ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

# Active Set Method for Second-Order Conic-Constrained Quadratic Programming

**Noam Goldberg and Sven Leyffer**

Mathematics and Computer Science Division

Preprint ANL/MCS-P5085-0214

February 19, 2014

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

## Contents

# Active-Set Method for Second-order Conic-Constrained Quadratic Programming

Noam Goldberg and Sven Leyffer

February 19, 2014

**Abstract**

We consider the minimization of a quadratic objective subject to second-order cone constraints that generalizes the well-studied bound-constrained quadratic programming (QP) problem. We propose a new two-phase method: in the first phase a projected-gradient method is used to quickly identify the active set of cones, and in the second phase Newton's method is applied to rapidly converge given the subsystem of active cones. Computational experiments confirm that the conically constrained QP is solved more efficiently by our method than by specialized conic optimization solvers and a general nonlinear programming solver. We further suggest how to generalize the two-phase method to solve problems with additional linear constraints using an augmented Lagrangian approach.

**Keywords: Conically Constrained Quadratic Program, Projected Gradient Method.**

**AMS-MSC2010: 90C20, 90C25, 90C52**

## 1 Introduction

Traditionally, interior-point methods have been used to solve (linear) conic optimization problems. Second-order cone programs (SOCPs) of moderate size can be efficiently solved by semidefinite programming (SDP) interior-point solvers such as SDPT3 [21] and SeDuMi [20]. One drawback of interior-point solvers is that they cannot be efficiently restarted from a current optimal solution. This is a disadvantage when solving conic relaxations of combinatorial optimization problems, for example, max cut [13, 18] or other binary quadratic problems [10], within a branch-and-bound method. Mixed-integer SOCPs [7] naturally result in SOCP relaxations. In addition, second-order cone relaxations of mixed-integer nonlinear programs include the recently proposed perspective relaxation of Günlük and Linderoth [9]. Another drawback of interior-point methods in attempting to solve large-scale continuous optimization problems in general is the computational cost of solving a large linear system at each iteration.

Recently, renewed interest has been expressed in first-order methods for linear and nonlinear conic optimization. Work includes a semismooth method that is proposed for nonlinear second-order cone programming [11] and an augmented Lagrangian method that has been developed for linear semidefinite programming [23].

In the following we propose a first-order two-phase active-set method for second-order cone-constrained quadratic programming. This method can be used as a building block to solve larger classes of problems that include linear constraints. The minimization of a quadratic function subject to second-order conic constraints generalizes bound-constrained quadratic programming, which has been considered by many authors including Bertsekas [3] and Moré and Toraldo [15]. The bound-constrained case is a simple special case in the sense that the projected gradient corresponds to a piecewise linear curve and the Cauchy point is given by a closed-form expression over each segment. The general second-order cone constrained case is more challenging; however, it maintains the advantage that it is fairly inexpensive to compute the projections onto the feasible region and the curve's breakpoints at which constraints become active.

We start by defining the second-order cone. For a a positive integer $n$ and a vector $w \in \mathbb{R}^n$, we let $||w||$ denote its Euclidean norm, let $\bar{w} = \begin{pmatrix} w_1 & \dots & w_{n-1} \end{pmatrix}^T$, and denote the $n$ dimensional second-order cone by $\mathcal{Q}_n = \{w \in \mathbb{R}^n \mid ||\bar{w}|| \leq w_n\}$. To define the conic optimization problem, we let $z \in \mathbb{R}^N$ be the vector of unknowns; and for $\ell = 1, \dots, m$ we let $z[\ell] \in \mathbb{R}^{n_\ell}$ be a subvector, which will refer to a sequence of consecutive components of $z$ and that lies in an $n_\ell$-dimensional cone. Accordingly, for all $\ell$, $z[\ell] \in \mathcal{Q}_{n_\ell}$, $z = (z[1], \dots, z[m])$, $N = n_1 + \dots + n_m$, and we note that that the $z[\ell]$'s are a partition of $z$.

We are interested in the following quadratic optimization problem over cones:

$$\begin{aligned} &\min && q(z) \equiv g^T z + \frac{1}{2} z^T G z && && \text{(1a)} \\ &\text{subject to} && z[\ell] \in \mathcal{Q}_{n_\ell} && \ell = 1, \dots, m. && \text{(1b)} \end{aligned}$$

In particular, (1b) implies for each $\ell$ that

$$c_\ell(z) \equiv z[\ell]_{n_\ell} - \sqrt{\sum_{j=1}^{n_\ell - 1} z[\ell]_j^2} \geq 0.$$

Note that nonnegativity constraints are a special case for all $\ell$ with $n_\ell = 1$, for which the sum over the empty set in square root is zero, and so $c_\ell(z) = z[\ell]_{n_\ell} \geq 0$. In the following, assume for convenience that $z = \begin{pmatrix} x \\ y \end{pmatrix}$ with $x[\ell] \in Q_1 = \mathbb{R}_+$ for $\ell = 1, \dots m_b$ and $y_{\ell - m_b} \in Q_{n_\ell}$, with $n_\ell \geq 2$ for $\ell = m_b + 1, \dots, m$. Further, let $m_c \equiv m - m_b$ denote the number of conic variables, and $\mathcal{Q} \equiv \mathcal{Q}_{n_1} \times \cdots \mathcal{Q}_{n_m}$. We also assume that the Hessian matrix, $G$, is positive semidefinite and introduce Lagrange multipliers $\nu \in \mathbb{R}^N$. The conic Karush-Kuhn Tucker (KKT) first-order conditions of (1) can then be written as

$$Gz + g - \nu = 0 \tag{2a}$$

$$z \in \mathcal{Q}, \quad \nu \in \mathcal{Q}, \quad z^T \nu = 0; \tag{2b}$$

see [12, 19].

A key challenge in solving (1) using common general nonlinear programming (NLP) techniques is the nondifferentiability of each $c_\ell(z)$ at $\bar{z}_\ell = 0$. This observation motivates us to consider a two-phase active-set approach: once the optimal active set is identified, then the associated constraint functions are continuously differentiable near the optimal solution in the reduced subspace of free variables (after having fixed some of the variables at zero). The two-phase active-set algorithm involves a projected-gradient phase to identify an active set and a Newton phase that solves a sequence of equality-constrained sequential quadratic programming (EQP) problems to obtain quadratic convergence once the active set is identified.

**Solving SOCPs with Standard NLP Solvers.** One approach to solving SOCPs is to simply use standard NLP solvers. However the following example illustrates the difficulty of traditional nonlinear solvers.

**Example 1.** *Consider the SOCP* (1) *with the following data:*

$$G = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix} \qquad\qquad g = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

*with $m = 2$, $n_1 = 1$, and $n_2 = 3$. Using* (2) *it can be verified that $z = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}^T$ is optimal for this instance with $\nu = \begin{pmatrix} 0 & 0 & 0 & -1 \end{pmatrix}^T$. If we let $\lambda \in \mathbb{R}^m$ denote the NLP Lagrange multipliers, then the NLP first-order conditions at this point,*

$$0 - \begin{pmatrix} \lambda_1 \\ \lambda_2 \nabla c_2(0) \end{pmatrix} \quad = \quad 0 - \begin{pmatrix} \lambda_1 \\ \frac{-z_2\lambda_2}{0} \\ \frac{-z_3\lambda_2}{0} \\ \lambda_2 \end{pmatrix} \quad = \quad 0$$

*clearly are undefined, since $\nabla c_2(0)$ involves division by zero.*

This is Problem SOCP-2 in Table 1, which shows the iteration count for a number of standard NLP solvers on four simple SOCPs. The problem data for all four problems is given in Appendix A.

