A FAST ALGORITHM FOR SPARSE RECONSTRUCTION BASED ON SHRINKAGE, SUBSPACE OPTIMIZATION, AND CONTINUATION[*]

ZAIWEN WEN[†], WOTAO YIN[‡], DONALD GOLDFARB[†], AND YIN ZHANG[‡]

Abstract. We propose a fast algorithm for solving the ℓ_1-regularized minimization problem $\min_{x\in\mathbb{R}^n} \mu\|x\|_1 + \|Ax-b\|_2^2$ for recovering sparse solutions to an undetermined system of linear equations $Ax = b$. The algorithm is divided into two stages that are performed repeatedly. In the first stage a first-order iterative "shrinkage" method yields an estimate of the subset of components of x likely to be nonzero in an optimal solution. Restricting the decision variables x to this subset and fixing their signs at their current values reduces the ℓ_1-norm $\|x\|_1$ to a linear function of x. The resulting subspace problem, which involves the minimization of a smaller and smooth quadratic function, is solved in the second phase. Our code FPC_AS embeds this basic two-stage algorithm in a continuation (homotopy) approach by assigning a decreasing sequence of values to μ. This code exhibits state-of-the-art performance in terms of both its speed and its ability to recover sparse signals.

Key words. ℓ_1-minimization, basis pursuit, compressive sensing, subspace optimization, active set, continuation, shrinkage

AMS subject classifications. 49M29, 65K05, 90C25, 90C06

DOI. 10.1137/090747695

1. Introduction.

Frequently, the dominant information in an image or signal is much "sparser" than the image or signal itself under a proper representation. The fundamental principal of the emerging technology of *compressive sensing* (CS) is that a K-sparse signal $\bar{x} \in \mathbb{R}^n$ can be recovered from relatively few incomplete measurements $b = A\bar{x}$ for a carefully chosen $A \in \mathbb{R}^{m\times n}$ by solving the ℓ_0-minimization problem

$$\min_{x\in\mathbb{R}^n} \|x\|_0 \text{ subject to (s.t.) } Ax = b, \tag{1.1}$$

where $\|x\|_0 := |\{i, x_i \neq 0\}|$ and $K \leq m \leq n$ (often $K \ll m \ll n$). Moreover, Candes, Romberg, and Tao (see [11, 12, 13]), Donoho [21], and their colleagues have shown that, under some reasonable conditions on $\bar{x}$ and A, the solution $\bar{x}$ of problem (1.1) can be found by solving the basis pursuit (BP) problem

$$\min_{x\in\mathbb{R}^n} \|x\|_1 \text{ s.t. } Ax = b. \tag{1.2}$$

For more information on compressive sensing, see, for example, [21, 54, 60, 61, 51, 41, 55, 37, 39, 38, 64].

[*]Received by the editors January 24, 2009; accepted for publication (in revised form) March 29, 2010; published electronically June 23, 2010.
http://www.siam.org/journals/sisc/32-4/74769.html

[†]Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027 (zw2109@columbia.edu, goldfarb@columbia.edu). The research of these authors was supported in part by NSF grant DMS 06-06712, ONR grants N000140310514 and N00014-08-1-1118, and DOE grants DE-FG01-92ER-25126 and DE-FG02-08ER58562.

[‡]Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005 (wotao.yin@rice.edu, yzhang@rice.edu). The second author's research was supported in part by NSF CAREER Award DMS-07-48839, ONR grant N00014-08-1-1101, and a Sloan Research Fellowship. The fourth author's research was supported in part by NSF grants DMS-0405831 and DMS-0811188 and ONR grant N00014-08-1-1101.

Various types of algorithms have been proposed to recover the solution of problem (1.1). Greedy algorithms work when the data satisfy certain conditions, such as the restricted isometry property [13]. These algorithms include Orthogonal Matching Pursuit (OMP) [42, 53], Stagewise OMP (StOMP) [23], CoSaMP [45], Subspace Pursuit (SP) [17], and many other variants. These algorithms, by and large, involve solving a sequence of subspace optimization problems of the form

$$\min_x \|Ax-b\|_2^2 \quad \text{s.t. } x_i = 0 \ \forall i \notin T, \tag{1.3}$$

where T is a subset of the indices $\{1, 2, \ldots, n\}$. Starting from $T = \emptyset$ and $x = 0$, OMP iteratively adds to T the index of the largest component of the current gradient $g(x)$ of $\frac{1}{2}\|Ax-b\|_2^2$ and solves (1.3) to obtain a new point x. StOMP adds one or more indices at each iteration. Rather than being a monotonically growing index set as in OMP and StOMP, at each iteration of CoSaMP and SP the index set T is the union of the indices of the K most significant components of the current point x and the K most significant components of the gradient $g(x)$.

Optimization algorithms find a solution of (1.1) by solving (1.2) or the closely related ℓ_1-regularized least squares problem

$$\min_{x\in\mathbb{R}^n} \psi_\mu(x) := \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2, \tag{1.4}$$

where $\mu > 0$. The theory for penalty functions implies that the solution (1.4) goes to the solution of (1.2) as μ goes to zero. It has been shown in [65] that (1.2) is equivalent to (1.4) for a suitable choice of b (which is different from the b in (1.4)). Furthermore, if the measurements are contaminated with noise, problem (1.4) is preferred. Other related problems include the lasso and BP denoising problems, i.e.,

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|_2^2 \ \text{ s.t. } \|x\|_1 \le t \quad \text{and} \quad \min_{x\in\mathbb{R}^n} \|x\|_1 \ \text{ s.t. } \|Ax-b\|_2 \le \sigma, \tag{1.5}$$

respectively, which are equivalent to (1.4) for an appropriate choice of the parameters t and σ.

One can transform problem (1.2) into a linear program (LP), problem (1.4) and the first problem in (1.5) into quadratic programs (QPs), and the second problem in (1.5) into a second-order cone program (SOCP). Hence, small instances of (1.2), (1.4), and (1.5) can be solved by standard LP, QP, and SOCP methods. However, computational challenges arise from the following facts. First, real-world applications are invariably large-scale. For example, there are more than a million variables in a problem to reconstruct a 1024×1024 image. Second, A is generally dense. Third, real-time or near real-time processing is required in some applications. Consequently, algorithms requiring matrix decompositions or factorizations are not practical. On the other hand, the measurement matrices A that arise in applications often correspond to partial transform matrices (e.g., discrete Fourier and cosine transforms), for which fast matrix-vector multiplications (e.g., FFT and direct cosine transform (DCT)) are available. Moreover, the sparsity of the solutions presents a unique opportunity for achieving relatively fast convergence with a first-order method. These features make the development of efficient optimization algorithms for CS applications an interesting research area. Examples of such algorithms include shrinkage-based algorithms [28, 48, 19, 4, 24, 25, 14, 33, 63, 56, 49], the interior-point algorithm $\ell_1\_\ell_s$ [36], SPGL1 [58] for the LASSO problem, NESTA [3] for the BP denoising problem, a smoothed penalty algorithm (SPA) [1] that solves problem (1.4) with the quadratic penalty replaced by an "exact" ℓ_2-penalty, the spectral gradient projection method GPSR [29], the fixed-point continuation method FPC [33] for the ℓ_1-regularized problem (1.4), and the gradient method in [46] for minimizing the more general function $J(x) + H(x)$, where J is nonsmooth, H is smooth, and both are convex.

In this paper, we propose an alternating-stage algorithm that combines the good features of both greedy algorithms and convex optimization approaches. It takes advantage of solution sparsity without requiring knowledge of the solution sparsity level K. One of the two stages uses a first-order method to obtain a working index set. The other stage uses a second-order method to solve a smooth subproblem defined by the working index set. When the working index set is accurate enough, the second stage yields an accurate solution to (1.4); otherwise, the second stage generates a good new starting point for the first stage (which then computes an improved working index set). The two stages are called alternately and integrated with continuation (homotopy) on μ. Rules are introduced to govern when to switch from one stage to the other or to continuation. The resulting algorithm exhibits state-of-the-art performance in terms of both its speed and its ability to recover sparse signals.

Our alternating two-stage algorithm behaves like an active-set method. Compared with interior-point methods, active-set methods are more robust and better able to take advantage of warm starts [47, 50]. For example, gradient projection and conjugate gradient steps have been combined to solve problems with bound constraints or linear constraints in [44, 6, 43, 7, 32], and LP and QP subproblems have been used to solve general nonlinear programs in [8, 9]. The difference between our algorithm and the active-set algorithm [40] lies in the way in which the working index set is chosen. Thanks to the solution sparsity, our approach is more aggressive and effective.

Our algorithm is different from any existing method in the literature of sparse optimization. GPSR [29] and FPC [33] employ debiasing to compute an accurate final solution, which is similar to our second stage. However, debiasing by GPSR and FPC is performed only just before termination rather than being integrated into the main iterations as in our algorithm. This integration is critical to the significant performance improvement. Our algorithm is also different from CoSaMP and SP in terms of both the working sets that are selected and the subproblems that are formulated.

The rest of this paper is organized as follows. In section 2, we introduce an abstract framework for our algorithm. In subsection 3.1, we discuss the shrinkage phase, a nonmonotone line search method, and an exact line search method. We state the formulation of the subspace optimization problem and present criteria for starting the subspace optimization phase in subsection 3.2 and then discuss methods for choosing the active set in subsection 3.3. Our continuation strategy is described in subsection 3.4, and the complete algorithm is presented in subsection 3.5. Finally, numerical results on an extensive collection of problems arising in CS are presented in section 4 to demonstrate the robustness and efficiency of our algorithm.

1.1. Preliminaries.

For convenience of notation let

$$f(x) := \frac{1}{2}\|Ax - b\|_2^2 \quad \text{and} \quad g(x) := \nabla f(x).$$

The complement of a given index set $T \subseteq \{1, 2, \ldots, n\}$ is denoted by $\overline{T}$. Let A_T and x_T denote the collections of columns and entries of A and x, whose indices are in T, respectively. For $x, y \in \mathbb{R}^n$, let $x \odot y$ denote the componentwise product of x and y, i.e., $(x \odot y)_i = x_i y_i$. Let X^* be the set of optimal solutions of (1.4). It is shown in

[33] that there exists a vector

$$g_i^* \begin{cases} = -\mu, & \max\{x_i : x \in X^*\} > 0, \\ = +\mu, & \min\{x_i : x \in X^*\} < 0, \\ \in [-\mu, \mu] & \text{otherwise,} \end{cases} \tag{1.6}$$

such that $g(x^*) \equiv g^*$ for all $x^* \in X^*$ and X^* is included in the orthant

$$\Omega^* := \{x \in \mathbb{R}^n : -\mathrm{sgn}^+(g_i^*)x_i \geq 0,\ i \in \{1, \ldots, n\}\},$$

where $\mathrm{sgn}^+(t) = 1$ if $t \geq 0$ and $\mathrm{sgn}^+(t) = -1$ otherwise. For a given vector $x \in \mathbb{R}^n$, the active set is denoted by $\mathcal{A}(x)$, and the inactive set (or support) is denoted by $\mathcal{I}(x)$; i.e.,

$$\mathcal{A}(x) := \{i \in \{1, \ldots, n\} \,|\, |x_i| = 0\} \quad \text{and} \quad \mathcal{I}(x) := \{i \in \{1, \ldots, n\} \,|\, |x_i| > 0\}. \tag{1.7}$$

The active set is further subdivided into two sets

$$\mathcal{A}_\pm(x) := \{i \in \mathcal{A}(x) \,|\, |g_i(x)| < \mu\} \quad \text{and} \quad \mathcal{A}_0(x) := \{i \in \mathcal{A}(x) \,|\, |g_i(x)| \geq \mu\}. \tag{1.8}$$

If x^* is an optimal solution of (1.4) and $i \in \mathcal{A}_0(x^*)$, then $|g_i(x^*)| = \mu$. The problem (1.4) is said to be *degenerate* at x^* if $\mathcal{A}_0(x^*) \neq \emptyset$. The components of x^* in $\mathcal{A}_0(x)$ are called degenerate, while those in $\mathcal{A}_\pm(x^*)$ are called nondegenerate.

