



# 1 Interior Point Methods for Second-Order Cone 2 Programming and OR Applications

3 YU-JU KUO

4 *Mathematics Department, Indiana University of Pennsylvania, Indiana, PA 15705, USA*

5 HANS D. MITTELMANN

beck@plato.la.asu.edu

6 *Department of Mathematics & Statistics, Arizona State University, Tempe, AZ 85287-1804, USA*

7 *Received June 19, 2002; Revised September 11, 2003*

8 **Abstract.** Interior point methods (IPM) have been developed for all types of constrained optimization problems.  
9 In this work the extension of IPM to second order cone programming (SOCP) is studied based on the work of  
10 Andersen, Roos, and Terlaky. SOCP minimizes a linear objective function over the direct product of quadratic  
11 cones, rotated quadratic cones, and an affine set. It is described in detail how to convert several application  
12 problems to SOCP. Moreover, a proof is given of the existence of the step for the infeasible long-step path-  
13 following method. Furthermore, variants are developed of both long-step path-following and of predictor-corrector  
14 algorithms. Numerical results are presented and analyzed for those variants using test cases obtained from a number  
15 of application problems.

16 **Keywords:**

Au: Pls.  
provide  
keywords.

## 17 1. Introduction and definitions

18 Over the past few years, primal-dual interior point methods (IPM) have been developed for  
19 all kinds of nonlinear optimization problems. Second-order cone programming (SOCP) is  
20 one of these. SOCP addresses the problem of minimizing a linear objective function over the  
21 intersection of an affine set and the direct product of quadratic cones. Numerous applications  
22 have been discussed in [5, 15]. Recently, many reseachers have shown that the primal-dual  
23 IPM has high theoretical efficiency for SOCP. In particular, Tsuchiya [13] and Monteiro  
24 and Tsuchiya [9] have proved the complexity of variants of IPMs based on different scaling  
25 directions. Specifically, Tsuchiya [13] shows that the long-step path-following algorithm  
26 using the Nesterov and Todd (NT) direction has  $O(k \log \varepsilon^{-1})$  iteration complexity which is  
27 also the best result for the scaling methods they considered.

28 In this section we introduce the second order problem (SOCP) and quote some basic  
29 definitions for the primal-dual interior point approach. The next section contains as on  
30 of the main results a proof of the existence of the step length for the long-step method.  
31 A number of applications and in each case their reformulation as SOCP is presented in  
32 Section 3. Subsequently, the various algorithms we have implemented are given in detail  
33 and compared on a test set derived from the application problems. We conclude with a  
34 summary in the last section.

Second-order cone programming addresses the problem of minimizing a linear objective function over the intersection of an affine set and the direct product of quadratic cones. 35 36

$$\begin{aligned}
 \text{(SOCP)} \quad & \min \quad c^T x \\
 & \text{s.t.} \quad Ax = b \\
 & \quad \quad x \in K,
 \end{aligned} \tag{1.1}$$

where  $K$  is a closed convex cone. In addition to (1.1), we suppose  $A$  is a  $m \times n$  real matrix and of full rank. Subsequently, the dual problem is 37 38

$$\begin{aligned}
 \text{(DSOCP)} \quad & \max \quad b^T y \\
 & \text{s.t.} \quad A^T y + s = c \\
 & \quad \quad s \in K^* = \{s \mid x^T s \geq 0, x \in K\}.
 \end{aligned} \tag{1.2}$$

The definition of the quadratic cone  $K$  is the following: 39

*Definition 1.1.* The second order cone  $K$  is defined as,  $K = K^1 \times K^2 \times \cdots \times K^k$ , i.e., the direct product of  $k$  cones, where  $K^i$  is one of the three cones below, 40 41

1.  $\mathbf{R}^+ = \{x^i \in \mathbf{R} \mid x \geq 0\}$ . 42
2. quadratic cone  $K^q = \{x^i \in \mathbf{R}^{n_i} \mid \|x_{2:n_i}^i\|^2 \leq (x_1^i)^2, x_1^i \geq 0\}$ . 43
3. rotated quadratic cone  $K^r = \{x^i \in \mathbf{R}^{n_i} \mid \|x_{3:n_i}^i\|^2 \leq 2x_1^i x_2^i, x_1^i, x_2^i \geq 0\}$ . 44

Note that  $K = K^*$ . Moreover, there is a linear transformation  $T$  between  $K^q$  and  $K^r$  such that 45 46

$$x^i \in K^r \Leftrightarrow T^i x^i \in K^q,$$

where 47

$$T^i = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbf{R}^{n_i \times n_i}. \tag{1.3}$$

Here,  $T^i$  is a  $n_i \times n_i$  identity matrix if  $x^i \in K^q$ . The existence of the linear transformation  $T^i$  not only extends the problem to rotated quadratic cones but also helps to apply the 48 49

50 convergence theorem for  $K^q$  to  $K^r$ . Next, we partition  $x$  according to the cones, i.e.,

$$x = \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^k \end{pmatrix}.$$

51 We introduce two nonnegative scalar variables  $\tau, \kappa$  and define the general Goldman-Tucker  
52 homogeneous model as follows:

$$\begin{aligned} -c^T x + b^T y - \kappa &= 0, \\ Ax - b\tau &= 0, \\ A^T y + s - c\tau &= 0, \\ (x, \tau) \in \bar{K}, \quad (s, \kappa) \in \bar{K}^*, \end{aligned} \tag{1.4}$$

53 where  $\bar{K} = K \times \mathbf{R}_+$ , and  $\bar{K}^* = K^* \times \mathbf{R}_+$ . In order to solve (1.4) with the complementarity  
54 conditions, the following definitions are necessary,

55 *Definition 1.2.* Given

$$u \in \mathbf{R}^n, \quad \text{then } U = \text{mat}(u) = \begin{pmatrix} u_1 & u_{2:n}^T \\ u_{2:n} & u_1 I \end{pmatrix},$$

56 where

$$u_{2:n} = \begin{pmatrix} u_2 \\ \vdots \\ u_n \end{pmatrix}$$

57 and  $I$  is the  $(n-1) \times (n-1)$  identity matrix.

58 Next, we define  $X^i$  for each type of cone:

59 *Definition 1.3.*

- 60 1.  $x^i \in \mathbf{R}_+$ :  $X^i = x^i$ , and  $S^i = s^i$ ,
- 61 2.  $x^i \in K^q$ :  $X^i = \text{mat}(x^i)$ , and  $S^i = \text{mat}(s^i)$ ,
- 62 3.  $x^i \in K^r$ :  $X^i = \text{mat}(Tx^i)$ , and  $S^i = \text{mat}(Ts^i)$ , where  $T$  is in (1.3).

63 Subsequently, the complementarity conditions can be written as a nonlinear system.

**Lemma 1.4.** *Let  $x, s \in K$ , then  $x$  and  $s$  are complementary, i.e.,  $x^T s = 0$ , if and only if* 64

$$X^i S^i e^i = S^i X^i e^i = 0, \quad i = 1 : k,$$

where  $X^i$  and  $S^i$  are given in Definition 1.3.  $e^i \in R^{n_i}$  is the first unit vector. 65

See [1] for proof. An initial point  $(x^{(0)}, \tau^{(0)}, y^{(0)}, s^{(0)}, \kappa^{(0)})$  is given by 66

$$(x^{(0)}, \tau^{(0)}), (s^{(0)}, \kappa^{(0)}) \in \text{int}(\bar{K}).$$

Thus, the central path can be defined as: 67

*Definition 1.5.* The points  $(x, \tau, y, s, \kappa)$  of the central path  $C$  are defined by the following nonlinear equations, 68  
69

$$\begin{aligned} Ax - b\tau &= \sigma(Ax^{(0)} - b\tau^{(0)}), \\ A^T y + s - c\tau &= \sigma(A^T y^{(0)} + s^{(0)} - c\tau^{(0)}), \\ -c^T x + b^T y - \kappa &= \sigma(-c^T x^{(0)} + b^T y^{(0)} - \kappa^{(0)}), \\ XSe &= \sigma\mu^{(0)}e, \\ \tau\kappa &= \sigma\mu^{(0)}, \end{aligned} \quad (1.5)$$

where the centering parameter  $\sigma \in [0, 1]$ , the duality measure is 70

$$\mu = \frac{x^T s + \tau\kappa}{k + 1},$$

$$X = \text{diag}\{X^1, \dots, X^k\}, S = \text{diag}\{S^1, \dots, S^k\}, \quad \text{and} \quad e = \begin{pmatrix} e^1 \\ \vdots \\ e^k \end{pmatrix}, \quad e^i \in R^{n_i}$$

is the first unit vector. 71

There are different ways to define the neighborhood. We choose the one similar to  $N_{-\infty}(\gamma)$ , 72  
73

$$N(\gamma) = \{(x, \tau), (s, \kappa) \in \text{int}(\bar{K}) \mid \sqrt{(x^i)^T Q^i x^i (s^i)^T Q^i s^i} \geq \gamma\mu, \forall i, \text{ and } \tau\kappa \geq \gamma\mu\}.$$

In the original infeasible interior point method, the residuals only appear in equations after applying Newton's method. It is easy to verify that the ratio between the new and old residual is  $(1 - \alpha)$ , where  $\alpha$  is the step length. It is faster than the rate at which the complementarity gap decreases. This central path gives the same rate for infeasibility and the complementary gap. In experiments, we see the latter version needs less iterations than the previous one. 74  
75  
76  
77  
78

79 The well-definedness of (1.5) is not guaranteed. One way to assure that the search direction  
80 is well-defined is applying scaling schemes before using Newton's method, then scaling  
81 back to the original space. Next, we give the definition of scaling matrices [1].

82 *Definition 1.6.*  $W^i \in \mathbf{R}^{n_i \times n_i}$  is a scaling matrix if it satisfies

- 83 1.  $W^i$  is a symmetric positive definite matrix, and  
84 2.  $W^i Q^i W^i = Q^i$ .