Table 1: Major iteration counts for some small SOCP example problems with NLP solvers.

| Problem | SNOPT | MINOS | Filter | Ipopt | Comments |
|---|---|---|---|---|---|
| SOCP-1 | 13 | 12 | 5 | 6 | superlinear convergence |
| SOCP-2 | 25 | 32 | 10 | 18 | linear convergence |
| SOCP-3 | 18 | 26 | 42 | 7 | IEEE exception & linear convergence |
| SOCP-4 | 17 | 30 | 6 | 8 | superlinear convergence |

The four nonlinear solvers converge superlinearly for problems SOCP-1 and SOCP-4. However, all solvers suffer from a slow linear rate of convergence on problem SOCP-2. For problem SOCP-3, both SNOPT and Filter converge linearly, with Filter also showing warning messages about IEEE exceptions that indicate it is trying to evaluate the square root function of the conic constraint for a negative argument. Surprisingly, both MINOS and IPOPT converge superlinearly for this problem. The examples illustrate that nonlinear solvers are not sufficiently robust to solve general conically constrained optimization problems and that IEEE errors may even result in failure. Our preliminary results motivate the derivation of new, robust methods for solving SOCPs. The solution of SOCPs, as well as smooth reformulations of SOCPs, using NLP solvers has been studied by Vanderbei and Yurttan [22]. In their experiments, LOQO, a general nonlinear programming solver, was better able to solve SOCPs than were special-purpose solvers such as SeDuMi. However, their approach resorts to solving perturbations of the original problems.

**Outline of Algorithm.** Our two-phase method first computes projected-gradient steps to identify the active set of constraints, including but specially treating those $\ell = 1, \dots, m$ for which $z[\ell]_{n_\ell} = 0$. The projected-gradient phase for this problem generalizes the projected-gradient algorithm for large-scale bound-constrained QPs, for example the method of Moré and Toraldo [15].

A particular challenge of the projected-gradient phase in our case is that the Cauchy point is not given in closed form. Consequently, the projected-gradient algorithm may revert to a projected line search along the last segment. The projected-gradient phase is then followed by a Newton phase in the reduced subspace defined by the free variables and the boundary of the active cones. This allows for fast convergence to a solution once an active set is identified. If the active set appears to be incorrect; for example, if a Lagrange multiplier is negative (i.e., of the wrong sign for the original inequality problem) or if the EQP iterations do not appear to converge – then the algorithm may need to revert back to the projected-gradient phase. The convergence criterion is always checked by using the projected gradient in the gradient-projection phase, ensuring that both phases terminate with the same optimality condition. Verifying termination in the gradient-projection phase also serves to determine the correctness of the identified active set, and we have found it to be a more robust alternative to evaluating the sign of the Lagrange multipliers.. The following is a high-level outline of the method.

A sketch of the two-phase method

```
while not converged do
  Perform a projected-gradient iteration.
  if active set is settled down then
    Save current iterate as $z^{(pg)}$.
    while EQP makes progress and not optimal do
      Perform EQP iteration
    end while
    if inactive constraints are violated then
      Project onto the feasible region
    end if
    if objective is made worse then
      Reset current iterate as $z^{(pg)}$.
    end if
  end if
end while
```

The algorithm has two nested loops. In the outer loop we make projected-gradient steps that ensure convergence to a stationary point. In the inner loop, which implements the Newton's phase, we perform a sequence of EQP iterations for a fixed active set until we find an optimal solution or until progress toward convergence is found to be insufficient. We restart from either the last iterate of the projected-gradient phase or the projection of the last EQP step onto the feasible region, depending on the objective value. The details of the method are described in Section 3.

**Beyond Conic-Constrained Quadratic Programs.** In practice, we are also interested in problems that include general linear constraints of the form $Az = b$. A straightforward way to adapt our approach to this class of problems would be to handle the linear constraints using an augmented Lagrangian approach. In the bound-constrained case (a simplification of our problem), projected-gradient methods have been shown to be efficient subproblem solvers within an augmented Lagrangian method for handling additional linear constraints; the reader is referred to [6, 8]. Thus, our algorithm can be seen as an important step toward the development of more general conic solvers. In particular the LANCELOT NLP solver is known to effectively solve large-scale problems through a quadratic penalty function of the constraint violation of nonlinear and linear constraints alike. The problem is solved through a sequence of easier bound-constrained subproblems and a corresponding sequence of penalty parameter values. In our case, a more general second-order cone-constrained quadratic program replaces the bound-constrained subproblem. Our approach differs from augmented Lagrangian methods considered for the dual form of conic optimization problems [11, 23]; instead of penalizing conic infeasibility, our method exploits the ease of computing the projection onto the second-order cones in order to maintain feasibility with respect to the conic constraints. The linear constraint infeasibility is penalized, and the algorithm terminates once their feasibility is restored.

**Outline.** In Section 2 we introduce the details of the projected-gradient phase. In Section 2.3 we elaborate on the EQP to be solved in the second phase of the algorithm. The two-phase algorithm, including the full details, is described formally in Section 3. We present our computational experiments in Section 4 and conclude in Section 5 with a brief summary and discussion of future work.

## 2 Projected-Gradient Method for Cones

A constraint is said to be active if it holds with equality. The projected-gradient algorithm is especially efficient in identifying the set of active constraints. The projected gradient is a piecewise curve whose pieces are defined by the active constraints. Gradient projection algorithms iteratively step through the ordered set of breakpoints in the steepest descent direction. Despite relatively inferior convergence properties, projected-gradient algorithms can also be effective in practice when the projection is inexpensively computed and the computational effort of each iteration remains low.

### 2.1 Projection onto Second-Order Conic Constraint Faces

For $w \in \mathbb{R}^n$ the Euclidean projection onto a convex set $\mathcal{S} \subseteq \mathbb{R}^n$ is defined as

$$P_{\mathcal{S}}(w) = \operatorname*{argmin}_{v \in \mathcal{S}} \{||v - w||^2\}. \tag{3}$$

It is well known that $P_{\mathcal{Q}_n}(w)$ can be computed in closed form for all $n \geq 1$. This result can be derived from the positive semidefinite projection formula and eigenvalue decomposition of the second-order cone algebra [1] or alternatively by using elementary techniques [2]. In particular the projection onto the second-order cone is

$$P_{\mathcal{Q}_n}(w) = \begin{cases} w & ||\bar{w}|| \leq w_n, \\ 0 & ||\bar{w}|| \leq -w_n, \\ \frac{||\bar{w}||+w_n}{2}\left(\frac{\bar{w}}{||\bar{w}||}, 1\right) = \frac{1}{2}\left(\bar{w}\left(1 + \frac{w_n}{||\bar{w}||}\right), ||\bar{w}|| + w_n\right) & \text{otherwise.} \end{cases} \tag{4}$$

Note that the first two cases in (4) correspond to $w \in \mathcal{Q}_n$ and $w \in -\mathcal{Q}_n$, respectively, where $-\mathcal{Q}_n$ denotes the reflection of $\mathcal{Q}_n$ through the origin. Also, note that in the case of $n = 1$, (4) reduces to the simple projection of $w$ onto the nonnegative orthant:

$$P_{\mathcal{Q}_1}(w) = \begin{cases} w_i & w_i \geq 0 \\ 0 & w_i < 0. \end{cases}$$

## 2.2 Evaluating the Breakpoints and Projected Line Search

Consider some $\ell \in \{1, \ldots, m\}$, and for convenience let $w = z[\ell]$ and $n = n_\ell$. Let $d \in \mathbb{R}^n$ be a direction of descent. Then, the breakpoints corresponding to intersection points with the cone's reflection through the origin and those corresponding to the intersection with the cone's boundary (excluding the origin), respectively, are

$$\{t_\ell^-\} = \min\Big( \big\{ t \mid \left|\left|\bar{w} + t\bar{d}\right|\right| \leq -w_n - td_n \big\} \cup \{\infty\} \Big)$$

and

$$\{t_\ell^+\} = \min\Big( \big\{ t \mid \left|\left|\bar{w} + t\bar{d}\right|\right| \geq w_n + td_n > 0 \big\} \cup \{\infty\} \Big).$$

Note that one or both of the intersection points may not necessarily exist. When an intersection point exists, its value is given by a (finite) $t \geq 0$ that solves

$$\left( \sum_{j=1}^{n-1} d_j^2 - d_n^2 \right) t^2 + 2 \left( \sum_{j=1}^{n-1} w_j d_j - w_n d_n \right) t + \sum_{j=1}^{n-1} w_j^2 - w_n^2 = 0,$$

with, respectively, $w_n + td_n \leq 0$ or $w_n + td_n > 0$. Otherwise if there is no such solution, by convention, a breakpoint's value is $\infty$. So, for $\ell = 1, \ldots, m$

$$t_\ell^-, t_\ell^+ \in \left\{ \frac{-2(\bar{w}^T\bar{d} - w_n d_n) \pm \sqrt{4(\bar{w}^T\bar{d} - w_n d_n)^2 - 4(\left|\left|\bar{d}\right|\right|^2 - d_n^2)(\left|\left|\bar{w}\right|\right|^2 - w_n^2)}}{2\left(\left|\left|\bar{d}\right|\right|^2 - d_n^2\right)}, \infty \right\}. \quad (5)$$

The projected-gradient direction and the corresponding conic breakpoints are illustrated in Figure 1(a). The second-order conic constraint corresponds to the shaded cone on top of the unshaded cone, which is the reflection of the feasible cone through the origin. The breakpoint $t_\ell^+$ corresponds to the arrow at the intersection of the gradient with the shaded cone. The breakpoint $t_\ell^-$ corresponds to the second arrowhead at the intersection of the solid line with the unshaded cone; the arrowhead of the dotted arrow corresponds to its reflection onto the feasible (shaded) cone.

