Projection methods for conic feasibility problems; applications to sum-of-squares decompositions

Didier Henrion, Jérôme Malick

Projection methods for conic feasibility problems, applications to polynomial sum-of-squares decompositions

Didier Henrion[1,2] Jérôme Malick[3]

May 18, 2009

Abstract

This paper presents a projection-based approach for solving conic feasibility problems. To find a point in the intersection of a cone and an affine subspace, we simply project a point onto this intersection. This projection is computed by dual algorithms operating a sequence of projections onto the cone, and generalizing the alternating projection method. We release an easy-to-use Matlab package implementing an elementary dual projection algorithm. Numerical experiments show that, for solving some semidefinite feasibility problems, the package is competitive with sophisticated conic programming software. We also provide a particular treatment of semidefinite feasibility problems modeling polynomial sum-of-squares decomposition problems.

1 Introduction

Conic and semidefinite feasibility problems. We consider in this paper the problem of finding a point in the intersection of a convex cone $\mathcal{K}$ with an affine subspace $\mathcal{A}$. In mathematical terms, this convex (conic) feasibility problem consists in finding $x \in \mathbb{R}^n$ such that

$$\begin{cases} Ax = b \\ x \in K, \end{cases} \tag{1}$$

where matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$ are given, and affine subspace $\mathcal{A}$ is defined by the equations $Ax = b$. In practice, $\mathcal{K}$ is often the nonnegative orthant, the second-order cone (or quadratic cone, or Lorentz cone) or the cone of positive semidefinite matrices (SDP cone for short) - as well as direct products of these cones. For instance, the basic SDP feasibility problem (with one SDP cone) consists in finding a matrix $X \in \mathbb{R}^{n \times n}$ satisfying

$$\begin{cases} \text{trace}(A_i X) = b_i, \quad i = 1, \ldots, m \\ X \succeq 0, \end{cases} \tag{2}$$

where the A_i are symmetric matrices and the notation $X \succeq 0$ means that X is positive semidefinite. Note that we can introduce $x \in \mathbb{R}^{n^2}$ as the vector obtained by stacking columns of matrix X and reformulate the matrix problem (2) as vector problem (1) where $A \in \mathbb{R}^{m \times n^2}$ is the matrix with the stacked A_i as rows, and $\mathcal{K}$ stands for the SDP cone.

Many engineering problems can be formulated as semidefinite or conic feasibility problems. In particular, control theory is one of the first areas that used SDP modelling [6]. For example in robust control, the search of quadratic Lyapunov functions for uncertain linear systems

[1]CNRS, LAAS, University of Toulouse (France)
[2]Czech Technical University in Prague (Czech Republic)
[3]CNRS, Laboratoire Jean Kuntzmann, University of Grenoble (France)

yields naturally an SDP feasibility problem. Another engineering example that leads to a conic feasibility problem is sensor network localization, see e.g. [50] and references therein. Solving efficiently conic feasibility can also have applications in conic programming itself, for example for finding starting points or, following [27], for solving linear conic programming problems as conic feasibility problems after formulation of the optimality conditions.

In this paper, we illustrate our developments on a particular application problem: the test of nonnegativity of polynomials via sum-of-squares (SOS) decompositions. The study of relationships between nonnegative and SOS polynomials has numerous applications, notably in polynomial programming, global optimization, signal processing and control theory, see e.g. [21, 10, 19, 29]. The bottomline is that checking if a polynomial is nonnegative is an NP-hard problem whereas checking if it is SOS is tractable since it reduces to an SDP feasibility problem, see Section 4.2.

Solving feasibility problems by conic programming. Once an engineering problem is modeled as a conic feasibility problem, it can be solved using powerful tools of conic programming. The natural approach is to use linear conic programming, see e.g. [51], [3]. We compute a solution of

$$\left\{ \begin{array}{ll} \min_x & c^\top x \\ & Ax = b \\ & x \in \mathcal{K}, \end{array} \right. \tag{3}$$

producing a feasible solution of (1), where $c \in \mathbb{R}^n$ is a given vector. Primal-dual infeasible interior points are among the most efficient methods to solve general conic problems [34].

In this paper, we introduce and study an alternative approach based on projections. Thinking about the problem geometrically, the idea consists in computing a feasible point by projecting onto the intersection of the cone and the affine subspace. In mathematical terms, this projection problem can be written as

$$\left\{ \begin{array}{ll} \min_x & \frac{1}{2}\|x - c\|^2 \\ & Ax = b \\ & x \in \mathcal{K}, \end{array} \right. \tag{4}$$

where $c \in \mathbb{R}^n$ is the point to project. The (squared) norm appearing in the objective function is $\|x\|^2 = x^\top x$, the (squared) Euclidean norm associated with the standard inner product in $\mathbb{R}^n$. For example, the projection problem involving the SDP cone that we consider to compute a feasible point of (2) can be written as

$$\left\{ \begin{array}{ll} \min_X & \frac{1}{2}\|X - C\|^2 \\ & \mathrm{trace}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0 \end{array} \right. \tag{5}$$

where $C \in \mathbb{R}^{n \times n}$ is the matrix to project. The norm here is the so-called Frobenius norm, associated with the standard inner product in the space of symmetric matrices $\|X\|^2 = \mathrm{trace}(X^2)$. Note that if the feasible domain is nonempty, there exists a unique solution to the projection problem, for any given c. The choice of the point c to be projected depends on some prior knowledge on the underlying conic feasibility problem. To fix ideas, one can choose $c = 0$ by default; as we explain later, this choice is often interesting.

The approach we propose in this paper is to use these projection problems, special cases of quadratic conic programming problems, to solve conic feasibility problems. This approach has a geometric appeal, but the key for practical success is that problem (4) is generally "easier" to solve than problem (3), given the same feasible domain. Problem (4) looks more complicated than problem (3) since its objective function is quadratic rather than linear. It turns out that

the coercivity of the objective function provides nice properties to the dual problem, which can then be solved efficiently by a succession of projections onto teh cone, as we explain later on in Section 2. Thus in practice, we can solve (4) with this dual approach as soon as we can project onto cone $\mathcal{K}$. It is the case in particular for the cones we consider (polyhedral, second-order and SDP cones).

Numerical solvers using this dual approach have been recently developed to solve problems (4) and (5) in general, or specific instances. The dual approach was proposed in [33], then revisited by [8] in the case of SDP cone, and then enhanced by [43] and [5] for a particular projection problem. Other approaches, among them interior-points [49] and alternative projections [25], have also been studied to solve (4) and (5). We aim at showing in this paper that these new solvers should be considered as alternatives to standard linear conic programming solvers for solving conic feasibility problems. For other projection algorithms based on different techniques, we refer to [12] (Dikin ellipsoids), and [44] (relaxed projections onto the intersection of two cones).

Contributions and structure of the paper. This paper proposes a new approach based on projections for solving conic feasibility problems. We list below the main contributions of this paper.

1. We introduce the projection methods in the context of solving conic feasibility problems. We aim at showing that these new conic programming methods can be considered as alternatives to standard approaches.

2. From a more theoretical point of view, we develop a geometric treatment of the conic feasibility problem. We explain the intrinsic meaning of the standard Slater condition and its special role in the projection methods.

3. We release a Matlab software, simple and easy-to-use, implementing an elementary projection algorithm. We experiment with it on feasibility problems, showing that a rough implementation can be competitive with more sophisticated conic programming software. More precisely, we compare with SeDuMi [47], which is known to be one of the best linear conic programming solvers, and which is widely used in engineering, see for example [20, 21, 10].

4. We illustrate the developments on polynomial SOS decomposition, a particular instance of SDP feasibility problems. In particular, we provide a regularization process that forces good properties for projection solvers.

The paper is structured as follows. Section 2 presents what we called the projection algorithms, a family of dual algorithms for solving the projection problem (4). Then Section 3 deals with the Slater condition, which has a geometric appeal in our context, in addition to its role in the convergence of the projection algorithms. In Section 4, some properties of the semidefinite feasibility problem (2) are further studied in the case of polynomial SOS decompositions. Section 5 presents a Matlab software implementing an elementary projection algorithm. Numerical illustrations of the projection algorithm for these semidefinite problems are finally exposed and discussed in Section 6.

2 Dual algorithm to project onto the intersection

As stated in the introduction, in order to find a feasible point for problem (1), we project a point onto the intersection of a cone and an affine subspace, or equivalently, we solve problem (4). In this section, we review the dual approach of [33] to solve this projection problem, and we introduce the (dual) projection algorithm we use later.

To start, note that if the intersection $\mathcal{K} \cap \mathcal{A}$ is nonempty, problem (4) has a unique solution, namely the projection of point c onto the intersection of these two closed convex sets. In the remainder of the paper, we make the following assumptions:

- the intersection $\mathcal{K} \cap \mathcal{A}$ is nonempty,
- the cone $\mathcal{K}$ has full dimension ($\operatorname{int} \mathcal{K} \neq \emptyset$),
- the affine constraints are linearly independent (matrix A has full row-rank).

These assumptions are usually satisfied in practice. It is the case for the applications listed in the introduction, in particular for our target application, testing if a polynomial is SOS.

