A Semismooth Newton Method for Fast, Generic Convex Programming

Alnur Ali [*1] **Eric Wong** [*1] **J. Zico Kolter** [2]

Abstract

We introduce Newton-ADMM, a method for fast conic optimization. The basic idea is to view the residuals of consecutive iterates generated by the alternating direction method of multipliers (ADMM) as a set of fixed point equations, and then use a *nonsmooth* Newton method to find a solution; we apply the basic idea to the Splitting Cone Solver (SCS), a state-of-the-art method for solving generic conic optimization problems. We demonstrate theoretically, by extending the theory of *semismooth operators*, that Newton-ADMM converges rapidly (*i.e.*, quadratically) to a solution; empirically, Newton-ADMM is significantly faster than SCS on a number of problems. The method also has essentially no tuning parameters, generates certificates of primal or dual infeasibility, when appropriate, and can be specialized to solve specific convex problems.

1. Introduction and related work

Conic optimization problems (or *cone programs*) are convex optimization problems of the form

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad c^T x \qquad \text{subject to} \quad b - Ax \in \mathcal{K}, \tag{1}$$

where $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, $\mathcal{K}$ are problem data, specified by the user, and $\mathcal{K}$ is a *proper cone* (Nesterov & Nemirovskii, 1994; Ben-Tal & Nemirovski, 2001; Boyd & Vandenberghe, 2004); we give a formal treatment of proper cones in Section 2, but a simple example of a proper cone, for now, is the *nonnegative orthant*, *i.e.*, the set of all points in $\mathbf{R}^m$ with nonnegative components. These problems are quite general, encapsulating a number of standard problem classes: *e.g.*, taking $\mathcal{K}$ as the nonnegative orthant yields a linear program; taking $\mathcal{K}$ as the *positive semidefinite cone*, *i.e.*, the space of $m \times m$ positive semidefinite matrices $\mathbf{S}_+^m$, yields a semidefinite program; and taking $\mathcal{K}$ as the *second-order* (or *Lorentz*) *cone* $\{(x, y) \in \mathbf{R}^{m-1} \times \mathbf{R} : \|x\|_2 \leq y\}$ yields a second-order cone program (a quadratic program is a special case).

Due, in part, to their generality, cone programs have been the focus of much recent work, and additionally form the basis of many convex optimization modeling frameworks, *e.g.*, sdpsol (Wu & Boyd, 2000), YALMIP (Lofberg, 2005), and the CVX family of frameworks (Grant, 2004; Diamond & Boyd, 2016; Udell et al., 2014). These frameworks generally make it easy to quickly solve small and medium-sized convex optimization problems to high accuracy; they work by allowing the user to specify a generic convex optimization problem in a way that resembles its mathematical representation, then convert the problem into a form similar to (1), and finally solve the problem. Primal-dual interior point methods, *e.g.*, SeDuMi (Sturm, 2002), SDPT3 (Toh et al., 2012), and CVXOPT (Andersen et al., 2011), are common for solving these cone programs. These methods are useful, as they generally converge to high accuracy in just tens of iterations, but they solve a Newton system on each iteration, and so have difficulty scaling to high-dimensional (*i.e.*, large-n) problems.

In recent work, O'Donoghue et al. (2016) use the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) to solve generic cone programs; operator splitting methods (*e.g.*, ADMM, Peaceman-Rachford splitting (Peaceman & Rachford, 1955), Douglas-Rachford splitting (Douglas & Rachford, 1956), and dual decomposition) generally converge to modest accuracy in just a few iterations, so the approach (called the splitting conic solver, or SCS) is scalable, and also has a number of other benefits, *e.g.*, provding certificates of primal or dual infeasibility.

In this paper, we introduce a new method (called "Newton-ADMM") for solving large-scale, generic cone programs rapidly to high accuracy. The basic idea is to view the usual ADMM recurrence relation as a fixed point iteration, and then use a truncated, *nonsmooth* Newton method to find a fixed point; to justify the approach, we extend the theory of *semismooth operators*, coming out of the applied mathematics literature over the last two decades (Mifflin, 1977; Qi & Sun, 1993; Martínez & Qi, 1995; Facchinei et al.,

[*]Equal contribution [1]Machine Learning Department, Carnegie Mellon University [2]Computer Science Department, Carnegie Mellon University. Correspondence to: Alnur Ali <alnurali@cmu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

1996), although it has received little attention from the machine learning community (Ferris & Munson, 2004). We apply the approach to the fixed point iteration associated with SCS, to obtain a general purpose conic optimizer. We show, under regularity conditions, that Newton-ADMM is quadratically convergent; empirically, Newton-ADMM is significantly faster than SCS, on a number of problems. Also, Newton-ADMM has essentially no tuning parameters, and generates certificates of infeasibility, helpful in diagnosing problem misspecification.

The rest of the paper is organized as follows. In Section 2, we give the background on cone programs, SCS, and semismooth operators, required to derive our method for solving generic cone programs, Newton-ADMM. . In Section 3, we present Newton-ADMM, and establish some of its basic properties. In Section 4, we give various convergence guarantees. In Section 5, we empirically evaluate Newton-ADMM, and describe an extension as a specialized solver. We conclude with a discussion in Section 6.

2. Background

We first give some background on cones. Using this background, we go on to describe SCS, the cone program solver of O'Donoghue et al. (2016), in more detail. Finally, we give an overview of semismoothness (Mifflin, 1977), a generalization of smoothness, central to our Newton method.

2.1. Cone programming

We say that a set $\mathcal{C}$ is a *cone* if, for all $x \in \mathcal{C}$, and $\theta \geq 0$, we get that $\theta x \in \mathcal{C}$. The dual cone $\mathcal{C}^*$, associated with the cone $\mathcal{C}$, is defined as the set $\{y : y^T x \geq 0,\ \forall x \in \mathcal{C}\}$. Additionally, a cone $\mathcal{C}$ is a *convex cone* if, for all $x, y \in \mathcal{C}$, and $\theta_1, \theta_2 \geq 0$, we get that $\theta_1 x + \theta_2 y \in \mathcal{C}$. A cone $\mathcal{C}$ is a *proper cone* if it is (i) convex; (ii) closed; (iii) *solid*, *i.e.*, its interior is nonempty; and (iv) *pointed*, *i.e.*, if both $x, -x \in \mathcal{C}$, then we get that $x = 0$.

The nonnegative orthant, second-order cone, and positive semidefinite cone are all proper cones (Boyd & Vandenberghe, 2004, Section 2.4.1); these cones, along with the *exponential cone* (defined below), can be used to represent most convex optimization problems encountered in practice. The exponential cone (see, *e.g.*, Serrano (2015)), $\mathcal{K}_{\text{exp}}$, is a three-dimensional proper cone, defined as the closure of the epigraph of the perspective of $\exp(x)$, with $x \in \mathbf{R}$:

$$\mathcal{K}_{\text{exp}} = \{(x, y, z) : x \in \mathbf{R},\ y > 0,\ z \geq y \exp(x/y)\} \cup \{(x, 0, z) : x \leq 0,\ z \geq 0\}.$$

Cone programs resembling (1) were first described by Nesterov & Nemirovskii (1994, page 67), although special cases were, of course, considered earlier. Standard references include Ben-Tal & Nemirovski (2001) and Boyd & Vandenberghe (2004, Section 4.6.1).

