A QUADRATICALLY CONVERGENT SEQUENTIAL PROGRAMMING METHOD FOR SECOND-ORDER CONE PROGRAMS CAPABLE OF WARM STARTS

XINYI LUO* AND ANDREAS WÄCHTER†

Abstract. We propose a new method for linear second-order cone programs. It is based on the sequential quadratic programming framework for nonlinear programming. In contrast to interior point methods, it can capitalize on the warm-start capabilities of active-set quadratic programming subproblem solvers and achieve a local quadratic rate of convergence.

In order to overcome the non-differentiability or singularity observed in nonlinear formulations of the conic constraints, the subproblems approximate the cones with polyhedral outer approximations that are refined throughout the iterations. For nondegenerate instances, the algorithm implicitly identifies the set of cones for which the optimal solution lies at the extreme points. As a consequence, the final steps are identical to regular sequential quadratic programming steps for a differentiable nonlinear optimization problem, yielding local quadratic convergence.

We prove the global and local convergence guarantees of the method and present numerical experiments that confirm that the method can take advantage of good starting points and can achieve higher accuracy compared to a state-of-the-art interior point solver.

Key words. nonlinear optimization, second-order cone programming, sequential quadratic programming

AMS subject classifications. 90C15, 90C30, 90C55

1. Introduction.

We are interested in the solution of second-order cone programs (SOCPs) of the form

$$\min_{x\in\mathbb{R}^n} \; c^T x \tag{1a}$$

$$\text{s.t. } Ax \le b, \tag{1b}$$

$$x_j \in \mathcal{K}_j \qquad j \in \mathcal{J} := \{1,\dots,p\}, \tag{1c}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, and x_j is a subvector of x of dimension n_j with index set $\mathcal{I}_j \subseteq \{1,\dots,n\}$. We assume that the sets $\mathcal{I}_j$ are disjoint. The set $\mathcal{K}_j$ is the second-order cone of dimension n_j, i.e.,

$$\mathcal{K}_j := \{y \in \mathbb{R}^{n_j} : \|\bar{y}\| \le y_0\}, \tag{2}$$

where the vector y is partitioned into $y = (y_0, \bar{y}^T)^T$ with $\bar{y} = (\bar{y}_1,\dots,\bar{y}_{n_j-1})^T$. These problems arise in a number of important applications (see, e.g., [1, 15])

Currently, most of the commercial software for solving SOCPs implements interior-point algorithms which utilize a barrier function for second-order cones, see, e.g. [9, 11, 16]. Interior-point methods have well-established global and local convergence guarantees [19] and are able to solve large-scale instances, but they cannot take as much of an advantage of a good estimate of the optimal solution as it would be desirable in many situations. For example, in certain applications, such as online optimal control, the same optimization problem has to be solved over and over again,

*Department of Industrial Engineering and Management Sciences, Northwestern University. This author was partially supported by National Science Foundation grant DMS-2012410. E-mail: xinyiluo2023@u.northwestern.edu

†Department of Industrial Engineering and Management Sciences, Northwestern University. This author was partially supported by National Science Foundation grant DMS-2012410. E-mail: andreas.waechter@northwestern.edu

with slightly modified data. In such a case, the optimal solution of one problem provides a good approximation for the new instance. Having a solver that is capable of "warm-starts", i.e., utilizing this knowledge, can be essential when many similar problems have to be solved in a small amount of time.

For some problem classes, including linear programs (LPs), quadratic programs (QPs), or nonlinear programming (NLP), active-set methods offer suitable alternatives to interior-point methods. They explicitly identify the set of constraints that are active (binding) at the optimal solution. When these methods are started from a guess of the active set that is close to the optimal one, they often converge rapidly in a small number of iterations. An example of this is the simplex method for LPs. Its warm-start capabilities are indispensable for efficient branch-and-bound algorithms for mixed-integer linear programs.

Active-set methods for LPs, QPs, or NLPs are also known to outperform interior-point algorithms for problems that are not too large [8].Similarly, active-set methods might be preferable when there are a large number of inequality constraints among which only a few are active, since an interior-point method is designed to consider all inequality constraints in every iteration and consequently solves large linear systems, whereas an active set method can ignore all inactive inequality constraints and encounters potentially much smaller linear systems.

Our goal is to propose an active-set alternative to the interior-point method in the context of SOCP that might provide similar benefits. We introduce a new sequential quadratic programming (SQP) algorithm that, in contrast to interior-point algorithms for SOCPs, has favorable warm-starting capabilities because it can utilize active-set QP solvers. We prove that it is globally convergent, i.e., all limit points of the generated iterates are optimal solutions under mild assumptions, and that it enjoys a quadratic convergence rate for non-degenerate instances. Our preliminary numerical experiments demonstrate that these theoretical properties are indeed observed in practice. They also show that the algorithm is able in some cases to compute a solution to a higher degree of precision than interior point methods. This is expected, again in analogy to the context of LPs, QPs, and NLPs, since an interior point method terminates at a small, but nonzero value of the barrier parameter that cannot be made smaller than some threshold (typically 10^{-6} or 10^{-8}) because the arising linear systems become highly ill-conditioned. In contrast, in the final iteration of the active-set method, the linear systems solved correspond directly to the optimality conditions, without any perturbation introduced by a barrier parameter, and are only as degenerate as the optimal solution of the problem.

The paper is structured as follows. Section 2 reviews the sequential quadratic programming method and the optimality conditions of SOCPs. Section 3 describes the algorithm, which is based on an outer approximation of the conic constraints. Section 4 establishes the global and local convergence properties of the method, and numerical experiments are reported in Section 5. Concluding remarks are offered in Section 6.

1.1. Related work. While a large number of interior-point algorithms for SOCP have been proposed, including some that have been implemented in efficient optimization packages [9, 11, 16], there are only very few approaches for solving SOCPs with an active-set framework. The method proposed by Goldberg and Leyffer [7] is a two-phase algorithm that combines a projected-gradient method with equality-constrained SQP. However, it is limited to instances that have only conic constraints (1c) and no additional linear constraints (1b). Hayashi et al. [10] propose a simplex-type method, where they reformulate the SOCP as a linear semi-infinite program to handle the fact that these instances have infinitely many extreme points. The resulting dual-simplex exchange method shows promising practical behavior. However, in contrast to the method proposed here, the authors conjecture that their method has only an R-linear local convergence rate. Zhadan [23] proposes a similar simplex-type method. Another advantage of the method presented in this paper is that the pivoting algorithm does not need to be designed and implemented from scratch. Instead, it can leverage existing implementations of active-set QP solvers, in particular the efficient handling of linear systems.

The proposed algorithm relies on polyhedral outer approximations based on well-known cutting planes for SOCPs. For instance, the methods for mixed-integer SOCP by Drewes and Ulbrich [4] and Coey et al. [2] use these cutting planes to build LP relaxations of the branch-and-bound subproblems. We note that an LP-based cutting plane algorithm for SOCP could be seen as an active-set method, but it is only linearly convergent. As pointed out in [3], it is crucial to consider the curvature of the conic constraint in the subproblem objective to achieve fast convergence.

The term "SQP method for SOCP" has also been used in the literature to refer to methods for solving nonlinear SOCPs [3, 12, 18, 24]. However, in contrast to the method here, in these approaches, the subproblems themselves are SOCPs (1) and include the linearization of the nonlinear objective and constraints. It will be interesting to explore extensions of the proposed method to nonlinear SOCPs in which feasibility is achieved asymptotically not only for the nonlinear constraints but also for the conic constraints.

1.2. Notation. For two vectors $x, y \in \mathbb{R}^n$, we denote with $x \circ y$ their componentwise product, and the condition $x \perp y$ stands for $x^T y = 0$. For $x \in \mathbb{R}^n$, we define $[x]^+$ as the vector with entries $\max\{x_i, 0\}$. We denote by $\|\cdot\|$, $\|\cdot\|_1$, $\|\cdot\|_\infty$ the Euclidean norm, the ℓ_1-norm, and the ℓ_∞-norm, respectively. For a cone $\mathcal{K}_j$, $e_{ji} \in \mathbb{R}^{n_j}$ is the canonical basis vector with 1 in the element corresponding to x_{ji} for $i \in \{0, \ldots, n_j - 1\}$, and $\text{int}(\mathcal{K}_j)$ and $\text{bd}(\mathcal{K}_j)$ denote the cone's interior and boundary, respectively.

2. Preliminaries. The NLP reformulation of the SOCP is introduced in Section 2.1. We review in Section 2.2 the local convergence properties of the SQP method and in Section 2.3 the penalty function as a means to promote convergence from any starting point. In Section 2.4, we briefly state the optimality conditions and our assumption for the SOCP (1).

2.1. Reformulation as a smooth optimization problem. The definition of the second-order cone in (2) suggests that the conic constraint (1c) can be replaced by the nonlinear constraint

$$r_j(x_j) := \|\bar{x}_j\| - x_{j0} \leq 0$$

without changing the set of feasible points. Consequently, (1) is equivalent to

$$\min_{x \in \mathbb{R}^n} \; c^T x \tag{3a}$$

$$\text{s.t. } Ax \leq b, \tag{3b}$$

$$r_j(x_j) \leq 0, \qquad j \in \mathcal{J}. \tag{3c}$$

Unfortunately, (3) cannot be solved directly with standard gradient-based algorithms for nonlinear optimization, such as SQP methods. The reason is that r_j is not differentiable whenever $\bar{x}_j = 0$. This is particularly problematic when the optimal solution x^* of the SOCP lies at the extreme point of a cone, $x_j^* = 0 \in \mathcal{K}_j$. In that case, the Karush-Kuhn-Tucker (KKT) necessary optimality conditions for the NLP formulation, which are expressed in terms of derivatives, cannot be satisfied. Therefore, any optimization algorithm that seeks KKT points cannot succeed. As a remedy, differentiable approximations of r_j have been proposed in the past; see, for example, [21]. However, high accuracy comes at the price of high curvature, which can make finding the numerical solution of the NLP difficult.

An alternative equivalent reformulation of the conic constraint is given by

$$\|\bar{x}_j\|^2 - x_{j0}^2 \le 0 \text{ and } x_{j0} \ge 0.$$

In this case, the constraint function is differentiable. But if $x_j^* = 0$, its gradient vanishes, and as a consequence, no constraint qualification applies and the KKT conditions do not hold. Therefore, again, a gradient-based method cannot be employed. By using an outer approximation of the cones that is improved in the course of the algorithm, our proposed variation of the SQP method is able to avoid these kinds of degeneracy.

To facilitate the discussion we define a point-wise partition of the cones.

Definition 1. *Let $x \in \mathbb{R}^n$.*
1. *We call a cone $\mathcal{K}_j$* extremal-active *at x, if $x_j = 0$, and we denote with $\mathcal{E}(x) = \{j \in \mathcal{J} : x_j = 0\}$ the set of extremal-active cones at x.*
2. *We define the set $\mathcal{D}(x) = \{j \in \mathcal{J} : \bar{x}_j \neq 0\}$ as the set of all cones for which the function r_j is differentiable at x.*
3. *We define the set $\mathcal{N}(x) = \{j \in \mathcal{J} : x_j \neq 0 \text{ and } \bar{x}_j = 0\}$ as the set of all cones that are not extremal-active and for which r_j is not differentiable x.*

If the set $\mathcal{E}(x^*)$ at an optimal solution x^* were known in advance, we could compute x^* as a solution of (1) by solving the NLP

$$\min_{x \in \mathbb{R}^n} \; c^T x \tag{4a}$$

$$\text{s.t. } Ax \le b, \tag{4b}$$

$$r_j(x) \le 0, \quad j \in \mathcal{D}(x^*), \tag{4c}$$

$$x_j = 0, \qquad j \in \mathcal{E}(x^*). \tag{4d}$$

The constraints involving the linearization of r_j are imposed only if r_j is differentiable at x^*, and variables in cones that are extremal-active at x^* are explicitly fixed to zero. With this, locally around x^*, all functions in (4) are differentiable and we could apply standard second-order algorithms to achieve fast local convergence.

In (4), we omitted the cones in $\mathcal{N}(x^*)$. If x^* is feasible for the SOCP and $j \in \mathcal{N}(x^*)$ we have $\bar{x}_j^* = 0$ and $x_{j0}^* > 0$, and so $r_j(x^*) < 0$. This implies that the nonlinear constraint (4c) for this cone is not active and we can omit it from the problem statement without impacting the optimal solution.

2.2. Local convergence of SQP methods. The proposed algorithm is designed to guide the iterates x^k into the neighborhood of an optimal solution x^*. If the optimal solution is not degenerate and the iterates are sufficiently close to x^*, the steps generated by the algorithm are eventually identical to the steps that the SQP method would take for solving the differentiable optimization problem (4). In this section, we review the mechanisms and convergence results of the basic SQP method [17].

Algorithm 1 Basic SQP Algorithm

Require: Initial iterate x^0 and multiplier estimates λ^0, μ^0, and η^0.

```
for k = 0, 1, 2 ··· do
    Compute H^k from (6).
    Solve QP (5) to get step d^k and multipliers λ̂^k, μ̂_j^k, and η̂^k.
    Set x^{k+1} ← x^k + d^k and μ_j^{k+1} ← μ̂_j^k for all j ∈ D(x*).
end for
```

At an iterate x^k, the basic SQP method, applied to (4), computes a step d^k as an optimal solution to the QP subproblem

$$\min_{d\in\mathbb{R}^n} \; c^T d + \tfrac{1}{2} d^T H^k d \tag{5a}$$

$$\text{s.t. } A(x^k + d) \le b, \tag{5b}$$

$$r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j \le 0, \qquad j \in \mathcal{D}(x^*), \tag{5c}$$

$$x_j^k + d_j = 0, \qquad j \in \mathcal{E}(x^*). \tag{5d}$$

Here, H^k is the Hessian of the Lagrangian function for (4), which in our case is

$$H^k = \sum_{j\in\mathcal{D}(x^*)} \mu_j^k \nabla^2_{xx} r_j(x_j^k), \tag{6}$$

where $\mu_j^k \ge 0$ are estimates of the optimal multipliers for the nonlinear constraint (4c), and where $\nabla^2_{xx} r_j(x_j)$ is the $n \times n$ block-diagonal matrix with

$$\nabla^2 r_j(x_j) = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\|\bar{x}_j\|} I - \frac{\bar{x}_j \bar{x}_j^T}{\|\bar{x}_j\|^3} \end{bmatrix} \tag{7}$$

in the rows and columns corresponding to x_j for $j \in \mathcal{J}$. It is easy to see that $\nabla^2 r_j(x_j)$ is positive semi-definite. The estimates μ_j^k are updated based on the optimal multipliers $\hat{\mu}_j^k \ge 0$ corresponding to (5c).

Algorithm 1 formally states the basic SQP method where $\hat{\lambda}^k \ge 0$ and $\hat{\eta}^k$ denote the multipliers corresponding to (4b) and (4d), respectively. Because we are only interested in the behavior of the algorithm when x^k is close to x^*, we assume here that $\bar{x}_j^k \neq 0$ for all $j \in \mathcal{D}(x^*)$ and for all k, and hence the gradient and Hessian of r_j can be computed. Note that the iterates $\hat{\lambda}^k$ and $\hat{\eta}^k$ are not explicitly needed in Algorithm 1, but they are necessary to measure the optimality error and define the primal-dual iterate sequence that is analyzed in Theorem 2.

A fast rate of convergence can be proven under the following sufficient second-order optimality assumptions [17].

Assumption 1. *Suppose that x^* is an optimal solution of the NLP (4) with corresponding KKT multipliers λ^*, μ^*, and η^*, satisfying the following properties:*

(i) Strict complementarity holds;

(ii) the linear independence constraint qualification (LICQ) holds at x^*, i.e., the gradients of the constraints that hold with equality at x^* are linearly independent;

(iii) the projection of the Lagrangian Hessian $H^* = \sum_{j\in\mathcal{D}(x^*)} \mu_j^* \nabla^2_{xx} r_j(x_j^*)$ into the null space of the gradients of the active constraints is positive definite.