The approach of [33] consists in using the mechanism of Lagrangian duality (see Chapter XII of [26]) and then taking advantage of the geometrical features of the problem. We introduce the Lagrangian, a function of primal variable $x \in \mathcal{K}$ and dual variable $y \in \mathbb{R}^m$

$$L(x, y) := \frac{1}{2}\|c - x\|^2 - y^\top (Ax - b), \tag{6}$$

and the corresponding concave dual function

$$\theta(y) := \min_{x \in \mathcal{K}} L(x; y), \tag{7}$$

which is to be maximized. We denote by P_K the projection operator onto K. We can show [33, Th.3.1] that there exists a unique point which reaches the above minimum,

$$x(y) := \mathrm{P}_K(c + A^\top y), \tag{8}$$

and that there holds (up to a constant)

$$\theta(y) = -\frac{1}{2}\|x(y)\|^2 + b^\top y. \tag{9}$$

It is also not difficult to show [33, Th.3.2] that the concave function θ is differentiable on $\mathbb{R}^m$, with a gradient

$$\nabla\theta(y) = -Ax(y) + b \tag{10}$$

which is Lipschitz continuous. This regularity has great impact in practice: it opens the way for using standard algorithms for unconstrained differentiable optimization; we can cite, among others, gradient, quasi-Newton, limited memory quasi-Newton, truncated generalized Newton, or Nesterov's method (see textbooks [38], [4] and paper [36]).

Among these methods, we consider in particular the gradient method in Section 3.1 for an interpretation of alternating projections, and quasi-Newton for our basic implementation of this approach, see Section 5. The dual problem can be tackled with classical tools, and we get directly the primal solution x^* of (4) from a dual solution y^*, more precisely we have $x^* = x(y^*)$, see [33]. Thus we have the following dual method to solve problem (4).

Algorithm 2.1 (computing the projection onto the intersection)**.** *Use an optimization code for functions with Lipschitz gradient to maximize θ on $\mathbb{R}^m$. This code generates a maximizing sequence $\{y_k\}_k$ together with:*

$$x_k = x(y_k), \qquad \nabla\theta(y_k) = -Ax_k + b \qquad \text{and} \qquad \theta(y_k) = -\frac{1}{2}\|x_k\|^2 + {y_k}^\top b.$$

If there exists a dual solution y^* and at iteration k the algorithm computes an approximate solution y_k, then x_k is an approximation of the unique solution of problem (4) *in the sense that if $\|y - y_k\| \leq \varepsilon$, then $\|x - x_k\| \leq \|A\|\varepsilon$.*

There is a standard assumption ensuring the existence of dual solutions and then the relevance of this algorithm for solving the problem (4): it is the (primal) Slater condition assuming that there exists a stricly feasible point in the intersection, that is

$$\exists\, \bar{x} \in \mathbb{R}^n \ : \ A\bar{x} = b,\ \bar{x} \in \operatorname{int} \mathcal{K}.$$

We emphasize that this assumption ensures a good behaviour of the projection algorithm whereas the dual Slater condition, or strong complementarity condition, does not have such a role. In the next section, we discuss its geometric meaning and its role in projection methods.

3 Slater condition and regular intersection

The role of Slater condition in conic programming has been underlined many times; see for instance the textbook [7] and the recent review [48] for more references. In this section, we argue that this condition is natural in our situation, by stating equivalent geometric conditions, Farkas-like alternative results, relating this to regularity and projection algorithms. We mention different points of view and techniques coming from variational analysis.

3.1 Regular intersection

The Slater condition ensures good behavior of the dual algorithm, but it is not just a technical assumption: it carries out a geometric meaning, intrinsic for the feasibility problem we consider. Before seeing this, we need more notation. Given a cone $\mathcal{K}$ in $\mathbb{R}^n$, its polar cone is defined as the set of points of $\mathbb{R}^n$ whose projection onto $\mathcal{K}$ is 0, or in mathematical terms,

$$\mathcal{K}^\circ := \{y \in \mathbb{R}^n : \quad y^\top x \leq 0,\ x \in \mathcal{K}\}.$$

The normal cone to $\mathcal{K}$ at $x \in \mathcal{K}$ is denoted $N_{\mathcal{K}}(x)$. The tangent cone to $\mathcal{K}$ at $x \in \mathcal{K}$ is denoted $T_{\mathcal{K}}(x)$. We also remind that $N_{\mathcal{K}}(x) = (T_{\mathcal{K}}(x))^\circ$. We say that $\mathcal{K}$ and $\mathcal{A}$ can be separated if

$$\exists u \neq 0 \in \mathbb{R}^n,\ \alpha \in \mathbb{R}, \quad \forall k \in \mathcal{K},\ a \in \mathcal{A} : \quad u^\top k \leq \alpha \leq u^\top a,$$

and we say that $\mathcal{K}$ and $\mathcal{A}$ are strictly separated if the second inequality is strict. For the definitions and properties of these basic notions of convex analysis, see e.g. [45], [26].

Proposition 3.1 (Slater assumption)**.** *In our context, the following properties are equivalent:*

(i) $\operatorname{int} \mathcal{K} \cap \mathcal{A} \neq \emptyset$,

(ii) $T_{\mathcal{K}}(x) + \ker A = \mathbb{R}^n,\ \forall x \in \mathcal{K} \cap \mathcal{A}$,

(iii) $N_{\mathcal{K}}(x) \cap \operatorname{span} A^\top = \{0\},\ \forall x \in \mathcal{K} \cap \mathcal{A}$,

(iv) $0 \notin \operatorname{int}(\mathcal{K} - \mathcal{A})$,

(v) $\mathcal{K}$ *and* $\mathcal{A}$ *cannot be separated.*

Proof. Properties (i)-(iii)-(v) are equivalent by Theorem 2.8 and the associated propositions of [35]. The idea is that these properties all correspond to the fact that x is not an "extreme point" of $\{\mathcal{K}, \mathcal{A}\}$ (such extremality is a key concept of [35]). The equivalence of (ii) and (iii) comes from polar duality: following [45, 16.4.2] we write

$$(T_{\mathcal{K}}(x) + \ker A)^\circ = (T_{\mathcal{K}}x)^\circ \cap (\ker A)^\circ = N_{\mathcal{K}}(x) \cap \operatorname{span} A^\top,$$

that gives the equivalence. Finally we just have to prove the equivalence of (i) and (iv). To see this, we use that $\operatorname{int}\mathcal{K} \neq \emptyset$ to first get $\operatorname{int}(\mathcal{K} - a) \neq \emptyset$ for $a \in \mathcal{A}$, and then $\operatorname{int}(\mathcal{K} - a) \subset \operatorname{int}(\mathcal{K} - \mathcal{A}) \neq \emptyset$. Therefore we can write with the help of [45, 6.6.2]

$$\operatorname{int}(\mathcal{K} - \mathcal{A}) = \operatorname{ri}(\mathcal{K} - \mathcal{A}) = \operatorname{ri}\mathcal{K} - \operatorname{ri}\mathcal{A} = \operatorname{int}\mathcal{K} - \mathcal{A}.$$

This leads to the equivalence and ends the proof. □

The above properties often come into play when studying general convex feasibility problems, see e.g. [2], [16]. Roughly speaking they mean that the feasibility problem is "well-posed" from a geometric point of view. The authors of [31] say that the intersection is "regular" and they connect that to the so-called metric regularity and the distance to ill-posedness. In our situation, the regularity of the intersection $\mathcal{K} \cap \mathcal{A}$ is equivalent to the Slater condition or, again, to the fact that $\mathcal{K}$ and $\mathcal{A}$ cannot be separated by an hyperplane. For another example, the intersection of smooth manifolds is regular if they intersect transversally [30]. This regularity has an important algorithmic impact for basic projection methods, as explained in the next section.

3.2 Alternating projections

The regularity of the intersection of $\mathcal{K}$ and $\mathcal{A}$ turns out to have interesting algorithmic consequences. This regularity is indeed the key assumption to have linear convergence of alternating and averaged projection methods [31]. In this section, we draw a clear connection between the dual projection algorithm 2.1 and an alternating projection scheme, giving one more argument that the Slater assumption is natural in our situation. However note already that the algorithm that will be presented in Section 5 is not an alternating projection method, but is related to it, as we explain after Proposition 3.3 below. An alternating projection algorithm for solving some conic feasibility problems of control theory (modeled as intersections of two cones) is presented in the recent work [44].

The alternating projection method is a simple and intuitive algorithmic scheme that can be applied to find a point in the intersection of two sets: it consists in projecting the initial point onto the first set, say $\mathcal{A}$, then projecting the new point onto the second set, $\mathcal{K}$, and then projecting again the new point onto $\mathcal{A}$ and keep on projecting alternatively. In other words, it consists in repeating the operation:

$$X \leftarrow \mathrm{P}_{\mathcal{K}}(\mathrm{P}_{\mathcal{A}}(X)). \tag{11}$$

If the sets have a regular intersection, this algorithm converges linearly to a point in the intersection and we have an estimate of the speed of convergence. For two convex sets, see for instance [16], for the general case, see the local result of [31].