The bound constraint breakpoints can be viewed as a special case where $\left|\left|\bar{w}\right|\right| = \left|\left|\bar{d}\right|\right| = 0$ in (5). Accordingly, for $\ell = 1, \ldots, m_b$ the breakpoints corresponding to nonnegativity bounds are simply given by

$$t_\ell^- = \begin{cases} -x_\ell / g_\ell & g_\ell > 0, \\ \infty & \text{otherwise,} \end{cases}$$

which is also consistent with the bound-constrained case as explored, for example, in [15]. Figure 1(b) illustrates the projected-gradient direction in a bound-constrained feasible region.

Next, the projected-gradient procedure sorts the breakpoint values $t_\ell^\alpha$ for $\alpha \in \{-, +\}$ and $\ell = 1, \ldots, m$. Denote the ordering of their distinct values by

$$0 \leq \tau_1 < \tau_2 < \cdots < \tau_r \leq \infty, \quad (6)$$

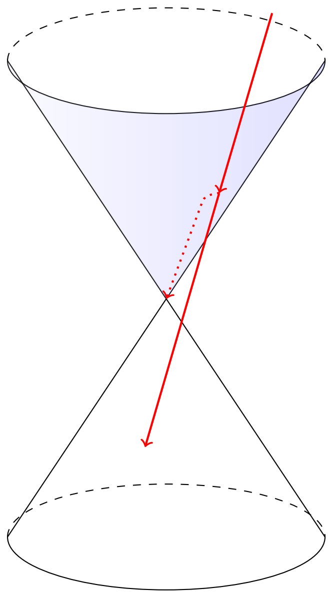
and for convenience also let $\tau_{r+1} = \infty$.

For a given feasible point $z = (x, y)$ of (1) we define the following active sets

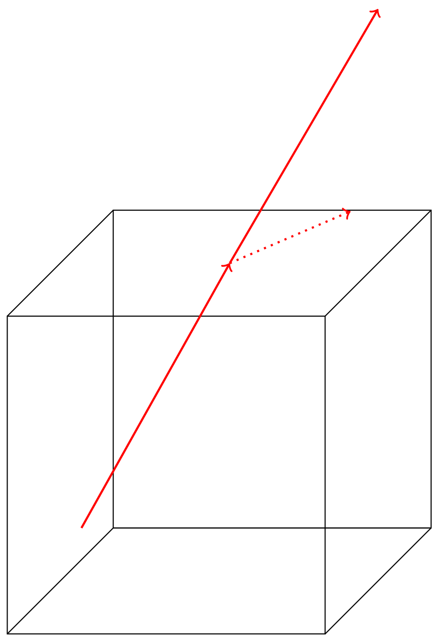
$$\mathcal{A}_+(z) = \left\{ \ell \in \{1, \ldots, m\} \mid \left|\left|\overline{z[\ell]}\right|\right| = z_{\ell n} > 0 \right\}, \quad (7a)$$

$$\mathcal{A}_0(z) = \left\{ \ell \in \{1, \ldots, m\} \mid \left|\left|\overline{z[\ell]}\right|\right| = z_{\ell n} = 0 \right\}, \quad (7b)$$

and $\mathcal{A}(z) = \mathcal{A}_0(z) \cup \mathcal{A}_+(z)$.



(a) A second-order cone feasible region $\mathcal{Q}_n$, its reflection through the origin $-\mathcal{Q}_n$, and the corresponding projected-gradient breakpoints, $t_\ell^+, t_\ell^0$.



(b) A bound-constrained feasible region. The breakpoints $t_\ell^-$, for $\ell = 1, \ldots, m$, correspond to the intersection points (shown as arrowheads) of the gradient with the faces of the cube.

Figure 1: Illustration of the gradient and projected gradient over a single second-order cone vs. a bound-constrained region. The arrowheads indicate the breakpoints in the projected-gradient direction.

Similarly, for a given, fixed $\epsilon > 0$, we define the following working sets:

$$\mathcal{W}_+(z) = \left\{ \ell \in \{1,\ldots,m\} \mid \left|\left|\overline{z[\ell]}\right|\right| \geq z_{\ell n} - \epsilon > 0 \right\}, \tag{8a}$$

$$\mathcal{W}_0(z) = \left\{ \ell \in \{1,\ldots,m\} \mid \left|\left|\overline{z[\ell]}\right|\right| \leq z_{\ell n} \leq \epsilon \right\}, \tag{8b}$$

and $\mathcal{W}(z) = \mathcal{W}_0(z) \cup \mathcal{W}_+(z)$.

We observe that the working sets are supersets of the active sets: for all $\epsilon \geq 0$ $\mathcal{A}_+(z) \subseteq \mathcal{W}_+(z)$ and $\mathcal{A}_0(z) \subseteq \mathcal{W}_0(z)$. Of course the working and active sets coincide for $\epsilon = 0$. Further, the working and active sets coincide for a range of small $\epsilon$ values as established by the following observation.

**Observation 1.** *Suppose that $z$ is feasible for* (1)*. Then, for all $\epsilon \in [0, \min_{\ell=1,\ldots,m} \{z[\ell]_{n_\ell} \mid z[\ell]_{n_\ell} > 0\})$ it follows that $\mathcal{W}_+(z) = \mathcal{A}_+(z)$ and $\mathcal{W}_0(z) = \mathcal{A}_0(z)$.*

We use the working sets in place of the active sets for the numerical robustness of our algorithm. Note that the definition requires $z$ to be feasible. If we drop this requirement, then $\mathcal{W}_+(z)$ may also include infeasible constraints.

The tangent cone at a point of a feasible region is the closure of the cone of all feasible directions. Specifically in our case, Bonnans and Remírez [4, Lemma 25] have shown that the tangent cone of $w \in \mathcal{Q}_n$ is given by

$$T_{\mathcal{Q}_n}(w) = \begin{cases} \mathbb{R}^n & ||\bar{w}|| < w_n \\ \mathcal{Q}_n & w = 0 \\ u \in \mathbb{R}^n : \bar{u}^T\bar{w} - u_n w_n \leq 0 & \text{otherwise.} \end{cases} \tag{9}$$

Accordingly define $T(z) = T_{\mathcal{Q}}(z) = T_{\mathcal{Q}_{n_1}}(z[1]) \times \cdots \times T_{\mathcal{Q}_{n_m}}(z[m])$. The projected gradient is defined as the projection of the gradient onto the tangent cone; it is given by $P_{T(z)}(-\nabla q(z))$. The following lemma establishes a closed-form expression for evaluating the projected directional derivative $\frac{dq}{dt_+}(P_{\mathcal{Q}}(z - t\nabla q(z)))$ while avoiding the use of projected-gradient components that are known to be zero. For the relation of directional derivatives and the projected gradient, more generally, the reader may refer to McCormick and Tapia [14].

