7. Note that whether or not the ith LMI is rank constrained influences the ith (and only the ith) projection.

When projecting onto '$\mathscr{B}$', the only difference is that, in Theorem 8, rather than considering (X, Y), we now have to consider a q-tuple of symmetric matrices. As a result, each column of b and B in the theorem now consists of q (rather than 2) stacked vectors. Similarly, each column of the block matrix and block vector in (6) consists of q stacked vectors. Each of these terms are calculated in an analogous manner to the ones appearing in Theorem 8.

6.3. Linear programming inequality constraints

Linear programming inequalities constraints in x of the form

$$a^{\mathrm{T}}x + b \geqslant 0, \quad a \in \mathbb{R}^m, \quad b \in \mathbb{R}$$

are just 1×1 LMIs and hence, by the prior subsection, can also be readily incorporated.

7. Numerical experiments

This section contains some numerical experiments. Algorithm performance is investigated using both randomly generated problems and by applying the algorithm to a particular output feedback problem.

Table 1
Experiments for random F and G with $n_F = 10$, $n_G = 10$ and $r = 5$

m	Iterations					i	T
	1	2–10	11–20	21–1000	NC		
10	965	35	0	0	0	1.1	0.16
20	333	448	84	112	23	21	0.36
30	279	559	70	71	21	21	0.48
40	756	214	19	9	2	3.2	0.46
50	967	26	6	1	0	1.5	0.54

i denotes the average number of iterations and T denotes average CPU time in seconds. i and T do not include the problems that had not converged after 1000 iterations. The number of such problems for each m is given in the 'NC' or 'non-convergence after 1000 iterations' column. Tolerance $\varepsilon_{\text{alg}} = 10^{-12}$.

All computational results were obtained using a 3 GHz Pentium 4 machine. Our algorithm was coded using Matlab 7.0. For each problem, the initial condition was found by solving the semidefinite programming problem (5) using SeDuMi (Sturm, 1999).

7.1. Random problems

All results in this section are for randomly generated problems. Each problem is generated as follows. Let $\mathscr{N}(0, 1)$ denote the normal distribution with zero mean and variance 1. Each entry of the matrices $F_1, \ldots, F_m$ and $G_1, \ldots, G_m$ is drawn from $\mathscr{N}(0, 1)$. To ensure feasibility, F_0 and G_0 are set to $F_0 = V_F D_F V_F^{\mathrm{T}} - \sum_{i=1}^m \xi_i F_i$ and $G_0 = V_G D_G V_G^{\mathrm{T}} - \sum_{i=1}^m \xi_i G_i$, where each ξ_i is drawn from $\mathscr{N}(0, 1)$; V_F and V_G are randomly generated orthogonal matrices; and D_F and D_G are randomly generated diagonal matrices: each diagonal entry in D_F is drawn from $\mathscr{N}(0, 1)$ and set to zero if it is negative, while r diagonal entries in D_G are drawn from the uniform distribution on the interval [0, 1] and the others set to zero.

For the problems in this section, the algorithm tolerance was set to $\varepsilon_{\text{alg}} = 10^{-12}$. (For the problems considered, typical nonzero eigenvalues have magnitudes between 10^1 and 10^{-2}.)

Table 1 contains results for $n_F = 10$, $n_G = 10$, $r = 5$ and various values of m. For each value of m in the table, the algorithm is given 1000 random problems to solve. Listed are a distribution of the number of iterations taken for the algorithm to converge, the average number of iterations, and the average CPU time. Iteration 1 is the initialization step using the trace minimization heuristic.

For $m = 10$ and 50, solutions for all 1000 problems were found. In both these cases the trace minimization heuristic was very effective, finding solutions for almost all the problems. Most of the few problems that were not solved in this first iteration, were solved using a small number of additional iterations. For both $m = 20$ and 30, the trace minimization heuristic was no longer quite as successful though it did still manage to find a solution in about 30% of cases. Overall, 95% of problems were solved in 20 iterations or less while less than 1% had not converged after 1000 iterations. Average solution times were

very small and tended to increase with m. (For some results for larger problems, see Orsi et al. (2006).)

