On Implementing a Homogeneous Interior-Point Algorithm for Nonsymmetric Conic Optimization

Skajaa, Anders; Jørgensen, John Bagterp; Hansen, Per Christian

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

IMM-Technical Report 2011-02

On Implementing a Homogeneous Interior-Point Algorithm for Nonsymmetric Conic Optimization

Anders Skajaa[*], John Bagterp Jørgensen[*] and Per Christian Hansen[*]

January, 2011

Abstract

Based on earlier work by Nesterov, an implementation of a homogeneous infeasible-start interior-point algorithm for solving nonsymmetric conic optimization problems is presented. Starting each iteration from (the vicinity of) the central path, the method computes (nearly) primal-dual symmetric approximate tangent directions followed by a purely primal centering procedure to locate the next central primal-dual point. Features of the algorithm include that it makes use only of the primal barrier function, that it is able to detect infeasibilities in the problem and that no phase-I method is needed. The method further employs quasi-Newton updating both to generate (pseudo) higher order directions and to reduce the number of factorizations needed in the centering process while still retaining the ability to exploit sparsity. Extensive and promising computational results are presented for the p-cone problem, the facility location problem, entropy problems and geometric programs; all formulated as nonsymmetric conic optimization problems.

Keywords: convex optimization, nonsymmetric, conic optimization, homogeneous model, infeasible-start, interior-point algorithm.

1 Introduction

In this paper we are concerned with conic optimization problem pairs of the form

$$\textsc{Primal}\ \begin{cases} \min_x & c^T x \\ \text{s.t.} & Ax = b \\ & x \in \mathcal{K} \end{cases} \qquad \textsc{Dual}\ \begin{cases} \max_{y,s} & b^T y \\ \text{s.t.} & A^T y + s = c \\ & s \in \mathcal{K}^*,\ y \in \mathbb{R}^m \end{cases} \tag{1}$$

where $x, c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, $\mathcal{K} \subseteq \mathbb{R}^n$ is a proper cone (i.e. it is convex, pointed, closed and has nonempty interior) and $\mathcal{K}^* = \{s \in \mathbb{R}^n : s^T x \geq 0,\ \forall x \in \mathcal{K}\}$, its dual cone, which is also proper. We are further assuming that $m \leq n$ and that A has full row-rank.

These problems are well described and many interior point methods for different variants of these problems exist, see for example [4, 5, 25, 21, 14].

When replacing $\mathcal{K}$ in (1) with $\mathbb{R}^n_+$, we obtain the simplest case: Linear Programming (LP). Solution methods for LP have been studied for long in different settings and until the emergence of interior point methods (IPMs), the most prominent method was the simplex method, developed by Dantzig in the 1940s. The introduction of IPMs is usually ascribed to Karmarkar [12] in 1984 and since then, research in the area has been extensive.

In [17] it was studied how to extend the ideas of IPMs to the nonlinear case. The papers [18, 19] showed that problems of the type (1) are efficiently solvable using symmetric primal-dual IPMs when $\mathcal{K}$ admits a *self-scaled* barrier function F: $\mathcal{K}^\circ \mapsto \mathbb{R}$. In [11], Güler demonstrated that such cones are limited to those that are homogeneous and self-dual; a class that encompasses just five cones of which only

[*] *Department of Informatics and Mathematical Modelling, Technical University of Denmark*

three are interesting for optimization. Specifically: the positive orthant (leading to LP), the Lorentz cone (leading to second order cone programming) and the positive semidefinite cone (leading to semidefinite programming).

These three self-scaled cones allow for modelling of a great variety of constraints and with the existence of a unique scaling point [18, 19] for each primal-dual pair, symmetric primal-dual interior-point methods (PDIPM) can be constructed and very efficiently implemented [1, 2, 23].

However many constraints do not fall in this class. Examples include entropy type constraints: $x \log x \leq t$, p-cone constraints: $\|x\|_p \leq t$, and constraints arising in geometric programming [5]. Some of these constraints can be modelled using self-scaled cones, but this usually requires the introduction of many extra variables and constraints [4].

Theoretically, one can solve problems involving *any* convex constraint using a purely primal barrier method and still obtain an algorithm with the best known complexity. However, such an algorithm is known to be practically less efficient than a PDIPM. Other approaches are also possible and special algorithms for certain problems exist [25, 27]. An approach known to be effective for general convex problems is to solve the monotone complementariy problem, see for example [3].

However, it may be beneficial to model non-symmetric constraints more directly using *non*-self-scaled cones, such as the power cone or the exponential cone. This approach was employed by Nesterov in [16]. He proposed a method that mimics the ideas of a PDIPM for symmetric cones by splitting each iteration into two phases. First, a pure primal centering (or correction) phase is used to find a primal central point x and a scaling point w. These points are used to compute a feasible dual point s such that an *exact scaling relation* is satisfied: $s = \nabla^2 F(w)x$. Second, a true symmetric primal-dual affine-scaling step (or prediction step) is taken.

Extending the algorithm of [16], we propose in this paper a primal-dual interior point algorithm for a *homogeneous* model of (1). This approach has been succesful for self-scaled cones [26, 2, 22] because it implies the following desirable properties: Ability to detect infeasibility and ease of finding a suitable starting point, removing the need for a phase I method. In addition to this, we suggest using BFGS-updating of the Hessian of the barrier function to obtain a pseudo-second order approximate tangent direction (ATD) and to speed up the centering process.

For all problems that we consider, $\mathcal{K}$ will have the form $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$ where each $\mathcal{K}_j$ is either a *three*-dimensional proper cone or $\mathbb{R}_+$. We assume that a logarithmically homogeneous self-concordant barrier function (LHSCB) F for $\mathcal{K}$, its gradient ∇F and its Hessian $\nabla^2 F$ are available and computable in polynomial time for all $x \in \mathcal{K}$.

The paper is organized as follows: We first list a few well known properties about LHSCBs and then we introduce notation and present our algorithm. We give a proof of convergence and then suggest some heuristic improvements to speed up convergence. Finally we present computational results for several problems.

2 Properties of LHSCB

We assume that $F : \mathcal{K}^\circ \mapsto \mathbb{R}$ is a LHSCB for $\mathcal{K}$ with parameter ν. This means that for all $x \in \mathcal{K}^\circ$ and $t > 0$:

$$F(tx) = F(x) - \nu \log t.$$

It follows that the dual barrier of F, denoted F^* and defined for $s \in \mathcal{K}^*$ by $F^*(s) = \sup_{x \in \mathcal{K}}\{-s^T x - F(x)\}$, is a LHSCB for the dual cone $\mathcal{K}^*$. Using the Hessians of F and F^*, local norms on $\mathcal{K}$ and $\mathcal{K}^*$ can be defined:

$$\|g\|_x = \sqrt{g^T(\nabla^2 F(x))g}, \quad \text{for } g, x \in \mathcal{K} \tag{2}$$

$$\|h\|_s^* = \sqrt{h^T(\nabla^2 F^*(s))h}, \quad \text{for } h, s \in \mathcal{K}^*. \tag{3}$$

Properties of F and F^* include (see e.g. [18, 19]):

$$\nabla^2 F(x)x = -\nabla F(x) \tag{4}$$

$$x \in \mathcal{K} \Rightarrow -\nabla F(x) \in \mathcal{K}^* \tag{5}$$

$$x^T \nabla F(x) = -\nu \tag{6}$$

$$\nabla^2 F^*(-\nabla F(x)) = \nabla^2 F(x)^{-1} \tag{7}$$

$$\|x\|_x = \nu. \tag{8}$$

It is well known that the *Dikin ellipsoid* [4] is feasible. That is:

$$x \in \mathcal{K} \quad \Rightarrow \quad W(x) = \{u, \|u - x\|_x \leq 1\} \subseteq \mathcal{K} \tag{9}$$

and we have similarly for the dual cone that

$$s \in \mathcal{K}^* \quad \Rightarrow \quad W^*(s) = \{h, \|h - s\|_s^* \leq 1\} \subseteq K^*. \tag{10}$$

If $x \in \mathcal{K}$, we have by (5) and (10) that

$$W^*(-\nabla F(x)) = \{h, \|h + \nabla F(x)\|_{-\nabla F(x)}^* \leq 1\} \subseteq K^*. \tag{11}$$

We are going to need the following identity:

$$\begin{aligned} \|\nabla^2 F(x) g\|_{-\nabla F(x)}^* &\overset{(3)}{=} \sqrt{g^T \nabla^2 F(x) \nabla^2 F^*(-\nabla F(x)) \nabla^2 F(x) g} \\ &\overset{(7)}{=} \sqrt{g^T \nabla^2 F(x) \nabla^2 F(x)^{-1} \nabla^2 F(x) g} \\ &\overset{(2)}{=} \|g\|_x. \end{aligned} \tag{12}$$

3 Notation

Generally z denotes an aggregated variable $(x; \tau; y; s; \kappa) \in \mathbb{R}^{n+1+m+n+1}$ and w denotes an aggregated variable of the type $(x; \tau; y; \kappa)$. Semicolons indicate vertical stacking of column vectors.

