A Finite Step Projective Algorithm for Solving Linear Matrix Inequalities

Robert Orsi[*] Mustapha Ait Rami[†] John B. Moore[‡]

Abstract

This paper presents an algorithm for finding feasible solutions of linear matrix inequalities. The algorithm is based on the method of alternating projections (MAP), a classical method for solving convex feasibility problems. Unlike MAP, which is an iterative method that converges asymptotically to a feasible point, the algorithm converges after a finite number of steps. The key computational component of the algorithm is an eigenvalue-eigenvector decomposition which is carried out at each iteration. Computational results for the algorithm are presented and comparisons are made with existing algorithms.

1 Introduction

The *linear matrix inequality* (LMI) problem is to find $x \in \mathbb{R}^m$ such that

$$F(x) := F_0 + x_1 F_1 + \cdots + x_m F_m > 0. \tag{1}$$

Here $F_i \in \mathbb{R}^{n\times n}$, $i = 0, 1, \ldots, m$, are given real symmetric matrices and > 0 stands for positive definite. LMIs are by now well known, having numerous applications in systems and control theory [4] (see also [3]).

In this paper we present a new algorithm for solving the LMI problem. The algorithm is iterative in nature and is based on the method of alternating projections (MAP), a classical method for finding a point in the intersection of a finite number of closed convex sets. Unlike MAP, which is an iterative method that in general only converges asymptotically [5], the algorithm converges to a solution in a finite number of steps.

The work in this paper relies on the ideas presented in [1]. In that paper a finite step method is given for solving problems involving finding a point in the intersection of two convex cones[1]. Specifically, given closed convex cones C_1 and C_2 in a Hilbert space H, a finite step method is given for finding a point in the intersection of the interior of C_1 and the closed set C_2. The work in this paper can be considered a specialization of the ideas in [1] to the LMI problem.

As well as giving the theoretical underpinnings of our algorithm, the paper contains computational results showing that, subject to constraints on how large m is in comparison to n, the algorithm compares very favorably with existing algorithms used for solving the LMI problem.

The key computational component of the algorithm is an eigenvalue-eigenvector decomposition which is carried out at each iteration. The algorithm also contains a single matrix inversion.

The paper is structured as follows. The remainder of this section lists some notation which is used in the rest of the paper. Section 2 introduces the MAP algorithm and related results. The algorithm is described in Sections 3. This section also contains a pseudo-code representation of the algorithm. Section 4 contains some computational results, including comparisons to other algorithms. Section 5 examines the effect of a particular algorithm parameter on performance. The paper ends with some concluding remarks.

Notation. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}_+$ is the set of non-negative real numbers. $\mathbb{R}_{++}$ is the set of positive real numbers. S^n is the set of real symmetric $n\times n$ matrices. S^n_+ is the set of positive semi-definite real symmetric $n \times n$ matrices. S^n_{++} is the set of positive definite real symmetric $n \times n$ matrices. $\overset{\circ}{C}$ denotes the interior of the set C. A^T denotes the transpose of a matrix A. $\mathrm{Tr}(A)$ denotes the sum of the diagonal elements of a square matrix A. $\mathrm{diag}(v)$ for $v \in \mathbb{R}^n$ denotes the $n \times n$ diagonal matrix whose i'th diagonal term is v_i.

[*]National ICT Australia, C/- Research School of Information Sciences and Engineering, The Australian National University, Canberra ACT, 0200, Australia. Email: robert.orsi@nicta.com.au

[†]Mathematisches Institut, Universität Würzburg, Würzburg, Germany. Email: rami@analysis.mathematik.uni-wuerzburg.de

[‡]Department of Systems Engineering, Research School of Information Sciences and Engineering, The Australian National University, Canberra ACT, 0200, Australia. Email: john.moore@anu.edu.au

[1]A subset C of a real vector space H is a *cone* if for each $x \in C$ and scalar $\theta > 0$, it follows that $\theta x \in C$.

2 The Method of Alternating Projections and the Conic Feasibility Problem

This section introduces the MAP algorithm and the key ideas from [1]. Also introduced are relaxed projections.