**Lemma 1.**

*(i) The projection of $d \in \mathbb{R}^n$ onto the tangent cone of $\mathcal{Q}_n$ at $w \in \mathcal{Q}_n$ is given by*

$$P_{T_{\mathcal{Q}_n}(w)}(d) = \begin{cases} d & d \in T_{\mathcal{Q}_n}(w), \\ P_{\mathcal{Q}_n}(d) & w = 0 \text{ and } d \notin \mathcal{Q}_n, \\ \frac{1}{2}\left(\bar{d} + \frac{d_n}{w_n}\bar{w}, \frac{\bar{w}^T\bar{d}}{w_n} + d_n\right) & \text{otherwise.} \end{cases} \tag{10}$$

*(ii) Suppose $z \in \mathcal{Q}$ and let $\nabla q = \nabla q(z)$. Then for all positive $\epsilon < \min_{i=2,\ldots,r}\{\tau_i - \tau_{i-1}\}$ the projected directional derivative of* (1) *at $z$ is given by*

$$\begin{aligned} \frac{dq}{dt_+}(P_{\mathcal{Q}}(z - t\nabla q)) = & \sum_{\substack{\ell \in \mathcal{W}_0(z): \\ -\nabla q[\ell] \notin \mathcal{Q}_{n_\ell} \cup -\mathcal{Q}_{n_\ell}}} \nabla q[\ell]^T P_{\mathcal{Q}_{n_\ell}}(-\nabla q[\ell]) - \sum_{\substack{\ell \in \{1,\ldots,m\}: \\ \ell \notin \mathcal{W}(z) \vee -\nabla q[\ell] \in \mathcal{Q}_{n_\ell}}} \nabla q[\ell]^T \nabla q[\ell] \\ & + \sum_{\ell \in \mathcal{W}_+(z): -\nabla q[\ell] \notin \mathcal{Q}_{n_\ell}} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]). \end{aligned} \tag{11}$$

*Proof.* (i) Let $w \in \mathcal{Q}_n$. Then the projected gradient is given by the (unique) optimum of

$$\min \left\{ ||d - x||^2 \ \mid \ x \in T_{\mathcal{Q}_n}(w) \right\}. \tag{12}$$

Note that this optimum exists, and let it be denoted by $x^*$. First consider the case that $d \in T_{\mathcal{Q}_n}(w)$. Then, $x^* = d$ is feasible with objective $||d - x^*|| = 0$, which is also optimal. Otherwise, consider the case that $w = 0$ and $||\bar{d}|| > d_n$. Then evidently $x^* = P_{T_{\mathcal{Q}_n}(w)}(d) = P_{\mathcal{Q}_n}(d)$ by (9) with $w = 0$. In all other cases, for a nonnegative Lagrange multiplier $\lambda \in \mathbb{R}$ consider the KKT first-order optimality conditions of (12) with $0 \neq ||\bar{w}|| = w_n$:

$$\bar{x} - \bar{d} - \lambda \bar{w} = 0, \qquad x_n - d_n + \lambda w_n = 0, \qquad \text{and} \qquad \lambda(\bar{x}^T \bar{w} - x_n w_n) = 0. \tag{13}$$

First we observe that the optimal Lagrange multiplier $\lambda^* > 0$. (If $\lambda^* = 0$, then $x^* = d$ with a 0 optimal objective value, but then as $w \in Q_n$ and $d = x^* \in T_{Q_n}(w)$.) Hence, $\lambda^* = \frac{d_n - x_n^*}{w_n} > 0$, and it then follows from (13) and from the fact that $||\bar{w}|| = w_n$ that $\bar{x}^* = \frac{1}{2}\left(\bar{d} + \frac{d_n}{w_n}\bar{w}\right)$ and $x_n^* = \frac{1}{2}\left(\frac{\bar{w}^T \bar{d}}{w_n} + d_n\right)$.

(ii) Suppose $z$ and $\epsilon$ that satisfy the supposition of the claim. First note that $z - \epsilon \nabla q$ is feasible and thus $-\nabla q \in T(z)$. Then, by [14, Proposition 1], the result of (i), and the projection formula (4), it follows that

$$\begin{aligned}
\frac{dq}{dt_+}(P_{\mathcal{Q}}(z - t\nabla q)) = {} & \sum_{\ell=1}^{m} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]) \quad = \sum_{\ell \in \mathcal{W}_0(z)} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]) \\
& - \sum_{\ell \in \{1,\dots,m\} \setminus \mathcal{W}(z)} \nabla q[\ell]^T \nabla q[\ell] + \sum_{\ell \in \mathcal{W}_+(z)} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]) \\
= {} & \sum_{\substack{\ell \in \mathcal{W}_0(z): \\ ||\overline{\nabla q[\ell]}|| \leq \nabla q[\ell]_{n_\ell}}} \nabla q[\ell]^T 0 + \sum_{\substack{\ell \in \mathcal{W}_0(z): \\ ||\overline{\nabla q[\ell]}|| > \nabla q[\ell]_{n_\ell}}} \nabla q[\ell]^T P_{\mathcal{Q}_{n_\ell}}(-\nabla q[\ell]) \\
& - \sum_{\ell \in \{1,\dots,m\} \setminus \mathcal{W}(z)} \nabla q[\ell]^T \nabla q[\ell] + \sum_{\ell \in \mathcal{W}_+(z)} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]) \\
= {} & \sum_{\substack{\ell \in \mathcal{W}_0(z): \\ -\nabla q[\ell] \notin \mathcal{Q}_{n_\ell} \cup -\mathcal{Q}_{n_\ell}}} \nabla q[\ell]^T P_{\mathcal{Q}_{n_\ell}}(-\nabla q[\ell]) - \sum_{\substack{\ell \in \{1,\dots,m\}: \\ \ell \notin \mathcal{W}(z) \vee -\nabla q[\ell] \in \mathcal{Q}_{n_\ell}}} \nabla q[\ell]^T \nabla q[\ell] \\
& + \sum_{\ell \in \mathcal{W}_+(z): -\nabla q[\ell] \notin \mathcal{Q}_{n_\ell}} \nabla q[\ell]^T P_{T_{\mathcal{Q}_{n_\ell}}(z[\ell])}(-\nabla q[\ell]). \square
\end{aligned}$$

Note that since (11) is expressed in terms of the working sets in place of the active sets, the claim of Lemma (ii) assumes that $\epsilon$ is sufficiently small (at most the distance of the closest pair of breakpoints according to the ordering (6)) within the corresponding projected-gradient iteration. However, even if this assumption is violated, the additive error of (11) can be bounded in terms of $N$ and $\epsilon$.

Further, note that the projected gradient could also be derived by using elementary calculus in cases in which the projection and $q$ are differentiable at a given point, by McCormick and Tapia [14, Proposition 2]; it applies to (4) other than in the case of $\bar{w} = 0$.

The projected-gradient algorithm is given by Algorithm 2. Note that in each iteration $j$ some of the components in $\mathcal{W}_0(z^{(j)})$ can be "masked out" and ignored in evaluating the directional derivative given by (11). Also, by using the identity (11), the projected gradient can be incrementally determined by updating the components corresponding to the constraints $\mathcal{W}_+(z^{(j)}) \setminus \mathcal{W}_+(z^{(j-1)})$ and $\mathcal{W}(z^{(j-1)}) \setminus \mathcal{W}_+(z^{(j)})$ and masking some of the components corresponding to the constraints $\mathcal{W}_0(z^{(j)}) \setminus \mathcal{W}_0(z^{(j-1)})$.

In the general case a Cauchy point cannot be determined analytically. If no (nonzero) conic constraint is active, then the Cauchy point is given by the same closed-form solution as in the bound-constrained case, and which is used in step 11 of Algorithm 2. Even in the general case, however, the identification of a projected-gradient segment that contains a Cauchy point can be used for making a line search more efficient when using the corresponding bounds on the step size. Accordingly we apply a standard backtracking projected line search to determine the step size $\alpha > 0$ starting from a given upper bound $\alpha_0$ [3]; in our case it is the distance to the next breakpoint following the projected-gradient curve at which the directional derivative becomes nonnegative. The projected backtracking line search is given by Algorithm 1.

**Algorithm 1** backtrack-linesearch$(z, p, \alpha_0)$

```
Input: $z, p \in \mathbb{R}^n, \alpha_0 > 0$ and $c \in (0,1)$
  $\alpha \leftarrow \min(\alpha_0, 1)$
  while $q(P_{\mathcal{Q}}(z + \alpha p)) > q(z) - \frac{1}{2\alpha} ||z - P_{\mathcal{Q}}(z + \alpha p)||^2$ do
    $\alpha \leftarrow c\alpha$
  end while
Output: $\alpha$
```

Lemma 2 establishes a lower bound for the step size that is the output of Algorithm 1. The bound established by the lemma immediately implies that Algorithm 1 terminates in a finite number of iterations.

