An Algorithm for Quadratic ℓ_1-Regularized Optimization with a Flexible Active-Set Strategy

Richard H. Byrd* Jorge Nocedal † Stefan Solntsev†

March 27, 2014

Abstract

We present an active-set method for minimizing an objective that is the sum of a convex quadratic and ℓ_1 regularization term. Unlike two-phase methods that combine a first-order active set identification step and a subspace phase consisting of a cycle of conjugate gradient iterations, the method presented here has the flexibility of computing one of three possible steps at each iteration: a relaxation step (that releases variables from the active set), a subspace minimization step based on the conjugate gradient iteration, and an active-set refinement step. The choice of step depends on the relative magnitudes of the components of the minimum norm subgradient. The paper establishes global rates of convergence, as well as work complexity estimates. Numerical results illustrating the behavior of the methods on four test sets are presented.

1 Introduction

In this paper, we present an active-set method for the solution of the regularized quadratic problem

$$\min_{x\in\mathbb{R}^n} F(x) \stackrel{\text{def}}{=} \tfrac{1}{2}x^T Ax - b^T x + \tau\|x\|_1, \tag{1.1}$$

where A is a symmetric positive semi-definite matrix and $\tau \geq 0$ is a regularization parameter. The motivation for this work stems from the numerous applications in signal processing, machine learning, and statistics that require the solution of problem (1.1); see e.g. [34, 21, 33] and the references therein.

Although non-differentiable, the quadratic-ℓ_1 problem (1.1) has a simple structure that can be exploited effectively in the design of algorithms, and in their analysis. Our focus in this paper is on methods that incorporate second-order information about the objective function $F(x)$ as an integral part of the iteration. A salient feature of our methods is the flexibility of switching between two types of steps: a) first-order steps that improve the active-set prediction; b) subspace steps that explore the current active set through an inner

*Department of Computer Science, University of Colorado, Boulder, CO, USA. This author was supported by National Science Foundation grant DMS-1216554 and Department of Energy grant DE-SC0001774.

†Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. This author was supported by National Science Foundation grant DMS-0810213, and by Department of Energy grant DE-FG02-87ER25047.

conjugate gradient iteration. A variant of our approach includes a third kind of step, the so-called relaxation step, which releases some of the variables at zero. The choice between these steps is controlled by the so-called gradient balance condition that compares the norms of the free (non-zero) components of the minimum norm subgradient of F with those of the components corresponding to the zero variables. This condition is motivated by the work of Dostal and Schoeberl [13] on the solution of bound constrained problems, but in extending the idea to the quadratic-ℓ_1 problem (1.1), we deviate from their approach in a significant way.

We provide global rates of convergence for the main variant of the algorithm, as well as work-complexity estimates that bound the total amount of computation needed to achieve an ϵ-accurate solution. We also investigate ways of refining the main components of the algorithm. Specifically, we introduce a line search in the identification phase, a modification of the subspace phase that allows the conjugate gradient (CG) iteration to cross orthants as long as a reduction in the objective F is obtained after every CG step, and a relaxation step that releases active variables that are predicted to be non-zero. Our numerical tests show that these algorithmic refinements yield substantial improvements in performance and give rise to effective methods.

The quadratic-ℓ_1 problem (1.1) has received considerable attention in the literature, and a variety of first and second order methods have been proposed for solving it. Most prominent are variants of the iterative soft-thresholding algorithm, ISTA [9, 12, 36], and its accelerated version [26, 3, 4], which have extensive theory and are often used in practice. The TFOCS package [4] provides five first-order methods based on proximal gradient iterations that enjoy optimal complexity bounds; i.e., they achieve ϵ accuracy in at most $O(1/\sqrt{\epsilon})$ iterations. One of these methods (N83) is tested in our numerical experiments. Other first-order methods for problem (1.1) include LARS [14], coordinate descent [16], a fixed point continuation method [19], and a gradient projection method [15].

Schmidt [31] proposes several scaled sub-gradient methods, which can be viewed as extensions, to quadratic ℓ_1 problem (1.1), of a projected quasi-Newton method [2], an active-set method [30], and a two-metric projection method [18] for bound constrained optimization. He compares these methods with some first-order methods such as GSPR [15] and SPARSA [36]. We include his best-performing method (PSSgb) in our numerical tests.

Other second order methods have been proposed as well; they compute a step by minimizing a local quadratic model of F. Some of these algorithms transform problem (1.1) into a smooth bound constrained quadratic programming problem and apply an interior point procedure [24] or a second order gradient projection algorithm [32, 15]. Methods that are closer in spirit to our approach include FPC_AS [35], orthant-based Newton-CG methods [2, 7, 29], and the semi-smooth Newton method in [25]. Our method differs from all these approaches in the adaptive step-by-step nature of the algorithm, where a different kind of step can be invoked at every iteration, depending on information based on the gradient balance condition. This gives the algorithm the flexibility to adapt itself to the characteristics of the problem to be solved, as we discuss in our numerical tests.

The paper is organized in six sections. In section 2 we motivate our approach and describe its most basic implementation. Section 3 provides a convergence analysis and a

work complexity estimate of the algorithm presented in section 2. Practical algorithmic variants are presented in section 4. Section 5 describes implementation details and presents numerical results. The contributions of the paper are summarized in section 6.

2 The Algorithm

Before describing the algorithm, we introduce some notation. Let us define

$$f(x) \stackrel{\text{def}}{=} \tfrac{1}{2}x^T Ax - b^T x,$$

and denote its gradient by

$$g(x) \stackrel{\text{def}}{=} \nabla f(x) = Ax - b.$$

Therefore,

$$F(x) = f(x) + \tau\|x\|_1.$$

The components of the minimum norm subgradient of F at x, which we denote by $v(x)$, are given by

$$v_i(x) = \left\{ \begin{array}{lll} g_i(x) + \tau\,\mathrm{sgn}(x_i) & \text{if} & x_i \neq 0 \\ 0 & \text{if} & x_i = 0 \text{ and } |g_i(x)| \leq \tau \\ g_i(x) - \tau\,\mathrm{sgn}(g_i(x)) & \text{if} & x_i = 0 \text{ and } |g_i(x)| > \tau \end{array} \right\}, \quad i = 1, \ldots, n. \tag{2.1}$$

We write $v(x)$ as the sum

$$v(x) = \omega(x) + \phi(x), \tag{2.2}$$

where $\omega(x)$ contains the components of $v(x)$ corresponding to the zero variables, and $\phi(x)$ the components corresponding to the non-zero variables. More precisely, for each $i \in \{1 \cdots n\}$,

$$\omega_i(x) = \left\{ \begin{array}{lll} 0 & \text{if} & x_i \neq 0 \\ 0 & \text{if} & x_i = 0 \text{ and } |g_i(x)| \leq \tau \\ g_i(x) - \tau\,\mathrm{sgn}(g_i(x)) & \text{if} & x_i = 0 \text{ and } |g_i(x)| > \tau \end{array} \right\}, \tag{2.3}$$

and

$$\phi_i(x) = \left\{ \begin{array}{lll} g_i(x) + \tau\,\mathrm{sgn}(x_i) & \text{if} & x_i \neq 0 \\ 0 & \text{if} & x_i = 0 \text{ and } |g_i(x)| \leq \tau \\ 0 & \text{if} & x_i = 0 \text{ and } |g_i(x)| > \tau \end{array} \right\}. \tag{2.4}$$

At a point x, a move along the direction $-\omega(x)$ releases some of the zero variables, i.e. relaxes the active set. Following Dostal and Schoeberl [13], we use the magnitudes of the vectors ω and ϕ to determine which type of step should be taken. The algorithm thus monitors the *gradient balance condition*

$$\|\omega(x)\|_2 \leq \|\phi(x)\|_2, \tag{2.5}$$

which governs the flow of the iteration and distinguishes it from both two-phase methods [2, 7, 36] and semi-smooth Newton methods [25] for problem (1.1).

The algorithm starts by computing a first order active-set identification step. This is done using an iterative soft-thresholding (ISTA) step [12, 9], i.e.,

$$\begin{aligned} x^{k+1} &= \arg\min_y (y - x^k)^T g(x^k) + \frac{1}{2\bar{\alpha}} \|(y - x^k)\|_2^2 + \tau \|y\|_1 \\ &= \max\{|x^k - \bar{\alpha} g(x^k)| - \bar{\alpha}\tau, 0\} \operatorname{sgn}(x^k - \bar{\alpha} g(x^k)), \end{aligned} \tag{2.6}$$

where $\bar{\alpha}$ is a given stepsize whose choice is discussed below.

Then a subspace minimization procedure is performed. It uses the conjugate gradient (CG) method to reduce a model of the objective $F(x)$ on the subspace

$$H = \{x |\, x_i = 0, \quad \text{for all } i \text{ such that } x_i^{\text{cg}} = 0\}, \tag{2.7}$$

where x^{cg} denotes the point at which the CG procedure was started (this point is provided by the ISTA step).