2. Motivation and overview of the algorithm.

The first stage of our algorithm uses an iterative shrinkage procedure to approximately solve (1.4). This procedure iteratively computes

$$x^{k+1} := \mathcal{S}\left(x^k - \lambda g^k, \mu\lambda\right), \tag{2.1}$$

where $g^k := g(x^k)$, $\lambda > 0$, and for $y \in \mathbb{R}^n$ and $\nu \in \mathbb{R}$, the shrinkage operator, defined as

$$\mathcal{S}(y, \nu) := \mathrm{sgn}(y) \odot \max\{|y| - \nu, \mathbf{0}\}, \tag{2.2}$$

yields the unique minimizer of the function $\nu\|x\|_1 + \frac{1}{2}\|x - y\|_2^2$. The iteration (2.1) has been independently proposed by different groups of researchers in various contexts [4, 15, 19, 24, 25, 28, 33, 48]. Various modifications and enhancements have been applied to (2.1), which has also been generalized to certain other nonsmooth functions; see [26, 5, 29, 63].

Although (2.1) is very easy to compute, it can take more than thousands of iterations to achieve an acceptable accuracy for difficult problems. It is proved in [33] that (2.1) yields x^k with the same support and signs as those of the solution x^* of (1.4) for all k greater than a finite k^0 under mild conditions. In practice, after a moderate number of iterations, $\mathcal{I}(x^*) \subseteq \mathcal{I}(x^k)$ holds or almost holds, but it often takes many more iterations before $\mathcal{I}(x^*) = \mathcal{I}(x^k)$ and $x_i^k \approx x_i^*$ for $i \in \mathcal{I}(x^*)$. Simply speaking, shrinkage is very effective in obtaining a support superset, but it is not efficient in recovering signal values. Therefore, we were motivated to utilize the first property of shrinkage and overcome the second property.

Let us describe the results in [33] mentioned in the previous paragraph. Suppose that $f(x)$ is a twice differentiable convex function and the eigenvalues of its Hessian are uniformly bounded. Assume that x^k is generated by (2.1) with $\lambda = \lambda^k \in (0, 2/\lambda_{\max})$, where

$$\lambda_{\max} := \text{maximum eigenvalue of } \nabla^2 f(x) < \infty.$$

Then the support and the sign of the optimal solution can be identified after a finite number of steps, i.e., $x_i^k = 0$ for $i \in \mathcal{A}_{\pm}(x^*)$ and $\text{sgn}(x_i^k - \lambda^k g^k) = \text{sgn}(x_i^* - \lambda^k g^*)$ for $i \in T := \mathcal{I}(x^*) \cup \mathcal{A}_0(x^*)$ for k large enough [33]. Let Ω_T^* be the subset of Ω^* with respect to the index set T, and let $P_{\Omega_T^*}$ be the orthogonal projection onto Ω_T^*. Consequently, for k large enough, the scheme (2.1) effectively works only on T and can be shown to be equivalent to the gradient projection method $x_T^{k+1} = P_{\Omega_T^*}(x_T^k - \lambda^k \nabla \phi_\mu(x_T^k))$ for solving the subspace minimization problem

$$\min_x \phi_\mu(x) := -(g_T^*)^\top x_T + f(x) \quad \text{s.t. } x_T \in \Omega_T^* \text{ and } x_i = 0 \ \forall i \in \mathcal{A}_{\pm}(x^*). \tag{2.3}$$

Our subspace optimization approach is partially motivated by our belief that a second-order type method should be faster than the iterative shrinkage scheme for solving (2.3). Note that we may solve subproblems of the form of (2.3) before the optimal support has been identified.

Shrinkage provides a strategy similar to those used by greedy algorithms to select the set T based on information about x and the gradient $g(x)$ in problem (1.4). This connection is most obvious if we look at a shrinkage step right after greedy algorithms perform the subspace optimization step (1.3). Assume that x^{k+} is generated by subspace optimization (1.3) with an index set $T = T^k$. For simplicity, these indices are the leading indices. The optimality conditions for (1.3) are $A_{T^k}^\top(A_{T^k} x_{T^k}^{k+} - b) = 0$, which implies that the gradient $g^{k+} = (\mathbf{0}, g_{\overline{T}^k}^{k+})^\top$. Substituting g^{k+} and $x^{k+} = (x_{T^k}^{k+}, \mathbf{0})^\top$ into the shrinkage operator, we obtain

$$x^{k+1} = \mathcal{S}\left(x^{k+} - \lambda g^{k+}, \mu\lambda\right) = \begin{cases} \text{sgn}(x_i^{k+}) \max(|x_i^{k+}| - \mu\lambda, 0) & \text{if } i \in T^k, \\ \text{sgn}(\lambda g_i^{k+}) \max(|\lambda g_i^{k+}| - \mu\lambda, 0) & \text{if } i \in \overline{T}^k. \end{cases} \tag{2.4}$$

Hence, shrinkage selects indices corresponding to components x_i in the previous working set T^k whose magnitudes are larger than the threshold $\mu\lambda$ and indices corresponding to components of the gradient in the complement of T^k whose magnitudes are larger than $\mu\lambda$.

We now introduce an abstract form of our algorithm. In continuation, μ decreases over a sequence of outer iterations until reaching a prescribed value. The first stage of the algorithm is based on (2.1) and is accelerated by the use of a line search. Once a "good" approximate solution x^k of (1.4) corresponding to the current μ is obtained, the set of indices corresponding to the zero and nearly zero components of x^k are selected as a working set. In the second stage, a smooth subspace optimization problem, formed by fixing these components to zero, is solved. We present effective rules for stopping the first stage and determining the frozen components in section 3.2 below.

3. An active-set algorithm.

3.1. The shrinkage phase. We now describe the first stage of our algorithm. In the fixed-point method in [33], the parameter λ in (2.1) is fixed so that the fixed-point iteration is a contraction at every iteration. Since a bad value of λ usually slows down the rate of convergence, we choose λ dynamically to improve the performance of

Algorithm 1. An abstract active set algorithm.

```
Initialization: Choose $x^0$, $\mu$.
for $k = 0, 1, \dots$ until convergence do
S1    Shrinkage phase: Select a parameter $\lambda^k$ and compute a direction
      $d^k \leftarrow \mathcal{S}(x^k - \lambda^k g^k, \mu\lambda^k) - x^k$. Do a line search to obtain a step size $\alpha_k$
      and set the new point $x^{k+1} \leftarrow x^k + \alpha_k d^k$.
S2    if certain conditions are met then
          Subspace optimization: Determine a working set based upon $x^{k+1}$.
          Set $x^{k+1}$ to the solution of the subspace optimization problem over
          the working set.
          Reduce $\mu$.
```

shrinkage. We also incorporate a line search scheme to guarantee global convergence. The theoretical properties of our algorithm, including global convergence, R-linear convergence, and the identification of the active set after a finite number of steps, are studied in a companion paper [62].

Our line search scheme is based on properties of the search direction determined by shrinkage (2.1)–(2.2). Let $d^k := d^{(\lambda^k)}(x^k)$ denote this direction, i.e.,

$$d^{(\lambda)}(x) := x^+ - x, \quad x^+ = \mathcal{S}(x - \lambda g, \mu\lambda), \tag{3.1}$$

for $x \in \mathbb{R}^n$, $\mu > 0$, and $\lambda > 0$. Since shrinkage operator (2.2) yields the solution of the nonsmooth unconstrained minimization problem $\min_{x\in\mathbb{R}^n} \nu\|x\|_1 + \frac{1}{2}\|x-y\|_2^2$, and the latter is equivalent to the smooth constrained problem

$$\min \frac{1}{2}\|x - y\|_2^2 + \nu\xi \quad \text{s.t. } (x, \xi) \in \Omega := \{(x, \xi) \mid \|x\|_1 \le \xi\},$$

we can obtain from the optimality conditions for the latter problem that

$$(\mathcal{S}(x, \nu) - x)^\top (y - \mathcal{S}(x, \nu)) + \nu(\xi - \|\mathcal{S}(x, \nu)\|_1) \ge 0 \tag{3.2}$$

for all $x \in \mathbb{R}^n$, $(y, \xi) \in \Omega$, and $\nu > 0$ [62]. Substituting $x - \lambda g$ for x, x for y, and $\|x\|_1$ for ξ and setting $\nu = \mu\lambda$ in (3.2), we obtain

$$(\mathcal{S}(x - \lambda g, \mu\lambda) - (x - \lambda g))^\top (x - \mathcal{S}(x - \lambda g, \mu\lambda)) + \mu\lambda(\|x\|_1 - \|\mathcal{S}(x - \lambda g, \mu\lambda)\|_1) \ge 0,$$

which gives

$$g^\top d + \mu(\|x^+\|_1 - \|x\|_1) \le -\frac{1}{\lambda}\|d\|_2^2 \tag{3.3}$$

after rearranging terms. An alternative derivation of (3.3) is given in Lemma 2.1 in [57].

We can also reformulate (1.4) as

$$\min f(x) + \mu\xi \quad \text{s.t. } (x, \xi) \in \Omega, \tag{3.4}$$

whose first-order optimality conditions for a stationary point x^* are

$$\nabla f(x^*)(x - x^*) + \mu(\xi - \|x^*\|_1) \ge 0 \quad \forall (x, \xi) \in \Omega, \tag{3.5}$$

since $\xi^* = \|x^*\|_1$. Hence, d^k is similar to a gradient projection direction for solving (3.4) and should have many properties of the latter. In particular, it has been shown in [57, 62] that, for any $x^* \in \mathbb{R}^n$ and $0 < \lambda < \infty$,

$$d^{(\lambda)}(x^*) = \mathbf{0} \tag{3.6}$$

if and only if x^* is a stationary point for (1.4).

Since in our method λ is not chosen to ensure contraction, a backtracking line search is necessary to guarantee global convergence. Consequently, at each iteration, we compute the next point as $x^{k+1} = x^k + \alpha_k d^k$, where $\alpha_k = \rho^h$, $0 < \rho < 1$, and h is the smallest integer that satisfies the Armijo-like condition

$$\psi_\mu(x^k + \rho^h d^k) \le C_k + \rho^h \sigma \Delta^k. \tag{3.7}$$

Here C_k is a reference value with respect to the previous values $\{\psi_\mu^0, \dots, \psi_\mu^k\}$, $\sigma \in (0, 1)$, and

$$\Delta^k := (g^k)^\top d^k + \mu\|x^{k+}\|_1 - \mu\|x^k\|_1 \le 0. \tag{3.8}$$

From (3.3) and the convexity of the ℓ_1-norm, it is easy to show that there exists a (backtracking) step size that satisfies (3.7) with $C^k = \psi_\mu(x^k)$. Such a line search method is monotone since $\psi_\mu(x^{k+1}) < \psi_\mu(x^k)$. Instead of using it, we use a nonmonotone line search method based on a strategy proposed in [66] (See algorithm NMLS (Algorithm 2)). In this method, the reference value C_k in the Armijo-like condition (3.7) is taken as a convex combination of the previous reference value C_{k-1} and the function value $\psi_\mu(x^k)$, and as the iterations proceed, the weight on C_k is increased. For further information on nonmonotone line search methods, see [18, 31, 52].