Let H be a Hilbert space with inner product $\langle \cdot\, , \cdot \rangle$ and corresponding norm $|| \cdot ||$. If $x \in H$ and C is a closed convex subset of H, there exists a unique element in C that is closest to x [6].

Theorem 1 *Let x be an element in a Hilbert space H and let C be a closed convex subset of H. Then there exists a unique $c_0 \in C$ such that $||x - c_0|| \leq ||x - c||$ for all $c \in C$.*

Given $x \in H$ and a closed convex subset C, the c_0 of Theorem 1 is called the *projection* of x onto C. As this projection is unique for each $x \in H$, we can define the projection operator $P_C : H \to H$ which takes each x to its projection on C.

We now state a classical method for finding a point in the intersection of a finite number of closed convex subsets, the method of alternating projections [5]. The method works by successively projecting onto the convex subsets in question and is guaranteed to asymptotically converge to a point in their intersection.

Theorem 2 (MAP) *Let $C_1, \ldots, C_N$ be closed convex sets in a real finite dimensional Hilbert space H. If $\bigcap_{i=1}^N C_i$ is nonempty, then starting from an arbitrary initial value, the following sequence*

$$x_{i+1} = P_{C_{\phi(i)}}(x_i), \text{ where } \phi(i) = (i \bmod N) + 1,$$

converges to an element in $\bigcap_{i=1}^N C_i$.

We remark that the usefulness of MAP for finding a point in the intersection of a number of convex sets is dependent on being able to compute the projections P_{C_i}.

In order to solve the LMI problem via the MAP algorithm, we will reformulate the problem into one of finding an element in the intersection of two convex cones. Specifically, we will show that it is equivalent to the following conic feasibility problem:

$$\text{Find } \; x \; \in \overset{\circ}{C_1} \cap C_2, \tag{2}$$

where C_1 and C_2 are closed convex cones in a Hilbert space H.

As pointed out in [1], problem (2) is equivalent to the problem:

$$\text{Find } \; x \; \in \; (C_1 + e) \cap C_2, \tag{3}$$

where e is any fixed element in the interior of C_1. ($C_1 + e$ denotes the set $\{c + e \mid c \in C_1\}$.) Their equivalence follows from the fact that [1]:

1. $C_1 + e \subset \overset{\circ}{C_1}$,
2. For any $y \in H$ and $x \in \overset{\circ}{C_1}$ there exists $\alpha > 0$ such that $\alpha x - y \in C_1$.

Property 1 implies that if x is a solution of (3) then it is also a solution of (2). Alternatively, property 2 with $y = e$ implies that if x is a solution of (2) then $\alpha x \in (C_1 + e) \cap C_2$ for some $\alpha > 0$.

Problem (2) can be solved using the following finite step algorithm [1] which works by applying MAP to the closed convex sets $(C_1 + e)$ and C_2.

Theorem 3 *Let C_1 and C_2 be closed convex cones in a real finite dimensional Hilbert space H and suppose $\overset{\circ}{C_1} \cap C_2$ is non-empty. Then for any initial condition $x_0 \in H$ and $e \in \overset{\circ}{C_1}$, the sequence*

$$\begin{aligned} x_1 &= P_{C_1+e}(x_0) \\ x_2 &= P_{C_2}(x_1) \\ &\vdots \\ x_{2k-1} &= P_{C_1+e}(x_{2k-2}) \\ x_{2k} &= P_{C_2}(x_{2k-1}) \\ &\vdots \end{aligned}$$

converges in a finite number of steps to a point in $\overset{\circ}{C_1} \cap C_2$.

Theorem 2 guarantees that the sequence converges asymptotically to $(C_1 + e) \cap C_2$. This, the fact that $C_1 + e \subset \overset{\circ}{C_1}$, and the fact that $x_{2k} \in C_2$ for all k, implies that, for some k sufficiently large, $x_{2k} \in \overset{\circ}{C_1} \cap C_2$.

2.1 Improving Convergence Rates: Relaxed Projections

We now introduce a modification to the MAP algorithm which can improve its speed of convergence. The relevant theory is presented in this subsection and the positive effect on speed of convergence for the algorithm developed in this paper is demonstrated in the computational results of Section 4.