In our situation, one of the two subsets is affine, so we have an explicit projection onto it: the projection of x on $\mathcal{A}$ is

$$\mathrm{P}_{\mathcal{A}}(x) = x - A^\top [AA^\top]^{-1}(Ax - b) \tag{12}$$

since we assumed that matrix A has full row-rank. We could apply this alternating projection scheme as soon as we know how to project onto cone $\mathcal{K}$. It is the case in particular for the nonnegative orthant, the second-order cone, and the SDP cone. However the convergence of this method is slow, more precisely it is linear, with a rate of convergence generally close to 1, depending on the cosine of the angle between $N_{\mathcal{K}}(\bar{x})$ and $\operatorname{span} A^\top$, see [31].

The so-called Dykstra's correction [11] ensures convergence to the projection of c onto the intersection - and not only to a point in the intersection. In practice a correction is added before projection onto the cone $\mathcal{K}$, roughly speaking, to compensate the curvature of $\mathcal{K}$.

Algorithm 3.2 (Corrected alternating projection)**.** *Set* $\delta_0 = 0, a_0 = c, x_0 = \mathrm{P}_{\mathcal{K}}(c)$*; then*
for $k = 1, 2, \ldots$
$\tilde{a}_k = a_{k-1} - \delta_{k-1}$ *(Dyskstra's correction)*
$x_k = \mathrm{P}_{\mathcal{K}}(\tilde{a}_k)$
$\delta_k = x_k - \tilde{a}_k$
$a_k = \mathrm{P}_{\mathcal{A}}(x_k)$
end

The alternating projection method has a dual interpretation: in the right metric, it is essentially a gradient optimization algorithm to maximize the dual function. This is stated in the following proposition, generalizing a result of [33] on the nearest correlation matrix problem.

Proposition 3.3. *The sequence* $(x_k)_k$ *generated by Algorithm 3.2 is the same as the sequence* $(x(y_k))_k$ *of Algorithm 2.1 applied to the sequence* $(y_k)_k$ *produced by the scheme*

$$y_0 = 0, \qquad y_{k+1} = y_k + [AA^\top]^{-1} \nabla\theta(y_k), \tag{13}$$

namely a gradient algorithm in the metric $[AA^\top]^{-1}$ *to maximize the dual function* (9)*.*

Proof. By definition, we just need to prove by recurrence that $\tilde{a}_k = c + A^\top y_k$. It is true for $k = 0$; suppose it holds for k, then:

$$\begin{array}{rcll}
\tilde{a}_{k+1} & = & a_k - \delta_k & \text{[definition of } \tilde{a}_{k+1}\text{]} \\
& = & a_k - x_k + \tilde{a}_k & \text{[definition of } \delta_k\text{]} \\
& = & -A^\top [AA^\top]^{-1}(Ax_k - b) + \tilde{a}_k & \text{[(12) and definition of } a_k\text{]} \\
& = & -A^\top [AA^\top]^{-1}(Ax(y_k) - b) + c + A^\top y_k & \text{[recurrence assumptions]} \\
& = & c + A^\top (y_k - [AA^\top]^{-1}(Ax(y_k) - b)) & \\
& = & c + A^\top y_{k+1}\,, & \text{[(10) and definition of } y_{k+1}\text{]}
\end{array}$$

hence $x_{k+1} = \mathrm{P}_{\mathcal{K}}(\delta_{k+1}) = \mathrm{P}_{\mathcal{K}}(c + A^\top y_{k+1}) = x(y_{k+1})$, and the result is proved. □

Thus (corrected) alternating projections have a dual interpretation, as a gradient method with constant step size to solve the dual problem. Therefore, we can say that Algorithm 2.1 is, in some sense, a generalization of the alternating projection method. This is an argument in favor of more evolved techniques to maximize the dual function in Algorithm 2.1: a Newton-like method for solving the dual problem may be more efficient. In Section 5 we give a basic implementation of this algorithm using a quasi-Newton method.

3.3 Empty intersection

We would like to be able to detect during the run of Algorithm 2.1 if the intersection $\mathcal{K} \cap \mathcal{A}$ is empty. The conic feasibility problem

$$\begin{cases} A^\top y \in \mathcal{K}^\circ \\ b^\top y > 0 \end{cases} \tag{14}$$

gives a certificate of infeasibility of (1). If there is a feasible solution of (14) then there is no feasible solution of (1), as formalized in the next statement.

Proposition 3.4. *(i) If* $\mathcal{K} \cap \mathcal{A} \neq \emptyset$*, then there is no feasible solution of* (14)*.*

(ii) Assume $c = 0$*; if there exists* y *such that* $x(y) = 0$*, then* $\mathcal{K} \cap \mathcal{A} = \emptyset$*.*

Proof. To prove (i), take a feasible point $x \in \mathcal{K} \cap \mathcal{A}$ and any point $y \in \mathbb{R}^m$ such that $A^\top y \in \mathcal{K}^\circ$, and observe that the fact that $A^\top y$ and x lie in polar cones implies

$$0 \geq x^\top A^\top y = (Ax)^\top y = b^\top y.$$

To prove (ii), assuming $c = 0$, note that we have the equivalence

$$A^\top y \in \mathcal{K}^\circ \iff \mathrm{P}_{\mathcal{K}}(A^\top y) = 0 \iff x(y) = 0,$$

the second equivalence coming from definition (8). Hence (ii) is just the converse statement of (i). □

This proposition has two interesting consequences. First, a vector y in (14) is a certificate of infeasibility of our problem (1). Second, if $c = 0$, checking this certificate during the run of Algorithm 2.1 costs nothing numerically: since we compute $\|x(y)\|^2$ when evaluating the dual function (9), we just test if this term is nonzero. If $c \neq 0$, we have to compute an extra projection onto the cone to check if $\mathrm{P}_{\mathcal{K}}(A^\top y) = 0$. This is an argument in favor of taking $c = 0$ when the modelling process of a problem into the form (1) does not give any clue on intrinsic choices of c.

Following Proposition 3.4, a question remains of whether there always exists a vector y satisfying (14) when $\mathcal{K}$ and $\mathcal{A}$ do not intersect. The answer is yes if $\mathcal{K}$ and $\mathcal{A}$ do not intersect "strictly", as precised in the next proposition.

Proposition 3.5. *If $\mathcal{K}$ and $\mathcal{A}$ can be strictly separated, then there exists a feasible solution to* (14).

Proof. Let $u \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ realizing the strict separation

$$\forall\, k \in \mathcal{K}, a \in \mathcal{A} : \quad u^\top k \leq \alpha < u^\top a.$$

Note first that $\alpha \geq 0$ since $0 \in \mathcal{K}$. We prove now that $u \in \mathcal{K}^\circ$. For $x \in \mathcal{K}$ and $t > 0$, we have $tx \in \mathcal{K}$ so $tu^\top x \leq \alpha$. Letting $t \to +\infty$, we get $u^\top x \leq 0$, which proves $u \in \mathcal{K}^\circ$. Finally, we prove that $u \in \operatorname{span} A^\top$. Decompose $u = A^\top y + z$ with $A^\top y \in \operatorname{span} A^\top$ and $z \in \ker A$. We have for all $a \in \mathcal{A}$,

$$\alpha < u^\top a = y^\top b + z^\top a.$$

Then, for a fixed $a_0 \in \mathcal{A}$ and $t > 0$, take $a_t = a_0 - tk$ in $\mathcal{A}$, and use the above inequality to get for all $t > 0$

$$\alpha - y^\top b - z^\top z_0 < -t\|z\|^2$$

which implies $z = 0$ and therefore $u = A^\top y \in \operatorname{span} A^\top$. We can conclude: $u = A^\top y$ lies in $\mathcal{K}^\circ$ and $0 \leq \alpha < A^\top y^\top a = y^\top b$, and we have a feasible y in (14). □

This result is a sort of Farkas alternative result: see the standard Farkas lemma for polyhedral cones in [46] and for SDP cones in [1]. As pointed out to us by Jean-Bernard Lasserre, there exist also extensions in more general settings, see [9] and [7] for instance, of which our situation is a particular case. The proofs of these results all use a separation argument. In our context, the proof is easy, so that we include it above for completeness.

4 SDP feasibility and SOS decomposition

In this section, we emphasize two features of particular SDP feasibility problems: first, the rank of the feasible matrix computed by solving problem (4); second, the absence of Slater condition for SDP feasibility problems coming from polynomial SOS decomposition.

4.1 Rank of the solution and semidefiniteness

Though it is required that SDP feasibility problem (2) has full-rank solutions to avoid numerical troubles, the dual algorithm turns out to find low-rank solutions, as detailed just below. To some extent, projection algorithms for solving conic feasibility problems are opposite to interior-point algorithms that yield solutions with the highest possible rank. Applied to problem (3), an interior-point algorithm aims indeed at finding the solution which lies deepest inside the feasible domain, by using a barrier on the boundary of $\mathcal{K}$ [3].

As a projection method, the dual algorithm gives naturally points on the boundary of the cone, since the last operation is a projection onto the cone. Thus, for the SDP cone, the last iterate does not have full rank usually. Actually we can state a more precise result: if the solution X^* of problem (5) has full rank, then it means that the problem has an explicit closed-form solution.