Under these assumptions, the basic SQP algorithm reduces to Newton's method applied to the optimality conditions of (4) and the following result holds [17].

THEOREM 2. *Suppose that Assumption 1 holds and that the initial iterate x^0 and multipliers μ^0 (used in the Hessian calculation) are sufficiently close to x^* and μ^*, respectively. Then the iterates $(x^{k+1}, \hat{\lambda}^k, \hat{\mu}^k, \hat{\eta}^k)$ generated by the basic SQP algorithm, Algorithm 1, converge to $(x^*, \lambda^*, \mu^*, \eta^*)$ at a quadratic rate.*

2.3. Penalty function. Theorem 2 is a local convergence result. Practical SQP algorithms include mechanisms that make sure that the iterates eventually reach such a neighborhood, even if the starting point is far away. To this end, we employ the exact penalty function

$$\varphi(x;\rho) = c^T x + \rho \sum_{j\in\mathcal{J}} [r_j(x_j)]^+ \tag{8}$$

in which $\rho > 0$ is a penalty parameter. Note that we define φ in terms of all conic constraints $\mathcal{J}$, even though r_j appears in (4c) only for $j \in \mathcal{D}(x^*)$. We do this because the proposed algorithm does not know $\mathcal{D}(x^*)$ in advance and the violation of all cone constraints needs to be taken into account when the original problem (1) is solved. Nevertheless, in this section, we may safely ignore the terms for $j \notin \mathcal{D}(x^*)$ because for $j \in \mathcal{E}(x^*)$ we have $x_j^k = 0$ and hence $[r_j(x^k)]^+ = 0$ for all k due to (5d), and when $j \in \mathcal{N}(x^*)$, we have $r_j(x_j^k) < 0$ when x^k is close to x^* since $r_j(x_j^*) < 0$.

It can be shown, under suitable assumptions, that the minimizers of $\varphi(\,\cdot\,;\rho)$ over the set defined by the linear constraints (4b),

$$X = \{x \in \mathbb{R}^n : Ax \le b\}, \tag{9}$$

coincide with the minimizers of (4) when ρ is chosen sufficiently large. Because it is not known upfront how large ρ needs to be, the algorithm uses an estimate, ρ^k, in iteration k, which might be increased during the course of the algorithm.

To ensure that the iterates eventually reach a minimizer of $\varphi(\,\cdot\,;\rho)$, and therefore a solution of (4), we require that the decrease of $\varphi(\,\cdot\,;\rho)$ is at least a fraction of that achieved in the piece-wise linear model of $\varphi(\,\cdot\,;\rho)$ given by

$$m^k(x^k+d;\rho) = c^T(x^k+d) + \rho \sum_{j\in\mathcal{D}(x^k)} [r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j]^+, \tag{10}$$

constructed at x^k. More precisely, the algorithm accepts a trial point $\hat{x}^{k+1} = x^k + d$ as a new iterate only if the sufficient decrease condition

$$\begin{aligned} \varphi(\hat{x}^{k+1};\rho^k) - \varphi(x^k;\rho^k) &\le c_{\text{dec}}\Big(m^k(x^k+d;\rho^k) - m^k(x^k;\rho^k)\Big) \\ &\overset{(10)}{=} c_{\text{dec}}\Big(c^T d - \rho^k \sum_{j\in\mathcal{D}(x^k)} [r_j(x_j^k)]^+\Big) \end{aligned} \tag{11}$$

holds with some fixed constant $c_{\text{dec}} \in (0,1)$. The trial iterate $\hat{x}^{k+1} = x^k + d^k$ with d^k computed from (5) might not always satisfy this condition. The proposed algorithm generates a sequence of improved steps of which one is eventually accepted.

However, to apply Theorem 2, it would be necessary that the algorithm take the original step d^k computed from (5); see Step 4 of Algorithm 1. Unfortunately, $\hat{x}^{k+1} = x^k + d^k$ might not be acceptable even when the iterate x^k is arbitrarily close to a non-degenerate solution x^* satisfying Assumption 1 (a phenomenon called the Maratos effect [14]). Our remedy is to employ the second-order correction step [6], s^k, which is obtained as an optimal solution of the QP

$$\min_{s\in\mathbb{R}^n} c^T(d^k+s)+\tfrac{1}{2}(d^k+s)^T H^k(d^k+s) \tag{12a}$$

$$\text{s.t. } A(x^k+d^k+s)\le b, \tag{12b}$$

$$r_j(x_j^k+d_j^k)+\nabla r_j(x_j^k+d_j^k)^T s_j\le 0, \qquad j\in\mathcal{D}(x^*), \tag{12c}$$

$$x_j^k+d_j^k+s_j=0, \qquad j\in\mathcal{E}(x^*). \tag{12d}$$

For later reference, let $\hat\lambda^{S,k}$, $\hat\mu^{S,k}$ and $\hat\eta^{S,k}$ denote optimal multiplier vectors corresponding to (12b)–(12d), respectively. The algorithm accepts the trial point $\hat{x}^{k+1}=x^k+d^k+s^k$ if it yields sufficient decrease (11) with respect to the original SQP step $d=d^k$. Note that (12) is a variation of the second-order correction that is usually used in SQP methods, for which (12c) reads

$$r_j(x_j^k+d_j^k)+\nabla r_j(x_j^k)^T(d_j^k+s_j)\le 0, \qquad j\in\mathcal{D}(x^*),$$

and avoids the evaluation of $\nabla r_j(x_j^k+d_j^k)$. In our setting, however, evaluating $\nabla r_j(x_j^k+d_j^k)$ takes no extra work and (12c) is equivalent to a supporting hyperplane, see Section 3.1. As the following theorem shows (see, e.g., [6]), this procedure computes steps with sufficient decrease (11) and results in quadratic convergence.

THEOREM 3. *Let Assumption 1 hold and assume that the initial iterate x^0 and multipliers μ^0 are sufficiently close to x^* and μ^*, respectively. Further suppose that $\rho^k=\rho^\infty$ for large k where $\rho^\infty>\mu_j^*$ for all $j\in\mathcal{D}(x^*)$.*

1. *Consider an algorithm that generates a sequence of iterates by setting $(x^{k+1},\lambda^{k+1},\mu^{k+1},\eta^{k+1})=(x^k+d^k,\hat\lambda^k,\hat\mu^k,\hat\eta^k)$ or $(x^{k+1},\lambda^{k+1},\mu^{k+1},\eta^{k+1})=(x^k+d^k+s^k,\hat\lambda^{S,k},\hat\mu^{S,k},\hat\eta^{S,k})$ for all $k=0,1,2,\ldots$. Then $(x^k,\lambda^k,\mu^k,\eta^k)$ converges to $(x^*,\lambda^*,\mu^*,\eta^*)$ at a quadratic rate.*
2. *Further, for all k, either $\hat{x}^{k+1}=x^k+d^k$ or $\hat{x}^{k+1}=x^k+d^k+s^k$ satisfies the acceptance criterion (11).*

2.4. Optimality conditions for SOCP. The proposed algorithm aims at finding an optimal solution of the SOCP (1), or equivalently, values of the primal variables, $x^*\in\mathbb{R}^n$, and the dual variables, $\lambda^*\in\mathbb{R}^m$ and $z_j^*\in\mathbb{R}^{n_j}$ for $j\in\mathcal{J}$, that satisfy the necessary and sufficient optimality conditions [1, Theorem 16]

$$c+A^T\lambda^*-z^*=0, \tag{13a}$$

$$Ax^*-b\le 0\perp\lambda^*\ge 0, \tag{13b}$$

$$\mathcal{K}_j\ni x_j^*\perp z_j^*\in\mathcal{K}_j, \qquad j\in\mathcal{J}. \tag{13c}$$

A thorough discussion of SOCPs is given in the comprehensive review by Alizadeh and Goldfarb [1]. The authors consider the formulation in which the linear constraints (1b) are equality constraints, but the results in [1] can be easily extended to inequalities.

The primal-dual solution (x^*,λ^*,z^*) is unique under the following assumption.

ASSUMPTION 2. *(x^*,λ^*,z^*) is a non-degenerate primal-dual solution of the SOCP (1) at which strict complementarity holds.*

The definition of non-degeneracy for SOCP is somewhat involved and we refer the reader to [1, Theorem 21]. Strict complementarity holds if $x_j^*+z_j^*\in\text{int}(\mathcal{K}_j)$ and implies that: (i) $x_j^*\in\text{int}(K_j)\implies z_j^*=0$; (ii) $z_j^*\in\text{int}(K_j)\implies x_j^*=0$; (iii) $x_j^*\in\text{bd}(K_j)\setminus\{0\}\iff z_j^*\in\text{bd}(K_j)\setminus\{0\}$; and (iv) not both x_j^* and z_j^* are zero.

3. Algorithm. The proposed algorithm solves the NLP formulation (3) using a variation of the SQP method. Since the functional formulation of the cone constraints (3c) might not be differentiable at all iterates or at an optimal solution, the cones are approximated by a polyhedral outer approximation using supporting hyperplanes.

The approximation is done so that the method implicitly identifies the constraints that are extremal-active at an optimal solution x^*, i.e., $\mathcal{E}(x^*) = \mathcal{E}(x^k)$ for large k. More precisely, we will show that close to a non-degenerate optimal solution, the steps generated by the proposed algorithm are identical to those computed by the QP subproblem (5) for the basic SQP algorithm for solving (4). Consequently, fast local quadratic convergence is achieved, as discussed in Section 2.2.

3.1. Supporting hyperplanes. In the following, consider a particular cone $\mathcal{K}_j$ and let $\mathcal{Y}_j$ be a finite subset of $\{y_j \in \mathbb{R}^{n_j} : \bar{y}_j \neq 0, y_{j0} \geq 0\}$. We define the cone

$$\mathcal{C}_j(\mathcal{Y}_j) = \left\{x_j \in \mathbb{R}^{n_j} : x_{j0} \geq 0 \text{ and } \nabla r_j(y_j)^T x_j \leq 0 \text{ for all } y_j \in \mathcal{Y}_j\right\} \tag{14}$$

generated by the points in $\mathcal{Y}_j$. For each $x_j \in \mathcal{K}_j$ we have $r_j(x_j) \leq 0$, and using

$$\nabla r_j(x_j) = \left(-1, \frac{\bar{x}_j^T}{\|\bar{x}_j\|}\right)^T, \tag{15}$$

we obtain for any $y_j \in \mathcal{Y}_j$ that

$$\nabla r_j(y_j)^T x_j = \frac{1}{\|\bar{y}_j\|}\bar{y}_j^T \bar{x}_j - x_{j0} \leq \frac{1}{\|\bar{y}_j\|}\|\bar{y}_j\|\|\bar{x}_j\| - x_{j0} = r_j(x_j) \leq 0.$$

Therefore $\mathcal{C}_j(\mathcal{Y}_j) \supseteq \mathcal{K}_j$. Also, for $y_j \in \mathcal{Y}_j$, consider $x_j = (1, \bar{y}_j^T/\|\bar{y}_j\|)^T$. Then

$$\nabla r_j(y_j)^T x_j = \frac{\bar{y}_j^T}{\|\bar{y}_j\|}\frac{\bar{y}_j}{\|\bar{y}_j\|} - 1 = 1 - 1 = 0,$$

and also $r_j(x_j) = \|\bar{x}_j\| - x_{j0} = \bar{y}_j/\|\bar{y}_j\| - 1 = 0$. Hence $x_j \in \mathcal{C}_j(\mathcal{Y}_j) \cap \mathcal{K}_j$. Therefore, for any $y_j \in \mathcal{Y}_j$, the inequality

$$\nabla r_j(y_j)^T x_j \leq 0 \tag{16}$$

defines a hyperplane that supports $\mathcal{K}_j$ at $(1, \bar{y}_j/\|\bar{y}_j\|)$. In summary, $\mathcal{C}_j(\mathcal{Y}_j)$ is a polyhedral outer approximation of $\mathcal{K}_j$, defined by supporting hyperplanes.

In addition, writing $\mathcal{Y}_j = \{y_{j,1}, \ldots, y_{j,m}\}$, we also define the cone

$$\mathcal{C}_j^\circ(\mathcal{Y}_j) := \left\{-\sum_{l=1}^m \sigma_{j,l}\nabla r_j(y_{j,l}) + \eta_j e_{j0} : \sigma_j \in \mathbb{R}_+^m, \eta_j \geq 0\right\}. \tag{17}$$

For all $x_j \in \mathcal{C}_j(\mathcal{Y}_j)$ and $z_j = -\sum_{l=1}^m \sigma_{j,l}\nabla r_j(y_{j,l}) + \eta_j e_{j0} \in \mathcal{C}_j^\circ(\mathcal{Y}_j)$, we have

$$x_j^T z_j = -\sum_{l=1}^m \sigma_{j,l}\nabla r_j(y_{j,l})^T x_j + \eta_j x_{j0} \geq 0$$

because $\nabla r_j(y_{j,l})^T x_j \leq 0$ and $x_{j0} \geq 0$ from the definition of $\mathcal{C}_j(\mathcal{Y}_j)$. Therefore $\mathcal{C}_j^\circ(\mathcal{Y}_j)$ is included in the dual of the cone $\mathcal{C}_j(\mathcal{Y}_j)$.

Now define $R = [-\nabla r_j(y_{j,1}), \ldots, -\nabla r_j(y_{j,m}), e_{j0}]$ and let $z_j \in \mathbb{R}^{n_j}$ be in the dual of $\mathcal{C}_j(\mathcal{Y}_j)$. Since this implies that $x_j^T z_j \geq 0$ for all $x \in \mathcal{C}_j(\mathcal{Y}_j) = \{\mathbb{R}^{n_j} : R^T x_j \geq 0\}$, Farkas' lemma yields that $z_j = R \cdot (\sigma^T, \eta)^T$ for some $\sigma_j \in \mathbb{R}_+^m$ and $\eta_j \geq 0$, i.e., $z_j \in \mathcal{C}_j^\circ(\mathcal{Y}_j)$.

Overall we proved that $\mathcal{C}_j^\circ(\mathcal{Y}_j)$ defined in (17) is the dual of $\mathcal{C}_j(\mathcal{Y}_j)$, and since $\mathcal{C}_j(\mathcal{Y}_j) \supseteq \mathcal{K}_j$, this implies $\mathcal{C}_j^\circ(\mathcal{Y}_j) \subseteq \mathcal{K}_j$.

Algorithm 2 Preliminary SQP Algorithm

Require: Initial iterate x^0 and sets $\mathcal{Y}_j^0$ for $j \in \mathcal{J}$.

1: **for** $k = 0, 1, 2 \cdots$ **do**
2: Choose H^k.
3: Solve subproblem (18) to get step d^k.
4: Set $x^{k+1} \leftarrow x^k + d^k$.
5: Set $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}^+_{pr,j}(\mathcal{Y}_j^k, x_j^k)$ for $j \in \mathcal{J}$.
6: **end for**

3.2. QP subproblem. In each iteration, at an iterate x^k, the proposed algorithm computes a step d^k as an optimal solution of the subproblem

$$
\begin{aligned}
&\text{(18a)} && \min_{d \in \mathbb{R}^n} \; c^T d + \tfrac{1}{2} d^T H^k d \\
&\text{(18b)} && \text{s.t. } A(x^k + d) \le b, \\
&\text{(18c)} && \qquad r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j \le 0, \qquad j \in \mathcal{D}(x^k), \\
&\text{(18d)} && \qquad x_j^k + d_j \in \mathcal{C}_j(\mathcal{Y}_j^k), \qquad\qquad j \in \mathcal{J}.
\end{aligned}
$$

Here, H^k is a positive semi-definite matrix that captures the curvature of the nonlinear constraint (3c), and for each cone, $\mathcal{Y}_j^k$ is the set of hyperplane-generating points that have been accumulated up to this iteration. From (14), we see that (18d) can be replaced by linear constraints. Consequently, (18) is a QP and can be solved as such.