85 The scaled point  $(\bar{x}, \bar{s})$  is defined by the transformation

$$\bar{x} = \Theta W x \quad \text{and} \quad \bar{s} = (\Theta W)^{-1} s,$$

86 where

$$W = \begin{pmatrix} W^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W^k \end{pmatrix}, \quad \Theta = \begin{pmatrix} \theta_1 I_{n_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \theta_k I_{n_k} \end{pmatrix},$$

87  $\theta_i \in \mathbf{R}$ , and  $I_{n_i}$  is the identity matrix  $\in \mathbf{R}^{n_i \times n_i}$ , for all  $i$ . It is easy to see that  $\bar{x}^i = \theta_i W^i x$ ,  
88 and  $\bar{s}^i = (\theta_i W^i)^{-1} s^i$ . Therefore, the linear system for the scaled search direction is defined  
89 by

$$\begin{aligned} A \Delta x - b \Delta \tau &= (\sigma - 1)(Ax - b\tau), \\ A^T \Delta y + \Delta s - c \Delta \tau &= (\sigma - 1)(A^T y + s - c\tau), \\ -c^T \Delta x + b^T \Delta y - \Delta \kappa &= (\sigma - 1)(-c^T x + b^T y - \kappa), \\ \bar{X} T(\Theta W) \Delta x + \bar{X} T(\Theta W)^{-1} \Delta s &= -\bar{X} \bar{S} e + \sigma \mu e, \\ \kappa \Delta \tau + \tau \Delta \kappa &= -\tau \kappa + \sigma \mu. \end{aligned} \tag{1.6}$$

90 The NT direction was proposed by Nesterov and Todd [10]. The paper [10] also shows  
91 that the primal-dual IPM maintains its efficiency when the nonnegative constraints in LP are  
92 replaced by the homogeneous self-dual cone. More recently, Tsuchiya [13] showed that the  
93 long-step algorithm for SOCP using NT direction has  $O(k \log \varepsilon^{-1})$ , iteration-complexity,  
94 where  $k$  is the number of cones, which is the best result among several scaling schemes.

95 The NT direction defines  $\Theta$ , and  $W$ , such that

$$\bar{x} = \bar{s}.$$

96 Hence,  $s = (\Theta W)^2 x$ . The definitions of  $\Theta$  and  $W$  are given in the following lemma. See  
97 [13] for the uniqueness.

98 To simplify notations through out this paper, we define the following symbols.

99 *Definition 1.7.* Define  $\phi = (x, \tau, y, s, \kappa)$ ,  $\chi = (x, \tau)$  and  $\zeta = (s, \kappa)$ .

## 2. Existence of step length for long-step method 100

In this section, we prove the existence of the step length  $\alpha$  using the method described in [18]. See also [11] for an analysis of the semidefinite case. Here is the algorithm which we analyze in this section. 101  
102  
103

**Algorithm 2.1.** *Given  $\gamma, \sigma, (x^{(0)}, \tau^{(0)}, (s^{(0)}, \kappa^{(0)}) \in \text{int}(\bar{K})$ ,  
for  $N = 1, 2, \dots$   
Set  $\phi = \phi^{(N-1)}$ ,  
Solve (1.6) for  $\Delta\phi$   
Find the largest  $\alpha \in [0, 1]$ , such that  $\phi^{(N-1)} + \alpha\Delta\phi \in N(\gamma)$ ,  
Update  $\phi^{(N)}$ ,  
end.* 104  
105  
106  
107  
108  
109  
110

Denote the maximal and minimal eigenvalues of  $X^i$  by  $\lambda_{\max}(x^i)$  and  $\lambda_{\min}(x^i)$ , respectively. From [9], we have that 111  
112

$$\lambda_{\max}(x^i) = x_1^i + \|x_{2:n_i}^i\|, \quad \text{and} \quad \lambda_{\min}(x^i) = x_1^i - \|x_{2:n_i}^i\|.$$

Since  $X$  is the diagonal matrix consisting of  $X^1, \dots, X^k$ , it follows that  $\lambda_{\max}(x)$  and  $\lambda_{\min}(x)$  are obtained from 113  
114

$$\lambda_{\max}(x) = \max \{x_1^i + \|x_{2:n_i}^i\|, i = 1 : k\}, \quad \text{and}$$

$$\lambda_{\min}(x) = \min \{x_1^i - \|x_{2:n_i}^i\|, i = 1 : k\}.$$

We now start to find upper bounds for vectors. We consider Algorithm 2.1, but we use (1.6) with the NT direction instead of (1.5). 115  
116

First, we introduce a useful quantity  $v_N$  defined by 117

$$v_N = \prod_{j=0}^{N-1} (1 - \alpha_j(1 - \sigma_j)), \quad (2.1)$$

where  $\alpha_j$  is the step length and  $\sigma_j$  is the centering parameter in the  $j$ -th iteration. According to Lemma 3.13 in [4] and the definitions of  $r_1, r_2, r_3$  in Algorithm 2.1, we have 118  
119

$$r_i^{(N)} = v_N r_i^{(0)}, \quad i = 1, 2, 3$$

$$\mu_N = v_N \mu_0.$$

We choose the initial point by the following: 120

$$\begin{aligned} \text{if } x^i, s^i \in K^q, \quad x^i &= \delta e^{n_i} = s^i, \\ \text{if } x^i, s^i \in K^r, \quad x^i &= \frac{\delta}{\sqrt{2}} [1, 1, 0, \dots, 0]^T = s^i, \\ \text{if } x^i, s^i \in R_+, \quad x^i &= \delta = s^i, \quad \text{and} \quad \tau = \delta = \kappa, \end{aligned} \quad (2.2)$$

121 where  $\|(x^*, \tau^*, s^*, \kappa^*)\|_\infty \leq \delta$ . The following lemma gives the bound on

$$v \|(x^{(N)}, \tau^{(N)}, s^{(N)}, k^{(N)})\|_\infty.$$

122 **Lemma 2.2.** *Let the initial point be given by (2.2). For all  $N \geq 0$ , there exists  $C_1 > 0$ ,*  
 123 *such that*

$$v \|(x^{(N)}, \tau^{(N)}, s^{(N)}, k^{(N)})\|_\infty \leq C_1 \mu_N.$$

124 **Proof:** For simplicity, we ignore the iteration index ( $N$ ) on the vectors. Let  $(x^*, \tau^*, y^*, s^*,$   
 125  $\kappa^*)$  be any primal-dual solution and define

$$\hat{\phi} = v_N \phi^{(0)} + (1 - v_N) \phi^* - \phi.$$

126 It is easy to verify that  $\hat{\phi}$  satisfies the following three equations:

$$\begin{aligned} A\hat{x} - b\hat{\tau} &= 0, \\ A^T \hat{y} + \hat{s} - c\hat{\tau} &= 0, \\ -c^T \hat{x} + b^T \hat{y} - \hat{\kappa} &= 0. \end{aligned} \quad (2.3)$$

127 Hence,

$$\begin{aligned} \hat{x}^T \hat{s} + \hat{\tau} \hat{\kappa} &= \hat{x}^T (c\hat{\tau} - A^T \hat{y}) + \hat{\tau} (-c^T \hat{x} + b^T \hat{y}) \\ &= 0. \end{aligned} \quad (2.4)$$

128 From (2.4), we have

$$\begin{aligned} 0 &= \hat{x}^T \hat{s} + \hat{\tau} \hat{\kappa} \\ &= v_N^2 \chi^{(0)T} \zeta^{(0)} + \chi^T \zeta + v_N(1 - v_N)(\chi^{(0)T} \zeta^* + \zeta^{(0)T} \chi^*) \\ &\quad - v_N(\chi^{(0)T} \zeta + \zeta^{(0)T} \chi) - (1 - v_N)(\chi^T \zeta^* + \zeta^T \chi^*) + (1 - v_N)^2 \chi^{*T} \zeta^*. \end{aligned} \quad (2.5)$$

129 Since  $\bar{K}$  is self-dual and  $(x^*, \tau^*, y^*, s^*, \kappa^*)$  is a solution, we have that

$$\chi^T \zeta^* + \zeta^T \chi^* \geq 0, \quad \text{and} \quad \chi^{*T} \zeta^* = 0.$$

130 After inserting the above equations into (2.5), we obtain the following inequality,

$$v_N(\chi^{(0)T} \zeta + \zeta^{(0)T} \chi) \leq v_N^2(\chi^{(0)T} \zeta^{(0)}) + \chi^T \zeta + v_N(1 - v_N)(\chi^{(0)T} \zeta^* + \zeta^{(0)T} \chi^*).$$

Next, we define the constant  $\xi$  by

131

$$\xi = \min \left\{ \min_{x^i \in K^q} \{x_1^{i(0)}, s_1^{i(0)}\}, \min_{x^i \in K^r} \{x_{1:2}^{i(0)}, s_{1:2}^{i(0)}\}, \min_{x^i \in \mathbf{R}_+} \{x^{i(0)}, s^{i(0)}\}, \tau^{(0)}, \kappa^{(0)} \right\},$$

where  $x_{1:2}^{i(0)}$  are the first two entries of  $x^{i(0)}$ . By (2.2), we have that  $\xi = \delta$ . One can also derive the following inequality based on the definition of the initial point,

132  
133

$$\begin{aligned} \xi \|\chi, \varsigma\|_\infty &\leq \xi \|x\|_\infty + \xi \|s\|_\infty + \xi \tau + \xi \kappa \\ &\leq \varsigma^{(0)T} \chi + \chi^{(0)T} \varsigma. \end{aligned}$$

Therefore,

134

$$\begin{aligned} \xi v_N \|\chi, \varsigma\|_\infty &\leq v_N^2 \chi^{(0)T} \varsigma^{(0)} + \chi^T \varsigma + v_N(1 - v_N)(\chi^{(0)T} \varsigma^* + \varsigma^{(0)T} \chi^*) \\ &\leq v_N^2(k+1)\mu_0 + (k+1)\mu_N \\ &\quad + v_N(1 - v_N)(\|x^{(0)}\|_\infty \|s^*\|_1 + \|s^{(0)}\|_\infty \|x^*\|_1 + \tau^* \kappa^{(0)} + \kappa^* \tau^{(0)}) \\ &\leq v_N(k+1)\mu_0 + (k+1)\mu_N + v_N \|\chi^{(0)}, \varsigma^{(0)}\|_\infty \|\chi^*, \varsigma^*\|_1 \\ &= 2(k+1)\mu_N + \delta \|\chi^*, \varsigma^*\|_1 \mu_N / \mu_0. \end{aligned}$$

Since  $\mu_0 = \delta^2$ ,

135

$$C_1 = (2(k+1) + \|(x^*, \tau^*, s^*, \kappa^*)\|_1 / \delta) / \delta.$$

Furthermore, because  $\|(x^*, \tau^*, s^*, \kappa^*)\|_1 \leq 2(n+1)\delta$ , it follows that

136

$$C_1 = 2(k+n+2) / \delta.$$

□ 137

Since we specifically discuss the system (1.6) with the NT search direction, we define the following notation for the scaled vectors at the  $N$ -th iteration:

138  
139

$$D_1 = (\Theta W)^{-1}, D_2 = (\kappa^{1/2} \tau^{-1/2})^{-1}, \quad \text{and} \quad D = \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix}.$$

The next lemma gives bounds on the scaled vectors  $\Delta \bar{x} = (D_1)^{-1} \Delta x$  and  $\Delta \bar{s} = D_1 \Delta s$ .