Let

$$G = \begin{pmatrix} 0 & A & -b \\ -A^T & 0 & c \\ b^T & -c^T & 0 \end{pmatrix}$$

and notice that G is skew-symmetric: $G = -G^T$. We will use the following abbreviations:

$$\begin{aligned} r(z) = (r_P(z); r_D(z); r_G(z)) &= G(y; x; \tau) - (0; s; \kappa) \\ &= \begin{pmatrix} Ax - b\tau \\ -A^T y - s + c\tau \\ -c^T x + b^T y - \kappa \end{pmatrix} \\ r_A(z) &= \left| c^T x - b^T y \right| / \left(\tau + \left| b^T y \right| \right) \\ r_C(w, \mu) &= -A^T y + \mu \nabla F(x) + c\tau \\ \psi(z, \mu) &= s - \mu \nabla^2 F(x) x \\ \mu(z) &= \left(x^T s + \tau\kappa \right) / \left(\nu + 1 \right). \end{aligned}$$

When needed, we will use a superscript, e.g. $z^{(k)}$, to denote iteration counter and a subscript, e.g. z_j, to denote the jth element of the vector z.

4 Homogeneous algorithm

For a parameter $\mu > 0$, the primal barrier problem corresponding to (1) is

$$\min_x \; c^T x + \mu F(x), \quad \text{s.t. } Ax = b \tag{13}$$

and the KKT conditions of this problem can be stated as follows: If $x \in \mathcal{K}$ is optimal for (13), then $\exists s \in \mathcal{K}^*$, $y \in \mathbb{R}^m$ s.t. that

$$Ax - b = 0 \tag{14}$$

$$-A^T y - s + c = 0 \tag{15}$$

$$s + \mu \nabla F(x) = 0. \tag{16}$$

The points that satisfy (14)-(16) are known as the primal-dual *central path* and it is easily seen that they satisfy $b^T y - c^T x = x^T s = \mu\nu$. The idea of the barrier method is to solve (13) for a decreasing sequence of μ's, thus obtaining a sequence of points with duality gap decreasing to zero and hence eventually being approximately optimal for (1).

In practice it is more efficient to take steps that are combinations of the direction approximately tangent to the central path (ATD) and the direction pointing towards the central path (the centering direction). After each step, μ is updated and the procedure repeated. Thus the iterates are *guided* by the central path, but not necessarily *on* it and such a method falls in the category of *path-following* methods or PDIPMs. We obtain different methods depending on the strategy for updating μ and how close the iterates are kept to the central path.

When $\mathcal{K}$ is self-scaled, PDIPMs can compute symmetric search directions. Here *symmetric* refers to the search directions (and thus the iterates) being *the same* regardless of whether the roles of the primal and dual problems in (1) are interchanged [24]. Thus no particular emphasis is put on either the primal or the dual problem, which is a desireable feature of an algorithm.

In the case when $\mathcal{K}$ is not self-scaled, this symmetry is lost. Instead, the idea of [16] is to split each iteration into two phases: 1. Prediction phase: Starting from a central point, compute a *symmetric* ATD and step in this direction and 2. Correction phase: compute a new central point using a purely primal iterative procedure.

In this paper, we propose solving the homogeneous model of problems (1) using the same two phases. We thus introduce two extra non-negative scalar variables τ and κ and seek to find $z = (x; \tau; y; s; \kappa)$ such that

$$Ax - b\tau = 0 \tag{17}$$

$$-A^T y - s + c\tau = 0 \tag{18}$$

$$-c^T x + b^T y - \kappa = 0 \tag{19}$$

$$x \in \mathcal{K},\ s \in \mathcal{K}^*,\ y \in \mathbb{R}^m,\ \tau \geq 0,\ \kappa \geq 0. \tag{20}$$

Lemma 1. *Assume $z = (x; \tau; y; s; \kappa)$ satisfies (17)-(20). Then*

(i) z is complementary. *That is: $(\nu + 1)\mu(z) = x^T s + \tau\kappa = 0$.*

(ii) If $\tau > 0$ then $(x, y, s)/\tau$ is optimal for (1).

(iii) If $\kappa > 0$ then, one or both of $b^T y > 0$ and $c^T x < 0$ hold. If the first holds, then (1) is primal-infeasible. If the second holds, then (1) is dual-infeasible.

Proof. (i) Observe that we can write (17)-(19) as $G(y; x; \tau) - (0; s; \kappa) = 0$. Pre-multiplying this equation by $(y; x; \tau)^T$ gives $x^T s + \tau\kappa = 0$. *(ii)* $\tau > 0$ implies $\kappa = 0$ and hence by (19): $b^T(y/\tau) - c^T(x/\tau) = 0$. Further dividing (17) and (18) by τ we obtain (14) and (15) respectively. Thus $(x, y, s)/\tau$ is optimal for (1). *(iii)* If $\kappa > 0$ then $\tau = 0$ so $Ax = 0$ and $A^T y + s = 0$. Further $c^T x - b^T y = -\kappa < 0$ so not both $c^T x$ and $-b^T y$ can be non-negative. Assume $-b^T x < 0$. If (1) is primal-feasible then there exists $\bar{x} \in \mathcal{K}$ such that $A\bar{x} = b$. But then $0 > -b^T y = -\bar{x}^T A^T y = \bar{x}^T s \geq 0$, a contradiction. We can argue similarly if $c^T x < 0$. □

Lemma 1 shows that any solution to (17)-(20) with $\tau + \kappa > 0$ provides either an optimal solution to our original problems (1) or a certificate of infeasibility of (one of) the original problems. See [13] for further details.

Let $z^{(0)}$ satisfy (20) and let $\mu^{(0)} = \mu(z^{(0)})$. Parametrized by $\gamma \in [0, 1]$, we will define the central path of problem (17)-(20) by

$$r(z) = \quad \gamma r(z^{(0)}) \tag{21}$$

$$s = -\gamma\mu^{(0)}\nabla F(x) \tag{22}$$

$$\tau\kappa = \quad \gamma\mu^{(0)}. \tag{23}$$

The central path connects the initial point ($\gamma = 1$) with a solution of (17)-(20) ($\gamma = 0$). The ATD z' is then determined by differentiating (21)-(23) w.r.t. γ and reusing (21)-(23):

$$r(z') = -\gamma^{-1} r(z) \tag{24}$$

$$s' + \gamma\mu^{(0)}\nabla^2 F(x)x' = -\gamma^{-1} s \tag{25}$$

$$\tau'\kappa + \kappa'\tau = -\gamma^{-1}\tau\kappa. \tag{26}$$

Our algorithm computes z' by solving the linear system of equations (24)-(26). We then compute $\hat{z} = z + \alpha z'$ where α is determined via a line search method such that $\hat{x} \in \mathcal{K}$, $\hat{s} \in \mathcal{K}^*$ and $\hat{\tau}, \hat{\kappa} \geq 0$ and such that $\hat{z}$ is not too far away from the central path (see Section 5.1.1). We then let $\gamma = \mu(\hat{z})/\mu(z^{(0)})$ and compute a new primal central point x^+ by solving (21)-(23) approximately using Newton's method, with $\hat{x}$ as the starting point. After that, we use x^+ to determine a new primal-dual feasible point z^+ and then repeat. Algorithm 1 shows this entire procedure in detail.

Algorithm 1 Basic homogeneous algorithm

Input: Initial point $z^{(0)} = (x^{(0)}; \tau^{(0)}; y^{(0)}; s^{(0)}; \kappa^{(0)})$ and barrier function F.
for $k = 0, 1, 2, \ldots$ **do**
Residuals: Compute $r(z^{(k)})$ and $r_A^{(k)}$
Stopping: If stopping criteria satisfied: **break**
ATD: Solve the following system for $d^{(k)} = (d_x^{(k)}, d_\tau^{(k)}, d_y^{(k)}, d_s^{(k)}, d_\kappa^{(k)})$:

$$r(d^{(k)}) = -r(z^{(k)}) \tag{27}$$
$$d_\tau^{(k)}\kappa^{(k)} + d_\kappa^{(k)}\tau^{(k)} = -\tau^{(k)}\kappa^{(k)} \tag{28}$$
$$\mu\nabla^2 F(x^{(k)})d_x^{(k)} + d_s^{(k)} = -s^{(k)} \tag{29}$$

Line search: Compute a step size $\alpha^{(k)}$
Update: $\hat{z}^{(k)} := z^{(k)} + \alpha^{(k)} d^{(k)}$ and $\mu := \mu(\hat{z}^{(k)})$ and $\hat{r}_D^{(k)} = r_D(\hat{z}^{(k)})$
Centering phase:
Set $w_c^{(0)} = (x_c^{(0)}; \tau_c^{(0)}; y_c^{(0)}; \kappa_c^{(0)}) := (\hat{x}^{(k)}; \hat{\tau}^{(k)}; \hat{y}^{(k)}; \hat{\kappa}^{(k)})$, $\beta^{(0)} = \infty$ and $\lambda^{(0)} = \infty$
for $j = 0, 1, 2, \ldots$ **do**
Residuals: Compute $r_1^{(j)} = \tau_c^{(j)}\kappa_c^{(j)} - \mu$ and $r_2^{(j)} = r_C(w_c^{(j)}) - \hat{r}_D^{(k)}$
Compute $B = (|r_1^{(j)}| \leq \rho_1) \wedge (\|r_2^{(j)}\| \leq \rho_2) \wedge (\lambda^{(j)} \leq \rho_3) \wedge (\beta^{(j)} = 1)$
Stopping: if $B =$ **true**, set $N := j$ and **break**.
CDIR: Solve the following system for $\delta^{(j)} = (\delta_x^{(j)}; \delta_\tau^{(j)}; \delta_y^{(j)}; \delta_\kappa^{(j)})$:

$$(r_P(\delta^{(j)}); r_D(\delta^{(j)})) = (0; 0) \tag{30}$$
$$\delta_\tau^{(j)}\kappa_c^{(j)} + \delta_\kappa^{(j)}\tau_c^{(j)} = -r_1^{(j)} \tag{31}$$
$$\mu\nabla^2 F(x_c^{(j)})\delta_x^{(j)} - A^T\delta_y^{(j)} + c\delta_\tau^{(j)} = -r_2^{(j)} \tag{32}$$

Line search: Compute a step size $\beta^{(j+1)}$
Update: $w_c^{(j+1)} := w_c^{(j)} + \beta^{(j+1)}\delta^{(j)}$ and $\lambda^{(j+1)} = \|\delta_x^{(j)}\|_{x_c^{(j)}}$
end for
Primal-dual lifting: Set $s^{(k+1)} = -A^T y_c^{(N)} + c\tau_c^{(N)} - \hat{r}_D^{(k)}$
Update: $z^{(k+1)} := (x_c^{(N)}; \tau_c^{(N)}; y_c^{(N)}; s^{(k+1)}; \kappa_c^{(N)})$
end for

5 Analysis of Algorithm 1

5.1 Approximate Tangent Direction

Let us first list some properties of the approximate tangent direction defined in (27)-(29). For ease of notation in this section, we are going write ψ for $\psi(z^{(k)}, \mu) = s^{(k)} - \mu\nabla^2 F(x^{(k)})x^{(k)}$ and drop the superscript iteration counter.