Given a Hilbert space H, let Id denote the identity operator on H, $Id(x) = x$ for all $x \in H$. We have the following generalization of the MAP algorithm.

Theorem 4 *Let $C_1, \ldots, C_N$ be closed convex sets in a real finite dimensional Hilbert space H. Given constants $t_1, \ldots, t_N$ in the interval $(0, 2)$, for $i = 1, \ldots, N$, define the operators*

$$R_i = (1 - t_i)Id + t_i P_{C_i}.$$

If $\bigcap_{i=1}^{N} C_i$ is nonempty, then starting from an arbitrary initial value, the following sequence

$$x_{i+1} = R_{\phi(i)}(x_i), \text{ where } \phi(i) = (i \bmod N) + 1,$$

converges to an element in $\bigcap_{i=1}^{N} C_i$.

For a proof of this result, see for example [10] or [2]. The R_i operators in the theorem are termed *relaxed projection operators*. Note that if the t_i's are all 1 then Theorem 4 reduces to Theorem 2.

Theorem 4 implies that the projection operator P_{C_1+e} of Theorem 3 can be replaced by a relaxed projection operator. This gives what we term a relaxed version of Theorem 3. Note that aside from invoking Theorem 4 rather than Theorem 2, the proof of the relaxed theorem is the same as that of the original theorem. The key point is that by relaxing only P_{C_1+e} and not P_{C_2}, x_{2k} remains an element of C_2 for all k.

3 Algorithm

In this section we show that the LMI problem is equivalent to a conic feasibility problem and derive an algorithm for solving the problem based on the results of the previous section. We will refer to the algorithm as LMI-Feas.

3.1 Reformulation of the LMI Problem

This subsection contains a reformulation of the LMI problem as a conic feasibility problem.

By introducing the slack variable $S \in S^n$ and the scalar variable $x_0 \in \mathbb{R}_{++}$, the LMI problem is equivalent to the following:

$$\begin{array}{ll} \text{Find} & (x_0, x, S) \in \mathbb{R}_{++} \times \mathbb{R}^m \times S^n_{++} \\ \text{such that} & x_0 F_0 + x_1 F_1 + \ldots + x_m F_m - S = 0. \end{array} \tag{4}$$

The equivalence of these problems can be seen as follows. If x satisfies (1), then $(1, x, F_0 + x_1 F_1 + \ldots + x_m F_m)$ is a solution of (4). Alternatively, if (x_0, x, S) is a solution of (4), then x/x_0 satisfies (1).

Problem (4) is a conic feasibility problem with the same form as problem (2). Indeed let H be the Hilbert space

$$H = \mathbb{R} \times \mathbb{R}^m \times S^n$$

with inner product

$$\langle (x_0, x, S), (y_0, y, T) \rangle = x_0 y_0 + x^T y + \mathrm{Tr}(ST). \tag{5}$$

Let

$$K = \mathbb{R}_+ \times \mathbb{R}^m \times S^n_+$$

and

$$\begin{aligned} L = \{ &(x_0, x, S) \in H \mid \\ &x_0 F_0 + x_1 F_1 + \ldots + x_m F_m - S = 0 \}. \end{aligned} \tag{6}$$

Both K and L are closed convex cones in H, indeed L is even a linear subspace of H.

The interior of K is $\mathbb{R}_{++} \times \mathbb{R}^m \times S^n_{++}$ and hence our original LMI problem is equivalent to:

$$\text{Find } (x_0, x, S) \in \overset{\circ}{K} \cap L. \tag{7}$$

3.2 Calculation of Projections

In order to solve problem (7) via Theorem 3, all that remains is to determine P_{K+e} and P_L, which we do now.

We start with P_{K+e}. The particular e we will use is $e = (\rho, 0, \rho I)$. Here $\rho > 0$ can be any positive constant[2] and I denotes the $n \times n$ identity matrix. Note, e is an element of the interior of K. For this choice of e, projection of (x_0, x, S) onto $K + e$ is equivalent to componentwise projection of x_0 onto $\mathbb{R}_+ + \rho$, x onto $\mathbb{R}^m$, and S onto $S^n_+ + \rho I$. That is, $P_{K+e}(x_0, x, S) = (P_{\mathbb{R}_+ + \rho}(x_0), P_{\mathbb{R}^m}(x), P_{S^n_+ + \rho I}(S))$. Referring back to Theorem 1, it is easily verified that $P_{\mathbb{R}_+ + \rho}(x_0) = \max\{\rho, x_0\}$ and $P_{\mathbb{R}^m}(x) = x$.