Algorithm 2 describes a preliminary version of the proposed SQP method based on this subproblem. Observe that the linearization (18c) can be rewritten as

$$
\begin{aligned}
0 &\ge r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j = \|\bar{x}_j^k\| - x_{j0}^k - d_{j0} + \frac{(\bar{x}_j^k)^T \bar{d}_j}{\|\bar{x}_j^k\|} \\
&= \frac{1}{\|\bar{x}_j^k\|}(\bar{x}_j^k)^T(\bar{x}_j^k + \bar{d}_j) - (x_{j0}^k + d_{j0}) = \nabla r_j(x_j^k)^T (x_j^k + d_j)
\end{aligned}
$$

and is equivalent to the hyperplane constraint generated at x_j^k. Consequently, if $x_j^k \notin \mathcal{K}_j$, then $r_j(x_j^k) > 0$ and (18c) acts as a cutting plane that excludes x_j^k. Using the update rule

$$
\text{(19)} \qquad \mathcal{Y}^+_{pr,j}(\mathcal{Y}_j, x_j) = \begin{cases} \mathcal{Y}_j \cup \{x_j\} & \text{if } \bar{x}_j \ne 0 \text{ and } r_j(x_j) > 0, \\ \mathcal{Y}_j & \text{otherwise,} \end{cases}
$$

in Step 5 makes sure that x_j^k is excluded in all future iterations.

In our algorithm, we initialize $\mathcal{Y}_j^0$ so that

$$
\text{(20)} \qquad \mathcal{Y}_j^0 \supseteq \hat{\mathcal{Y}}_j^0 := \{e_{ji} : i = 1, \ldots, n_j - 1\} \cup \{-e_{ji} : i = 1, \ldots, n_j - 1\}.
$$

In this way, $x_j = 0$ is an extreme point of $\mathcal{C}_j(\mathcal{Y}_j^0)$, as it is for $\mathcal{K}_j$, and the challenging aspect of the cone is already captured in the first subproblem. By choosing the coordinate vectors e_{ji} we have $\nabla r_j(e_{ji})^T x_j = x_{ji} - x_{j0}$, and the hyperplane constraint (16) becomes a very sparse linear constraint.

When $H^k = 0$ in each iteration, this procedure becomes the standard cutting plane algorithm for the SOCP (1). It is well-known that the cutting plane algorithm is convergent in the sense that every limit point of the iterates is an optimal solution of the SOCP (1), but the convergence is typically slow. In the following sections, we describe how Algorithm 2 is augmented to achieve fast local convergence. The full method is stated formally in Algorithm 3.

3.3. Identification of extremal-active cones. We now describe a strategy that enables our algorithm to identify those cones that are extreme-active at a non-degenerate solution x^* within a finite number of iterations, i.e., $\mathcal{E}(x^k) = \mathcal{E}(x^*)$ for all large k. This will make it possible to apply a second-order method and achieve quadratic local convergence.

Consider the optimality conditions for the QP subproblem (18):

$$c + H^k d^k + A^T \hat{\lambda}^k + \sum_{j \in \mathcal{D}(x^k)} \hat{\mu}_j^k \nabla_x r_j(x^k) - \hat{\nu}^k = 0, \tag{21a}$$

$$A(x^k + d^k) - b \le 0 \perp \hat{\lambda}^k \ge 0, \tag{21b}$$

$$r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j^k \le 0 \perp \hat{\mu}_j \ge 0, \qquad j \in \mathcal{D}(x^k), \tag{21c}$$

$$\mathcal{C}_j(\mathcal{Y}_j^k) \ni x_j^k + d_j^k \perp \hat{\nu}_j^k \in \mathcal{C}_j^\circ(\mathcal{Y}_j^k), \quad j \in \mathcal{J}. \tag{21d}$$

Here, $\hat{\lambda}^k$, $\hat{\mu}_j^k$, and $\hat{\nu}_j^k$ are the multipliers corresponding to the constraints in (18); for completeness, we define $\hat{\mu}_j^k = 0$ for $j \in \mathcal{J} \setminus \mathcal{D}(x^k)$. In (21a), $\nabla_x r_j(x^k)$ is the vector in $\mathbb{R}^n$ that contains $\nabla r_j(x_j^k)$ in the elements corresponding to x_j and is zero otherwise. Similarly, $\hat{\nu}^k \in \mathbb{R}^n$ is equal to $\hat{\nu}_j^k$ in the elements corresponding to x_j for all $j \in \mathcal{J}$ and zero otherwise.

Let us define

$$\hat{\mathcal{Y}}_j^k := \begin{cases} \mathcal{Y}_j^k \cup \{x_j^k\}, & \text{if } j \in \mathcal{D}(x^k), \\ \mathcal{Y}_j^k, & \text{if } j \in \mathcal{J} \setminus \mathcal{D}(x^k). \end{cases} \tag{22}$$

It is easy to verify that, for $j \in \mathcal{D}(x^k)$, $\nabla r_j(x_j^k) x_j^k = r_j(x_j^k)$ and hence $r_j(x_j^k)^T(x_j^k + d^k) \le 0$ from (21c). As a consequence we obtain $x_j^k + d_j^k \in \mathcal{C}_j(\hat{\mathcal{Y}}_j^k)$ for all $j \in \mathcal{J}$. Furthermore, $\hat{\nu}_j^k \in \mathcal{C}_j^\circ(\mathcal{Y}_j^k)$ implies that

$$\hat{\nu}_j^k = -\sum_{l=1}^{m} \sigma_{j,l}^k \nabla r_j(y_{j,l}^k) + \eta_j^k e_{j0}$$

for suitable values of $\sigma_{j,l}^k \ge 0$ and $\eta_j^k \ge 0$. Then $\hat{z}_j^k := -\hat{\mu}_j^k \nabla r_j(x^k) + \hat{\nu}_j^k \in \mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k)$ and

$$\hat{z}^k = c + H^k d^k + A^T \hat{\lambda}^k \tag{23}$$

from (21a). In conclusion, if $(d, \hat{\lambda}^k, \hat{\mu}^k, \hat{\nu}^k)$ is a primal-dual solution of the QP subproblem (18), then $(d, \hat{\lambda}^k, \hat{z}^k)$ satisfies the conditions

$$c + H^k d^k + A^T \hat{\lambda}^k - \hat{z}^k = 0, \tag{24a}$$

$$A(x^k + d^k) - b \le 0 \perp \hat{\lambda}^k \ge 0, \tag{24b}$$

$$\mathcal{C}_j(\hat{\mathcal{Y}}_j^k) \ni x_j^k + d_j^k \perp \hat{z}_j^k \in \mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k), \quad j \in \mathcal{J}, \tag{24c}$$

which more closely resembles the SOCP optimality conditions (13). Our algorithm maintains primal-dual iterates $(x^{k+1}, \hat{\lambda}^k, \hat{z}^k)$ that are updated based on (24).

Suppose that strict-complementarity holds at a primal-dual solution (x^*, λ^*, z^*) of the SOCP (1) and that $(x^{k+1}, \hat{\lambda}^k, \hat{z}^k) \to (x^*, \lambda^*, z^*)$. If $j \notin \mathcal{E}(x^*)$ then $x_j^* \in \mathcal{K}_j$ implies $x_{j0}^* > 0$. As x_j^k converges to x_j^*, we have $x_{j0}^k > 0$ and therefore $j \notin \mathcal{E}(x^k)$ for sufficiently large k. This yields $\mathcal{E}(x^k) \subseteq \mathcal{E}(x^*)$. We now derive a modification of Algorithm 2 that ensures that $\mathcal{E}(x^*) \subseteq \mathcal{E}(x^k)$ for all sufficiently large k under Assumption 2.

Consider any $j \in \mathcal{E}(x^*)$. We would like to have

$$\hat{z}_j^k \in \operatorname{int}(\mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k)) \tag{25}$$

for all large k, since then complementarity in (24c) implies that $x_j^{k+1} = x_j^k + d_j^k = 0$ and hence $j \in \mathcal{E}(x^{k+1})$ for all large k. We will later show that Assumption 2 implies that $\hat{z}_j^k \to z_j^*$ and that there exists a neighborhood $N_\epsilon(z_j^*) = \{z_j \in \mathbb{R}^{n_j} : \|z_j - z_j^*\| \leq \epsilon\}$ of z_j^* so that $z_j \in \operatorname{int}(\mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^0 \cup \{-y_j\}))$ if $z_j, y_j \in N_\epsilon(z_j^*)$; see Remark 14. This suggests that some vector close to $-z_j^*$ should eventually be included in $\hat{\mathcal{Y}}_j^k$ because then (25) holds when $\hat{z}_j^k$ is close enough to z_j^*. For this purpose, the algorithm computes

$$\check{z}^k = c + A^T \hat{\lambda}^k,$$

which also converges to z_j^* (see (13a)), and sets $\mathcal{Y}_j^{k+1}$ to $\mathcal{Y}_{du,j}^+(\mathcal{Y}_j^k, x_j^k, \check{z}_j^k)$, where

$$\mathcal{Y}_{du,j}^+(\mathcal{Y}_j, x_j, z_j) = \begin{cases} \mathcal{Y}_j \cup \{-z_j\} & \text{if } x_j \neq 0, \bar{z}_j \neq 0 \text{ and } r_j(z_j) < 0, \\ \mathcal{Y}_j & \text{otherwise.} \end{cases} \tag{26}$$

The update is skipped when $x_j^k = 0$ (because then j is already in $\mathcal{E}(x^k)$ and no additional hyperplane is needed), and when $\bar{\check{z}}_j^k = 0$ or $r_j(\check{z}_j^k) \geq 0$, which might indicate that $z_j^* \notin \operatorname{int}(\mathcal{K}_j)$ and $j \notin \mathcal{E}(x^*)$.

3.4. Fast NLP-SQP steps. Now that we have a mechanism in place that makes sure that the extremal-active cones are identified in a finite number of iterations, we present a strategy that emulates the basic SQP Algorithm 1 and automatically takes quadratically convergent SQP steps, i.e., solutions of the SQP subproblem (5), close to x^*. For the discussion in this section, we again assume that x^* is a unique solution at which Assumption 2 holds.

Suppose that $\mathcal{E}(x^k) = \mathcal{E}(x^*)$ for large k due to the strategy discussed in Section 3.3. This means that the outer approximation (18d) of $\mathcal{K}_j$ for $j \in \mathcal{E}(x^*)$ is sufficient to fix x_j^k to zero and is therefore equivalent to the constraint (5d) in the basic SQP subproblem. However, (18) includes the outer approximations for all cones, including those for $j \notin \mathcal{E}(x^*)$, which are not present in (5). Consequently, the desired SQP step from (5) might not be feasible for (18).

As a remedy, at the beginning of an iteration, the algorithm first computes an NLP-SQP step as an optimal solution $d^{S,k}$ of a relaxation of (18),

$$\min_{d \in \mathbb{R}^n} \; c^T d + \tfrac{1}{2} d^T H^k d \tag{27a}$$

$$\text{s.t. } A(x^k + d) \leq b \tag{27b}$$

$$r_j(x_j^k) + \nabla r_j(x_j^k)^T d_j \leq 0, \qquad j \in \mathcal{D}(x^k) \tag{27c}$$

$$x_{j0}^k + d_{j0} \geq 0, \qquad j \in \mathcal{D}(x^k) \setminus \hat{\mathcal{E}}^k \tag{27d}$$

$$x_j^k + d_j \in \mathcal{C}_j(\mathcal{Y}_j^k) \qquad j \in \hat{\mathcal{E}}^k, \tag{27e}$$

where $\hat{\mathcal{E}}^k = \mathcal{E}(x^k)$. In this way, the outer approximations are imposed only for the currently extremal-active cones, while for all other cones only the linearization (27c) is considered, just like in (5), with the additional restriction (27d) that ensure $x_{j0}^{k+1} \geq 0$. Let $\hat{\lambda}^k$, $\hat{\mu}_j^k$, $\hat{\eta}_j^k$, and $\hat{\nu}_j^k$ be the optimal corresponding to the constraints in (27) (set to zero for non-existing constraints) and define $\hat{z}^k$ as in (23). Then the optimality conditions (24) hold again, this time with $d^k = d^{S,k}$, but instead of (22) we have

$$\hat{\mathcal{Y}}_j^k := \begin{cases} \{x_j^k\} & \text{if } j \in \mathcal{D}(x^k) \setminus \hat{\mathcal{E}}^k, \\ \mathcal{Y}_j^k \cup \{x_j^k\} & \text{if } j \in \hat{\mathcal{E}}^k \cap \mathcal{D}(x^k), \\ \mathcal{Y}_j^k & \text{if } j \in \hat{\mathcal{E}}^k \setminus \mathcal{D}(x^k). \end{cases} \tag{28}$$

When x^k is not close to x^* and $\mathcal{E}(x^*) \neq \mathcal{E}(x^k)$, QP (27) might result in poor steps that go far outside of $\mathcal{K}_j$ for some $j \in \mathcal{D}(x^k) \setminus \hat{\mathcal{E}}^k$ and undermine convergence. Therefore, we iteratively add more cones to $\hat{\mathcal{E}}^k$ until

$$x_{j0}^k + d_{j0}^{S,k} > 0 \text{ only for } j \in \mathcal{J} \setminus \hat{\mathcal{E}}^k, \tag{29}$$

i.e., when a cone is approximated only by its linearization (27c), the step does not appear to target its extreme point. This property is necessary to show that $\mathcal{E}(x^k) = \mathcal{E}(x^*)$ for all large k also for the case that new iterates are computed from (27) instead of (18). Note that in the extreme case $\hat{\mathcal{E}}^k = \mathcal{J}$ and (27) is identical to (18). This loop can be found in Steps 6–9 in Algorithm 3.

Since there is no guarantee that (27) yields iterates that converge to x^*, the algorithm discards the NLP-SQP step in certain situations and falls back to the original method to recompute a new step from (18), as in the original method. In Section 3.6 we describe how we use the exact penalty function (8) to determine when this is necessary.

3.5. Hessian matrix. Motivated by (6), we compute the Hessian matrix H^k in (18) and (27) from

$$H^k = \sum_{j \in \mathcal{D}(x^k)} \mu_j^k \nabla_{xx}^2 r_j(x^k), \tag{30}$$

where $\mu_j^k \geq 0$ are multiplier estimates for the nonlinear constraint (3c). Because $\nabla^2 r_j(x_j^k)$ is positive semi-definite and $\mu_j^k \geq 0$, also H^k is positive semi-definite.

In the final phase, when we intend to emulate the basic SQP Algorithm 1. Therefore, we set $\mu_j^{k+1} = \hat{\mu}_j^k$ for $j \in \mathcal{D}(x^k)$, where $\hat{\mu}_j^k$ are the optimal multipliers for (27c), when the fast NLP-SQP step was accepted. But we also need to define a value for μ_j^{k+1} when the step is computed from (18) where, in addition to the linearization of r_j, hyperplanes (18d) are used to approximate all cones. By comparing the optimality conditions of the QPs (18) and (5) we now derive an update for μ_j^{k+1}.