140

**Lemma 2.3.** *Applying the NT scaling scheme to (1.6), i.e., using  $\bar{x} = \bar{s}$ , there is a positive constant  $C_2$ , such that*

141  
142

$$\left\| (D)^{-1} \begin{pmatrix} \Delta x \\ \Delta \tau \end{pmatrix} \right\| \leq C_2 \mu_N^{1/2}, \quad \left\| (D) \begin{pmatrix} \Delta s \\ \Delta \kappa \end{pmatrix} \right\| \leq C_2 \mu_N^{1/2}, \quad \text{for all } N \geq 0.$$

**143 Proof:** Let us define

$$\hat{\phi} = \Delta\phi + \nu_N\phi^{(0)} + \nu_N\phi^*.$$

**144** It is easy to verify that  $\hat{\phi}$  satisfies (2.3) and (2.4). In order to obtain the fourth equation in  
**145** (1.6), we use the following expression

$$\begin{aligned} & \bar{S}T\Theta W\hat{x} + \bar{X}T(\Theta W)^{-1}\hat{s} \\ &= \bar{S}T\Theta W\Delta x + \bar{X}T(\Theta W)^{-1}\Delta s + \nu_N\bar{S}T\Theta W(x^{(0)} - x^*) \\ & \quad + \nu_N\bar{X}T(\Theta W)^{-1}(s^{(0)} - s^*) \\ &= -\bar{X}\bar{S}e + \sigma_N\mu_N e + \nu_N\bar{S}T\Theta W(x^{(0)} - x^*) + \nu_N\bar{X}T(\Theta W)^{-1}(s^{(0)} - s^*). \end{aligned}$$

**146** Since we consider the NT direction, i.e.,  $\bar{S} = \bar{X}$ , we multiply by  $T\bar{S}^{-1}$  from the left and  
**147** obtain

$$D_1^{-1}\hat{x} + D_1\hat{s} = -T\bar{S}^{-1}(\bar{X}\bar{S}e - \sigma_N\mu_N e) + \nu_N D_1^{-1}(x^{(0)} - x^*) + \nu_N D_1(s^{(0)} - s^*).$$

**148** Similarly,

$$D_2^{-1}\hat{\tau} + D_2\hat{\kappa} = -(\kappa\tau)^{-1/2}(\tau\kappa - \sigma_N\mu_N) + \nu_N D_2^{-1}(\tau^{(0)} - \tau^*) + \nu_N D_2(\kappa^{(0)} - \kappa^*).$$

**149** We now consider the 2-norm of the vector,

$$\left\| D^{-1} \begin{pmatrix} \hat{x} \\ \hat{\tau} \end{pmatrix} + D \begin{pmatrix} \hat{s} \\ \hat{\kappa} \end{pmatrix} \right\|^2 = \left\| \begin{pmatrix} D_1^{-1}\hat{x} \\ D_2^{-1}\hat{\tau} \end{pmatrix} \right\|^2 + \left\| \begin{pmatrix} D_1\hat{s} \\ D_2\hat{\kappa} \end{pmatrix} \right\|^2.$$

**150** On the other hand,

$$\begin{aligned} \left\| \begin{pmatrix} D_1^{-1}\hat{x} + D_1\hat{s} \\ D_2^{-1}\hat{\tau} + D_2\hat{\kappa} \end{pmatrix} \right\| &\leq \left\| \begin{pmatrix} T\bar{S}^{-1} & \\ & \kappa^{-1/2}\tau^{-1/2} \end{pmatrix} \begin{pmatrix} \bar{X}\bar{S}e - \sigma_N\mu_N e \\ \tau\kappa - \sigma_N\mu_N \end{pmatrix} \right\| \\ &\quad + \nu_N \left\| \begin{pmatrix} D_1^{-1}(x^{(0)} - x^*) \\ D_2^{-1}(\tau^{(0)} - \tau^*) \end{pmatrix} \right\| + \nu_N \left\| \begin{pmatrix} D_1(s^{(0)} - s^*) \\ D_2(\kappa^{(0)} - \kappa^*) \end{pmatrix} \right\|. \end{aligned}$$

**151** Combining the above two observations, we obtain an upper bound,

$$\begin{aligned} \left\| \begin{pmatrix} D_1^{-1}\Delta x \\ D_2^{-1}\Delta \tau \end{pmatrix} \right\| &= \left\| \begin{pmatrix} D_1^{-1}(\hat{x} - \nu_N(x^{(0)} - x^*)) \\ D_2^{-1}(\hat{\tau} - \nu_N(\tau^{(0)} - \tau^*)) \end{pmatrix} \right\| \\ &\leq \left\| \begin{pmatrix} T\bar{S}^{-1} & \\ & \kappa^{-1/2}\tau^{-1/2} \end{pmatrix} \right\| \left\| \begin{pmatrix} \bar{X}\bar{S}e - \sigma_N\mu_N e \\ \tau\kappa - \sigma_N\mu_N \end{pmatrix} \right\| \\ &\quad + 2\nu_N \left\| \begin{pmatrix} D_1^{-1}(x^{(0)} - x^*) \\ D_2^{-1}(\tau^{(0)} - \tau^*) \end{pmatrix} \right\| + 2\nu_N \left\| \begin{pmatrix} D_1(s^{(0)} - s^*) \\ D_2(\kappa^{(0)} - \kappa^*) \end{pmatrix} \right\| \end{aligned}$$

We now need to show that each term in the right hand side of the above inequality is  $O(\mu_N^{1/2})$ . **152**  
 First, we consider  $\|(\bar{S}^i)^{-1}\|$ . From the property of  $\bar{S}$  and the definition of  $N(\gamma)$ , we derive **153**

$$\begin{aligned} \|(\bar{S}^i)^{-1}\| &= \|(\bar{X}^i)^{-1/2}(\bar{S}^i)^{-1/2}\| \\ &\leq \frac{1}{\sqrt{\bar{x}_1^i - \|\bar{x}_{2:n_i}^i\|}} \frac{1}{\sqrt{\bar{s}_1^i - \|\bar{s}_{2:n_i}^i\|}} \\ &= \frac{1}{\sqrt{\theta_i^2 x^{iT} Q^i x^i}} \frac{1}{\sqrt{\theta_i^{-2} s^{iT} Q^i s^i}} \\ &\leq \frac{1}{\sqrt{\gamma \mu_N}}. \end{aligned}$$

Hence, **154**

$$\begin{aligned} \|T \bar{S}^{-1}\| &= \max_{i=1:k} \|(\bar{S}^i)^{-1}\| \leq \frac{1}{\sqrt{\gamma \mu_N}} \\ (\kappa \tau)^{-1/2} &\leq \frac{1}{\sqrt{\gamma \mu_N}}. \end{aligned}$$

Thus, **155**

$$\begin{aligned} \left\| \begin{pmatrix} \bar{X} \bar{S} e - \sigma_N \mu_N e \\ \tau \kappa - \sigma_N \mu_N \end{pmatrix} \right\|^2 &= \|\bar{X} \bar{S} e\|^2 - 2\sigma_N \mu_N \bar{x}^T \bar{s} + \|\sigma_N \mu_N e\|^2 + \|\tau \kappa\|^2 \\ &\quad - 2\sigma_N \mu_N \tau \kappa + (\sigma_N \mu_N)^2 \\ &\leq \|\bar{X} \bar{S} e\|_1^2 - 2\sigma_N \mu_N \chi^T \zeta + (k+1)(\sigma_N \mu_N)^2 + (\tau \kappa)^2 \\ &\leq (k+1)\mu_N^2. \end{aligned}$$

For the remaining terms in the inequality, we have **156**

$$\begin{aligned} &\nu_N \left\| \begin{pmatrix} D_1^{-1}(x^{(0)} - x^*) \\ D_2^{-1}(\tau^{(0)} - \tau^*) \end{pmatrix} \right\| + \nu_N \left\| \begin{pmatrix} D_1(s^{(0)} - s^*) \\ D_2(\kappa^{(0)} - \kappa^*) \end{pmatrix} \right\| \\ &\leq \nu_N (\|D^{-1}\| + \|D\|) \max \left\{ \left\| \begin{pmatrix} x^{(0)} - x^* \\ \tau^{(0)} - \tau^* \end{pmatrix} \right\|, \left\| \begin{pmatrix} s^{(0)} - s^* \\ \kappa^{(0)} - \kappa^* \end{pmatrix} \right\| \right\}. \end{aligned} \quad (2.6)$$

Since  $\|D^{-1}\| = \max \{\|\Theta W\|, \kappa^{1/2} \tau^{-1/2}\}$ , we find upper bounds for  $\|\Theta W\|$  and  $\kappa^{1/2} \tau^{-1/2}$ . **157**  
 So, **158**

$$\begin{aligned} \|\Theta W\| &\leq \max_{i=1:k} \|\Theta^i W^i \bar{S}^i T^i\| \|T^i (\bar{S}^i)^{-1}\| \\ &\leq \max_{i=1:k} \|T^i (\bar{S}^i)^{-1}\| \sqrt{n_i} \|\Theta^i W^i \bar{S}^i\|_1 \end{aligned}$$

$$\begin{aligned}
&= \max_{i=1:k} \|T^i (\bar{S}^i)^{-1}\| \sqrt{n_i} \|\Theta^i W^i \bar{S}^i e^i\|_1 \\
&= \|T \bar{S}^{-1}\| \left( \sum \sqrt{n_i} \right) \|s\|_1,
\end{aligned}$$

159 since  $s = \Theta W \bar{s}$ . Thus,

$$\begin{aligned}
\|D^{-1}\| &\leq \sum \sqrt{n_i} \|T \bar{S}^{-1}\| \|s\|_1 + \frac{\kappa}{\sqrt{\gamma \mu_N}} \\
&\leq \left( \sum \sqrt{n_i} + 1 \right) \left\| \begin{pmatrix} s \\ \kappa \end{pmatrix} \right\|_1 / \sqrt{\gamma \mu_N}, \quad \text{and} \\
\|D\| &\leq \left( \sum \sqrt{n_i} + 1 \right) \left\| \begin{pmatrix} x \\ \tau \end{pmatrix} \right\|_1 / \sqrt{\gamma \mu_N}.
\end{aligned}$$

160 Now (2.6) is bounded by

$$\frac{2C_1 (\sum \sqrt{n_i} + 1)(n+1)}{\sqrt{\gamma}} \max \left\{ \left\| \begin{pmatrix} x^{(0)} - x^* \\ \tau^{(0)} - \tau^* \end{pmatrix} \right\|, \left\| \begin{pmatrix} s^{(0)} - s^* \\ \kappa^{(0)} - \kappa^* \end{pmatrix} \right\| \right\} \mu_N^{1/2}.$$