The results indicate that on average the problems that are easiest to solve are those with either a rather small number of variables or those problems with a rather large number of variables. This situation is rather puzzling. However, as we will now explain, theoretical rank bound results do suggest why at least problems with a rather large number of variables may be easier to solve.

We will use the notation $\triangle_k$ to denote the kth triangular number, $\triangle_k := (k+1)k/2$. Consider a SDP of the form (5) where the cost is replaced by a general linear cost $c^{\mathrm{T}}x$. It follows from the results in Alizadeh, Haeberly, and Overton (1997) that for a generic choice of c, F_i's and G_i's, a solution x of such a problem satisfies

$$\triangle_{\operatorname{rank} F(x)} + \triangle_{\operatorname{rank} G(x)} \leqslant \triangle_{n_F} + \triangle_{n_G} - m. \tag{8}$$

Hence, for a given rank bound r, if m is large enough, the solution will satisfy $\operatorname{rank} G(x) \leqslant r$. The rank bound (8) is certainly interesting however its practical value as a means of ensuring low rank may be limited: for our largest value of m, $m = 50$, Eq. (8) only guarantees $\operatorname{rank} G(x) \leqslant 10$ (in the worst case scenario that $\operatorname{rank} F(x) = 0$) and hence does not even guarantee that $G(x)$ will not have full rank.

7.2. Reduced order output feedback

Consider a continuous time, linear time invariant (LTI) system

$$\dot{x} = Ax + Bu, \quad y = Cx, \tag{9}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control, and $y \in \mathbb{R}^p$ is the output.

A dynamic output feedback controller of order n_c, $0 \leqslant n_c \leqslant n$, will be understood to be a controller of the form $\begin{bmatrix} \dot{x}_c \\ u \end{bmatrix} = K \begin{bmatrix} x_c \\ y \end{bmatrix}$ where $K \in \mathbb{R}^{(n_c+m)\times(n_c+p)}$ is a constant matrix and $x_c \in \mathbb{R}^{n_c}$.

The problem that interests us in this section is the following reduced order output feedback stabilization problem.

Problem 9. Given a system (9) and a scalar $\alpha > 0$, find a dynamic output feedback controller of order $\leqslant n_c$ that places the closed loop poles of the system in the set

$$\{z \in \mathbb{C} \mid \operatorname{Re}(z) \leqslant -\alpha\}. \tag{10}$$

Recall that a system with its poles in (10) is said to have stability degree (of at least) α.

Define $\tilde{A} = \begin{bmatrix} A & 0 \\ 0 & 0_{n_c} \end{bmatrix}$, $\tilde{B} = \begin{bmatrix} 0 & B \\ I_{n_c} & 0 \end{bmatrix}$, and $\tilde{C} = \begin{bmatrix} 0 & I_{n_c} \\ C & 0 \end{bmatrix}$. As is well known, K is an order n_c solution of Problem 9 if and only if the augmented closed loop system matrix $\tilde{A} + \tilde{B}K\tilde{C}$ has its eigenvalues in (10). In addition, Problem 9 is solvable if and only if Problem 10, given below, is solvable (see for example Grigoriadis & Beran, 1999).

Problem 10. Given a system (9) and a scalar $\alpha > 0$, find $X, Y \in \mathscr{S}^n$ such that

$$-B^{\perp}(AX + XA^{\mathrm{T}} + 2\alpha X)B^{\perp\mathrm{T}} \succcurlyeq 0, \tag{11}$$

$$-C^{\mathrm{T}\perp}(YA + A^{\mathrm{T}}Y + 2\alpha Y)C^{\mathrm{T}\perp\mathrm{T}} \succcurlyeq 0, \tag{12}$$

$$\begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succcurlyeq 0, \tag{13}$$

$$\operatorname{rank} \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \leqslant n + n_c. \tag{14}$$

Here, $B^{\perp}$ is a matrix of maximal rank such that its rows are orthonormal and $B^{\perp}B = 0$. Similar comments apply for $C^{\mathrm{T}\perp}$.