Suppose that $j \in \mathcal{D}(x^{k+1}) \cap \mathcal{D}(x^k)$. Then (21a) yields

$$c_j + H_{jj}^k d_j^k + A_j^T \hat{\lambda}^k + \hat{\mu}_j^k \nabla r_j(x_j^k) - \hat{\nu}_j^k = 0, \tag{31}$$

where $H_{jj}^k = \mu_j^k \nabla^2 r_j(x_j^k)$ because of (30). Here, the dual information for the nonlinear constraint is split into $\hat{\mu}_j^k$ and $\hat{\nu}_j^k$ and needs to be condensed into a single number, μ_j^{k+1}, so that we can compute H^k from (30) in the next iteration.

Recall that, in the basic SQP Algorithm 1, the new multipliers μ_j^{k+1} are set to the optimal multipliers of the QP (5), which satisfy

$$c_j + H_{jj}^k d_j^k + A_j^T \hat{\lambda}^k + \mu_j^{k+1} \nabla r_j(x_j^k) = 0. \tag{32}$$

A comparison with (31) suggests to choose μ_j^{k+1} so that $\mu_j^{k+1} \nabla r_j(x_j^k) \approx \hat{\mu}_j^k \nabla r_j(x_j^k) - \hat{\nu}_j^k$. Multiplying both sides with $\nabla r_j(x_j^k)^T$ and solving for μ^{k+1} yields

$$\mu_j^{k+1} = \hat{\mu}_j^k - \frac{\nabla r_j(x_j^k)^T \hat{\nu}_j^k}{\|\nabla r_j(x_j^k)\|^2}.$$

Note that $\mu_j^{k+1} = \hat{\mu}_j^k$ if the outer approximation constraint (18d) is not active and therefore $\hat{\nu}_j^k = 0$ for j. In this case, we recover the basic SQP update, as desired.

Now suppose that $j \in \mathcal{D}(x^{k+1}) \setminus \mathcal{D}(x^k)$. Again comparing (31) with (32) suggests a choice so that $\mu_j^{k+1} \nabla r_j(x_j^{k+1}) \approx -\hat{\nu}_j^k$, where we substituted $\nabla r_j(x_j^k)$ by $\nabla r_j(x_j^{k+1})$ because the former is not defined for $j \notin \mathcal{D}(x^k)$. In this case, multiplying both sides with $\nabla r_j(x_j^{k+1})^T$ and solving for μ^{k+1} yields

$$\mu_j^{k+1} = -\frac{\nabla r_j(x_j^{k+1})^T \hat{\nu}_j^k}{\|\nabla r_j(x_j^{k+1})\|^2}.$$

In summary, in each iteration in which (18) determines the new iterate, we update

$$\mu_j^{k+1} = \begin{cases} \hat{\mu}_j^k - \frac{\nabla r_j(x_j^k)^T \hat{\nu}_j^k}{\|\nabla r_j(x_j^k)\|^2} & j \in \mathcal{D}(x^{k+1}) \cap \mathcal{D}(x^k) \\ -\frac{\nabla r_j(x_j^{k+1})^T \hat{\nu}_j^k}{\|\nabla r_j(x_j^{k+1})\|^2} & j \in \mathcal{D}(x^{k+1}) \setminus \mathcal{D}(x^k) \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

The choice above leads to quadratic convergence for non-degenerate instances, but it is common for the global convergence analysis of SQP methods to permit any positive semi-definite Hessian matrix H^k, as long as it is bounded. Since we were not able to exclude the case that μ_j^k or $1/x_{j0}^k$ are unbounded for some cone $j \in \mathcal{J}$, in which case H^k defined in (30) is unbounded, we fix a large threshold $c_H > 0$ and rescale the Hessian matrix according to

$$H_k \leftarrow H_k \cdot \min\{1, c_H / \|H_k\|\} \tag{34}$$

so that $\|H_k\| \leq c_H$ in every iteration. In this way, global convergence is guaranteed, but the fast local convergence rate might be impaired if c_H is chosen too small so that H^k defined in (30) must be rescaled. Therefore, in practice, we set c_H to a very large number and (34) is never actually triggered in our numerical experiments.

3.6. Penalty function. The steps computed from (18) and (27) do not necessarily yield a convergent algorithm and a safeguard is required to force the iterates into a neighborhood of an optimal solution. Here, we utilized the exact penalty function (8) and accept a new iterate only if the sufficient decrease condition (11) holds.

As discussed in Section 3.4, at the beginning of an iteration, the algorithm first computes an NLP-SQP step $d^{S,k}$ from (27). The penalty function can now help us to decide whether this step makes sufficient progress towards an optimal solution, and we only accept the trial point $\hat{x}^{k+1} = x^k + d^{S,k}$ as a new iterate if (11) holds with $d = d^{S,k}$.

If the penalty function does not accept $d^{S,k}$, there is still a chance that $d^{S,k}$ is making rapid progress towards the solution, but, as discussed in Section 2.2, the Maratos effect is preventing the acceptance of $d^{S,k}$. As a remedy, we compute, analogously to (12), a second-order correction step s^k for (27) as a solution of

$$
\begin{aligned}
\min_{s\in\mathbb{R}^n}\ & c^T(d^{S,k}+s) + \tfrac{1}{2}(d^{S,k}+s)^T H^k (d^{S,k}+s) \\
\text{s.t. } & A(x^k + d^{S,k} + s) \le b, \\
& r_j(x_j^k + d_j^{S,k}) + \nabla r_j(x_j^k + d_j^{S,k})^T s_j \le 0, && j \in \mathcal{D}(x^k), \\
& x_{j0}^k + d_{j0}^{S,k} + s_{j0} \ge 0, && j \in \mathcal{D}(x^k) \setminus \hat{\mathcal{E}}^k, \\
& x_j^k + d_j^{S,k} + s_j \in \mathcal{C}_j(\mathcal{Y}_j^k), && j \in \hat{\mathcal{E}}^k,
\end{aligned} \tag{35}
$$

and accept the trial point $\hat{x}^{k+1} = x^k + d^{S,k} + s^k$ if it satisfies (11) with $d = d^{S,k}$. Let again $\hat{\lambda}^k$, $\hat{\mu}_j^k$, $\hat{\eta}_j^k$, and $\hat{\nu}_j^k$ denote the optimal multipliers in (35) and define $\hat{z}^k$ as in (23). The optimality conditions (24) still hold, this time with $d^k = d^{S,k} + s^k$ and

$$
\hat{\mathcal{Y}}_j^k := \begin{cases} \{x_j^k + d_j^{S,k}\}, & \text{if } j \in \mathcal{D}(x^k) \setminus \hat{\mathcal{E}}^k, \\ \mathcal{Y}_j^k \cup \{x_j^k + d_j^{S,k}\}, & \text{if } j \in \mathcal{D}(x^k) \cap \hat{\mathcal{E}}^k, \\ \mathcal{Y}_j^k, & \text{if } j \in \hat{\mathcal{E}}^k. \end{cases} \tag{36}
$$

If neither $d^{S,k}$ nor $d^{S,k} + s^k$ has been accepted, we give up on fast NLP-SQP steps and instead revert to QP (18) which safely approximates every cone with an outer approximation. However, the trial point $\hat{x}^{k+1} = x^k + d^k$ with the step d^k obtained from (18) does not necessarily satisfy (11). In that case, the algorithm adds $x^k + d^k$ to $\mathcal{Y}_j^k$ to cut off $x^k + d^k$ and resolves (18) to get a new trial step d^k. In an inner loop (Steps 21–31), this procedure is repeated until, eventually, a trial step is obtained that satisfies (11). We will show that (11) holds after a finite number of iterations of the inner loop.

It remains to discuss the update of the penalty parameter estimate ρ^k. One can show (see Lemma 4) that an optimal solution of x^* of the SOCP with multipliers z^* is a minimizer of $\phi(\ \cdot, \rho)$ over the set X defined in (9) if $\rho > \|z^*_{\mathcal{J},0}\|_\infty$, where $z^*_{\mathcal{J},0} = (z^*_{1,0}, \ldots, z^*_{p,0})^T$. Since z^* is not known *a priori*, the algorithm uses the update rule $\rho^k = \rho_{\text{new}}(\rho^{k-1}, z^k)$ where

$$
\rho_{\text{new}}(\rho_{\text{old}}, z) := \begin{cases} \rho_{\text{old}} & \text{if } \rho_{\text{old}} > \|z_{\mathcal{J},0}\|_\infty \\ c_{\text{inc}} \cdot \|z_{\mathcal{J},0}\|_\infty & \text{otherwise,} \end{cases} \tag{37}
$$

with $c_{\text{inc}} > 1$. We will prove in Lemma 8 that the sequence $\{z^k\}_{k=1}^\infty$ is bounded under Slater's constraint qualification. Therefore, this rule will eventually settle at a final penalty parameter ρ^∞ that is not changed after a finite number of iterations.

During an iteration of the algorithm, several trial steps may be considered and a preliminary parameter value is computed from (37) for each one. At the end of the iteration, the parameter value corresponding to the accepted trial step is stored. Note that the acceptance test for the second-order correction step from (35) needs to be done with the penalty parameter computed for the regular NLP-SQP step from (27).

Algorithm 3 SQP Algorithm for SOCP.

Require: Initial iterate $x^0 \in X$ with $x_{j,0} \geq 0$, multipliers $\mu_j^0 \in \mathbb{R}_+$, penalty parameter $\rho^{-1} > 0$; constants $c_{\text{dec}} \in (0,1)$, $c_{\text{inc}} > 1$, and $c_H > 0$.

1: Initialize $\mathcal{Y}_j^0$ so that (20) is satisfied.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Compute H^k using (30). Rescale according to (34) if $\|H^k\| > c_H$.
4: Set $\hat{\mathcal{E}}^k \leftarrow \mathcal{E}(x^k)$.
5: Compute $d^{S,k}$, $\hat{\lambda}^k$, $\hat{\mu}^k$, $\hat{z}^k$ from (27) and (23) and set $\hat{x}^{k+1} = x^k + d^{S,k}$.
6: **while** $\{j \in \mathcal{J} : x_{j0}^k + d_{j0}^{S,k} = 0\} \not\subseteq \hat{\mathcal{E}}^k$ **do**
7: Set $\hat{\mathcal{E}}^k \leftarrow \hat{\mathcal{E}}^k \cup \{j \in \mathcal{J} : x_{j0}^k + d_{j0}^{S,k} = 0\}$.
8: Recompute $d^{S,k}$, $\hat{\lambda}^k$, $\hat{\mu}^k$, $\hat{z}^k$ from (27) and (23) and set $\hat{x}^{k+1} = x^k + d^{S,k}$.
9: **end while**
10: Compute candidate penalty parameter $\rho^k = \rho_{\text{new}}(\rho^{k-1}, \hat{z}^k)$, see (37).
11: **if** (11) holds for $d = d^{S,k}$ **then**
12: Set $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}_{pr,j}^+(\mathcal{Y}_j^k, x_j^k)$ using (19) and set $d^k = d^{S,k}$.
13: Set $\mu^{k+1} = \hat{\mu}^k$ and go to Step 33.
14: **end if**
15: Compute s^k, $\hat{\lambda}^k$, $\hat{\mu}^k$, $\hat{z}^k$ from (35) and (23) and set $\hat{x}^{k+1} = x^k + d^{S,k} + s^k$.
16: **if** (11) holds for $d = d^{S,k}$ and $\{j \in \mathcal{J} : x_{j0}^k + d_{j0}^{S,k} + s^k = 0\} \subseteq \hat{\mathcal{E}}^k$ **then**
17: Set $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}_{pr,j}^+(\mathcal{Y}_j^k, x_j^k)$ and $d^k = d^{S,k}$.
18: Set $\mu^{k+1} = \hat{\mu}^k$ and go to Step 33.
19: **end if**
20: Set $\mathcal{Y}_j^{k,0} \leftarrow \mathcal{Y}_j^k$.
21: **for** $l = 0, 1, 2, \ldots$ **do**
22: Compute $d^{k,l}$, $\hat{\lambda}^k$, $\hat{\mu}^k$, $\hat{z}^k$ from (18) and (23) and set $\hat{x}^{k+1} = x^k + d^{k,l}$.
23: Compute candidate penalty parameter $\rho^k = \rho_{\text{new}}(\rho^{k-1}, \hat{z}^k)$.
24: **if** (11) holds for $d = d^{k,l}$ **then**
25: Set $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}_{pr,j}^+(\mathcal{Y}_j^{k,l}, x_j^k)$ and $d^k = d^{k,l}$.
26: Go to Step 32.
27: **end if**
28: Set $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}_{pr,j}^+(\mathcal{Y}_j^{k,l}, \hat{x}_j^{k+1})$, see (19).
29: Compute $\check{z}^k = c + A^T \hat{\lambda}^k$.
30: Update $\mathcal{Y}_j^{k,l+1} \leftarrow \mathcal{Y}_{du,j}^+(\mathcal{Y}_j^{k,l+1}, \hat{x}_j^{k+1}, \check{z}_j^k)$, see (26).
31: **end for**
32: Compute μ^{k+1} from (33).
33: Compute $\check{z}^k = c + A^T \hat{\lambda}^k$ and update $\mathcal{Y}_j^{k+1} \leftarrow \mathcal{Y}_{du,j}^+(\mathcal{Y}_j^{k+1}, x_j^k, \check{z}_j^k)$.
34: Set $x^{k+1} \leftarrow \hat{x}^{k+1}$.
35: If $(x^{k+1}, \hat{\lambda}^k, \hat{z}^k)$ satisfy (13), **stop**.
36: **end for**

3.7. Complete algorithm. The complete method is stated in Algorithm 3. To keep the notation concise, we omit "for all $j \in \mathcal{J}$" whenever the index j is used. We assume that all QPs in the algorithm are solved exactly.

Each iteration begins with the computation of the fast NLP-SQP step where an inner loop repeatedly adds cones to $\hat{\mathcal{E}}^k$ until (29) holds. If the step achieves a sufficient decrease in the penalty function, the trial point is accepted. Otherwise, the second-order correction for the NLP-SQP step is computed and accepted if it yields a sufficient decrease for the NLP-SQP step. Note that the second-order correction step is discarded if it does not satisfy (29) since otherwise finite identification of $\mathcal{E}(x^*)$ cannot be guaranteed. If none of the NLP-SQP steps was acceptable, the algorithm proceeds with an inner loop in which hyperplanes cutting off the current trial point are repeatedly added until the penalty function is sufficiently decreased. No matter which step is taken, both x_j^k and $\check{z}_j^k$ are added to $\mathcal{Y}_j^k$ according to the update rules (19) and (26) and the multiplier μ^k for the nonlinear constraints is updated.

In most cases, a new QP is obtained by adding only a few constraints to the most recently solved QP, and a hot-started QP solver will typically compute the new solution quickly. For example, in each inner iteration in Steps 6–9, hyperplanes for the polyhedral outer approximation for cones augmenting $\hat{\mathcal{E}}^k$ are added to QP (27). Similarly, each inner iteration in Steps 21–31 adds one cutting plane for a violated cone constraint. In Steps 5 and 15, some constraints are removed compared to the most recently solved QP, but also this structure could be utilized.

The algorithm might terminate because one of QPs solved for the step computation is infeasible. Since the feasible regions of the QP are outer approximations of the SOCP (1), this proves that the SOCP instance is infeasible; see also Remark 11.

4. Convergence analysis.

4.1. Global convergence. In this section, we prove that, under a standard regularity assumption, all limit points of the sequence of iterates are optimal solutions of the SOCP, if the algorithm does not terminate with an optimal solution in Step 35. We also explore what happens when the SOCP is infeasible.

We make the following assumption throughout this section.

ASSUMPTION 3. *The set X defined in* (9) *is bounded.*

Since $x^0 \in X$ by the initialization of Algorithm 3 and any step satisfies (21b), we have $x^k \in X$ for all k. Similarly, (24c) and (14) imply that

$$x_{j0}^k \geq 0 \text{ for all } k \geq 0 \text{ and } j \in \mathcal{J}. \tag{38}$$

We start the analysis with some technical results that quantify the decrease in the penalty function model.

