On Handling Free Variables in Interior-Point Methods for Conic Linear Optimization

Miguel F. Anjos* Samuel Burer†

August 2006 (Revised March 2007)

Abstract

We revisit a regularization technique of Mészáros for handling free variables within interior-point methods for conic linear optimization. We propose a simple computational strategy, supported by a global convergence analysis, for handling the regularization. Using test problems from benchmark suites and recent applications, we demonstrate that the modern code SDPT3 modified to incorporate the proposed regularization is able to achieve the same or significantly better accuracy over standard options of splitting variables, using a quadratic cone, and solving indefinite systems.

Keywords: Infeasible primal-dual path-following algorithm, semidefinite programming, equality constraints, free variables, regularization.

*Department of Management Sciences, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research partially supported by Discovery Grant 312125 and RTI Grant 314668 from the Natural Sciences and Engineering Research Council of Canada. Email `anjos@stanfordalumni.org`

†Department of Management Sciences, University of Iowa, Iowa City, IA, 52242-1994, U.S.A. Research partially supported by NSF Grants CCR-0203426 and CCF-0545514. Email `samuel-burer@uiowa.edu`

1 Introduction

Conic linear optimization, and in particular semidefinite optimization, has arisen since the early 1990s as an increasingly powerful and useful technique for tackling a variety of problems arising from both applications and theory. We refer the reader to the SDP webpage of Helmberg (see References) as well as the books of de Klerk (2002) and Wolkowicz et al. (2000) for a thorough coverage of the theory and algorithms in this area, as well as of several application areas where researchers in conic linear optimization have made significant contributions.

Conic linear optimization refers to the class of optimization problems where a linear function of a variable x is optimized subject to linear constraints on the elements of x and the additional constraint that x lie in a symmetric self-dual cone. This includes linear programming (LP) problems as a special case, namely when the cone is the non-negative orthant. Since all of these cones can be described as a conic section of the cone of positive semidefinite matrices (in a polynomially bounded dimension, see Faraut and Korányi (1994)), attention has focused particularly on the development of algorithms for solving semidefinite linear optimization, commonly referred to as semidefinite programming (SDP). Beyond LP and SDP, a third specific cone that is useful in applications is the second-order (or Lorentz) cone, which gives rise to second-order cone programming (SOCP).

As a result, a variety of algorithms for solving LP, SOCP, and SDP problems, including polynomial-time infeasible path-following interior-point methods (IPMs), have been implemented and benchmarked (see e.g. Mittelmann (2003)), and several excellent solvers are available. Two of these solvers handle LP, SOCP, and SDP in a unified way, namely SeDuMi (Sturm, 1999) and SDPT3 (Tütüncü et al., 2001).

Notwithstanding the substantial progress made in recent years, work continues on methods and software for conic linear optimization. One outstanding issue is that of handling free variables.

Handling free variables in conic linear optimization is an important modeling and algorithmic issue for IPMs. The issue arises from the fact that nearly all theory and algorithms for IPMs are based on the following standard-form primal and dual problems:

$$\min\left\{c^T x : Ax = b,\ x \in K\right\} \qquad \text{and} \qquad \max\left\{b^T y : A^T y + s = c,\ s \in K\right\},$$

where K is the symmetric self-dual cone (specifically, a direct product of linear, semidefinite, and second-order cones). However, in many applications, free variables naturally appear in the primal. Examples of such applications occur in quantum chemistry (Zhao et al. (2004)), polynomial optimization problems (Parrilo (2003); Lasserre (2000/01)), and combinatorial optimization problems (Lasserre (2002); Anjos (2005)), among others.

Allowing free variables, the (non-standard) primal problem is

$$\min\left\{c^T x + g^T z : Ax + Ez = b,\ x \in K\right\} \tag{1}$$

with corresponding dual problem

$$\max\left\{b^T y : A^T y + s = c,\ E^T y = g,\ s \in K\right\}. \tag{2}$$

We denote by (n, p, m, n) the dimensions of the vectors (x, z, y, s), respectively, which determines the sizes of the data (A, b, c, E, g).

Theoretically, it is not so difficult to extend standard-form IPMs to handle (1–2). However, computation is not so easy. Each iteration of IPMs for the standard-form problem is based on solving a positive definite linear system, and accordingly most codes use high-quality, fast, sparsity-preserving implementations of the Cholesky factorization. When free variables are present, the corresponding system is still invertible but becomes indefinite, which makes it more difficult to solve in a quick, stable fashion. We emphasize that free variables are a computational issue, not a theoretical one.

Researchers have attempted various alternative ways to handle free variables computationally. Each method can be viewed as an attempt to enable the use of the Cholesky factorization.

Probably the simplest and most often suggested method is to split the variable z into a difference $z^+ - z^-$ of nonnegative vectors $z^+ \geq 0$ and $z^- \geq 0$ which transforms (1–2) into standard form. The effect in the dual is that the equality $E^T y = g$ is split into $E^T y \geq g$ and $E^T y \leq g$. In part because of its simplicity, this approach is often implemented as the default behavior for interior-point methods (e.g., in the LP code LIPSOL (Zhang, 1999) that is the basis of Matlab's large-scale LP solver). For interior-point methods, this splitting of variables can be problematic because the primal optimal solution set becomes unbounded and the dual feasible set has no interior. Empirically, a typical behavior is that z^+ and z^- individually become unbounded, while their difference z stays bounded. For LP, Wright (1997) asserts that, for methods which achieve superlinear convergence, the tendency of z^+ and z^- to grow large is mitigated in a satisfactory manner. Also, an alternative way to handle the splitting of variables in LP that leads to the solution of a symmetric quasi-definite system (Vanderbei (1995)) is outlined in Vanderbei (2001). However, the recent results in Waki et al. (2004) and Kobayashi et al. (2005) suggest that the degeneracy caused by splitting free variables in SDP problems makes it difficult to solve the resulting SDPs stably and/or highly accurately.

A quick overview of other alternative methods for handling free variables is as follows:

- One can convert to standard form by eliminating z in the equations $Ax + Ez = b$. However, structural properties of (A, E) such as sparsity tend to be destroyed by such an approach. In the context of SDP, Kobayashi et al. (2005) consider this approach and in particular make efforts to manage the loss of sparsity by eliminating z via different bases.