A solution of Problem 10 can be used to construct a solution of Problem 9 (and vice versa). As discussed in Section 6.1, our algorithm computes solutions to a user specified tolerance $\varepsilon_{\mathrm{alg}}$. Hence, an algorithm computed solution to Problem 10 will not in general satisfy the constraints exactly. We take this fact into account in our method of controller synthesis. The first step of our method is to use the algorithm to solve the following perturbed problem.

Problem 11. Given a system (9), a scalar $\alpha > 0$, and a tolerance $\varepsilon > 0$, find $X, Y \in \mathscr{S}^n$ such that

$$\begin{aligned}
&-B^{\perp}(AX + XA^{\mathrm{T}} + 2\alpha X)B^{\perp\mathrm{T}} - \varepsilon I \succcurlyeq 0,\\
&-C^{\mathrm{T}\perp}(YA + A^{\mathrm{T}}Y + 2\alpha Y)C^{\mathrm{T}\perp\mathrm{T}} - \varepsilon I \succcurlyeq 0,\\
&\begin{bmatrix} X & I \\ I & Y \end{bmatrix} - \varepsilon I \succcurlyeq 0,\\
&\operatorname{rank}\left(\begin{bmatrix} X & I \\ I & Y \end{bmatrix} - \varepsilon I\right) \leqslant n + n_c.
\end{aligned}$$

By choosing $\varepsilon_{\mathrm{alg}}$ equal to the ε of Problem 11, an algorithm calculated solution to Problem 11 will satisfy (11)–(13). In addition, (14) will be satisfied to a tolerance of 2ε, that is, at least $2n - (n + n_c)$ eigenvalues of the matrix in (14) will have magnitude 2ε or less.

By the Schur complement result, (13) holds if and only if $Y \succ 0$ (Y is positive definite) and $X - Y^{-1} \succcurlyeq 0$. If $X - Y^{-1}$ has eigenvalue–eigenvector decomposition $X - Y^{-1} = V \operatorname{diag}(\lambda_1, \cdots, \lambda_n) V^{\mathrm{T}}$, with $\lambda_1 \geqslant \ldots \geqslant \lambda_n$, define $R = V(:, 1 : n_c)\operatorname{diag}(\lambda_1^{1/2}, \ldots, \lambda_{n_c}^{1/2}$ and $\tilde{X} = \left[\begin{smallmatrix} X & R \\ R^{\mathrm{T}} & I \end{smallmatrix}\right]$. Note that $\tilde{X} \succ 0$. The output feedback matrix K is reconstructed via the following SDP,

$$\begin{aligned}
\max_{\gamma\in\mathbb{R}, K} \quad & \gamma \\
\text{s.t.} \quad & (\tilde{A} + \tilde{B}K\tilde{C})\tilde{X} + \tilde{X}(\tilde{A} + \tilde{B}K\tilde{C})^{\mathrm{T}} + 2\gamma\tilde{X} \preceq 0.
\end{aligned} \tag{15}$$

See Orsi et al. (2006) for further comments on this approach.

We now consider a particular reduced order output feedback problem from Beran and Grigoriadis (1996) (see also Grigoriadis & Beran, 1999; Grigoriadis & Skelton, 1996; Skelton et al., 1998, Chapter 10;). The system considered is a

Table 2
Results for the two-mass-spring system

α	$\varepsilon = 10^{-4}$			$\varepsilon = 10^{-9}$		
	$\hat{\alpha}$	i	T	$\hat{\alpha}$	i	T
0.2	0.20	59	0.55	0.21	195	1.4
0.42	0.42	644	4.2	0.42	1536	9.5
0.46	0.46	1187	8.0	0.46	2846	19

α denotes the desired stability degree, $\hat{\alpha}$ the stability degree achieved, i the number of iterations, and T the CPU time in seconds.

two-mass-spring system with state space representation given by

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}}.$$

Given $\alpha > 0$, we wish to find an order 2 dynamic controller that places the closed loop poles in (10).