Algorithm 2. Nonmonotone line search algorithm (NMLS).

```
Initialization: Choose a starting guess $x^0$ and parameters $0 < \eta < 1$, $0 < \rho < 1$,
and $0 < \lambda_m < \lambda_M < \infty$. Set $C_0 = \psi_\mu(x^0)$, $Q_0 = 1$, and $k = 0$.
while "not converge" do
    Compute a search direction: Choose a $\lambda_m \le \lambda^k \le \lambda_M$. Set
    $d^k = \mathcal{S}(x^k - \lambda^k g^k, \mu\lambda^k) - x^k$.
    Select a step size: Set $x^{k+1} = x^k + \alpha_k d^k$, where $\alpha_k = \rho^{h_k}$ and $h_k$ is the
    smallest integer such that $\alpha_k$ satisfies the nonmonotone Armijo-like
    condition (3.7).
    Update: Set $Q_{k+1} = \eta Q_k + 1$, $C_{k+1} = (\eta Q_k C_k + \psi_\mu(x^{k+1}))/Q_{k+1}$. Set
    $k \leftarrow k + 1$.
```

In [62], we show that there exists a step size satisfying the Armijo-like condition (3.7) for C_k generated in Algorithm NMLS. Therefore, every iteration of Algorithm NMLS is well defined. From that algorithm, we have

$$\psi_\mu(x^{k+1}) \le C_k \le C_{k-1} \le \cdots \le C_0 = \psi_\mu(x^0). \tag{3.9}$$

We also prove that Algorithm NMLS converges. Specifically, let $\mathcal{L}$ be the level set $\mathcal{L} := \{x \in \mathbb{R}^n : \psi_\mu(x) \le \psi_\mu(x^0)\}$, and let $\widetilde{\mathcal{L}}$ be the set of points of $x \in \mathbb{R}^n$ whose distance to $\mathcal{L}$ is at most $\sup_k \|d^k\| < \infty$. Assuming that $f(x)$ is bounded from below on $\widetilde{\mathcal{L}}$ and ∇f is Lipschitz continuous on $\widetilde{\mathcal{L}}$, we prove that the sequence $\{x^k\}$ is globally convergent in the sense that $\lim_{k\to\infty}\|d^k\| = 0$. It is also proved that the sequence $\{x^k\}$ is at least R-linearly convergent under some mild assumptions.

We now specify a strategy, which is based on the Barzilai–Borwein (BB) method [2], for choosing the parameter λ^k. The shrinkage iteration (2.1) first takes a gradient descent step with step size λ^k along the negative gradient direction g^k of the smooth function $f(x)$ and then applies the shrink operator $\mathcal{S}(\cdot,\cdot)$ to accommodate the nonsmooth term $\|x\|_1$. Hence, it is natural to choose λ^k based on the function $f(x)$ alone. Let

$$s^{k-1} = x^k - x^{k-1}, \quad y^{k-1} = g^k - g^{k-1}.$$

The BB step is defined so that it corresponds to premultiplying the negative gradient by a multiple of identity that has a quasi-Newton property; specifically,

$$\lambda^{k,BB1} = \frac{(s^{k-1})^\top s^{k-1}}{(s^{k-1})^\top y^{k-1}} \quad \text{or} \quad \lambda^{k,BB2} = \frac{(s^{k-1})^\top y^{k-1}}{(y^{k-1})^\top y^{k-1}}. \tag{3.10}$$

To avoid the parameter λ being either too small or too large, we take

$$\lambda^k = \max\{\lambda_m, \min\{\lambda^{k,BB1}, \lambda_M\}\} \quad \text{or} \quad \lambda^k = \max\{\lambda_m, \min\{\lambda^{k,BB2}, \lambda_M\}\}, \tag{3.11}$$

where $0 < \lambda_m \le \lambda_M < \infty$ are fixed parameters. We should point out that the idea of using BB steps in CS has also appeared in [29, 34, 63]. However, Algorithm NMLS requires only that λ^k be bounded, and other strategies could easily be adopted.

3.1.1. An exact line search.

An exact line search is possible if $\psi_\mu(\cdot)$ is a piecewise quadratic function [16, 30]. We want to solve

$$\begin{aligned}\min_{\alpha\in[0,1]} \psi_\mu(x+\alpha d) &:= \mu\|x+\alpha d\|_1 + \frac{1}{2}\|A(x+\alpha d) - b\|_2^2 \\ &= \mu\|x+\alpha d\|_1 + \frac{1}{2}c_1\alpha^2 + c_2\alpha + c_3,\end{aligned} \tag{3.12}$$

where $c_1 = \|Ad\|_2^2$, $c_2 = (Ad)^\top(Ax-b)$, and $c_3 = 0.5\,\|Ax-b\|_2^2$. The break points of $\psi_\mu(x+\alpha d)$ are $\{\alpha_i = -x_i/d_i,\ d_i \ne 0,\ i = 1,\dots,n\}$. Since $\alpha\in[0,1]$, we select those $\alpha_i \in [0,1]$ and sort these points together with 0 and 1 as

$$\alpha_{(0)} = 0 < \alpha_{(1)} < \cdots < \alpha_{(\kappa-1)} < 1 = \alpha_{(\kappa)}. \tag{3.13}$$

For each interval $[\alpha_{(l)}, \alpha_{(l+1)}]$ for $l = 0,\dots,\kappa$ the function $\psi_\mu(x+\alpha d)$ is a smooth quadratic function of α. Let the minimizer of the function determined by the interval $[\alpha_{(l)}, \alpha_{(l+1)}]$ be denoted by $\bar\alpha_{(l)}$. Then $\bar\alpha_{(l)}$ is the optimal solution of (3.12) if $\bar\alpha_{(l)} \in [\alpha_{(l)}, \alpha_{(l+1)}]$. Hence, we only have to search each interval to obtain the optimal solution of (3.12). This algorithm is outlined in Algorithm 3.

In our algorithm, we perform an exact line search if the Armijo-like condition (3.7) is not satisfied with a unit step size, but we still update the parameters Q_{k+1} and C_{k+1} for the next iteration. Such a hybrid method works well in practice.

3.2. The subspace optimization phase.

We now describe the second stage of our algorithm. When $\mathcal{A}(x^k)$ produced by shrinkage is a good estimate of the true active set $\mathcal{A}(x^*)$, we define subspace optimization as follows. Since $|x_i| = \operatorname{sgn}(x_i)x_i$, $\mu\operatorname{sgn}(x_i^*) = -g_i^*$, and $\operatorname{sgn}(x_i^*) = -\operatorname{sgn}(g_i^*)$ for $i \in \mathcal{I}(x^*)$, we approximate $\operatorname{sgn}(x_i^*)$ by $\operatorname{sgn}(x_i^k)$ and replace $\phi_\mu(x)$ in (2.3) by the smooth function

$$\varphi_\mu(x) := \mu\,\operatorname{sgn}(x^k_{\mathcal{I}^k})^\top x_{\mathcal{I}^k} + f(x). \tag{3.14}$$

Algorithm 3. An exact line search algorithm for solving (3.12).

```
Initialization: Compute $c_1 = \|Ad\|_2^2$, $c_2 = (Ad)^\top (Ax - b)$. Compute
$\alpha_i = -x_i/d_i$ for $d_i \neq 0$ and sort the $\alpha_i$ such that (3.13) is satisfied.
for $i = \kappa, \dots, 1$ do
    Compute $x^l = x + \alpha_{(i-1)} d$ and $x^u = x + \alpha_{(i)} d$.
    Set $\mathcal{I}^l = \{i : x_i^l \leq 0 \text{ and } x_i^u \leq 0\}$ and $\mathcal{I}^u = \{i : x_i^l \geq 0 \text{ and } x_i^u \geq 0\}$.
    Compute $\rho = \mu(\sum_{i \in \mathcal{I}^u} d_i - \sum_{i \in \mathcal{I}^l} d_i)$ and $\alpha = -(c_2 + \rho)/c_1$.
    if $\alpha_{(i-1)} \leq \alpha \leq \alpha_{(i)}$ then return $\alpha$ and exit the loop.
Return $\alpha = \alpha_i$ such that $i = \arg\min_{i=1,\dots,\kappa} \psi_\mu(x + \alpha_i d)$.
```

We require that each x_i either has the same sign as x_i^k or is zero; i.e., x is required to be in the set

$$\Omega(x^k) := \left\{ x \in \mathbb{R}^n : \operatorname{sgn}(x_i^k) x_i \geq 0, i \in \mathcal{I}(x^k) \text{ and } x_i = 0, i \in \mathcal{A}(x^k) \right\}. \tag{3.15}$$

Therefore, our subspace optimization problem is

$$\min \varphi_\mu(x) \quad \text{s.t. } x \in \Omega(x^k), \tag{3.16}$$

which can be solved by a limited-memory quasi-Newton method for problems with simple bound constraints (L-BFGS-B) [67]. In our implementations, we also consider subspace optimization without the bound constraints, i.e.,

$$\min_{x \in \mathbb{R}^n} \varphi_\mu(x) \quad \text{s.t. } x_i = 0 \ \forall i \in \mathcal{A}(x^k). \tag{3.17}$$

This is essentially an unconstrained minimization problem which can be solved by a linear conjugate gradient method or a limited-memory quasi-Newton method.

We switch to the subspace optimization phase if for some fixed constants $\delta > 0$ and $\epsilon_f, \epsilon_g \in (0,1)$ either of the two conditions

$$\frac{\lambda^{k-1} \|g^k_{\mathcal{I}(x^k)}\|_2}{\|d^{(\lambda^{k-1})}\|_2} > \delta \quad \text{and} \quad \|(|g(x^k)| - \mu)_{\mathcal{I}(x^k) \cup \mathcal{A}_0(x^k)}\|_\infty \leq \epsilon_g \max(\|x^k\|_2, 1), \tag{3.18}$$

$$|\psi_\mu^{k-1} - \psi_\mu^k| \leq \epsilon_f \max(|\psi_\mu^k|, |\psi_\mu^{k-1}|, 1) \tag{3.19}$$

is satisfied during the shrinkage phase. The justification for tests (3.18) and (3.19) is based on the convergence properties of Algorithm NMLS. On the one hand, we want to start subspace optimization as soon as possible; on the other hand, we want the active set that defines the subspace optimization problem to be as accurate as possible. If there is at least one nonzero component in x^*, then $\|g^*_{\mathcal{I}^*}\|_2 \geq \mu$ since $|g_i^*| = \mu$ for $i \in \mathcal{I}^*$ from the optimality conditions. Suppose that the sequence $\{x^k\}$ generated by the first stage converges to an optimal solution x^* of (1.4); then $g(x^k)$ converges to $g(x^*)$, and $\|d^{(\lambda^k)}(x^k)\|_2$ converges to zero from (3.6). Hence, the quantity $\lambda^{k-1} \|g^k_{\mathcal{I}(x^k)}(x^k)\|_2 / \|d^{(\lambda^{k-1})}(x^{k-1})\|_2$ tends to infinity, and the first part of condition (3.18) will be satisfied after a finite number of iterations. However, the quantity $\lambda^{k-1} \|g^k_{\mathcal{I}(x^k)}(x^k)\|_2 / \|d^{(\lambda^{k-1})}(x^{k-1})\|_2$ cannot tell us whether the current point x^k is nearly optimal or not. Hence, we also check the second condition in (3.18) in which $\|(|g(x^k)| - \mu)_{\mathcal{I}(x^k) \cup \mathcal{A}_0(x^k)}\|_\infty$ is a measure of optimality (see subsection 3.3). If it happens that the shrinkage phase converges slowly and cannot make sufficient progress after a large number of iterations, the relative change of the objective function value between two consecutive iterations usually will be small. Hence, satisfaction of condition (3.19) indicates that Algorithm NMLS is stagnating.