To describe the CG iteration, we define a smooth function q to equal the objective F on the current orthant defined by x^{cg}, i.e.,

$$q(x; x^{\text{cg}}) \stackrel{\text{def}}{=} \tfrac{1}{2} x^T A x + (-b + \tau \operatorname{sgn}(x^{\text{cg}}))^T x, \tag{2.8}$$

where we use the convention $\operatorname{sgn}(0) = 0$ and the fact that

$$\|x\|_1 = \operatorname{sgn}(x)^T x.$$

Clearly, $F(x) = q(x; x^{\text{cg}})$ for all x such that $\operatorname{sgn}(x) = \operatorname{sgn}(x^{\text{cg}})$. The algorithm applies the projected CG iteration [28, chap 16] to the problem

$$\min_x \quad q(x; x^{\text{cg}}) \tag{2.9}$$

$$\text{s.t.} \quad x_i = 0, \quad \text{for all } i \text{ such that } x_i^{\text{cg}} = 0. \tag{2.10}$$

Let $P(\cdot)$ denote a projection onto the set H given in (2.7). An iteration of the projected CG method is given by

$$\begin{aligned} x^{k+1} &= x^k + \alpha_{\text{cg}} d^k, \quad \text{with} \quad \alpha_{\text{cg}} = \frac{(r^k)^T \rho^k}{(d^k)^T A d^k}; \\ r^{k+1} &= r^k + \alpha_{\text{cg}} A d^k; \\ \rho^{k+1} &= P(r^{k+1}); \\ d^{k+1} &= -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k. \end{aligned}$$

The gradient balance condition (2.5) is tested after every CG iteration, and if it is not satisfied, the CG loop is terminated. This is a sign that subsantial improvements in the objective value can be achieved by releasing some of the variables at zero. This loop is also terminated if a CG iterate crosses orthants, as in this case we cannot guarantee that the CG method continues to minimize F. Specifically, when $\operatorname{sgn}(x^{k+1}) \neq \operatorname{sgn}(x^{\text{cg}})$, we cut back the step and redefine the last iterate of the CG procedure as

$$x^{k+1} = x^k + \alpha_b d^k, \qquad \alpha_b = \arg\max_{\alpha > 0} \{\alpha : \operatorname{sgn}(x^k + \alpha d^k) = \operatorname{sgn}(x^{\text{cg}})\},$$

A variant of the algorithm that allows crossing orthants is discussed in section 4.

A precise description of the method for solving problem (1.1) is given in Algorithm 1. Here and henceforth $\|\cdot\|$ stands for the ℓ_2 norm.

Algorithm 1

Require: A, b, τ, x^0, and $\bar{\alpha}$

1: $k = 0$
2: **loop**
3: $x^{k+1} = \max\{|x^k - \bar{\alpha} g(x^k)| - \bar{\alpha}\tau, 0\} \operatorname{sgn}(x^k - \bar{\alpha} g(x^k))$ *ISTA step*
4: $k = k + 1$
5: $r^k = g(x^k) + \tau \operatorname{sgn}(x^k)$, $\rho^k = P(r^k)$,$d^k = -\rho^k$ $x^{\mathrm{cg}} = x^k$
6: **loop**
7: **if** $\|\omega(x^k)\| > \|\phi(x^k)\|$ **then**
8: **break**
9: **end if**
10: $x^{k+1} = x^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} d^k$ *CG step*
11: $r^{k+1} = r^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} A d^k$
12: **if** $\operatorname{sgn}(x^{k+1}) \neq \operatorname{sgn}(x^{\mathrm{cg}})$ **then**
13: $\alpha_b = \arg\max_\alpha \{\alpha : \operatorname{sgn}(x^k + \alpha d^k) = \operatorname{sgn}(x^{\mathrm{cg}})\}$
14: $x^{k+1} = x^k + \alpha_b d^k$ *Cut-back*
15: $k = k + 1$
16: **break**
17: **end if**
18: $\rho^{k+1} = P(r^{k+1})$
19: $d^{k+1} = -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k$
20: $k = k + 1$
21: **end loop**
22: **end loop**

Alternative first order steps (that replace line 3) and CG stopping conditions (that replace lines 12-17) are described in Section 4. A common choice for the stepsize $\bar{\alpha}$ is $\bar{\alpha} = 1/L$ (where L is the largest eigenvalue of A) and is motivated by the convergence analysis in Section 3.

Note that in Algorithm 1 the index k may be incremented multiple times during every outer iteration loop. While it is possible to express the same algorithmic logic in a more conventional way, with a single k increment in each outer iteration, our description emphasizes that every inner CG step and ISTA step require approximately the same amount of computational effort, which is dominated by a matrix-vector product.

3 Convergence Analysis

We establish global convergence for Algorithm 1 by showing that its constitutive steps, ISTA and CG, provide sufficient decrease in the objective function. We also show a global 2-step Q-linear rate of convergence, and based on the fact that the number of cut CG steps cannot exceed a half of the total number of steps, we also establish a complexity result.

In this section, we assume that A is nonsingular, and denote its largest and smallest eigenvalues by L, λ, respectively. Thus, for any $x \in \mathbb{R}^n$,

$$\lambda \|x\|^2 \leq x^T A x \leq L\|x\|^2, \tag{3.1}$$

where $\|\cdot\|$ stands for the ℓ_2 norm. We denote the minimizer of F by x^*.

We start by demonstrating a Q-linear decrease in the objective F for every ISTA step.

Lemma 3.1. *The ISTA step,*

$$x^{k+1} = \arg\min_y f(x^k) + (y - x^k)^T g(x^k) + \frac{1}{2\bar{\alpha}}\|(y - x^k)\|^2 + \tau\|y\|_1, \tag{3.2}$$

with $0 < \bar{\alpha} \leq 1/L$, satisfies

$$F(x^{k+1}) - F(x^*) \leq (1 - \lambda\bar{\alpha})(F(x^k) - F(x^*)).$$

Proof. By forming an upper quadratic approximation of f centered at x^k and evaluated at x^{k+1}, we have

$$F(x^{k+1}) \leq f(x^k) + g(x^k)^T(x^{k+1} - x^k) + \frac{1}{2\bar{\alpha}}\|x^{k+1} - x^k\|^2 + \tau\|x^{k+1}\|_1.$$

By definition of x^{k+1}, the objective function in (3.2) cannot take on a lower value for any $y \in \mathbb{R}^n$. Therefore, defining $y = x^k + \lambda\bar{\alpha}d$, for some $d \in \mathbb{R}^n$, we have

$$\begin{aligned} F(x^{k+1}) &\leq f(x^k) + g(x^k)^T(\lambda\bar{\alpha}d) + \frac{1}{2\bar{\alpha}}\|\lambda\bar{\alpha}d\|^2 + \tau\|x^k + \lambda\bar{\alpha}d\|_1 \\ &= F(x^k + \lambda\bar{\alpha}d) - \tfrac{1}{2}(\lambda\bar{\alpha}d)^T A(\lambda\bar{\alpha}d) + \tfrac{1}{2\bar{\alpha}}\|\lambda\bar{\alpha}d\|^2 \\ &\leq F(x^k + \lambda\bar{\alpha}d) + \tfrac{1}{2}\lambda^2\bar{\alpha}(1 - \lambda\bar{\alpha})\|d\|^2. \end{aligned} \tag{3.3}$$

Since F is a strongly convex function with parameter λ, it satisfies

$$F(tx + (1-t)y) \leq tF(x) + (1-t)F(y) - \tfrac{1}{2}\lambda t(1-t)\|x - y\|^2, \tag{3.4}$$

for any $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$. Setting $x \leftarrow x^k$, $y \leftarrow x^*$, and $t \leftarrow (1 - \lambda\bar{\alpha})$ (which is valid because $\lambda\bar{\alpha} \in (0, 1]$), inequality (3.4) yields

$$F(x^k + \lambda\bar{\alpha}(x^* - x^k)) \leq \lambda\bar{\alpha}F(x^*) + (1 - \lambda\bar{\alpha})F(x^k) - \tfrac{1}{2}\lambda^2\bar{\alpha}(1 - \lambda\bar{\alpha})\|x^* - x^k\|^2.$$

By setting $d = x^* - x^k$, and substituting this inequality in (3.3), we conclude that

$$F(x^{k+1}) - F(x^*) \leq (1 - \lambda\bar{\alpha})(F(x^k) - F(x^*)).$$

□

The next result shows that a conjugate gradient step also guarantees sufficient decrease in the objective function, provided it is not cut-back, i.e., that the vector x^{k+1} computed in step 10 of Algorithm 1 satisfies $\operatorname{sgn}(x^{k+1}) = \operatorname{sgn}(x^{\text{cg}})$, where x^{cg} is the starting point of the CG cycle.

Lemma 3.2. *If Algorithm 1 takes a full conjugate gradient step from x^k to x^{k+1}, then*

$$F(x^{k+1}) \leq F(x^k) - \frac{1}{8L}\|v(x^k)\|^2. \tag{3.5}$$

Proof. Since a projected CG step is taken from an iterate x^k only if $\operatorname{sgn}(x^k) = \operatorname{sgn}(x^{cg})$, and since we assume x^{k+1} is given by a full CG step, it follows that x^{k+1} is the result of a sequence of projected CG steps on problem (2.9), starting at x^{cg}. It is well known that a CG iterate x^{k+1} is a global minimizer of $q(\cdot\,; x^{\text{cg}})$ in the subspace $S = \operatorname{span}\{d^k, d^{k-1}, \ldots\}$, i.e.,

$$q(x^{k+1}) = \min\{q(x^k + y) : y \in S\}.$$

It is also known that $P(\nabla q(x^k; x^{\text{cg}})) \in S$, see [28, Theorem 5.3], and it follows from (2.4) that $P(\nabla q(x^k; x^{\text{cg}})) = \phi(x^k)$. Thus,

$$F(x^{k+1}) = q(x^{k+1}) \leq q(x^k - \alpha\phi(x^k)),$$

for any α. Let us choose

$$\alpha = \frac{\phi(x^k)^T\phi(x^k)}{\phi(x^k)^T A\phi(x^k)}.$$

Then, recalling (2.8) and defining $z = \operatorname{sgn}(x^{\text{cg}})$, we have

$$\begin{aligned} F(x^{k+1}) &\leq \tfrac{1}{2}(x^k - \alpha\phi(x^k))^T A(x^k - \alpha\phi(x^k)) + (-b + \tau z)^T(x^k - \alpha\phi(x^k)) \\ &= F(x^k) + \frac{1}{2}(\alpha\phi(x^k))^T A(\alpha\phi(x^k)) - (\alpha\phi(x^k))^T Ax^k + (-b + \tau z)^T(-\alpha\phi(x^k)) \\ &= F(x^k) + \frac{\alpha}{2}\|\phi(x^k\|^2 - \alpha(\phi(x^k))^T(Ax^k - b + \tau z). \end{aligned}$$