**Lemma 2.** *The step size $\alpha$ that is the output of Algorithm 1 satisfies* $\alpha \geq \min\left\{1, \alpha_0, \frac{c}{||G||_2}\right\}$.

*Proof.* First observe that for every $z^{(1)}, z^{(2)} \in \mathbb{R}^N$ we have that the Lipschitz condition

$$\left|\left|\nabla q(z^{(1)}) - \nabla q(z^{(2)}\right|\right|_2 \leq \left|\left||G|(z^{(2)} - z^{(1)})\right|\right|_2 \leq ||G||_2 \left|\left|z^{(2)} - z^{(1)}\right|\right|_2$$

is satisfied with constant $||G||_2$. Hence, it follows from [3, Lemma 2] that for all $\alpha \leq \frac{1}{||G||_2}$, $\alpha \geq \frac{1}{2} ||z - P_{\mathcal{Q}}(z + \alpha p)||^2 / (q(z) - q(P_{\mathcal{Q}}(z + \alpha p)))$. Then the claim follows by Steps 1 and 3 of the algorithm. $\square$

An advantage of having precomputed the breakpoints (6) when searching over a given projected-gradient segment is that at most the projections onto active cones in $\mathcal{W}_+(z)$ need to be computed and updated at each iteration of Algorithm 1. The projection onto inactive and zero-active cones is immediately given, respectively by the first two cases of the definition in (4). Note that Algorithm 2 may revert to the backtracking line search (Algorithm 1) over the last segment in Step 13 of the algorithm. In particular, the last segment to be processed by Algorithm 2 is either the one starting at the $r$th breakpoint following the ordering of (6) or the first segment according to the same order over which the objective function decreases.

**Algorithm 2** PG$(z^{(0)}, p)$

```
Input: $z^{(0)}$ feasible for (1), descent direction $p$
 1: Generate the breakpoints $t_\ell^0$ for $\ell = 1, \dots, m$ and $t_\ell^+$ for $\ell = m_b + 1, \dots, m$.
 2: Sort the (unique) breakpoint values $\tau \in \mathbb{R}^r$ according to (6).
 3: for $j = 1, \dots, r$ do
 4:     $z^{(j)} \leftarrow P_{\mathcal{Q}}(z^{(0)} + \tau_j p), z^{(j+1)} \leftarrow P_{\mathcal{Q}}(z^{(0)} + \tau_{j+1} p)$ and $q_j \leftarrow q(z^{(j)})$, and $q_{j+1} \leftarrow q(z^{(j+1)})$.
 5:     $p[\ell] \leftarrow 0$ for $\ell \in \mathcal{W}_0(z^{(j)})$
 6:     if $\frac{dq}{dt_+}(P_{\mathcal{Q}}(z - t\nabla q)) > 0$ then
 7:         $z^* \leftarrow z^{(j)}$
 8:         break
 9:     else if $j = r$ or $q_{j+1} > q_j$ then
10:         if $\mathcal{W}_+(z^{(j)}) = \emptyset$ then
11:             $t^* \leftarrow -\frac{(Gz+g)^T p}{p^T G p}$
12:         else
13:             $t^* \leftarrow$ backtrack-linesearch$(z^{(j)}, p, \tau_{j+1} - \tau_j)$          // invoking Algorithm 1
14:         end if
15:         $z^* \leftarrow P_{\mathcal{Q}}(z^{(j)} + t^* p)$
16:         break
17:     end if
18: end for
Output: $z^*$
```

## 2.3 Phase II: Newton Iterations and EQP Subproblems

Having identified the optimal active set $\mathcal{A}(z^*)$, the reduced subspace problem is to solve (1) with conic constraints in $\mathcal{W}_+(z^*)$ replaced by equality constraints (constraining feasible points to lie on the boundary of the cone), and for all $\ell \in \mathcal{W}_0(z^*)$ fixing $z[\ell] = 0$. Although the reduced subproblem may seem like a more challenging nonconvex optimization problem, typically this problem has significantly fewer variables. Further, having identified a correct active set, then it will have only equality constraints that are differentiable. Given an active set, we apply a Newton-like method consisting of solving a sequence of EQPs [16, Chapter 18.2]. This method maintains the desirable properties of the subproblem, in particular that for all $\ell \in \mathcal{W}_+(z^*)$, $c_\ell(z)$ is differentiable for all $z \in \mathcal{B}_\epsilon(z^*) = \{x \in \mathbb{R}^N \mid ||x - z^*|| \leq \epsilon\}$ for some $\epsilon > 0$ (that is consistent with the choice of $\epsilon$ in (8)). Suppose that the active set $\mathcal{A}(z^*)$ is identified in a finite number of steps. Then, there exist an $\epsilon$ and $k < \infty$ such that $z^{(k)} \in \mathcal{B}_\epsilon(z^*)$, $c_\ell$ is differentiable for all $z \in B_\epsilon(z^{(k)})$, and this method starting at $z^{(k)}$ converges at a quadratic rate; see, for example, [16, Theorem 18.4].

For a given solution $z^{(k)} \in \mathbb{R}^N$ the EQP step consists of solving the subproblem

$$\min_z \quad q(z) \tag{14a}$$

$$\text{subject to} \quad z[\ell] = 0 \qquad \ell \in \mathcal{W}_0(z^{(k)}) \tag{14b}$$

$$c_\ell(z) = z[\ell]_{n_\ell} - \left|\left|\overline{z[\ell]}\right|\right| = 0 \qquad \ell \in \mathcal{W}_+(z^{(k)}). \tag{14c}$$

For $\ell \in \mathcal{W}_+(z^{(k)})$ we have that $\left|\left|\overline{z[\ell]}\right|\right| + \epsilon = z[\ell]_{n_\ell} > \epsilon$. Then, $\nabla c_\ell(z) = \begin{pmatrix} \frac{-z[\ell]_1}{||\overline{z[\ell]}||} & \cdots & \frac{-z[\ell]_{n_\ell}}{||\overline{z[\ell]}||} & 1 \end{pmatrix}^T$

and

$$\nabla^2 c_\ell(z) = \left(\begin{array}{c|c} C^\ell & 0 \\ \hline 0 & 0 \end{array}\right) \quad \text{with} \quad C^\ell_{ij} = \begin{cases} \frac{z[\ell]_i z[\ell]_j}{\left|\left|\overline{z[\ell]}\right|\right|^3} & i \neq j \\ \frac{-\left|\left|\overline{z[\ell]}\right|\right| + \frac{z[\ell]_i^2}{\left|\left|\overline{z[\ell]}\right|\right|}}{\left|\left|\overline{z[\ell]}\right|\right|^2} = \frac{1}{\left|\left|\overline{z[\ell]}\right|\right|}\left(-1 + \frac{z[\ell]_i^2}{\left|\left|\overline{z[\ell]}\right|\right|^2}\right) & \text{otherwise.} \end{cases}$$

Further, define the subvector $\hat{c}(z) = (c_\ell(z[\ell]))_{\ell \in \mathcal{W}_+(z)}$ and $\hat{n} = N - \sum_{\ell \in \mathcal{W}_0(z^{(k)})} n_\ell$. Let $\hat{q}$ be the reduced objective function for $z \in \mathbb{R}^{\hat{n}}$, and let $\hat{G}$ be the corresponding reduced Hessian in $\mathbb{R}^{\hat{n} \times \hat{n}}$. Then, the Lagrangian of (14) can be written as $\hat{L}(z, \lambda) = \hat{q}(z) - \lambda^T \hat{c}(z)$. Then at the $k$th Newton iteration, $\hat{p}^{(k)}, \hat{\lambda}^{(k)}$ are determined by the solution of the linear system:

$$\begin{pmatrix} \nabla^2 \hat{L}(z^{(k)}, \lambda^{(k)}) & -\nabla \hat{c}(z^{(k)}) \\ -\nabla \hat{c}(z^{(k)})^T & 0 \end{pmatrix} \begin{pmatrix} \hat{p}^{(k)} \\ \hat{\lambda}^{(k)} \end{pmatrix} = \begin{pmatrix} -\nabla \hat{q}(z^{(k)}) \\ \hat{c}(z^{(k)}) \end{pmatrix}. \tag{15}$$

Let $p^{(k)}$ denote the extension of $\hat{p}^{(k)}$ to $\mathbb{R}^N$ with $p^{(k)}[\ell] = 0$ for all $\ell \in \mathcal{W}_0(z^{(k)})$. Then the next iterate is given by $z^{(k+1)} = z^{(k)} + p^{(k)}$.