161 Hence,

$$\begin{aligned}
C_2 &= \left( k + 1 + 4C_1 (\sum \sqrt{n_i} + 1)(n+1) \right) \max \left\{ \left\| \begin{pmatrix} x^{(0)} - x^* \\ \tau^{(0)} - \tau^* \end{pmatrix} \right\|, \right. \\
&\quad \left. \left\| \begin{pmatrix} s^{(0)} - s^* \\ \kappa^{(0)} - \kappa^* \end{pmatrix} \right\| \right\} / \sqrt{\gamma}.
\end{aligned}$$

162

□

163 If the initial point satisfies (2.2), then the next lemma follows immediately.

164 **Lemma 2.4.** *Under the same conditions as in the previous lemma and if the initial point*

165 *satisfies the (2.2), then*

$$\left\| D^{-1} \begin{pmatrix} \Delta x \\ \Delta \tau \end{pmatrix} \right\| \leq C_3 \mu_N^{1/2}, \quad \text{and} \quad \left\| D \begin{pmatrix} \Delta s \\ \Delta \kappa \end{pmatrix} \right\| \leq C_3 \mu_N^{1/2}.$$

166 **Proof:** From (2.2), we have that

$$-\delta e_1 \leq \begin{pmatrix} x^{(0)} - x^* \\ \tau^{(0)} - \tau^* \end{pmatrix} \leq \delta e_1,$$

where  $e_1 = [1, \dots, 1]^T$ . Then, we can derive the following two estimates,

167

$$\begin{aligned} \left\| D^{-1} \begin{pmatrix} x^{(0)} - x^* \\ \tau^{(0)} - \tau^* \end{pmatrix} \right\| &\leq \delta \sqrt{n+1} \left( \sum \sqrt{n_i} + 1 \right) \left\| \begin{pmatrix} s \\ \kappa \end{pmatrix} \right\|_1 / \sqrt{\gamma \mu_N}, \\ \left\| D \begin{pmatrix} s^{(0)} - s^* \\ \kappa^{(0)} - \kappa^* \end{pmatrix} \right\| &\leq \delta \sqrt{n+1} \left( \sum \sqrt{n_i} + 1 \right) \left\| \begin{pmatrix} x \\ \tau \end{pmatrix} \right\|_1 / \sqrt{\gamma \mu_N}. \end{aligned}$$

It is easy to check that

168

$$C_3 = 4C_1(n+1)^{3/2} \left( \sum \sqrt{n_i} + 1 \right) / \sqrt{\gamma}.$$

□ 169

The next lemma shows the existence of the step length  $\alpha$ .

170

**Lemma 2.5.** *There exists  $\bar{\alpha} \in [0, 1]$  such that  $(\chi(\alpha), \varsigma(\alpha)) \in N(\gamma)$ , for all  $\alpha \in [0, \bar{\alpha}]$ , where  $(\chi(\alpha), \varsigma(\alpha)) \in N(\gamma)$  is defined as  $(\chi^{(N)}, \varsigma^{(N)}) + \alpha(\Delta\chi, \Delta\varsigma)$ .*

171  
172

**Proof:** For simplicity, we consider the quadratic cone case only. By Lemma 2.4, we have

173

$$\Delta x^T \Delta s + \Delta \tau \Delta \kappa \leq \left\| D^{-1} \begin{pmatrix} \Delta x \\ \Delta \tau \end{pmatrix} \right\| \left\| D \begin{pmatrix} \Delta s \\ \Delta \kappa \end{pmatrix} \right\| \leq C_2^2 \mu_N.$$

From the last two equations of the system (1.6), we obtain

174

$$\begin{aligned} \varsigma^T \Delta \chi + \chi^T \Delta \varsigma &= (\Theta W \bar{s})^T \Delta x + ((\Theta W)^{-1} \bar{x})^T \Delta s + \tau \Delta \kappa + \kappa \Delta \tau \\ &= e^T (\bar{S} \Theta W \Delta x + \bar{X} (\Theta W)^{-1} \Delta s) + \tau \Delta \kappa + \kappa \Delta \tau \\ &= e^T (-\bar{X} \bar{S} e + \sigma_N \mu_N e) + \tau \Delta \kappa + \kappa \Delta \tau \\ &= -\bar{x}^T \bar{s} + k \sigma_N \mu_N - \tau \kappa + \sigma_N \mu_N \\ &= (\sigma_N - 1)(k+1) \mu_N. \end{aligned}$$

In order to consider  $N(\gamma)$ , we estimate  $\mu(\alpha)$  by

175

$$\begin{aligned} \mu(\alpha) &= \frac{1}{k+1} \{ (\chi + \alpha \Delta \chi)^T (\varsigma + \alpha \Delta \varsigma) \} \\ &\leq \frac{1}{k+1} \{ (1-\alpha)(k+1) \mu_N + \alpha \sigma_N \mu_N (k+1) + \alpha^2 C_2^2 \mu_N \} \\ &= (1-\alpha) \mu_N + \alpha \sigma_N \mu_N + \frac{\alpha^2 C_2^2}{k+1} \mu_N. \end{aligned}$$

176 We now consider the conditions of  $N(\gamma)$ . We note that

$$\tau(\alpha)\kappa(\alpha) - \gamma\mu(\alpha) \geq \alpha\sigma_N\mu_N(1 - \gamma) - 2\alpha^2C_2^2\mu_N.$$

If

$$\alpha \leq \frac{\sigma_N(1 - \gamma)}{2C_2^2}$$

177 holds, then

$$\tau(\alpha)\kappa(\alpha) \geq \gamma\mu(\alpha).$$

178 Therefore,

$$\bar{\alpha} = \min \left\{ 1, \frac{\sigma_N(1 - \gamma)}{2C_2^2} \right\}. \quad (2.7)$$

179

□

180 A similar result in [9] is based on feasible initial points, however, the condition (2.7) is  
181 good for infeasible initial points.

### 182 3. Application problems

183 We collect the following problems as our test cases and also show that our SOCP algorithm  
184 works in these cases. The first is a production and inventory management problem. Next,  
185 we convert the single facility location problem to our format. Furthermore, we look at the  
186 more complicated class of facility location problems which plan to insert more than one  
187 new facility. In the fourth section, we discuss portfolio optimization, followed by FIR Filter  
188 design. In the sixth section, we describe the equilibrium of a system of piecewise-linear  
189 springs. The last two problems are robust least squares and finding feasible initial points.

#### 190 3.1. Management problem

191 In this section, we will look at the following model [2]:

$$\begin{aligned} \min \quad & \sum_{i=1}^n c_i + d_i x_i + \frac{e_i}{x_i} \\ \text{s.t.} \quad & \sum_{i=1}^n b_i x_i \leq b, \\ & l_i \leq x_i \leq u_i, \quad i = 1 : n, \\ & x_i : \text{integer}, \quad i = 1 : n. \end{aligned} \quad (3.1)$$

This model can be interpreted as a multi-item production and inventory management problem with a limited resource, where the  $x_i$  are called decision variables [17, 20]. In general, the objective function minimizes the cost, which could be the sum of setup (ordering) costs, inventory holding costs, and purchase costs. The constraints provide the restrictions of a shared resource as well as non-shared resources. Moreover, the model represents numerous application problems based on the interpretations of the decision variables. There are at least four different applications using the model (3.1).

Let us consider the continuous relaxation of (3.1). In order to solve it as a second-order cone problem, we must ensure that (3.1) can be cast as SOCP. Let  $t = \max_j \frac{1}{x_j}$ . Then  $tx_j \geq 1$ , for all  $j$ . We introduce a  $s$  in this inequality and rewrite it as

$$tx_j \geq s^2, \quad \text{for all } j$$

with  $s = 1$ .

By using the following fact [5]:

$$\omega^2 \leq xy, \quad x \geq 0, \quad y \geq 0 \Leftrightarrow \left\| \begin{pmatrix} 2\omega \\ x - y \end{pmatrix} \right\| \leq x + y, \quad (3.2)$$

the previous inequality becomes

$$\left\| \begin{pmatrix} 2 \\ t - x_j \end{pmatrix} \right\| \leq t + x_j, \quad \text{for all } j.$$

Therefore, we can rewrite the inequalities as

$$2^2 + (t - x_j)^2 \leq (t + x_j)^2, \quad \text{for all } j.$$

Next, define  $e, u, v$ , and  $w$  as follows,

$$e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbf{R}^n, \quad u = te + x, \quad v = te - x, \quad \text{and} \quad w = 2e.$$

It is then clear that  $x = \frac{u-v}{2}$  and  $te = \frac{u+v}{2}$ . Plugging this into our model, we obtain

$$\begin{aligned} \min \quad & \sum_{i=1}^n c_i + d_i \left( \frac{u-v}{2} \right)_i + e_i \left( \frac{u+v}{2} \right)_i \\ \text{s.t.} \quad & \sum_{i=1}^n b_i \left( \frac{u-v}{2} \right)_i \leq b, \end{aligned} \quad (3.3)$$

$$l_i \leq \left( \frac{u-v}{2} \right)_i \leq u_i, \quad i = 1 : n,$$

$$u_i^2 \geq v_i^2 + w_i^2, \quad i = 1 : n.$$

**207** In this form, the objective function and constraints, except the last  $n$  inequalities, are linear.

**208** Therefore, we can rewrite them as LP. Each of the last  $n$  constraints is a quadratic cone with

**209**  $u_i$ ,  $v_i$ , and  $w_i$ . We use the random problems described in [2] to generate test cases.

**210** *Example 1.* Given  $50 \leq n \leq 200$ ,  $S_i \in [5, 20]$ ,  $H_i \in [1, 4]$ ,  $D_i \in [10, 100]$ ,  $T_i \in [1, 15]$ ,

**211**  $l_i, u_i \in [8, 25]$ ,  $i = 1, \dots, n$ . The total machine time available  $T_i$  is generated to ensure the

**212** feasibility of problems, i.e.,

$$T_i = \sum_{i=1}^n T_i \bar{x}_i + \text{random positive number},$$

**213** where  $l_i \leq \bar{x}_i \leq u_i$ , for all  $i$ . In Chapter 5, we use Example 1 with  $n = 50, 75, 100, 200$  as

**214** part of our testing set.

**215** 3.2. *Facility location problem (I)*

**216** This problem is given as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^m w_i t_i \\ \text{s.t.} \quad & \sqrt{\sum_{j=1}^d (x_j - a_{ij})^2} \leq t_i, \quad \text{for } i = 1 : m, \end{aligned} \tag{3.4}$$