- One can convert to standard form by adding an auxiliary scalar variable z_0 and requiring that (z_0, z) be in a second-order cone. To the best of our knowledge, this was first suggested by Andersen (2002), who reports good results and improved accuracy when solving SOCPs. This approach is also available as an option in the most recent release of SeDuMi, version 1.1 (see Pólik (2005)).

- One can handle the free variables directly and regularize the indefinite system faced at each iteration, i.e., make the system symmetric quasi-definite by perturbing it in a controlled way. This approach was suggested by Mészáros (1998) in the context of LP. Beyond permitting the use of the Cholesky factorization, another advantage of regularization is that the structure of (A, E) is not destroyed. On the other hand, the downside of this approach is that the solution of the system is also perturbed and care must be taken to ensure that the global convergence of the method is not negatively affected.

Finally, we reiterate that one certainly still has the option to handle the free variables directly and solve the indefinite systems (equation (7) below). In fact, this is the default option of SDPT3 version 3.02, ostensibly because the authors of the software found this approach better than, say, splitting variables.

It seems safe to say that no consensus has been reached on how to handle free variables in all cases. Indeed, it is our opinion (and that of Kobayashi et al. (2005)) that solving general conic linear optimization problems with free variables in a reliably stable and accurate manner remains a relevant research topic.

In this paper, we revisit the regularization method of Mészáros (1998) and formalize a strategy for handling and updating the regularization so that global convergence is not affected. In contrast to the strategy suggested by Mészáros, which is based on iterative refinement and is somewhat *ad hoc*, our strategy is supported by a global convergence result. Using the code SDPT3, we illustrate the effectiveness of our regularization strategy on a diverse collection of problems. Our approach achieves the same or significantly better accuracy over the approaches of splitting variables, using a quadratic cone, and solving indefinite systems.

1.1 Some remarks

A few remarks are in order. First, our intention in this paper is not to claim that regularization is the best in all situations. Indeed, this is most likely *not* the case. Instead, we simply hope to establish that regularization, properly handled, is a viable alternative to other methods for handling free variables. (One consequence is that we have chosen not to compare with the method of eliminating z as in Kobayashi et al. (2005) in part because careful comparisons are given in Kobayashi et al. (2005) and because we prefer to maintain the structure of (A, E).)

Second, there are several different publicly available codes for LP, SOCP, and/or SDP on which we could test the regularization. Ultimately, we have chosen to test SDPT3 for several reasons: we wished to test both SOCP and SDP, which SDPT3 can handle; SDPT3's algorithm matches the algorithmic framework of our analysis very closely; and SDPT3's code is easily accessible, customizable, and verifiable in Matlab.

Finally, significant variation between different codes makes it unclear whether regularization would have the same effect within all codes as it does with SDPT3. For example, tests that we have performed indicate that split variables perform quite well within SeDuMi. This is to be expected because, by design, the homogeneous self-dual embedding model used by SeDuMi has a bounded optimal solution set, and thus the iterates cannot diverge to infinity. Hence, SeDuMi will not suffer additional numerical instabilities from the splitting; in fact, split variables perform so well in SeDuMi that there does not appear to be much room for improvement in SeDuMi using regularization. In our opinion, these code-by-code differences make the discussion of free variables richer.

1.2 Structure of this paper

This paper is structured as follows. In Section 2, we recall the basic framework of infeasible primal-dual path-following algorithms. In Section 3, we summarize the key ideas behind a global

convergence result of Kojima et al. (1993), and in Section 4, we recall the regularization approach originally proposed by Mészáros (1998). In Section 5 we propose a specific methodology to update the regularization at each iteration, and extend the analysis of Kojima et al. (1993) to show that the resulting infeasible primal-dual path-following method is globally convergent. In Section 6, we report computational results which show that the proposed regularization leads to an overall improvement in the performance of SDPT3 for instances with free variables. Finally, Section 7 summarizes our findings and mentions some possible directions for future research.

2 The Basic Infeasible Primal-Dual Path-Following Framework

In this section, we recall the basic framework of infeasible primal-dual path-following algorithms, which is implemented in nearly all interior-point codes for conic linear optimization. Even though standard texts treat the standard-form problem, we state the framework with respect to the problems (1–2). To recover the standard-form framework, one can simply take $p = 0$.

For simplicity, the framework (and indeed all the results in the paper) are stated with K expressed as a linear cone, i.e., the SOCP and SDP cases are not explicitly handled. By now, it is well-known that all standard convergence results for LP can be extended to SOCP and SDP. With this in mind and in hopes of keeping this paper as clean and accessible as possible, we choose to state everything in terms of LP.

Without loss of generality, we make the standard assumptions that A has full-row rank and E has full-column rank. We also assume that both (1–2) are interior feasible so that strong duality holds. Strong duality occurs when both primal and dual attain their optimal values with no gap, i.e., there exists a primal-dual feasible point (x, z, y, s) such that $\mu(x, s) = 0$, where $\mu(x, s) := x^T s/n$ is the (scaled) duality gap.

A consequence of these assumptions is that, for all $\nu > 0$, the system

$$\begin{aligned} Ax + Ez &= b && \text{(3a)}\\ A^T y + s &= c && \text{(3b)}\\ E^T y &= g && \text{(3c)}\\ XSe &= \nu e && \text{(3d)}\\ (X, S) &\in K^0 \times K^0 && \text{(3e)} \end{aligned}$$

has a unique solution, which we write as $(x_\nu, z_\nu, y_\nu, s_\nu)$. (We adopt common notation in the field of IPMs, so that $X := \mathrm{Diag}(x)$, $S := \mathrm{Diag}(s)$, $K^0 := \mathrm{int}(K)$, and e denotes the vector of all ones.) The set $\mathcal{C} := \{(x_\nu, z_\nu, y_\nu, s_\nu) : \nu > 0\}$ is called the *central path*, and is a smooth trajectory that converges to the primal-dual optimal solution set as $\nu \to 0$ (note, for example, that $\mu(x_\nu, s_\nu) = \nu$).