By (2.4), for i such that $x_i^k = 0$ we have $\phi_i(x^k) = 0$, and for $x_i^k \neq 0$ we have that $\phi_i(x^k) = (Ax^k - b + \tau z)_i$. Therefore,

$$\begin{aligned} F(x^{k+1}) &= F(x^k) - \frac{\alpha}{2}\|\phi(x^k)\|^2 \\ &\leq F(x^k) - \frac{1}{2L}\|\phi(x^k)\|^2. \end{aligned}$$

Since the step is only taken when condition (2.5) is true, we have that

$$\|v(x^k)\| \leq 2\|\phi(x^k)\|.$$

Therefore, we have that the following bound holds after one CG iteration,

$$F(x^{k+1}) \leq F(x^k) - \frac{1}{8L}\|v(x^k)\|^2$$

□

We can now establish a 2-step Q-linear convergence result by combining the properties of the two steps in Algorithm 1.

Theorem 3.3. *Suppose that the stepsize $\bar{\alpha}$ in the ISTA step (2.6) satisfies $\frac{1}{16L} \leq \bar{\alpha} \leq \frac{1}{L}$. Then, for the entire sequence $\{x^k\}$ generated by Algorithm 1 we have*

$$F(x^{k+2}) - F(x^*) \leq \left(1 - \tfrac{\lambda}{16L}\right)(F(x^k) - F(x^*)) \tag{3.6}$$

and thus $\{x^k\} \to x^*$.

Proof. By Lemma 3.1 and the lower bound on $\bar{\alpha}$, we have that the ISTA step satisfies

$$F(x^{k+1}) - F(x^*) \leq (1 - \tfrac{\lambda}{16L})(F(x^k) - F(x^*)).$$

By Lemma 3.2, we have that full CG steps provide the decrease

$$F(x^{k+1}) \leq F(x^k) - \tfrac{1}{8L}\|v(x^k)\|^2. \tag{3.7}$$

By convexity of F we have that

$$F(x^k) - F(x^*) \leq -v(x^k)^T(x^* - x^k) \leq \|v(x^k)\|\|x^* - x^k\|,$$

which combined with (3.7) gives

$$F(x^k) - F(x^{k+1}) \geq \frac{1}{8L}\frac{(F(x^k) - F(x^*))^2}{\|x^* - x^k\|^2}.$$

Furthermore, since F is strongly convex, it satisfies

$$F(x^k) - F(x^*) \geq \tfrac{\lambda}{2}\|x^k - x^*\|^2,$$

see [27, pp. 63-64]. Using this bound we conclude that

$$F(x^{k+1}) - F(x^*) \leq (1 - \tfrac{\lambda}{16L})(F(x^k) - F(x^*)). \tag{3.8}$$

Let us assume now that all CG steps are cut-back; i.e., that the worst case happens. After every such shortened CG step, the algorithm falls back to an ISTA step. Therefore, the Q-linear decrease (3.8) is guaranteed for every 2 matrix-vector products, yielding (3.6).

Since Algorithm 1 is a descent method, this implies the entire sequence satisfies $F(x^k) \to F(x^*)$ monotonically. Moreover, since F is strictly convex it follows that $x^k \to x^*$. □

The most costly computations in Algorithm 1 are matrix-vector products; the rest of the computations consist of vector operations. Therefore, when establishing bounds on the total amount of computation required to obtain an ϵ-accurate solution, it is appropriate to measure work in terms of matrix-vector products. Since there is a single matrix-vector product in each of the two constitutive steps of Algorithm 1, a work complexity result can be derived from Theorem 3.3.

Corollary 3.4. *The number of matrix-vector products required by Algorithm 1 to compute an iterate $\hat{x}$ such that*

$$F(\hat{x}) - F(x^*) \leq \epsilon \tag{3.9}$$

is at most

$$\log\left[\frac{\epsilon}{F(x^0) - F(x^*)}\right] \Big/ \log\sqrt{1 - \tfrac{\lambda}{16L}}. \tag{3.10}$$

Proof. By (3.6), the condition (3.9), with $\hat{x} = x^{k+2}$, will be satisfied by an integer k such that

$$\left(1 - \tfrac{\lambda}{16L}\right)^{\frac{k}{2}} (F(x^0) - F(x^*)) \leq \epsilon.$$

We obtain (3.10) by solving for k. □

This result represents worst-case analysis, and is not indicative of the typical performance of the algorithm in practice. In particular, the analysis of the conjugate gradient step relies on the fact that it is no worse than a standard gradient step – a statement that hides the power of the subspace procedure, which is evident in the result that follows. We establish, under strict complementarity, that the algorithm identifies the optimal active manifold and the optimal orthant in a finite number of iterations.

Since $v(x^*) = 0$, it follows from (2.1) that for all i such that $x_i^* = 0$ we must have $|g_i(x^*)| \leq \tau$. We say that the solution x^* satisfies strict complementarity if $x_i^* = 0$ implies that $|g_i(x^*)| < \tau$.

Theorem 3.5. *If the solution x^* of problem* (1.1) *satisfies strict complementarity, then for all sufficiently large k, the iterates x^k will lie in the same orthant and active manifold as x^*. This implies that the algorithm identifies the optimal solution x^* in a finite number of iterations.*

Proof. We start by defining the sets

$$Z^* = \{i : x_i^* = 0\}, \quad N^* = \{i : x_i^* < 0\}, \quad P^* = \{i : x_i^* > 0\},$$

and the constants

$$\delta_1 = \min_{i \in N^* \cup P^*} \frac{|x_i^*|}{2}, \quad \delta_2 = \min_{i \in Z^*} \left[\frac{\tau - |g_i(x^*)|}{2}\right].$$

Clearly $\delta_1 > 0$, and by the strict complementarity assumption we have that $\delta_2 > 0$.

Since, from Theorem 3.3 we have that $\{x^k\} \to x^*$, there exists an integer k_0 such that for any $k \geq k_0$ we have

$$x_i^k < -\delta_1 \ \ \forall i \in N^*, \quad x_i^k > \delta_1 \ \ \forall i \in P^*"$$

$$|x_i^k| < \frac{\bar{\alpha}\delta_2}{2} \ \ \forall i \in Z^* \tag{3.11}$$

$$|g_i(x^k)| < \tau - \delta_2 \ \ \forall i \in Z^*. \tag{3.12}$$

Thus, all variables that are positive at the solution will be positive for $k > k_0$; and similarly for all negative variables. For the rest of the variables, we consider the ISTA step,

$$x^{k+1} = \max\{|x^k - \bar{\alpha}g(x^k)| - \bar{\alpha}\tau, 0\} \operatorname{sgn}(x^k - \bar{\alpha}g(x^k)).$$

Using (3.11) and (3.12), we have that for any $k \geq k_0$ and $i \in Z^*$,

$$\begin{aligned} |x_i^k - \bar{\alpha} g_i(x^k)| - \bar{\alpha}\tau &\leq |x_i^k| + |\bar{\alpha} g_i(x^k)| - \bar{\alpha}\tau \\ &\leq \frac{\bar{\alpha}\delta_2}{2} + \bar{\alpha}(\tau - \delta_2) - \bar{\alpha}\tau \\ &= -\frac{\bar{\alpha}\delta_2}{2} < 0. \end{aligned}$$

Therefore, for all $i \in Z^*$ and all $k \geq k_0$, the ISTA step sets $x^{k+1} = 0$.

An ISTA step must be taken within n iterations of k_0, because of the finite termination property of the conjugate gradient algorithm. Therefore there exists a k_1 such that for any $k \geq k_1$, $\mathrm{sgn}(x^k) = \mathrm{sgn}(x^*)$, and by (3.12), $\omega(x^k) = 0$. These two facts imply that for $k \geq k_1$, once the algorithm enters the CG iteration it will not leave, since the two `break` conditions cannot be satisfied. Finite termination of CG implies the optimal solution x^* will be found in a finite number of iterations. □

4 Algorithmic Variants

We now describe variants of the two phases that form Algorithm 1, namely a modified first-order step, and a more tolerant CG phase that allows changes in orthants. We also present an extension of Algorithm 1, that in addition to the modifications just mentioned, includes an extra step (the so-called relaxation step). These algorithmic variants provide significant gains in performance compared with Algorithm 1, as shown in section 5. We now discuss each of these techniques in detail.

4.1 Relaxed CG Strategy and a Modified ISTA Step

It can be beneficial to allow the CG iteration to leave the current orthant, as long as the objective F is reduced sufficiently after every CG step. Inspired by (3.5), we require that

$$F(x^{k+1}) \leq F(x^k) - c\|v(x^k)\|^2, \tag{4.1}$$

for some $c \geq 0$.