Lemma 4.1 (Trivial case)**.** *Assume that there exists a primal-dual solution (X^*, y^*) to problem (5). If X^* is full-rank, then we have the closed form solution*

$$y^* = G^{-1}(\gamma - b)$$

where $G \in \mathbb{R}^{m\times m}$ has entries $G_{ij} = \operatorname{trace}(A_i A_j)$, and $\gamma \in \mathbb{R}^m$ has entries $\gamma_i = \operatorname{trace}(A_i C)$.

Proof. Note first that G is a Gram matrix constructed with independent elements A_i, so it is nonsingular. Note also that the projection operator (17) always decreases the rank. Thus, if X^* is full-rank, so is $C + \sum_i A_i y_i$, and therefore $X^* = C + \sum_{i=1}^m A_i y_i^*$. Now use the fact that $b_j = \operatorname{trace}(A_j X^*)$, to get $b_j = \operatorname{trace}(A_j C) + (Gy^*)_j$ which permits to conclude. □

Hence the matrices obtained by the projection algorithm are typically not full-rank. It is interesting for the construction of SOS decompositions of polynomials: as explained in the next section, the rank of the matrix X is the number of squared polynomials in the decomposition. We are then naturally interested in "simple" decompositions, that is, with a minimum number of squared polynomials. In contrast, an interior-point solver returns a maximum rank solution, so a maximum number of squared polynomials.

The projection algorithm can provide a first answer for some SDP problems with rank constraints. Some control problems (including controller design, model reduction) require extra rank constraint on the target matrix X, and can be modelled as:

$$\begin{cases} \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, m \\ X \succeq 0 \\ \operatorname{rank}(X) = r, \end{cases} \tag{15}$$

see [17] for an early treatment. These problems appear also in the current trends to use sparsity in signal processing and control theory. A popular heuristic algorithm is to introduce the trace of the matrix to replace the rank, recent techniques use the nuclear norm [18]. Projection methods give automatically a low-rank solution, that can be used as an initial guess to solve problem (15). Obviously, there is no guarantee of getting the target rank r. This approach could be nevertheless a first step or a preprocessing step for a more evolved technique. Recently, a blend of projection and Newton's methods was successfully used to solve rank-constrained SDP problems from systems control [39].

We mention though that it is possible to guarantee positive definiteness if necessary. We can even control the level of requested positive definiteness. The approach of Section 3 can deal

with the constraint $X \succeq \gamma I$. To see this directly we use the change of variables $X_\gamma = X - \gamma I$, $C_\gamma = C - \gamma I$, $b_{\gamma_i} = b_i - \gamma \operatorname{trace}(A_i)$ to transform

$$\begin{cases} \min_X & \frac{1}{2}\|X - C\|^2 \\ & \operatorname{trace}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq \gamma I \end{cases}$$

into a problem of form (5):

$$\begin{cases} \min_{X_\gamma} & \frac{1}{2}\|X_\gamma - C_\gamma\|^2 \\ & \operatorname{trace}(A_i Y) = b_{\gamma_i}, \quad i = 1, \ldots, m \\ & X_\gamma \succeq 0. \end{cases}$$

This is an important feature for applications in control theory when computing coercive Lyapunov functions, with γ the expected level of performance [6].

4.2 SOS decomposition

Let

$$p(t) = \sum_{|\alpha| \leq 2d} p_\alpha t^\alpha$$

be a polynomial of the variable $t = (t_1, \ldots, t_p) \in \mathbb{R}^p$ and of total degree $2d$, with $\alpha \in \mathbb{N}^p$ containing powers of each monomial, i.e. $t^\alpha = t_1^{\alpha_1} \cdots t_p^{\alpha_p}$.

Proposition 4.2. *Polynomial $p(t)$ is a sum-of-squares (SOS) if and only if there exists a symmetric matrix X such that*

$$p(t) = \pi(t)^\top X \, \pi(t), \qquad X \succeq 0 \tag{16}$$

with $\pi(t)$ a vector forming a basis of the space of polynomials of p variables of total degree d. The rank of X corresponds to the number of squared polynomials in the decomposition.

This result is as fundamental as easy to prove. The idea is to use the eigenvectors $(q_i)_{i=1,\ldots,r}$ associated to nonzero eigenvalues $(\lambda_i)_{i=1,\ldots,r}$ of matrix X, to write

$$p(t) = \sum_{i=1}^{r} \lambda_i (q_i{}^\top \pi(t))^2,$$

where r is the rank of X, see [15, 42] and more recently [28, 37, 40, 41]. Note that the SOS decomposition of p is not unique, and so is the decomposition (16). In particular, the number of squared polynomials can differ from one SOS decomposition to another.

Here is an easy example of decomposition. Let $\pi(t) = [1, t, t^2]^\top$ with $p = 1$ and $d = 2$. The polynomial $t^4 + 2t^2 + 1$ is SOS and it can be written as a sum of 3 squares:

$$t^4 + 2\,t^2 + 1 = \pi(t)^\top \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \pi(t).$$

There is another formulation with one square:

$$t^4 + 2\,t^2 + 1 = (t^2 + 1)^2 = \pi(t)^\top \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \pi(t).$$

Observe now that the Slater condition is not always satisfied for SDP feasibility problems (2) and (16). The condition corresponds to the existence of positive-definite (full-rank) matrix X, or equivalently to an SOS decomposition with a maximum number of squares. We call such polynomials strictly SOS.

Various strategies are available to reformulate the problem and reduce the size of the polynomial basis, and hence of the matrix X. First, we can use the Newton polytope to restrict the basis to monomials appearing in the convex hull of the exponents of the polynomial. Second, we can exploit sign symmetries, and more generally, symmetry reductions based on invariant group theory. See [41] for a recent overview, and [32] for an implementation-oriented survey. Such a pre-processing step should be computationally cheaper than solving the SDP problem itself. Even after a pre-processing phase, we have no guarantee that the Slater condition is satisfied, and a direct application of the projection algorithm may run into troubles. We therefore propose a regularization procedure to enforce the Slater condition as follows.

Given a polynomial $p(t)$, the idea is to perturb it by a given strictly SOS polynomial $q(t)$, and we define

$$p_\varepsilon(t) := p(t) + \varepsilon q(t) \qquad \text{with} \qquad q(t) := \|\pi(t)\|^2, \quad \varepsilon \in \mathbb{R}$$

which has better properties for our purpose.

Lemma 4.3. *For all $\varepsilon > 0$, we have*

$$\begin{aligned} p \text{ is nonnegative} &\Rightarrow p_\varepsilon \text{ is nonnegative,} \\ p \text{ is SOS} &\Rightarrow p_\varepsilon \text{ is SOS and the Slater condition is satisfied.} \end{aligned}$$

Proof. Since $q(t)$ is SOS, it is obvious that p_ε is nonnegative (resp. SOS) as soon as p is nonnegative (resp. SOS). The proof that the Slater condition is satisfied when $p(t)$ is SOS is direct as well: let $X \succeq 0$ be such that $p(t) = \pi(t)^\top X \pi(t)$, then

$$p_\varepsilon(t) = \pi(t)^\top X \pi(t) + \varepsilon \|\pi(t)\|^2 = \pi(t)^\top X\, \pi(t) + \varepsilon \pi(t)^\top \pi(t) = \pi(t)^\top (X + \varepsilon I) \pi(t),$$

and by construction, the matrix $X + \varepsilon I$ is positive definite for any $\varepsilon > 0$. □

We also have a converse property.

Lemma 4.4. *For all polynom p, there exists $\varepsilon_0 > 0$ such that for all $\varepsilon \leq \varepsilon_0$, we have*

$$p_\varepsilon \text{ is nonnegative} \quad \Rightarrow \quad p \text{ is nonnegative.}$$

Proof. We prove the contraposed statement. Assume that there is a $t_0 \in \mathbb{R}^p$ such that $p(t_0) < 0$. Set

$$\varepsilon_0 = \frac{-p(t_0)}{2q(t_0)} > 0.$$

Note that ε_0 is well-defined since $p(t_0) < 0$ and for all t, $q(t) > 0$. This second inequality comes from the fact that for all t, $\pi(t) \neq 0$ (indeed, if it was not the case, all the polynoms of the entries of $\pi(t)$ would have a common zero, which contradicts the fact it is a basis). This choice of ε_0 implies that for all $\varepsilon \leq \varepsilon_0$

$$p_\varepsilon(t_0) \leq p(t_0)/2 < 0$$

which shows that p_ε is not nonnegative. □

Thus if p is SOS, p_ε has the good properties for our projection algoritm. Therefore testing SOS, hence nonnegativity, of p_ε can be done efficiently with dual algorithms, see the numerical experiments in Section 6. This leads to the following regularization process for testing nonnegativity of p :

1. Choose a small $\varepsilon > 0$.

2. Test if $p_\varepsilon(t) = p(t) + \varepsilon\|\pi(t)\|^2$ is SOS (with the projection method, since the Slater condition is satisfied by Lemma 4.3).

3. Conclude on the nonnegativity of p (following Lemma 4.4).

If the initial polynomial p is SOS, this process indeed certifies that p is positive as soon as ε is chosen small enough. Unfortunately, there is no automatic way to choose parameter ε a priori, so we cannot give a general guarantee with this regularization. The numerical experiments of Section 6.5 also show that the choice of $\varepsilon \in [10^{-16}, 10^{-2}]$ impact the performance of the dual solver.