LEMMA 4. *Consider an iteration k and let d^k be computed in Step 5 or Step 22 in Algorithm 3. Further let $\rho^k > \rho_{\min}^k$, where $\rho_{\min}^k = \|\hat{z}_{\mathcal{J},0}^k\|_\infty$ with $\hat{z}^k$ defined in* (23)*. Then the following statements are true.*

(i) We have

$$m^k(x^k + d^k; \rho^k) - m^k(x^k; \rho^k) \leq -(d^k)^T H^k d^k - (\rho^k - \rho_{\min}^k) \sum_{j \in \mathcal{J}} [r_j(x_j^k)]^+ \leq 0.$$

(ii) If x^k is not an optimal solution of the SOCP, then

$$m^k(x^k + d^k; \rho^k) - m^k(x^k; \rho^k) < 0. \tag{39}$$

Proof. Proof of (i): Consider any $j \in \mathcal{D}(x^k)$. Because d^k is a solution of (18) or (27), there exist $\hat{\lambda}^k$ and $\hat{z}^k$ so that the optimality conditions (24) hold. Let $j \in \mathcal{J}$.

Since $\hat{z}_j^k \in \mathcal{C}^\circ(\hat{\mathcal{Y}}_j^k)$, the definition (17) implies that

$$\hat{z}_j^k = -\sum_{l=1}^{m_j^k} \hat{\sigma}_{l,j}^k \nabla r_j(y_{j,l}^k) + \hat{\eta}_j^k e_{j0},$$

where $\hat{\mathcal{Y}}_j^k = \left\{ y_{j,1}^k, \ldots, y_{j,m_j^k}^k \right\}$ and $\hat{\sigma}_{l,j}^k, \hat{\eta}_j^k \in \mathbb{R}_+$.

Using (15) we have $\hat{z}_{j0}^k = \sum_{l=1}^{m_j^k} \hat{\sigma}_{l,j}^k + \hat{\eta}_j^k \geq \sum_{l=1}^{m_j^k} \hat{\sigma}_{l,j}^k$. Together with $(x_j^k + d_j^k)^T \hat{z}_j^k = 0$ from (24c) and $(\bar{x}_j^k)^T \bar{y}_{l,j}^k \leq \|\bar{x}_j^k\| \cdot \|\bar{y}_{l,j}^k\|$ this overall yields

$$\begin{aligned} -(d_j^k)^T \hat{z}_j^k = (x_j^k)^T \hat{z}_j^k &= x_{j0}^k \hat{z}_{j0}^k - \sum_{l=1}^{m_j^k} \hat{\sigma}_{l,j}^k (\bar{x}_j^k)^T \frac{\bar{y}_{l,j}^k}{\|\bar{y}_{l,j}^k\|} \geq x_{j0}^k \hat{z}_{j0}^k - \sum_{l=1}^{m_j^k} \hat{\sigma}_{l,j}^k \|\bar{x}_j^k\| \\ &\geq x_{j0}^k \hat{z}_{j0}^k - \hat{z}_{j0}^k \|\bar{x}_j^k\| = -z_{j0}^k r_j(x_j^k) \geq -z_{j0}^k [r_j(x_j^k)]^+. \end{aligned}$$

Further, we have from (24b) that $0 = (Ax^k + Ad^k - b)^T \hat{\lambda}^k$ and therefore $(d^k)^T A^T \hat{\lambda}^k = -(Ax^k - b)^T \hat{\lambda}^k \geq 0$ since $\hat{\lambda}^k \geq 0$ and $x^k \in X$.

Using these inequalities and (24a), the choice of $\rho_{\min}^k$ yields

$$\begin{aligned} 0 &= (d^k)^T \left(c + H^k d^k + A^T \hat{\lambda}^k - \hat{z}^k \right) \\ &\geq c^T d^k + (d^k)^T H^k d^k - \sum_{j \in \mathcal{J}} \hat{z}_{j0}^k [r_j(x_j^k)]^+ \\ &\geq c^T d^k + (d^k)^T H^k d^k - \rho_{\min}^k \sum_{j \in \mathcal{J}} [r_j(x_j^k)]^+. \end{aligned}$$

Finally, combining this with (10) and (18c) or (27c), respectively, we obtain

$$\begin{aligned} m^k(x^k + d^k; \rho^k) - m^k(x^k; \rho^k) &= c^T d^k - \rho^k \sum_{j \in \mathcal{D}(x^k)} [r_j(x_j^k)]^+ \\ &= c^T d^k - \rho^k \sum_{j \in \mathcal{J}} [r_j(x_j^k)]^+ \\ &\leq -(d^k)^T H^k d^k - (\rho^k - \rho_{\min}^k) \sum_{j \in \mathcal{J}} [r_j(x_j^k)]^+. \end{aligned}$$

For the second equality, we used that $r_j(x_j^k) = 0 - x_{j0}^k \leq 0$ for $j \notin \mathcal{D}(x^k)$ by (38) and the definition of $\mathcal{D}(x^k)$. Since H^k is positive semi-definite, $\rho^k > \rho_{\min}^k$, and $[r_j(x_j^k)]^+ \geq 0$, the right-hand side must be non-positive.

Proof of (ii): Suppose $x^k \in X$ is not an optimal solution for the SOCP. If x^k is not feasible for the SOCP, x^k must violate a conic constraint and we have $[r_j(x_j^k)]^+ > 0$ for some $j \in \mathcal{J}$. Since H^k is positive semidefinite and $\rho^k - \rho_{\min}^k > 0$, part (i) yields (39).

It remains to consider the case when x^k is feasible for the SOCP, i.e., $[r_j(x_j^k)]^+ = 0$ for all j. To derive a contradiction, suppose that (39) does not hold. Then part (i) yields

$$\begin{aligned} 0 &= m^k(x^k + d^k; \rho^k) - m^k(x^k; \rho^k) \\ &= -(d^k)^T H^k d^k - (\rho^k - \rho_{\min}^k) \sum_{j \in \mathcal{J}} [r_j(x_j^k)]^+ = -(d^k)^T H^k d^k \leq 0. \end{aligned}$$

Because H^k is positive semi-definite, this implies $H^k d^k = 0$. Further, since also

$$0 = m^k(x^k + d^k; \rho^k) - m^k(x^k; \rho^k) \overset{(10)}{=} c^T d^k - \rho^k \sum_{j \in \mathcal{D}(x^k)} [r_j(x_j^k)]^+ = c^T d^k,$$

the optimal objective value of (18) or (27), respectively, is zero. At the same time, choosing $d^k = 0$ is also feasible for (18) or (27) and yields the same objective value. Therefore, also $d^k = 0$ is an optimal solution of (18) or (27) and the optimality conditions (24) hold for some multipliers. Because $\mathcal{C}_j^\circ(\hat{\mathcal{Y}}_k^k) \subseteq \mathcal{K}_j$, the same multipliers and $d^k = 0$ show that the optimality conditions of the SOCP (13) also hold. So, x^k is an optimal solution for the SOCP, contradicting the assumption. ∎

The following lemma shows that the algorithm is well-defined and will not stay in an infinite loop in Steps 21–31.

LEMMA 5. *Consider an iteration k and let d^k be computed in Step 5 or Step 22 in Algorithm 3. Suppose that x^k is not an optimal solution of the SOCP. Then*

$$\varphi(x^k + d^{k,l}; \rho^k) - \varphi(x^k; \rho^k) \le c_{dec}\Big(m^k(x^k + d^{k,l}; \rho^k) - m^k(x^k; \rho^k)\Big) \tag{40}$$

after a finite number of iterations in the inner loop in Steps 21–31.

Proof. Suppose the claim is not true and let $\{d^{k,l}\}_{l=0}^\infty$ be the infinite sequence of trial steps generated in the loop in Steps 21–31 for which the stopping condition in Step 24 is never satisfied, and let $d^{k,\infty}$ be a limit point of $\{d^{k,l}\}_{l=0}^\infty$. We will first show that

$$[r_j(x_j^k + d_j^{k,\infty})]^+ = 0 \text{ for all } j \in \mathcal{J}. \tag{41}$$

Let us first consider the case when $\bar{x}_j^k + \bar{d}_j^{k,\infty} = 0$ for some $j \in \mathcal{J}$. Then $r_j(x_j^k + d_j^{k,\infty}) = \|\bar{x}_j^k + \bar{d}_j^{k,\infty}\| - (x_{j0}^k + d_{j0}^{k,\infty}) = -(x_{j0}^k + d_{j0}^{k,\infty}) \le 0$ and (41) holds.

Now consider the case that $\bar{x}_j^k + \bar{d}_j^{k,\infty} \neq 0$ for $j \in \mathcal{J}$. Since $d^{k,\infty}$ is a limit point of $\{d^{k,l}\}_{l=0}^\infty$, there exists a subsequence $\{d^{k,l_t}\}_{t=0}^\infty$ that converges to $d^{k,\infty}$. We may assume without loss of generality that $\bar{x}_j^k + \bar{d}_j^{k,l_t} \neq 0$ for all t. Then, for any t, by Step 30, $x_j^k + d_j^{k,l_t} \in \mathcal{Y}_j^{k,l_{t+1}}$. In the inner iteration l_{t+1}, the trial step $d_j^{k,l_{t+1}}$ is computed from (18) and satisfies $x_j^k + d_j^{k,l_{t+1}} \in \mathcal{C}_j(\mathcal{Y}_j^{k,l_t})$, which by definition (14) implies

$$\nabla r_j(x_j^k + d_j^{k,l_t})^T (x_j^k + d_j^{k,l_{t+1}}) \le 0.$$

Taking the limit $t \to \infty$ and using the fact that $\nabla r_j(v_j)^T v_j = r_j(v_j)$ for any $v_j \in \mathcal{K}_j$ yields

$$r_j(x_j^k + d_j^{k,\infty}) = \nabla r_j(x_j^k + d_j^{k,\infty})^T (x_j^k + d_j^{k,\infty}) \le 0,$$

proving (41). In turn (41) implies that the ratio

$$\frac{\varphi(x^k + d^{k,l}; \rho^k) - \varphi(x^k; \rho^k)}{m^k(x^k + d^{k,l}; \rho^k) - m^k(x^k; \rho^k)} = \frac{c^T d^{k,l} + \rho^k([r_j(x_j^k + d_j^{k,l})]^+ - [r_j(x_j^k)]^+)}{c^T d^{k,l} - \rho^k [r_j(x_j^k)]^+}$$

converges to 1. Note that the ratio is well-defined since $m^k(x^k + d^{k,l}; \rho^k) - m^k(x^k; \rho^k) < 0$ due to Lemma 5(ii). It then follows that (40) is true for sufficiently large l. ∎

LEMMA 6. *Suppose that there exists $\rho^\infty > 0$ so that $\rho^k = \rho^\infty > 0$ for all large k. Then any limit point of $\{x^k\}_{k=0}^\infty$ is an optimal solution of the SOCP (1).*

Proof. From (11) and the updates in the algorithm, we have that

$$\varphi(x^{k+1};\rho^\infty) - \varphi(x^{K_\rho};\rho^\infty) = \sum_{t=K_\rho}^{k} \Big(\varphi(x^{t+1};\rho^\infty) - \varphi(x^t;\rho^\infty)\Big)$$
$$\le c_{\text{dec}} \sum_{t=K_\rho}^{k} \Big(m^t(x^t + d^t;\rho^\infty) - m^t(x^t;\rho^\infty)\Big)$$

for $k \ge K_\rho$. Since the SOCP cannot be unbounded below by Assumption 3, the left-hand side is bounded below as $k \to \infty$. Lemma 4(i) shows that all summands are non-positive and we obtain

$$\lim_{k\to\infty} \Big(m^k(x^k + d^k;\rho^\infty) - m^k(x^k;\rho^\infty)\Big) = 0. \tag{42}$$

Using Lemma 4(i), we also have

$$\lim_{k\to\infty} \left((d^k)^T H^k d^k + (\rho^\infty - \rho^k_{\min}) \sum_{j\in\mathcal{J}} [r_j(x^k_j)]^+\right) = 0.$$

Since H^k is positive semi-definite and $\rho^\infty - \rho^k_{\min} \ge \rho^\infty - \rho^\infty_{\min} > 0$, this implies that $[r_j(x^k_j)]^+ \to 0$ for all $j \in \mathcal{J}$, i.e., all limit points of $\{x^k\}_{k=0}^\infty$ are feasible. This also yields $\lim_{k\to\infty}(d^k)^T H^k d^k = 0$, and since H^k is positive semi-definite and uniformly bounded due to (34), we have

$$\lim_{k\to\infty} H^k d^k = 0. \tag{43}$$

Using (42) together with (10) and $[r_j(x^k_j)]^+ \to 0$, we obtain

$$0 = \lim_{k\to\infty} \left(c^T d^k - \rho^\infty \sum_{j\in\mathcal{D}(x^k)} [r_j(x^k_j)]^+\right) = \lim_{k\to\infty} c^T d^k. \tag{44}$$

Now let x^* be a limit point of $\{x^k\}_{k=0}^\infty$. Since X is bounded, d^k is bounded, and consequently there exists a subsequence $\{k_t\}_{t=0}^\infty$ of iterates so that x^{k_t} and d^{k_t} converge to x^* and d^∞, respectively, for some limit point d^∞ of d^k. Define $g^{k_t} = H^{k_t} d^{k_t}$ for all t. Then, looking at the QP optimality conditions (24), we see that d^{k_t} is also an optimal solution of the linear optimization problem

$$\begin{aligned} \min_{d\in\mathbb{R}^n}\ & (c + g^{k_t})^T d \\ \text{s.t. } & A(x^{k_t} + d) \le b, \\ & x^{k_t}_j + d_j \in \mathcal{C}_j(\hat{\mathcal{Y}}^{k_t}_j), \qquad j \in \mathcal{J}. \end{aligned} \tag{45}$$

Now suppose, for the purpose of deriving a contradiction, that x^* is not an optimal solution of the SOCP. Since we showed above that x^* is feasible, there then exists a step $\tilde{d}^* \in \mathbb{R}^n$ so that $\tilde{x} = x^* + \tilde{d}^*$ is feasible for (1) and $c^T\tilde{d}^* < 0$. Then, because $\mathcal{K}_j \subseteq \mathcal{C}_j(\hat{\mathcal{Y}}^{k_t}_j)$, for each t, $\tilde{d}^{k_t} = x^* - x^{k_t} + \tilde{d}^*$ is feasible for (45), and because d^{k_t} is an optimal solution of (45), we have $(c + g^{k_t})^T d^{k_t} \le (c + g^{k_t})^T \tilde{d}^{k_t}$. Taking the limit $t \to \infty$, we obtain $c^T d^\infty \le c^T \tilde{d}^* < 0$, where we used $\lim_{t\to\infty} g^{k_t} = \lim_{t\to\infty} H^{k_t} d^{k_t} = 0$, due to the definition of g^{k_t} and (43). However, this contradicts (44). Therefore, x^* must be a solution of the SOCP. ∎

For later reference, we highlight the limit (43) established in the above proof.

LEMMA 7. *Suppose that there exists $\rho^\infty > 0$ so that $\rho^k = \rho^\infty > 0$ for all large k. Then $\lim_{k\to\infty} H^k d^k = 0$.*

We are now ready to prove that the algorithm is globally convergent under the following standard regularity assumption.

ASSUMPTION 4. *The SOCP is feasible and Slater's constraint qualification holds, i.e., there exists a feasible point $\tilde{x} \in \mathbb{R}^n$ and $\epsilon > 0$ so that $\tilde{x} + v$ is feasible for any $v \in \mathbb{R}^n$ with $\|v\| \le \epsilon$.*

This assumption implies that the multiplier estimates are bounded.