Lemma 2. *Assume z and d satisfy (27)-(29). Then*

$$\textit{(i)} \quad d_x^T s + x^T d_s + x^T s = d_x^T \psi \tag{33}$$
$$\textit{(ii)} \quad (x + d_x)^T(s + d_s) + (\tau + d_\tau)(\kappa + d_\kappa) = 0 \tag{34}$$
$$\textit{(iii)} \quad d_x^T d_s + d_\tau d_\kappa = -d_x^T \psi. \tag{35}$$

Proof. (i): We get $d_x^T s + x^T d_s + x^T s \overset{(29)}{=} d_x^T s + x^T(-s - \mu\nabla^2 F(x)d_x) + x^T s$, which, after reduction, gives $d_x^T(s - \mu\nabla^2 F(x)x) = d_x^T\psi$. *(ii)*: Equation (27) is equivalent to $r(z+d) = 0$ or $G(y+d_y; x+d_x; \tau+d_\tau) - (0; s+d_s; \kappa+d_\kappa) = 0$. Pre-multiplying with $(y+d_y; x+d_x; \tau+d_\tau)^T$ gives (34). *(iii)*: Follows from expanding (34) and using (33) and (28). □

Lemma 3. *Denote $\hat{z} = z + \alpha d$. Then*

$$r(\hat{z}) = (1-\alpha)r(z) \tag{36}$$

$$\mu(\hat{z}) = (1-\alpha)\mu(z) + \alpha(1-\alpha)\frac{d_x^T\psi}{\nu+1}. \tag{37}$$

Proof. Equation (36) follows directly from elementary linear algebra. To show (37):

$$\begin{aligned}
(\nu+1)\mu(\hat{z}) &= (x+\alpha d_x)^T(s+\alpha d_s) + (\tau+\alpha d_\tau)(\kappa+\alpha d_\kappa) \\
&= x^T s + \tau\kappa + \alpha(d_x^T s + x^T d_s) \\
&\quad + \alpha(\tau d_\kappa + \kappa d_\tau) + \alpha^2(d_x^T d_s + d_\tau d_\kappa) \\
&\overset{\text{Lemma 2}}{=} x^T s + \tau\kappa + \alpha(-x^T s + d_x^T\psi) \\
&\quad + \alpha(-\tau\kappa) + \alpha^2(-d_x^T\psi) \\
&= (1-\alpha)(x^T s + \tau\kappa) + \alpha(1-\alpha)d_x^T\psi
\end{aligned}$$

which after dividing by $(\nu+1)$ gives (37). □

5.1.1 Line Search

After computing the search direction, we determine a step size α using backtracking line search [6] to ensure that $\hat{x} \in \mathcal{K}$, $\hat{s} \in \mathcal{K}^*$ and $\hat{\tau}, \hat{\kappa} > 0$. To stay near the central path, we also require

$$\left\| \begin{pmatrix} \|\nabla F(\hat{x})\|_\infty^{-1}(\hat{s} + \hat{\mu}\nabla F(\hat{x})) \\ \hat{\tau}\hat{\kappa} - \hat{\mu} \end{pmatrix} \right\|_\infty \leq \sigma\hat{\mu}, \quad \text{with } \sigma \in [0,1]. \tag{38}$$

This neighborhood is related to the proximity measure $\|\mu^{-1}s + \nabla F(x)\|^*_{-\nabla F(x)}$. See e.g. [21], Thm. 3.7.1 for a discussion.

5.2 Centering Process

Let $\hat{\mu} = \mu(\hat{z}^{(k)})$. After the ATD-step, the goal of the centering process is to find a primal-dual feasible point z that (approximately) satisfies the equations

$$r(z) = r(\hat{z}^{(k)}) \tag{39}$$

$$s = -\hat{\mu}\nabla F(x) \tag{40}$$

$$\tau\kappa = \hat{\mu} \tag{41}$$

where $\hat{\mu}$ is fixed throughout the centering process.

We solve problem (39)-(41) by using Newton's method. Because of the nonsymmetric nature of this problem, we are not going to require that all iterates stay primal-dual feasible. Instead, we require that all primal iterates be in $\mathcal{K}$, but not that the dual iterates be in $\mathcal{K}^*$. For this reason, we eliminate s from the equations (39)-(41) and get the step equations

$$A\delta_x^{(j)} - b\delta_\tau^{(j)} = 0 \tag{42}$$

$$-c^T\delta_x^{(j)} + b^T\delta_y^{(j)} - \delta_\kappa^{(j)} = 0 \tag{43}$$

$$\tau_c^{(j)}\delta_\kappa^{(j)} + \kappa_c^{(j)}\delta_\tau^{(j)} = \hat{\mu} - \tau_c^{(j)}\kappa_c^{(j)} \tag{44}$$

$$\hat{\mu}\nabla^2 F(x_c^{(j)})\delta_x^{(j)} - A^T\delta_y^{(j)} + c\delta_\tau^{(j)} = r_D(\hat{z}^{(k)}) - r_C(w_c^{(j)}). \tag{45}$$

We then let $w_c^{(j+1)} = w_c^{(j)} + \beta\delta^{(j)}$, where β is a step length determined by a line search procedure using the merit function

$$\phi(z) = \|r_D(\hat{z}^{(k)}) - r_C(w_c)\|_\infty + |\tau\kappa - \mu|.$$

Let ρ_1, ρ_2 and ρ_3 be tolerances. We terminate the process when

$$|r_1| = |\tau_c \kappa_c - \hat{\mu}| \leq \rho_1 \tag{46}$$

$$\|r_D(\hat{z}^{(k)}) - r_C(w_c)\|_\infty \leq \rho_2 \tag{47}$$

$$\|\delta_x\|_{x_c} \leq \rho_3 \tag{48}$$

$$\beta = 1. \tag{49}$$

Lemma 4. *The centering problem (39)-(41) can be solved to precision (46)-(49) in polynomial time using Newton's method with the worst case complexity* $\mathcal{O}(\phi(\hat{z}))$.

Proof. See [9] and [17], Sec. 2.2, Thm 2.2.3 and comments on page 27. □

Notice that $\phi(\hat{z})$ is bounded by the neighborhood requirement (38), thus putting an upper bound on the computational effort needed to solve the centering problem.

5.3 Primal-dual lifting

After having solved the centering problem to the desired accuracy, we compute a dual point $s^{(k+1)}$ and we thus have a new primal-dual point $z^{(k+1)}$:

$$\begin{aligned} s^{(k+1)} &= -A^T y_c^{(N)} + c\tau_c^{(N)} - \hat{r}_D^{(k)} \\ z^{(k+1)} &= (x_c^{(N)}; \tau_c^{(N)}; y_c^{(N)}; s^{(k+1)}; \kappa_c^{(N)}). \end{aligned} \tag{50}$$

Lemma 5. *The new primal-dual point satisfies:*

$$r(z^{(k+1)}) = r(\hat{z}^{(k)}).$$

Proof. Follows from using elementary linear algebra with (42), (43) and (50). □

Lemma 6. *If* $\rho_3 \leq \nu + 1$*, the new primal-dual point satisfies*

$$\left|\mu(z^{(k+1)}) - \mu(\hat{z}^{(k)})\right| \leq \frac{\rho_1}{\nu+1}.$$

Proof. Because of (48) and (49), we have

$$\begin{aligned} w_c^{(N)} &= w_c^{(N-1)} + \delta^{(N-1)} \\ \|\delta_x^{(N-1)}\|_{x_c^{(N-1)}} &\leq \rho_3 \leq \nu + 1. \end{aligned}$$

From (45), we also have:

$$\begin{aligned} \hat{\mu}\nabla^2 F(x_c^{(N-1)})\delta_x^{(N-1)} - A^T\delta_y^{(N-1)} + c\delta_\tau^{(N-1)} = -r_2^{(N-1)} \quad &\Rightarrow \\ \hat{\mu}\nabla^2 F(x_c^{(N-1)})(x_c^{(N-1)} - \delta_x^{(N-1)}) = s^{k+1}. \end{aligned} \tag{51}$$