5 SDLS: semidefinite least-squares software

This section introduces the Matlab package SDLS, a basic implementation of Algorithm 2.1. The SDLS package is available by following instructions at

`www.laas.fr/~henrion/software/sdls`

This implementation is not meant to be efficient and robust. Our main goal is to provide a simple, user-friendly software for solving and experimenting with conic least-squares problems, especially in the context of conic feasibility problems. Up to our knowledge, no such freeware exists at this date. We aim at showing that a basic implementation can already be competitive with the best interior-point solvers for solving certain classes of feasibility problems.

SDLS is a basic implementation of Algorithm 2.1 and then, in view of Proposition 3.3, it can be seen as a generalization of the alternating projection method. Among all the numerical methods to maximise the dual function, we choose a quasi-Newton algorithm which is known to be efficient. Moreover it is required to compute $x(y)$, for given dual variable y, and that boils down to computing a projection onto $\mathcal{K}$, remember (8). It is easy to compute the projections onto polyhedral and second-order cones. It is also well-known, see for instance [24], that there is an explicit projection onto the SDP cone. Suppose that X has the spectral decomposition

$$X = U \operatorname{Diag}(\lambda_1, \ldots, \lambda_n) U^\top$$

where $\lambda_1 \geq \cdots \geq \lambda_n$ are the (ordered) eigenvalues of X and U is a corresponding orthonormal matrix of eigenvectors of X. Then the projection of X onto the SDP cone $\mathcal{K}$ is

$$\mathrm{P}_K(X) = U \operatorname{Diag}\big(\max(0, \lambda_1), \ldots, \max(0, \lambda_n)\big) U^\top. \tag{17}$$

Hence projecting onto the SDP cone boils down essentially to computing an eigendecomposition. This is the most expensive operation in the algorithm.

Thus the two key numerical components of SDLS are the following:

- For the optimization code, we use an off-the-shelf Matlab implementation of quasi-Newton: the freeware HANSO [13], a BFGS algorithm with Wolfe line-search [4]. HANSO is already used in the Matlab package HIFOO to solve problems of control theory, following a totally different approach than conic optimization [14].

- The eigenvalue decomposition for symmetric matrices, used for projecting onto the SDP cone, is simply achieved by Matlab's built-in linear algebra function `eig`, in turn based on LAPACK's function `DSYEV`.

We insist on the following features:

- Simplicity: the SDLS package consists of only 4 Matlab interpreted m-files, summing up to about 50 lines of code (excluding comments and input argument parsing), calling one external package and a built-in linear algebra function.
- Easy-to-use: SDLS has the same syntax as the widely used freeware SeDuMi for solving linear problems over convex symmetric cones [47]. Note also that, like in SeDuMi, symmetric matrices are represented as a vector of n^2 entries by stacking the columns.

The basic calling syntax is:

```
[x,y] = sdls(A,b,c,K)
```

Input arguments `A`, `b`, `c` are real-valued matrices as in (4). If the third input argument `c` is empty or unspecified, then it is set to zero. Though data sparsity is not directly exploited by the algorithm, SDLS benefits of sparsity for matrix-vector multiplications, and `A`, `b`, `c` can be Matlab objects of class `sparse`. We refer to the online documentation [22] for the advanced use: we can control the algorithm behaviour (accuracy, maximum number of iterations...) by specifying optional parameters. Note finally that, even though SeDuMi and SDLS share basically the same calling syntax, they do not solve the same problem: SeDuMi is aimed at solving (3) and not (4). We can compare them only for conic feasibility problems, in the absence of an objective function.

Constructed by projection, the primal iterates $x_k = x(y_k)$ always satisfy the conic constraints of the problem (1). The affine constraint is only attained asymptotically. The natural stopping test in unconstrained differentiable optimization is that the gradient of the function is close to zero. This has an interpretation in our framework: the norm of the gradient corresponds indeed to the primal infeasibility residual $\|Ax_k - b\|$, remember relation (10). In practice, the algorithm stops when the relative infeasibility residual $\|Ax_k - b\|/(1 + \|b\|)$ is less than a given threshold, chosen by default at 10^{-6}.

Again, the implementation of the algorithm in SDLS is probably not the most efficient. It is not meant to outperform neither the private advanced implementations of this method, nor other methods solving the same problem. Our main goal is to provide a simple, user-friendly, free software for solving projection problems onto conic feasibility domains. For more details on the syntax, for installation and advanced use, refer to the web site and the online documentation [22].

6 Numerical experiments

In this section, we illustrate the behavior of the projection algorithm and we show its validity for conic feasibility problems. We also compare it with the standard approach consisting in using interior-point methods. A more complete numerical study and comparison with SDP feasibility solvers is beyond the scope of this paper. We compare our solver to SeDuMi [47], one of the best SDP solver under Matlab. We focus on two types of semidefinite feasibility problems: random SDP problems and polynomial SOS decomposition problems. The latter problems are generated with the Matlab toolbox GloptiPoly 3 [23].

Our experiments are carried out with Matlab 7.4 under Linux, running on a PC with Intel Pentium D CPU at 3.00Ghz equipped with 2GB of RAM. We use SDLS 1.2, SeDuMi 1.1R3 and GloptiPoly 3.3.

6.1 Random dense SDP instances

We generate random dense SDP feasibility problems with the following Matlab code:

```
m = 100; n = 100;
A = randn(m, n^2);
X = orth(randn(n));
X = X'*diag(rand(n,1))*X;
b = A*X(:);
c = [];
K = []; K.s = n;
```

Entries of matrix A are normally distributed random numbers (mean zero, standard deviation one). Vector b is generated in such a way that there exists a strictly feasible solution X with eigenvalues uniformly distributed between 0 and 1. Vector c is empty since we are only interested in finding a feasible point. Structure `K` has a field `K.s` indicating that entries in vector x correspond to a symmetric matrix X of size n in the SDP cone.

We solve the SDP problems with the following instructions:

```
pars = []; pars.eps = 1e-6;
[x,y] = sdls(A,b,c,K,pars);
[x,y] = sedumi(A,b,c,K,pars);
```

where the first instruction indicates the expected relative accuracy ε on solution x. The meaning is as follows:

- SDLS: relative accuracy on primal infeasibility residual: $\|Ax - b\| \leq \varepsilon(1 + \|b\|)$;
- SeDuMi: duality gap $c^\top x - b^\top y \leq \varepsilon$.

Other solver parameters are set to their default values.

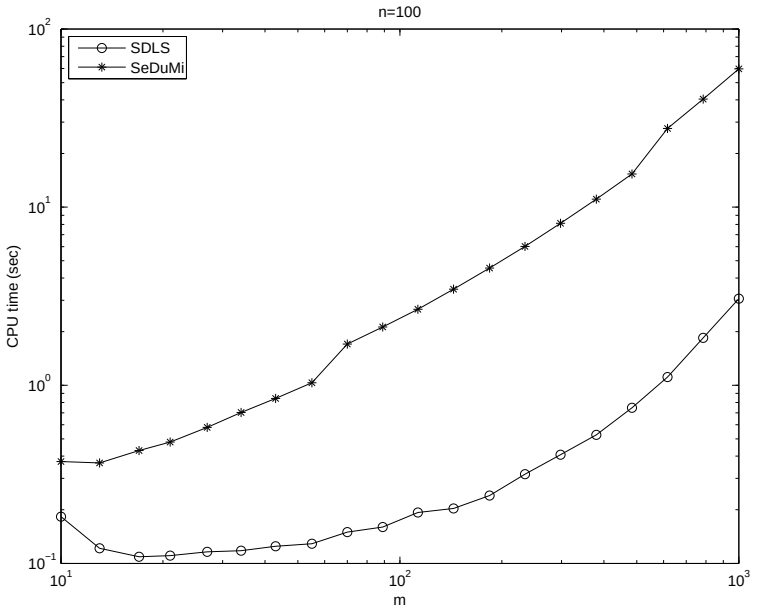

Figure 1: Comparative CPU times for random dense SDP feasibility problems with fixed cone size $n = 100$ and varying number of constraints m.

On Figure 1 we represent computational times required by SDLS and SeDuMi to output a solution satisfying the above accuracy constraints, when the size of matrix X is fixed to $n = 100$,

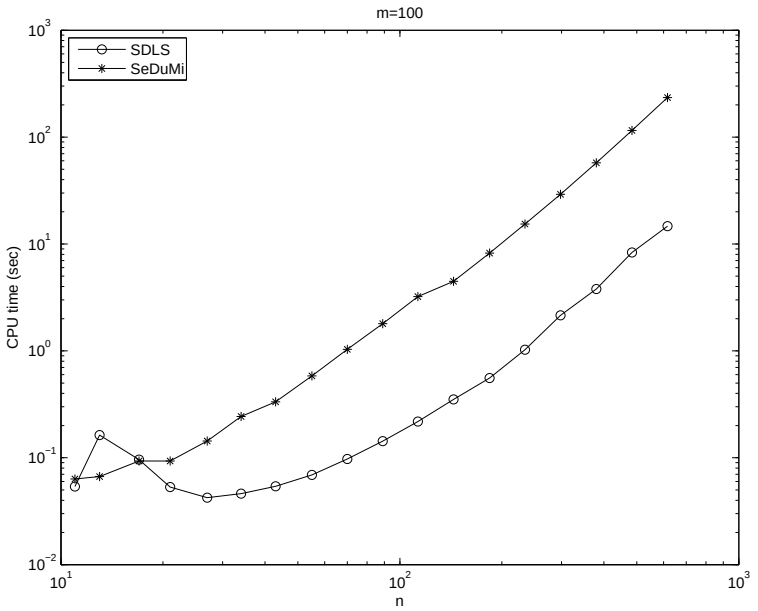

Figure 2: Comparative CPU times for random dense SDP feasibility problems with fixed number of constraints $m = 100$ and varying cone size n.

and the number of linear constraints m varies from 10 to 1000. We observe that both solvers have an experimental computational cost of $O(m^2)$, but SDLS is significantly faster than SeDuMi.