Given an initial point (x^1, z^1, y^1, s^1) — which is not necessarily primal-dual feasible but does satisfy $(x^1, s^1) \in K^0 \times K^0$ — the k-th iteration of the path-following framework attempts to solve

(3) for some $\nu_k \in (0, \mu(x^k, s^k))$ by taking a step via Newton's method. More specifically, the system

$$\begin{aligned} A\Delta x^k + E\Delta z^k &= r_p^k && (4a)\\ A^T\Delta y^k + \Delta s^k &= r_{d_1}^k && (4b)\\ E^T\Delta y^k &= r_{d_2}^k && (4c)\\ S^k\Delta x^k + X^k\Delta s^k &= r_c^k, && (4d) \end{aligned}$$

where

$$\begin{aligned} r_p^k &:= b - Ax^k - Ez^k && (5a)\\ r_{d_1}^k &:= c - A^Ty^k - s^k && (5b)\\ r_{d_2}^k &:= g - E^Ty^k && (5c)\\ r_c^k &:= \nu_k\, e - X^kS^ke, && (5d) \end{aligned}$$

is solved for $(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$, and a step-size $\alpha_k \in (0, 1]$ is selected such that

$$(x^{k+1}, z^{k+1}, y^{k+1}, s^{k+1}) := (x^k, z^k, y^k, s^k) + \alpha_k(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$$

satisfies $\mu(x^{k+1}, s^{k+1}) < \mu(x^k, s^k)$ and $(x^{k+1}, s^{k+1}) \in K^0 \times K^0$. By construction,

$$(r_p^{k+1}, r_{d_1}^{k+1}, r_{d_2}^{k+1}) = (1 - \alpha_k)(r_p^k, r_{d_1}^k, r_{d_2}^k). \tag{6}$$

In other words, one can interpret a single iteration as decreasing the duality gap and decreasing infeasibility, while staying inside the cone. (An important technical issue is whether (4) is uniquely solvable. This is guaranteed by $(x^k, s^k) \in K^0 \times K^0$; see also below.)

Various implementations of the above basic framework are possible. For example, many implementations do not monitor the decrease of μ since, in practice, a decrease is typically observed all the way to the boundary of $K^0 \times K^0$. Most implementations also take different step-sizes in the primal and dual spaces. Another popular variant is the predictor-corrector strategy of Mehrotra (1992), which provides a highly effective scheme for choosing ν_k and for altering $(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$ so that the central path is followed more closely.

The Newton system can be reduced to the following smaller system (where the superscript k is understood):

$$\begin{pmatrix} AXS^{-1}A^T & E \\ E^T & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} AXS^{-1}r_{d_1} - AS^{-1}r_c + r_p \\ r_{d_2} \end{pmatrix}. \tag{7}$$

If $p > 0$ (i.e., if there are free variables), then this system is indefinite. On the other hand, if $p = 0$, then the system is positive definite and can be solved with the Cholesky factorization.

3 A Basic Global Convergence Analysis

As discussed in Section 2, the infeasible primal-dual path-following framework reduces both the gap and the infeasibility in each iteration. Global convergence is achieved if the gap and infeasibility

converge to zero. In this section, we recapitulate the first global convergence result, which was given by Kojima et al. (1993) (for the case of LP with no free variables). This will serve as the basis of results in Section 5. (Some comments on why we have chosen to present the approach of Kojima et al. are given below in Section 3.2.)

We caution the reader that we do not present Kojima et al.'s method verbatim, but instead we make some simplifying assumptions. The simplifications are for the sake of brevity; all the fundamental content is retained. For example, Kojima et al. analyze the use of different primal and dual step-sizes, whereas we analyze a common step-size. Another simplification is the following assumption: the initial point (x^1, z^1, y^1, s^1) satisfies $(r_p^1, r_{d_1}^1) = (0, 0)$ so that $(r_p^k, r_{d_1}^k) = (0, 0)$ for all k. Our reasons for this assumption are as follows: the essentials of the global convergence result are clear with only one infeasible equation; and the assumption $r_{d_2}^k \neq 0$ is sufficient to develop the techniques of Section 5. In accordance with this assumption, we will write $r^k := r_{d_2}^k$ to streamline notation. Also to make notation easier, we let $\mu_k := \mu(x^k, s^k)$.

Within the framework of Section 2, convergence results typically require some restrictions on the iterates. Kojima et al. require that the iterates remain in a neighborhood of the central path having the following form, which is dependent on constants $\gamma \in (0, 1)$ and $\beta > 0$:

$$\mathcal{N}(\gamma, \beta) := \left\{ \; (x, z, y, s) \in K^0 \times \Re^p \times \Re^m \times K^0 \;\; : \;\; XS\, e \geq \gamma\, \mu(x, s)\, e, \;\; \|r(y)\| \leq \beta\, \mu(x, s) \; \right\}, \quad (8)$$

where $r(y) := g - E^T y$. In particular, γ and β should be chosen so that $(x^1, z^1, y^1, s^1) \in \mathcal{N}(\gamma, \beta)$. At times we will write $\mathcal{N} := \mathcal{N}(\gamma, \beta)$ for convenience. The neighborhood aids the convergence analysis by guaranteeing that the iterates do not get too close to the cone boundary and that infeasibility decreases at the same rate as the duality gap.

Algorithm 1 Infeasible Path-Following Algorithm

Let $\varepsilon > 0$, $\omega > 0$, $\sigma \in (0, 0.99)$, and $(x^1, z^1, y^1, s^1) \in \mathcal{N}$ be given.

for $k = 1, 2, 3, \ldots$ **do**

If $\mu_k \leq \varepsilon$ or $\|(x^k, s^k)\|_1 \geq \omega$, then stop.

Set $\nu_k := \sigma \mu_k$ and solve (4) for $(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$.

Set $(x^{k+1}, z^{k+1}, y^{k+1}, s^{k+1}) = (x_{\alpha_k}, z_{\alpha_k}, y_{\alpha_k}, s_{\alpha_k})$, where $\alpha_k \in (0, 1]$ is the largest step-size such that the relations

$$(x_\alpha, z_\alpha, y_\alpha, s_\alpha) \in \mathcal{N} \quad (9)$$

$$\mu_\alpha \leq (1 - 0.01\alpha)\mu_k \quad (10)$$

hold for every $\alpha \in [0, \alpha_k]$.

end for

The precise algorithm is stated as Algorithm 1. Note that the algorithm depends on user-defined tolerances $\varepsilon > 0$ and $\omega > 0$ as well as a duality gap "dampening" factor $\sigma \in (0, 0.99)$. In addition, the algorithm also utilizes the following definitions for $\alpha \in [0, 1]$:

$$(x_\alpha, z_\alpha, y_\alpha, s_\alpha) := (x^k, z^k, y^k, s^k) + \alpha(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$$
$$\mu_\alpha := \mu(x_\alpha, s_\alpha).$$

The convergence result is stated next.