We then see that

$$\begin{aligned} (\nu+1)\mu(z^{(k+1)}) &= \hat{\mu}\left(\nabla^2 F(x_c^{(N-1)})(x_c^{(N-1)} - \delta_x^{(N-1)})\right)^T x_c^{(N)} + \tau_c^{(N)}\kappa_c^{(N)} \\ &= \hat{\mu}\left(\nabla^2 F(x_c^{(N-1)})(x_c^{(N-1)} - \delta_x^{(N-1)})\right)^T (x_c^{(N-1)} + \delta_x^{(N-1)}) + \tau_c^{(N)}\kappa_c^{(N)} \\ &\overset{(8)}{=} \hat{\mu}\left(\nu - \|\delta_x^{(N-1)}\|_{x_c^{(N-1)}}\right) + \tau_c^{(N)}\kappa_c^{(N)} \\ &= (\nu+1)\hat{\mu} - \hat{\mu}\|\delta_x^{(N-1)}\|_{x_c^{(N-1)}} + \tau_c^{(N)}\kappa_c^{(N)} - \hat{\mu} \end{aligned}$$

and hence

$$\begin{aligned} \mu(z^{(k+1)}) &= \mu(\hat{z}^{(k)})\left(1 - \frac{\|\delta_x^{(N-1)}\|_{x_c^{(N-1)}}}{\nu+1}\right) + \frac{r_1^{(N)}}{\nu+1} \\ &\leq \mu(\hat{z}^{(k)}) + \frac{r_1^{(N)}}{\nu+1} \quad \Rightarrow \\ \left|\mu(z^{(k+1)}) - \mu(\hat{z}^{(k)})\right| &\leq \frac{\rho_1}{\nu+1}. \end{aligned}$$

□

Lemma 7. *The next ψ satisfies*

$$\|\psi^{(k+1)}\|_\infty \leq \rho_2.$$

Proof.

$$\begin{aligned}
\psi^{(k+1)} &= s^{(k+1)} - \hat{\mu}\nabla^2 F(x^{(k+1)})x^{(k+1)} \\
&\overset{(4)}{=} s^{(k+1)} + \hat{\mu}\nabla F(x^{(k+1)}) \\
&= -A^T y_c^{(N)} + c\tau_c^{(N)} - \hat{r}_D^{(k-1)} + \hat{\mu}\nabla F(x_c^{(N)}) \\
&= r_C(w_c^{(N)}) - \hat{r}_D^{(k)} \quad \Rightarrow \\
\|\psi^{(k)}\|_\infty &= \|r_C(w_c^{(N)}) - \hat{r}_D^{(k)}\|_\infty \leq \rho_2.
\end{aligned}$$

□

Lemma 8. *If $\rho_3 \leq 1$ then the new dual point $s^{(k+1)}$ is dual feasible.*

Proof. By (32) we have

$$\mu^{-1}s^{(k+1)} + \nabla F(x^{(N-1)}) = -\nabla^2 F(x^{(N-1)})\delta_x^{(N-1)}$$

and so

$$\begin{aligned}
\|\mu^{-1}s^{(k+1)} + \nabla F(x^{(N-1)})\|^*_{-\nabla F(x_c^{(N-1)})} &= \|\nabla^2 F(x^{(N-1)})\delta_x^{(N-1)}\|^*_{-\nabla F(x_c^{(N-1)})} \\
&\overset{(12)}{=} \|\delta_x^{(N-1)}\|_{x_c^{(N-1)}} \\
&= \lambda^{(N)} \overset{(48)}{\leq} \rho_3 \leq 1.
\end{aligned}$$

We know by (5) that $-\nabla F(x_c^{(N-1)})$ is a dual feasible point. So by (11), $\mu^{-1}s^{(k+1)}$ is dual feasible, which implies that $s^{(k+1)}$ is dual feasible. □

5.4 Convergence

The lemmas above imply the convergence of Algorithm 1. Assuming that F is sufficiently regular, there exists $\xi > 0$ such that ξ is a lower bound on all step lengths [6], i.e.

$$\forall k : \alpha^{(k)} \geq \xi. \tag{52}$$

Using lemma 3 and (52), we see that, in a worst case sense, getting

$$\|r(z^{(k+1)})\| \leq \epsilon\|r(z^{(0)})\|$$

would take on the order of $\mathcal{O}\left(\xi^{-1}\log(1/\epsilon)\right)$ iterations for $\epsilon \ll 1$.

Lemma 9. *Let $\eta \in (0,1)$ and assume that we in all iterations can obtain*

$$(d_x^{(k)})^T\psi^{(k)} \leq (\nu+1)\mu(z^{(k)}) \tag{53}$$

then

$$\mu(z^{(k+1)}) \leq \left(1-\eta\xi^2\right)^k \mu(z^{(0)}).$$

Proof. If $\mu(z^{k+1}) \leq \mu(\hat{z}^{(k)})$, then by lemma 3 and (53)

$$\mu(z^{(k+1)}) \leq (1-\alpha^{(k)})(1+\alpha^{(k)})\mu(z^{(k)}).$$

If, on the other hand, $\mu(z^{k+1}) > \mu(\hat{z}^{(k)})$, then by lemma 6, we get

$$\begin{aligned}
(\nu+1)\mu(z^{(k+1)}) &\leq (\nu+1)\mu(\hat{z}^{(k)}) + \rho_1 \\
&\leq (\nu+1)(1-\alpha^{(k)})(1+\alpha^{(k)})\mu(z^{(k)}) + \rho_1.
\end{aligned}$$

So by choosing $\rho_1 = (1-\eta)(\nu+1)\mu(z^{(k)})\left(\alpha^{(k)}\right)^2$, we get either way

$$\begin{aligned}(\nu+1)\mu(z^{(k+1)}) &\leq (\nu+1)(1-(\alpha^{(k)})^2+(1-\eta)(\alpha^{(k)})^2)\mu(z^{(k)})+\rho_1\\ &\leq (\nu+1)(1-\eta\xi^2)\mu(z^{(k)})\\ &\leq (\nu+1)(1-\eta\xi^2)^k\mu(z^{(0)})\end{aligned}$$

and hence $\mu(z^{(k+1)}) \leq \left(1-\eta\xi^2\right)^k \mu(z^{(0)})$. □

So to reach the user specified tolerance $\epsilon \ll 1$, we require

$$k \geq \frac{\log\epsilon}{\log\left(1-\eta\xi^2\right)} = \mathcal{O}\left(\xi^{-2}\log(1/\epsilon)\right).$$

Remark 1. We ensure that the assumption (53) holds in the following way: After computing an ATD step, we check the inequality (53) and if it does not hold, we take more centering steps thus decreasing $\|\psi\|_\infty$ (see lemma 7). Eventually (53) will hold. In fact, *all* of our computational experience with this algorithm shows that (53) *always* holds as long as $s^{(k)}$ is dual feasible, which is ensured by lemma 8. In practice, this safeguard was therefore never activated. It is possible that (53) can be shown to hold generally under certain assumptions.

Remark 2. In [16], a slightly different new primal-dual point $z^{(k+1)}$ is used. The next primal point is instead $x^{(k+1)} = x_c^{(N-1)} - \delta_x^{(N-1)}$ while $\nabla^2 F$ is evaluated at $x_c^{(N-1)}$. This way, the *exact* scaling relation (51) holds which would imply $\psi^{(k+1)} = 0$. With this choice, we would obtain a much simpler right hand side in (37) of lemma 3. However, because the homogenous self-dual model ties the primal and dual problems together via the variables τ and κ, it would not be possible to choose a combination of $y^{(k+1)}$,$\tau^{(k+1)}$ and $\kappa^{(k+1)}$ in such a way that e.g. lemma 5 would still hold. Further, it certainly makes intuitive sense that it is more beneficial to use the better primally centered $x_c^{(N)}$ as the next primal point. Our computational experience indicates that this is better in practice.

Remark 3. Although the two lemmas 3 and 9 indicate that μ and the residuals r_P, r_G and r_D decrease to zero at different rates, we observe in numerical experiments that $d_x^T\psi$ is of much smaller magnitude than $(\nu+1)\mu$. Thus we always see $\mu^{(k+1)} \approx (1-\alpha)\mu^{(k)}$ in practice.

6 Heuristic improvements

In this section we introduce and motivate two heuristic methods to speed up convergence.

6.1 Higher order ATD using BFGS updating

Since each iteration of the centering process is as costly as computing a single ATD, it may be worth investing more computational effort in obtaining a search direction that incorporates higher order information. This could lead to a greater decrease in μ thus saving many future centering steps.

Let us denote the central path by $z(\mu)$. The ATD defined in (27)-(29) can be considered an approximation to the tangent to the central path at μ, i.e. $z'(\mu)$. Let us write the system (27)-(29) as

$$K(z,\mu)z'(\mu) = \phi(z,\mu) \quad \text{or} \quad z'(\mu) = K(z,\mu)^{-1}\phi(z,\mu) =: f(z,\mu).$$

The central path is thus the solution of the ordinary differential equation defined by $z'(\mu) = f(z,\mu)$. A step in the approximate tangent direction is then the same as taking one Euler step for this ODE. We can, however, obtain a direction that contains, for example, second order information by computing a stage-2 Runge-Kutta direction d_2, remembering that each evaluation of f requires solving a system of the type (27)-(29). Such a direction is defined by

$$d_2 = h\left(1-\frac{1}{2\theta}\right)f(z,\mu) + h\frac{1}{2\theta}f(\zeta,\mu(\zeta)) \tag{54}$$

$$\text{where} \quad \zeta = z(\mu) + \theta h f(z,\mu) \tag{55}$$

and where h is the stepsize possible in the direction $f(z,\mu)$ and $\theta \in (0,1]$ is a parameter. The choices $\theta = 1/2$ and $\theta = 1$ correspond to the classical midpoint and trapezoidal rules respectively [7].