On Figure 2, the number of constraints is fixed to $m = 100$, and the SDP cone size n varies from 10 to 1000. Here too, we observe that both solvers have an experimental complexity between $O(n^2)$ and $O(n^3)$, but SDLS is significantly faster than SeDuMi, except for small problems ($n \leq 20$). For $n = 785$, for memory reasons, SeDuMi could not solve the problem on our computer. For $n = 1000$, SDLS failed for the same reason (a dense matrix A of this size requires 80MB of storage in IEEE double precision).

We also note that SDLS typically requires many iterations, but each iteration, as a first-order quasi-Newton step, is relatively cheap. In comparison, SeDuMi requires a few iterations, but as a second-order method, one iteration is often more expensive than a whole run of SDLS.

6.2 Random sparse SDP instances

Data sparsity is fully exploited in the linear algebra routines of SeDuMi, but not in SDLS. We test both solvers on random SDP instances with a sparse A matrix generated by the instruction:

```
A = sprandn(m,n^2,density);
```

where `density` is a number between 0 and 1, the expected ratio of nonzero entries over the total number of entries.

On Figure 3 we represent computation times required by SDLS and SeDuMi on sparse problems with $n = m = 200$ and a density ranging from 5% to 100%. We observe that SDLS does not really benefit from data sparsity, whereas the performance of SeDuMi (in terms of computation time) is significantly improved for sparse problems. Yet, for these examples, SDLS is always significantly faster than SeDuMi. Thus it performs well on sparse structured semidefinite feasibility problems, even though it does not exploit sparsity.

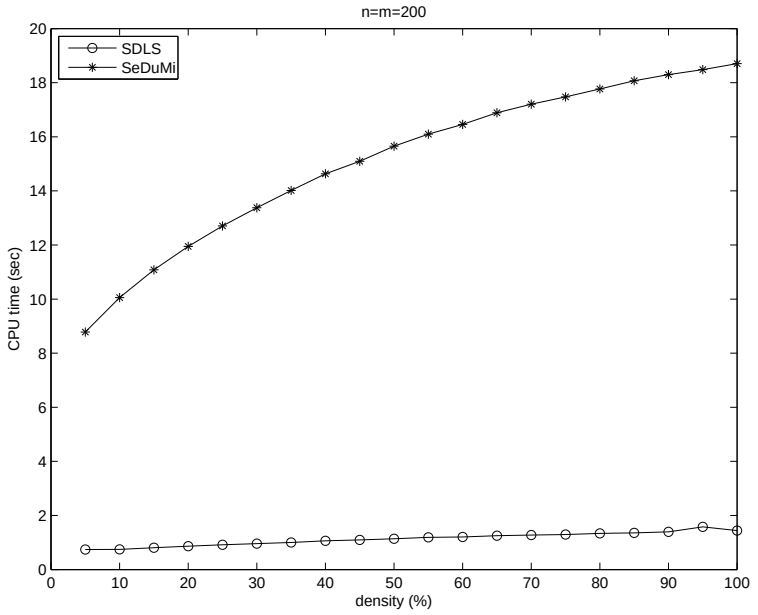

Figure 3: Comparative CPU times for random sparse SDP feasibility problems with fixed SDP cone size $n = 200$, fixed number of linear constraints $m = 200$ and density varying between 5% and 100%.

6.3 Infeasible SDP example

In the case of degenerate problems with empty interior, successive iterates produce an approximate dual Farkas vector (remind Section 3.3). Let us illustrate this observation on a simple example.

We consider problem (1) with data

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -\varepsilon \end{bmatrix}$$

with $\mathcal{K}$ the 2-by-2 SDP cone, and ε a small positive parameter. For (4) the optimal value is $-\infty$ and the dual problem has no solution.

When $\varepsilon = 10^{-6}$, the behavior of SDLS is as follows:

```
>> A=[1 1 0 0;0 0 0 1];b=[1;-1e-6];c=[];K=[];K.s=2;
>> [x,y]=sdls(A,b,c,K);
(...)
SDLS suspects infeasibility
>> x'
ans =
   1.0e+10 *
    1.4629    0.0000    0.0000    0.0000
>> y'
ans =
   1.0e+26 *
    0.0000   -4.2802
```

The vector y gives an approximate Farkas direction. First, the vector y can be normalized, for example as $y/\|y\|$. It turns out that it is interesting to consider

$$y_F := y \,/\, b^\top y.$$

We observe indeed that y_F is an approximation of Farkas vector, in the sense that $\|x(y_F)\|$ is small (remind Lemma 3.4(ii)). Moreover this norm can be computed directly from previous calculations: the properties of the projection yields

$$x(y_F) = \mathrm{P}_K(A^\top y / b^\top y) = x(y)/b^\top y,$$

and thus

$$\|x(y_F)\| = \frac{\|x(y)\|}{|b^\top y|}.$$

Both terms of the quotient are computed when computing the value of dual function (remind (9)). In the illustrative example of this section, we have

```
>> norm(x/(b'*y))
ans =
   3.4179e-11
```

The above argumentation is valid for any problem with $c = 0$. (Note that in this case $b \neq 0$, otherwise $X = 0$ is an obvious solution). So we add in SDLS a test of the form

$$\|x(y_k)\|^2 \leq n\varepsilon(b^\top y_k)^2$$

when infeasibility is suspected.

When $\varepsilon = 10^{-9}$, the behavior of SDLS is as follows:

```
>> A=[1 1 0 0;0 0 0 1];b=[1;-1e-9];c=[];K=[];K.s=2;
>> [x,y]=sdls(A,b,c,K);
(...)
SDLS suspects marginal feasibility
>> x'
ans =
    1.0694    0.0000    0.0000    0.0000
>> y'
ans =
   1.0e+08 *
    0.0000   -6.9104
>> norm(x/(b'*y))
ans =
    0.3779
```

which illustrates that in this case, SDLS suspects that the problem is almost feasible.

6.4 Random SOS SDP instances

In this section we compare the performance of SDLS and SeDuMi on SDP problems coming from polynomial SOS decompositions. The SDP problems are generated with GloptiPoly 3 as follows:

```
p = 3; d = 2;
mpol('z',p,1);
P = msdp(min((z'*z)^d));
[A,b,c,K] = msedumi(P);
A = [c';-A]; c = [];
n = K.s; m = size(A,1);
X = orth(randn(n));
X = X'*diag(rand(n,1))*X;
b = A*X(:);
```

In the above script, `p` denotes the number of variables of the polynomial and `d` is equal to half its degree (`p=3` and `d=2` hence corresponds to a trivariate quartic). The matrix A generated with this code has a quasi-Hankel sparsity pattern typical of SOS SDP problems. Vector b is generated in such a way that there exists a strictly feasible solution X, and therefore that there exists a corresponding strictly positive SOS polynomial.

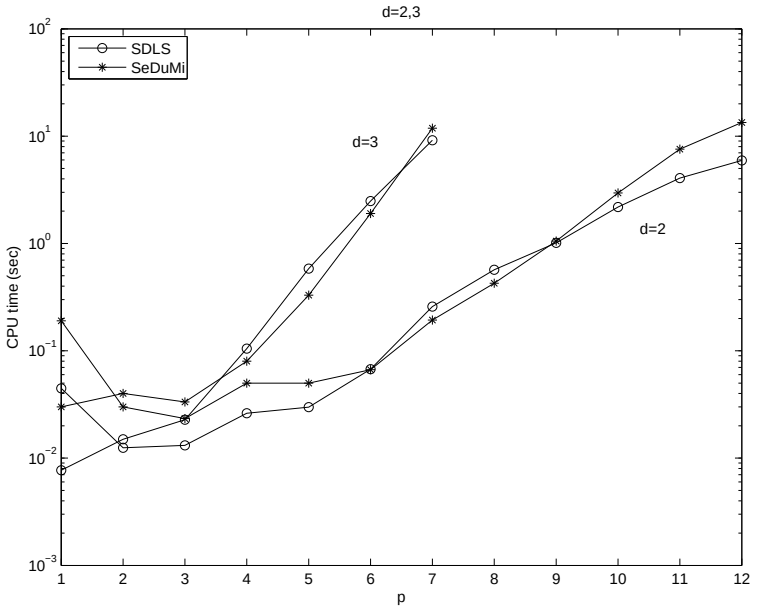

Figure 4: Comparative CPU times for random SOS SDP feasibility problems with fixed half-degree $d = 2, 3$ and number of variables p varying between 1 and 12.