Suppose that subspace optimization starts from the point x^k. Clearly, $\varphi_\mu(x^k) = \psi_\mu(x^k)$ from the definition of $\varphi_\mu(x)$. We denote the (approximate) solution of the subspace optimization problem (3.16) by x^{k+1}. Since subspace optimization will not cause a zero component in $\mathcal{A}(x^k)$ to become nonzero and $\mathcal{I}(x^{k+1}) \subset \mathcal{I}(x^k)$, it follows that

$$\varphi_\mu(x^{k+1}) := \mu \operatorname{sgn}(x^k_{\mathcal{I}^k})^\top x^{k+1}_{\mathcal{I}^k} + f(x^{k+1}) \equiv \mu \operatorname{sgn}(x^{k+1}_{\mathcal{I}^{k+1}})^\top x^{k+1}_{\mathcal{I}^k} + f(x^{k+1}) =: \psi_\mu(x^{k+1}).$$

Hence, if we use a descent method to solve (3.16), x^{k+1} will satisfy $\varphi_\mu(x^{k+1}) \leq \varphi_\mu(x^k)$, and we can guarantee that there exists at least a subsequence generated by the abstract Algorithm 1 that converges. We terminate subspace optimization if the norm of the projected gradient $P_{\Omega^k}(\nabla\varphi(x^{k+1}))$ is small or the relative change of the objective function value between two consecutive iterations is small.

If the active sets provided to two subspace optimizations are identical, we refer to this as a *cycle*. It is hard to detect a cycle in practice unless we store all of the support sets that have been supplied to subspace optimization. However, it is easy to check whether there is a cycle between two consecutive subspace optimizations. In such a case, we do not start a second subspace optimization and continue doing iterative shrinkage.

3.3. Identification of the active set and measures of optimality. The efficiency of our active-set algorithm depends on how fast and how well the active set is identified. Assume that the sequence $\{x^k\}$ converges to x^*. Then there exists a finite number $\bar{k} > 0$ so that, for all $k > \bar{k}$, $\operatorname{sgn}(x^k_i) = \operatorname{sgn}(x^*_i)$ for all $i \in \mathcal{I}(x^*)$ and $|x^k_i| < \epsilon$ for all $i \in \mathcal{A}(x^*)$ if $0 < \epsilon < \min\{|x^*_i| \text{ for all } i \in \mathcal{I}(x^*)\}$. The true nonzero components that are not too small in magnitude can easily be identified. However, the true zero components may be nonzero after many iterations in practice. Hence, the size of the subspace optimization problem which equals the size of the support $\mathcal{I}(x^k)$ can be quite large. One approach is to replace the active set $\mathcal{A}(x^k)$ and the support $\mathcal{I}(x^k)$ by the sets

(3.20)
$$\mathcal{A}(x^k, \xi_k) := \{i \in \{1, \dots, n\} \mid |x^k_i| \leq \xi_k\}, \quad \mathcal{I}(x^k, \xi_k) := \{i \in \{1, \dots, n\} \mid |x^k_i| > \xi_k\},$$

where $\xi_k > 0$.

The threshold ξ_k in (3.20) can simply be set to a number $\bar{\xi}_m$ that is approximately equal to the machine accuracy. We now present some criteria for checking optimality which can also be used to choose the value of ξ_k. Let $\mathcal{A}(x^k, \xi_k)$ be divided into two sets

(3.21)
$$\mathcal{A}_\pm(x^k, \mu, \xi_k) := \{i \in \mathcal{A}(x, \xi_k) \mid |g^k_i| < \mu\}, \quad \mathcal{A}_0(x^k, \mu, \xi_k) := \{i \in \mathcal{A}(x^k, \xi_k) \mid |g^k_i| \geq \mu\}.$$

Then the value

$$\chi(x^k, \mu, \xi_k) := \|(|g^k| - \mu)_{\mathcal{I}(x^k,\xi_k) \cup \mathcal{A}_0(x^k,\mu,\xi_k)}\|_\infty \tag{3.22}$$

is a measure of the violation of the first-order optimality conditions (1.6), since $\chi(x^*, \mu, 0) = 0$ follows from the fact that $|g^*_i| = \mu$ for $i \in \mathcal{A}_0(x^*, \mu, 0)$. Suppose that x^* satisfies (1.6). Then the *complementary* conditions $x^*_i\,(|g^*_i| - \mu) = 0$ for all

$i \in \{1, \dots, n\}$ also have to be satisfied. Hence,

$$\zeta^k = \|x^k \odot (|g^k| - \mu)\|_2 \tag{3.23}$$

provides a measure of the violation of the complementary conditions at the point x^k.

To calculate ξ_k, we use an identification function

$$\rho(x^k, \xi_k) := \sqrt{\chi(x^k, \mu, \xi_k) + \zeta^k} \tag{3.24}$$

proposed in [27] for nonlinear programming that is based on the amount that the current iterate x^k violates the optimality conditions for (1.4). Specifically, we set the threshold ξ_k initially to $\xi_0 = \bar{\xi}_m$ and then update it as

$$\xi_{k+1} := \min\left(\max\left(\eta_2 \rho(x^k, \xi_k), \bar{\xi}_m\right),\ \|x^{k+1}\|_1/n\right), \tag{3.25}$$

where $0 < \eta_2 < 1$. Note that inequality $\xi_{k+1} \geq \|x^{k+1}\|_1/n$ ensures that $\mathcal{I}(x^{k+1}, \xi_{k+1}) \neq \emptyset$.

Since the cardinality of the estimate of the support $|\mathcal{I}(x^k, \xi_k)|$ can be greater than m, we check if $|\mathcal{I}(x^k, \xi_k)| \leq m$ before doing subspace optimization. If $|\mathcal{I}(x^k, \xi_k)| > m$, we set ξ_k to be $|x_{(\Pi)}|$ and recalculate the set $\mathcal{I}(x^k, \xi_k)$, where $x_{(\Pi)}$ is the component of x^k with the Πth largest magnitude and the parameter Π satisfies $1 \leq \Pi \leq m$.

3.4. The continuation (homotopy) strategy. Instead of solving problem (1.4) directly from scratch, we use a continuation (homotopy) procedure to solve a sequence of problems $\{x^*_{\mu_k} := \arg\min_{x \in \mathbb{R}^n} \psi_{\mu_k}(x)\}$, where $\mu_0 > \mu_1 > \cdots > \mu$, using the solution (or approximate solution) $x^*_{\mu_{k-1}}$ as the initial estimate of the solution to the next problem. It has been shown empirically in [33] that using the basic shrinkage scheme (2.1) to obtain each $x^*_{\mu_k}$ in a continuation strategy is far superior to applying the basic shrinkage scheme to (1.4) directly. Experiments in [58, 63] have further confirmed the effectiveness of continuation. Therefore, we embed our two-stage algorithm in a continuation procedure.

We now describe in detail our method for updating μ_k. First, we check whether or not the zero vector $\mathbf{0}$ satisfies the first-order optimality conditions (1.6). If the inequality $\|g(\mathbf{0})\|_\infty \leq \mu$ holds, the zero vector is a stationary point. Otherwise, the initial μ_0 is chosen to be $\max(\gamma_1 \|g(\mathbf{0})\|_\infty, \mu/\gamma_1)$, where $0 < \gamma_1 < 1$. For each intermediate value of μ, our algorithm needs only to compute $x(\mu)$ approximately before decreasing μ. Specifically, at the end of iteration k, the next parameter μ_{k+1} is set to a value smaller than μ_k if for $\epsilon_x \in (0, 1)$ the point x^k satisfies the scaled condition

$$\chi(x^k, \mu_k, \xi_k) \leq \epsilon_x \max(\|x^k\|_2, 1) \tag{3.26}$$

which implies that x^k is a good estimate of the solution of the problem $\min_{x \in \mathbb{R}^n} \psi_{\mu_k}(x)$. If x^k is a solution to the subspace optimization problem, we update μ_{k+1} even if condition (3.26) does not hold. A heuristic strategy for this update is to set

$$\mu_{k+1} = \gamma_1 \|g_{\mathcal{A}(x^k, \mu_k, \xi_k)}\|_\infty,$$

since by (1.6) the norm $\|g_{\mathcal{A}(x^k)}\|_\infty$ converges to a number less than or equal to μ as x^k converges to a stationary point x^*. A fixed fractional reduction of μ_k is also enforced to make sure that continuation will be terminated after a finite number of steps. Since the parameter should not be less than μ, we use in our algorithm the updating formula

$$\mu_{k+1} = \max(\gamma_1 \min(\|g_{\mathcal{A}(x^k, \mu_k, \xi_k)}\|_\infty, \mu_k), \mu). \tag{3.27}$$

Table 1
Summary of parameters and their default values for FPC_AS.

$\lambda_m, \lambda_M, \eta, \sigma$	parameters for the nonmonotone line search, $\eta = 0.85$, $\sigma = 10^{-3}$, $\lambda_m = 10^{-4}$, and $\lambda_M = 10^3$
$\bar{\xi}_m, \Pi$	$\bar{\xi}_m$ is a fixed threshold and Π is a fixed index for the hard truncation, $\bar{\xi}_m = 10^{-10}$ and $\Pi = \lfloor m/2 \rfloor$
$\epsilon_g, \delta, \gamma_2$	parameters for the subspace optimization activation rule (3.18), $\epsilon_g = 10^{-6}$, $\delta = 10$, and $\gamma_2 = 10$
γ_1	the factor to reduce the weight μ_k in continuation, $\gamma_1 = 0.1$.
ϵ, ϵ_x	tolerances for the termination rules, $\epsilon = 10^{-6}$ and $\epsilon_x = 10^{-12}$
ϵ_f	tolerance for the relative change of the objective function value corresponding to the final μ, $\epsilon_f = 10^{-20}$

3.5. The complete algorithm and default parameter values. The complete pseudocode for our algorithm FPC_AS (fixed-point continuation active set) is presented in Algorithm 4 below. FPC_AS uses about a dozen parameters, whose purposes and default values are described in Table 1. Only a few of these parameters are critical to the convergence and performance of FPC_AS. The default values of the threshold $\bar{\xi}_m$ and the index Π for the hard truncation are generally fine for most CS problems. However, in order to improve the performance of FPC_AS on relatively easy CS problems, one can let $\bar{\xi}_m$ be as large as one thousandth of the smallest magnitude of the nonzero entries of the solution and let Π be slightly more than the number of nonzeros in the solution if the estimates of the quantities are known. The values of the parameters related to the activation of subspace optimization are also critical to the performance of FPC_AS.

4. Numerical results. In order to demonstrate the effectiveness of the active-set algorithm FPC_AS (version 1.1), we tested it on four different sets of problems and compared it with the state-of-the-art codes including FPC (version 2.0) [33], spg_bp and its variant spg_bp_sub with subspace minimization in the software package SPGL1 (version 1.7) [58], NESTA (version 1.0) [3], SolveOMP in SparseLab [22], and cplex_pp [35] (the primal simplex method in CPLEX). For our comparisons, we used two different versions of FPC_AS: FPC_AS_CG, the default version that uses the linear conjugate gradient (CG) method to solve the unconstrained subspace optimization problem (3.17), and FPC_AS_BD, which uses the limited-memory BFGS method [67] to solve the bound constrained subspace optimization problem (3.16). The main parts of FPC_AS, FPC, NESTA, and spg_bp were written in MATLAB, and all tests described in this section were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4GHZ CPU and 6GB of RAM running Linux (2.6.9) and MATLAB 7.3.0.