LEMMA 8. *Suppose that Assumption 4 holds. Then $\{\hat{z}^k\}$ is bounded.*

Proof. Let $\tilde{x}$ and ϵ be the quantities from Assumption 4. Note that the QP corresponding to the optimality conditions (24) is

$$\begin{aligned} &\min_{d\in\mathbb{R}^n} \ c^T d + \tfrac{1}{2} d^T H^k d \\ &\quad \text{s.t. } A(x^k + d) \le b,\ x_j^k + d_j \in \mathcal{C}_j(\hat{\mathcal{Y}}_j^k),\ j \in \mathcal{J}. \end{aligned}$$

Since $x^{k+1} = x^k + d^k$ when d^k is the step accepted by the algorithm, it follows that x^{k+1} is an optimal solution of the QP

$$\begin{aligned} O_{\text{primal}} = &\min_{x\in\mathbb{R}^n} \ (c^T - H^k x^k)x + \tfrac{1}{2} x^T H^k x \\ &\quad \text{s.t. } Ax^{k+1} \le b,\ x_j^{k+1} \in \mathcal{C}_j(\hat{\mathcal{Y}}_j^k),\ j \in \mathcal{J}, \end{aligned}$$

the Lagrangian dual of which is

$$O_{\text{dual}} = \max_{x,z\in\mathbb{R}^n,\lambda\in\mathbb{R}^m} \ -b^T\lambda - \tfrac{1}{2} x^T H^k x \tag{46a}$$

$$\text{s.t. } c - H^k x^k + H^k x + A^T\lambda - z = 0, \tag{46b}$$

$$z \in \mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k),\ j \in \mathcal{J}, \quad \lambda \ge 0. \tag{46c}$$

By (24), $(x^{k+1}, \hat{\lambda}^k, \hat{z}^k)$ is a primal-dual optimal solution of these QPs.

Define $v = -\epsilon \frac{\hat{z}^k}{\|\hat{z}^k\|}$. Then $\|v\| \le \epsilon$, and Assumption 4 implies that $\tilde{x} + v \in \mathcal{K}_j \subseteq \mathcal{C}_j(\hat{\mathcal{Y}}_j^k)$. Since $\hat{z}^k \in \mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k)$, we have with (46b) that

$$0 \le (\tilde{x} + v)^T \hat{z}^k = v^T \hat{z}^k + \tilde{x}^T(c - H^k\tilde{x} + H^k x^{k+1} + A^T\hat{\lambda}^k). \tag{47}$$

Since H^k is positive definite, it is

$$0 \le (\tilde{x} - x^{k+1})^T H^k (\tilde{x} - x^{k+1}) = \tilde{x}^T H^k \tilde{x} - 2\tilde{x}^T H^k x^{k+1} + (x^{k+1})^T H^k x^{k+1}. \tag{48}$$

Furthermore, Slater's condition implies strong duality, that is

$$\begin{aligned} b^T\hat{\lambda}^k + \frac{1}{2}(x^{k+1})^T H^k x^{k+1} &= -O_{\text{dual}} = -O_{\text{primal}} \\ &= -(c - H^k x^k)^T x^{k+1} - \tfrac{1}{2}(x^{k+1})^T H^k x^{k+1}. \end{aligned} \tag{49}$$

Finally, since $\tilde{x}$ is feasible for the SOCP, (1b) and $\hat{\lambda}^k \ge 0$ imply $\tilde{x}^T A^T \hat{\lambda}^k \le b^T\hat{\lambda}^k$. Subtracting $v^T\hat{z}^k$ on both sides of (47), this, together with (48) and (49), yields

$$\begin{aligned} \epsilon\|\hat{z}^k\| &\le \tilde{x}^T c - \frac{1}{2}\tilde{x}^T H^k \tilde{x} + \frac{1}{2}(x^{k+1})^T H^k x^{k+1} + b^T\hat{\lambda}^k \\ &= \tilde{x}^T c - \frac{1}{2}\tilde{x}^T H^k \tilde{x} - c^T x^{k+1} - \frac{1}{2}(x^{k+1})^T H^k x^{k+1}. \end{aligned}$$

The first two terms are independent of k, and since X is bounded by Assumption 4 and H^k is uniformly bounded by (34), we can conclude that $\hat{z}^k$ is uniformly bounded. ∎

It is easy to see that the penalty parameter update rule (37) and Lemma 8 imply the following result.

LEMMA 9. *Suppose Assumption 4 holds. Then there exists ρ^∞ and K_ρ so that $\rho^k = \rho^\infty > \rho^\infty_{\min}$, where $\rho^\infty_{\min} \geq \rho^k_{\min} = \|z^k_{\mathcal{J},0}\|_\infty$ for all $k \geq K_\rho$.*

We can now state the main convergence theorem of this section.

THEOREM 10. *Suppose that Assumptions 3 and 4 hold. Then Algorithm 3 either terminates in Step 35 with an optimal solution, or it generates an infinite sequence of iterates $\{(x^{k+1}, \hat{\lambda}^k, \hat{z}^k)\}_{k=0}^\infty$, each limit point of which is a primal-dual solution of the SOCP (1).*

Proof. Let $\{(x^{k_t+1}, \hat{\lambda}^{k_t}, \hat{z}^{k_t})\}$ be a subsequence converging to a limit point (x^*, λ^*, z^*). No matter whether an iterate is computed from the optimal solution of (18), (27), or (35), the iterates satisfy the optimality conditions (24). In particular, from (24c) we have for any $j \in \mathcal{J}$ that $\hat{z}^{k_t}_j \in \mathcal{C}^\circ_j(\hat{\mathcal{Y}}^{k_t}_j) \subseteq \mathcal{K}_j$ and $(x^{k_t+1}_j)^T \hat{z}^{k_t}_j = 0$. In the limit, we obtain $z^*_j \in \mathcal{K}_j$ (since $\mathcal{K}_j$ is closed) and $(x^*_j)^T z^*_j = 0$. Lemma 9 yields that $\rho^k = \rho^\infty$ for all large k, and so Lemma 6 implies that x^* is feasible, i.e., $x^*_j \in \mathcal{K}_j$. Therefore, (13c) holds. Using Lemma 7 we can take the limit in (24a) and (24b) and deduce also the remaining SOCP optimality conditions (13a) and (13b) hold at the limit point.

REMARK 11. *In case the SOCP is infeasible, we have two possible outcomes. Either, Algorithm 3 terminates in some iterations because one of the QPs is infeasible, or $\lim_{k\to\infty} \rho^k = \infty$ (reverse conclusion of Lemma 6).*

4.2. Identification of extremal-active cones. We can only expect fast local convergence under some non-degeneracy assumptions. Throughout this section, we assume that Assumption 2 holds. Under this assumption, (x^*, λ^*, z^*) is the unique optimal solution [1, Theorem 22], and Theorem 10 then implies that

$$\lim_{k\to\infty} (x^{k+1}, \hat{\lambda}^k, \hat{z}^k) = (x^*, \lambda^*, z^*).$$

First, we prove a technical result that describes elements in $\mathcal{C}^\circ_j(\mathcal{Y}_j)$ in a compact manner. For this characterization to hold, condition (20) for the initialization $\mathcal{Y}^0_j$ of the set of hyperplane-generating points is crucial.

LEMMA 12. *Let $y_j \in \mathbb{R}^{n_j}$ with $\bar{y}_j \neq 0$ and $y_{j0} \geq 0$. Further, let $\Phi_j(z_j, y_j) := z_{j0} - \|\bar{z}_j + \bar{y}_j\|_1 - \|\bar{y}_j\|$. Then the following statements hold for $z_j, y_j \in \mathbb{R}^{n_j}$:*

(i) $z_j \in \mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\})$ if $\Phi_j(z_j, y_j) \geq 0$.

(ii) $z_j \in \operatorname{int}(\mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\}))$ if $\Phi_j(z_j, y_j) > 0$.

Proof. For (i): Suppose $\Phi_j(z_j, y_j) \geq 0$, then

$$z_{j0} \geq \|\bar{z}_j + \bar{y}_j\|_1 + \|\bar{y}_j\|.$$

Define $\bar{s}_j = \bar{z}_j + \bar{y}_j$ and choose $\sigma^+_j \in \mathbb{R}^{n_j-1}_+$ and $\sigma^-_j \in \mathbb{R}^{n_j-1}_+$ so that $\bar{s}_j = \sigma^+_j - \sigma^-_j$

and $|s_{ji}| = \sigma^+_{ji} + \sigma^-_{ji}$ for all $i = 1, \dots, n_j - 1$. Then we have

$$z_{j0} = \sum_{i=1}^{n_j-1} \sigma^+_{ji} + \sum_{i=1}^{n_j-1} \sigma^-_{ji} + \sigma_j + \eta_j$$

$$\bar z_j = \bar s_j - \bar y_j = \sigma^+_j - \sigma^-_j - \sigma_j \frac{\bar y_j}{\|\bar y_j\|}$$

with $\sigma_j = \|\bar y_j\|$ and some $\eta_j \in \mathbb{R}_+$. Using (15), this can be rewritten as

$$z_j = -\sum_{i=1}^{n_j-1} \sigma^+_{ji} \nabla r_j(-e_{ji}) - \sum_{i=1}^{n_j-1} \sigma^-_{ji} \nabla r_j(e_{ji}) - \sigma_j \nabla r_j(y_j) + \eta_j e_{j0}.$$

By the definition of $\mathcal{C}^\circ_j$ in (17), this implies that $z_j \in \mathcal{C}^\circ_j(\hat{\mathcal{Y}}^0_j \cup \{y_j\})$ where $\hat{\mathcal{Y}}^0_j$ is defined in (20). Since $\hat{\mathcal{Y}}^0_j \subseteq \mathcal{Y}^0_j$ from (20), we have $\mathcal{C}^\circ_j(\hat{\mathcal{Y}}^0_j \cup \{y_j\}) \subseteq \mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\})$, and the claim follows.

For (ii): Suppose $\Phi_j(z_j, y_j) > 0$. Because Φ_j is a continuous function, there exists a neighborhood $N_\epsilon(z_j)$ around z_j so that $\Phi_j(\hat z_j, y_j) > 0$ for all $\hat z_j \in N_\epsilon(z_j)$. From part (i) we then have $N_\epsilon(z_j) \subseteq \mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\})$, and consequently $z_j \in \mathrm{int}(\mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\}))$.□

THEOREM 13. *For all k sufficiently large, we have $\mathcal{E}(x^k) = \mathcal{E}(x^*)$.*

Proof. Choose $j \notin \mathcal{E}(x^*)$, then $x^*_j \neq 0$. Because $x^k_j \to x^*_j$, it is $x^k_j \neq 0$ or, equivalently, $j \notin \mathcal{E}(x^k)$ for k sufficiently large. For the remainder of this proof we consider $j \in \mathcal{E}(x^*)$ and show that $j \in \mathcal{E}(x^k)$ for large k. Note that strict complementarity in Assumption 2 implies that $z^*_j \in \mathrm{int}(\mathcal{K}_j)$, i.e., $r_j(z^*_j) < 0$, and consequently $z^*_{j0} > 0$.

First consider the iterations in which fast NLP-SQP steps are accepted in Steps 11 or 16. For the purpose of deriving a contradiction, suppose there exists an infinite subsequence so that $x^{k_t+1} = x^{k_t} + d^{S,k_t}$ or $x^{k_t+1} = x^{k_t} + d^{S,k_t} + s^{k_t}$ and $j \notin \hat{\mathcal{E}}^{k_t}$. Then $j \notin \hat{\mathcal{E}}^{k_t}$ implies $x^{k_t+1}_{j0} > 0$ (according to the termination condition in the while loop in Step 6). We also have $\hat{\mathcal{Y}}^{k_t}_j = \{\breve x^{k_t}_j\}$ where $\breve x^{k_t}_j = x^{k_t}_j$ from (28) or $\breve x^{k_t}_j = x^{k_t}_j + d^{S,k_t}_j$ from (36). Condition (24c) yields $z^{k_t}_j \in \mathcal{C}^\circ_j(\{\breve x^{k_t}_j\})$, so by (17) it is $z^{k_t}_j = -\sigma_j \nabla r_j(\breve x^{k_t}_j) + \eta_j e_{j0}$ for some $\sigma_j, \eta_j \geq 0$, as well as $x^{k_t+1}_j \in \mathcal{C}_j(\{\breve x^{k_t}_j\})$, which by (14) implies $\nabla r_j(\breve x^{k_t}_j)^T x^{k_t+1}_j \leq 0$. Then complementarity yields

$$0 = (z^{k_t}_j)^T x^{k_t+1}_j = -\sigma_j \nabla r_j(\breve x^{k_t}_j)^T x^{k_t+1}_j + \eta_j x^{k_t+1}_{j0} \geq \eta_j x^{k_t+1}_{j0}.$$

Since $x^{k_t+1}_{j0} > 0$ and $\eta_j \geq 0$, we must have $\eta_j = 0$, and consequently $z^{k_t}_j = -\sigma_j \nabla r_j(\breve x^{k_t}_j)$. It is easy to see that $r_j(-\sigma_j \nabla r_j(\breve x^{k_t}_j)) = 0$. Since $z^{k_t}_j \to z^*_j$, continuity of r_j yields $r_j(z^*_j) = 0$, in contradiction to $z^*_j \in \mathrm{int}(\mathcal{K}_j)$. We thus showed that $j \in \hat{\mathcal{E}}^k$ for all large iterations k in which the NLP-SQP step was accepted, and consequently (28) and (36) yield $\hat{\mathcal{Y}}^k_j = \mathcal{Y}^k_j$ for such k.

In all other iterations (22) holds, and overall we obtain

$$\mathcal{Y}^0_j \subseteq \mathcal{Y}^k_j \subseteq \hat{\mathcal{Y}}^k_j \text{ for all sufficiently large } k. \tag{50}$$

Let us first consider the case when $\bar z^*_j = 0$. Then $\|\bar z^*_j\| - z^*_{j0} = r_j(z^*_j) < 0$ yields $z^*_{j0} > 0$. To apply Lemma 12 choose any $i \in \{1, \dots, n_j - 1\}$ and let $y_j = e_{ji}$. Then $\|y_j\|_1 = \|y_j\| = 1$ and $\Phi_j(z^*_j, y_j) = z^*_{j0} > 0$. Since $\hat z^k_j \to z^*_j$ and Φ_j is continuous, $\Phi_j(\hat z^k_j, y_j) > 0$ for sufficiently large k, and by Lemma 12, $\hat z^k_j \in \mathrm{int}(\mathcal{C}^\circ_j(\mathcal{Y}^0_j \cup \{y_j\}))$. Since

$y_j \in \mathcal{Y}_j^0$ and (50) holds, we also have $\hat{z}_j^k \in \operatorname{int}(\mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^k))$. General conic complementarity in (24c) then implies that $x_j^{k+1} = x_j^k + d_j^k = 0$ for all large k, or equivalently, $j \in \mathcal{E}(x^k)$ for k sufficiently large, as desired.