The relaxed CG strategy is as follows. If x^{k+1} is the first CG iterate that leaves the orthant, then we either accept it, if it produces the sufficient decrease (4.1) in F, or we perform the cut-back (as before) to the boundary of the current orthant. On the other hand, if both x^{k+1} and x^k lie outside the current orthant and if sufficient decrease is not obtained at x^{k+1}, then the algorithm reverts to x^k. This new termination test for the CG iteration, which replaces lines 12-17 in Algorithm 1, is thus given as follows:

Relaxed CG Stop Test

1: **if** $F(x^{k+1}) > F(x^k) - c\|v(x^k)\|^2$ **then**
2: **if** $\mathrm{sgn}(x^k) = \mathrm{sgn}(x^{\mathrm{cg}})$ **then**
3: $\alpha_b = \arg\max_\alpha \{\alpha : \mathrm{sgn}(x^k + \alpha d^k) = \mathrm{sgn}(x^{\mathrm{cg}})\}$
4: $x^{k+1} = x^k + \alpha_b d^k$ *Cut-back*
5: **else**

6: $x^{k+1} = x^k$
7: **end if**
8: $k = k + 1$
9: **break**
10: **end if**

The condition $\text{sgn}(x^{k+1}) \neq \text{sgn}(x^{\text{cg}})$ present in step 12 of Algorithm 1 is now redundant, since by Lemma 3.2, condition (4.1) would not be triggered when $\text{sgn}(x^{k+1}) = \text{sgn}(x^{\text{cg}})$. One of the main benefits of allowing the CG iteration to move more freely across an orthant boundary is to prevent the generation of unnecessarily short subspace steps. In addition, if the quadratic model (2.8) does not change much as orthants change (for example, when τ is small), CG steps can be beneficial even if they are based on information from another orthant. The convergence analysis of Theorem 3.3 still holds for this strategy, since the CG phase still satisfies decrease of the form of Lemma 3.2.

Additional improvements in performance can be obtained by implementing a more sophisticated ISTA step. As suggested by Wright et al. [36], the Barzilai-Borwein stepsize with a non-monotone linesearch is usually preferable to the constant stepsize scheme in Algorithm 1. Line 3 in Algorithm 1 is replaced by the following procedure.

ISTA-BB-LS Step
At the beginning of the overall algorithm, we initialize $M = 5$, $\xi = 0.005$, and $F^i = F(x^0)$ for $i \in \{1 \dots M\}$. Let x^k be current iterate and let x^{k-1} denote the previous iterate.

1: $\alpha_B = \frac{(x^k - x^{k-1})^T (x^k - x^{k-1})}{(x^k - x^{k-1})^T A (x^k - x^{k-1})}$
2: **repeat**
3: $x_F = \max\{|x^k - \alpha_B g(x^k)| - \alpha_B \tau, 0\} \, \text{sgn}(x^k - \bar{\alpha} g(x^k))$
4: $\alpha_B = \frac{\alpha_B}{2}$
5: **until** $F(x_F) \leq \max_{i \in \{1 \dots M\}} F^i - \alpha_B \xi \|x - x_F\|^2$
6: $F^{i+1} = F^i$ for all $i \in \{1 \dots M - 1\}$
7: $F^1 = F(x_F)$
8: $x^{k+1} = x_F$

The constants are identical to the ones used in [36], and we do not attempt to fine-tune them. Since $A(x^k - x^{k-1}) = g(x^k) - g(x^{k-1})$, and these two gradient values can be stored prior to the computation of the ISTA step, no matrix-vector products are needed in step 1.

The two variants just described, when implemented together, perform well in our tests. We call the resulting method Algorithm 2.

4.2 Refinement-Relaxation (RR) framework

Dostal and Schoeberl [13] proposed a method for the minimization of a convex quadratic function subject to bounds on the variables that is composed of three types of steps: a projection step, a relaxation step and a subspace step. An extension of this framework to the regularized quadratic problem (1.1) suggests the method described below.

In contrast to the implementations of the ISTA step described above, which operate in the full space of variables, the algorithm computes the ISTA step in a reduced space.

Specifically, we set

$$x^{k+1} = \arg\min_y f(x^k) + (y - x^k)^T g^s(x^k) + \frac{1}{2\bar{\alpha}^s}\|(y - x^k)\|^2 + \tau\|y\|_1, \tag{4.2}$$

where $g^s(x)$ is defined by

$$g_i^s(x) \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} A_i(x) - b_i & \text{if} & x_i \neq 0 \\ 0 & \text{if} & x_i = 0 \end{array} \right\}, \qquad i = 1, \ldots, n.$$

The stepsize, $\bar{\alpha}^s$ is computed by the Barzilai-Borwein procedure described in the previous section. The goal of this *reduced subspace ISTA step* is to refine the estimate of the active set, without releasing any variables. That task is assigned to the *relaxation step*, which is given by

$$x^{k+1} = x^k - \alpha_r \omega(x^k), \qquad \alpha_r = \frac{\omega(x^k)^T \omega(x^k)}{\omega(x^k)^T A \omega(x^k)}. \tag{4.3}$$

A move along the direction $-\omega(x^k)$ cannot result in a change of orthants, and hence it is safe to define the stepsize α_r as the minimizer of the objective function $F(x)$ along the direction $-\omega(x^k)$. The third type of step is a subspace CG step, which is performed as in Algorithm 1 and Algorithm 2.

Due to the form of the reduced ISTA step, it is necessary to modify the gradient balance condition (2.5), as follows,

$$\|\omega(x^k)\|^2 \leq -\phi(x^k)^T \tilde{\phi}(x^k; \hat{\alpha}), \tag{4.4}$$

where $\tilde{\phi}(x; \alpha)$ is a truncated version of $\phi(x^k)$. $\tilde{\phi}(x; \alpha)$ depends on an ISTA step. Precisely,

$$\begin{aligned} \tilde{\phi}(x; \alpha) &\stackrel{\text{def}}{=} \frac{\arg\min_y (y - x)^T g(x) + \frac{1}{2\alpha}\|(y - x)\|^2 + \tau\|y\|_1 - x}{\alpha} \\ &= \frac{\max\{|x - \alpha g(x)| - \alpha\tau, 0\} \operatorname{sgn}(x - \alpha g(x)) - x}{\alpha} \end{aligned}$$

The reason for this modification is similar to that given in [13, p.28]: if the condition is satisfied, the reduced ISTA step makes sufficient progress in the algorithm. Small $\hat{\alpha}$ makes the condition identical to (2.5). A common choice for $\hat{\alpha}$ is $\frac{1}{L}$, but the algorithm is rather insensitive to the choice of $\hat{\alpha}$ as indicated in Appendix B. There is much freedom in choosing this parameter as its used only in testing the gradient balance condition (in Steps 3, 14 and 21 of Algorithm 3), and not in the computation of the steps of the algorithm.

The algorithm proceeds as follows. If the modified gradient balance condition (4.4) is satisfied, the reduced ISTA step (4.2) is taken. Then, the gradient balance condition (4.4) is checked again, and if it is violated, it is an indication that releasing some of the active variables may be beneficial, and the relaxation step (4.3) is computed. Finally, the CG loop is commenced. A detailed description of this algorithm is given as Algorithm 3.

Algorithm 3

Require: A, b,τ, x^0, $\hat{\alpha}$,c

$k = 0$, $M = 5$, $\xi = 0.005$ $F^i = F(x^0)$ for all $i \in \{1 \dots M\}$
loop
  if $\|\omega(x^k)\|^2 \leq -\phi(x^k)^T \tilde{\phi}(x^k; \hat{\alpha})$ **then**
    $\alpha_B = \frac{(x^k - x^{k-1})^T (x^k - x^{k-1})}{(x^k - x^{k-1})^T A (x^k - x^{k-1})}$
    repeat
      $x_F = \max\{|x^k - \alpha_B g^s(x^k)| - \alpha_B \tau, 0\} \operatorname{sgn}(x^k - \bar{\alpha} g^s(x^k))$ *Reduced ISTA step*
      $\alpha_B = \frac{\alpha_B}{2}$
    until $F(x_F) \leq \max_{i \in \{1,\dots,M\}} F^i - \alpha_B \xi \|x - x_F\|^2$
    $F^{i+1} = F^i$ for all $i \in \{1, \dots, M-1\}$
    $F^1 = F(x_F)$
    $x^{k+1} = x_F$
    $k = k + 1$
  end if
  if $\|\omega(x^k)\|^2 > -\phi(x^k)^T \tilde{\phi}(x^k; \hat{\alpha})$ **then**
    $\alpha_r = \frac{\omega(x^k)^T \omega(x^k)}{\omega(x^k)^T A \omega(x^k)}$
    $x^{k+1} = x^k - \alpha_r \omega(x^k)$ *Relaxation step*
    $k = k + 1$
  end if
  $r^k = g(x^k) + \tau \operatorname{sgn}(x^k)$, $\rho^k = \phi(x^k)$, $d^k = -\rho^k$, $x^{\mathrm{cg}} = x^k$
  loop
    if $\|\omega(x^k)\|^2 > -\phi(x^k)^T \tilde{\phi}(x^k; \hat{\alpha})$ **then**
      break
    end if
    $x^{k+1} = x^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} d^k$ *CG step*
    $r^{k+1} = r^k + \frac{(r^k)^T \rho^k}{(d^k)^T A d^k} A d^k$
    if $F(x^{k+1}) > F(x^k) - c\|v(x^k)\|^2$ **then**
      if $\operatorname{sgn}(x^k) = \operatorname{sgn}(x^{\mathrm{cg}})$ **then**
        $\alpha_b = \arg\max_\alpha \{\alpha : \operatorname{sgn}(x^k + \alpha d^k) = \operatorname{sgn}(x^{\mathrm{cg}})\}$
        $x^{k+1} = x^k + \alpha_b d^k$ *Cut-back*
      else
        $x^{k+1} = x^k$
      end if
      $k = k + 1$
      break
    end if
    $\rho^{k+1} = P(r^{k+1})$
    $d^{k+1} = -\rho^{k+1} + \frac{(r^{k+1})^T \rho^{k+1}}{(r^k)^T \rho^k} d^k$
    $k = k + 1$
  end loop
end loop

5 Numerical Results

We developed a MATLAB implementation of Algorithms 1, 2, and 3, and in the first part of this section we compare their performance relative to two proximal gradient methods. This allows us to study the algorithmic components of our methods in a controlled setting, and to identify the most promising approach, which is then tested against three state-of-the-art codes. Our numerical experiments are performed on four groups of test problems with varying characteristics

5.1 Evaluating the Three Versions of the Algorithm

We implemented the following methods.