**217** where  $d$  is the dimension,  $m$  is the number of existing facilities,  $(a_{i1}, a_{i2})$  are the coordinates

**218** of the existing facility (if  $d = 2$ ), and  $w_i$  is the weight associated with the old-new facility.

**219** The model describes that one plans to build a new facility among existing facilities and

**220** chooses the location which minimizes the weights associated with the Euclidean distance

**221** between the new and existing locations. For simplicity, we consider  $d = 2$ . It is trivial to

**222** extend to higher dimensions. Let

$$u_{ij} = x_j - a_{ij}, \quad \text{for } i = 1 : m, \quad \text{and } j = 1 : 2.$$

**223** We then introduce new linear constraints

$$u_{11} + a_{11} = u_{i1} + a_{i1}, \quad u_{12} + a_{12} = u_{i2} + a_{i2}, \quad \text{for } i = 2 : m.$$

270

KUO AND MITTELMANN

Therefore, the problem becomes

224

$$\begin{aligned}
\min \quad & \sum_{i=1}^m w_i t_i \\
\text{s.t.} \quad & u_{11} - u_{i1} = a_{i1} - a_{11}, i = 2 : m, \\
& u_{12} - u_{i2} = a_{i2} - a_{12}, i = 2 : m, \\
& \sum_{j=1}^2 u_{ij}^2 \leq t_i^2, t_i \geq 0, i = 1 : m.
\end{aligned} \tag{3.5}$$

We denote

225

$$x^i = \begin{pmatrix} t_i \\ u_{i1} \\ u_{i2} \end{pmatrix}, \quad c^i = \begin{pmatrix} w_i \\ 0 \\ 0 \end{pmatrix}, \quad b = \begin{pmatrix} a_{21} - a_{11} \\ \vdots \\ a_{m1} - a_{11} \\ a_{22} - a_{12} \\ \vdots \\ a_{m2} - a_{22} \end{pmatrix}$$

and

226

$$x = \begin{pmatrix} x^1 \\ \vdots \\ x^m \end{pmatrix}, \quad c = \begin{pmatrix} c^1 \\ \vdots \\ c^m \end{pmatrix}.$$

Hence, one obtains the SOCP formulation.

227

*Example 2.* The coordinates of the existing facilities and the weight associated with each existing facility are both generated by a uniform random number generator. We use  $m = 50, 100, 150, 200$  to create four problems.

228

229

230

### 3.3. Facility location problem (II)

231

The second facility location problem we present has the following model:

232

$$\min \quad \sum_{i=1}^m \sum_{j=1}^n w_{ij} \sqrt{\sum_{k=1}^2 (x_{jk} - a_{ik})^2} + \sum_{j=1}^n \sum_{jj=1}^j v_{j,jj} \sqrt{\sum_{k=1}^2 (x_{j,k} - x_{jj,k})^2}, \tag{3.6}$$

233 where the definitions for  $m$ ,  $a$ , and  $w$  are the same as in the previous case,  $n$  is the number  
 234 of new facilities, and  $v_{j,jj}$  are the weights of the new-new connections. In a AMPL model,  
 235 emfl.mod in [16],  $v$  is defined as constant and  $w$  is decided by the initial guess. The way  
 236 of defining  $v$  and  $w$  does not seem realistic; however, they are still reasonable if one has  
 237 a good a initial guess. We now convert this problem to SOCP format. First, we introduce  
 238  $s_{ij} \geq 0, i = 1 : m, j = 1 : n$ , and  $t_{j,jj} \geq 0, j = 1 : n, jj < j$ . Similarly, we need some  
 239 additional variables to replace  $x_{jk} - a_{jk}$ , and  $x_{j,k} - x_{jj,k}$ . By rewriting  $w$  and  $v$  appropriately,  
 240 we obtain the SOCP formulation.

241 *Example 3.* To consider a simplified problem, we take an AMPL model, emfl.mod, from  
 242 [16]. In this model, the coordinates of the existing facilities are generated randomly, the  
 243 weight associated with the new-new connections is fixed at .2, and the weight associated  
 244 with the old-new connection is fixed by the distance between the old facilities and the initial  
 245 guesses. We use the following four cases, (i)  $m = 10, n = 4$ , (ii)  $m = 20, n = 4$ , (iii)  
 246  $m = 30, n = 9$ , and (iv)  $m = 40, n = 9$ .

#### 247 3.4. Portfolio optimization

248 Consider a portfolio problem with  $n$  investments held over a period of time. Let  $x_j$  denote  
 249 the amount of investment  $j$  throughout the period with a unit total budget, i.e.,

$$\sum_{j=1}^n x_j = 1,$$

250 and let  $p_j$  denote the price change of investment  $j$  over the period. Then, the overall expected  
 251 return is given by  $r = p^T x$ . Furthermore, let  $\sigma_{jj'}$  denote the covariance between the returns  
 252 for investment  $j$  and  $j'$ . The variance of the return is

$$\sigma = \sum_{jj'} x_j \sigma_{jj'} x_{j'},$$

253 referred to the risk. Here we consider that the  $\sigma_{jj'}$  are fixed and computed from historical  
 254 data. A similar application can also be found in [15].

255 One approach is to minimize the risk over the return with a lower bound  $r_{\min}$ ,

$$\begin{aligned} \min \quad & \sum_{jj'} x_j \sigma_{jj'} x_{j'} \\ \text{s.t.} \quad & \sum_j p_j x_j \geq r_{\min}, \\ & \sum_j x_j = 1, \quad x_j \geq 0, \quad j = 1 : n. \end{aligned} \tag{3.7}$$

Another approach is to maximize the return over the risk with an upper bound  $\sigma_{\max}$ , **256**

$$\begin{aligned} \max \quad & \sum_j p_j x_j \\ \text{s.t.} \quad & \sum_{jj'} x_j \sigma_{jj'} x_{j'} \leq \sigma_{\max}, \\ & \sum_j x_j = 1, \quad x_j \geq 0, \quad j = 1 : n. \end{aligned} \quad (3.8)$$

The above models are based on historical data. **257**

*Example 4.* Two test cases from (3.7) and (3.8) are cast as in formula (1.1) with the real **258**  
data from `optrisk.mod` in [16]. Another case which combines (3.7) and (3.8), **259**

$$\begin{aligned} \min \quad & \sum_{jj'} x_j \sigma_{jj'} x_{j'} - \sum_j p_j x_j \\ \text{s.t.} \quad & \sum_j x_j = 1, \quad x_j \geq 0, \quad j = 1 : n, \end{aligned}$$

is also considered. **260**

### 3.5. FIR filter design **261**

Let  $h_0, \dots, h_{n-1} \in \mathbf{R}$  be the coefficients of a finite impulse response (FIR) filter of length **262**  
 $n$ . The filter output signal  $y$  is defined by the input signal  $u$ , by **263**

$$y(k) = \sum_{j=0}^{n-1} h_j u(k-j).$$

The function of the frequency response is  $H : [0, 2\pi] \rightarrow \mathbf{C}$  defined by **264**

$$H(\omega) = \sum_{k=0}^{n-1} h_k e^{-ik\omega},$$

where  $i = \sqrt{-1}$  and  $\omega$  is the frequency parameter. There are several different FIR filter **265**  
designs, see, for example [5]. **266**

In this section, we consider the minimax dB linear phase lowpass FIR filter design, **267**

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & \frac{1}{t} \leq 2 \sum_{k=0}^{n/2-1} h_k \cos((k - (n-1)/2)\omega) \leq t, \quad 0 \leq \omega \leq \omega_p, \\ & -\beta \leq 2 \sum_{k=0}^{n/2-1} h_k \cos((k - (n-1)/2)\omega) \leq \beta, \quad \omega_s \leq \omega \leq \pi, \\ & t \geq 1, \end{aligned}$$

268 where  $\beta \geq 0$  and  $0 < \omega_p < \omega_s < \pi$ . For simplicity we assume  $n$  is even.

269 Note that this problem has infinite constraints. Subsequently, we discretize the frequency  
 270 parameter  $\omega$  to obtain a finite-constraint approximation. Moreover, there is a nonlinear term  
 271  $\frac{1}{t}$  in the first part of the constraints. To convert it to a second order cone constraint, we  
 272 introduce  $u$  such that

$$\frac{1}{t} \leq u.$$

273 We take the following approach,

$$1 \leq ut \Leftrightarrow v^2 \leq 2ut, \quad \text{for } u \geq 0, t \geq 1, \quad \text{and } v = \sqrt{2}.$$

274 Now, we obtain the problem in SOCP form,

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & u \leq 2 \sum_{k=0}^{n/2-1} h_k \cos((k - (n-1)/2)\omega_j) \leq t, \quad j = 0 : N_1 - 1, \\ & -\beta \leq 2 \sum_{k=0}^{n/2-1} h_k \cos((k - (n-1)/2)\omega_j) \leq \beta, \quad j = N_1 : N, \\ & t \geq 1, v = \sqrt{2}, \\ & v^2 \leq 2ut, u \geq 0, \end{aligned} \quad (3.9)$$

275 where  $0 = \omega_0 < \dots < \omega_{N_1-1} = \omega_p$ ,  $\omega_s = \omega_{N_1} < \dots < \omega_N = \omega$ . We use the following  
 276 data in our test cases.

277 *Example 5.* Given  $\beta = 0.01$ ,  $\omega_p = \pi/2$ , and  $\omega_s = 2\pi/3$ . We discretize the frequency  
 278 parameter  $\omega$  by the uniform step  $\frac{\pi}{180}$ . We choose  $n = 10, 20, 40, 80$  as our test samples.