In Table 2, we summarize a list of symbols used in the subsequent tables and figures. Since solvers often return solutions with tiny but nonzero entries that can be regarded as zero we use "nnzx" to denote the number of nonzeros in x which we estimate (as in [58]) by the minimum cardinality of a subset of the components of x that account for 99.9% of $\|x\|_1$; i.e.,

$$\text{nnzx} := \min\left\{ |\Omega| : \sum_{i\in\Omega} |x_i| > 0.999\|x\|_1 \right\} = \min\left\{ k : \sum_{i=0}^{k} |x_{(i)}| \geq 0.999\|x\|_1 \right\}. \tag{4.1}$$

Algorithm 4. FPC_AS algorithm.

```
    Choose $\mu > 0$ and $x^0$. Set parameters $\sigma, \eta, \gamma_1 \in (0, 1)$, $\bar{\xi}_m > 0$, $\epsilon, \epsilon_x, \epsilon_f, \epsilon_g > 0$,
    $\delta, \gamma_2 > 1$, $1 \le \Pi \le m$, $\mu_0 = \max(\gamma_1 \|g(\mathbf{0})\|_\infty, \mu/\gamma_1)$, $C_0 = \psi_\mu(x^0)$, $Q_0 = 1$, and
    $\xi_0 = \bar{\xi}_m$.
    for $k = 0, 1, \dots$ do
S1      Compute $\lambda^k$ by (3.11) and $d^k = x^{k+1} - x^k$, where
        $x^{k+1} = \mathcal{S}(x^k - \lambda^k g^k, \mu_k \lambda^k)$.
        Set $\alpha_k = 1$. if $\psi_{\mu_k}(x^{k+}) > C_k + \sigma \alpha_k \Delta^k$ then
            Set $\alpha_k$ as the minimizer of (3.12) or select $\alpha_k$ satisfying the Armijo
            conditions (3.7).
        Set $x^{k+1} = x^k + \alpha_k d^k$, $Q_{k+1} = \eta Q_k + 1$, and
        $C_{k+1} = (\eta Q_k C_k + \psi_\mu(x^{k+1}))/Q_{k+1}$.
        Calculate $\mathcal{I}(x^{k+1}, \xi_k)$ and $\mathcal{A}(x^{k+1}, \xi_k)$ by (3.20). Update the threshold
        $\xi_{k+1}$ by (3.25).
        if $\chi(x^{k+1}, \mu, \xi_k) \le \max(\epsilon, \epsilon_x \max(\|x^{k+1}\|, 1))$ then return the solution.
        Check rules for doing subspace optimization: set do_sub = 0.
        if $\mathcal{I}(x^k, \xi_{k-1})$ is not equal to $\mathcal{I}(x^{k+1}, \xi_k)$ then
            if $\frac{\lambda^k \|g^{k+1}_{\mathcal{I}(x^{k+1}, \xi_k)}\|}{\|d^{\lambda^k}\|_2} > \delta$ and $\chi(x^{k+1}, \mu_k, \xi_k) \le \epsilon_g \max(\|x^{k+1}\|, 1)$ then
                set do_sub = 1. Set $\delta = \gamma_2 \delta$.
            else if $\frac{|\psi^{k+1}_{\mu_k} - \psi^k_{\mu_k}|}{\max(|\psi^k_{\mu_k}|, |\psi^{k+1}_{\mu_k}|, 1)} \le \epsilon_f$ then set do_sub = 1.
S2      Do subspace optimization: if do_sub = 1 then
            if $|\mathcal{I}(x^{k+1}, \xi_k)| > m$ then set $\xi_k = |x^{k+1}_{(\Pi)}|$ and recalculate $\mathcal{I}(x^{k+1}, \xi_k)$.
            Solve the subspace optimization problem (3.17) or (3.16) to obtain a
            solution $x^{k+1}$.
            if $\chi(x^{k+1}, \mu, \xi_k) \le \max(\epsilon, \epsilon_x \max(\|x^{k+1}\|, 1))$ then return the solution.
S3      if ($\chi(x^{k+1}, \mu_k, \xi_k) \le \epsilon_x \max(\|x^{k+1}\|, 1)$ or do_sub = 1) and $\mu_k > \mu$ then
            Compute $\mu_{k+1} = \max(\gamma_1 \min(\|g_{\mathcal{A}(x^{k+1}, \xi_k)}\|_\infty, \mu_k), \mu)$.
            Set $\delta = \delta_0$, $C_{k+1} = \psi_\mu(x^{k+1})$, and $Q_{k+1} = 1$.
        else set $\mu_{k+1} = \mu_k$.
```

In order to compare the supports of x and $\bar{x}$, we first remove tiny entries of x by setting all of its entries with a magnitude smaller than $0.1|\bar{x}_m|$ to zero, where $\bar{x}_m$ is the smallest entry of $\bar{x}$ in magnitude, and then compute the quantities "sgn," "miss," and "over." If x matches $\bar{x}$ in terms of support and sign, the values of "sgn," "miss," and "over" should all be zero.

4.1. Recoverability for some "pathological" problems. We tested FPC_AS on a set of small-scale, pathological problems described in Table 3. Only the performance of FPC_AS_CG is reported because FPC_AS_BD performed similarly. The first test set includes four problems, CaltechTest1, ..., CaltechTest4 [10], which are pathological because the magnitudes of the nonzero entries of the exact solutions $\bar{x}$ lie in a large range. Such pathological problems are exaggerations of a large number of realistic problems in which the signals have both large and small entries. The second test set includes one problem, Ameth6Xmeth2K150, which is difficult because the number of nonzero elements in its solution is close to the limit of where the l_0-minimization problem (1.1) is equivalent to the BP problem (1.2). The coefficient

Table 2
Summary of symbols used in all subsequent tables and figures.

m, n	number of rows and columns of A
cpu	cpu time
$\|r\|_2$	ℓ_2-norm of the recovered residual, where $r = Ax - b$
$\|x\|_1$	ℓ_1-norm of the recovered solution
nnzx	number of the nonzeros in the recovered solution; see (4.1)
nMat	total number of matrix-vector products involving A and $A^\top$
rel.err	the relative error between the recovered solution x and the exact sparsest solution $\bar{x}$, i.e., rel.err $= \frac{\|x-\bar{x}\|}{\|\bar{x}\|}$
sgn	$\|\{i \mid x_i\bar{x}_i < 0\}\|$, the number of corresponding entries of x and $\bar{x}$ that are both nonzero but have opposite signs
miss	$\|\{i \mid x_i = 0, \bar{x}_i \neq 0\}\|$, the number of zero entries in x with a corresponding nonzero entry in $\bar{x}$
over	$\|\{i \mid x_i \neq 0, \bar{x}_i = 0\}\|$, the number of nonzero entries in x with a corresponding zero entry in $\bar{x}$

Table 3
Problem information: the last column gives the distinct orders of magnitude O_i of the nonzero entries in $\bar{x}$, as well as the number of elements N_i of $\bar{x}$ that are of order of magnitude O_i in the form of (O_i, N_i) pairs. For example, for the problem CaltechTest3, "$(10^{-1}, 31), (10^{-6}, 1)$" means that there are 31 entries in $\bar{x}$ with a magnitude of order 10^{-1} and one entry with a magnitude of order 10^{-6}.

ID	Problem	n	m	K	(Magnitude, num. of elements on this level)
1	CaltechTest1	512	128	38	$(10^5, 33)$, $(1, 5)$
2	CaltechTest2	512	128	37	$(10^5, 32)$, $(1, 5)$
3	CaltechTest3	512	128	32	$(10^{-1}, 31)$, $(10^{-6}, 1)$
4	CaltechTest4	512	102	26	$(10^4, 13)$, $(1, 12)$, $(10^{-2}, 1)$
5	Ameth6Xmeth2K150	1024	512	150	$(1, 150)$

matrix A here is the partial DCT matrix whose m rows were chosen randomly from the $n \times n$ DCT matrix.

We compared the results from FPC_AS_CG with the results from the solvers FPC, spg_bp, NESTA, and SolveOMP. We note that we were able to solve these problems using CPLEX because they are "small." We set the termination criteria sufficiently small for each solver. Specifically, we set the parameters $xtol = 10^{-10}$, $gtol = 10^{-8}$, and $mxitr = 2{*}10^4$ for FPC; the parameters $bpTol = 10^{-10}$, $optTol = 10^{-10}$, $decTol = 10^{-10}$, and $iteration = 10^4$ for spg_bp; the parameters $tolvar = 10^{-12}$ and $muf = 10^{-8}$ for NESTA; the parameter $OptTol = 10^{-12}$ for SolveOMP; and the parameters $\mu = 10^{-10}$, $\epsilon = 10^{-12}$, and $\epsilon_x = 10^{-16}$ for FPC_AS_CG. All other parameters of each solver were set to their default values. The termination criteria are not directly comparable due to the different formulations of the problems used by the solvers, but we believe that on average the chosen criteria for FPC_AS_CG are tighter than those of the other four solvers.

A summary of the computational results for all six problems is presented in Table 4. From that table, the superiority of FPC_AS_CG is obvious in this set of problems. The solutions of the BP problem (1.2) are the same as the sparsest signal $\bar{x}$ if

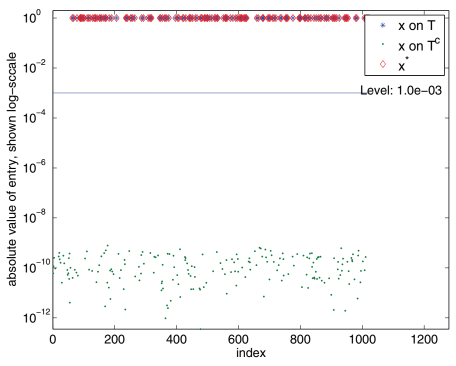
TABLE 4

Computational results for the difficult problems. The matrices A are constructed explicitly by the command "dctmtx" of MATLAB for all of the solvers.

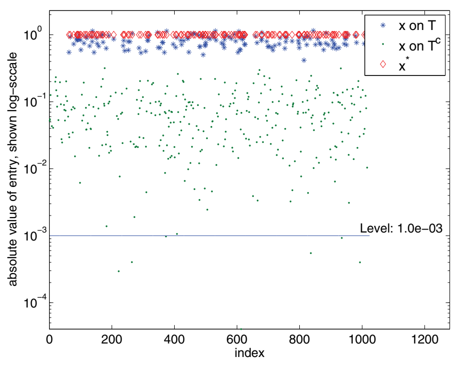
ID	Solver	CPU (sec.)	rel.err	$\|x\|_1$	$\|r\|_2$	nMat	nnzx	(sgn, miss, over)
1	FPC_AS_CG	0.105	5.04e-12	3.300e+06	1.67e-09	441	33	(0, 0, 0)
	FPC	12.048	3.09e-06	3.300e+06	1.78e-01	40001	33	(0, 0, 18)
	spg_bp	11.552	3.34e-06	3.300e+06	3.63e-03	29733	33	(0, 0, 29)
	NESTA	1.691	6.90e-06	3.300e+06	3.09e-10	12054	33	(0, 0, 69)
	SolveOMP	0.056	1.14e+00	5.090e+06	1.77e-10	256	90	(3, 16, 104)
	cplex_pp	0.245	5.10e-12	3.300e+06	8.87e-10		33	(0, 0, 0)
2	FPC_AS_CG	0.102	7.44e-14	3.200e+06	1.75e-09	322	32	(0, 0, 0)
	FPC	12.204	8.57e-08	3.200e+06	9.56e-03	40001	32	(0, 0, 0)
	spg_bp	11.227	8.47e-10	3.200e+06	3.71e-06	29238	32	(0, 0, 0)
	NESTA	3.171	8.88e-06	3.200e+06	2.67e-10	22608	32	(0, 1, 66)
	SolveOMP	0.065	1.14e+00	4.938e+06	2.14e-10	256	91	(2, 17, 107)
	cplex_pp	0.258	1.91e-13	3.200e+06	7.05e-10		32	(0, 0, 0)
3	FPC_AS_CG	0.067	1.51e-09	6.200e+00	1.26e-09	249	31	(0, 0, 0)
	FPC	11.471	3.54e-05	6.200e+00	9.18e-06	40001	31	(0, 1, 46)
	spg_bp	6.685	4.31e-09	6.200e+00	9.99e-11	17346	31	(0, 0, 0)
	NESTA	1.018	4.70e-07	6.200e+00	4.42e-16	7375	31	(0, 0, 0)
	SolveOMP	0.060	1.21e+00	9.716e+00	3.48e-16	256	92	(1, 13, 108)
	cplex_pp	0.217	4.56e-09	6.200e+00	1.21e-13		31	(0, 0, 0)
4	FPC_AS_CG	0.131	5.75e-13	1.300e+05	3.51e-09	498	13	(0, 0, 0)
	FPC	11.255	1.60e-07	1.300e+05	1.11e-03	40001	13	(0, 0, 0)
	spg_bp	8.330	3.77e-12	1.300e+05	3.81e-10	22824	13	(0, 0, 0)
	NESTA	1.062	9.03e-07	1.300e+05	1.81e-11	8142	13	(0, 0, 37)
	SolveOMP	0.009	1.12e-13	1.300e+05	3.89e-10	52	13	(0, 0, 0)
	cplex_pp	0.141	1.24e-13	1.300e+05	3.28e-11		13	(0, 0, 0)
5	FPC_AS_CG	0.362	7.25e-10	1.500e+02	2.26e-09	448	150	(0, 0, 0)
	FPC	60.765	4.84e-01	1.414e+02	3.42e-01	40001	424	(0, 1, 170)
	spg_bp	50.670	4.29e-01	1.500e+02	5.34e-03	29346	452	(0, 0, 167)
	NESTA	112.549	2.02e-01	1.501e+02	1.05e-14	86177	427	(0, 0, 51)
	SolveOMP	4.077	8.72e-01	1.867e+02	1.04e-14	1024	387	(3, 45, 214)
	cplex_pp	19.815	1.02e-12	1.500e+02	1.30e-12		150	(0, 0, 0)