2.2. SCS

Roughly speaking, SCS is an application of ADMM to a particular feasibility problem arising from the Karush-Kuhn-Tucker (KKT) optimality conditions associated with a cone program. To see this, consider a reformulation of the cone program (1), with slack variable $s \in \mathbf{R}^m$:

$$\underset{x \in \mathbf{R}^n,\, s}{\text{minimize}} \ c^T x \quad \text{subject to} \ \ Ax + s = b,\ s \in \mathcal{K}. \tag{2}$$

The KKT conditions can be seen, after introducing dual variables $r \in \mathbf{R}^n,\ y \in \mathcal{K}^*$, for the implicit constraint $x \in \mathbf{R}^n$ and the explicit constraints, respectively, to be

$$\begin{aligned} A^T y + c &= r && \text{(stationarity)} \\ Ax + s = b,\ s &\in \mathcal{K} && \text{(primal feasibility)} \\ r \in \{0\}^n,\ y &\in \mathcal{K}^* && \text{(dual feasibility)} \\ -c^T x - b^T y &= 0 && \text{(complementary slackness),} \end{aligned}$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$; thus, we can obtain a solution to (2), by solving the KKT system

$$\begin{bmatrix} 0 & A^T \\ -A & 0 \\ -c^T & -b^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ b \\ 0 \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}, \tag{3}$$
$$x \in \mathbf{R}^n,\ y \in \mathcal{K}^*,\ r \in \{0\}^n,\ s \in \mathcal{K}.$$

Self-dual homogeneous embedding. When the cone program (2) is primal/dual infeasible, there is no solution to the KKT system (3); so, consider embedding the system (3) in a larger system, with new variables τ, κ, and solving

$$\begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \tau \end{bmatrix} = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}, \tag{4}$$
$$x \in \mathbf{R}^n,\ y \in \mathcal{K}^*,\ \tau \in \mathbf{R}_+,\ r \in \{0\}^n,\ s \in \mathcal{K},\ \kappa \in \mathbf{R}_+,$$

which is always solvable. The embedding (4), due to Ye et al. (1994), has a number of other nice properties. Observe that when $\tau^\star = 1$, $\kappa^\star = 0$ are solutions to the embedding (4), we recover the KKT system (3); it turns out that the solutions $\tau^\star, \kappa^\star$ characterize the primal or dual (in)feasibility of the cone program (2). In particular, if $\tau^\star > 0$, $\kappa^\star = 0$, then the cone program (2) is feasible, with a primal-dual solution $(1/\tau^\star)(x^\star, y^\star, r^\star, s^\star)$; on the other hand, if $\tau^\star = 0$, $\kappa^\star \geq 0$, then (2) is primal or dual infeasible (or both), depending on the exact values of $\tau^\star$, $\kappa^\star$ (O'Donoghue et al., 2016, Section 2.3). The embedding (4) can also be seen as first-order homogeneous, in the sense that $(x^\star, y^\star, \tau^\star, r^\star, s^\star, \kappa^\star)$ being a solution to (4) implies that $k(x^\star, y^\star, \tau^\star, r^\star, s^\star, \kappa^\star)$, for $k \geq 0$, is also a solution. Finally, viewing the embedding (4) as a feasibility problem, the dual of the feasibility problem turns out to be the original feasibility problem, *i.e.*, the embedding is self-dual.

ADMM-based algorithm. As mentioned, the embedding (4) can be viewed as the feasibility problem

$$\text{find} \quad u, v \qquad \text{subject to} \quad Qu = v, \ (u, v) \in \mathcal{C} \times \mathcal{C}^*,$$

where we write $\mathcal{C} = \mathbf{R}^n \times \mathcal{K}^* \times \mathbf{R}_+$, $\mathcal{C}^* = \{0\}^n \times \mathcal{K} \times \mathbf{R}_+$,

$$Q = \begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix}, \quad u = \begin{bmatrix} x \\ y \\ \tau \end{bmatrix}, \quad v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}. \tag{5}$$

Introducing new variables $\tilde{u}, \tilde{v} \in \mathbf{R}^k$, where $k = n+m+1$, and rewriting so that we may apply ADMM, we get:

$$\begin{array}{ll} \underset{u,\, v,\, \tilde{u},\, \tilde{v}}{\text{minimize}} & I_{\mathcal{C} \times \mathcal{C}^*}(u, v) + I_{Qu^\star = v^\star}(\tilde{u}, \tilde{v}) \\ \text{subject to} & \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}, \end{array}$$

where $I_{\mathcal{C} \times \mathcal{C}^*}$ and $I_{Qu^\star = v^\star}$ are the indicator functions of the product space $\mathcal{C} \times \mathcal{C}^*$, and the affine space of solutions to $Qu = v$, respectively; after simplifying (see O'Donoghue et al. (2016, Section 3)), the ADMM recurrences are just

$$\tilde{u} \leftarrow (I + Q)^{-1}(u + v). \tag{6}$$
$$u \leftarrow P_{\mathcal{C}}(\tilde{u} - v) \tag{7}$$
$$v \leftarrow v - \tilde{u} + u, \tag{8}$$

where $P_{\mathcal{C}}$ denotes the projection onto $\mathcal{C}$. For the update (6), Q is a skew-symmetric matrix, hence $I + Q$ is nonsingular, so the update can be done efficiently via the Schur complement, matrix inversion lemma, and LDL^T factorization.

Projections onto dual cones. For the update (7), the projection onto $\mathcal{C}$ boils down to separate projections onto the "free" cone $\mathbf{R}^n$, the dual cone of $\mathcal{K}$, and the nonnegative orthant $\mathbf{R}_+$. These projections, for many $\mathcal{K}$, are well-known:

- *Free cone.* Here, $P_{\mathbf{R}^n}(z) = z$, for $z \in \mathbf{R}^n$.
- *Nonnegative orthant, $\mathcal{K}_{no}$.* The projection onto $\mathcal{K}_{\text{no}}$ is simply given by applying the positive part operator:

$$P_{\mathcal{K}_{\text{no}}}(z) = \max\{z, 0\}. \tag{9}$$

- *Second-order cone, $\mathcal{K}_{soc}$.* Write $z = (z_1, z_2) \in \mathbf{R}^m$, $z_1 \in \mathbf{R}^{m-1}$, $z_2 \in \mathbf{R}$. Then the projection is

$$P_{\mathcal{K}_{\text{soc}}}(z) = \begin{cases} 0, & \|z_1\|_2 \le -z_2 \\ z, & \|z_1\|_2 \le z_2 \\ \frac{1}{2}(1 + \frac{z_2}{\|z_1\|_2})(z_1, \|z_1\|_2), & \text{otherwise.} \end{cases} \tag{10}$$

- *Positive semidefinite cone, $\mathcal{K}_{psd}$.* The projection is

$$P_{\mathcal{K}_{\text{psd}}}(Z) = \sum_i \max\{\lambda_i, 0\} q_i q_i^T, \tag{11}$$

where $Z = \sum_i \lambda_i q_i q_i^T$ is the eigenvalue decomposition of Z.