279 More details about this problem and the interior point approach can be found in [19].

### 280 3.6. Equilibrium of system of piecewise linear springs

281 This problem describes a mechanical system to find a hanging chain consisting of  $N$  links.

282 The first and last nodes are fixed. Each node has a weight  $m_j$  hanging from it. The problem

283 is to minimize the energy of the system (see [5, 15]):

$$\begin{aligned} \min \quad & \sum_j m_j g y_j + \frac{k}{2} \|t\|^2 \\ \text{s.t.} \quad & \|(x_j, y_j) - (x_{j-1}, y_{j-1})\| - l_0 \leq t_j, \quad j = 1 : N, \\ & (x_0, y_0) = (a_1, a_2), \\ & (x_N, y_N) = (b_1, b_2), \\ & t \geq 0, \end{aligned} \quad (3.10)$$

where  $g$  is the acceleration due to gravity,  $k$  is the stiffness constant of the springs,  $l_0$  is the rest length of each spring, and  $t_j$  is an upper bound on the spring energy of the  $j$ -th spring. Consider  $\|t\|^2 \leq 2uv$ ,  $u = 1$ ,  $v \geq 0$ ,  $s_j = t_j + l_0$ ,  $e_j = x_j - x_{j-1}$ ,  $f_j = y_j - y_{j-1}$ ,  $j = 1 : N$ . Then  $\sum_j f_j = b_2 - a_2$  and  $\sum_j e_j = b_1 - a_1$ . The objective function becomes  $\sum_j m_j g y_j + kv$ . Furthermore,

$$\begin{aligned} \sum_j m_j g y_j + kv &= g \sum_j m_j \left( \sum_{i=1}^j f_i + a_2 \right) + kv \\ &= g \sum_{i=1}^N \left( \sum_{j=i}^N m_j \right) f_i + g \sum_{j=1}^N a_2 + kv. \end{aligned}$$

The linear and cone constraints are

$$\begin{aligned} \sum f_j &= b_2 - a_2, \\ \sum e_j &= b_1 - a_1, \\ s_j - t_j &= l_0, \quad j = 1 : N, \\ u &= 1, \\ e_j^2 + f_j^2 &\leq s_j^2, \quad j = 1 : N, \\ \|t\|^2 &\leq 2uv, \quad u, v \geq 0. \end{aligned}$$

*Example 6.* The data is the same as in springs.mod in [16]. We have  $k = 100$ ,  $l_0 =$  284  
 $2\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}/N$ ,  $g = 9.8$ ,  $m = 1$ ,  $(a_1, a_2) = (0, 0)$ , and  $(b_1, b_2) = (2, -1)$ . 285  
 We use  $N = 10, 20, 40, 60$  as test cases. 286

#### 4. Numerical realization and results 287

This chapter presents numerical results for quadratic programming and second-order cone 288  
 programming. We implemented algorithms in the MATLAB environment. First, we discuss 289  
 some computational issues arising in second-order cone programming and different variants 290  
 of the algorithms. We use examples described in the previous chapter as test sets and compare 291  
 their results. Detailed information on the test set is in [4]. 292

##### 4.1. Linear system 293

Here, we will discuss some computational issues in second order cone programming, es- 294  
 pecially the linear system (1.6). Denote by  $[r_1, r_2, r_3, r_4, r_5]^T$  the right hand side of (1.6). 295

The augmented system following from (1.6) is

$$\begin{pmatrix} -(\bar{X}T(\Theta W)^{-1})^{-1}\bar{S}T\Theta W - \frac{\tau}{\kappa}cc^T & A^T + \frac{\tau}{\kappa}cb^T \\ A - \frac{\tau}{\kappa}bc^T & \frac{\tau}{\kappa}bb^T \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ = \begin{pmatrix} r_2 - \Theta WT\bar{X}^{-1}r_4 + mc \\ r_1 + mb \end{pmatrix},$$

where  $m = \frac{\tau}{\kappa}(r_3 + r_5/\tau)$ . We use the NT direction, i.e.,  $\bar{X} = \bar{S}$ , apply the Sherman-Morrison-Woodbury formula to the coefficient matrix, and denote

$$p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} -(\Theta W)^2 & A^T \\ A & 0 \end{pmatrix}^{-1} \begin{pmatrix} c \\ b \end{pmatrix}, \quad \text{and} \\ q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} -(\Theta W)^2 & A^T \\ A & 0 \end{pmatrix}^{-1} \begin{pmatrix} r_2 - \Theta WT\bar{X}^{-1}r_4 \\ r_1 \end{pmatrix}. \quad (4.1)$$

Hence,

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} + h \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}, \quad \text{where } h = \frac{m - \frac{\tau}{\kappa}(-c^T \quad b^T)q}{1 + \frac{\tau}{\kappa}(-c^T \quad b^T)p}. \quad (4.2)$$

296 The next lemma follows immediately.

297 **Lemma 4.1.**  $h = \Delta\tau$ .

**Proof:**

$$\begin{aligned} \Delta\tau - h &= \frac{\tau}{\kappa}(r_3 + r_5/\tau + c^T \Delta x - b^T \Delta y) - h \\ &= \frac{\tau}{\kappa}(r_3 + r_5/\tau + c^T(q_1 + hp_1) - b^T(q_2 + hp_2)) - h \\ &= m - \frac{\tau}{\kappa}(-c^T \quad b^T) - h \left(1 + \frac{\tau}{\kappa}(-c^T \quad b^T)p\right) \\ &= 0. \end{aligned}$$

298

□

Moreover, the normal equations are

$$\begin{aligned} A(\Theta W)^{-2}A^T p_2 &= b + A(\Theta W)^{-2}c, \\ A(\Theta W)^{-2}A^T q_2 &= r_1 + A(\Theta W)^{-2}(r_2 - \Theta WT\bar{X}^{-1}r_4), \\ p_1 &= -(\Theta W)^{-2}(c - A^T p_2), \\ q_1 &= -(\Theta W)^{-2}(r_2 - \Theta WT\bar{X}^{-1}r_4 - A^T q_2). \end{aligned}$$

Subsequently, the search direction can be computed as follows:

$$\begin{aligned}\Delta\tau &= h, \\ \Delta x &= q_1 + \Delta\tau p_1, \\ \Delta y &= q_2 + \Delta p_2, \\ \Delta s &= -(\Theta W)^2 \Delta x + \Theta W T \bar{X}^{-1} r_4, \\ \Delta\kappa &= -\frac{\kappa}{\tau} \Delta\tau + r_5/\tau.\end{aligned}$$

Furthermore, one can rewrite  $\Delta s$  as

$$\Delta s = r_2 - A^T q_2 + \Delta\tau(c - A^T p_2).$$

Two vectors  $r_2 - A^T q_2$  and  $c - A^T p_2$  have been calculated while computing  $p_1$  and  $q_1$ . Because of the definition of  $W^i$ ,  $(W^i)^2$  and  $(W^i)^{-2}$  are easy to obtain:

$$\begin{aligned}(W^i)^{-2} &= -Q^i - 2Q^i w^i (1 - 2(w^i)^T Q^i w^i)^{-1} (w^i)^T Q^i \\ &= Q^i (-Q^i + 2w^i (w^i)^T) Q^i \\ &= -Q^i + 2 \begin{pmatrix} -w_1^i \\ w_{2:n_i}^i \end{pmatrix} \begin{pmatrix} -w_1^i & (w_{2:n_i}^i)^T \end{pmatrix},\end{aligned}$$

since  $1 - 2(w^i)^T Q^i w^i = -1$ .

299

#### 4.2. Long-step path-following method

300

In this section, we start by stating the long-step path-following algorithm with the scaled search direction and then compare the heuristic choices of the centering parameter  $\sigma$  with the proposed choices. First, we give the linear system for the scaled search direction,

$$\begin{pmatrix} A & -b & 0 & 0 & 0 \\ 0 & -c & A^T & I & 0 \\ -c^T & 0 & b^T & 0 & -1 \\ \bar{S}T\Theta W & 0 & 0 & \bar{X}T(\Theta W)^{-1} & 0 \\ 0 & \kappa & 0 & 0 & \tau \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\tau \\ \Delta y \\ \Delta s \\ \Delta\kappa \end{pmatrix} = \begin{pmatrix} (\sigma - 1)r_1 \\ (\sigma - 1)r_2 \\ (\sigma - 1)r_3 \\ -\bar{X}\bar{S}e + \sigma\mu e \\ -\tau\kappa + \sigma\mu \end{pmatrix}. \quad (4.3)$$

In algorithms, we denote  $\phi = (x, \tau, y, s, \kappa)$  and all other 5-tuple vectors are also converted to  $\Delta\phi$ ,  $\phi^{(N)}$ , and etc.

301

**Algorithm 4.2.** Given  $(x^{(0)}, \tau^{(0)}), (s^{(0)}, \kappa^{(0)}) \in \bar{K}$ ,  $\sigma_{\min} < \sigma_0 < \sigma_{\max}$ ,  $\gamma$   
 $\mu_0 = ((x^{(0)})^T s^{(0)} + \tau^{(0)} \kappa^{(0)}) / (k + 1)$ ,

For  $N = 0, 1, \dots$   
 Set  $\phi = \phi^{(N)}$ .  
 Solve (4.3) for  $\Delta\phi$ ,  
 where  $r_1 = Ax - b\tau$ ,  
 $r_2 = A^T y + s - c\tau$ ,  
 $r_3 = -c^T x + b^T y - \kappa$ .  
 Find  $\alpha_{\max} = \arg \max\{\alpha \in [0, 1] | \phi + \alpha\Delta\phi \in N(\gamma)\}$ .  
 Update  $\phi^{(N+1)} = \phi + \alpha_{\max}\Delta\phi$ .  
 Update  $\mu_{N+1}, \sigma_{N+1}, \gamma_{N+1}$ .  
 end.

302 In the basic algorithm, the parameter  $\gamma_{N+1}$  is fixed for all iterations. Here, we consider  
 303 that  $\gamma$  can be changed at each iteration and propose a way to choose  $\sigma_{N+1}$  and  $\gamma_{N+1}$ . We also  
 304 compare this choice with the heuristic choices of  $\sigma_{N+1}$  and fixed  $\gamma$ . Since the comparisons  
 305 are based on the same algorithm, we only compare the number of iterations.

Let us consider three heuristic choices with fixed  $\gamma = 0.001$ :

$$\text{H2 } \sigma = \left( \frac{\mu_{N+1}}{\mu_N} \right)^2,$$

$$\text{H3 } \sigma = \left( \frac{\mu_{N+1}}{\mu_N} \right)^3,$$

$$\text{H4 } \sigma = \left( \frac{\mu_{N+1}}{\mu_N} \right)^4.$$

306 We now present the algorithm for choosing  $\mu$  and  $\sigma$ .

**Algorithm 4.3.**  $\sigma = (\frac{\mu_N}{\mu_{N-1}})^2; \sigma_{\max} = 0.9, \sigma = 0.005$

if  $\sigma \geq \sigma_{\max}$   
 $\gamma = 0.0001$ ;  
 elseif  $\sigma < 10 * \sigma_{\min}$   
 if  $\mu_N < 10^{-4}$   
 $\sigma = (\frac{\mu_N}{\mu_{N-1}})^3$ ;  
 $\gamma = 0.01$ ;  
 else  
 $\sigma = (\frac{\mu_N}{\mu_{N-1}})^4$ ;  
 end  
 else  
 $\gamma = 0.001$ ;  
 end

In the literature, authors seldom show exactly how  $\gamma$  and  $\sigma$  are chosen. Although these heuristic choices of  $\sigma$  are mentioned frequently in the literature, practical algorithms must develop more robust methods to choose  $\sigma$ . As one can see in Table 1, the performances of H2, H3, and H4 vary in each problem. The main idea of Algorithm 4.3 is to use three heuristic choices and to change the size of the neighborhood according to the result of H2. If  $\sigma$  is greater than  $\sigma_{\max}$ , it follows that the descent of  $\mu_{N+1}$  is small. One way to achieve larger descent is to have a larger neighborhood, i.e., choosing a smaller  $\gamma$ . If  $\sigma$  is less than  $10 \times \sigma_{\min}$ , it means that a large descent has been made. When  $\mu_N$  is getting closer to the tolerance, it is important to pull the new point back to the central path, i.e., choosing larger  $\gamma$ . If  $\mu_N$  is still large, we use H4 to loosen the centering parameter. The reason is that the small value of H2 indicates that the current neighborhood is large enough to allow for a large descent. The following table shows the number of iterations for reaching the optimum using Algorithm 4.2 combined with the above four methods to choose  $\sigma$  and  $\gamma$ . According to this table, we see that the number of iterations using Algorithm 4.3 is overall less than using the heuristic choices. The average is about 6 iterations less than H2, and H3, 7 iterations less

Table 1. The number of iterations for different methods of choosing  $\sigma$ .