Theorem 3.1 (Kojima et al. (1993)). *Let $\gamma \in (0,1)$, $\beta > 0$, and an initial point $(x^1, z^1, y^1, s^1) \in \mathcal{N}(\gamma, \beta)$ be given. Suppose, moreover, that positive tolerances ε and ω and a dampening factor $\sigma \in (0, 0.99)$ are specified. Then Algorithm 1 eventually generates an iterate $(x^k, z^k, y^k, s^k) \in \mathcal{N}(\gamma, \beta)$ such that $\mu_k \leq \varepsilon$ or $\|(x^k, s^k)\|_1 \geq \omega$. If the first case occurs, then $\|r^k\| \leq \beta\,\varepsilon$ as well.*

If the second case occurs, then Kojima et al. show that the infeasibility of (1–2) is implied over a wide region of the primal-dual ground space $K \times \Re^p \times \Re^m \times K$. Although this information is not a full infeasibility certificate, the intuition is that this is a strong indication of infeasibility, especially when ω is large.

The key to establishing Theorem 3.1 is to prove the existence of a positive constant α_* such that $\alpha_k \geq \alpha_*$ for all k generated by the algorithm. We state this lemma and prove the theorem; portions of the proof of the lemma, which are relevant to Section 5, are given below in Section 3.1.

Lemma 3.2 (Kojima et al. (1993)). *Suppose that there exists some constant $\eta > 0$ such that, for all k generated by the algorithm,*

$$|\,\Delta x_i^k \Delta s_i^k - \gamma\,(\Delta x^k)^T \Delta s^k / n\,| \;\leq\; \eta \quad \forall\; i = 1, \ldots, n, \tag{11a}$$

$$|\,(\Delta x^k)^T \Delta s^k / n\,| \;\leq\; \eta. \tag{11b}$$

Then $\alpha_k \geq \alpha_* > 0$, where

$$\alpha_* := \min\left\{1, \frac{(1-\gamma)\sigma\varepsilon}{\eta}, \frac{\sigma\varepsilon}{\eta}, \frac{(0.99-\sigma)\varepsilon}{\eta}\right\}. \tag{12}$$

Proof of Theorem 3.1. The proof is by contradiction. Assume that Algorithm 1 does not terminate. Then the entire infinite sequence $\{(x^k, z^k, y^k, s^k)\}$ lies in the compact set

$$\mathcal{N}^* := \{\; (x,z,y,s) \in \mathcal{N} \;:\; \mu(x,s) \geq \varepsilon,\; \|(x,s)\|_1 \leq \omega \;\},$$

Combining this with the fact that the Newton direction is a continuous function of the iterates (since (4) is nonsingular for each k), any continuous function of the direction is uniformly bounded over all k. So the hypothesis of Lemma 3.2 holds, implying that $\alpha_k \geq \alpha_*$ for all k. Thus, by Algorithm 1, the duality gap is decreased by at least a multiplicative factor of $1 - 0.01\alpha_* < 1$ in each iteration, and so $\mu_k \to 0$, which contradicts the assumption that Algorithm 1 does not terminate. □

3.1 Proof of Lemma 3.2

Assume that (11) holds for all k generated by the algorithm and note also that

$$\mu_k \geq \varepsilon \tag{13}$$

for the same k. We define $r_\alpha := g - E^T y_\alpha$ and recall that $r_\alpha = (1-\alpha)\,r$ by (6).

Using the definitions of $\mathcal{N}$ and Algorithm 1 together with the following three propositions, the proof of Lemma 3.2 is straightforward. We give only the proof of Proposition 3.4 because of its relevance for the results in Section 5. The super- and subscripts k are dropped since the arguments below are irrespective of k.

Proposition 3.3 (Kojima et al. (1993)). *$X_\alpha S_\alpha e \geq \gamma \mu_\alpha e$ for all $\alpha \leq (1-\gamma)\sigma\varepsilon/\eta$.*

Proposition 3.4 (Kojima et al. (1993)). *$\|r_\alpha\| \leq \beta\mu_\alpha$ for all $\alpha \leq \sigma\varepsilon/\eta$.*

Proof. Recall the standard relation

$$\mu_\alpha = (1-(1-\sigma)\alpha)\mu + \alpha^2 \Delta x^T \Delta s/n. \tag{14}$$

Thus, we have

$$\begin{aligned}
\beta\mu_\alpha - \|r_\alpha\| &= \beta\left[(1-(1-\sigma)\alpha)\mu + \alpha^2\Delta x^T\Delta s/n\right] - (1-\alpha)\|r\| \\
&= (1-\alpha)(\beta\mu - \|r\|) + \beta\alpha\sigma\mu + \beta\alpha^2\Delta x^T\Delta s/n \\
&\geq \beta\alpha\sigma\mu - \beta\alpha^2\eta \geq \alpha\beta\left[\sigma\varepsilon - \alpha\eta\right],
\end{aligned}$$

where the first equality follows from (14), the first inequality follows from $(x,z,y,s) \in \mathcal{N}$ and (11b), and the second inequality follows from (13). This proves the result. □

Proposition 3.5 (Kojima et al. (1993)). *$\mu_\alpha \leq (1-0.01\alpha)\mu$ for all $\alpha \leq (0.99-\sigma)\varepsilon/\eta$.*

3.2 Other Convergence Results

Before proceeding, we discuss other known convergence results for the infeasible framework of Section 2 and explain why we have chosen to analyze the result of Kojima et al. (1993).

Following the above global convergence result of Kojima et al., Zhang (1994) strengthened the result by proving that an ε-approximate optimal solution is delivered within $O(n^2)$ iterations (if an optimal solution exists). In particular, Zhang resolved the ambiguity surrounding Kojima et al.'s infeasibility "certificate" $\|(x,s)\|_1 > \omega$. For example, under the assumption of primal-dual interior feasibility, all iterates stay bounded. Later, Zhang (1998) extended these ideas to the case of SDP.