- *Exponential cone, $\mathcal{K}_{exp}$.* If $z \in \mathcal{K}_{\text{exp}}$, then $P_{\mathcal{K}_{\text{exp}}}(z) = z$. If $-z \in \mathcal{K}_{\text{exp}}^*$, then $P_{\mathcal{K}_{\text{exp}}}(z) = 0$. If $z_1, z_2 < 0$, *i.e.*, the first two components of z are negative, then $P_{\mathcal{K}_{\text{exp}}} = (z_1, \max\{z_2, 0\}, \max\{z_3, 0\})$. Otherwise, the projection is given by

$$\begin{array}{ll} \underset{\tilde{z} \in \mathbf{R}^3 : \tilde{z}_2 > 0}{\text{argmin}} & (1/2)\|\tilde{z} - z\|_2^2 \\ \text{subject to} & \tilde{z}_2 \exp(\tilde{z}_1/\tilde{z}_2) = \tilde{z}_3, \end{array} \tag{12}$$

which can be computed using a Newton method (Parikh & Boyd, 2014, Section 6.3.4).

The nonnegative orthant, second-order cone, and positive semidefinite cone are all self-dual, so projecting onto these cones is equivalent to projecting onto their dual cones; to project onto the dual of the exponential cone, we use the Moreau decomposition to get

$$P_{\mathcal{K}_{\text{exp}}^*}(z) = z + P_{\mathcal{K}_{\text{exp}}}(-z). \tag{13}$$

2.3. Semismooth operators

Here, we give an overview of semismoothness; good references include Ulbrich (2011) and Izmailov & Solodov (2014). We consider maps $F : \mathbf{R}^k \to \mathbf{R}^k$ that are locally Lipschitz, *i.e.*, for all $z_1 \in \mathbf{R}^k$, and $z_2 \in \mathcal{N}(z_1, \delta)$, where $\mathcal{N}(z_1, \delta)$ is a ball centered at z_1 with radius $\delta > 0$, there exists some $L_{z_1} > 0$, such that $\|F(z_1) - F(z_2)\|_2 \le L_{z_1}\|z_1 - z_2\|_2$. By a result known as Rademacher's theorem (Evans & Gariepy, 2015, Section 3.1.2, Theorem 2), we get that F is differentiable almost everywhere; we let $\mathcal{D}$ denote the points at which F is differentiable, so that $\mathbf{R}^k \setminus \mathcal{D}$ is a set of measure zero.

The generalized Jacobian. Clarke (1990) suggested the *generalized Jacobian* as a way to define the derivative of a locally Lipschitz map $F : \mathbf{R}^k \to \mathbf{R}^k$, at all points. The generalized Jacobian is related to the subgradient, as well as the directional derivative, as we discuss later on; the generalized Jacobian, though, turns out to be quite useful for defining effective nonsmooth Newton methods. The generalized Jacobian $\mathcal{J}(z)$ at a point $z \in \mathbf{R}^k$ of a map $F : \mathbf{R}^k \to \mathbf{R}^k$, is defined as (**co** denotes convex hull)

$$\mathcal{J}(z) = \mathbf{co}\left\{ \lim_{i \to \infty} J(z_i) : (z_i) \in \mathcal{D}, \ (z_i) \to z \right\}, \tag{14}$$

where $J(z_i) \in \mathbf{R}^{k \times k}$ is the usual Jacobian of F at z_i. Two useful properties of the generalized Jacobian (Clarke, 1990,

Proposition 1.2): (i) $\mathcal{J}(z)$, at any z, is always nonempty; and (ii) if each component F_i is convex, then the ith row of any element of $\mathcal{J}(z)$ is just a subgradient of F_i at z.

(Strong) semismoothness and consequences. We say that a map $F : \mathbf{R}^k \to \mathbf{R}^k$ is semismooth if it is locally Lipschitz, and if, for all $z, \delta \in \mathbf{R}^k$, the limit

$$\lim_{\delta \to 0,\ J \in \mathcal{J}(z+\delta)} J\delta \tag{15}$$

exists (see, *e.g.*, Mifflin (1977, Definition 1) and Qi & Sun (1993, Section 2)). The above definition is somewhat opaque, so various works have provided an alternative characterization of semismoothness: F is semismooth if and only if it is (i) locally Lipschitz; (ii) directionally differentiable, in every direction; and (iii) we get

$$\lim_{\delta \to 0,\ J \in \mathcal{J}(z+\delta)} \frac{\|F(z+\delta) - F(z) - J\delta\|_2}{\|\delta\|_2} = 0,$$

i.e., $\|F(z+\delta) - F(z) - J\delta\|_2 = o(\|\delta\|_2),\ \delta \to 0$ (see, *e.g.*, Qi & Sun (1993, Theorem 2.3), Hintermüller (2010, Theorem 2.9), Qi & Sun (1999, page 2), and Martínez & Qi (1995, Proposition 2)). Examples of semismooth functions include $\log(1 + |x|)$, all convex functions, and all smooth functions (Mifflin, 1977; Śmietański, 2007); on the other hand, $\sqrt{|x|}$ is not semismooth. A linear combination of semismooth functions is semismooth (Izmailov & Solodov, 2014, Proposition 1.75). Finally, we say that a map is *strongly semismooth* if, under the same conditions as above, we can replace (15) with

$$\limsup_{\delta \to 0,\ J \in \mathcal{J}(z+\delta)} \frac{\|F(z+\delta) - F(z) - J\delta\|_2}{\|\delta\|_2^2} < \infty,$$

i.e., $\|F(z+\delta) - F(z) - J\delta\|_2 = O(\|\delta\|_2^2),\ \delta \to 0$ (see Facchinei et al. (1996, Proposition 2.3) and Facchinei & Kanzow (1997, Definition 1)).

3. Newton-ADMM and its basic properties

Next, we describe Newton-ADMM, our nonsmooth Newton method for generic convex programming; again, the basic idea is to view the ADMM recurrences (6) – (8), used by SCS, as a fixed point iteration, and then use a nonsmooth Newton method to find a fixed point. Accordingly, we let

$$F(z) = \begin{bmatrix} \tilde{u} - (I+Q)^{-1}(u+v) \\ u - P_{\mathcal{C}}(\tilde{u} - v) \\ \tilde{u} - u \end{bmatrix},$$

which are just the residuals of the consecutive ADMM iterates given by (6) – (8), and $z = (\tilde{u}, u, v) \in \mathbf{R}^{3k}$; multiplying by $\mathbf{diag}(I+Q, I, I)$ to change coordinates gives

$$F(z) = \begin{bmatrix} (I+Q)\tilde{u} - (u+v) \\ u - P_{\mathcal{C}}(\tilde{u} - v) \\ \tilde{u} - u \end{bmatrix}. \tag{16}$$

Now, we would like to apply a Newton method to F, but projections onto proper cones are not differentiable, in general. However, for many cones of interest, they are (strongly) semismooth; the following lemma summarizes.