	Algorithm 4.3	H2	H3	H4
EX1-50	20	25	23	24
EX1-75	21	27	25	25
EX1-100	22	28	25	26
EX1-200	23	28	28	28
EX2-50	13	18	18	21
EX2-100	13	19	18	21
EX2-150	14	19	20	21
EX2-200	14	18	18	18
EX3-10-4	18	24	25	25
EX3-20-4	16	21	22	23
EX3-30-9	18	23	24	25
EX3-40-9	18	24	23	26
EX4-mark	18	27	25	27
EX4-return	23	30	30	29
EX4-risk	21	27	26	29
EX5-10	27	30	33	31
EX5-20	26	32	31	33
EX5-40	30	35	37	36
EX5-80	31	36	35	38
EX6-10	26	28	34	33
EX6-20	35	36	42	43
EX6-40	50	57	57	58
EX6-60	62	66	71	79
Total	559	678	690	721

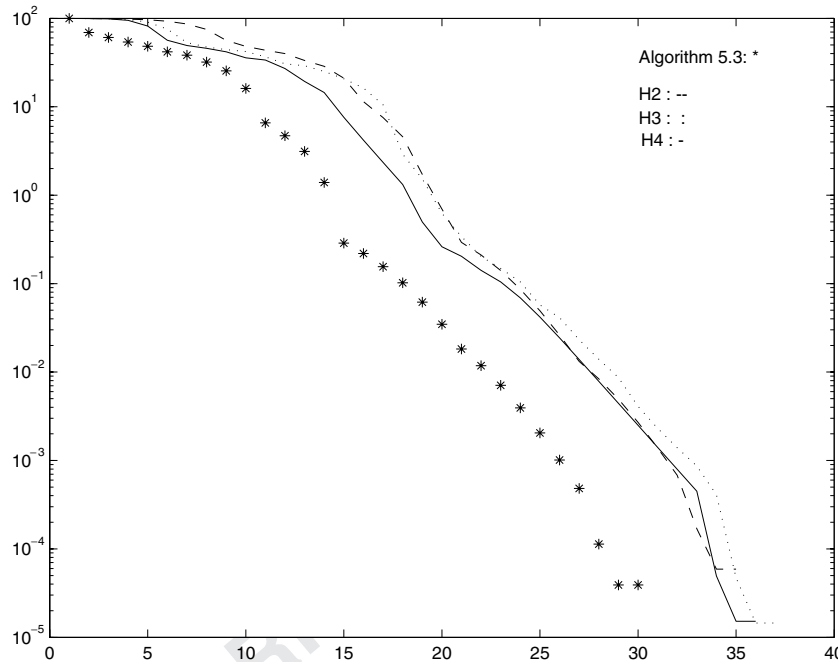


Figure 1. Comparison of  $\mu$  from Algorithm 4.3, H2, H3, H4.

322 than H4. Since the problem is large, the running time for 6 or 7 iterations is still significant.  
 323 Figure 1 from EX6-40 shows that  $\mu_N$  decreases much more in Algorithm 4.3 than in the  
 324 other three methods. Similar graphs can be obtained for each test case.

325 4.3. Predictor-corrector algorithms

326 The predictor-corrector method for linear programming was proposed by Mehrotra [6] based  
 327 on a second-order correction to the pure Newton direction. This method works quite well for  
 328 LP and QP in practice, although its theoretical result in [18] has the same complexity as the  
 329 short-step method. In this section we start by describing the idea of the predictor-corrector  
 330 method and then derive its linear system for SOCP and give two variants of the algorithm.

Given a point  $(x, \tau, y, s, \kappa)$  with  $(x, \tau), (s, \kappa) \in \bar{K}$ . The predictor direction is found by solving the following system:

$$\begin{pmatrix} A & -b & 0 & 0 & 0 \\ 0 & -c & A^T & I & 0 \\ -c^T & 0 & b^T & 0 & -1 \\ \bar{S}T\Theta W & 0 & 0 & \bar{X}T(\Theta W)^{-1} & 0 \\ 0 & \kappa & 0 & 0 & \tau \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta \tau_p \\ \Delta y_p \\ \Delta s_p \\ \Delta \kappa_p \end{pmatrix} = \begin{pmatrix} -r_1 \\ -r_2 \\ -r_3 \\ -\bar{X}\bar{S}e \\ -\tau\kappa \end{pmatrix}, \quad (4.4)$$

where  $r_1, r_2, r_3$  are defined as in (4.3). Next, we find the step length  $\alpha_p$  along this direction:

$$\alpha_p = \arg \max\{\alpha \in [0, 1] \mid (x, \tau) + \alpha(\Delta x_p, \Delta \tau_p), (s, \kappa) + \alpha(\Delta s_p, \Delta \kappa_p) \in \bar{K}\}, \quad (4.5)$$

and define

$$\mu_p = [(x + \alpha_p \Delta x_p)^T (s + \alpha_p \Delta s_p) + (\tau + \alpha_p \Delta \tau_p)(\kappa + \alpha_p \Delta \kappa_p)] / (k + 1). \quad (4.6)$$

Note that the reduction obtained using the predictor is larger than that for the long-step path-following method since

$$\begin{aligned} (x + \Delta x_p)^T (s + \Delta s_p) &= x^T s + x^T \Delta x_p + s^T \Delta x_p + \Delta x_p^T \Delta s_p \\ &= \Delta x_p^T \Delta s_p \\ &< \Delta x_p^T \Delta s_p + \sigma \mu e. \end{aligned}$$

We now consider the corrector direction and the centralizing step at once. Denote a corrector-centering step by  $(\Delta x_c, \Delta \tau_c, \Delta y_c, \Delta s_c, \Delta \kappa_c)$ . Assuming that the full step of predictor and corrector-centering directions are taken, we expect

$$(x^i + \Delta x_p^i + \Delta x_c^i)^T (s^i + \Delta s_p^i + \Delta s_c^i) = \sigma \mu, \quad i = 1 : k. \quad (4.7)$$

After expanding the left hand side of (4.7) and ignoring the higher order terms  $(\Delta x_p^i)^T \Delta s_c^i$ ,  $(\Delta s_p^i)^T \Delta x_c^i$ , and  $(\Delta x_c^i)^T \Delta s_c^i$ , one has

$$(s^i)^T \Delta x_c^i + (x^i)^T \Delta s_c^i \approx \sigma \mu - (\Delta x_p^i)^T \Delta s_p^i.$$

Then, the corrector-centering direction is obtained by solving

$$\begin{pmatrix} A & -b & 0 & 0 & 0 \\ 0 & -c & A^T & I & 0 \\ -c^T & 0 & b^T & 0 & -1 \\ \bar{S}^T \Theta W & 0 & 0 & \bar{X}^T (\Theta W)^{-1} & 0 \\ 0 & \kappa & 0 & 0 & \tau \end{pmatrix} \begin{pmatrix} \Delta x_c \\ \Delta \tau_c \\ \Delta y_c \\ \Delta s_c \\ \Delta \kappa_c \end{pmatrix} = \begin{pmatrix} \sigma r_1 \\ \sigma r_2 \\ \sigma r_3 \\ \sigma \mu_p e - \Delta \bar{X}_p \Delta \bar{S}_p e \\ \sigma \mu_p - \Delta \tau_p \Delta \kappa_p \end{pmatrix}. \quad (4.8)$$

- 331** We describe two algorithms using the predictor direction and the corrector-centering  
**332** direction mentioned above. All initial data are chosen as in Algorithm 4.2.

**Algorithm 4.4.** *Initial Data as in Algorithm 4.2*

Let  $\chi = (x, \tau)$  and  $\zeta = (s, \kappa)$   
 For  $N = 0, 1, \dots$   
 Set  $\phi = \phi^{(N)}$   
 Solve (4.4) for  $\Delta\phi_p$   
 Calculate  $\mu_p$  from (4.6) and  $\sigma$   
 Solve (4.8) for  $\Delta\phi_c$   
 Set  $\Delta\phi = \Delta\phi_p + \Delta\phi_c$   
 Find  $\alpha_{\max} = \arg \max\{\alpha \in [0, 1] \mid \chi + \alpha\Delta\chi, \zeta + \alpha\Delta\zeta \in \bar{K}\}$   
 Update  $\phi^{(N+1)} = \phi + \alpha_{\max}\Delta\phi$   
 end.

Table 2. Comparison of our methods with the codes [1, 12, 14, 15].

	A. 4.4	A. 4.5	A. 4.2	SeDu	SDPT	MOSK	LOQO
EX1-50	13	15	20	13	12	13	44
EX1-75	16	14	21	16	11	13	44
EX1-100	14	16	22	16	12	14	51
EX1-200	17	17	23	16	12	16	60
EX2-50	9	10	13	15	9	8	13
EX2-100	8	13	13	12	10	8	14
EX2-150	9	9	14	15	10	10	12
EX2-200	10	9	14	14	10	10	13
EX3-10-4	13	14	18	13	21	10	88
EX3-20-4	11	13	16	14	16	10	f
EX3-30-9	11	14	18	14	15	10	f
EX3-40-9	13	14	18	14	12	10	f
EX4-mark	12	14	18	10	na	9	na
EX4-return	15	14	23	12	na	9	na
EX4-risk	14	16	21	12	na	9	na
EX5-10	17	21	27	6	na	6	na
EX5-20	17	25	26	7	na	6	na
EX5-40	18	22	30	7	na	5	na
EX5-80	23	24	31	8	na	5	na
EX6-10	11	16	26	13	na	10	na
EX6-20	11	15	35	15	na	10	na
EX6-40	11	26	50	13	na	12	na
EX6-60	14	26	62	14	na	11	na

**Algorithm 4.5.** *Initial Data as in Algorithm 4.2*

Let  $\chi = (x, \tau)$  and  $\zeta = (s, \kappa)$   
 For  $N = 0, 1, \dots$   
   Set  $\phi = \phi^{(N)}$   
   Solve (4.4) for  $\Delta\phi_p$   
   Calculate  $\mu_p$  from (4.6) and  $\sigma$   
   Solve (4.8) for  $\Delta\phi_c$   
   Set  $\Delta\phi(\alpha) = \alpha\Delta\phi_p + \alpha^2\Delta\phi_c$   
   Find  $\alpha_{\max} = \arg \max\{\alpha \in [0, 1] \mid \chi + \Delta\chi(\alpha), \zeta + \Delta\zeta(\alpha) \in \bar{K}\}$   
   Update  $\phi^{(N+1)} = \phi + \Delta\phi(\alpha_{\max})$   
 end.