Other implementations of the framework have also been analyzed. In particular, a few authors have studied variations of the original predictor-corrector strategy of Mehrotra (1992). First, Mehrotra himself suggests a proof of global convergence for his method by appealing to certain "fall back" search directions, which are different from his own predictor-corrector direction. Then Zhang and Zhang (1995) prove polynomial convergence of a variation of Mehrotra's original method, which they suggest, but to our knowledge the Zhang-Zhang variant has not actually been implemented in practice. Finally, a third variant, which is used in most modern interior-point codes, is analyzed by Salahi et al. (2005). They make the simplifying assumption that all iterates are feasible and show by example that this variant can converge quite slowly on certain problems. By studying a suitable modification, they prove polynomial convergence. To our knowledge, no one has proved global or polynomial convergence of the infeasible version of this third variant (i.e., the one implemented in most codes).

A different line of research has analyzed the convergence of the framework when using inexact Newton directions, which are directions that satisfy (4) only approximately. Inexact directions arise, for example, when iterative methods are used to solve the system (7) to moderate accuracy. Depending on their precise form, inexact directions can lead to infeasible iterates, even if the algorithm is supplied with an initial feasible iterate. This is because the key relation (6) does not

hold from iteration to iteration. Nevertheless, global and polynomial convergence results can be proved under suitable conditions on the degree of the inexactness of the direction (see Mizuno and Jarre (1999); Korzak (2000); Freund et al. (1999) for LP, and more recently Zhou and Toh (2004) for SDP).

In Section 5, we propose Algorithm 2, a variant of Algorithm 1 that is based on an inexact Newton direction arising from regularization. We also extend the analysis in this section to prove a global convergence result for Algorithm 2. Although we were unable to prove a polynomial convergence result for Algorithm 2, and although the aforementioned work on inexact Newton directions can be applied to obtain a provably polynomially-convergent method with regularization, we deliberately choose to advocate Algorithm 2 for the following reasons:

- our extension of the original result of Kojima et al. has the advantages that: (i) it allows for infeasible iterates and a straightforward analysis of our particular inexact direction; and (ii) the resulting regularization strategy is quite simple to implement and has an intuitive appeal;
- our experiments in the direction of following the insights provided by the above research on inexact methods led us to the following conclusions: (i) the resulting regularization strategies are significantly more difficult to implement; and (ii) we implemented the approach of Freund et al. (1999), and found by experimentation that it did not work as well as our proposed approach.

It should also be noted that we chose *not* to analyze a predictor-corrector variant because the theoretical basis for implemented predictor-corrector strategies is less well understood, especially with regards to infeasible and inexact aspects. Nonetheless, we did test our method successfully within such a strategy (more details are given in Section 6).

Ultimately, we believe that our analysis allows us to identify the essence of a good regularization strategy. It should: (i) be convergent, (ii) be easy to implement, and (iii) work well in practice.

4 The Regularization Approach of Mészáros

Mészáros (1998) proposes to replace equation (4c) of the Newton system (4) with the following equation for a specified $\delta_k > 0$:

$$E^T \Delta y^k - \delta_k \, \Delta z^k = r_{d_2}^k. \tag{15}$$

Just as (4) can be reduced to (7), the regularized system of Mészáros can be reduced to

$$\begin{pmatrix} AXS^{-1}A^T & E \\ E^T & -\delta \, I \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} AXS^{-1}r_{d_1} - AS^{-1}r_c + r_p \\ r_{d_2} \end{pmatrix}, \tag{16}$$

where the k subscript is understood. In contrast to (7), however, (16) can further be reduced to the positive definite system

$$\left(AXS^{-1}A^T + \delta^{-1}EE^T\right) \Delta y = AXS^{-1}r_{d_1} - AS^{-1}r_c + r_p + \delta^{-1}Er_{d_2}. \tag{17}$$

Hence, the Cholesky factorization can be employed to calculate the direction. The obvious downside is that the resulting direction is not the true Newton direction. It is shown in Mészáros (1998) that the difference between these two directions is $O(\delta)$.

Yet, a more important question in practice is the choice of δ_k throughout the course of the algorithm, since poor choices can certainly have a negative impact on convergence. Under restrictive assumptions, an adaptive heuristic for updating δ_k is proposed in Mészáros (1998), and iterative refinement is also suggested for improving the quality of the search direction at each iteration. Furthermore, Maros and Mészáros (1998) investigate the choice of a constant δ_k throughout the algorithm. From our perspective, these suggestions are somewhat *ad hoc* and do not consider the effect of the regularization on the global convergence of the algorithm. We feel that the question of how to select a global strategy for updating δ_k was left open by Mészáros.

5 Global Convergence with Regularization

In contrast to Mészáros, we take the perspective that the regularization of the Newton system leads to an inexact interior-point method. In this section, we propose a specific methodology to update δ_k at each iteration, and show that the resulting interior-point method is globally convergent.

Recall that the direction determined by the regularization differs from the true Newton direction in that (15) replaces (4c). Expressed differently, we allow the Newton direction to satisfy

$$E^T \Delta y^k = r_{d_2}^k + \delta_k \Delta z^k,$$

even though we hope that $E^T \Delta y^k$ could equal $r_{d_2}^k$. So the direction given by the regularization is an inexact direction. But what effect does the inexact direction have on convergence? If one tries to extend the convergence proof of Kojima et al., all concepts and proofs extend easily except that we no longer have r_α equal to $(1-\alpha)r$, which in turn causes a direct extension of Proposition 3.4 to fail (see Section 3.1).

However, it is not so difficult to repair the proof of Proposition 3.4. The key insight is that the degree of inexactness of the direction needs to be controlled in a certain manner. Specifically, if we have

$$\delta_k \|\Delta z^k\| \leq \beta\sigma\mu_k/2 \tag{18}$$

for all k, then global convergence is established by Theorem 5.1 below.

Our method for enforcing (18) is essentially to decrease δ_k if (18) does not hold. The resulting algorithm is stated as Algorithm 2. Algorithm 2 incorporates all of the features of Algorithm 1, while adding steps to handle δ_k that are fairly straightforward. It is worth mentioning three items:

- In the `while` loop, each time δ_k is updated, it is decreased by a factor of at least 2. As a result, the `while` loop will terminate after a finite (usually quite small) number of loops.

- In each loop of the `while` loop, the direction must be recalculated. This necessitates re-forming and re-factoring the matrix $AXS^{-1}A^T + \delta_k^{-1}EE^T$ of (17) because of the dependence on δ_k. This is a potential downside of Algorithm 2. However, the computational results of Section 6 (particularly Table 5) show that, overall, this extra work does not constitute a disadvantage, most likely because the `while` loop is repeated only a small number of times.