The problem was solved for two of the same values of α given in Beran and Grigoriadis (1996), $\alpha = 0.2$ and 0.42, and also an additional value, $\alpha = 0.46$. For each value of α, the algorithm was applied to Problem 11 with tolerances $\varepsilon = 10^{-4}$ and 10^{-9}. In each case the controller matrix K was constructed via (15). The results are listed in Table 2.

The first main point to note from these results is that it took more than twice as many iterations and more than twice as long to solve the problems using $\varepsilon = 10^{-9}$ compared to $\varepsilon = 10^{-4}$. Hence, taking relatively large values of ε such as $\varepsilon = 10^{-4}$ may in general be a good solution strategy for these types of problems. The second main point to note is that, for both values of ε, convergence time increases with α. Hence, speed of convergence seems to be influenced by the size of the feasible set.

While α=0.46 was the best stability degree that we were able to achieve, it turns out this value is still not the best possible. The second order controller

$$K(s) = \frac{(43/5)s^2 - (54\sqrt{15}/125)s - (27/125)}{s^2 + (6\sqrt{15}/5)s + 7} \tag{16}$$

achieves a stability degree of $\alpha = \sqrt{15}/5 \approx 0.77$. This controller can be found by considering system and controller transfer functions and requiring that the denominator of the closed loop transfer function equal the polynomial $(s+\alpha)^6$. For comparison purposes, we also tried the cone complementarity linearization algorithm of El Ghaoui et al. (1997) on the same problem. Our experience is that this algorithm works quite well in general though for this particular problem the greatest stability degree we were able to achieve using this algorithm was $\alpha = 0.20$.

8. Conclusions

In this paper we have presented an algorithm for solving the rank constrained LMI problem, as well as more general LMI problems such as those with multiple rank constraints. Like all other algorithms that attempt to solve the rank constrained LMI problem, convergence from an arbitrary initial condition is not guaranteed. Though the convergence properties of the algorithm are not yet completely understood, as demonstrated by the experiments, the algorithm can be quite effective.

Acknowledgments

The authors thank the reviewers for their comments, in particular for pointing out material that lead to the rank bound discussion in Section 7.1, and for pointing out the controller (16). The authors also thank Johan Löfberg for helpful suggestions that greatly improved the functionality of the LMIRank code.

The first and third authors acknowledge the support of the Australian Research Council through Grants DP0450539 and A00105829. The last two authors were partially supported by DAAD project PPP Australia/Germany D0243869. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

Alizadeh, F., Haeberly, J. P., & Overton, M. (1997). Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming Series B*, *77*, 111–128.

Beran, E., & Grigoriadis, K. (1996). A combined alternating projections and semidefinite programming algorithm for low-order control design. In *Proceedings IFAC 96*, (Vol. C, pp. 85–90). San Franscisco.

Boyd, S., & Vandenberghe, L. (1997). Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization. In: A. Paulraj, V. Roychowdhury, & C. Schaper (Eds.), *Communications, computation, control and signal processing—a tribute to Thomas Kailath* New York: Kluwer Academic Publishers.

Chu, M. T. (1992). Numerical methods for inverse singular value problems. *SIAM Journal of Numerical Analysis*, *29*(3), 885–903.

El Ghaoui, L., & Gahinet, P. (1993). Rank minimization under LMI constraints: A framework for output-feedback problems. In *Proceedings European control conference*, 1993.

El Ghaoui, L., Oustry, F., & Ait Rami, M. (1997). A cone complementarity linearization algorithm for static output-feedback and related problem. *IEEE Transactions on Automatic Control*, *42*(8), 1171–1176.

Fares, B., Apkarian, P., & Noll, D. (2001). An augmented Lagrangian method for a class of LMI-constrained problems in robust control theory. *International Journal of Control*, *74*(4), 348–360.

Fares, B., Noll, D., & Apkarian, P. (2002). Robust control via sequential semidefinite programming. *SIAM Journal of Control Optim.*, *40*(6), 1791–1820.

Fazel, M. (2002). *Matrix rank minimization with applications*. Ph.D. thesis, Stanford University, Stanford, CA.