Now consider the case $\bar{z}_j^* \neq 0$. For the purpose of deriving a contradiction, suppose there exists a subsequence $\{x^{k_t}\}_{t=0}^\infty$ so that $j \notin \mathcal{E}(x^{k_t})$, i.e., $x^{k_t} \neq 0$, for all t. Because $\check{z}_j^k \to z_j^*$, $\bar{z}_j^* \neq 0$, and $r_j(z_j^*) < 0$, we may assume without loss of generality that $r_j(\check{z}_j^{k_t}) < 0$ and $\bar{\check{z}}_j^{k_t} \neq 0$ for all t. Using this and $x_j^{k_t} \neq 0$, we see that the update rule (26) in Step 33 adds $-\check{z}_j^{k_t}$ to $\mathcal{Y}_j^{k_t+1}$. With (50), we have

$$(51) \qquad -\check{z}_j^{k_t} \in \mathcal{Y}_j^{k_t+1} \subseteq \mathcal{Y}_j^{k_{t+1}} \subseteq \hat{\mathcal{Y}}_j^{k_{t+1}} \text{ for all } t.$$

Recall the mapping Φ_j defined in Lemma 12 and note that $\Phi_j(z_j^*, -z_j^*) = z_{j0}^* - \|\bar{z}_j^*\| = -r_j(z_j^*) > 0$. Since both $\hat{z}_j^k$ and $\check{z}_j^k$ converge to z_j^* and Φ_j is continuous, it is $\Phi_j(\hat{z}_j^{k_{t+1}-1}, -\check{z}_j^{k_t}) > 0$ for all large t, and therefore, by Lemma 12, $\hat{z}_j^{k_{t+1}-1} \in \operatorname{int}(\mathcal{C}_j^\circ(\mathcal{Y}_j^0 \cup \{-\check{z}_j^{k_t}\})) \overset{(51)}{\subseteq} \operatorname{int}(\mathcal{C}_j^\circ(\hat{\mathcal{Y}}_j^{k_{t+1}-1}))$ for all large t. Conic complementarity in (24c) then implies that $x_j^{k_{t+1}} = x_j^{k_{t+1}-1} + d_j^{k_{t+1}-1} = 0$. This is a contradiction of the definition of the subsequence $\{x^{k_t}\}_{t=0}^\infty$. ∎

Remark 14. *In the proof of Theorem 13, we saw that $\Phi_j(z_j^*, -z_j^*) > 0$ if $j \in \mathcal{E}(x^*)$ and $\bar{z}_j^* \neq 0$. Since Φ_j is continuous, this implies that there exists a neighborhood $N_\epsilon(z_j^*)$ so that $\Phi_j(z_j, -y_j) > 0$, and consequently $z_j \in \operatorname{int}(\mathcal{C}_j^\circ(\mathcal{Y}_j^0 \cup \{-y_j\}))$, for all $z_j, y_j \in N_\epsilon(z_j^*)$.*

4.3. Quadratic local convergence. As discussed in Section 2.1, since x^* is a solution of the SOCP (1), it is also a solution of the nonlinear problem (4). We now show that Algorithm 3 eventually generates steps that are identical to SQP steps for (4). Then Theorem 3 implies that the iterates converge locally at a quadratic rate.

We first need to establish that the assumptions for Theorem 3 hold.

Lemma 15. *Suppose that Assumption 2 holds for the SOCP (1). Then Assumption 1 holds for the NLP (4).*

Proof. Let λ^* and z^* be the optimal multipliers for the SOCP corresponding to x^*, satisfying (13). Assumption 2 implies that λ^* and z^* are unique [1, Theorem 22].

Let $j \in \mathcal{D}(x^*)$ and define $\mu_j^* = z_{j0}^* \geq 0$. If $0 = r_j(x_j^*) = x_{j0}^* - \|\bar{x}_j^*\|$, complementarity (13c) implies, for all $i \in \{1, \ldots n_j\}$, that $0 = x_{j0}^* z_{ji}^* + x_{ji}^* z_{j0}^* = \|\bar{x}_j^*\| z_{ji}^* + x_{ji}^* z_{j0}^*$,, or equivalently, $z_{ji}^* = -z_{j0}^* \frac{x_{ji}^*}{\|\bar{x}_j^*\|}$; see [1, Lemma 15]. Using (15), this can be written as

$$(52) \qquad z_j^* = -z_{j0}^* \nabla r_j(x_j^*) = -\mu_j^* \nabla r_j(x_j^*).$$

On the other hand, if $r_j(x_j^*) < 0$, i.e., the constraint (4c) is inactive, then $x_j^* \in \operatorname{int}(\mathcal{K}_j)$ and complementarity (13c) yields $z_j^* = 0$ (see [1, Definition 23]) and therefore $\mu_j^* = 0$. Consequently, (52) is also valid in that case. Finally, we define $\nu_j^* = z_j^*$ for all $i \in \mathcal{E}(x^*)$. With these definitions, (13a) can be restated as

$$(53) \qquad c + A^T \lambda^* + \sum_{j \in \mathcal{D}(x^*)} \mu_j^* \nabla r_j(x^*) - \nu^* = 0,$$

where $\nu^* \in \mathbb{R}^n$ is the vector with the values of ν_j^* at the components corresponding to $j \in \mathcal{E}(x^*)$ and zero otherwise. We now prove parts (i), (ii), and (iii) of Assumption 1.

Proof of (i): Let $j \in \mathcal{D}(x_j^*)$. We already established that $r_j(x_j^*) < 0$ yields $\mu_j^* = 0$. Now suppose that $r_j(x_j^*) = 0$. Then $x_j^* \in \mathrm{bd}(\mathcal{K}_j) \setminus \{0\}$. Since strict complementarity is assumed, we have $z_j^* \in \mathrm{bd}(\mathcal{K}_j) \setminus \{0\}$ (see the comment after Assumption 2), which in turn yields $z_j^* \neq 0$ and hence $\mu_j^* \neq 0$.

Proof of (ii): Since we need to prove linear independence only of those constraints that are active at x^*, we consider only those rows $A_\mathcal{A}$ of A for which (4b) is binding.

Without loss of generality suppose x^* is partitioned into four parts, $(x^*)^T = ((x_\mathcal{B}^*)^T \; (x_\mathcal{I}^*)^T \; (x_\mathcal{E}^*)^T \; (x_\mathcal{F}^*)^T)$, where $x_\mathcal{B}^*$, $x_\mathcal{I}^*$, and $x_\mathcal{E}^*$ correspond to the variables in the cones $\mathcal{B} = \{j \in \mathcal{J} : r_j(x_j^*) = 0, x_j^* \neq 0\}$, $\mathcal{I} = \{j \in \mathcal{J} : r_j(x_j^*) < 0\}$, and $\mathcal{E} = \mathcal{E}(x^*)$, respectively, and $x_\mathcal{F}^*$ includes all components of x^* that are not in any of the cones. Further suppose that $(x_\mathcal{B}^*)^T = ((x_1^*)^T \; \ldots (x_{p_\mathcal{B}}^*)^T)$, where $\mathcal{B} = \{1, \ldots, p_\mathcal{B}\}$, and that $A_\mathcal{A}$ is partitioned in the same way.

Primal non-degeneracy of the SOCP implies all that matrices of the form

$$\begin{pmatrix} [A_\mathcal{A}]_1 & \cdots & [A_\mathcal{A}]_{p_\mathcal{B}} & [A_\mathcal{A}]_\mathcal{I} & [A_\mathcal{A}]_\mathcal{E} & [A_\mathcal{A}]_\mathcal{F} \\ \alpha_1 \nabla r_1(x_1^*)^T & \cdots & \alpha_{p_\mathcal{B}} \nabla r_{p_\mathcal{B}}(x_{p_\mathcal{B}}^*)^T & 0^T & v^T & 0^T \end{pmatrix}$$

have linear independent rows for all scalars α_i and vectors v, not all zero [1, Eq. (50)]. This implies that the rows of $A_\mathcal{A}$, together with the gradient of any one of the binding constraints in (4c) and (4d) are linearly independent. Because the constraint gradients, which are of the form $\nabla r_j(x_j^*)$ and e_{ij}, share no nonzero components when extended to the full space, we conclude that the gradients of all active constraints are linearly independent at x^*, i.e., the LICQ holds.

Proof of (iii): For the purpose of deriving a contradiction, suppose that there exists a direction $d \in \mathbb{R}^n \setminus \{0\}$ that lies in the null space of the constraints of (4) that are binding at x^* and for which $d^T H^* d \leq 0$.

Since d is in the null space of the binding constraints, we have $A_\mathcal{A} d = 0$, $\nabla r_j(x^*)^T d = 0$ for $j \in \mathcal{B}$, and $d_j = 0$ for all $j \in \mathcal{E}$. Premultiplying (53) by d^T gives

$$(54) \quad 0 = c^T d + (\lambda^*)^T \underbrace{A_\mathcal{A} d}_{0} + \sum_{j \in \mathcal{B}} \mu^* \underbrace{\nabla r_j(x^*)^T d}_{0} + \sum_{j \in \mathcal{I}} \underbrace{\mu^*}_{0} \nabla r_j(x^*)^T d + \underbrace{(\nu^*)^T d}_{0} = c^T d.$$

What remains to show is that d is a feasible direction for the SOCP, i.e., there exists $\beta > 0$ so that $x^* + \beta d$ is feasible for the SOCP. Because of (54), this point has the same objective value as x^* and is therefore also an optimal solution of the SOCP. This contradicts the fact that Assumption 2 implies that the optimal solution is unique [1, Theorem 22].

By the definition of H^* in Assumption 1 and the choice of d, we have

$$0 \geq d^T H^* d = \sum_{j \in \mathcal{D}(x^*)} \mu_j^* d_j^T \nabla^2 r_j(x_j^*) d_j = \sum_{j \in \mathcal{B}} \mu_j^* d_j^T \nabla^2 r_j(x_j^*) d_j.$$

Since for all $j \in \mathcal{B}$, the Hessian $\nabla^2 r_j(x_j^*)$ is positive semi-definite and $\mu_j^* > 0$ from Part (i), this yields $d_j^T \nabla^2 r_j(x_j^*) d_j = 0$ for all $j \in \mathcal{B}$.

Let $j \in \mathcal{B}$. Then from (7)

$$(55) \qquad 0 = d_j^T \nabla^2 r_j(x^*) d_j = \frac{\|\bar{d}_j\|^2 \|\bar{x}_j^*\|^2 - (\bar{d}_j^T \bar{x}_j^*)^2}{\|\bar{x}_j^*\|^3}.$$

The definition of $\mathcal{B}$ implies $r_j(x_j^*) = 0$ and so $x_{j0}^* = \|\bar{x}_j^*\|$. Since d_j is in the null space of $\nabla r_j(x_j^*)$, we have $0 = \nabla r_j(x_j^*)^T d_j = -d_{j0} + \frac{\bar{d}_j^T \bar{x}_j}{\|\bar{x}_j^*\|}$, which in turn yields $d_{j0} x_{j0}^* = \bar{d}_j^T \bar{x}_j^*$. Finally, using these relationships together with (55) gives

$$0 = \|\bar{d}_j\|^2 \|\bar{x}_j^*\|^2 - (\bar{d}_j^T \bar{x}_j^*)^2 = \|\bar{d}_j\|^2 \, (x_{j0}^*)^2 - (d_{j0} x_{j0}^*)^2$$

and so $d_{j0}^2 = \|\bar{d}_j\|^2$. All of these facts imply that for any $\beta \in \mathbb{R}$,

$$\begin{aligned} &\|\bar{x}_j^* + \beta \bar{d}_j\|^2 - (x_{j0}^* + \beta d_{j0})^2 \\ =&\|\bar{x}_j^*\|^2 + 2\beta \bar{d}_j^T \bar{x}_j^* + \beta^2 \|\bar{d}_j\|^2 - \left((x_{j0}^*)^2 + 2\beta d_{j0} x_{j0}^* + \beta^2 d_{j0}^2\right) = 0, \end{aligned}$$

which implies $r_j(x_j^* + \beta d_j) = 0$ and therefore $x_j^* + \beta d_j \in \mathcal{K}_j$.

Further, because d lies in the null space of the active constraints, we have, for any $\beta \in \mathbb{R}$, that $x_j^* + \beta d_j = 0 \in \mathcal{K}_j$ for all $j \in \mathcal{E}(x^*)$ and $A_{\mathcal{A}}(x^* + \beta d) = b_{\mathcal{A}}$. Finally, since $r_j(x_j^*) < 0$ and hence $x_j^* \in \text{int}(\mathcal{K}_j)$ for all $j \in \mathcal{J} \setminus (\mathcal{E}(x^*) \cup \mathcal{B})$, and since x_j^* is strictly feasible for all non-binding constraints in (1b), there exists $\beta > 0$ so that $x^* + \beta d$ satisfies all constraints in (1). ∎

THEOREM 16. *Suppose that* $c_H > \|H^*\|$. *Then the primal-dual iterates* $(x^{k+1}, \hat{\lambda}^k, \hat{z}^k)$ *converge locally to* (x^*, λ^*, z^*) *at a quadratic rate.*

Proof. We already established in Theorem 10 that the iterates converge to the optimal solution, and since $H^k \to H^*$ and $c_H > \|H^*\|$, the Hessian is not rescaled according to (34) in Step 3. Using Theorem 13 we know that, once k is sufficiently large, the step $d^{S,k}$ computed in Step 5 of Algorithm 3 is identical with the SQP step from (5) for (4); we can ignore (27d) here because $x_{j0}^* > 0$ and $d_{j0}^{S,k} \to 0$ and therefore this constraints is not active for large k. This also implies that the condition in Step 6 is never true and thus $\hat{\mathcal{E}}_k = \mathcal{E}(x^*)$. If the decrease condition in Step 11 is not satisfied, by a similar argument we have that s^k computed in Step 15 is the second-order correction step from (12) for (4). Due to Lemma 15 we can now apply Theorem 3 to conclude that either $d^{S,k}$ or $d^{S,k} + s^k$ is accepted to define the next iterate for large k and that the iterates converge at a quadratic rate. ∎

5. Numerical Experiments.

In this section, we examine the performance of Algorithm 3. First, using randomly generated instances, we consider three types of starting points: (i) uninformative default starting point (cold start), (ii) solution of a perturbed instance, (iii) solution computed by an interior-point SOCP solver whose accuracy we wish to improve. Then we briefly report results using the test library CBLIB. The numerical experiments were performed on an Ubuntu 22.04 Linux server with a 2.1GHz Xeon Gold 5128 R CPU and 256GB of RAM.

5.1. Implementation.

We implemented Algorithm 3 in MATLAB R2021b, with parameters $c_{\text{dec}} = 10^{-6}$, $c_{\text{inc}} = 2$, $c_H = 10^{12}$, and $\rho^{-1} = 50$. In each iteration, we identify $\mathcal{E}(x^k) = \{j \in \mathcal{J} : \|x_j^k\|_\infty < 10^{-6}\}$ and $\mathcal{D}(x^k) = \{j \in \mathcal{J} \setminus \mathcal{E}(x^k) : \|\bar{x}_j^k\| > 10^{-8}\}$. The set $\mathcal{Y}_j^0$ is initialized to $\hat{\mathcal{Y}}_j^0$ (see (20)), and λ^0 is a given starting value for λ, if provided, and zero otherwise. In addition, since the identification of the optimal extremal-active set $\mathcal{E}(x^*)$ requires $z_j^* \in \mathcal{C}_j^\circ(\mathcal{Y}_j)$, we add $-\check{z}_j^0$ to $\mathcal{Y}_j^0$, where $\check{z}^0 = c + A^T \lambda^0$.

The algorithm terminates when the violation of the SOCP optimality conditions

(13) for the current iterate satisfies

(56)
$$E(x^k, \lambda^k, \check{z}^k) = \max \left\{ \begin{array}{l} \|[Ax^k - b]^+\|_\infty, \|(Ax^k - b) \circ \lambda^k\|_\infty, \|[-\lambda^k]^+\|_\infty \\ \max_{j \in \mathcal{J}} \left\{ [r_j(x^k)]^+, [r_j(\check{z}^k)]^+, |(x_j^k)^T \check{z}_j^k| \right\} \end{array} \right\} \leq \epsilon_{\text{tol}}$$

with $\check{z}^k = c + A^T \lambda^k$, for some $\epsilon_{\text{tol}} > 0$.