Lemma 3.1. *Projections onto the nonnegative orthant, second-order cone, and positive semidefinite cone are all strongly semismooth; see, e.g., Kong et al. (2009, Section 1), Kanzow & Fukushima (2006, Lemma 2.3), and Sun & Sun (2002, Corollary 4.15), respectively.*

Additionally, we give the following new result, for the exponential cone, which may be of independent interest.

Lemma 3.2. *The projection onto the exponential cone is semismooth.*

We defer all proofs to the supplement.

Putting the pieces together, the following lemma establishes that F, defined in (16), is (strongly) semismooth.

Lemma 3.3. *When $\mathcal{K}$, from the cone program (1), is the nonnegative orthant, second-order cone, or positive semidefinite cone, then the map F, defined in (16), is strongly semismooth; when $\mathcal{K}$ is the exponential cone, then the map F is semismooth.*

The preceding results lay the groundwork for us to use a semismooth Newton method (Qi & Sun, 1993), applied to F, where we replace the usual Jacobian with any element of the generalized Jacobian (14); however, as many have observed (Khan & Barton, 2017), it is not always straightforward to compute an element of the generalized Jacobian. Fortunately, for us, we can just compute a subgradient of each row of F, as the following lemma establishes.

Lemma 3.4. *The ith row of each element of the generalized Jacobian $\mathcal{J}(z)$ at z of the map F is just a subgradient of $F_i,\ i = 1, \ldots, 3k$, at z.*

Using the lemma, an element $J \in \mathbf{R}^{3k \times 3k}$ of the generalized Jacobian of the map $F \in \mathbf{R}^{3k}$ is then just

$$J = \begin{bmatrix} I+Q & -I & -I \\ & J_u & \\ I & -I & 0 \end{bmatrix}, \tag{17}$$

where

$$J_u = \begin{bmatrix} -I & 0 & 0 & I & 0 & 0 & I & 0 & 0 \\ 0 & -J_{P_{\mathcal{K}^*}} & 0 & 0 & I & 0 & 0 & J_{P_{\mathcal{K}^*}} & 0 \\ 0 & 0 & -\ell & 0 & 0 & 1 & 0 & 0 & \ell \end{bmatrix} \tag{18}$$

is a $(k \times 3k)$-dimensional matrix forming the second row of J; ℓ equals 1 if $\tilde{u}_\tau - v_\kappa \geq 0$ and 0 otherwise; and $J_{P_{\mathcal{K}^*}} \in \mathbf{R}^{m \times m}$ is the Jacobian of the projection onto the dual cone $\mathcal{K}^*$. Here and below, we use subscripts to select components, *e.g.*, $\tilde{u}_\tau$ selects the τ-component of $\tilde{u}$ from (5), and we write J to mean $J(z)$, where $z = (\tilde{u}, u, v) \in \mathbf{R}^{3k}$.

3.1. Final algorithm

Later, we discuss computing $J_{P_{\mathcal{K}^*}}$, the Jacobian of the projection onto the dual cone $\mathcal{K}^*$, for various cones $\mathcal{K}$; these pieces let us compute an element J, given in (17) – (18), of the generalized Jacobian of the map F, defined in (16), which we use instead of the usual Jacobian, in a semismooth Newton method; below, we describe a way to scale the method to larger problems (*i.e.*, values of n).

Truncated, semismooth Newton method. The conjugate gradient method is, seemingly, an appropriate choice here, as it only approximately solves the Newton system

$$J\Delta = -F, \tag{19}$$

with variable $\Delta \in \mathbf{R}^{3k}$; unfortunately, in our case, J is nonsymmetric, so we appeal instead to the generalized minimum residual method (GMRES) (Saad & Schultz, 1986). We run GMRES until

$$\|F + J\hat{\Delta}\|_2 \leq \varepsilon \|F\|_2, \tag{20}$$

where $\hat{\Delta}$ is the approximate solution from a particular iteration of GMRES, and ε is a user-defined tolerance; *i.e.*, we run GMRES until the approximation error is acceptable. After GMRES computes an approximate Newton step, we use backtracking line search to compute a step size.

Now recall, from Section 2, that $\Delta^\star = 0$ is always a trivial solution to the Newton system (19), due to homogeneity; so, we initialize the $\tilde{u}_\tau,\ u_\tau,\ v_\kappa$-components of z to 1, which avoids converging to the trivial solution. Finally, we mention that when $\mathcal{K}$, in the cone program (1), is the direct product of several proper cones, then J_u, in (18), simply consists of multiple such matrices, just stacked vertically.

We describe the entire method in Algorithm 1. The method has essentially no tuning parameters, since, for all the experiments, we just fix the maximum number of Newton iterations $T = 100$; the backtracking line search parameters $\alpha = 0.001,\ \beta = 0.5$; and the GMRES tolerances $\varepsilon^{(i)} = 1/(i+1)$, for each Newton iteration i. The cost of each Newton iteration is the number of backtracking line search iterations times the sum of two costs: the cost of projecting onto a dual cone and the cost of GMRES, *i.e.*, $O(\max\{n^2, m^2\})$, assuming GMRES returns early. Similarly, the cost of each ADMM iteration of SCS is the cost of projecting onto a dual cone plus $O(\max\{n^2, m^2\})$.

3.2. Jacobians of projections onto dual cones

Here, we derive the Jacobians of projections onto the dual cones of the nonnegative orthant, second-order cone, positive semidefinite cone, and the exponential cone; here, we write $J_{P_{\mathcal{K}^*}}$ to mean $J_{P_{\mathcal{K}^*}}(z)$, where $z = \tilde{u}_y - v_s \in \mathbf{R}^m$.