Friedland, S., Nocedal, J., & Overton, M. L. (1987). The formulation and analysis of numerical methods for inverse eigenvalue problems. *SIAM Journal of Numerical Analysis*, *24*(3), 634–667.

Goh, K. C., Safonov, M. G., & Ly, J. H. (1996). Robust synthesis via bilinear matrix inequalities. *International Journal of Robust and Nonlinear Control*, *6*(9–10), 1079–1095.

Grigoriadis, K. M., & Beran, E. B. (1999). Alternating projection algorithms for linear matrix inequalities problems with rank constraints. In: L. El Ghaoui, & S. -I. Niculescu (Eds.), *Advances on linear matrix inequality methods in control* (pp. 251–267). Philadelphia: SIAM.

Grigoriadis, K. M., & Skelton, R. E. (1996). Low-order control design for LMI problems using alternating projection methods. *Automatica*, *32*(8), 1117–1125.

Lawson, C. L., & Hanson, R. J. (1995). *Solving least squares problems*. In *Classics in applied mathematics* (Vol. 15). Philadelphia: SIAM.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD conference*, Taipei, Taiwan. Available from ⟨http://control.ee.ethz.ch/∼joloef/yalmip.php⟩.

Luenberger, D. (1969). *Optimization by vector space methods*. New York: Wiley.

Mardia, K. V. (1978). Some properties of classical multi-dimensional scaling. *Communications in Statistics—Theory and Methods*, *A7*(8), 1233–1241.

Mesbahi, M., Safonov, M. G., & Papavassilopoulos, G. P. (1999). Bilinearity and complementarity in robust control. In: L. El Ghaoui, & S. -I. Niculescu (Eds.), *Advances on linear matrix inequality methods in control* (pp. 269–292). Philadelphia: SIAM.

Nesterov, Y., & Nemirovskii, A. (1994). *Interior-point polynomial algorithms in convex programming*. Philadelphia: SIAM.

Orsi, R. (2005). *LMIRank*: software for rank constrained LMI problems. ⟨http://rsise.anu.edu.au/∼robert/lmirank/⟩.

Orsi, R., Helmke, U., & Moore, J. B. (2006). A Newton-like method for solving rank constrained linear matrix inequalities. Ext. vers., ⟨http://users.rsise.anu.edu.au/∼robert/publications⟩.

Pare, T. E. (2000). *Analysis and control of nonlinear systems*. Ph.D. thesis, Stanford University, Stanford, CA.

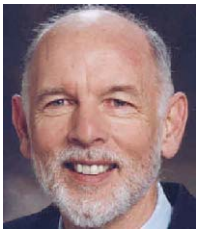
Skelton, R. E., Iwasaki, T., & Grigoriadis, K. M. (1998). *A unified algebraic approach to linear control design*. London: Taylor & Francis.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11–12*, 625–653 (Special issue on Interior Point Methods (CD supplement with software)).

Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, *38*(1), 49–95.

Robert Orsi received his Ph.D. in 2000 from the University of Melbourne, Australia. Since then he has held positions with Voyan Technology, the Australian National University, and National ICT Australia. Currently, since July 2004, he is the recipient of an Australian Research Council Australian Postdoctoral Fellowship, which he has taken up at the Australian National University. His current research interests include optimization methods for engineering problems, convex optimization, rank constrained LMIs, and inverse eigenvalue problems.

U. Helmke (Ph.D. Bremen 1983, Habilitation Regensburg 1991) is full professor and chair in mathematics at the University of Wuerzburg. His research interests are in dynamical systems and control theory, with special emphasis on computational dynamics. He is co-author of over 130 research publications, including the graduate level textbook *Optimization and Dynamical Systems*, Springer-Publ., with J. B. Moore.



John B. Moore received his doctorate in Electrical Engineering from the University of Santa Clara, California, in 1967. He was appointed full Professor (personal chair) at the University of Newcastle, Australia, in 1973. Since 1982 he has been with the Australian National University. Currently he is also a researcher in the Systems Engineering and Complex Systems Program within National ICT Australia. Professor Moore is a Fellow of the IEEE, the Australian Academy of Technological Sciences, and the Australian Academy of Science.