Alg1 This is Algorithm 1, which is the most basic implementation of the approach proposed in this paper. The stepsize for the ISTA iteration (Step 3) is set to $\bar{\alpha} = 1/L$.

Alg2 This is the modification of Algorithm 1 that incorporates the ISTA-BB-LS Step and the Relaxed CG Stop Test described in section 4.1 (with $c = 0$ in (4.1)).

Alg3 This is Algorithm 3, which was described in detail in the previous section. We set $\hat{\alpha} = 1/L$ and $c = 0$.

FISTA The Fast Iterative Shrinkage-Thresholding Algorithm [3], using a constant stepsize given by $1/L$.

ISTA-BB-LS This method is composed purely of the ISTA-BB-LS steps described in section 4.1, which are repeated until convergence.

These five methods allow us to perform a per-iteration comparison of the progress achieved by each method. FISTA and the ISTA-BB-LS method are known to be efficient in practice and serve as a useful benchmark.

Our first three test problems have the following form, which is sometimes called the elastic net regularization problem [37],

$$\min_x \tfrac{1}{2}\|y - Bx\|^2 + \gamma\|x\|^2 + \tau\|x\|_1. \tag{5.1}$$

The data y and B was obtained from three different data sets that we call `spectra`, `sigrec`, and `myrand`. The sources of these data sets are as follows.

Spectra. The gasoline spectra problem is a regularized linear regression problem [22]; it is available in MATLAB by typing `load spectra`. This problem has a slightly different form than (5.1) in the sense that ℓ_1 regularization is imposed on all but one of the variables (which represents the constant term in linear regression).

Sigrec. This signal recovery problem is described by Wright et al. [36]. The authors generate random sparse signals, introduce noise, and encode the signals in a lower dimensional vector by applying a random matrix multiplication. We generated an instance using the code by the authors of [36].

Myrand. We generated a random 200 variable problem using the following MATLAB commands

`B = randn(100,200); y = 200*randn(100,1),`

and employed this matrix and vector in (5.1).

Proxnewt. The 4th problem in our test set is of the form

$$\min_x \tfrac{1}{2}x^T Bx - y^T x + \gamma\|x\|^2 + \tau\|x\|_1. \tag{5.2}$$

It was generated by applying the proximal Newton method described in [6] to an ℓ_1 regularized convex optimization problem of the form $\psi(x) + \tau\|x\|_1$, where $\psi(x)$ is a logistic regression function and the data is given by the `gisette` test set in the LIBSVM repository [8]. Each iteration of the proximal Newton method computes a step by solving a subproblem of the form (1.1). We extracted one of these subproblems, and added the ℓ_2 regularization term $\gamma\|x\|^2$ to yield a problem of the form (5.2).

We created twelve versions of each of the four problems listed above, by choosing different values of γ and τ. This allowed us to create problems with various degrees of ill conditioning and different percentages on non-zeros in the solution. In our datasets, B in (5.1) and (5.2) was always rank deficient; therefore, when $\gamma = 0$ the resulting matrix A in (1.1) is singular.

The following naming conventions are used. The last digit, as in problems

$$\text{spectras}\mathbf{1}, \cdots, \text{spectras}\mathbf{4},$$

indicates one of the four values of τ that were chosen for each problem so as to generate different percentages of non-zeros in the optimal solution. The degree of ill conditioning, which is controlled by γ, is indicated in the second-to-last character, as in

$$\text{spectra}\mathbf{s}1, \quad \text{spectra}\mathbf{i}1, \quad \text{spectra}\mathbf{m}1,$$

which correspond to the **s**ingular, **i**ll conditioned and **m**oderately conditioned versions of the problem. The characteristics of the test problems are given in Appendix A.

Accuracy in the solution is measured by the ratio

$$\frac{F(x^k) - F^*}{|F^*|} = \texttt{tol}, \tag{5.3}$$

where F^* is the best known objective value for each problem. Given the nature of the five algorithms listed above, it is easy to compute and report the ratio (5.3) after each matrix-vector product computation. We initialized x^0 to the zero vector, and imposed a limit of 10,000 matrix vector products on all the runs.

In our first set of experiments, we ran the three versions of the algorithm presented in the paper (namely Algorithms 1, 2, 3) on all test problems. The results are summarized in Figure 1 using the Dolan-Moré [11] performance profiles, based on the total number of matrix vector (MV) multiplications required to reach $\texttt{tol} = 10^{-4}$. We observe from this figure that the refinements given in Section 4 pay off in that Algorithm 2 and Algorithm 3 outperform Algorithm 1.

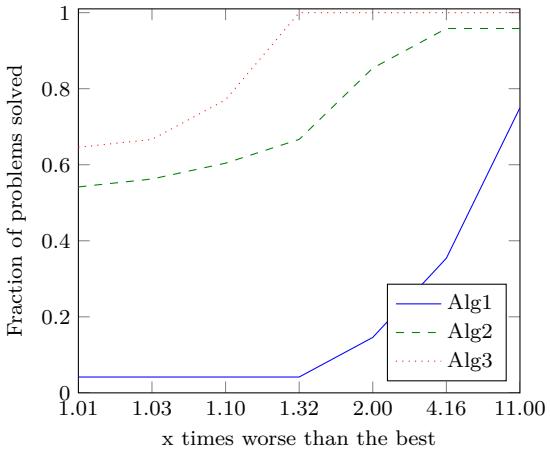


Figure 1: Logarithmic Dolan-Moré performance profiles for Algorithms 1, 2 and 3, on all problems, based on the total number of matrix-vector (MV) multiplications needed to reach $\texttt{tol} = 10^{-4}$.

Since Algorithms 2 and 3 are uniformly more efficient than Algorithm 1 (which we consider a base algorithm), we will not report results for the latter in the rest of the paper.

Next, we report in Tables 1 and 2 the results for Algorithm 2, Algorithm 3, FISTA and ISTA-BB-LS on all the test problems. Table 1 presents results when the iteration was terminated as soon as the ratio in (5.3) is less than $\texttt{tol} = 10^{-4}$, and in Table 2 for $\texttt{tol} = 10^{-10}$. Dashes signify failures to find a solution after 10,000 matrix-vector products, and bold numbers mark the best-performing algorithm. We observe from these tables that problems with an intermediate value of τ (typically) require the largest effort. This suggests that the values of τ chosen in our tests gave rise to an interesting collection of problems that range from nearly quadratic to highly regularized piecewise quadratic, with the most challenging problems in the middle range. An analysis of the data given in Tables 1 and 2 indicates that Algorithm 3 is the most efficient in these tests, but not uniformly so. Overall, we regard Algorithms 2 and 3 as promising methods for solving the regularized quadratic problem (1.1).

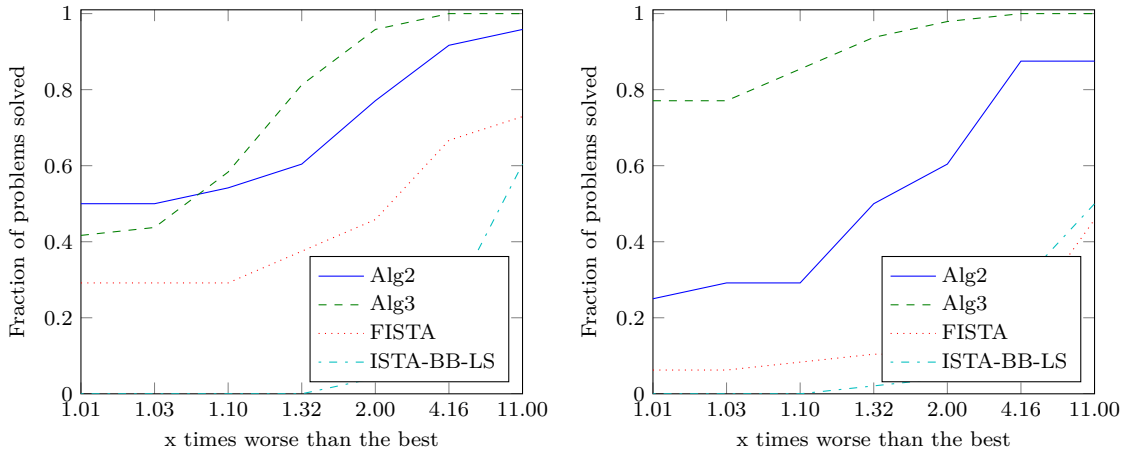
	Alg2	Alg3	FISTA	ISTA-BB-LS
myrands1	**76**	78	139	510
myrands2	89	112	**69**	185
myrands3	**25**	26	30	107
myrands4	6	**4**	14	91
myrandi1	**12**	**12**	27	105
myrandi2	221	155	**106**	324
myrandi3	**25**	26	30	107
myrandi4	6	**4**	14	91
myrandm1	**12**	**12**	27	105
myrandm2	**13**	**13**	37	106
myrandm3	187	170	**101**	297
myrandm4	**25**	26	31	107
spectras1	**4**	5	265	108
spectras2	**4**	5	264	110
spectras3	**4**	5	263	117
spectras4	**4**	5	270	115
spectrai1	**4**	5	258	114
spectrai2	**4**	5	257	117
spectrai3	**4**	5	256	115
spectrai4	60	**48**	1036	6798
spectram1	**2**	**2**	**2**	93
spectram2	**2**	**2**	**2**	93
spectram3	**5**	**5**	51	98
spectram4	100	**90**	126	352
sigrecs1	2222	1149	**446**	3786
sigrecs2	1044	525	**291**	1571
sigrecs3	106	106	**75**	147
sigrecs4	**11**	13	25	85
sigreci1	**8**	**8**	14	89
sigreci2	2257	1134	**442**	3628
sigreci3	1149	530	**291**	1519
sigreci4	**11**	13	25	85
sigrecm1	**8**	**8**	14	88
sigrecm2	65	64	**61**	169
sigrecm3	199	173	**103**	284
sigrecm4	**11**	13	25	85
proxnewts1	-	**3240**	-	-
proxnewts2	-	**5072**	6423	-
proxnewts3	7180	2108	**2026**	5280
proxnewts4	287	**120**	1582	348
proxnewti1	**279**	293	2795	-
proxnewti2	2511	**909**	2212	-
proxnewti3	4963	1949	**1702**	3521
proxnewti4	265	**149**	1499	290
proxnewtm1	32	**31**	881	302
proxnewtm2	**40**	44	784	532
proxnewtm3	221	**160**	592	802
proxnewtm4	58	**31**	472	154