we trust the solutions obtained by CPLEX. It is interesting that FPC_AS_CG is faster than CPLEX in all problems for achieving comparable accuracy. This is most obvious in problem Ameth6Xmeth2K150, in which FPC_AS_CG exhibited an approximately 50-fold improvement in terms of CPU time over CPLEX. FPC_AS_CG also performs significantly better than FPC, spg_bp, and NESTA in terms of both CPU time and the total number of matrix-vector products in these problems. FPC, spg_bp, and NESTA failed to identify the sparsest solution of Ameth6Xmeth2K150. The corresponding recovered solutions are depicted in Figure 1.

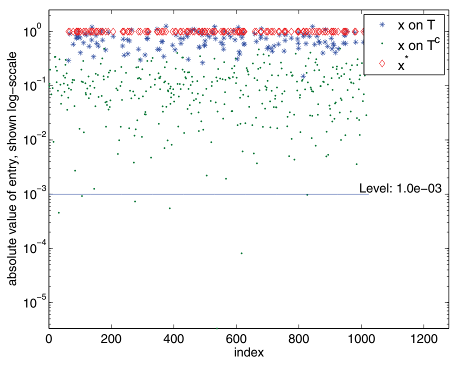
Normally, our algorithm should never fail to minimize the ℓ_1-regularized problem regardless of whether or not BP is equivalent to the ℓ_0-minimization problem. However, since our ultimate goal is to recover the sparsest solution, we observed that our algorithm could often recover sparse signals, even for problems for which ℓ_1-minimization is not equivalent to ℓ_0-minimization, if a hard truncation is added in step S2 of Algorithm 4 to identify the nonzero components. Here we investigate the recoverability success of FPC_AS_CG and cplex_pp with respect to Gaussian sparse signals and zero-one sparse signals, by using a partial DCT matrix with $n = 256$ and $m = 128$. Figure 2 depicts the empirical frequency of exact reconstruction. The numerical values on the x-axis denote the sparsity level K, while the numerical values on the y-axis represent the reconstruction rate, i.e., the fraction of 500 instances that are recovered with a relative error less than 10^{-2}. From Figure 2, we observe that for Gaussian signals, FPC_AS_CG was able to continue recovering sparse signals even



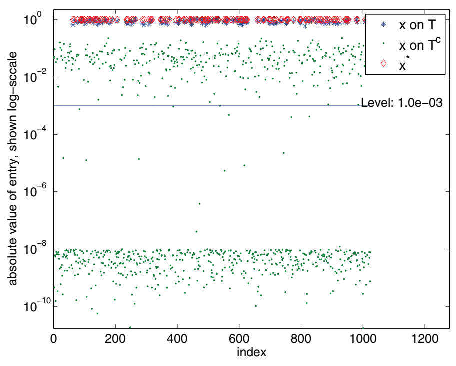
(a) FPC_AS_CG



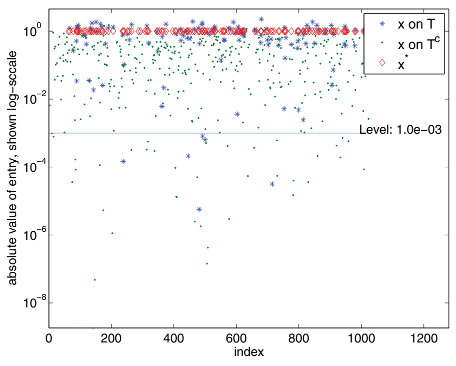
(b) FPC



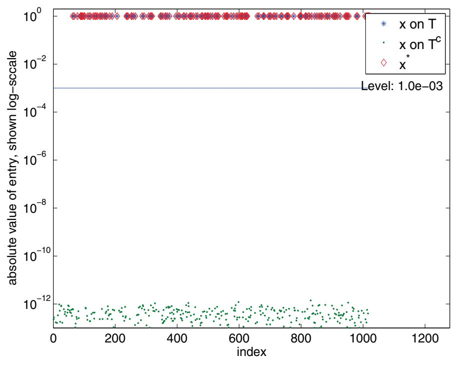
(c) spg_bp



(d) NESTA

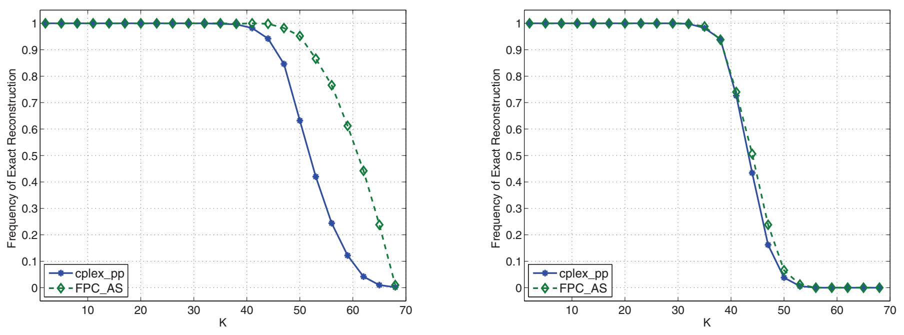


(e) SolveOMP



(f) cplex_pp

FIG. 1. *Recovered solutions of "Ameth6Xmeth2K150."*



(a) Gaussian signals: FPC_AS_CG is able to recover the sparse solution until $K \geq 41$, while the comparison with cplex_pp indicates that the equivalence between the ℓ_0 and ℓ_1 minimization problems begins to break down when $K \geq 35$.

(b) Zero-one signals: Both FPC_AS_CG and cplex_pp start to fail to reconstruct the signal when $K \geq 29$.

FIG. 2. *Frequency of exact reconstruction* (500 *replications*)*:* $m = 128$*,* $n = 256$.

after the equivalence between the ℓ_0- and ℓ_1-minimization problems started to break down as CPLEX started to find nonsparse solutions.

4.2. Quality of compressed sensing reconstruction. In this subsection, we evaluate the suitability of FPC_AS for CS on some randomly generated problems. Given the dimension of the signal n, the number of observations m, and the number of nonzeros K, we generated a random matrix A and a random vector $\bar{x}$ as follows. First, the type of matrix A was chosen from the following types:

Type 1: Gaussian matrix whose elements are generated independent and identically distributed from the normal distribution $\mathcal{N}(0,1)$;

Type 2: orthogonalized Gaussian matrix whose rows are orthogonalized using a QR decomposition;

Type 3: Bernoulli matrix whose elements are ± 1 independently with equal probability;

Type 4: Hadamard matrix H, which is a matrix of ± 1 whose columns are orthogonal;

Type 5: DCT matrix.

We randomly selected m rows from these matrices to construct the matrix A. To avoid potential numerical issues, we scaled the matrix A constructed from matrices of types 1, 3, and 4 by the largest eigenvalue of $AA^\top$. To generate the signal $\bar{x}$, we first generated the support by randomly selecting K indices between 1 and n and then assigned a value to x_i for each i in the support by one of the following eleven methods:

Type 1: a normally distributed random variable (Gaussian signal);

Type 2: a uniformly distributed random variable in$(-1,1)$;

Type 3: one (zero-one signal);

Type 4: the sign of a normally distributed random variable;

Types 5, 6, 7, 8: Types 1, 2, 3, 4 scaled by 10^5, respectively;

Type 9: Type 4, but half of the elements in the support are scaled by 10^5;

Type 10: a signal x with power-law decaying entries (also known as compressible sparse signals) whose components satisfy $|x_i| \leq c_x \cdot i^{-p}$, where we take $c_x = 10^5$ and $p = 1.5$;

Type 11: a signal x with exponentially decaying entries whose components satisfy $|x_i| \leq c_x \cdot e^{-pi}$, where we take $c_x = 1$ and $p = 0.005$.

Table 5
Robustness results.

	nMat ($\leq 10^3$)		$\|r\|_2$ ($\leq 10^{-6}$)		rel.err ($\leq 10^{-8}$)	
Solver	num.	per.	num.	per.	num.	per.
FPC_AS_CG	329	99.70	330	100.00	329	99.70
FPC_AS_BD	295	89.39	330	100.00	327	99.09
spg_bp	171	51.82	320	96.97	264	80.00
spg_bp_sub	100	30.30	321	97.27	228	69.09

Table 6
Statistics of the variants of FPC_AS.

Solver	nSubOpt		nCont		nMat in shrinkage (per.)		nMat in Sub-Opt (per.)	
	mean	std.	mean	std.	mean	std.	mean	std.
FPC_AS_CG	3.66	2.12	3.37	2.08	0.39	0.06	0.61	0.07
FPC_AS_BD	4.95	3.71	3.83	2.15	0.36	0.12	0.64	0.13

Finally, the observation b was computed as $b = A\bar{x}$. The matrices of Types 1, 2, 3, and 4 were stored explicitly, and we tested signals with three different sizes $n = 2^{10}, 2^{11}, 2^{12}$. The matrices of Type 5 were stored implicitly, and we tested signals with three different sizes $n = 2^{10}, 2^{12}, 2^{15}$. Given n, we set the number of observations $m = n/2$ and the number of nonzeros $K = \text{round}(\rho m)$ for $\rho = 0.2$ and 0.3. The above procedure gave us a total of 330 problems.

Since spg_bp has been proved to be robust in many different applications [58], we continue to compare FPC_AS with spg_bp and its variant spg_bp_sub enhanced with subspace minimization. FPC and CPLEX are not included in this comparison because they take too long to solve all problems. Although NESTA is fast on most test problems, its current release requires the measurement matrix to have orthogonal rows, i.e., $AA^\top = I$. Since NESTA can slow down significantly on problems for which $AA^\top$ is not the identity, we do not include NESTA in this comparison. We set the parameters $bpTol = 10^{-8}$, $optTol = 10^{-8}$, and $decTol = 10^{-8}$ for spg_bp and spg_bp_sub, and the parameters $\mu = 10^{-10}$, $\epsilon = 10^{-12}$, and $\epsilon_x = 10^{-16}$ for the two variants of FPC_AS. All other parameters of each solver were set to their default values.