Our experience shows that this approach reduces the *total* number of iterations as well as the number of factorizations needed for convergence, *even though* two factorizations are needed to compute d_2.

We can, however, restrict ourselves to just one factorization by using in place of $\nabla^2 F(\zeta_x)$ the BFGS update [20] of $\nabla^2 F(x)$. This requires only the extra evaluation of $\nabla F(\zeta_x)$ and extra back-substitutions. In the next section, we show how this can be done without destroying sparsity in the KKT-system.

6.2 BFGS updating in the centering process

Solving either for a centering step or an ATD step requires the factorization of the sparse $n \times n$ matrix $H := \nabla^2 F(x)$ and of the possibly sparse $m \times m$ matrix $Q = AH^{-1}A^T$. To reduce the total number of factorizations needed in the centering process, we suggest taking J BFGS steps for each normal centering step.

Let us show how this can be done computationally efficient. Let B and M denote the BFGS approximation of the *inverses* of H and Q respectively. Conceptually, we update B to B^+ using BFGS updating, a rank-2 updating scheme: $B^+ = B + k^{(v)}vv^T + k^{(w)}ww^T$. In order to keep the ability to exploit sparsity of A and Q, we do not actually store B or M but simply the Cholesky factors of the most recent H and Q and the sequence of BFGS update vectors. More specifically, for $q \leq J$, let $B^{(q)}$ be the q'th update of H^{-1}, i.e.

$$B^{(q)} = C^{-1}C^{-T} + \Psi\Lambda\Psi^T$$

where $\Psi = [v^{(1)}, \ldots, v^{(q)}, w^{(1)}, \ldots, w^{(q)}], \Lambda = \mathrm{diag}(k_1^{(v)}, \ldots, k_q^{(v)}, k_1^{(w)}, \ldots, k_q^{(w)})$. Then we compute products such as $B^{(q)}r$ by means of

$$B^{(q)}r = C^{-1}(C^{-T}r) + \Psi\left(\Lambda(\Psi^T r)\right).$$

For M, the situation is similar:

$$\begin{aligned} M^{(q)} &= \left(AB^{(q)}A^T\right)^{-1} \\ &= \left(A(H^{-1} + \Psi\Lambda\Psi^T)A^T\right)^{-1} \\ &= \left(Q + \Phi\Lambda\Phi^T\right)^{-1} \end{aligned}$$

where $\Phi = A\Psi$. By the Sherman-Morrison-Woodbury formula, we get

$$M^{(q)} \quad = \quad Q^{-1} - Q^{-1}\Phi\left(\Lambda^{-1} + \Phi^T Q^{-1}\Phi\right)^{-1}\Phi^T Q^{-1}.$$

We can thus compute products like $M^{(q)}r$ by

$$\begin{aligned} M^{(q)}r &= Q^{-1}\left(I - \Phi\left(\Lambda^{-1} + \Phi^T Q^{-1}\Phi\right)^{-1}\Phi^T Q^{-1}\right)r \\ &= D^{-1}D^{-T}\left(r - \Phi\left(\Lambda^{-1} + \Phi^T D^{-1}D^{-T}\Phi\right)^{-1}\Phi^T D^{-1}D^{-T}r\right) \end{aligned}$$

where we remark that 1) only two columns are added to Φ in each iteration so that only two new backsubstitutions in the product $D^{-1}D^{-T}\Phi$ need to be made, 2) Λ is diagonal and thus cheap to invert and 3) the matrix $\left(\Lambda^{-1} + \Phi^T D^{-1}D^{-T}\Phi\right)$ is only of size $2q \times 2q$ and is therefore also cheap to invert.

Using this approach, we need to address how to terminate the centering process. An obvious way is to allow termination only after a normal centering step (that is, not a BFGS step), when the criteria (46)-(49) are satisfied. This, however, means that each centering process will involve at least one full centering step.

Alternatively we can, after each BFGS step, check if $\|r_C - \hat{r}_D\|^*_{-\nabla F(x)} \leq \hat{\mu}\rho_3$. If so, we know that the next dual point is dual feasible (see (50) and lemma 8). Notice that because of (7), the norm $\|v\|^*_{-\nabla F(x)}$ can be computed as $(v^T H^{-1} v)^{1/2}$. Computing this number requires the evaluation and factorization of H. However, since H is blockdiagonal, this operation is cheap. In fact, it is possible simply to analytically compute H^{-1} at each x, since H is block diagonal with block sizes 3×3.

We finally remark that whether or not it is beneficial to take BFGS steps, and if it is, how many should be taken, depends on the cost of building and Cholesky factorizing $AH^{-1}A^T$ relative to the cost of subsequent backsubstitutions. This ratio depends on the dimension and sparsity pattern of A, so we can not say anything about it before knowing A. Since the dimension and sparsity pattern of $AH^{-1}A^T$ are the same regardless of the stage of the algorithm, it is possible to determine the above mentioned ratio at initialization time and that way determine how many BFGS steps should be taken in each iteration of the centering process throughout the algorithm.

Algorithm 2 Homogeneous algorithm with heuristic improvements

Input: Initial point $z^{(0)} = (x^{(0)}; \tau^{(0)}; y^{(0)}; s^{(0)}; \kappa^{(0)})$, μ, J, θ and F.
for $k = 0, 1, 2, \ldots$ **do**
Residuals: Compute $r_P^{(k)}$, $r_D^{(k)}$, $r_G^{(k)}$ and $r_A^{(k)}$
Stopping: If stopping criteria satisfied: **break**
ATD1: Solve the system (27)-(29) for $d^{(k)} = (d_x^{(k)}; d_\tau^{(k)}; d_y^{(k)}; d_s^{(k)}; d_\kappa^{(k)})$.
Line search: Compute a step size h
ATD2: Compute the second order direction $d_2^{(k)}$ by (54)-(55) using h, BFGS updating and reusing the factorization from ATD1.
Line search: Compute a step size $\alpha^{(k)}$
Update: $\hat{z}^{(k)} := z^{(k)} + \alpha^{(k)} d_2^{(k)}$ and $\mu := \mu(\hat{z}^{(k)})$ and $\hat{r}_D^{(k)} = r_D(\hat{z}^{(k)})$
Centering: Compute iterates as in Algorithm 1 but take J BFGS steps for each normal step, stopping according to Section 6.2.
Primal-dual lifting: Set $s^{(k+1)} = -A^T y_c^{(N)} + c\tau_c^{(N)} - \hat{r}_D^{(k)}$
Update: $z^{(k+1)} = (x_c^{(N)}; \tau_c^{(N)}; y_c^{(N)}; s^{(k+1)}; \kappa_c^{(N)})$
end for

6.3 Modified algorithm

Our improved algorithm, which is shown in algoritm 2, implements the pseudo higher order direction described in Section 6.1 and in addition, it takes J BFGS steps as described in Section 6.2. We allow stopping after a BFGS step if $\|r_C - \hat{r}_D\|^*_{-\nabla F(x)} \leq \hat{\mu}\rho_3$ because this seems most effective on the problems we consider. However, we remark that this may not be the case for all problems.

7 Computational results

In this section we present results from running Algorithms 1 and 2 on different test problems. We first introduce the nonsymmetric cones needed for our test problems and then present the test problems. Finally, we include tables with numerical results and discussion.

7.1 Two three-dimensional nonsymmetric cones

In the rest of this paper, we are going to be considering problems involving the following two nonsymmetric convex cones.

The three-dimensional *exponential cone* is defined by

$$\mathcal{K}_{\exp} = \text{closure}\,\{(x_1; x_2; x_3) \in \mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_{++} : \exp(x_1/x_3) \leq x_2/x_3\}$$

for which we are using the LHSCB

$$F_{\exp}(x) = -\log(x_3 \log(x_2/x_3) - x_1) - \log x_2 - \log x_3$$

with barrier parameter $\nu = 3$.

The three-dimensional *power cone* is defined by

$$\mathcal{K}_\alpha = \left\{(x_1; x_2; x_3) \in \mathbb{R} \times \mathbb{R}_+^2 : |x_1| \leq x_2^\alpha x_3^{1-\alpha}\right\}$$

where $\alpha \in [0, 1]$ is a parameter. Notice that $\mathcal{K}_{1/2}$ is the rotated quadratic cone. For all other $\alpha \in (0, 1)$, $\mathcal{K}_\alpha$ is not symmetric. In [8], it was proved that the function

$$F_\alpha(x) = -\log(x_2^{2\alpha} x_3^{2-2\alpha} - x_1^2) - (1-\alpha)\log x_2 - \alpha \log x_3$$

is a LHSCB with parameter $\nu = 3$ for $\mathcal{K}_\alpha$. It is this barrier function we are using in our experiments. Nesterov proposed in [16] a barrier function for the three-dimensional power cone with parameter $\nu = 4$. Our computational experiences show that F_α is better in practice.

7.2 Test problems

In this section, e will denote the vector of all ones. The dimension of e will be clear from the context.