Algorithm 1 Newton-ADMM for convex optimization

Input: problem data $c \in \mathbf{R}^n,\ \mathcal{A} \in \mathbf{R}^{m\times n},\ b \in \mathbf{R}^m$; cones $\mathcal{K}$; maximum number of Newton iterations T; backtracking line search parameters $\alpha \in (0, 1/2),\ \beta \in (0,1)$; GMRES approximation tolerances $(\varepsilon^{(i)})_{i=1}^T$
Output: a solution to (2)
initialize $\tilde{u}^{(1)} = u^{(1)} = v^{(1)} = 0$ and $\tilde{u}_\tau^{(1)} = u_\tau^{(1)} = v_\kappa^{(1)} = 1$ // avoids trivial solution
initialize $z^{(1)} = (\tilde{u}^{(1)}, u^{(1)}, v^{(1)})$
for $i = 1, \ldots, T$ **do**
  compute $J(z^{(i)}),\ F(z^{(i)})$ // see (16), (17), Sec. 3.2
  compute the Newton step $\Delta^{(i)}$, *i.e.*, by approximately solving $J(z^{(i)})\Delta^{(i)} = -F(z^{(i)})$ using GMRES with approximation tolerance $\varepsilon^{(i)}$ // see (20)
  initialize $t^{(i)} = 1$ // initialize step size $t^{(i)}$
  while $\|F(z^{(i)} + t^{(i)}\Delta^{(i)})\|_2^2 \geq (1 - \alpha t^{(i)})\|F(z^{(i)})\|_2^2$ **do**
    $t^{(i)} = \beta t^{(i)}$ // for backtracking line search
  end while
  update $z^{(i+1)} = z^{(i)} + t^{(i)}\Delta^{(i)}$
end for
return the u_x- divided by the u_τ-components of $z^{(T)}$

Nonnegative orthant. Since the nonnegative orthant is self-dual, we can simply find a subgradient of each component in (9), to get that $J_{P_{\mathcal{K}^*}}$ is diagonal with, say, $(J_{P_{\mathcal{K}^*}})_{ii}$ set to 1 if $(\tilde{u}_y - v_s)_i \geq 0$ and 0 otherwise, for $i = 1, \ldots, m$.

Second-order cone. Write $z = (z_1, z_2),\ z_1 \in \mathbf{R}^{m-1},\ z_2 \in \mathbf{R}$. The second-order cone is self-dual, as well, so we can find subgradients of (10), to get that

$$J_{P_{\mathcal{K}^*}} = \begin{cases} 0, & \|z_1\|_2 \leq -z_2 \\ I, & \|z_1\|_2 \leq z_2 \\ D, & \text{otherwise,} \end{cases} \tag{21}$$

where D is a low-rank matrix (details in the supplement).

Positive semidefinite cone. The projection map onto the (self-dual) positive semidefinite cone is matrix-valued, so computing the Jacobian is more involved. We leverage the fact that most implementations of GMRES need only the product $J_{P_{\mathcal{K}^*}}(\mathbf{vec}\, Z)$, provided by the below lemma using matrix differentials (Magnus & Neudecker, 1995); here, **vec** is the vectorization of a real, symmetric matrix Z.

Lemma 3.5. *Let $Z = Q\Lambda Q^T$ be the eigenvalue decomposition of Z, and let $\tilde{Z}$ be a real, symmetric matrix. Then*

$$J_{P_{\mathcal{K}_{psd}}}(\mathbf{vec}\, Z)(\mathbf{vec}\, \tilde{Z}) = \mathbf{vec}\left((dQ)\max(\Lambda, 0)Q^T + Q(d\max(\Lambda, 0))Q^T + Q\max(\Lambda, 0)(dQ)^T\right),$$

where, here, the max *is interpreted diagonally;*

$$dQ_i = (\Lambda_{ii}I - Z)^+ \tilde{Z} Q_i;\ [d\max(\Lambda, 0)]_{ii} = I_+(\Lambda_{ii})Q_i^T \tilde{Z} Q_i;$$

Z^+ denotes the pseudo-inverse of Z; and $I_+(\cdot)$ is the indicator function of the nonnegative orthant.

Exponential cone. Recall, from (12), that the projection onto the exponential cone is not analytic, so computing the Jacobian is much more involved, as well. The following lemma provides a Newton method for computing the Jacobian, using the KKT conditions for (12) and differentials.

Lemma 3.6. *Let $z \in \mathbf{R}^3$. Then $J_{P_{\mathcal{K}^*_{exp}}}(z) = I - J_{P_{\mathcal{K}_{exp}}}(-z)$, where*

$$J_{P_{\mathcal{K}_{exp}}}(z) = \begin{cases} I, & z \in \mathcal{K}_{exp} \\ -I, & z \in \mathcal{K}^*_{exp} \\ \mathbf{diag}(1, I_+(z_2), I_+(z_3)), & z_1, z_2 < 0; \end{cases}$$

otherwise, $J_{P_{\mathcal{K}_{exp}}}(z)$ is a particular 3x3 matrix given in the supplement, due to space constraints.

4. Convergence guarantees

Here, we give some convergence results for Newton-ADMM, the method presented in Algorithm 1.

First, we show that, under standard regularity assumptions, the iterates $(z^{(i)})_{i=1}^{\infty}$ generated by Algorithm 1 are *globally convergent*, *i.e.*, given some initial point, the iterates converge to a solution of $F(z) = 0$, where i is a Newton iteration counter. We break the statement (and proof) of the result up into two cases. Theorem 4.1 establishes the result, when the sequence of step sizes $(t^{(i)})_{i=1}^{\infty}$ converges to some number bounded away from zero and one. Theorem 4.2 establishes the result when the step sizes converge zero.

Below, we state our regularity conditions, which are similar to those given in Han et al. (1992); Martínez & Qi (1995); Facchinei et al. (1996); we elaborate in the supplement.

A1. For Theorem 4.1, we assume $\limsup_{i\to\infty} t^{(i)} < 1$.

A2. For Theorem 4.2, we assume $\limsup_{i\to\infty} t^{(i)} = 0$.

A3. For Theorem 4.2, we assume (i) that the GMRES approximation tolerances $\varepsilon^{(i)}$ are uniformly bounded by ε as in $\varepsilon^{(i)} \leq \varepsilon < 1 - \alpha^{1/2}$, (ii) that $(\varepsilon^{(i)})_{i=1}^{\infty} \to 0$, and (iii) that $\varepsilon^{(i)} = O(\|F(z^{(i)})\|_2)$.

A4. For Theorem 4.2, we assume, for every convergent sequence $(z^{(i)})_{i=1}^{\infty} \to z$, $(\gamma^{(i)})_{i=1}^{\infty}$ satisfying assumption (A2) above, and $(\Delta^{(j)})_{j=1}^{\infty} \to \Delta$, that

$$\lim_{i,j\to\infty} \frac{\|F(z^{(i)} + \gamma^{(i)}\Delta^{(j)})\|_2^2 - \|F(z^{(i)})\|_2^2}{\gamma^{(i)}} \leq \lim_{i,j\to\infty} \alpha^{1/2} F(z^{(i)})^T \hat{F}(z^{(i)}, \Delta^{(j)}),$$

where, for notational convenience, we write

$$\hat{F}(z^{(i)}, \Delta^{(j)}) = J(z^{(i)})\Delta^{(j)}.$$

A5. For Theorem 4.2, we assume, for all $z \in \mathbf{R}^{3k}$ and $\Delta \in \mathbf{R}^{3k}$, and for some $C_2 > 0$, that

$$C_2\|\Delta\|_2 \leq \|\hat{F}(z, \Delta)\|_2.$$

A6. For Theorem 4.3, we assume, for all $z \in \mathbf{R}^{3k}$, $J(z) \in \mathcal{J}(z)$, (i) that $\|J(z)\|_2 \leq C_3$, for some constant $C_3 > 0$; and (ii) that every element of $\mathcal{J}(z)$ is invertible.