- The purpose of the `while` loop is to drive δ_k lower and lower until (18) holds. Based on the (ultimately flawed) idea that δ_k should never increase during the course of the algorithm, our initial implementation of Algorithm 2 maintained a non-increasing sequence $\{\delta_k\}$. We found by experimentation, however, that sometimes δ_k would go to 0 too quickly causing numerical difficulties. By this we mean, for example, that one iteration would require $\delta_k \leq 10^{-4}$ to enforce (18), while a later iteration would require only $\delta_k \leq 10^{-2}$. For numerical stability, it makes sense to take δ_k as large as possible, which was not allowed by our initial implementation. Hence, our final implementation (i.e., Algorithm 2) allows δ_k to increase via the last line of the `for` loop, which sets δ_{k+1} to our best guess given current information.

Algorithm 2 Infeasible Path-Following Algorithm (with Regularization)

Let $\varepsilon > 0$, $\omega > 0$, $\sigma \in (0, 0.99)$, $\delta_1 > 0$, and $(x^1, z^1, y^1, s^1) \in \mathcal{N}$ be given.

for $k = 1, 2, 3, \ldots$ **do**
  If $\mu_k \leq \varepsilon$ or $\|(x^k, s^k)\|_1 \geq \omega$, then stop.
  Set ACCEPT $= 0$.
  while ACCEPT $= 0$ **do**
    Set $\nu_k = \sigma\mu_k$ and solve (4) with (4c) replaced by (15) for $(\Delta x^k, \Delta z^k, \Delta y^k, \Delta s^k)$.
    if $\delta_k \|\Delta z^k\| \leq \beta\sigma\mu_k/2$ **then**
      ACCEPT $= 1$
    else
      $\delta_k \leftarrow \frac{1}{2} \cdot \beta\sigma\mu_k/(2\|\Delta z^k\|)$
    end if
  end while
  Set $(x^{k+1}, z^{k+1}, y^{k+1}, s^{k+1}) = (x_{\alpha_k}, z_{\alpha_k}, y_{\alpha_k}, s_{\alpha_k})$, where $\alpha_k \in (0, 1]$ is the largest step-size such that the relations

$$(x_\alpha, z_\alpha, y_\alpha, s_\alpha) \in \mathcal{N} \tag{19}$$

$$\mu_\alpha \leq (1 - 0.01\alpha)\mu_k \tag{20}$$

  hold for every $\alpha \in [0, \alpha_k]$.
  Set $\delta_{k+1} \leftarrow \beta\sigma\mu_{k+1}/(2\|\Delta z^k\|)$.
end for

Regarding convergence, the following result holds for this algorithm.

Theorem 5.1. *Let $\gamma \in (0, 1)$, $\beta > 0$, and an initial point $(x^1, z^1, y^1, s^1) \in \mathcal{N}(\gamma, \beta)$ be given. Suppose, moreover, that positive tolerances ε and ω, a dampening factor $\sigma \in (0, 0.99)$, and an initial regularization parameter $\delta_1 > 0$ are specified. Then Algorithm 2 eventually generates an iterate $(x^k, z^k, y^k, s^k) \in \mathcal{N}(\gamma, \beta)$ such that $\mu_k \leq \varepsilon$ or $\|(x^k, s^k)\|_1 \geq \omega$. If the first case occurs, then $\|r^k\| \leq \beta\,\varepsilon$ as well.*

The proof of Theorem 5.1 follows the same steps as that of Theorem 3.1 with two changes: r_α

and α_* are now given by

$$r_\alpha = (1-\alpha)r - \alpha\delta\Delta z \tag{21}$$

and

$$\alpha_* := \min\left\{1, \frac{(1-\gamma)\sigma\varepsilon}{\eta}, \frac{\sigma\varepsilon}{2\eta}, \frac{(0.99-\sigma)\varepsilon}{\eta}\right\},$$

respectively, and Proposition 3.4 is replaced by the following proposition:

Proposition 5.2. *If $\delta_k\|\Delta z^k\| \le \beta\sigma\mu_k/2$, then $\|r_\alpha\| \le \beta\mu_\alpha$ for all $\alpha \le \sigma\varepsilon/2\eta$.*

Proof. We have

$$\begin{aligned}
&\beta\mu_\alpha - \|r_\alpha\| \\
&\quad= \beta\left[(1-(1-\sigma)\alpha)\mu + \alpha^2\Delta x^T\Delta s/n\right] - \|(1-\alpha)r - \alpha\delta\Delta z\| \\
&\quad\ge \beta\left[(1-(1-\sigma)\alpha)\mu + \alpha^2\Delta x^T\Delta s/n\right] - (1-\alpha)\|r\| - \alpha\delta\|\Delta z\| \\
&\quad\ge \beta\left[(1-(1-\sigma)\alpha)\mu + \alpha^2\Delta x^T\Delta s/n\right] - (1-\alpha)\|r\| - \beta\alpha\sigma\mu/2 \\
&\quad= (1-\alpha)(\beta\mu - \|r\|) + \beta\alpha\sigma\mu/2 + \beta\alpha^2\Delta x^T\Delta s/n \\
&\quad\ge \beta\alpha\sigma\mu/2 - \beta\alpha^2\eta \ge \alpha\beta\left[\sigma\varepsilon/2 - \alpha\eta\right],
\end{aligned}$$

where the first equality follows from (14) and (21), the first inequality follows from $(x,z,y,s) \in \mathcal{N}$ and (11b), and the second inequality follows from the assumption $(x,z,y,s) \in \mathcal{N}^*$. This proves the result. □

6 Implementation and Computational Results

We compare our proposed regularization (hereafter denoted REGULARIZE) with:

- explicitly solving the indefinite system (7) (EXPLICIT);
- splitting the free variables into the difference of two nonnegative variables (SPLIT); and
- putting the free variables into a second-order cone (QCONE).

The comparisons are carried out using SDPT3 (version 3.02), which supports EXPLICIT by default and supports SPLIT and QCONE via modified data input. Moreover, SDPT3 requires only simple code modifications to implement REGULARIZE. All tests are run on a dual Opteron 2.8GHz, using the HKM direction and the predictor-corrector option.