7.2.1 The p-cone problem

Given $A \in \mathbb{R}^{M\times N}$ and $b \in \mathbb{R}^M$, the p-cone problem is the problem

$$\min_x \quad \|x\|_p, \quad \text{s.t. } Ax = b.$$

In [15], it is shown that this is equivalent to

$$\begin{array}{ll} \min\limits_{x,y,t} & t \\ \text{s.t.} & Ax = b, \;\; e^T y = t \\ & (x_j; y_j; t) \in K_{(1/p)}, \quad j = 1, \ldots, M. \end{array}$$

7.2.2 The facility location problem

Given M points (locations) in $\mathbb{R}^N$: $C^{(j)}, j = 1, \ldots, M$, we want to find the point z with the minimal sum of (weighted) distances to the locations $C^{(j)}$ measured in p_j-norms, $p_j \geq 1$. That is

$$\min_z \quad \sum_{j=1}^{M} a_j \|z - C^{(j)}\|_{p_j} \tag{56}$$

where $a_j \geq 0$ are the weights. Let us define the variable

$$x = (z^+; z^-; v^{(1)}; \ldots; v^{(M)}; w^{(1)}; \ldots; w^{(M)}; u^{(1)}; \ldots; u^{(M)}) \in \mathbb{R}^{2N+3MN}.$$

We can then formulate (56) as

$$\begin{array}{lll} \min\limits_{z^+,z^-,v,w,u} & \sum\limits_{j=1}^{M} a_j u_1^{(j)} & \\ \text{s.t.} & v^{(j)} = z^+ - z^- - C^{(j)} & j = 1, \ldots, M \\ & e^T w^{(j)} = u_1^{(j)}, \;\; u_1^{(j)} = u_2^{(j)} = \cdots = u_N^{(j)} & j = 1, \ldots, M \\ & (v_i^{(j)}; w_i^{(j)}; u_i^{(j)}) \in \mathcal{K}_{1/p_j} & j = 1, \ldots, M, \; i = 1, \ldots, N \\ & z^+ \geq 0, \;\; z^- \geq 0 & \end{array}$$

and it is this formulation we use in Section 7.3.

7.2.3 Entropy maximization

Given $A \in \mathbb{R}^{M\times N}$, $b \in \mathbb{R}^M$ and $d \in \mathbb{R}^N$, the entropy maximization problem is

$$\begin{array}{ll} \min\limits_x & \sum\limits_{j=1}^{N} d_j x_j \log x_j \\ \text{s.t.} & Ax = b \\ & x_j \geq 0, \quad j = 1, \ldots, N \end{array}$$

which can be formulated as

$$\begin{array}{ll} \min\limits_{x,u} & -d^T u \\ \text{s.t.} & Ax = b, \; v = e \\ & (u_j; v_j; x_j) \in \mathcal{K}_{\exp}, \;\; j = 1, \ldots, N. \end{array}$$

7.2.4 Geometric programming

This is a problem of the type

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & f^{(0)}(\boldsymbol{x}) \\ \text{s.t.} & g^{(j)}(\boldsymbol{x}) = 1, \quad j = 1, \ldots, M \\ & f^{(j)}(\boldsymbol{x}) \leq 1, \quad j = 1, \ldots, P \end{array}$$

where $g^{(j)}$ are monomials and $f^{(j)}$ are posynomials:

$$g(\boldsymbol{x}) = k_j \boldsymbol{x}^{\boldsymbol{b}^{(j)}}, \quad f^{(j)}(\boldsymbol{x}) = \sum_{i=1}^{N_j} d_i \boldsymbol{x}^{\boldsymbol{a}_i^{(j)}}$$

where we have used the notation

$$\boldsymbol{x}^{\boldsymbol{v}} := \prod_{i=1}^{n} x_i^{v_i}, \quad x_i > 0.$$

With the j'th posynomial $f^{(j)}$, we associate

- the matrix $\boldsymbol{A}^{(j)} := \left(\boldsymbol{a}_1^{(j)}, \boldsymbol{a}_2^{(j)}, \ldots, \boldsymbol{a}_{N_j}^{(j)}\right)^T \in \mathbb{R}^{N_j \times N}$,
- the vector $\boldsymbol{d}^{(j)} = (d_1^{(j)}, \ldots, d_{N_j}^{(j)})^T \in \mathbb{R}^{N_j \times 1}$ and
- the vector $\boldsymbol{c}^{(j)} = \log\left(\boldsymbol{d}^{(j)}\right) = (\log(d_1), \ldots, \log(d_{N_j}))^T \in \mathbb{R}^{N_j \times 1}$

Similarly, we associate with the j'th monomial $g^{(j)}$

- the vector $\boldsymbol{b}^{(j)}$
- the scalar $k^{(j)}$
- the scalar $h^{(j)} = \log(k^{(j)})$

Using the change of variables

$$u_i = \log(x_i) \quad \Leftrightarrow \quad x_i = \exp(u_i)$$

for all i, we can write the problem in conic form:

$$\begin{array}{lll} \min\limits_{\boldsymbol{u}_+, \boldsymbol{u}_-, \boldsymbol{w}, \boldsymbol{v}, \boldsymbol{y}, t^{(0)}} & t^{(0)} & \\ \text{s.t.:} & \boldsymbol{B}(\boldsymbol{u}_+ - \boldsymbol{u}_-) + \boldsymbol{h} = 0 & \\ & \boldsymbol{w}^{(j)} = \boldsymbol{A}^{(j)}(\boldsymbol{u}_+ - \boldsymbol{u}_-) + \boldsymbol{c}^{(j)} & j = 0, \ldots, P \\ & \boldsymbol{e}^T \boldsymbol{v}^{(j)} = t^{(j)}, \quad \boldsymbol{y}^{(j)} = \boldsymbol{e} & j = 0, \ldots, P \\ & \boldsymbol{u}_+, \boldsymbol{u}_-, t^{(0)} \geq 0 & \\ & \left(w_i^{(j)}; v_i^{(j)}; y_i^{(j)}\right) \in K_{\exp} & j = 0, \ldots, P, \ i = 1, \ldots, N_j \end{array}$$

where $\boldsymbol{h} = (h^{(1)}, \ldots, h^{(M)})^T \in \mathbb{R}^{M \times 1}$ and $\boldsymbol{B} = \left(\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(M)}\right)^T \in \mathbb{R}^{M \times N}$.

Param.	J	θ	σ	ρ_P	ρ_D	ρ_A	ρ_G	ρ_I
Value	3	0.70	0.80	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-8}

Table 1: Parameters used.

7.3 Results

Whenever we find a point $z^* = (x^*; \tau^*; y^*; s^*; \kappa^*)$ such that

$$(\|r_P(z^*)\|_\infty \leq \rho_P) \ \wedge \ (\|r_D(z^*)\|_\infty \leq \rho_D)$$

we conclude that

- if $\|r_A(z^*)\|_\infty \leq \rho_A$ then $(x^*, y^*, s^*)/\tau^*$ is feasible and optimal for (1). Otherwise
- if $\|r_G(z^*)\|_\infty \leq \rho_G$ and $\tau^* \leq \rho_I \max(1, \kappa^*)$, (1) is deemed infeasible, specifically
  - primal infeasible if $b^T y^* > 0$
  - dual infeasible if $c^T x^* < 0$.

Throughout we used the parameters displayed in table 1 on the preceding page.

The remaining tables in this section show the number of iteration (it), the *total* number of factorizations made (chols) and the average number of *full* centering steps per iteration (ce). Finally, for all but the facility location problems, we show the termination status (st). `opt` means that an optimal solution was found, `ipr/idu` means a primal/dual infeasibility certificate was found and `roa/roc` means that the algorithm was terminated because rounding errors prevented further progress.