The following lemma from [1] gives $P_{S^n_+}(S)$.

Lemma 5 *The projection of S onto S^n_+ is calculated as follows. Let $S = VDV^T$ be an eigenvalue-eigenvector decomposition of S with $D = \mathrm{diag}(d_1, \ldots, d_n)$. If $\bar{D} = \mathrm{diag}(\max\{0, d_1\}, \ldots, \max\{0, d_n\})$, then $P_{S^n_+}(S) = V\bar{D}V^T$.*

Calculation of $P_{S^n_+ + \rho I}$ is aided by the following result [1].

Lemma 6 *Given a closed convex subset C of a Hilbert space H, and any $y \in H$, then the projection onto $C + y$ is given by $P_{C+y}(x) = P_C(x - y) + y$, for all $x \in H$.*

Combining Lemmas 5 and 6, if $S = VDV^T$ is an eigenvalue-eigenvector decomposition of S with $D = \mathrm{diag}(d_1, \ldots, d_n)$, then, defining $\bar{D} = \mathrm{diag}(\max\{\rho, d_1\}, \ldots, \max\{\rho, d_n\})$, one can show that $P_{S^n_+ + \rho I}(S) = V\bar{D}V^T$.

To calculate P_L, we will need the following result [6].

Lemma 7 *Suppose H is a Hilbert space and that $\{y_1, \ldots, y_p\}$ is a set of linearly independent vectors in H. If $L = \{y \mid \langle y, y_i \rangle = 0,\ i = 1, \ldots, p\}$, then the projection of $x \in H$ onto L is given by*

$$P_L(x) = x - \sum_{i=1}^{p} \alpha_i y_i,$$

[2]The constant ρ can be thought of as an algorithm parameter. Section 5 contains a discussion of how the choice of ρ effects the speed of convergence of the algorithm.

where $\alpha \in \mathbb{R}^p$ is given by $\alpha = G^{-1}(\langle x, y_1\rangle, \ldots, \langle x, y_p\rangle)^T$ and $G \in \mathbb{R}^{p\times p}$ is given by $G_{ij} = \langle y_i, y_j\rangle$.

The linear constraint that characterizes the elements of L, see (6), can be re-written as $p = n(n+1)/2$ inner product constraints:

$$L = \{(x_0, x, S) \in H \mid \langle (x_0, x, S), y_{ij}\rangle = 0, \quad i = 1, \ldots, n, \; j = i, \ldots, n\} \tag{8}$$

where $y_{ij} \in H$ is given by

$$y_{ij} = ((F_0)_{ij}, ((F_1)_{ij}, \ldots, (F_m)_{ij})^T, -E_{ij}) \tag{9}$$

with $E_{ij} \in S^n$ defined as

$$(E_{ij})_{kl} = \begin{cases} 1, & \text{if } i = j = k = l, \\ 1/2, & \text{if } i \neq j,\ k = i,\ l = j, \\ 1/2, & \text{if } i \neq j,\ k = j,\ l = i, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

P_L can now be calculated by applying Lemma 7 to the characterization of L just given[3]. P_L is given in the pseudo-code in next subsection. (The detailed calculations of P_L are straightforward though tedious and due to space limitations have been omitted.)

We now have all the ingredients needed for our algorithm, which is presented in the next subsection.

3.3 An Algorithm for Solving the LMI Problem: LMI-Feas

This subsection contains our algorithm for finding a x that satisfies (1). In the algorithm, the relaxed projection operator

$$R_{K+e} = (1-t)Id + tP_{K+e}$$

is used in place of P_{K+e}. As noted in Section 2.1, t can take any value in the open interval $(0, 2)$. In Section 4 we will see that, in comparison to the unrelaxed algorithm ($t = 1$), performance of the algorithm improves by choosing t close to 2. Before presenting the algorithm, we introduce some additional notation.