On Figure 4 we represent computation times required by SDLS and SeDuMi when d is fixed and p varies. Only values of d and p corresponding to problems with $m \leq 2000$ are considered. On Figure 5, p is fixed and d varies under the same size constraints. We observe that SDLS and SeDuMi perform similarly for these specialized sparse SDP problems.

Note that SDLS has no preconditioning process, and it does not exploit either the special structure of the problem nor the sparsity parttern. There is room for improvement in the direction of designing a special implementation of SDLS for a target problem. For example, a more evolved version of SDLS designed for SOS would require an algorithm to compute eigendecomposition for quasi-Hankel matrices. We are not aware of such an algorithm.

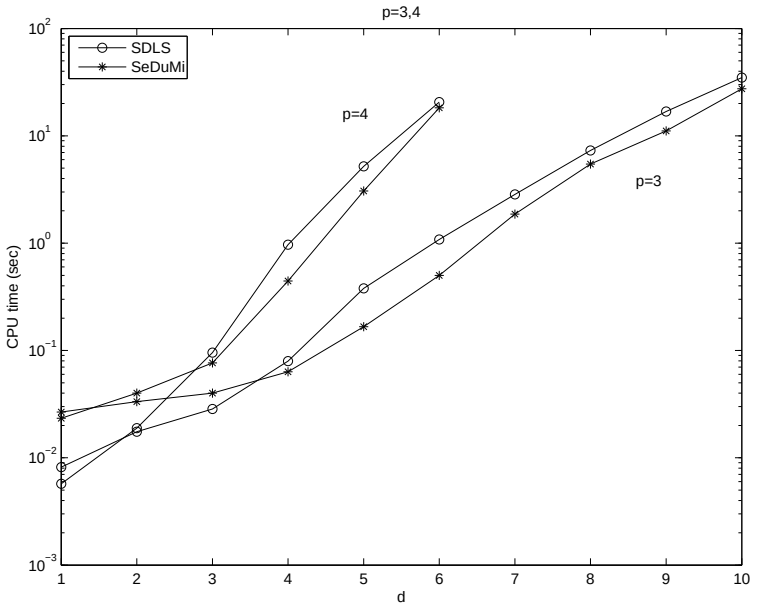

Figure 5: Comparative CPU times for random SOS SDP feasibility problems with fixed number of variables $p = 3, 4$ and half-degree d varying between 1 and 10.

6.5 Regularization for SOS SDP problems

To illustrate the regularization for SOS SDP feasibility problems proposed in Section 4.2, we consider the bivariate quartic polynomial

$$p(t) = (1 - t_1 t_2)^2 + {t_1}^2$$

which is a sum of two squares. The corresponding SDP feasibility problem (1) can be generated with the following GloptiPoly commands:

```
mpol t 2
p=(1-t(1)*t(2))^2+t(1)^2;
P=msdp(min(p));
[A,b,c,K,b0]=msedumi(P);
A=[c';-A];b=-[b0;b];c=[];
```

Matrix A has size 14-by-36, and $\mathcal{K}$ is the cone of 6-by-6 positive semidefinite matrices. The behavior of SDLS on this problem is as follows:

```
>> [x,y]=sdls(A,b,c,K);
(...)
bfgs: iter 197: step = 1.0e+00, f = -2.49911, gnorm = 5.7e-04
bfgs: iter 198: step = 1.0e+00, f = -2.49912, gnorm = 2.6e-04
bfgs: iter 199: step = 1.0e+00, f = -2.49914, gnorm = 7.1e-04
bfgs: iter 200: step = 1.0e+00, f = -2.49915, gnorm = 9.2e-04
bfgs: 200 iterations reached, f = -2.49915, gnorm = 9.2e-04
SDLS suspects marginal feasibility
>> yf=y/(b'*y);norm(yf)
ans =
```

```
   7.8996e+06
>> xf=x/(b'*y);norm(xf)
ans =
    0.1227
```

The maximum number of iterations (default = 200) is reached and SDLS failed to reduce the gradient norm below the threshold (default = 10^{-6}). From the computed quantities $\|y_F\|$ and $\|x_F\|$ (see the definitions in the previous section) infeasibility cannot be suspected, and hence the Slater assumption is likely to be violated, i.e. there may be no interior point to the SDP feasibility problem.

To avoid this, we proposed in Section 4.2 a regularization process for SOS decomposition problems, that consists in testing positivity of the perturbed polynomial $p_\varepsilon(t) = p(t) + \varepsilon q(t)$ with $q(t)$ a strictly SOS polynomial, and ε a small positive number. With this regularization, SDLS behaves much better, even with a quite small ε. Consider for example the following numerical example with $\varepsilon = 10^{-8}$. Here we build the perturbed polynomial with the help of

$$q(t) = \|[1, t_1, t_2, {t_1}^2, t_1t_2, {t_2}^2]\|^2 = 1 + {t_1}^2 + {t_2}^2 + {t_1}^4 + {t_1}^2{t_2}^2 + {t_2}^4.$$

The GloptiPoly commands to generate the associated SDP feasibility problem are:

```
q=mmon(t,2);q=q'*q;
e=1e-8;
P=msdp(min(p+e*q));
[A,b,c,K,b0]=msedumi(P);
A=[c';-A];b=-[b0;b];c=[];
```

The behaviour of SDLS is as follows:

```
>> [x,y]=sdls(A,b,c,K);
(...)
bfgs: iter 130: step = 1.0e+00, f = -2.48887, gnorm = 6.6e-06
bfgs: iter 131: step = 1.0e+00, f = -2.48887, gnorm = 4.4e-06
bfgs: iter 132: step = 1.0e+00, f = -2.48887, gnorm = 7.4e-07
bfgs: gradient norm below tolerance, quit at iteration 132, f = -2.48887
SDLS finds a feasible point
```

We finish by illustrating the sensitivity of the behaviour with respect to ε, showing that, as expected, SDLS has more troubles when ε gets smaller. We build the perturbed polynomial $p_\varepsilon(t)$ for various values of the regularization parameter ε. On Figure 6 we represent the number of BFGS iterations as a function of parameter ε. On Figure 7 we represent the norm of the last iterate y of SDLS as a function of parameter ε.

6.6 Concluding remarks

The paper investigates a new approach to solve conic feasibility problems. The idea is to project a point onto the feasible region; the structure of this least-squares problem permits to develop a family of dual methods to solve them (see Algorithm 2.1). This approach enjoys nice geometrical features and interesting theoretical properties - in particular when the Slater assumption (existence of a strictly feasible point) is in force.

We have presented a simple Matlab implementation of one instance of the family to assess the approach. We call this software SDLS for semidefinite least-squares. First numerical experiments illustrate that our implementation is competitive with state-of-the-art interior-point algorithms,

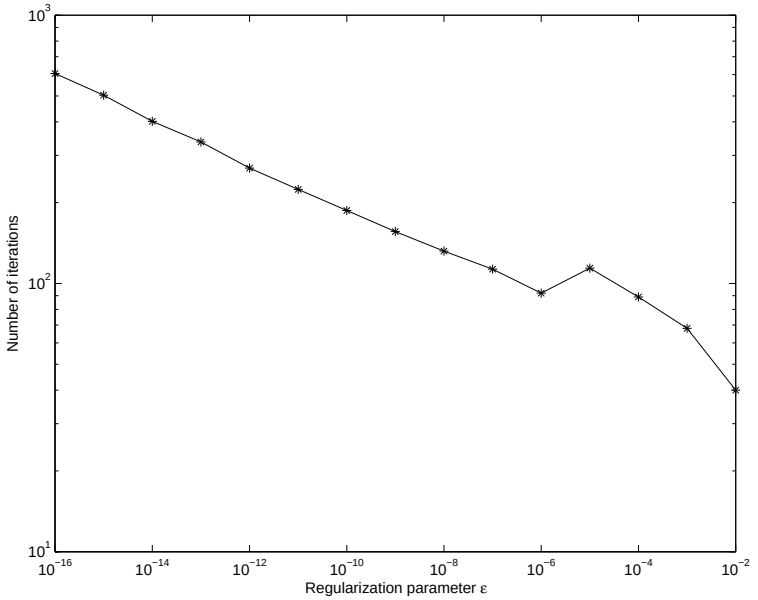

Figure 6: Number of BFGS iterations as a function of regularization parameter ε for an SOS SDP problem with no interior point.