	Problem		*Alg. 1*				*Alg. 2*				`CVX/SeDuMi`	
	name & size	p	it	chols	ce	st	it	chols	ce	st	it	st
	`stocfor1`	1.0	11	37	2.4	`opt`	9	13	0.4	`opt`	12	`opt`
	$M = 117$	3.0	14	62	3.4	`opt`	12	15	0.2	`opt`	19	`opt`
	$N = 165$	7.0	28	134	3.8	`opt`	21	22	0.0	`opt`	21	`roa`
	$\text{nnz}(A) = 501$	12.0	29	137	3.7	`opt`	22	26	0.2	`opt`	22	`roa`
		20.0	28	131	3.7	`opt`	20	24	0.2	`opt`	25	`opt`
	`blend`	1.0	11	35	2.2	`opt`	9	13	0.4	`opt`	5	`opt`
	$M = 74$	3.0	13	58	3.5	`opt`	11	21	0.9	`opt`	18	`opt`
	$N = 114$	7.0	17	77	3.5	`opt`	13	24	0.8	`opt`	20	`opt`
	$\text{nnz}(A) = 522$	12.0	19	90	3.7	`opt`	14	27	0.9	`opt`	21	`opt`
		20.0	19	97	4.1	`opt`	14	25	0.8	`opt`	22	`opt`
	`share2b`	1.0	11	40	2.6	`opt`	10	16	0.6	`opt`	13	`opt`
	$M = 96$	3.0	12	56	3.7	`opt`	10	18	0.8	`opt`	19	`opt`
	$N = 162$	7.0	13	62	3.8	`opt`	11	22	1.0	`opt`	17	`roa`
	$\text{nnz}(A) = 777$	12.0	13	62	3.8	`opt`	11	20	0.8	`opt`	20	`opt`
		20.0	13	62	3.8	`opt`	11	19	0.7	`opt`	20	`opt`
	`share1b`	1.0	15	50	2.3	`opt`	14	19	0.4	`opt`	18	`opt`
	$M = 117$	3.0	17	72	3.2	`opt`	13	22	0.7	`opt`	21	`opt`
	$N = 253$	7.0	16	67	3.2	`opt`	12	22	0.8	`opt`	21	`opt`
	$\text{nnz}(A) = 1179$	12.0	16	66	3.1	`opt`	12	20	0.7	`opt`	22	`opt`
		20.0	15	64	3.3	`opt`	11	19	0.7	`opt`	23	`opt`
	`bore3d`	1.0	11	25	1.3	`opt`	8	9	0.1	`opt`	3	`opt`
	$M = 231$	3.0	11	25	1.3	`opt`	8	9	0.1	`opt`	6	`opt`
	$N = 334$	7.0	11	25	1.3	`opt`	8	9	0.1	`opt`	6	`opt`
	$\text{nnz}(A) = 1444$	12.0	12	27	1.2	`opt`	8	9	0.1	`opt`	6	`opt`
		20.0	12	27	1.2	`opt`	8	9	0.1	`opt`	6	`opt`
	`scagr25`	1.0	14	50	2.6	`opt`	12	23	0.9	`opt`	13	`opt`
	$M = 471$	3.0	10	37	2.7	`opt`	9	14	0.6	`opt`	18	`opt`
	$N = 671$	7.0	11	41	2.7	`opt`	8	14	0.8	`opt`	17	`opt`
	$\text{nnz}(A) = 1725$	12.0	11	41	2.7	`opt`	8	12	0.5	`opt`	18	`opt`
		20.0	11	41	2.7	`opt`	8	11	0.4	`opt`	19	`opt`
	`sctap1`	1.0	13	43	2.3	`opt`	12	17	0.4	`opt`	12	`opt`
	$M = 300$	3.0	12	48	3.0	`opt`	8	15	0.9	`opt`	17	`opt`
	$N = 660$	7.0	14	59	3.2	`opt`	10	20	1.0	`opt`	18	`opt`
	$\text{nnz}(A) = 1872$	12.0	18	80	3.4	`opt`	13	27	1.1	`opt`	21	`opt`
		20.0	21	94	3.5	`opt`	14	28	1.0	`opt`	23	`opt`
	`bandm`	1.0	12	42	2.5	`opt`	10	16	0.6	`opt`	14	`opt`
	$M = 305$	3.0	25	117	3.7	`opt`	19	38	1.0	`opt`	21	`opt`
	$N = 472$	7.0	31	147	3.7	`opt`	22	43	1.0	`opt`	25	`opt`
	$\text{nnz}(A) = 2494$	12.0	31	152	3.9	`opt`	22	43	1.0	`opt`	25	`opt`
		20.0	30	148	3.9	`opt`	21	40	0.9	`opt`	25	`opt`

Table 2: Results for p-cone problems.

	Problem				*Alg. 1*			*Alg. 2*		
	N	M	ν	$\bar{p}$	it	chols	ce	it	chols	ce
	4	3	44	5.0	13.8	33.6	1.4	13.4	16.5	0.2
	8	3	88	7.2	16.7	42.3	1.5	16.2	19.3	0.2
	4	10	128	6.0	14.8	41.2	1.8	13.5	18.5	0.4
	12	3	132	5.3	20.5	53.2	1.6	19.0	23.4	0.2
	20	3	220	6.4	22.2	60.7	1.7	21.0	24.9	0.2
	4	19	236	5.7	14.6	42.5	1.9	13.4	18.8	0.4
	8	10	256	6.6	17.9	50.4	1.8	16.3	22.0	0.4
	12	10	384	6.0	21.2	66.1	2.1	19.0	25.9	0.4
	4	32	392	5.9	14.2	44.8	2.2	12.2	18.2	0.5
	8	19	472	5.7	17.7	56.3	2.2	16.1	22.7	0.4
	20	10	640	5.6	25.5	78.4	2.1	22.8	30.0	0.3
	12	19	708	5.4	20.5	65.5	2.2	18.9	26.4	0.4
	8	32	784	6.1	18.4	61.4	2.4	16.1	23.2	0.5
	12	32	1176	5.7	20.5	71.9	2.5	16.8	26.3	0.6
	20	19	1180	5.5	23.6	79.2	2.4	20.5	28.8	0.4
	20	32	1960	5.8	24.9	87.4	2.5	21.4	30.3	0.4

Table 3: Results for facility location problems. The algorithms always terminated after reaching optimality as all problem instances were feasible by construction.

Table 2 shows results for the p-cone problem. We show the results for the two variants of our algorithm and for the solver `SeDuMi` (see [22]) when called through `CVX` (see [10]). `CVX` approximates the solution by solving an approximately equivalent self-scaled problem using `SeDuMi`.

The matrices used are from the NETLIB collection. We see that particularly Algorithm 2 performs very well compared to `SeDuMi`. While the table shows the size of A, it does not show the problem size of the final problem solved by either of the solvers. We remark that the number of variables and constraints for Algorithms 1 and 2 stay the same regardless of p while the problem size for `SeDuMi` grows with p. Of course, sufficient sparsity in the problem may mean that it is not necessarily slower. For Algorithm 2, the number of iterations never exceeded 23 and the number of Cholesky factorizations of $AH^{-1}A^T$ never exceeded 43.

Table 3 shows the performances of the two variants of the algorithm when run on random instances of the facility location problem. For each pair (N, M) we generated 10 instances each with $C^{(j)}$ chosen at random from the standard normal distribution and p_j and a_j chosen randomly from a uniform distribution. The column labelled $\bar{p}$ shows the average $M^{-1}\sum_{j=1}^{M} p_j$. We see again that for Algorithm 2, the iteration count is in the region between 10 and 22 while the number of Cholesky factorizations never exceeds 31. These results can be loosely compared with the computational results in [8, Table 4.1, page 142]. There, a dual variant of the algorithm of [16] is used. Overall, our Algorithm 2 performs better, both in terms of iterations and factorizations.

Table 4 shows results from solving a set of real-world entropy problems kindly supplied to us by Erling D. Andersen and Joachim Dahl of MOSEK ApS, Denmark. Generally the problems have many variables compared to the number of constraints resulting in a very "fat" constraint matrix A. For these problems we compare our algorithms to the commercial solver from MOSEK, which solves the monotone complementarity problem corresponding to the entropy problem. We see that, except for a few of the problems, Algorithm 1 is clearly inferior to MOSEKs solver. Our Algorithm 2 performs much better than Algorithm 1, but still used cummulatively about 12% more iterations and about 55% more Cholesky factorizations than MOSEK.

Finally, table 5 shows results from applying our algorithms to a set of geometric programs supplied to us by MOSEK. The column labelled `dod` denotes the *degree of difficulty* of the problem. For these problems we see that Algorithm 2 performs quite well producing results comparable to those of MOSEK. Algorithm 2 uses cummulatively about 29% less iterations and 14% less Cholesky factorizations than MOSEK. Notice that Algorithm 2 performs particularly well on the large problems `cx02-100`, `cx02-200`, `mra01` and `mra02`, all of which have `dod` > 800.

	Problem			*Alg. 1*				*Alg. 2*				mskenopt	
	name	N	M	it	chols	ce	st	it	chols	ce	st	it	st
	prob	17	15	14	35	1.5	opt	9	15	0.7	opt	8	opt
	prob2	18	14	11	34	2.1	opt	9	16	0.8	opt	8	opt
	ento46	130	21	37	139	2.8	opt	27	43	0.6	opt	42	opt
	ento47	255	21	42	139	2.3	opt	30	48	0.6	opt	54	opt
	ento28	740	16	66	236	2.6	opt	52	77	0.5	opt	63	opt
	ento30	740	21	75	246	2.3	opt	58	70	0.2	opt	55	opt
	ento31	740	21	75	246	2.3	opt	58	70	0.2	opt	55	opt
	ento22	794	28	41	163	3.0	ipr	31	50	0.6	ipr	14	ipr
	ento21	931	28	123	276	1.2	ipr	73	83	0.1	ipr	18	ipr
	a_tb	1127	25	68	233	2.4	opt	64	82	0.3	opt	97	opt
	ento23	1563	28	71	186	1.6	ipr	54	63	0.2	ipr	14	ipr
	ento20	1886	28	70	243	2.5	ipr	68	93	0.4	ipr	21	ipr
	a_12	2183	37	88	326	2.7	opt	77	110	0.4	opt	47	opt
	ento12	2183	37	38	162	3.3	ipr	28	53	0.9	ipr	13	ipr
	a_13	3120	37	88	770	7.8	ipr	82	101	0.2	ipr	20	ipr
	a_23	3301	37	48	191	3.0	ipr	37	71	0.9	ipr	20	ipr
	a_34	3905	37	52	224	3.3	opt	37	74	1.0	ipr	17	ipr
	a_14	3986	37	85	285	2.4	ipr	77	102	0.3	ipr	20	ukn
	a_35	4333	37	79	267	2.4	ipr	69	90	0.3	ipr	18	ipr
	a_bd	4695	26	73	312	3.3	opt	63	96	0.5	opt	78	opt
	ento2	4695	26	73	312	3.3	opt	63	96	0.5	opt	78	opt
	a_24	5162	37	61	248	3.1	ipr	49	84	0.7	ipr	23	ipr
	ento3	5172	28	98	348	2.6	opt	80	114	0.4	opt	146	opt
	ento50	5172	28	98	348	2.6	opt	80	114	0.4	opt	146	opt
	a_15	5668	37	224	542	1.4	ipr	180	198	0.1	ipr	34	ipr
	a_25	6196	37	107	403	2.8	opt	99	136	0.4	opt	122	opt
	a_36	7497	37	63	267	3.2	ipr	46	87	0.9	ipr	18	ipr
	a_45	7667	37	119	362	2.0	ipr	110	131	0.2	ipr	23	ipr
	ento26	7915	28	79	277	2.5	opt	85	112	0.3	opt	131	opt
	a_16	8528	37	200	656	2.3	opt	172	208	0.2	opt	135	opt
	a_26	9035	37	66	274	3.2	opt	48	97	1.0	opt	113	opt
	ento45	9108	37	92	332	2.6	opt	75	110	0.5	opt	149	opt
	a_46	9455	37	68	265	2.9	ipr	49	91	0.9	ipr	20	ipr
	a_56	9702	37	131	434	2.3	opt	120	154	0.3	opt	123	opt
	ento25	10142	28	276	670	1.4	opt	228	247	0.1	opt	149	opt
	entodif	12691	40	85	359	3.2	opt	60	117	0.9	opt	155	opt
	ento48	15364	31	28	119	3.2	opt	21	51	1.4	opt	47	opt

Table 4: Results for entropy problems.