To ensure convergence of the Newton phase, for a penalty constant $\mu < 1/\max_{i \in \mathcal{W}(z^{(k)})} \lambda_i^{(k)}$, we define a merit function $m_k(z) = \mu^T \hat{q}(z) + \left|\left|(c_i(z))_{i \in \mathcal{W}(z^{(k)})}\right|\right|_1$ and require that the merit function decrease in every EQP step, that is, for $k \geq 2$,

$$m_k(z^{(k)}) < m_k(z^{(k-1)}), \tag{16}$$

and also, for a constant $\sigma \in (0, 1)$,

$$m_k(z^{(k)}) - m_k(z^{(k-1)}) \geq -\sigma \left(\mu \left(\frac{1}{2} p^{k^T} \nabla^2_{zz} \hat{L} p^{(k)} + \nabla_z \hat{L}^T p^{(k)}\right) + \left|\left|\hat{c}(z^{k-1})\right|\right|_1\right). \tag{17}$$

If one of the conditions (16) or (17) is violated, then the EQP step is rejected; otherwise it is accepted.

## 3 The Two-Phase Algorithm

The resulting two-phase method is given by Algorithm 3. The transition from the projected-gradient phase to the Newton phase occurs once the working set "settles down." The Newton phase consists of EQP steps in a subspace that is defined by the set of active variables and constraints. For EQP steps of the Newton phase to be effective, the starting point should also be sufficiently close to the optimal solution. In Step 7 of the algorithm it is determined whether the working set seems to have settled in order to switch to the Newton phase. It is determined by testing whether the step size is smaller than a threshold parameter $\tau_P$ in addition to the absence of a change to the working set in the most recent iteration (the latter is the sole criterion used for bound-constrained problems [15, 8]). The use of the step-size criterion is motivated by the observation that projected-gradient iterations with no change to the working set (i.e., with no additional constraints being identified) may be encountered before the working set settles down. This is illustrated by Example 2.

**Example 2.** *Let $m = 2, n_1 = 1$, and $n_2 = 3$, and consider Problem* (1) *with data and projected gradient phase iterates of Algorithm 3:*

$$G = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 2 \end{pmatrix} \qquad g = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}, \qquad \textit{and}$$

$$z^{(0)} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 5 \end{pmatrix}, \quad \nabla q(z^{(0)}) = \begin{pmatrix} -3 \\ 3 \\ 0 \\ 8 \end{pmatrix} \quad z^{(1)} = \begin{pmatrix} 4 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix}, \quad \nabla q(z^{(1)}) = \begin{pmatrix} 4.5 \\ 0.5 \\ -0.5 \\ -4.0 \end{pmatrix} \quad \textit{and} \quad z^{(2)} = \begin{pmatrix} 0 \\ 0.44 \\ 0.94 \\ 4.06 \end{pmatrix}$$

*Evidently,* $\mathcal{W}(z^{(0)}) = \{2\} = \mathcal{W}(z^{(1)}) \neq \mathcal{W}(z^{(2)}) = \{1\}$.

Example 2 shows that with SOCP constraints there may be projected-gradient iterates without a change to the active set while it is not yet settled down; this situation is in contrast to the case of active set methods for bound or linear constraints with an additional constraint being identified in each iteration.

In Step 11, Algorithm 3 reverts back to the projected-gradient phase from the Newton phase. This may occur when one of the following occurs:

- The inactive constraints become infeasible (beyond a predetermined tolerance), in which case the current working set is suspected to be incorrect.
- Conditions (16) or (17) are found to be violated in which case the EQP is not showing satisfactory progress. For example, this may be due to starting the Newton phase too far from the optimal solution.

If the algorithm reverts back to the projected-gradient phase, then the step-size tolerance parameter $\tau_P$, which determines the switch to the Newton phase, is halved in Step 12 of the algorithm in order to prevent undesirable oscillation between the gradient-projection and Newton phase. We note that when we return to the gradient projection phase, we start that phase at the projection of the last Newton step, provided that that step reduced the objective. Otherwise, we restart from the last projected-gradient step.

Proposition 1 establishes the finite termination of the projected-gradient phase of Algorithm 3. For its proof we use Lemma 3, which is due to Calamai and Moré [5, Lemma 3.1] in a general context of optimization over an arbitrary convex set. Lemma 3 specializes their result to a quadratic objective $q$ over the (convex) feasible region $\mathcal{Q}$.

**Lemma 3** (Calamai and Moré [5])**.** *Consider Problem* (1) *over the convex cone $\mathcal{Q}$. It follows that*

*(i)* $\min \left\{ \nabla q(z)^T v \mid v \in T(z), ||v|| \leq 1 \right\} = -\left|\left| P_{T(z)}(-\nabla q(z)) \right|\right|$.

*(ii) If $z^*$ is optimal for* (1), *then* $P_{T(z)}(-\nabla q(z^*)) = 0$.

**Proposition 1.** *Suppose that* (1) *is strictly feasible (i.e., the strict interior of the feasible region is not empty) and that strict complementarity holds (at every local minimum). Then for Algorithm 3 with $\tau_P = \tau_O$ (i.e., omitting the Newton phase) the following holds:*

*(i) The sequence $\{z^{(k)}\}$ of Algorithm 3 satisfies* $\lim_{k\to\infty} z^{(k)} \to z^*$ *for $z^*$ that is optimal for* (1).

**Algorithm 3** TwoPhasePg($G, g, z^{(0)}$): Two-Phase Projected-Gradient Method for Conic-Constrained QPs.

```
Input: $G \in \mathbb{R}^{N \times N}$, $g \in \mathbb{R}^N$, $z^{(0)} \in \mathcal{Q}$, tolerances $\tau_F, \tau_O, \tau_P \in (0,1)$, with $\tau_P \geq \tau_O$.
 1: $h \leftarrow 0$
 2: for $h = 1, \ldots$ do
 3:   $z^{(h)} \leftarrow$ PG-linesearch $\left(z^{(h-1)}, -\nabla q(z^{(h-1)})\right)$        // invoking Algorithm 2
 4:   if $\left|\left|z^{(h)} - z^{(h-1)}\right|\right| < \tau_O$ then
 5:     break
 6:   end if
 7:   if $\left|\left|z^{(h)} - z^{(h-1)}\right|\right| \leq \tau_P$ and $\mathcal{W}(z^{(h)}) = \mathcal{W}(z^{(h-1)})$ then
 8:     for k=h+1,... do
 9:       Perform an EQP iteration of (14), to obtain a solution $(p^{(k)}, \lambda^{(k)})$ of (15).
10:       $z^{(k)} \leftarrow z^{(k-1)} + p^{(k)}$
11:       if $\left|\left|\overline{z[i]}\right|\right| - z_{n_i} > \tau_F$ for some $i \notin \mathcal{W}(z^{(k)})$, or (16) is violated then
12:         $\tau_P \leftarrow \tau_p/2$
13:         break
14:       else if $\left|\left|\nabla L(z^{(k)}, \lambda^{(k)})\right|\right| < \tau_O$ then        // possible termination
15:         break
16:       end if
17:     end for
18:     if $q(P_{\mathcal{Q}}(z^{(k)})) > q(z^{(h)})$ then
19:       $z^{(k)} \leftarrow z^{(h)}$        // return to last PG iterate
20:     else
21:       $z^{(k)} \leftarrow P_{\mathcal{Q}}(z^{(k)})$        // continue from projected Newton
22:     end if
23:     $h \leftarrow k$
24:   end if
25: end for
Output: $z^{(k)}$
```

(ii) *Suppose $z^*$ is optimal for (1) and $0 < \epsilon < \min_{i=1,\ldots,m} \{z^*[\ell]_{n_\ell} \mid z^*[\ell]_{n_\ell} > 0\}$. Then, there exists a $K < \infty$ such that all $k \geq K$ satisfy $\mathcal{W}(z^{(k)}) = \mathcal{A}(z^*)$. In other words, the algorithm determines $\mathcal{A}(z^*)$ in a finite number of steps.*

*Proof.*

(i) By [3, Proposition 1] a limit point of the sequence generated by Algorithm 1 is a stationary point. Hence, and since the line search of Algorithm 2 terminates either at an exact Cauchy point or at a point generated by Algorithm 1, the limit point $z^*$ of the sequence $\{z^{(h)}\}$ generated by first phase of Algorithm 3, using Algorithm 2, is a stationary point. Then, since (1) is strictly feasible (and is convex), there exists a $\nu^* \in \mathcal{Q}$ so that $(z^*, \nu^*)$ is a solution of the optimality conditions (2) and $z^*$ is optimal for (1).