Table 2 lists the results for these algorithms as well as for the long-step path-following method, i.e., Algorithm 4.2 combined with Algorithm 4.3 and it compares those with all four

333

334

Table 3. Comparison between full system, normal equations, and other methods.

	Alg. 4.4 (full)	ppc	inexact_pc
EX1-50	13	16	13
EX1-75	16	19	13
EX1-100	15	19	14
EX1-200	14	21	18
EX2-50	9	13	12
EX2-100	8	11	8
EX2-150	9	15	8
EX2-200	10	11	10
EX3-10-4	13	15	13
EX3-20-4	11	17	15
EX3-30-9	11	17	11
EX3-40-9	13	19	13
EX4-mark	12	15	13
EX4-return	15	17	13
EX4-risk	14	19	13
EX5-10	17	20	18
EX5-20	17	27	29
EX5-40	18	f	29
EX5-80	23	43	29
EX6-10	11	20	11
EX6-20	11	11	11
EX6-40	11	17	12
EX6-60	14	17	13

335 algorithms for SOCP problems we had evaluated for the Seventh DIMACS implementation  
336 Challenge [7]. Similar to LP, Table 2 also shows that in our test case Mehrotra's version of the  
337 predictor-corrector method works better than long-step path-following methods. Moreover,  
338 the quadratic combination of predictor and corrector directions, i.e., Algorithm 4.5, has a  
339 few great results, but it is still worse than Algorithm 4.4 overall, sometimes even worse than  
340 the long-step algorithm. Besides the above two algorithms, we would have two variants  
341 for Algorithm 4.4. First, inspired by [3, 8], we implemented a code for the inexact Newton  
342 method by adding a small residual term to the right hand side. We use `inexact.pc` to denote  
343 this algorithm. Second, using the predictor direction one more time if the reduction of  $\mu$   
344 is significant. We denote this method as `ppc`. Moreover, instead of solving the full system,  
345 we use the normal equations in Table 3. In order to solve large scale problems, solving  
346 the full linear system is not as efficient as solving the normal equations. Nevertheless, for  
347 some problems, the condition number of the coefficient matrix for the normal equations is  
348 much larger than for the full linear system as the point is getting closer to the solution, for  
349 example, EX2-75 (see figure 2).

350 We also show the results for using the full system and the normal equations. In the tables,  
351 'f' denotes failure. SDPT3 and LOQO do not handle rotated cone constraints. Our best  
352 method performs quite well compared to the other codes. As the evaluation [7] had shown,

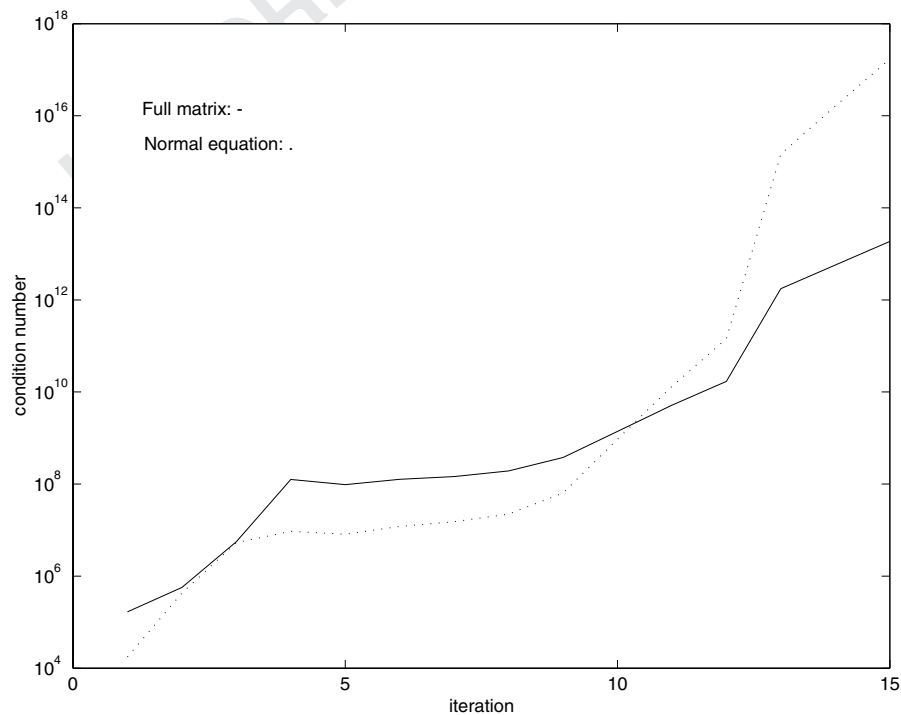


Figure 2. The condition number of the coefficient matrix for full linear system and normal equations.

the code MOSEK which is the only one specialized for convex optimization including SOCP 353  
 is the most robust and efficient for such problems. Its performance is very comparable to that 354  
 of our best method because the approaches are very similar except for a better exploitation 355  
 of sparsity in MOSEK. SeDuMi uses also a self-dual embedding technique and was mainly 356  
 optimized for robustness, which is reflected in the uniformly moderate but not lowest 357  
 iteration counts and the lack of failures. SDPT3 uses an infeasible path-following method 358  
 and could probably handle larger sparse cases nearly as well as MOSEK. LOQO finally, is 359  
 a general NLP code and it treats the SOCP problems as smoothed nondifferentiable NLPs 360  
 which in some cases leads to larger iteration counts. 361

## 5. Conclusion 362

Throughout this study, we discuss second-order cone programming theoretically as well 363  
 as practically. Both the long-step path-following and predictor-corrector algorithms are 364  
 studied. Existence of the step length  $\alpha$  is shown for the long-step method. Furthermore, a 365  
 specific algorithm for choosing the centering parameter  $\sigma$  and the size of the neighborhood 366  
 $\gamma$  is developed and tested successfully on our test cases. Next, several variants of predictor- 367  
 corrector algorithms are also compared. We analyze the numerical behavior of the linear and 368  
 quadratic combinations of predictor and corrector directions. The idea of inexact Newton's 369  
 method and of repeating the predictor direction one more time are also integrated into 370  
 the algorithm. According to our numerical results, the inexact method for SOCP can be 371  
 expected to behave similarly as in LP. Overall, Algorithm 4.4 using the normal equations 372  
 works best for both full and sparse cases. The linear constraints in Example 5 lead to a full 373  
 matrix. The inexact method works competitively in every case except Example 5. 374

## References 375

1. E.D. Andersen, C. Roos, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Mathematical Programming Ser. B*, vol. 95, pp. 249–277, 2003. 376
2. K.M. Bretthauer and B. Shetty, "The nonlinear resource allocation problem," *Operations Research*, vol. 43, pp. 670–683, 1995. 377
3. R.W. Freund, F. Jarre, and S. Mizuno, "Convergence of a class of inexact-interior-point algorithm for linear programs," *Mathematics of Operation Research*, vol. 24, pp. 50–71, 1999. 378
4. Y.J. Kuo, "Interior point algorithm for second-order cone problems with applications," PhD dissertation, Department of Mathematics and Statistic, Arizona State University, 2002. 379
5. M.S. Lobo, L. Vandenberghe, and S. Boyd, "Application of second-order cone programming," *Linear Algebra Application*, vol. 284, pp. 193–228, 1998. 380
6. S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, pp. 575–601, 1992. 381
7. H.D. Mittelmann, "An independent benchmarking of SDP and SOCP solvers," *Mathematical Programming Ser. B*, vol. 95, pp. 407–430, 2003. 382
8. S. Mizuno and F. Jarre, "Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation," *Mathematical Programming*, vol. 84, pp. 357–373, 1999. 383
9. R.D.C. Monteiro and T. Tsuchiya, "Polynomial convergence of primal-dual algorithms for the second-order cone program based on the MZ-family of directions," *Mathematical Programming*, vol. 72, pp. 61–83, 2000. 384
10. Y. Nesterov and M. Todd, "Self-scaled barriers and interior point methods for convex programming," *Mathematics of Operation Research*, vol. 22, pp. 1–42, 1997. 385

- 396** 11. F. Potra and R.Q. Sheng, "Homogeneous interior-point algorithms for semidefinite programming," *Optimization Methods and Software*, vol. 9, pp. 161–184, 1998.
- 397**
- 398** 12. J.F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vols. 11/12, pp. 625–653, 1999.
- 399**
- 400** 13. T. Tsuchiya, "A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming," *Optimization Methods and Software*, vols. 11/12, pp. 141–182, 1999.
- 401**
- 402** 14. R.H. Tütüncü, K.C. Toh, and M.J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming Ser. B*, vol. 95, pp. 189–217, 2003.
- 403**
- 404** 15. R.J. Vanderbei and H. Yurttan, "Using LOQO to solve second-order cone programming problems," Report SOR 98-09, Princeton University, 1998.
- 405**
- 406** 16. R.J. Vanderbei and H. Yurttan, <http://www.princeton.edu/~rvdb/ampl/nlmodels/index.html>.
- 407** 17. J.A. Ventura and C.M. Klein, "A note on multi-item inventory systems with limited capacity," *Operation Research Letters*, vol. 7, pp. 71–75, 1988.
- 408**
- 409** 18. S.J. Wright, *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.
- 410**
- 411** 19. S.-P. Wu, S. Boyd, and L. Vandenberghe, "FIR filter design via spectral factorization and convex optimization." *Applied and Computational Control, Signals, and Circuits*, Biswa Datta editor, Birkhauser, Chap. 5, vol. 1, pp. 215–245, 1998.
- 412**
- 413**
- 414** 20. H. Ziegler, "Solving certain singly constrained convex optimization problems in production planning," *Operation Research Letters*, vol. 1, pp. 246–252, 1982.
- 415**