Table 1: Number of matrix-vector products to reach accuracy $\texttt{tol} = 10^{-4}$.

	Alg2	Alg3	FISTA	ISTA-BB-LS
myrands1	5880	**3553**	4214	-
myrands2	**182**	212	1044	493
myrands3	**45**	48	291	137
myrands4	9	**8**	59	96
myrandi1	**384**	1024	7520	-
myrandi2	396	**317**	1632	1079
myrandi3	**45**	48	291	137
myrandi4	9	**8**	59	96
myrandm1	**50**	**50**	252	667
myrandm2	698	**324**	1447	4146
myrandm3	339	**289**	1423	845
myrandm4	**44**	48	290	135
spectras1	-	**8695**	-	-
spectras2	-	**9770**	-	-
spectras3	4965	**2349**	-	-
spectras4	-	**9930**	-	-
spectrai1	**42**	44	-	-
spectrai2	164	**147**	-	-
spectrai3	3317	**1644**	-	-
spectrai4	1899	**718**	-	-
spectram1	**10**	**10**	1897	108
spectram2	15	**13**	2024	195
spectram3	**11**	**11**	1445	200
spectram4	107	**97**	4799	466
sigrecs1	3545	2088	**1338**	7023
sigrecs2	1250	**654**	696	1809
sigrecs3	124	**121**	296	159
sigrecs4	**19**	21	145	92
sigreci1	-	**6753**	-	-
sigreci2	3652	2131	**1350**	7060
sigreci3	1358	**655**	659	1709
sigreci4	**19**	21	145	92
sigrecm1	52	**51**	114	511
sigrecm2	652	**317**	864	1224
sigrecm3	421	**354**	1301	664
sigrecm4	**18**	23	144	92
proxnewts1	-	**8984**	-	-
proxnewts2	-	**6453**	-	-
proxnewts3	7722	**2378**	-	-
proxnewts4	309	**142**	-	465
proxnewti1	3273	**1033**	-	-
proxnewti2	3645	**1697**	-	-
proxnewti3	5351	**2182**	-	6919
proxnewti4	284	**174**	-	474
proxnewtm1	124	**110**	-	2326
proxnewtm2	103	**93**	-	1590
proxnewtm3	283	**192**	-	1079
proxnewtm4	70	**44**	-	189

Table 2: Number of matrix-vector products to reach accuracy $\texttt{tol} = 10^{-10}$.

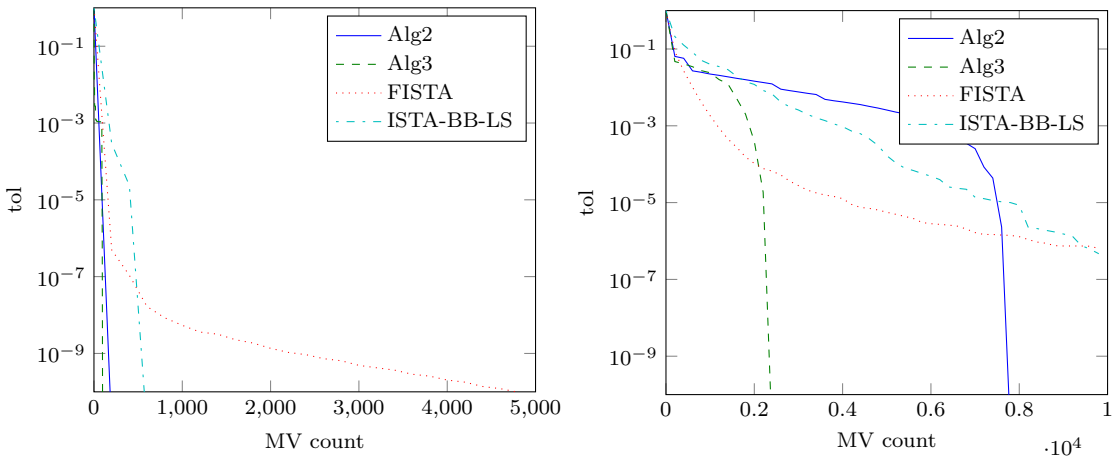
Using the data from Tables 1 and 2, we illustrate in Figure 2 the relative performance of Algorithm 2, Algorithm 3, FISTA and ISTA-BB-LS, using the Dolan-Moré profiles (based on the number of matrix-vector multiplications required for convergence). We report results for two different values of `tol`. While Algorithms 2 and 3 are efficient in the case $\texttt{tol} = 10^{-4}$, Algorithm 3 demonstrates superior performance in reaching the higher accuracy.



(a) tol = 1e-4 (b) tol = 1e-10

Figure 2: Comparison of the four Algorithms using the logarithmic Dolan-Moré profiles, based on the number of matrix-vector products required for convergence. We report results for low and high accuracy (5.3) in the objective function.

In Figure 3 we illustrate typical behavior of Algorithms 2 and 3 in our tests. We plot the ratio in (5.3) as a function of the number of matrix-vector multiplications, for problems `proxnewts3` and `spectram4`. Both plots show that Algorithms 2 and 3 are able to estimate the solution to high accuracy. These two algorithms sometimes outperform the other methods from the very start, as in Figure 3a, but in other cases Algorithms 2 and 3 show their strength later on in the runs; see Figure 3b. We note that the ISTA-BB-LS method is more efficient than FISTA when high accuracy in the solution is required.

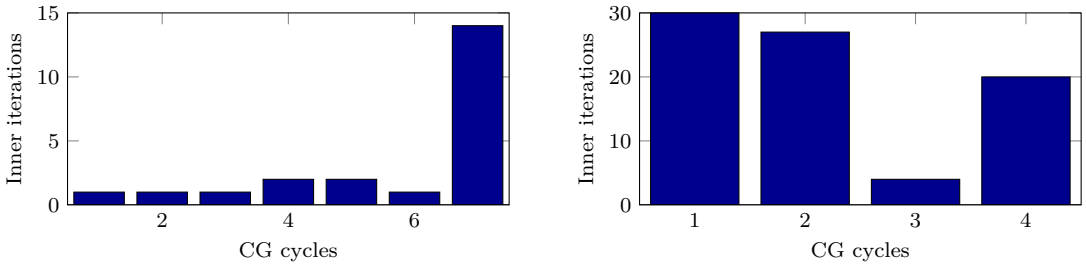


(a) Problem `spectram4` (b) Problem `proxnewts3`

Figure 3: Accuracy (5.3) in the objective function (vertical axis) as a function of the number of matrix-vector products performed (MV count)

5.2 Behavior of the CG Phase

We now discuss the behavior of the subspace CG phase, which has a great impact on the overall efficiency of the proposed algorithms. In Figure 4 we report two representative runs of Algorithm 3, given by test problems `myrandm1` and `sigreci4`. The horizontal axis labels each of the subspace phases invoked by the algorithm, and the vertical axis gives the number of CG iterations performed during that subspace phase.



(a) Alg3 on `sigreci4` (b) Alg3 on `myrandm1`

Figure 4: Number of CG iterations in each subspace phase

Figure 4a, illustrates a behavior that is often observed for moderate and large values of the penalty parameter τ, namely that the bulk of the CG iterations are performed towards the end of the run. This is desirable, as the CG phase makes a moderate contribution earlier on towards identifying the optimal active set, and is then able to compute a highly accurate solution of the problem in one (or two) CG cycles. Figure 4b, considers the case when the penalty parameter is very small, i.e., when F is nearly quadratic. We observe now that

the effort expended by the CG phase is more evenly distributed throughout the run. Since the number of CG iterations is large in most cycles, most of the matrix-vector products computed by the algorithm are used to power the CG iteration. It is reassuring that the number of CG iterations does not tail off for this problem, but that a significant number of CG steps is performed in the last cycle, yielding an accurate solution to the problem. These examples illustrate the flexibility of the approach presented in this paper.