(ii) The following analysis is similar to [5, 15] for linearly and bound-constrained problems. (Note that if $n_\ell = 1$ for all $\ell$, then the algorithm and proof reduce to the bound-constrained case of [15] and the correctness of the statement follows, for example from [15, Theorem 5.2].) Consider an infinite sequence $\{z^{(k)}\}_{k=1}^{\infty}$ that it generated through Algorithm 3. To show that there exists some $K$ such that $\mathcal{W}(z^*) \subseteq \mathcal{W}(z^{(k)})$ for all $k \geq K$, assume for the sake of deriving a contradiction that there is some $\ell \in \mathcal{W}(z^*)$ such that $\ell \notin \mathcal{W}(z^{(k)})$ for all $k$.

Let $\nu^* \in \mathcal{Q}$ be KKT dual multipliers that correspond to $z^* \in \mathcal{Q}$, that is, $(z^*, \nu^*)$ is a solution of (2). Further, let $p \in \mathbb{R}^N$ be defined by

$$p[i] = \begin{cases} -z^*[i] & i = 1, \ldots, m, \text{ and } i \neq \ell, \\ \nu^*[i] & i = \ell, \end{cases} \tag{18}$$

and by (9) note that $-p \in T_{\mathcal{Q}}(z^{(k)})$ for all $k$; in particular $-p[i] = z^*[i] \in Q_{n_i} \subseteq T_{Q)n}(z^{(k)})$ for $i \neq \ell$ and $-p[i] = -\nu^*[i] \in \mathbb{R}^{n_i} \subseteq T_{Q_{n_i}}(z^{(k)})$ for $i = \ell \notin \mathcal{W}(z^{(k)})$. By Lemma 3(i)

$$\nabla q(z^{(k)})^T p \leq \left|\left| P_{T(z)}(-\nabla q(z^{(k)}) \right|\right| \, ||p|| \, .$$

Then, it follows from $\lim_{k\to\infty} z^{(k)} = z^*$ and Lemma 3(ii) that

$$\nabla q(z^*)^T p \leq 0.$$

By strict complementarity

$$\ell \in \mathcal{W}_+(z^*) \Rightarrow \ell \in \mathcal{W}_+(\nu^*) \qquad \text{or} \qquad \ell \in \mathcal{W}_0(z^*) \Rightarrow \ell \notin \mathcal{W}(\nu^*), \tag{19}$$

where the working sets $\mathcal{W}_+(\nu^*)$ and $\mathcal{W}_0(\nu^*)$ are defined by (8) from the self-duality of the second-order cone. This implies that in either case $\nu^*[\ell] \neq 0$. Then, by the KKT conditions (2) applied to $(z^*, \nu^*)$, (18), and complementary slackness, it follows that

$$\nabla q(z^*)^T p = {\nu^*}^T p = -\sum_{\substack{i=1 \\ i\neq\ell}}^{m} \nu^*[i]^T z^*[i] + ||\nu^*[\ell]||^2 = ||\nu^*[\ell]||^2 > 0,$$

thereby establishing a contradiction. Next, the equality of the working and active set at optimality follows by the choice of $\epsilon$ and Observation 1. □

Table 2: Comparison of performance statistics of our implementation of Algorithm 3 with the conic solver SDPT3 and nonlinear solver SQP-Filter over 30 runs using random data with well conditioned $G$ matrices.

| $N$ | $m_c$ | Dense | Algorithm 3 | | | | SDPT3 | | SQP-Filter | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | f-eval | $\nabla$-eval | $\nabla^2$-eval | CPU sec | Iter | CPU sec | f-eval | $\nabla$-eval | $\nabla^2$-eval | CPU sec |
| 100 | 20 | .1 | 146.1 | 88.6 | 0 | .16 | 22.9 | .99 | 498.7 | 186.4 | 220.2 | .30 |
| 100 | 20 | 1 | 128.4 | 82.0 | 0 | 0.15 | 24.5 | 1.62 | 508.1 | 184.0 | 215.1 | 4.38 |
| 200 | 20 | .1 | 152.0 | 95.7 | 0 | .24 | 22.7 | 4.80 | 416.8 | 157.5 | 181.8 | 3.80 |
| 200 | 20 | 1 | 144.0 | 97.4 | 0 | 0.20 | 24.2 | 7.99 | 466.1 | 171.3 | 197.0 | 34.92 |
| 500 | 50 | .1 | 228.0 | 170.3 | 0 | .52 | 26.0 | 15.39 | 467.6 | 190.1 | 226.5 | 57.66 |
| 500 | 50 | 1 | 224.97 | 180.0 | 0 | .72 | 26.8 | 25.86 | 478.8 | 191.8 | 229.0 | 417.0 |
| 1000 | 100 | .1 | 359.6 | 303.0 | 0 | 1.97 | 28.1 | 202.52 | 400.6 | 196.5 | 239.7 | 341.1 |
| 1000 | 100 | 1 | 364.6 | 322.3 | 0 | 5.87 | 28.9 | 133.69 | 446.5 | 228.2 | 273.9 | 2852.6 |
| 2000 | 100 | .1 | 431.0 | 390.23 | 0 | 3.97 | 27.1 | 309.44 | 429.5 | 207.3 | 241.7 | 1202.81 |
| 2000 | 100 | 1 | 495.0 | 460.0 | 0 | 23.99 | 28.1 | 756.13 | 374.5 | 178.1 | 212.3 | 8914.58 |

Proposition 1 implies that Algorithm 3 takes a finite number of iterations and invocations of Algorithm 2. By Lemma 2 each iteration of Algorithm 2 takes finitely many steps. Therefore, Algorithm 3 takes overall a finite number of steps to identify the active set of constraints. Following the convergence properties of the Newton's phase [16, Theorem 18.4], and the discussion in Section 2.3, Proposition 1 also establishes the quadratic convergence of Algorithm 3 for $\tau_P > \tau_O$ (i.e., when the Newton phase is applied near the optimal solution).

## 4 Computational Results

We now compare the computational performance of the two-phase Algorithm 3 with the conic solver SDPT3 and with the NLP solver SQP-Filter. The first set of well-conditioned instances is generated randomly as follows: $g$ is generated uniformly at random from $(-.5, .5)^N$, and a set of eigenvalues is generated uniformly at random and used to generate $G$ for some density in $\{.1, 1\}$ using the Matlab sprandsym procedure. Next, the randomly generated entries of $G$ and $g$ are truncated to have six significant digits. Note that this modification of the matrix entries also tends to increase the condition number of $G$. For the well-conditioned instances the intial set of eigenvalues is generated uniformly at random from $(0.5, 1)^N$, and so otherwise their condition number (before truncation) is bounded by 2. For a given $m_c$, a random number of elements is selected for each constraint in $\{2, \ldots, \lfloor (N - m_b)/m_c \rfloor + 1\}$. The last constraint contains all remaining variables. The computational results for the well-conditioned random instances are given in Table 2. The running time reported for SDPT3 is the total run time of CVX and SDPT3. Note that CVX converts (1) into a linear SOCP. Observe that in Table 2 the number of $\nabla^2$ evaluations is zero for Algorithm 3, indicating that for these instances we never compute an EQP step. The reason for this behavior is that for this class of well-conditioned problems, the projected-gradient method converges sufficiently well and that the steps are always larger than the switching tolerance $\tau_P$.