We point out that since our implementation is based on SDPT3, REGULARIZE differs from Algorithm 2 in the same ways that typical implementations of interior-point methods differ from theory. For example, the iterates are not explicitly forced to stay in a neighborhood, and Mehrotra's predictor-corrector method is used. The lack of explicit neighborhood does have an impact on how we enforce condition (18). In each iteration, (18) is enforced with the definition $\beta := \mu/\|r\|$. This can be interpreted as assigning to β the smallest value such that the current iterate is actually a member of $\mathcal{N}$ (if membership in $\mathcal{N}$ were maintained). In this sense, the choice of β is also

Class	# of Instances	K
DIMACS Challenge	9	LP+SOCP
Modified SDPLIB ($\text{rank}(E) = p$)	27	SDP
Modified SDPLIB ($\text{rank}(E) = p/2$)	27	SDP
Quantum chemistry	12	SDP
Moment relaxations	60	SDP

Table 1: Overview of the test problems sets

	p			n			m		
Class	min	med	max	min	med	max	min	med	max
DIMACS Challenge	1	2	7201	2379	4191	261364	123	3680	130141
Modified SDPLIB ($\text{rank}(E) = p$)	52	125	1514	351	7750	31375	104	250	3028
Modified SDPLIB ($\text{rank}(E) = p/2$)	52	125	1514	351	7750	31375	104	250	3028
Quantum chemistry	35	69	95	38865	480459	1356933	465	2354	4743
Moment relaxations	136	2040	2907	9316	13031	17613	3875	4930	5984

Table 2: Characteristics of the test problems sets

Class	Reg	Exp	Split	qCone
DIMACS Challenge	-10.3	-10.4	-5.6	-7.6
Modified SDPLIB ($\text{rank}(E) = p$)	-7.8	-7.9	-4.2	-4.5
Modified SDPLIB ($\text{rank}(E) = p/2$)	-7.9	1.3	-4.4	-4.7
Quantum chemistry	-5.1	-2.1	-1.8	-2.0
Moment relaxations	-6.0	-2.0	-2.5	-3.4

Table 3: Accuracy of each method, averaged over instances in each set

Class	Reg	Exp	Split	qCone
DIMACS Challenge	29.7	31.3	17.1	32.0
Modified SDPLIB ($\text{rank}(E) = p$)	15.6	15.4	13.0	13.3
Modified SDPLIB ($\text{rank}(E) = p/2$)	14.5	11.1	12.1	12.4
Quantum chemistry	22.7	10.0	11.8	14.0
Moment relaxations	21.9	10.8	13.2	16.0

Table 4: Number of iterations of each method, averaged over instances in each set

Class	Reg	Exp	Split	qCone
(all)	100%	138%	107%	114%

Table 5: Average CPU-time per iteration (as percentage of Regularize time)

conservative in that it results in the strictest realization of (18). One additional implementation detail: the parameter δ_k is initialized to $\delta_0 = \mu_0$.

As mentioned previously, SDPT3 has been chosen for several reasons, including the fact that its fundamental algorithm closely matches the algorithmic framework of our analysis. As a consequence, we caution that the conclusions supported by our computational results do not necessarily say anything about the effect of regularization within other codes (although we are optimistic that the regularization can have benefits elsewhere; see Section 7).

We also note that SDPT3 offers the option of handling dense columns of (A, E) in such a way that computational effort is minimized. We have tested our regularization with this option enabled (which is the default) or disabled, and the method works just as well in both cases.

There do not appear to be many existing test instances of linear conic optimization problems with free variables. For example, the commonly used DIMACS set of benchmark problems (Pataki and Schmieta (1999)) contains only 9 instances with free variables, while SDPLIB (Borchers (1999)) contains none. We include the 9 DIMACS test problems in our experiments. Kobayashi et al. (2005) generated modified problems having free variables from SDPLIB to test their approach; starting with the same SDPLIB problems, we generated our own sparse versions of these problems having free variables via a random matrix $E \in \Re^{m \times p}$, where $p := m/2$. In fact, we generated two sets of modified SDPLIB problems: one set with $\mathrm{rank}(E) = p$, and a second set with $\mathrm{rank}(E) = p/2$. In contrast to the theoretical assumption in Section 2 that E has full column rank, we test instances with E having small column rank because in practice E may have (nearly) dependent columns.

On the other hand, problems with free variables have become very relevant due to recent applications of SDP to certain classes of problems. In particular, we report test results on two additional sets of problems, one from quantum chemistry, and another obtained by generating moment relaxations of combinatorial optimization problems (Lasserre, 2002) using YALMIP (Löfberg, 2004). The set of moment relaxations consists of (small) randomly generated maximum-cut, quadratic-knapsack, and stable-set instances. Half of the underlying instances have 15 variables; the other half have 17. The moment relaxations are of order 2. For the maximum-cut and stable-set instances, we wrote our own simple random generation procedure, whereas the quadratic-knapsack problems were generated as in the study by Caprara et al. (1999). The stable-set formulation is due to Motzkin and Straus (1965). The large number of free variables in these instances arise from the number of equality constraints in the original problems. Characteristics of the test problems that we use are summarized in Tables 1 and 2.

The main criterion for comparing the various methods is the resulting accuracy, i.e., we wish to determine which approach yields the most accurate solutions. Specifically, we report accuracies as $\log_{10}$ of the maximum of the standard DIMACS accuracy errors (Mittelmann, 2003). Roughly speaking, an accuracy of $-k$ in our results corresponds to k digits of accuracy in the reported optimal value.

It is important to point out that SDPT3 tries to the improve the accuracy of the solution from iteration to iteration, until the accuracy deteriorates for a few iterations, at which point SDPT3 stops. For all the approaches, we let SDPT3 run until this happens and report the best accuracy obtained overall.

The results of our computational experiments are summarized in Tables 3, 4 and 5. Table 3 shows that for the test sets *DIMACS Challenge* and *Modified SDPLIB*, the proposed regularization

approach basically matches the best accuracy among the other three approaches. More interestingly, the accuracy obtained for the three other test sets is significantly higher. The results for the moment relaxations are particularly interesting because these instances contain the largest proportions of free variables.

Looking at the results for the set of problems with E having linearly dependent columns, we note that EXPLICIT achieves very poor accuracy for these problems due to numerical difficulties. The other three approaches, including REGULARIZE, seem unaffected by the dependencies in E.

Table 4 shows that the higher accuracy obtained by REGULARIZE typically requires a higher number of iterations (the other methods stop when accuracy deteriorates). Nonetheless, with respect to CPU time, Table 5 shows that REGULARIZE requires, on average, the same or less computational effort per iteration as that required by the other approaches. In summary, the proposed regularization seems to lead to an overall improvement in the performance of SDPT3 for instances with free variables.