The two global convergence results are given below; the proofs are based on arguments in Martínez & Qi (1995, Theorem 5a), but we use fewer user-defined parameters, and a different line search method.

Theorem 4.1 (Global convergence, with $\limsup_{i\to\infty} t^{(i)} = t$, for some $0 < t < 1$)**.** *Assume condition (A1) stated above. Then $\lim_{i\to\infty} F(z^{(i)}) = 0$.*

Theorem 4.2 (Global convergence, with $\limsup_{i\to\infty} t^{(i)} = 0$)**.** *Assume conditions (A2), (A3), (A4), and (A5) stated above. Suppose the sequence $(z^{(i)})_{i=1}^{\infty}$ converges to some $z \in \mathbf{R}^{3k}$. Then $F(z) = 0$.*

Next, we show, in Theorem 4.3, that when F is strongly semismooth, *i.e.*, $\mathcal{K}$ is the nonnegative orthant, second-order cone, or positive semidefinite cone, the iterates $(z^{(i)})_{i=1}^{\infty}$ generated by Algorithm 1 are locally quadratically convergent; the proof is similar to that of Facchinei et al. (1996, Theorem 3.2b), for semismooth maps.

Theorem 4.3 (Local quadratic convergence)**.** *Assume condition (A6) stated above. Then the sequence of iterates $(z^{(i)})_{i=1}^{\infty} \to z$ generated by Algorithm 1 converges quadratically, with $F(z) = 0$, for large enough i.*

When $\mathcal{K}$ is the exponential cone, *i.e.*, F is semismooth, the iterates generated by Algorithm 1 are locally superlinearly convergent (Facchinei et al., 1996, Theorem 3.2b).

5. Numerical examples

Next, we present an empirical evaluation of Newton-ADMM, on several problems; in these, we directly compare to SCS, which Newton-ADMM builds on, as it is the most relevant benchmark for us (O'Donoghue et al. (2016) observe that, with an optimized implementation, SCS outperforms SeDuMi, as well as SDPT3). We evaluate, for both methods, the time taken to reach the solution as well as the optimal objective value; we obtained these by running an interior point method (Andersen et al., 2011) to high accuracy. Table 1 describes the problem sizes, for both the cone form of (1), as well as the familiar form that the problem is usually written in. Later, we also describe extending Newton-ADMM to accelerate any ADMM-based algorithm, applied to *any* convex problem; here, we compare to state-of-the-art baselines for specific problems.

Table 1: Problem sizes, for the cone form (n, m) of (1), and the familiar form (p, N) that the problem is usually written in.

PROBLEM	n	m	p	N	CONES
LINEAR PROG.	600	1,200	600	300	$\mathcal{K}_{\text{no}}$
PORTFOLIO OPT.	2,501	2,504	2,500	–	$\mathcal{K}_{\text{soc}}, \mathcal{K}_{\text{no}}$
LOGISTIC REG.	3,200	7,200	100	1,000	$\mathcal{K}_{\text{exp}}, \mathcal{K}_{\text{no}}$
ROBUST PCA	4,376	8,103	25	25	$\mathcal{K}_{\text{psd}}, \mathcal{K}_{\text{no}}$

5.1. Random linear programs (LPs)

We compare Newton-ADMM and SCS on a linear program

$$\underset{x\in\mathbf{R}^p}{\text{minimize}} \quad c^T x \qquad \text{subject to} \quad Gx = h,\ x \geq 0,$$

where $c \in \mathbf{R}^p$, $G \in \mathbf{R}^{N\times p}$, $h \in \mathbf{R}^N$ are problem data, and the inequality is interpreted elementwise. To ensure primal feasibility, we generated a solution $x^\star$ by sampling its entries from a normal distribution, then projecting onto the nonnegative orthant; we generated G (with $p = 600,\ N = 300$, so G is wide) by sampling entries from a normal distribution, then taking $h = Gx^\star$. To ensure dual feasibility, we generated dual solutions $\nu^\star,\ \lambda^\star$, associated with the equality and inequality constraints, by sampling their entries from a normal and Uniform$(0, 1)$ distribution, respectively; to ensure complementary slackness, we set $c = -G^T\nu^\star + \lambda^\star$. Finally, to put the linear program into the cone form of (1), and hence (2), we just take

$$A = \begin{bmatrix} G \\ -G \\ I \end{bmatrix}, \quad b = \begin{bmatrix} h \\ -h \\ 0 \end{bmatrix}, \quad \mathcal{K} = \mathcal{K}_{\text{no}}.$$

The first column of Figure 1 presents the time taken, by both Newton-ADMM and SCS, to reach the optimal objective value, as well as to reach the solution; we see that Newton-ADMM outperforms SCS in both metrics.

5.2. Minimum variance portfolio optimization

We consider a *minimum variance portfolio optimization* problem (see, *e.g.*, Khare et al. (2015); Ali et al. (2016)),

$$\underset{\theta\in\mathbf{R}^p}{\text{minimize}} \quad \theta^T\Sigma\theta \qquad \text{subject to} \quad \mathbf{1}^T\theta = 1, \tag{22}$$

where, here, the problem data $\Sigma \in \mathbf{S}^p_{++}$ is the covariance matrix associated with the prices of $p = 2,500$ assets; we generated Σ by sampling a positive definite matrix. The goal of the problem is to allocate wealth across p assets such that the overall risk is minimized; shorting is allowed. Putting the above problem into the cone form of (1) yields, for $\mathcal{K}$, the direct product of the second-order cone and the nonnegative orthant (details in the supplement). The second column of Figure 1 shows the results; we again see that Newton-ADMM outperforms SCS.

5.3. ℓ_1-penalized logistic regression

We consider ℓ_1-penalized logistic regression, *i.e.*,

$$\underset{\theta\in\mathbf{R}^p}{\text{minimize}} \quad \sum_{i=1}^N \log(1 + \exp(y_i X_{i\cdot}\theta)) + \lambda\|\theta\|_1, \tag{23}$$

where, here, $y \in \mathbf{R}^N$ here is a response vector; $X \in \mathbf{R}^{N\times p}$ is a data matrix, with $X_{i\cdot}$ denoting the ith row of X; and $\lambda \geq 0$ is a tuning parameter. We generated $p = 100$ sparse underlying coefficients $\theta^\star$, by sampling entries from a normal distribution, then setting $\approx 90\%$ of the entries to zero; we generated X (with $N = 1,000$) by sampling its entries from a normal distribution, then set $y = X\theta^\star + \delta$, where δ is (additive) Gaussian noise. For simplicity, we set the tuning parameter $\lambda = 1$. Putting the above problem into the cone form of (1) yields, for $\mathcal{K}$, the direct product of the exponential cone and the nonnegative orthant (details in the supplement); the problem size in cone form ends up being large (see Table 1). In the third column of Figure 1, we see that Newton-ADMM outperforms SCS.