Given a symmetric matrix $A \in S^n$, we will use $\hat{A}$ to denote A as an element in $\mathbb{R}^p$, $p = \frac{n(n+1)}{2}$. For example if $n = 3$ and

$$A = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix},$$

then $\hat{A}$ will be understood to be the vector $\hat{A} = (a, b, c, d, e, f)^T$. In a similar manner, if $v \in \mathbb{R}^p$, we will use $\check{v}$ to denote the corresponding element in S^n. For $x_0 \in \mathbb{R}$ and $x \in \mathbb{R}^m$, we will use $[x_0; x]$ to denote $[x_0, x^T]^T \in \mathbb{R}^{m+1}$.

Here is our algorithm for solving the LMI feasibility problem.

LMI-Feas:

```
*Problem Data.* Real $n \times n$ symmetric matrices
    $F_0, F_1, \ldots, F_m$.

*Parameter Selection.*
    Choose $\rho > 0$. (For example, take $\rho = 1$.)
    Choose $t \in (0, 2)$. (For example, take $t = 1.99$.)

*Initialization.* Choose any $x_0 \in \mathbb{R}$, $x \in \mathbb{R}^m$ and $S \in S^n$.

*Calculate G inverse.* $Q := [\hat{F}_0, \hat{F}_1, \ldots, \hat{F}_m] \in \mathbb{R}^{p\times(m+1)}$,
    $p = \frac{n(n+1)}{2}$. Let $I$ denote the $n \times n$ identity matrix
    and $\mathbf{1} \in \mathbb{R}^p$ the vector whose every entry is 1. $G :=$
    $QQ^T + \frac{1}{2}\mathrm{diag}(\hat{I} + \mathbf{1})$. $Ginv := G^{-1}$.

**repeat**

  1. *Apply $R_{K+e}$:*
       • $x_0 := (1-t)x_0 + t\max\{\rho, x_0\}$.
       • Find an eigenvalue-eigenvector decomposition
         of $S$. If $S = VDV^T$ with $D = \mathrm{diag}(d_1, \ldots, d_n)$,
         define $\bar{D} := \mathrm{diag}(\max\{\rho, d_1\}, \ldots, \max\{\rho, d_n\})$.
         $S := V((1-t)D + t\bar{D})V^T$.

  2. *Apply $P_L$:*
       • $\alpha := Ginv(Q[x_0; x] - \hat{S})$.
       • $[x_0; x] := [x_0; x] - Q^T\alpha$.
       • $T := -\frac{1}{2}\check{\alpha}$. For $i = 1, \ldots, n$, $T_{ii} := 2T_{ii}$. $S :=$
         $S - T$.

**until** $x_0 > 0$ and the minimum eigenvalue of $S$ is $> 0$.

*Solution:* $x/x_0$.
```

4 Computational Results

This section contains some computational results. Presented is a comparison of our algorithm with SeDuMi [8], a freely available convex optimization package that can be used to solve the LMI problem. A preliminary comparison with another package, SDPT3 [9], on trial problems showed that SeDuMi was roughly twice as fast as SDPT3, so we have omitted a detailed comparison of LMI-Feas with SDPT3. Note that both SeDuMi and SDPT3 are leading algorithms in their class [7].

All the computational results we are about to present were obtained using a 2 GHz Pentium 4 machine, with 512Mb of memory, running Windows XP Professional.

[3]Lemma 7 can be applied as the y_{ij}'s are linearly independent. This fact follows from the linear independence of the E_{ij}'s. In particular, the F_i's are *not* required to be linearly independent.

Our algorithm was coded using Matlab 6.5 and we note that SeDuMi is an add-on for Matlab.

All results presented are for randomly chosen LMI problems: for each problem, each element of each F_i matrix was drawn from a normal distribution of zero mean and variance 1. Problems generated in this manner which were infeasible were discarded. For all computational results, unless indicated otherwise, the parameter ρ that appears in LMI-Feas was set to $\rho = 1$. The initial value for LMI-Feas was always $(x_0, x, S) = (1, (0, \dots, 0)^T, I)$.