7 Conclusion and Future Research

We have considered the regularization approach for handling free variables within interior-point methods for conic linear optimization. Using a global convergence analysis, we derive a simple computational strategy for handling and updating the regularization. Straightforward modifications to the modern code SDPT3 allow the regularization to be incorporated within an inexact infeasible primal-dual path-following interior-point framework. Computational results with SDPT3 on a variety of test problems suggest that the regularization is able to achieve the same or significantly better numerical accuracy than other strategies for handling free variables, while requiring less CPU-time per iteration on average.

It remains to be studied what impact regularization can have within other SDP codes, including the well-known homogeneous self-dual embedding algorithm implemented in SeDuMi, a package known to yield excellent accuracy in its computations. As indicated in the introduction, our experience with SeDuMi confirms SeDuMi's reputation; SeDuMi achieves 10 to 11 digits of accuracy on the test problems of this paper. It would be interesting to investigate how regularization can be applied to SeDuMi and if it can improve SeDuMi further. Another intriguing possibility is the study of an easily implementable update strategy for the regularization that would permit a proof of polynomial-time convergence.

Finally, we hope that this paper will stimulate further research on techniques to handle conic linear optimization problems with a significant proportion of free variables, as such problems have become relevant in the context of recent applications of SDP to several challenging problems.

Acknowledgments

The authors are in debt to two anonymous referees for numerous insightful comments and suggestions, which have greatly improved the paper.

References

E. Andersen. Handling free variables in primal-dual interior-point methods using a quadratic cone, 2002. Talk given at SIAM Conference on Optimization, Toronto.

M. Anjos. An improved semidefinite programming relaxation for the satisfiability problem. *Math. Program.*, 102(3):589–608, 2005.

B. Borchers. SDPLIB 1.2, library of semidefinite programming test problems. *Optim. Methods Softw.*, 11/12(1-4):683–690, 1999.

A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS J. Comput.*, 11(2):125–137, 1999. ISSN 1091-9856. Combinatorial optimization and network flows.

E. de Klerk. *Aspects of Semidefinite Programming*, volume 65 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, 2002. ISBN 1-4020-0547-4.

J. Faraut and A. Korányi. *Analysis on symmetric cones*. Oxford Mathematical Monographs. The Clarendon Press Oxford University Press, New York, 1994. ISBN 0-19-853477-9. Oxford Science Publications.

R. W. Freund, F. Jarre, and S. Mizuno. Convergence of a class of inexact interior-point algorithms for linear programs. *Math. Oper. Res.*, 24(1):50–71, 1999. ISSN 0364-765X.

C. Helmberg. `http://www-user.tu-chemnitz.de/~helmberg/semidef.html`.

K. Kobayashi, K. Nakata, and M. Kojima. A conversion of an SDP having free variables into the standard form SDP. Technical Report B-416, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, June 2005.

M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Math. Programming*, 61(3, Ser. A):263–280, 1993.

J. Korzak. Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems. *SIAM J. Optim.*, 11(1):133–148 (electronic), 2000. ISSN 1052-6234.

J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11(3):796–817 (electronic), 2000/01.

J. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM J. Optim.*, 12(3):756–769 (electronic), 2002.

J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. Available from `http://control.ee.ethz.ch/~joloef/yalmip.php`.

I. Maros and C. Mészáros. The role of the augmented system in interior point methods. *Eur. J. Oper. Res.*, 107:720–736, 1998.

S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2 (4):575–601, 1992. ISSN 1052-6234.

C. Mészáros. On free variables in interior point methods. *Optim. Methods Softw.*, 9(1-3):121–139, 1998.

H. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Math. Program.*, 95(2, Ser. B):407–430, 2003. ISSN 0025-5610.

S. Mizuno and F. Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Math. Program.*, 84(1, Ser. A):105–122, 1999. ISSN 0025-5610.

T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17(4):533–540, 1965.

P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96 (2, Ser. B):293–320, 2003.

G. Pataki and S. Schmieta. The DIMACS library of semidefinite-quadratic-linear programs. Technical report, 1999. See `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.

I. Pólik. Addendum to the SeDuMi user guide version 1.1. Technical report, Advanced Optimization Laboratory, McMaster University, 2005.

M. Salahi, J. Peng, and T. Terlaky. On Mehrotra-type predictor-corrector algorithms. AdvOl-Report No. 2005/4, Advanced Optimization Laboratory, March 2005.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11/12(1-4):625–653, 1999.

R. H. Tütüncü, K. C. Toh, and M. J. Todd. *SDPT3: A Matlab software package for semidefinite-quadratic-linear programming, version 3.0*, August 2001. Available from `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

R. Vanderbei. Symmetric quasidefinite matrices. *SIAM J. Optim.*, 5(1):100–113, 1995. ISSN 1052-6234.

R. Vanderbei. *Linear programming.* International Series in Operations Research & Management Science, 37. Kluwer Academic Publishers, Boston, MA, second edition, 2001. ISBN 0-7923-7342-1.

H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. Technical report, Tokyo Inst. Tech., 2004.

H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, Boston, MA, 2000. ISBN 0-7923-7771-0.

S. J. Wright. *Primal-dual interior-point methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

Y. Zhang. On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem. *SIAM J. Optim.*, 4(1):208–227, 1994. ISSN 1052-6234.

Y. Zhang. On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming. *SIAM J. Optim.*, 8(2):365–386 (electronic), 1998. ISSN 1052-6234.

Y. Zhang. User's guide to LIPSOL: linear-programming interior point solvers V0.4. *Optim. Methods Softw.*, 11/12(1-4):385–396, 1999. ISSN 1055-6788.

Y. Zhang and D. T. Zhang. On polynomiality of the Mehrotra-type predictor-corrector interior-point algorithms. *Math. Programming*, 68(3, Ser. A):303–318, 1995. ISSN 0025-5610.

Z. Zhao, B. Braams, M. Fukuda, M. Overton, and J. Percus. The reduced density matrix method for electronic structure calculations and the role of three-index representability. *J. Chem. Phys.*, 120:2095–2104, 2004.

G. Zhou and K.-C. Toh. Polynomiality of an inexact infeasible interior point algorithm for semidefinite programming. *Math. Program.*, 99(2, Ser. A):261–282, 2004. ISSN 0025-5610.