5.4. Robust principal components analysis (PCA)

Finally, we consider robust PCA,

$$\underset{L,S\in\mathbf{R}^{N\times p}}{\text{minimize}} \quad \|L\|_* \ \text{ subject to } \|S\|_1 \leq \lambda,\ L + S = X, \tag{24}$$

where $\|\cdot\|_*$ and $\|\cdot\|_1$ are the nuclear and elementwise ℓ_1-norms, respectively, and $X \in \mathbf{R}^{N\times p},\ \lambda \geq 0$ (Candès et al., 2011, Equation 1.1). We generated a low-rank matrix $L^\star$, with rank $\approx \frac{1}{2}N$; a sparse matrix $S^\star$, by sampling entries from Uniform$(0, 1)$, then setting $\approx 90\%$ of the entries to zero; and finally set $X = L^\star + S^\star$. We set $\lambda = 1$. The goal is to decompose the obsevations X into low-rank L and sparse S components. Putting the above problem into the cone form of (1) yields, for $\mathcal{K}$, the direct product of the positive semidefinite cone and nonnegative orthant (details in the supplement). We see that Newton-ADMM and SCS are comparable, in the fourth column of Figure 1.

5.5. Extension as a specialized solver

Finally, we observe that the basic idea of treating the residuals of consecutive ADMM iterates as a fixed point iteration, and then finding a fixed point using a Newton method, is completely general, *i.e.*, the same idea can be used to accelerate (virtually) any ADMM-based algorithm, for a convex problem. To illustrate, consider the lasso problem,

$$\underset{\theta\in\mathbf{R}^p}{\text{minimize}} \quad (1/2)\|y - X\theta\|_2^2 + \lambda\|\theta\|_1, \tag{25}$$

where $y \in \mathbf{R}^N,\ X \in \mathbf{R}^{N\times p},\ \lambda \geq 0$; the ADMM recurrences (Parikh & Boyd, 2014, Section 6.4) are

$$\theta \leftarrow (X^TX + \rho I)^{-1}(X^Ty + \rho(\kappa - \mu)) \tag{26}$$

$$\kappa \leftarrow S_{\lambda/\rho}(\theta + \mu) \tag{27}$$

$$\mu \leftarrow \mu + \theta - \kappa, \tag{28}$$

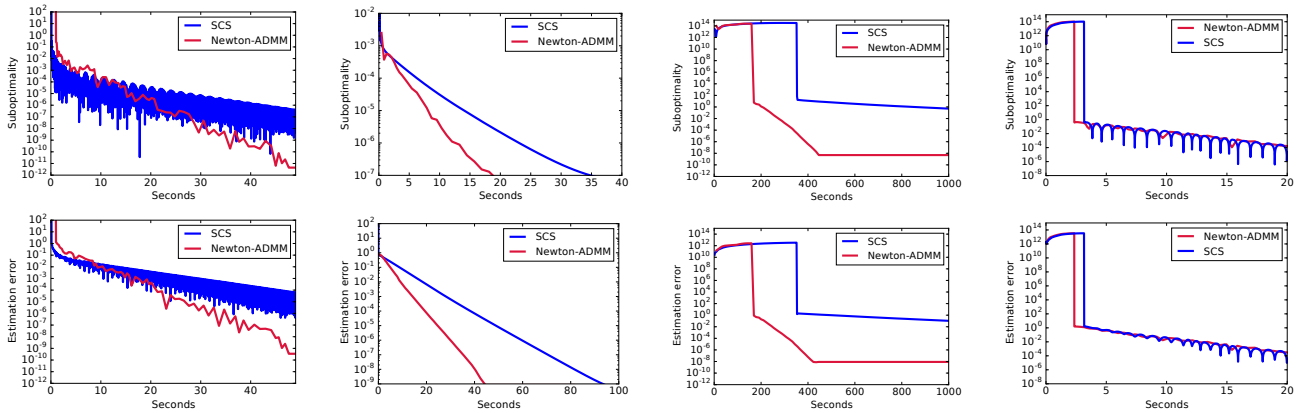

Figure 1: Comparison of Newton-ADMM and SCS (O'Donoghue et al., 2016), on several convex problems. Columns, from left to right: linear programming, portfolio optimization, ℓ_1-penalized logistic regression, robust PCA. Top row: wallclock time vs. log-distance to the optimal objective value, obtained by running an interior point method. Bottom row: wallclock time vs. log-distance, in a Euclidean norm sense, to the solution. Each plot is one representative run out of 20 (the variance was negligible). Best viewed in color.

where $\rho > 0$, $\kappa, \mu \in \mathbf{R}^p$ are the tuning parameter and auxiliary variables, introduced by ADMM, respectively, and $S_{\lambda/\rho}(\cdot)$ is the soft-thresholding operator. The map $F : \mathbf{R}^{3p} \to \mathbf{R}^{3p}$, from (16), with components set to the residuals of the ADMM iterates given in (26) – (28), is then

$$F(z) = \begin{bmatrix} (X^T X + \rho I)\theta - (X^T y + \rho(\kappa - \mu)) \\ \kappa - S_{\lambda/\rho}(\theta + \mu) \\ \kappa - \theta \end{bmatrix},$$

where $z = (\theta, \kappa, \mu) \in \mathbf{R}^{3p}$, and we also changed coordinates, similar to before. An element $J \in \mathbf{R}^{3p \times 3p}$ of the generalized Jacobian of F is then

$$J = \begin{bmatrix} X^T X + \rho I & -\rho I & \rho I \\ -D & I & D \\ -I & I & 0 \end{bmatrix},$$

where $D \in \mathbf{R}^{p \times p}$ is diagonal with D_{ii} set to 1 if $|\theta_i + \mu_i| > \lambda/\rho$ and 0 otherwise, for $i = 1, \ldots, m$.

In the left panel of Figure 2, we compare a specialized Newton-ADMM applied *directly* to the lasso problem (25), with the ADMM algorithm for (26) – (28), a proximal gradient method (Beck & Teboulle, 2009), and a heavily-optimized implementation of coordinate descent (Friedman et al., 2007); we set $p = 400$, $N = 200$, $\lambda = 10$, $\rho = 1$. Here, the specialized Newton-ADMM is quite competitive with these strong baselines; the specialized Newton-ADMM outperforms Newton-ADMM applied to the cone program (2), so we omit the latter from the comparison. Stella et al. (2016) recently described a related approach.

In the right panel of Figure 2, we present a similar comparison, for sparse inverse covariance estimation, with the QUIC method of Hsieh et al. (2014); Newton-ADMM clearly performs best ($p = N = 1,000$, $\lambda = \rho = 1$, details in the supplement).

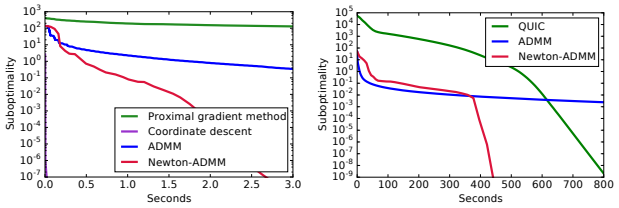

Figure 2: Left: wallclock time vs. log-distance to the optimal objective value, on the lasso problem, for the specialized Newton-ADMM method, standard ADMM, a proximal gradient method, and a heavily-optimized coordinate descent implementation (as a reference benchmark). Right: for a sparse inverse covariance estimation problem, with specialized Newton-ADMM, standard ADMM, and QUIC (Hsieh et al., 2014). Best viewed in color.