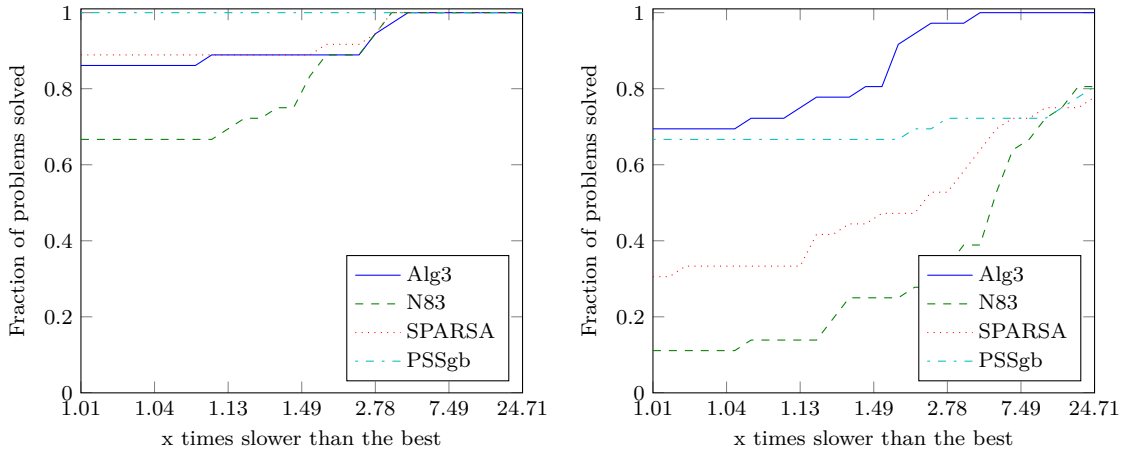
5.3 Comparisons with Established Codes

We also performed comparisons with the following three state-of-the-art codes. To facilitate our comparisons, and ease of implementation, we only considered codes written in MATLAB.

- **SPARSA** This is the well known implementation of the ISTA method described in [36]. The code can be found at `http://www.lx.it.pt/~mtf/SpaRSA/`
- **PSSgb** The motivation for the algorithm implemented in this code stems from the two-metric projection method [18] for bound constrained optimization. That method is extended to the regularized ℓ_1 problem; curvature information is incorporated in the form of a BFGS matrix. `http://www.di.ens.fr/~mschmidt/Software/thesis.html`
- **N83** Is one of the codes provided by the TFOCS package [4]. It implements the optimal first order Nesterov method described in [26]. `http://cvxr.com/tfocs/download/`

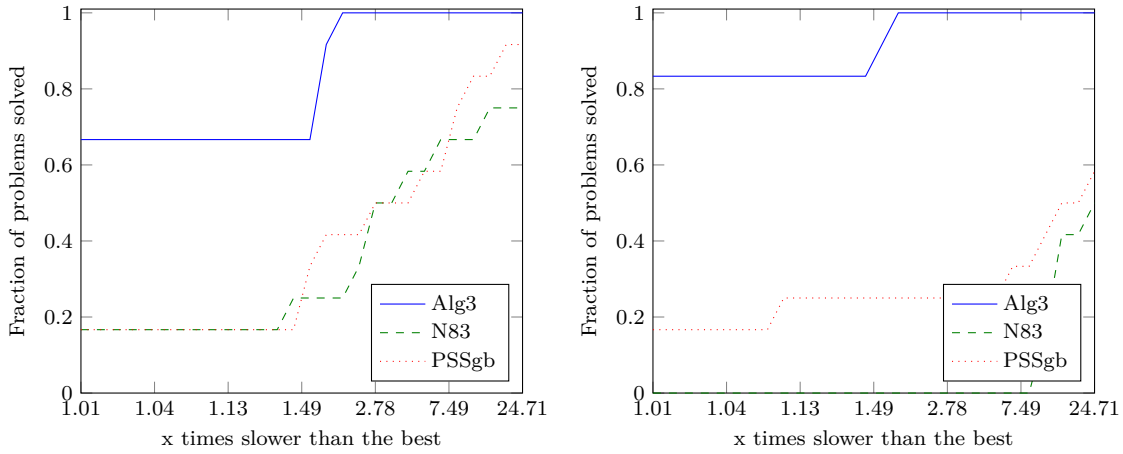
We also experimented with SALSA [1], TWIST [5] and FPC_AS [20], l1_ls [23], YALL1 [10], but these codes were not competitive on our test set with the three packages mentioned above. To limit the amount of information we only test Algorithm 3 (and not Algorithm 2) in the remainder of this section.

In Figure 5 we compare Algorithm 3 with the three codes listed above on problems `spectra`, `sigrec` and `myrand`. The figure plots the Dolan-Moré performance profiles based on CPU time; we report results for two values of the convergence tolerance (5.3). Figure 6 reports results for problem `proxnewt`; they exclude SPARSA because that code is not directly applicable to problem (5.2).



(a) `tol` $= 10^{-4}$ (b) `tol` $= 10^{-10}$

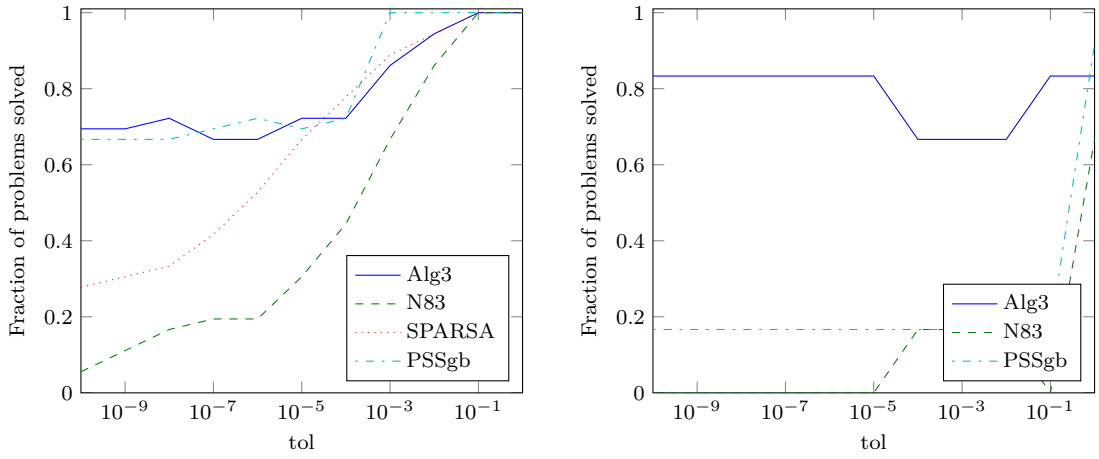
Figure 5: Comparison of Algorithm 3, SPARSA, N83 and PSSgb. The figure plots the logarithmic Dolan-Moré performance profiles based on CPU time for problems `spectra`, `sigrec` and `myrand`.



(a) `tol` $= 10^{-4}$ (b) `tol` $= 10^{-10}$

Figure 6: Comparison of Algorithm 3, N83 and PSSgb on problem `proxnewt`. The figure plots the logarithmic Dolan-Moré performance profiles based on CPU time, for two values of `tol`. SPARSA is not directly applicable to this problem.

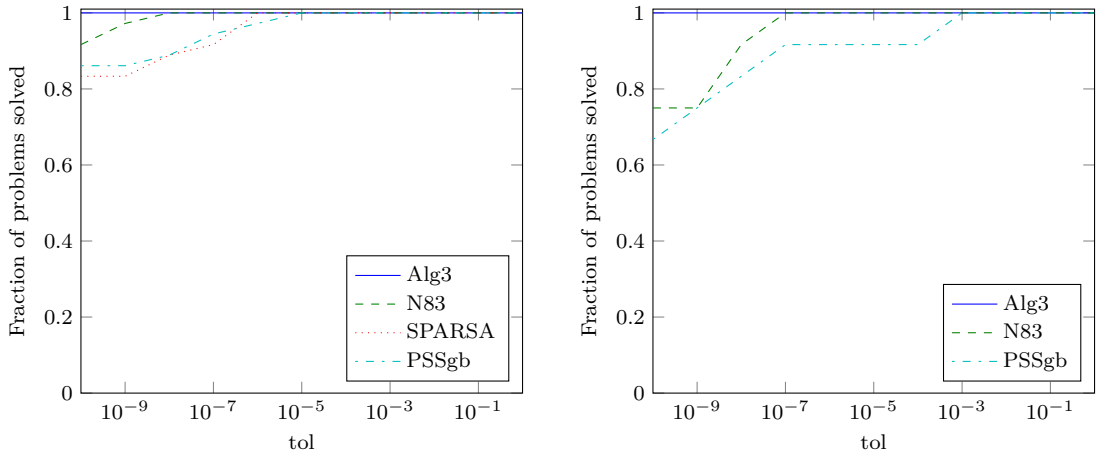
Figure 5 indicates that PSSgb dominates all other codes for accuracy `tol`$=10^{-4}$, but not for `tol`$=10^{-10}$. We now examine the behavior of the codes for intermediate values of accuracy. Figure 7 shows the fraction of problems that a method was able to solve faster than the other methods (to within 1% accuracy), as a function of the accuracy measure (5.3), in a log-scale. Figure 8 reports the fraction of problems that a method was able to solve within the allotted limit of 10,000 matrix-vector (MV) products, also as a function of the relative error in F given in (5.3).



(a) All four codes on `spectra`, `sigrec` and `myrand`.

(b) All codes, except SPARSA on problem `proxnewt`.

Figure 7: Efficiency figures. For given accuracy in the function value (horizontal axis), the plots show the percentage of problems solved to that accuracy within 1 percent of the best method, in terms of MV count



(a) All four codes on `spectra`, `sigrec` and `myrand`.