In Table 3 we show our results for random instances with matrices $G$ having more challenging condition numbers (although technically these would still be considered well conditioned): the set of eigenvalues is initially generated uniformly at random from $(0.5, 50)^N$. Since only six significant digits of the matrix were kept, the condition number tended to increase, and in practice the condition number varied between 90 and 400.

Table 3: Comparison of performance statistics of our implementation of Algorithm 3 with the conic solver SDPT3 and nonlinear solver SQP-Filter over 30 runs using random data with "poorly conditioned" $G$.

| $N$ | $m_c$ | Dense | Algorithm 3 | | | | SDPT3 | | SQP-Filter | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | f-eval | $\nabla$-eval | $\nabla^2$-eval | CPU sec | Iter | CPU sec | f-eval | $\nabla$-eval | $\nabla^2$-eval | CPU sec |
| 100 | 20 | .1 | 282.4 | 175.5 | 4.4 | .89 | 23.0 | 1.09 | 434.4 | 167.1 | 194.5 | .15 |
| 100 | 20 | 1 | 214.2 | 155.2 | 3.7 | .45 | 25.2 | 1.70 | 462.1 | 175.2 | 202.2 | 4.13 |
| 200 | 20 | .1 | 304.5 | 202.5 | 2.3 | 1.0 | 24.0 | 5.07 | 362.9 | 141.2 | 159.8 | 3.33 |
| 200 | 20 | 1 | 266.8 | 200.0 | 2.9 | 1.12 | 23.7 | 7.9 | 368.7 | 144.3 | 164.5 | 26.2 |
| 500 | 50 | .1 | 482.5 | 384.8 | 3.1 | 2.56 | 27.4 | 15.86 | 410.1 | 171.6 | 207.7 | 49.73 |
| 500 | 50 | 1 | 474.7 | 407.6 | 3.1 | 2.92 | 28.3 | 28.12 | 413.3 | 178.1 | 209.1 | 377.02 |
| 1000 | 100 | .1 | 832.4 | 719.0 | 5.3 | 7.08 | 30.0 | 67.09 | 360.3 | 182.7 | 216.9 | 306.71 |
| 1000 | 100 | 1 | 829.7 | 759.1 | 3.0 | 14.84 | 31.1 | 98.77 | 353.4 | 188.9 | 225.8 | 2249.81 |
| 2000 | 100 | .1 | 1262.6 | 1152.8 | 1.5 | 14.39 | 30.1 | 330.24 | 339.9 | 159.9 | 186.6 | 941.67 |
| 2000 | 100 | 1 | 1323.0 | 1251.1 | 2.4 | 65.86 | 31.7 | 838.49 | 323.8 | 155.5 | 183.2 | 7679.65 |

The results of Tables 2 and 3 indicate that although the number of iterations and running times of Algorithm 3 significantly increase with the condition number of $G$, the running times are an order of magnitude better that those of SDPT3 and SQP-Filter. Although the running times of both SDPT3 and SQP-Filter significantly increase for dense $G$, it does not appear to be the case for Algorithm 3, as can be expected from a first-order method. While the number of iterations of SDPT3 appears to grow slowly in $N$, the running times grow rapidly, consistent with previous results for interior-point methods [17].

## 5 Conclusions

In this paper we generalized the active-set methodology of bound-constrained quadratic programming (e.g., that of Moŕe and Toraldo [15]) to second-order cone-constrained quadratic programming. We proved that subject to strict complementarity, our method identifies an active set in a finite number of steps. The method is a two-phase algorithm that inherits the convergence properties of the second-phase method; Newton-like EQP steps that quadratically converge to an optimal solution. Computational experiments confirm that our method outperforms the conic interior-point solver SDPT3 and the nonlinear programming solver SQP-Filter. The relative advantage of our method increases with the dimension of the problem and the density of the objective's Hessian. The computational advantage of our method decreases with the condition number of the Hessian but remains significant even with condition numbers that exceed several hundreds.

We plan to extend our method to solve more general problems with additional linear constraints. In particular we will deploy the two-phase algorithm to solve the subproblem within an augmented Lagrangian method. We will also investigate cases in which specially structured linear constraints can be more directly incorporated within the projection steps of the algorithm. Moreover, we will extend the method for solving nonconvex conically constrained QPs.

**Acknowledgments.** We gratefully acknowledge support by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357 and by the U.S. Department of Energy through grant DE-FG02-05ER25694. The first author is also grateful for a discussion of active-set methods with Javier Peña. Gail Pieper is gratefully acknowledged for proof reading this manuscript.

## A Small SOCP Examples that Challenge NLP Solvers

Here we consider four basic four-dimensional examples of the form

$$\begin{array}{ll} \max & \begin{pmatrix} x & y_1 & y_2 & y_3 \end{pmatrix} G \begin{pmatrix} x & y_1 & y_2 & y_3 \end{pmatrix}^T + \begin{pmatrix} x & y_1 & y_2 & y_3 \end{pmatrix} g \\ \text{subject to} & ||y_1 + y_2|| \leq y_3 \\ & x \geq 0. \end{array}$$

In all four we have

$$G = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix}.$$

The four instances differ in the vector $g$ that is assigned as follows:

1. $g = \begin{pmatrix} 0 & 0 & -1 & -1 \end{pmatrix}$, for this example the conic constraint is active in the optimal solution.
2. $g = \begin{pmatrix} 0 & 0 & 0 & -1 \end{pmatrix}$, in which case the optimal solution is degenerate.
3. $g = \begin{pmatrix} 1 & 1 & 0 & -2 \end{pmatrix}$, in which case the conic constraint is inactive in the optimal solution.
4. $g = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$ for which the solution $(x, y^T) = \mathbf{0}$ is optimal.

## References

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

[2] H.H. Bauschke and S.G. Kruk. Reflection-projection method for convex feasibility problems with an obtuse cone. *Journal of Optimization Theory and Applications*, 120:503–531, 2004.

[3] D.P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, AC-21:174–184, 1976.

[4] J.F. Bonnans and H. Ramirez. Pertubation analysis of second-order cone programming problems. *Math. Programming*, 104:205–227, 2005.

[5] P.H. Calamai and J.J. Moré. Projected gradient methods for linearly constrained problems. *Math. Programming*, 39:93–116, 1987.

[6] A.R. Conn, N.I.M. Gould, and P.L. Toint. A globally convergent augmented langrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[7] S. Drewes and S. Ulbrich. Subgradient based outer approximation for mixed integer second order cone programming. *Mixed Integer Nonlinear Programming, Springer Series: The IMA Volumes in Mathematics and its Applications*, 154(1):41–59, 2012.

[8] M.P. Friedlander and S. Leyffer. Global and finite termination of a two-phase augmented lagrangian filter method for general quadratic programs. *SIAM Journal on Scientific Computing*, 30(4):1706–1729, 2008.

[9] O. Günlük and J. Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Math. Programming*, 124:183–205, 2010.

[10] C. Helmberg and F. Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Math. Programming*, 82(3):291–315, 1998.

[11] C. Kanzow, I. Ferenczi, and M. Fukushima. On the local convergence of semismooth Newton methods for linear and nonlinear second-order cone programs without strict complementarity. *SIAM Journal on Optimization*, 20:297–320, 2009.

[12] L. Kong and Q. Meng. A semismooth newton method for nonlinear symmetric cone programming. *Mathematical Methods of Operations Research*, 76(2):129–145, 2012.

[13] M. Laurent and S. Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223/224:431–461, 1995.

[14] G.P. McCormick and R.A. Tapia. The gradfient projection method under mild differentiability conditions. *SIAM Journal on Control*, 10:93–98, 1972.

[15] J.J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991.

[16] J. Norcedal and S.J. Wright. *Numerical Optimization.* Springer, Berlin, 2006.

[17] I. Pólik. Conic optimization software. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[18] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Programming*, 121(2):307–335, 2010.

[19] R.T. Rockafellar and R.J-B. Wets. *Variational analysis.* Springer, Berlin, 2009.

[20] J.F. Strum. Using SeDumi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[21] K.C. Toh, M.J. Todd, and R.H. Tutuncu. SDPT3—a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

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

[23] X-Y Zhao, D. Sun, and K-C Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.