We present several statistics on the robustness of these solvers in Table 5. In the second column, we present the number (num.) and percentage (per.) of the problems that were solved within 1000 matrix-vector products by each solver. We present the number and percentage of problems for which the norms of the computed residual were less than 10^{-6} and the relative errors between the solution x and the exact sparsest solution $\bar{x}$ were less than 10^{-8} in the third and fourth column, respectively. The active-set algorithms required fewer matrix-vector products to achieve a higher reconstruction rate than spg_bp on average. We also report the means and standard deviations of the number of subspace optimizations ("nSubOpt") and the number of continuations ("nCont") performed by FPC_AS in Table 6. The percentages of the matrix-vector products spent on various tasks in the shrinkage and subspace optimization stages are also presented.

We present our numerical results by using performance profiles as proposed in [20]. These profiles provide a way to graphically compare the quantities $t_{p,s}$, such as the number of iterations or CPU time required to solve problem p by each solver s. Define $r_{p,s}$ to be the ratio between the quantity $t_{p,s}$ obtained on problem p by solver

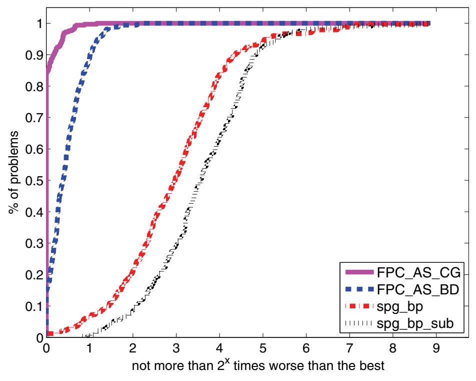
s over the lowest such quantity obtained by any of the solvers on problem p, i.e., $r_{p,s} := \frac{t_{p,s}}{\min\{t_{p,s}:1\leq s\leq n_s\}}$. Whenever solver s fails to solve problem p, the ratio $r_{p,s}$ is set to infinity or some sufficiently large number. Then, for $\tau \geq 0$,

$$\pi_s(\tau) := \frac{\text{number of problems where } \log_2(r_{p,s}) \leq \tau}{\text{total number of problems}}$$

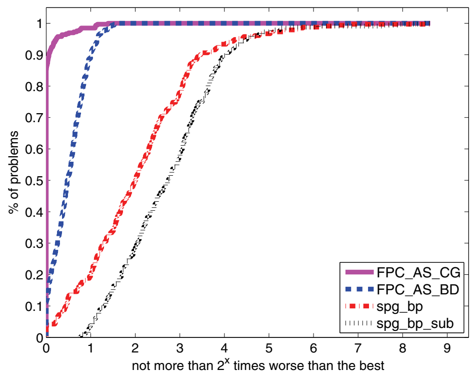
is the fraction of the test problems that were solved by solver s within a factor $2^\tau \geq 1$ of the performance obtained by the best solver. The performance plots present π_s for each solver s as a function of τ. A performance plot for the CPU time is presented in Figure 3(a). Both variants of the active-set algorithm were faster than spg_bp on our test set. A performance plot for the total number of matrix-vector products involving A or $A^\top$ is presented in 3(b). The variant FPC_AS_CG requires overall the fewest matrix-vector products for the given test set. Figure 3(c) presents a performance plot for the ℓ_1-norm $\|x\|_1$ achieved by each solver. It shows that the objective function values obtained by all four solvers are essentially identical. Figure 3(d) compares the l_2-norms of the residual $\|r\|_2$ obtained by the solvers. Again, the FPC_AS solvers outperformed the SPGL1 solvers. Figure 3(e) presents a comparison of the ratio of the number of nonzero components recovered over the number of the nonzero components in the sparsest solution. Here, as in Figure 3(c), there appears to be no discernible difference between these solvers. The relative error between the recovered solution and the exact solution is depicted in Figure 3(f). On this measure, FPC_AS_CG and FPC_AS_BD performed better than the SPGL1 solvers.

4.3. Sparco collection. In this subsection, we compare FPC_AS and spg_bp on 13 problems from the Sparco collection [59] (also see [58]). The parameters of spg_bp and spg_bp_sub were set to their default values, and the parameters $\mu = 10^{-10}$, $\epsilon = 10^{-12}$, and $\epsilon_x = 10^{-16}$ were set for the two variants of FPC_AS. A summary of the computational results is reported in Table 7. From the table, we can see that FPC_AS took less CPU time to achieve results comparable to spg_bp on many problems, such as "blocksig," "blurrycam," "blurspike," "seismic," and "spiketrn."

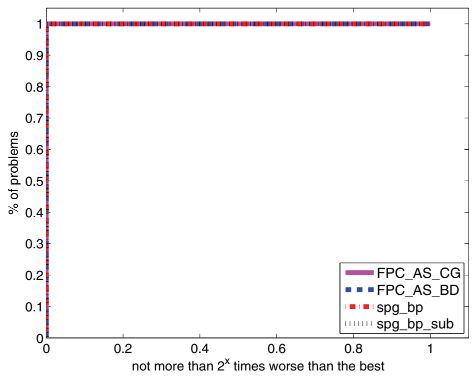
4.4. Realistic examples. In this subsection, we demonstrate the efficiency of FPC_AS_CG for solving the l_1-regularized problem on six images: a Shepp–Logan phantom available through the MATLAB Image Processing Toolbox and five medical images (three magnetic resonance images and two computed tomography scans) in the public domain. These signals have relatively sparse representations in Haar wavelets; that is, there exists an $x^* \in \mathbb{R}^n$ such that $z = Wx^*$ for a true signal $z \in \mathbb{R}^n$, where $W \in \mathbb{R}^{n\times n}$ is the Haar wavelet basis and x^* is approximately sparse. The measurements were constructed as $b = \bar{A}z$, where $\bar{A} \in \mathbb{R}^{m\times n}$ is the partial discrete cosine transformation and the number of observations $m = \tau n$ with $\tau = m/n = 0.25, 0.50, 0.75$. We then obtained approximate wavelet coefficients x of z by solving the ℓ_1-regularized problem (1.4) with $A = \bar{A}W$ and $\mu = 10^{-3}$. Finally, we completed the recovery by computing $\hat{z} = Wx$. We compared FPC_AS with FPC and spg_bp. Since x^* is not really sparse and is corrupted by noise, we used a relatively large termination criterion. We used the default parameters for FPC and the parameters $bpTol = 10^{-3}$, $optTol = 10^{-3}$, and $decTol = 10^{-3}$ for spg_bp. For FPC_AS_CG, we set the tolerances $\epsilon = 10^{-3}$, $\epsilon_x = 10^{-6}$ and set the maximal iteration number for subspace optimization to 10. The reconstruction results are summarized in Table 8. In this table, the relative error (rel.err) between the true image z and the recovered image $\hat{z}$ is defined as $= \frac{\|z-\hat{z}\|}{\|\hat{z}\|}$. FPC_AS_CG is considerably faster than FPC and spg_bp



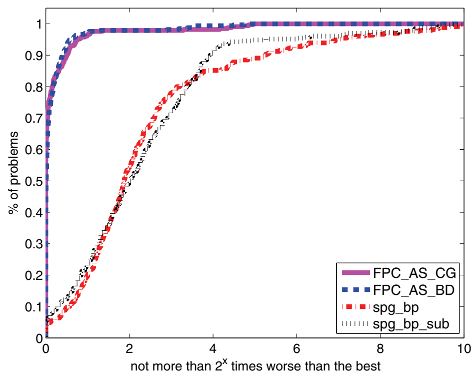
(a) cpu



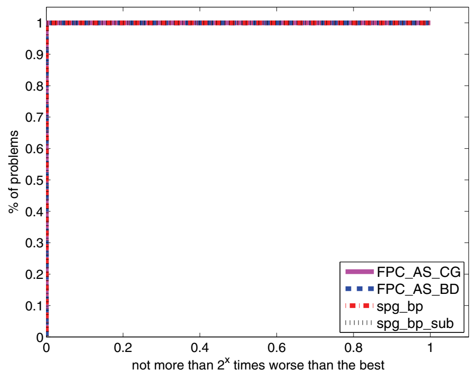
(b) nMat



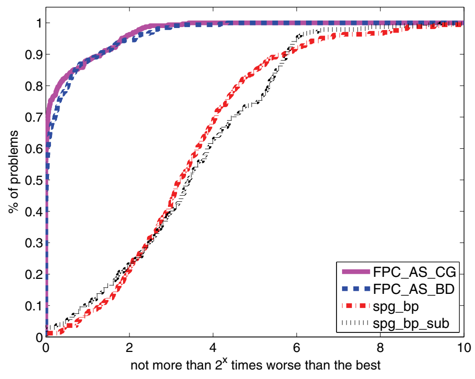
(c) $\|x\|_1$



(d) $\|r\|_2$



(e) nnzx



(f) rel.err

FIG. 3. *Performance profiles.*

Table 7

Basis pursuit comparison on Sparco. In this table, for convenience of notation, "CG" denotes FPC_AS_CG*, "BD" denotes* FPC_AS_BD*, "bp" denotes* spg_bp*, and "sub" denotes* spg_bp_sub.

FPC_AS						SPGL1					
Solver	CPU	$\|r\|$	$\|x\|_1$	nnzx	nMat	Solver	CPU	$\|r\|$	$\|x\|_1$	nnzx	nMat
						blocksig					
CG	0.02	8.4e-10	4.5e+02	71	9	bp	0.29	1.5e-14	4.5e+02	71	23
BD	0.01	8.4e-10	4.5e+02	71	9	sub	0.21	2.2e-14	4.5e+02	71	39
						blurrycam					
CG	72.19	1.3e-04	1.0e+04	62757	1653	bp	422.04	1.0e-04	1.0e+04	62755	7889
BD	212.71	8.6e-07	1.0e+04	62757	4125	sub	661.18	9.7e-05	1.0e+04	62754	14715
						blurspike					
CG	11.76	3.9e-06	3.5e+02	15592	1451	bp	56.76	9.4e-05	3.5e+02	15587	5431
BD	30.95	4.9e-07	3.5e+02	15592	2987	sub	82.68	9.9e-05	3.5e+02	15579	9465
						cosspike					
CG	0.28	1.1e-09	2.2e+02	115	180	bp	0.21	8.6e-05	2.2e+02	115	112
BD	0.38	1.1e-09	2.2e+02	115	261	sub	0.32	7.9e-05	2.2e+02	115	223
						finger					
CG	179.91	6.7e-02	6.0e+03	10595	1486	bp	421.36	6.4e-05	5.5e+03	13359	2916
BD	934.91	8.8e-07	5.6e+03	10595	7479	sub	1545.79	9.7e-05	5.5e+03	12822	12426
						gcosspike					
CG	19.90	2.5e-05	1.8e+02	236	7039	bp	7.14	1.0e-04	1.8e+02	140	2565
BD	20.71	3.3e-06	1.8e+02	249	7071	sub	9.97	7.0e-05	1.8e+02	59	4015
						jitter					
CG	0.06	3.8e-10	1.7e+00	3	42	bp	0.06	5.3e-05	1.7e+00	3	39
BD	0.06	3.9e-10	1.7e+00	3	47	sub	0.08	3.4e-05	1.7e+00	3	67
						seismic					
CG	912.91	9.5e-04	4.0e+03	28643	1788	bp	1777.43	8.9e-05	3.9e+03	23141	3047
BD	3881.17	3.0e-07	3.9e+03	28637	7427	sub	8758.55	1.0e-04	3.9e+03	19793	16691
						sgnspike					
CG	0.36	9.4e-10	2.0e+01	20	70	bp	0.30	7.3e-05	2.0e+01	20	61
BD	0.30	9.3e-10	2.0e+01	20	67	sub	0.58	7.8e-05	2.0e+01	20	130
						spiketrn					
CG	5.40	2.8e-09	1.3e+01	12	6510	bp	22.73	1.0e-04	1.3e+01	12	28927
BD	6.56	2.5e-04	1.3e+01	12	6973	sub	67.80	5.2e-03	1.3e+01	356	102227
						srcsep1					
CG	355.44	4.8e-05	1.1e+03	42737	7026	bp	203.05	9.4e-05	1.1e+03	24707	3151
BD	368.96	2.7e-04	1.1e+03	42778	6937	sub	668.38	9.9e-05	1.1e+03	23771	12039
						srcsep2					
CG	553.37	5.2e-08	1.1e+03	28112	7026	bp	215.04	9.8e-05	1.1e+03	26429	2328
BD	580.87	6.3e-08	1.1e+03	27735	6951	sub	4107.26	1.0e-04	1.1e+03	24619	52686
						yinyang					
CG	83.78	8.4e-09	2.7e+02	929	7017	bp	18.28	9.4e-05	2.6e+02	1023	1576
BD	87.31	2.1e-06	2.6e+02	922	7205	sub	73.41	3.5e-05	2.6e+02	937	6684

Table 8
Statistics of recovering medical images.