As in [22], the sufficient descent condition (11) is slightly relaxed by

$$\varphi(\hat{x}^{k+1}; \rho^k) - \varphi(x^k; \rho^k) - 10\epsilon_{\text{mach}} |\varphi(x^k; \rho^k)| \leq c_{\text{dec}} \left(m^k(x^k + d; \rho^k) - m^k(x^k; \rho^k) \right)$$

to account for cancellation error, where ϵ_{mach} is the machine precision. Finally, to avoid accumulating very similar hyperplanes that would lead to degenerate QPs, we do not add a new generating point v_j to $\mathcal{Y}_j^k$ if there already exists $y_j \in \mathcal{Y}_j^k$ such that $\left\| \frac{\bar{v}_j}{\|\bar{v}_j\|} - \frac{\bar{y}_j}{\|\bar{y}_j\|} \right\|_\infty \leq 10^{-10}$.

In these experiments, we disabled the second-order correction step (Steps 15–19) because we noticed that it was never accepted in practice. In a more sophisticated implementation, one would include a heuristic that attempts to detect the Maratos effect and then triggers the second-order correction step in specific situations.

The QPs were solved using ILOG CPLEX V12.10, with optimality and feasibility tolerances set to 10^{-9} and "dependency checker" and "numerical precision emphasis" enabled, using the primal simplex method. When CPLEX did not report a solution status "optimal" and the QP KKT error was above 10^{-9}, a small perturbation was added to the Hessian matrix, i.e., we replaced H^k by $H^k + 10^{-7} \cdot I$. This helped in some cases in which CPLEX (incorrectly) reported that H^k was not positive semi-definite. If CPLEX still did not find a QP solution with KKT error less than 10^{-9}, we attempted to resolve the QP with the barrier method, the dual simplex method, and the primal simplex method again, until one was able to compute a solution. If all solvers failed for QP (27), the algorithm continued in Step 21. If no solver was able to solve (18), we terminated the main algorithm and declared a failure.

We emphasize that the purpose of our implementation is to assess whether the proposed algorithm exhibits behavior that validates the stated goals: Convergence from any starting point and rapid local convergence to highly accurate solutions. In its current implementation, it requires more computation time than highly sophisticated commercial solvers such as MOSEK or CPLEX, which were developed over decades and have highly specialized linear algebra routines that are tightly integrated into the algorithms. As we observed at the end of Section 3.7, many of the QPs in Algorithm 3 that are solved in succession are similar to each other, and savings in computation times should therefore be achievable. However, our prototype implementation based on the Matlab CPLEX interface does not allow us to utilize callback functions for adding or removing hyperplanes. Achieving these savings in computation time thus requires a more sophisticated implementation, a task that is outside of the scope of this paper. Consequently, we do not report solution times here.

5.2. Randomly generated QCQPs. The experiments were performed on randomly generated SOCP instances of varying sizes, specified by (n, m, K). Here, $n, m \geq 1$ are the number of variables and linear constraints, respectively. $K \geq 1$ specifies the number of cones of each "activity type": $|\mathcal{E}(x^*)| = K$, $|\{j \in \mathcal{J} : r_j(x_j^*) = 0, x_j^* \neq 0\}| = K$, and $|\{j \in \mathcal{J} : r_j(x_j^*) < 0\}| = K$, i.e., there are K cones that are extremal-active, K that are active at the boundary, and K that are inactive at the optimal solution x^*. The dimensions of the cones are randomly chosen. In addition,

n	m	K	solved	total iter	SQP iter	total QP (27)	total QP (18)
200	60	10	30	6.67	6.67	9.77	0.00
400	120	20	30	7.20	7.20	11.57	0.00
1000	300	50	30	7.23	7.23	12.17	0.00
200	60	4	30	7.53	7.07	11.83	0.90
400	120	8	30	8.27	7.77	14.20	1.00
1000	300	20	30	8.67	7.80	15.93	1.83
200	60	2	30	8.47	7.87	13.90	1.20
400	120	4	30	8.87	8.07	15.30	1.60
1000	300	10	30	9.47	8.43	17.27	1.97

Table 1: Results with $x^0 = 0$, $\epsilon_{\text{tol}} = 10^{-7}$, average per-size statistics taken over 30 random instances. "solved": number of instances solved (out of 30); "total iter": total number of iterations in Algorithm 3; "SQP iter": number of iteration in which NLP-SQP step was accepted in Steps 11 or 16; "total QP (27)" / "total QP (18)": Total number of QPs of that type solved.

there are variables that are not part of any cone, with bounds chosen in a way so that the non-degeneracy assumption, Assumption 2, holds. A detailed description of the problem generation is stated in Appendix A in [13].

Table 1 summarizes the performance of the algorithm with an uninformative starting point $x^0 = 0$. Each row lists average statistics for a given problem size (n, m, K), taken over 30 random instances. We see that the proposed algorithm is very reliable and solved every instance to the tolerance $\epsilon = 10^{-7}$. The average number of iterations is mostly between 7–9, during most of which the second-order NLP-SQP step was accepted.

To give an idea of the computational effort, we report the number of times QPs (27) and (18) were solved. And we can draw further conclusions from this data: Consider, for example, the last row. At the beginning of each iteration, QP (27) is solved to obtain the NLP-SQP step. The difference with the total number of iterations, i.e., 17.27-9.47=7.80, gives us the total number of times in which the guess $\hat{\mathcal{E}}^k$ of the extremal-active cones needed to be corrected in Steps 6–9. In other words, on average, the loop Steps 6–9 is executed 7.80/9.47=0.82 times per iteration. Similarly, the last column tells us the total number of iterations of the loop in Steps 21–31. The loop was only executed when the NLP-SQP step was not accepted, so in 9.47-8.43=1.04 iterations, taking 1.97/1.04=1.89 loop iterations on average.

The experiments are presented in three groups where the ratio between n and K is kept constant. As the number of cones, K, decreases from one group to the next, the average size of the individual cones increases by a factor of 2.5 and 2, respectively. This increase seems to result in slightly more iterations in which the SQP step was rejected, indicating that the simple linearization (27c) of the non-extremal-active cones becomes sometimes insufficiently accurate.

In comparison, the pure primal cutting plane method (Algorithm 3 without Newton steps and without step 30) required up to three times more total iterations.

The remaining experiments in this section investigate to which degree the algorithm is able to achieve our primary goal of taking advantage of a good starting point. We begin with an extreme situation, in which we first solve an instance with

n	m	K	total iter	SQP iter	total QP (27)	total QP (18)	Mosek error	final error
200	60	10	1.10	1.07	1.10	0.07	2.33e-06	1.63e-10
400	120	20	1.03	1.00	1.03	0.03	2.67e-06	1.70e-10
1000	300	50	1.07	1.03	1.07	0.03	3.49e-06	1.76e-10
200	60	4	1.03	1.03	1.03	0.00	5.97e-06	1.69e-10
400	120	8	1.00	1.00	1.00	0.00	2.28e-06	1.87e-10
1000	300	20	1.03	0.83	1.03	0.27	5.20e-06	1.72e-10
200	60	2	1.00	1.00	1.00	0.00	2.02e-06	1.53e-10
400	120	4	1.13	1.10	1.13	0.03	4.85e-06	2.03e-10
1000	300	10	1.20	1.10	1.20	0.13	1.22e-05	2.41e-10

Table 2: Result with MOSEK solution as x^0, $\epsilon_{\text{tol}} = 10^{-9}$. All instances were solved. "Mosek error": Optimality error E (56) at Mosek solution; "final error": Optimality error E at final iterate of Algorithm 3.

n	m	K	solved	total iter	SQP iter	total QP (27)	total QP (18)
200	60	10	30	1.00	0.97	1.00	0.07
400	120	20	30	1.00	0.97	1.00	0.03
1000	300	50	30	1.00	0.97	1.00	0.07
200	60	4	30	1.00	1.00	1.00	0.00
400	120	8	30	1.00	0.93	1.00	0.07
1000	300	20	30	1.00	0.87	1.00	0.20
200	60	2	30	1.00	1.00	1.00	0.00
400	120	4	30	1.00	0.97	1.00	0.03
1000	300	10	30	1.07	1.00	1.03	0.07

Table 3: Result with 10^{-3} perturbation, $\epsilon_{\text{tol}} = 10^{-7}$.

the interior-point SOCP solver MOSEK V9.1.9 (called via CVX), using the setting `cvx_precision=high` corresponding to the MOSEK tolerance $\epsilon = \epsilon_{\text{mach}}^{2/3}$, and give the resulting primal-dual solution as starting point to Algorithm 3. Choosing any tighter MOSEK tolerances leads to failures in several problems. Table 2 summarizes the results. In all cases, the algorithm converges rapidly to an improved solution, reducing the error by 4 orders of magnitude, most of the time with only a single second-order iteration. The Mosek error was dominated by the violation of complementarity. This demonstrates the ability of the proposed method to improve the accuracy of a solution computed by an interior-point method.

For the final experiments, summarized in Tables 3 and 4, the starting point is the MOSEK solution of a perturbed problem, in which 10% of the objective coefficients c were perturbed by uniformly distributed random noise of the order of 10^{-3} and 10^{-1}, respectively. For the small perturbation, similar to Table 2, Algorithm 3 terminated in one iteration most of the time. More iterations were required for the larger perturbation, but still significantly fewer compared to the uninformative starting point, see Table 1.

n	m	K	solved	total iter	SQP iter	total QP (27)	total QP (18)
200	60	10	30	1.20	1.07	1.00	0.19
400	120	20	30	1.33	1.17	1.00	0.73
1000	300	50	30	1.60	1.23	1.02	1.29
200	60	4	30	1.27	1.13	1.02	0.71
400	120	8	30	1.67	1.27	1.16	0.48
1000	300	20	30	2.10	1.40	1.27	0.57
200	60	2	30	1.67	1.33	1.24	0.42
400	120	4	30	2.30	1.87	1.30	0.38
1000	300	10	30	3.67	2.53	1.53	0.59

Table 4: Result with 10^{-1} perturbation, $\epsilon_{\text{tol}} = 10^{-7}$.

5.3. CBLIB instances. To demonstrate the robustness of the algorithm we also solved instances from the Conic Benchmark Library CBLIB [25]. Some instances involve rotated second-order cone constraints, and we reformulated them so that they fit into our standard form (1). We chose all 1,575 instances with at most 10,000 variables and 10,000 constraints. Integer variables were relaxed to be continuous.

Using the starting point $x^0 = 0$, the method was able to solve 99.2% of the instances, where 10 problems could not be solved due to failures of the QP subproblem solver, and Algorithm 3 exceeded the maximum number of 200 iterations in 2 cases. In comparison, MOSEK, with default settings, failed on 5 instances (those were solved correctly with Algorithm 3), incorrectly declared 3 instances to be infeasible, and labeled 6 instances to be unbounded (of which 3 were solved by Algorithm 3). Table 5 in [13] gives detailed statistics for the different problem groups in the CBLIB test collection. We observed that some instances, especially those in the `clay*`, `fo[7-9]*`, `m[3-9]*`, `no7*`, `o[7-9]_*` subsets, are degenerate, having an optimal objective function value of 0, and the assumption necessary to prove fast local convergence is violated. This matches our observation that the SQP step was accepted only in a relatively small fraction of the iterations for these instances.

To showcase the warm-starting feature of the algorithm, we took the 1,563 previously successfully solved instances, perturbed 10% of the entries of the final primal-dual iterate by a random perturbation, uniformly chosen in $[-0.1, 0.1]$, and used this as the starting point for a warm-started run. Here, QP subproblem failure occurred in 3 cases and 2 instances exceeded the iteration limit. The number of iterations was reduced in most cases. Specifically, for 14 out of the 26 problem subsets, the iteration count was reduced by at least 60%.

6. Concluding remarks. We presented an SQP algorithm for solving SOCPs and proved that it converges from any starting point and achieves local quadratic convergence for non-degenerate SOCPs. Our numerical experiments indicate that the algorithm is reliable, converges quickly when a good starting point is available, and produces more accurate solutions than a state-of-the-art interior-point solver.

Future research would investigate whether the proposed algorithm is a valuable alternative for interior-point methods for small problems or for the solution of a sequence of related SOCPs. An efficient implementation of the algorithm beyond our Matlab prototype would be tightly coupled with a tailored active-set QP solver that efficiently adds or removes cuts instead of solving each QP essentially from scratch.

Parametric active-set solvers such as qpOASES [5] or QORE [20] might be suitable options since they do not require primal or dual feasible starting points.

Acknowledgments. We thank Javier Peña for his suggestion for the proof of Lemma 8, as well as three referees whose comments helped us to improve the paper.

REFERENCES

[1] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[2] C. Coey, M. Lubin, and J. P. Vielma. Outer approximation with conic certificates for mixed-integer convex problems. *Mathematical Programming Computation*, 12(2):249–293, 2020.

[3] M. Diehl, F. Jarre, and C. H. Vogelbusch. Loss of superlinear convergence for an SQP-type method with conic constraints. *SIAM Journal on Optimization*, 16(4):1201–1210, 2006.

[4] S. Drewes and S. Ulbrich. Subgradient based outer approximation for mixed integer second order cone programming. In *Mixed integer nonlinear programming*, pages 41–59. Springer, 2012.

[5] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6:327–363, 2014.

[6] R. Fletcher. Second order corrections for non-differentiable optimization. In *Numerical analysis*, pages 85–114. Springer, 1982.

[7] N. Goldberg and S. Leyffer. An active-set method for second-order conic-constrained quadratic programming. *SIAM Journal on Optimization*, 25(3):1455–1477, 2015.

[8] N. Goswami, S. K. Mondal, and S. Paruya. A comparative study of dual active-set and primal-dual interior-point method. *IFAC Proceedings Volumes*, 45(15), 2012.

[9] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2022.

[10] S. Hayashi, T. Okuno, and Y. Ito. Simplex-type algorithm for second-order cone programmes via semi-infinite programming reformulation. *Optimization Methods and Software*, 31(6):1272–1297, 2016.

[11] IBM ILOG. *User's Manual for CPLEX*, 2019.

[12] H. Kato and M. Fukushima. An SQP-type algorithm for nonlinear second-order cone programs. *Optimization Letters*, 1(2):129–144, 2007.

[13] X. Luo and A. Wächter. A Quadratically Convergent Sequential Programming Method for Second-Order Cone Programs Capable of Warm Starts, 2022. arXiv:2207.03081.

[14] N. Maratos. *Exact penalty function algorithms for finite dimensional and control optimization problems*. PhD thesis, Imperial College London (University of London), 1978.

[15] D. K. Molzahn and I. A. Hiskens. A survey of relaxations and approximations of the power flow equations. *Foundations and Trends in Electric Energy Systems*, 4(1-2):1–221, 2019.

[16] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.1.*, 2019.

[17] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006.

[18] T. Okuno, K. Yasuda, and S. Hayashi. Sl1QP based algorithm with trust region technique for solving nonlinear second-order cone programming problems. *Interdisciplinary Information Sciences*, 21(2):97–107, 2015.

[19] F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000.

[20] L. Schork. A parametric active set method for general quadratic programming. Master's thesis, Heidelberg University, Germany, 2015.

[21] R. J. Vanderbei and H. Yurttan. Using LOQO to solve second-order cone programming problems. Technical report, Priceton University, 1998.

[22] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[23] V. Zhadan. The variant of primal simplex-type method for linear second-order cone programming. In *Optimization and Applications: 12th International Conference, OPTIMA 2021, Petrovac, Montenegro, September 27–October 1, 2021, Proceedings*, pages 64–75. Springer, 2021.

[24] X. Zhang, Z. Liu, and S. Liu. A trust region SQP-filter method for nonlinear second-order cone programming. *Computers & Mathematics with Applications*, 63(12):1569–1576, 2012.

[25] Zuse Institute Berlin. CBLIB - The Conic Benchmark Library. https://cblib.zib.de/.