8 Conclusions and Future Work

We have presented a homogeneous primal-dual interior point algorithm for nonsymmetric conic optimization based on an algorithm previously presented by Nesterov [16]. We have proven convergence of the simplest variant of the algorithm and shown that the efficiency can be significantly improved by employing quasi-Newton techniques to update the Hessian of the barrier function. We have shown how this can be done without loosing the ability to exploit sparsity. Finally we have presented extensive computational results that indicate the algorithm works well in practice.

By inspecting the tables in Section 7.3, we see that

- The performance of the algorithms depends a lot on the type of problem.
- Throughout, Algorithm 2 is superior to Algorithm 1.
- For the p-cone problems, Algorithm 2 is comparable in performance to `SeDuMi` when called via `CVX`.
- For the facility location problems, Algorithm 2 is superior to previously published results [8] of an algorithm similar to the one presented in [16].
- For the entropy maximization problems, Algorithm 1 is clearly inferior to Mosek while Algorithm 2 is just slightly inferior.
- For the geometric programs, Algorithm 2 performed slightly better than Mosek when measuring iterations or the total number of Cholesky factorizations.

	Problem			*Alg. 1*				*Alg. 2*				`mskgpopt`	
	name	n	dod	it	chols	ce	st	it	chols	ce	st	it	st
	`beck751`	7	10	24	62	1.6	`opt`	15	18	0.2	`opt`	18	`opt`
	`beck752`	7	10	23	59	1.6	`opt`	15	17	0.1	`opt`	28	`opt`
	`beck753`	7	10	6	14	1.3	`opt`	7	7	0.0	`opt`	10	`opt`
	`bss2`	2	1	9	21	1.3	`opt`	8	8	0.0	`opt`	5	`opt`
	`car`	37	104	26	58	1.2	`opt`	19	20	0.1	`opt`	46	`opt`
	`cx02-100`	100	5146	22	64	1.9	`opt`	30	36	0.2	`opt`	68	`opt`
	`cx02-200`	200	20296	29	86	2.0	`opt`	25	32	0.3	`opt`	70	`opt`
	`demb761`	11	19	20	48	1.4	`ipr`	14	14	0.0	`ipr`	10	`opt`
	`demb762`	11	19	11	28	1.5	`opt`	10	14	0.4	`opt`	11	`opt`
	`demb763`	11	19	11	28	1.5	`opt`	10	14	0.4	`opt`	11	`opt`
	`demb781`	2	1	6	13	1.2	`opt`	7	7	0.0	`opt`	7	`opt`
	`fang88`	11	16	11	28	1.5	`opt`	10	12	0.2	`opt`	11	`opt`
	`fiac81a`	22	50	18	57	2.2	`opt`	14	20	0.4	`opt`	16	`opt`
	`fiac81b`	10	9	18	45	1.5	`ipr`	13	13	0.0	`ipr`	10	`opt`
	`gptest`	4	1	11	24	1.2	`opt`	10	10	0.0	`opt`	5	`opt`
	`jha88`	30	274	33	116	2.5	`opt`	23	35	0.5	`opt`	13	`opt`
	`mra01`	61	844	26	79	2.0	`opt`	22	30	0.4	`opt`	58	`opt`
	`mra02`	126	3494	62	157	1.5	`roc`	47	53	0.1	`opt`	53	`opt`
	`rijc781`	4	1	11	24	1.2	`opt`	10	10	0.0	`opt`	5	`opt`
	`rijc782`	3	5	12	31	1.6	`opt`	10	14	0.4	`opt`	8	`opt`
	`rijc783`	4	7	19	45	1.4	`opt`	13	15	0.2	`opt`	7	`opt`
	`rijc784`	4	3	11	28	1.5	`ipr`	9	9	0.0	`ipr`	6	`opt`
	`rijc785`	8	3	11	27	1.5	`opt`	9	13	0.4	`opt`	9	`opt`
	`rijc786`	8	3	10	25	1.5	`opt`	9	13	0.4	`opt`	6	`opt`
	`rijc787`	7	40	15	44	1.9	`opt`	13	17	0.3	`opt`	36	`opt`

Table 5: Results for geometric programs.

Comparing the kind of algorithm we have presented with a PDIPM for self-scaled cones, we see that the major difference is the need for a seperate centering process. Nesterov remarks in [16] that this process can be seen as the process of finding a scaling point, which naturally is a more complex problem when the cone is not symmetric. We can not compute it analytically, so an iterative procedure is necessary.

This difference is interesting theoretically as well as practically. For the problems we have considered, the centering problem certainly is a relatively "easy" problem compared to the full problem, in the sense that we do not need a very *accurately* centered point. We have seen in our experiments with Algorithm 1 that rarely more than 3 or 4 full centering steps are required.

If some of the work in the centering process can be saved – as we have tried to do in Algorithm 2 by employing BFGS updating – it may be possible to bring the total computational effort per iteration close to that of a PDIPM for self-scaled cones. Saving even more work in the centering process is a topic of interest for future work.

Acknowledgments

The authors thank Erling D. Andersen and Joachim Dahl of MOSEK ApS for lots of insights and for supplying us with test problems for the geometric programs and the entropy problems.

References

[1] E. D. Andersen and K. D. Andersen. *The MOSEK interior point optimization for linear programming: an implementation of the homogeneous algorithm*, pages 197–232. Kluwer Academic Publishers, 1999.

[2] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, February 2003.

[3] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Mathematical Programming*, 84(2):375–399, February 1999.

[4] A. Ben-Tal and A. S. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications (MPS-SIAM Series on Optimization)*. Society for Industrial Mathematics, August 2001.

[5] S. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi. A Tutorial on Geometric Programming. *Optimization and Engineering*, 8:67–127, 2007.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[7] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2 edition, June 2008.

[8] P. R. Chares. *Cones and Interior-Point Algorithms for Structured Convex Optimization involving Powers and Exponentials*. PhD thesis, Uni. Catholique de Louvain, 2009.

[9] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics)*. Society for Industrial Mathematics, January 1987.

[10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, October 2010.

[11] O. Güler. Barrier Functions in Interior Point Methods. *Mathematics of Operations Research*, 21(4), 1996.

[12] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984.

[13] Z. Q. Luo, J. F. Sturm, and S. Zhang. Conic convex programming and self-dual embedding. *Optimization Methods and Software*, 14(3):169–218, 2000.

[14] A. S. Nemirovski and M. J. Todd. Interior-point methods for optimization. *Acta Numerica*, 17(-1):191–234, 2008.

[15] Y. E. Nesterov. Constructing self-concordant barriers for convex cones. *CORE Discussion Paper*, (2006/30), March 2006.

[16] Y. E. Nesterov. Towards Nonsymmetric Conic Optimization. *CORE Discussion Paper*, (2006/28), March 2006.

[17] Y. E. Nesterov and A. S. Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming (Studies in Applied and Numerical Mathematics)*. Society for Industrial Mathematics, 1994.

[18] Y. E. Nesterov and M. J. Todd. Self-Scaled Barriers and Interior-Point Methods for Convex Programming. *Mathematics of Operations Research*, 22(1), 1997.

[19] Y. E. Nesterov and M. J. Todd. Primal-Dual Interior-Point Methods for Self-Scaled Cones. *SIAM J. on Optimization*, 8(2):324–364, 1998.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer, 2nd edition, July 2006.

[21] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization (MPS-SIAM Series on Optimization)*. Society for Industrial Mathematics, 1st edition, January 1987.

[22] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, (12):625–653, 1999.

[23] J. F. Sturm. Implementation of Interior Point Methods for Mixed Semidefinite and Second Order Cone Optimization Problems. *Optimization Methods and Software*, 17(6):1105–1154, 2002.

[24] L. Tuncel. Primal-Dual Symmetry and Scale Invariance of Interior-Point Algorithms for Convex Optimization. *Mathematics of Operations Research*, 23(3):708–718, August 1998.

[25] L. Tuncel. Generalization Of Primal-Dual Interior-Point Methods To Convex Optimization Problems In Conic Form, 1999.

[26] X. Xu, P. F. Hung, and Y. Ye. A simplified homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 62(1):151–171, December 1996.

[27] G. Xue and Y. Ye. An Efficient Algorithm for Minimizing a Sum of p-Norms. *SIAM J. on Optimization*, 10(2):551–579, 1999.