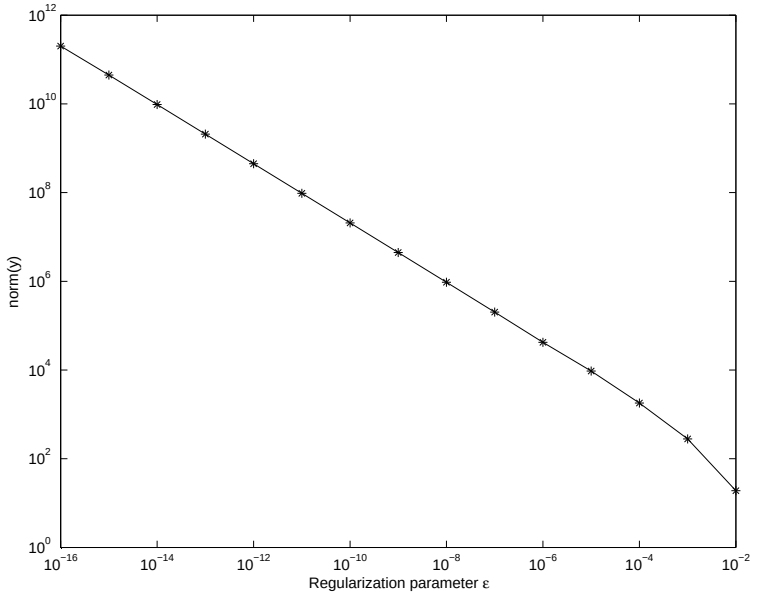

Figure 7: Norm $\|y\|$ of final iterate as a function of regularization parameter ε for an SOS SDP problem with no interior point.

such as the infeasible primal-dual method of SeDuMi, for a relatively small expected relative accuracy. One should bear in mind however that SeDuMi is a sophisticated software consisting of highly optimized compiled code, whereas our implementation in SDLS consists of approximately 50 lines of interpreted Matlab code (excluding input argument parsing and comments) using only Matlab's built-in numerical linear algebra functions, and an external public-domain optimization package.

Moreover, projection algorithms following the scheme of Algorithm 2.1 are still under development; recent algorithms of that form [43, 5] use Newton-like to solve the dual problem and they appear to be very efficient. So we believe that more evolved versions of SDLS may still give better numerical results for conic feasibility problems. Obviously, it is not the purpose of this paper to compare different software to solve projection problems. Here we presented a new methodology based on projections for solving the important conic feasibility problems, and we gathered tokens that this projection approach could be considered as an alternative to interior points methods in this context.

Instructions for reviewers

Reviewers of this paper are invited to download SDLS 1.2 and its documentation at the following address

`http://www.laas.fr/~henrion/software/sdls/sdls1.2.tar.gz`

Alternatively, they may contact the Editor or Associate Editor who should have the software files.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J. Opt., 5(1):13-51, 1995.

[2] H. H. Bauschke and J. M. Borwein. On the convergence of von Neumann's alternating projection algorithm for two sets. Set Valued Analysis, 1(2):185-212, 1993.

[3] A. Ben-Tal and A. Nemirovski. Lectures on modern convex optimization. SIAM, 2001.

[4] J. F. Bonnans, J. Ch. Gilbert, C. Lemaréchal and C. Sagastizábal. Numerical optimization. Springer, 2003.

[5] R. Borsdorf and N. Higham. A preconditioned Newton algorithm for the nearest correlation matrix. Preprint, 2008.

[6] S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan. Linear matrix inequalities in system and control theory. SIAM, 1994.

[7] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge Univ. Press, 2005.

[8] S. Boyd and L. Xiao. Least-squares covariance matrix adjustment. SIAM J. Matrix Analysis Appl., 27(2):532-546, 2005.

[9] B. D. Craven and J. J. Koliha. Generalization of Farkas' theorem. SIAM J. Math. Anal., 8(6):983-997, 1977.

[10] B. Dumitrescu. Positive trigonometric polynomials and signal processing applications. Springer, 2007.

[11] R.L. Dykstra. An algorithm for restricted least-square regression. J. Amer. Stat. Assoc., 78(384):837-842, 1983.

[12] P. Gahinet and A. Nemirovski. The projective method for solving linear matrix inequalities. Math. Prog. 77:163-190, 1997.

[13] J.V. Burke, A.S. Lewis and M.L. Overton. HANSO: hybrid algorithm for nonsmooth optimization, a Matlab package based on BFGS, bundle and gradient sampling methods. 2006. www.cs.nyu.edu/overton/software/hanso

[14] J. V. Burke, D. Henrion, A. S. Lewis and M. L. Overton. HIFOO: a Matlab package for fixed-order controller design and H-infinity optimization. Proc. IFAC Symp. Robust Control Design, Toulouse, France, 2006. www.cs.nyu.edu/overton/software/hifoo

[15] M.D. Choi, T.Y. Lam and B. Reznick. Sums of squares of real polynomials. Proc. Symp. Pure Math., 58:103-126, Amer. Math. Soc., 1995.

[16] F. Deutsch. Best approximation in inner product spaces. Springer, 2001.

[17] L. El Ghaoui and P. Gahinet. Rank minimization under LMI constraints: a framework for output feedback problems. Proc. Europ. Control. Conf., Groningen, The Netherlands, 1993.

[18] M. Fazel. Matrix rank minimization with applications. Ph.D. thesis, Stanford Univ., 2002.

[19] J. W. Helton and M. Putinar. Positive polynomials in scalar and matrix variables, the spectral theorem and optimization. In: M. Bakonyi, A. Gheondea, M. Putinar, J. Rovnyak (Editors). Operator theory, structured matrices and dilations: Tiberiu Constantinescu memorial volume. Theta Foundation, Bucharest, 229-306, 2007.

[20] D. Henrion and J.B. Lasserre. GloptiPoly: global optimization over polynomials with Matlab and SeDuMi. ACM Trans. Math. Software, 29(2):165-194, 2003. www.laas.fr/∼henrion/software/gloptipoly2

[21] D. Henrion and A. Garulli (Editors). Positive polynomials in control. LNCIS 312, Springer Verlag, 2005.

[22] D. Henrion and J. Malick. SDLS: a Matlab package for solving conic least-squares problems. 2007. www.laas.fr/∼henrion/software/sdls

[23] D. Henrion, J.B. Lasserre and J. Löfberg. GloptiPoly 3: moments, optimization and semidefinite programming. 2007. www.laas.fr/∼henrion/software/gloptipoly3

[24] N. Higham. Computing a nearest symmetric positive semidefinite matrix. Lin. Alg. Appl., 103:103–118, 1988.

[25] N. Higham. Computing a nearest symmetric correlation matrix - a problem from finance. IMA J. Numer. Analysis, 22(3):329-343, 2002.

[26] J. B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms. Vol. 1 and 2, Springer, 1993.

[27] F. Jarre and F. Rendl. An augmented primal-dual method for linear conic problems. Preprint, 2007.

[28] J.B. Lasserre. Global optimization with polynomials and the problem of moments. SIAM J. Optim., 11(3):796–817, 2001.

[29] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In: M. Putinar, S. Sullivant (Editors). Emerging applications of algebraic geometry. IMA Vol. Math. Appli., 149:157-270, Springer, 2009.

[30] A. Lewis and J. Malick. Alternating projections on manifolds. Math. Oper. Research, 33(1):216-234, 2008.

[31] A. Lewis, R. Luke and J. Malick. Local convergence of nonconvex averaged and alternating projections. Preprint, 2008.

[32] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. Preprint, 2008.

[33] J. Malick. A dual approach to semidefinite least-squares problems. SIAM J. Matrix Analysis Appl., 26(1):272-284, 2004.

[34] H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. Math. Prog. 95(2):407-430, 2003.

[35] B.N. Mordukhovich. Variational analysis and generalized differentiation. Vol. 1 and 2, Springer, 2006.

[36] Y. Nesterov. Smooth minimization of nonsmooth functions. Mathematical Programming. 103(1):127-152, 2005.

[37] Y. Nesterov. Squared functional systems and optimization problems. In: J. Frenk, C. Roos, T. Terlaky and S. Zhang (Editors). High Performance Optimization. Kluwer, 405-440, 2000.

[38] J. Nocedal and S.J. Wright. Numerical optimization. Springer, 1999.

[39] R. Orsi, U. Helmke and J. B. Moore. A Newton-like method for solving rank constrained linear matrix inequalities. Automatica, 42(11):1875-1882, 2006.

[40] P. A. Parrilo. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Ph.D. Thesis, California Inst. Tech., 2000.

[41] H. Peyrl, P. A. Parrilo. Computing sum of squares decompositions with rational coefficients. Theoretical Computer Science, 409(2):269-281, 2008.

[42] V. Powers and T. Wörmann. An algorithm for sums of squares of real polynomials. J. Pure Appl. Algebra, 127:99-104, 1998.

[43] H. Qi and D. Sun. Quadratic convergence and numerical experiments of Newton's method for computing the nearest correlation matrix. SIAM J. Matrix Analysis Appl. 28:360-385, 2006.

[44] M. Ait Rami, U. Helmke and J.B. Moore. A finite step algorithm for solving convex feasibility problems J. Glob. Optim. 38:143-160, 2007.

[45] R.T. Rockafellar. Convex analysis. Princeton Univ. Press, 1970.

[46] A. Schrijver. Theory of linear and integer programming. Wiley, 1986.

[47] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Opt. Methods and Software, 11-12:625-653, 1999.

[48] L. Tunçel and H. Wolkowicz. Strong duality and minimal representation for cone optimization. Preprint, 2008.

[49] R. H. Tütüncü, K. C. Toh and M. J. Todd. Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems. Pacific J. Optimization, 3:135-164, 2007.

[50] Z. Wang, S. Zheng, Y. Ye and S. Boyd. Further relaxations of the SDP approach to sensor networks localization. SIAM J. Opt., 19(2):655-673, 2008.

[51] H. Wolkowicz, R. Saigal and L. Vandenberghe. Handbook of semidefinite programming. Kluwer, 2000.