Table 1 compares LMI-Feas with SeDuMi. Results for LMI-Feas are given for two different values of the relaxation parameter: $t = 1$ and $t = 1.99$. Results are for $n = 10$ and for each value of m in the table, each algorithm was given the same 1000 random problems to solve and the results are average values determined from these problems. Listed are the average CPU time and the average number of iterations taken.

	LMI-Feas				SeDuMi	
	$t = 1$		$t = 1.99$			
m	T	i	T	i	T	i
50	0.0037	2.1	0.0033	1.6	0.071	3.3
40	0.0078	13	0.0058	7.4	0.073	3.4
30	0.16^a	330^a	0.13^b	290^b	0.11	4.6

Table 1: A comparison of LMI-Feas and SeDuMi, $n = 10$. T denotes average CPU time in seconds and i the average number of iterations. Results with a superscript are averaged values that do not include problems which had not converged after 10^4 iterations. The number of such problems in each case was: (a) 47 , (b) 30.

This is a small list of computationally results and some comments are in order. In the first two cases, LMI-Feas outperformed SeDuMi. For $m = 50$, LMI-Feas was about 20 times faster. For $m = 40$ and $t = 1.99$, LMI-Feas was about 13 times faster. These are clearly encouraging results. For $m = 30$, LMI-Feas had instances of non-convergence after 10^4 iterations. This combined with the other results in Table 1 highlight that the performance of LMI-Feas worsens as m is decreased (while keeping n fixed). Indeed finding a x that satisfies (1) becomes harder as m is decreased since the smaller m is, the smaller are the number of variables that can be adjusted in trying to satisfy (1).

Performance for $n = 10$, $m = 30$ can be improved by using a different value of ρ. Solving the same 1000 problems that produced the last line in Table 1 using $\rho = 0.001$ rather than $\rho = 1$ produces the results in Table 2.

Notice that by using $\rho = 0.001$ the number of problems that had not converged after 10^4 iterations has reduced substantially and the average time to convergence has also improved. These result demonstrate that ρ can have a considerable effect on performance. We consider this matter more fully in Section 5.

		LMI-Feas			
		$t = 1$		$t = 1.99$	
n	m	T	i	T	i
10	30	0.098^a	230^a	0.053^b	110^b

Table 2: Performance can improve by choosing a different ρ, in this case $\rho = 0.001$. Each of these results are averaged values that do not include problems which had not converged after 10^4 iterations. The number of such problems in each case was: (a) 4 , (b) 3.

For $m > 50$ (and $n = 10$), we have not included any computational results but we note the relative performance of LMI-Feas remains much the same as when $m = 50$. It remains able to solve problems about 20 times faster than SeDuMi.

The results of Tables 1 and 2 also demonstrate the benefit of using relaxed projections. In all cases, the relaxed algorithm ($t = 1.99$) performs better than the unrelaxed algorithm ($t = 1$). In all cases tested, including tests not documented here, the algorithm performs best with t close to 2. Indeed, solving the same problem a number of times using different values of t it becomes apparent that performance improves monotonically with increasing $t \in (0, 2)$.

Similar comments to the ones made above apply for other values of n: if m is large in comparison to n, LMI-Feas performs very favorably. Averaged results for some larger values of n are given in Table 3 below.

		LMI-Feas		SeDuMi	
n	m	T	i	T	i
40	800	2.2	1.6	22	3.2
60	1800	23	2.1	213	3.0

Table 3: Average CPU times and average number of iterations for two larger n's. Parameter values for LMI-Feas: $\rho = 1$ and $t = 1$.

In both cases, LMI-Feas was about 10 faster than SeDuMi.

5 The Effect of the Parameter ρ on Performance

Recall that Theorem 3 allows us to choose $e \in \overset{\circ}{K}$. For LMI-Feas, $\overset{\circ}{K} = \mathbb{R}_{++} \times \mathbb{R}^m \times S^n_{++}$ and our choice was $e = (\rho, 0, \rho I)$, with $\rho > 0$. Of course, we could have chosen $e = (\rho_1, 0, \rho_2 I)$ with $\rho_1 \neq \rho_2$, or we could have chosen a completely different e altogether. This raises the question, which value of e should be used in order to maximize speed of convergence?