(b) All codes, except SPARSA on problem `proxnewt`.

Figure 8: Robustness figures. Fraction of problems solved within the allotted limit of matrix-vector products

Viewed as a whole, Algorithm 3 is competitive with the three state-of-the-art codes in terms of efficiency and robustness.

6 Final Remarks

In this paper, we presented a class of second-order methods for solving the ℓ_1 regularized quadratic problem (1.1) that are different other methods proposed in the literature in that they can invoke one of three possible steps at each iteration: a subspace conjugate gradient step, a first order (ISTA) step, or a relaxation step. This flexibility is designed to allow the algorithms to adapt themselves to the problem to be solved — and the results presented in the paper suggest that it is generally successful at reaching this goal. The decision of what type of step to invoke is based on the relative components of the minimum norm subgradient – an idea proposed by Dostal and Schoeberl [13] for the solution of bound constrained quadratic optimization problems.

To provide a theoretical foundation for the approach presented in this paper, we established global rates of convergence and complexity bounds based on the total amount of work expended by the algorithm (which is measured using the total number of matrix-vector products). We also gave careful consideration to the main components of the algorithm, namely the selection of an active set and the procedure for exploring a reduced space using the conjugate gradient method. One of the features of the algorithm that was particularly successful is allowing the CG iteration to cross orthants, as long as sufficient decrease in the objective function is achieved.

We performed controlled comparisons of 3 variants of our algorithm with FISTA and a popular implementation of ISTA. We also reported comparative tests with three state-of-the-art codes, SPARSA, N83 and PSSgb, and the results suggest that the method proposed in this paper is highly competitive with those algorithms.

References

[1] M.V. Afonso, J.M. Bioucas-Dias, and M. A T Figueiredo. Fast image recovery using variable splitting and constrained optimization. *Image Processing, IEEE Transactions on*, 19(9):2345–2356, 2010.

[2] G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine Learning*, pages 33–40. ACM, 2007.

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

[4] Stephen R. Becker, Emmanuel J. Candés, E. J., and Michael C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.

[5] J.M. Bioucas-Dias and M.A.T. Figueiredo. A new TwIST: two-step iterative Shrinkage/Thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, December 2007.

[6] R.H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for convex l1 regularized optimization. Technical report, Optimization Center, Northwestern University, 2013.

[7] Byrd, R., G. M Chin, J. Nocedal and F. Oztoprak. A family of second-order methods for convex L1 regularized optimization. Technical report, Optimization Center Report 2012/2, Northwestern University, 2012.

[8] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[10] Wei Deng, Wotao Yin, and Yin Zhang. Group sparse optimization by alternating direction method. *TR11-06, Department of Computational and Applied Mathematics, Rice University*, 2011.

[11] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.

[12] D.L. Donoho. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on*, 41(3):613–627, 1995.

[13] Z. Dostal and Joachim Schoeberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications*, 30(1):23–43, 2005.

[14] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2), 2004.

[15] M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586 –597, dec. 2007.

[16] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[17] Jean Jacques Fuchs. More on sparse representations in arbitrary bases. *IEEE Trans. on I.T*, pages 1341–1344, 2004.

[18] Eli M. Gafni and Dimitri P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6):936–964, 1984.

[19] Elaine T. Hale, Wotao Yin, and Yin Zhang. *Fixed-Point Continuation for L1-Minimization: Methodology and Convergence*. 2007.

[20] Elaine T. Hale, Wotao Yin, and Yin Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*, 2007.

[21] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001.

[22] John H. Kalivas. Two data sets of near infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, 37(2):255–259, June 1997.

[23] Seung-Jean Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007.

[24] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-control-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[25] A. Milzarek and M. Ulbrich. A semismooth newton method with multi-dimensional filter globalization for l1-optimization. Technical report, Technical University, Munich, 2012.

[26] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o(1/k2). *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983.

[27] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Boston: Kluwer Academic Publishers, 2004.

[28] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, second edition, 2006.

[29] Peder Olsen, Figen Oztoprak, Jorge Nocedal, and Steven Rennie. Newton-like methods for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems 25*, pages 764–772, 2012.

[30] Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356, 2003.

[31] Mark Schmidt. *Graphical Model Structure Learning with L1-Regularization*. PhD thesis, University of British Columbia, 2010.

[32] Mark Schmidt, Glenn Fung, and Romer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches suplemental material.

[33] S. Sra, S. Nowozin, and S.J. Wright. *Optimization for Machine Learning*. Mit Pr, 2011.

[34] M.J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using -constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on*, 55(5):2183–2202, May.

[35] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857, 2010.

[36] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[37] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A Dataset Details and Sparsity Patterns

The τ values for each problem were chosen by experimentation so as to span a range of solution sparsities. This is preferable to setting τ as a multiple of $\|b\|_\infty$ (as is often done in the literature based on the fact when $\tau = \|b\|_\infty$ the optimal solution to problem (1.1) is the zero vector [17]). We prefer to select the value of τ for each problem, as there sometimes is a very small range of values that yields interesting problems.

Table 3: `myrand` $n = 200$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
5.506968e+02	-	0	`myrands1`	1	101
			`myrands2`	200	108
			`myrands3`	1000	142
			`myrands4`	5000	195
5.506969e+02	5.506969e+06	1.0e-04	`myrandi1`	1.8e-04	16
			`myrandi2`	90	104
			`myrandi3`	1000	142
			`myrandi4`	5000	195
5.507968e+02	5.507968e+03	1.0e-01	`myrandm1`	9.5e-03	1
			`myrandm2`	1	44
			`myrandm3`	90	104
			`myrandm4`	1000	142

Table 4: `spectra` $n = 402$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
2.056413e+03	-	0	`spectras1`	1.0e-06	322
			`spectras2`	1.0e-04	348
			`spectras3`	1.0e-03	372
			`spectras4`	1.0e-02	389
2.056414e+03	2.056414e+06	1.0e-03	`spectrai1`	3.0e-05	2
			`spectrai2`	1.0e-03	91
			`spectrai3`	1.0e-02	313
			`spectrai4`	5.0e-01	398
2.057413e+03	2.057413e+03	1	`spectram1`	1.0e-03	1
			`spectram2`	2.0e-01	109
			`spectram3`	1	332
			`spectram4`	30	388

Table 5: `sigrec` $n = 4096$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
1.119904e+00	-	0	`sigrecs1`	5.0e-05	3549
			`sigrecs2`	2.0e-04	3816
			`sigrecs3`	5.0e-03	3860
			`sigrecs4`	1.0e-01	3973
1.119905e+00	1.119905e+06	1.0e-06	`sigreci1`	5.0e-08	828
			`sigreci2`	5.0e-05	3535
			`sigreci3`	2.0e-04	3813
			`sigreci4`	1.0e-01	3973
1.120904e+00	1.120904e+03	1.0e-03	`sigrecm1`	4.5e-07	16
			`sigrecm2`	1.0e-04	1519
			`sigrecm3`	2.0e-03	3310
			`sigrecm4`	1.0e-01	3973

Table 6: `proxnewt` $n = 5000$

$norm(A)$	$cond(A)$	γ	problem	τ	num zeros
1.103666e+02	-	0	`proxnewts1`	6.7e-06	1893
			`proxnewts2`	6.7e-05	3192
			`proxnewts3`	6.7e-04	4365
			`proxnewts4`	6.7e-03	4960
1.103667e+02	1.103667e+06	1.0e-04	`proxnewti1`	6.7e-06	1395
			`proxnewti2`	6.7e-05	3060
			`proxnewti3`	6.7e-04	4344
			`proxnewti4`	6.7e-03	4959
1.103771e+02	1.051211e+04	1.0e-02	`proxnewtm1`	6.7e-06	193
			`proxnewtm2`	6.7e-05	1283
			`proxnewtm3`	6.7e-04	3602
			`proxnewtm4`	6.7e-03	4926

Appendix B Effect of overestimating $\|A\|$ in Algorithm 3

In our experiments, we set $\hat{\alpha} = 1/L$ in Algorithm 3. Often L is not known and is hard to compute (for medium and large-scale problems computing L may take longer than running the Algorithm itself). Figure 9 shows that while $\hat{\alpha} = \frac{1}{L}$ is a good choice, Algorithm 3 is fairly insensitive to the choice of $\hat{\alpha}$, particularly if the value $1/L$ is overestimated.

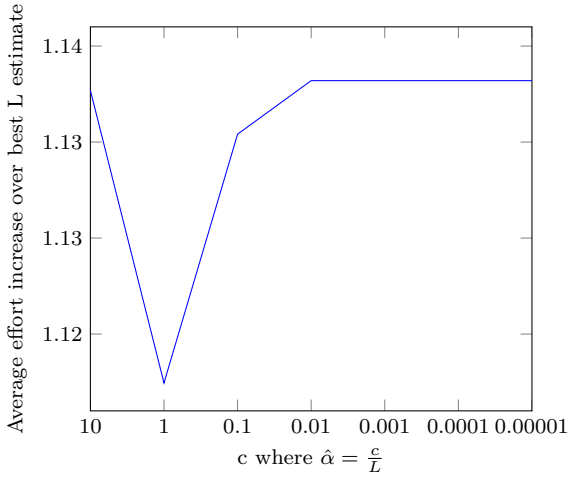


Figure 9: Average increase in matrix-vector products relative to the optimal choice for $\hat{\alpha}$ (obtained by experimentation), for various choices of $\hat{\alpha}$. The results are compiled from all 48 test problems, and the runs were stopped when `tol`$=10^{-4}$.