6. Discussion

We introduced Newton-ADMM, a new method for generic convex programming. The basic idea is use a nonsmooth Newton method to find a fixed point of the residuals of the consecutive ADMM iterates generated by SCS, a state-of-the-art solver for cone programs; we showed that the basic idea is fairly general, and can be applied to accelerate (virtually) any ADMM-based algorithm. We presented theoretical and empirical support that Newton-ADMM converges rapidly (*i.e.*, quadratically) to a solution, outperforming SCS across several problems.

Acknowledgements. AA was supported by the DoE Computational Science Graduate Fellowship DE-FG02-97ER25308. EW was supported by DARPA, under award number FA8750-17-2-0027. We thank Po-Wei Wang and the referees for a careful proof-reading.

References

Ali, Alnur, Khare, Kshitij, Oh, Sang-Yun, and Rajaratnam, Bala. Generalized pseudolikelihood methods for inverse covariance estimation. Technical report, 2016. Available at http://arxiv.org/pdf/1606.00033.pdf.

Andersen, Martin, Dahl, Joachim, Liu, Zhang, and Vandenberghe, Lieven. Interior point methods for large-scale cone programming. *Optimization for machine learning*, pp. 55–83, 2011.

Beck, Amir and Teboulle, Marc. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Ben-Tal, Aharon and Nemirovski, Arkadi. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

Boyd, Stephen and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

Boyd, Stephen, Parikh, Neal, Chu, Eric, Peleato, Borja, and Eckstein, Jonathan. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Candès, Emmanuel, Li, Xiaodong, Ma, Yi, and Wright, John. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

Clarke, Frank. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

Diamond, Steven and Boyd, Stephen. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Douglas, Jim and Rachford, Henry. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82(2):421–439, 1956.

Evans, Lawrence and Gariepy, Ronald. *Measure Theory and Fine Properties of Functions*. CRC Press, 2015.

Facchinei, Francisco and Kanzow, Christian. A nonsmooth inexact Newton method for the solution of large-scale nonlinear complementarity problems. *Mathematical Programming*, 76(3):493–512, 1997.

Facchinei, Francisco, Fischer, Andreas, and Kanzow, Christian. Inexact Newton methods for semismooth equations with applications to variational inequality problems, 1996.

Ferris, Michael and Munson, Todd. Semismooth support vector machines. *Mathematical Programming*, 101(1): 185–204, 2004.

Friedman, Jerome, Hastie, Trevor, Höfling, Holger, and Tibshirani, Robert. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

Grant, Michael. *Disciplined Convex Programming*. PhD thesis, Stanford University, 2004.

Han, Shih-Ping, Pang, Jong-Shi, and Rangaraj, Narayan. Globally convergent Newton methods for nonsmooth equations. *Mathematics of Operations Research*, 17(3): 586–607, 1992.

Hintermüller, Michael. Semismooth Newton methods and applications. Technical report, 2010. Available at http://www.math.uni-hamburg.de/home/hinze/Psfiles/Hintermueller_OWNotes.pdf.

Hsieh, Cho-Jui, Sustik, Mátyás, Dhillon, Inderjit, and Ravikumar, Pradeep. QUIC: Quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(1):2911–2947, 2014.

Izmailov, Alexey and Solodov, Mikhail. *Newton-Type Methods for Optimization and Variational Problems*. Springer, 2014.

Kanzow, Christian and Fukushima, Masao. Semismooth methods for linear and nonlinear second-order cone programs. Technical report, 2006.

Khan, Kamil A and Barton, Paul. Generalized derivatives for hybrid systems. *IEEE Transactions on Automatic Control*, 2017.

Khare, Kshitij, Oh, Sang-Yun, and Rajaratnam, Bala. A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society: Series B*, 77(4):803–825, 2015.

Kong, Lingchen, Tunçel, Levent, and Xiu, Naihua. Clarke generalized Jacobian of the projection onto symmetric cones. *Set-Valued and Variational Analysis*, 17(2):135–151, 2009.

Lofberg, Johan. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Symposium on Computer Aided Control Systems Design*, pp. 284–289. IEEE, 2005.

Magnus, Jan and Neudecker, Heinz. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, 1995.

Martínez, José and Qi, Liqun. Inexact Newton methods for solving nonsmooth equations. *Journal of Computational and Applied Mathematics*, 60(1):127–145, 1995.

Mifflin, Robert. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

Nesterov, Yurii and Nemirovskii, Arkadii. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

O'Donoghue, Brendan, Chu, Eric, Parikh, Neal, and Boyd, Stephen. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, pp. 1–27, 2016.

Parikh, Neal and Boyd, Stephen. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

Peaceman, Donald and Rachford, Henry. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

Qi, Liqun and Sun, Defeng. A survey of some nonsmooth equations and smoothing Newton methods, 1999.

Qi, Liqun and Sun, Jie. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1-3):353–367, 1993.

Saad, Youcef and Schultz, Martin. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

Serrano, Santiago. *Algorithms for Unsymmetric Cone Optimization and an Implementation for Problems with the Exponential Cone*. PhD thesis, Stanford University, 2015.

Śmietański, Marek. A generalized Jacobian based Newton method for semismooth block triangular system of equations. *Journal of Computational and Applied Mathematics*, 205(1):305–313, 2007.

Stella, Lorenzo, Themelis, Andreas, and Patrinos, Panagiotis. Forward-backward quasi-Newton methods for nonsmooth optimization problems. Technical report, 2016. Available at https://arxiv.org/pdf/1604.08096.pdf.

Sturm, Jos. Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. *Optimization Methods and Software*, 17(6): 1105–1154, 2002.

Sun, Defeng and Sun, Jie. Semismooth matrix-valued functions. *Mathematics of Operations Research*, 27(1):150–169, 2002.

Toh, Kim-Chuan, Todd, Michael, and Tütüncü, Reha. On the implementation and usage of SDPT3 — a MATLAB software package for semidefinite/quadratic/linear programming, version 4.0. In *Handbook on Semidefinite, Conic, and Polynomial Optimization*, pp. 715–754. Springer, 2012.

Udell, Madeleine, Mohan, Karanveer, Zeng, David, Hong, Jenny, Diamond, Steven, and Boyd, Stephen. Convex optimization in Julia. In *Proceedings of the 1st First Workshop for High Performance Technical Computing in Dynamic Languages*, pp. 18–28. IEEE, 2014.

Ulbrich, Michael. *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*. SIAM, 2011.

Wu, Shao-Po and Boyd, Stephen. sdpsol: A parser/solver for semidefinite programs with matrix structure. *Advances in Linear Matrix Inequality Methods in Control*, pp. 79–91, 2000.

Ye, Yinyu, Todd, Michael, and Mizuno, Shinji. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.