Restricting ourselves to the current algorithm where $e = (\rho, 0, \rho I)$, Figure 1 shows that ρ can have a rather large effect on performance. Here we have solved the same LMI problem a number of times, each time using a different value of ρ. (The LMI problem examined was randomly generated with $n = 10$ and $m = 30$, and $t = 1$.)

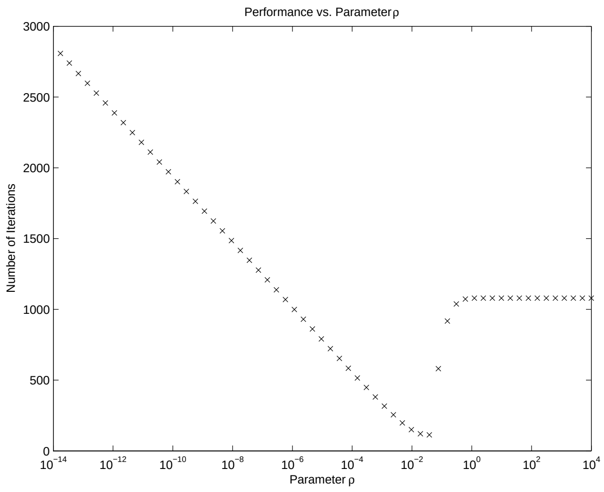

Figure 1: The effect of the parameter ρ on performance.

From Figure 1, a number of interesting observations can be made. Firstly, the performance of LMI-Feas appears to vary continuously with ρ and appears to have an optimal setting. Secondly, performance of the algorithm becomes independent of ρ if ρ is sufficiently large. (In this particular example, if ρ is about 10 or larger.) Unfortunately, performance for these large values of ρ can be an order of magnitude worse than at the optimal value. Finally, choosing ρ too small results in very poor performance. Indeed in the limiting case of $\rho = 0$, our algorithm reduces to applying the MAP algorithm to the sets K and L rather than to $K + e$ and L with $e \in \overset{\circ}{K}$. This highlights the benefit of our scheme over a more direct application of the MAP algorithm. Also, such a direct application will in general only converge to a point on the boundary of our feasible set rather than to a point in its interior, as desired.

While the optimal value of ρ seems to be problem dependent, depending both on the size of the problem, that is, on n and m, as well as on the problem data, the results of Table 2 demonstrate that given n and m there may be a ρ that works very well for a high percentage of problems but does not necessarily work well for all problems. These slower to converge problems can actually be solved just as quickly as problems which are quicker to converge if they are solved using a different value of ρ. An open question is whether there is a method of determining a priori an optimal or near optimal ρ? Alternatively, perhaps one could adapt ρ from one iteration to the next, starting at say a larger value and reducing ρ until sufficiently fast convergence was detected.

6 Conclusions

In this paper we have presented an algorithm for solving LMI problems. Assuming that a given problem is solvable, that is, that there exists x satisfying (1), the algorithm has been shown to converge to a solution of the problem in a finite number of steps. Computational results indicate that the algorithm perform relatively well, compared to other algorithms tested, for problems with a larger number of F_i matrices though not as well as such algorithms when the number of F_i matrices is smaller.

References

[1] M. Ait Rami, U. Helmke, and J. B. Moore. A finite step algorithm for solving convex feasibility problems. Submitted to SIAM Journal of Optimization.

[2] H. H. Bauschke and J. M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3):367–426, 1996.

[3] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2001.

[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia, 1994.

[5] L. M. Brègman. The method of successive projection for finding a common point of convex sets. *Soviet Mathematics*, 6(3):688–692, 1965.

[6] D. Luenberger. *Optimization by Vector Space Methods.* Wiley, New York, 1969.

[7] H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. To appear in Math. Prog.

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

[9] R. Tütüncü, K. Toh, and M. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. 2001.

[10] D. Youla and H. Webb. Image restoration by the method of convex projections: Part 1 – theory. *IEEE Trans. on Medical Imaging*, MI-1(2):81–94, 1982.