Problem		FPC			spg_bp			FPC_AS_CG		
	τ	CPU	nMat	rel.err	CPU	nMat	rel.err	CPU	nMat	rel.err
ct_thighs	0.25	45.78	403	5.7e-02	70.49	433	6.4e-02	16.30	130	5.9e-02
ct_thighs	0.50	31.34	263	9.8e-03	71.21	394	1.0e-02	15.63	120	1.0e-02
ct_thighs	0.75	17.60	147	3.2e-03	39.14	210	3.0e-03	11.28	81	3.9e-03
ct_thorax	0.25	50.80	451	7.2e-02	79.86	494	7.5e-02	18.28	142	7.1e-02
ct_thorax	0.50	37.06	315	1.9e-02	70.75	391	1.9e-02	16.63	126	2.0e-02
ct_thorax	0.75	23.55	195	6.0e-03	60.98	325	6.2e-03	16.08	116	7.3e-03
mri_abdomen	0.25	10.36	531	1.9e-01	18.64	606	2.1e-01	2.77	146	1.9e-01
mri_abdomen	0.50	8.28	417	9.7e-02	17.93	519	9.7e-02	2.26	124	9.0e-02
mri_abdomen	0.75	4.80	233	3.9e-02	11.74	322	4.0e-02	2.07	110	4.5e-02
mri_brain	0.25	46.98	417	7.3e-02	98.50	606	7.8e-02	17.46	136	7.1e-02
mri_brain	0.50	38.02	325	2.8e-02	72.28	400	3.0e-02	14.87	116	3.0e-02
mri_brain	0.75	21.18	175	7.9e-03	65.19	336	8.3e-03	18.81	141	7.8e-03
mri_pelvis	0.25	11.88	613	1.5e-01	15.26	501	1.5e-01	2.52	138	1.4e-01
mri_pelvis	0.50	7.22	379	7.3e-02	10.94	327	7.4e-02	2.58	134	7.5e-02
mri_pelvis	0.75	4.01	205	2.7e-02	9.14	258	3.1e-02	2.33	112	4.8e-02
phantom	0.25	3.47	799	3.6e-01	4.41	586	3.8e-01	1.04	136	3.6e-01
phantom	0.50	2.23	569	1.5e-01	3.72	506	1.6e-01	0.62	126	1.6e-01
phantom	0.75	1.06	259	2.7e-03	1.46	236	2.4e-03	0.58	116	4.1e-03

in terms of CPU time and the number of matrix-vector products needed to achieve comparable relative errors, except on "phantom" with $\tau = 0.75$ (although it is still faster). The reconstructed images obtained by FPC_AS_CG are depicted in Figure 4. In each row of Figure 4, the first image is the original image with a caption stating its resolution. The second, third, and fourth are recovered images with respect to $\tau = 0.25, 0.5, 0.75$, respectively, together with a caption stating the relative errors between the true image z and the recovered image $\hat{z}$.

5. Conclusions.

An alternating-stage active-set algorithm with continuation for the ℓ_1-norm regularized optimization is presented and tested. It starts with an easier problem and strategically applies a decreasing sequence of weights μ_k to the ℓ_1-norm term in the objective to gradually transform this easier problem to the given, more difficult problem with the prescribed regularization weight μ. Shrinkage is performed iteratively until the support of the current point becomes a good estimate of the support of the solution corresponding to the current weight. This estimate is used to define a subset of the solution domain over which a smaller subspace optimization problem is solved to yield a relatively accurate point. Usually, after only a small number of subproblems, a solution of high accuracy can be obtained. At each iteration of shrinkage in the first stage, a search direction is generated along with an automatically adjusting step-size parameter λ, and either an exact or an inexact line search is carried out to guarantee global convergence. In the second stage, the subspace optimization problem has a simple objective and may include bound constraints to restrict the signs of decision variables. The numerical results presented in section 4 demonstrate the effectiveness of the algorithm for solving CS problems of varying difficulties.

Acknowledgments. We want to thank Shiqian Ma and Lifeng Chen for their helpful discussions and comments. We want to thank Elaine T. Hale for providing the code for accessing the medical images in subsection 4.4, Stephen Becker for providing the CaltechTest problems and NESTA, and Michael P. Friedlander and Ewout van den Berg for providing SPGL1. The authors are grateful to Associate Editor Irad Yavneh and two anonymous referees for their detailed and valued comments and suggestions.

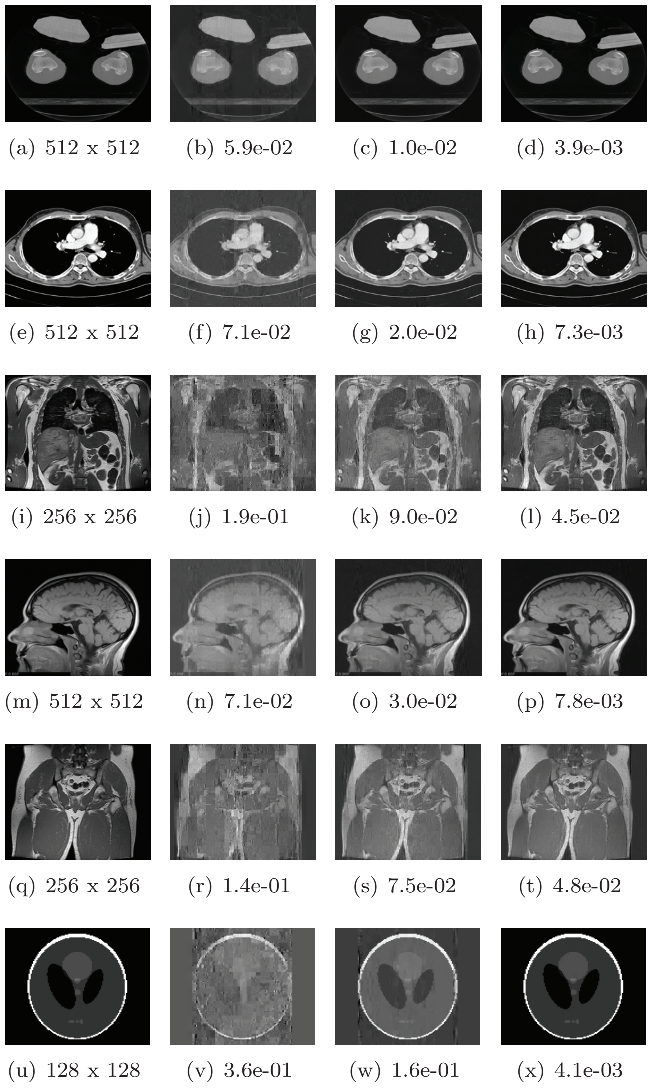


FIG. 4. *Medical image recovery by* FPC_AS_CG. *In each row,* (a) *is the original image with a caption stating its resolution, and* (b), (c), *and* (d) *are the recovered images for* $\tau = 0.25, 0.5,$ *and* 0.75*, respectively, together with a caption stating the relative errors between the true image* z *and the recovered image* $\hat{z}$.

REFERENCES

[1] S. AYBAT AND G. IYENGAR, *A first-order smoothed penalty method for compressed sensing*, SIAM J. Optim., submitted.

[2] J. BARZILAI AND J. M. BORWEIN, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[3] S. BECKER, J. BOBIN, AND E. CANDÈS, *NESTA: A Fast and Accurate First-order Method for Sparse Recovery*, Technical report, California Institute of Technology, Pasadena, 2009; available online from http://arxiv.org/abs/0904.3367.

[4] J. BECT, L. BLANC-FERAUD, G. AUBERT, AND A. CHAMBOLLE, *A ℓ_1-unified variational framework for image restoration*, in Proceedings of the European Conference on Computer Vision, Prague, Lecture Notes in Comput. Sci. 3024, Springer, Berlin, Heidelberg, 2004, pp. 1–13.

[5] J. M. Bioucas-Dias and M. A. T. Figueiredo, *A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration*, IEEE Trans. Image Process., 16 (2007), pp. 2992–3004.

[6] J. V. Burke and J. J. Moré, *On the identification of active constraints*, SIAM J. Numer. Anal., 25 (1988), pp. 1197–1211.

[7] J. V. Burke and J. J. Moré, *Exposing constraints*, SIAM J. Optim., 4 (1994), pp. 573–595.

[8] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, *An algorithm for nonlinear optimization using linear programming and equality constrained subproblems*, Math. Program., 100 (2004), pp. 27–48.

[9] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, *On the convergence of successive linear-quadratic programming algorithms*, SIAM J. Optim., 16 (2005), pp. 471–489.

[10] E. Candes and S. Becker, *private communication*, 2008.

[11] E. Candès and J. Romberg, *Quantitative robust uncertainty principles and optimally sparse decompositions*, Found. Comput. Math., 6 (2006), pp. 227–254.

[12] E. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[13] E. Candès and T. Tao, *Near optimal signal recovery from random projections: Universal encoding strategies*, IEEE Trans. Inform. Theory, 52 (2004), pp. 5406–5425.

[14] P. L. Combettes and J.-C. Pesquet, *Proximal thresholding algorithm for minimization over orthonormal bases*, SIAM J. Optim., 18 (2007), pp. 1351–1376.

[15] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Model. Simul., 4 (2005), pp. 1168–1200.

[16] A. R. Conn and J. W. Sinclair, *Quadratic Programming via a Nondifferentiable Penalty Function*, Technical report CORR 75/15, Faculty of Mathematics, University of Waterloo, Waterloo, ON, Canada, 1975.

[17] W. Dai and M. Olgica, *Subspace Pursuit for Compressive Sensing: Closing the Gap between Performance and Complexity*, Technical report, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 2008.

[18] Y. H. Dai, *On the nonmonotone line search*, J. Optim. Theory Appl., 112 (2002), pp. 315–330.

[19] C. De Mol and M. Defrise, *A note on wavelet-based inversion algorithms*, in Inverse Problems, Image Analysis, and Medical Imaging, Contemp. Math. 313, AMS, Providence, RI, 2002, pp. 85–96.

[20] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[21] D. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[22] D. Donoho, I. Drori, V. Stodden, and Y. Tsaig, *SparseLab*, 2008, http://sparselab.stanford.edu/.

[23] D. Donoho, Y. Tsaig, I. Drori, and J.-C. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, IEEE Trans. Inform. Theory, submitted.

[24] M. Elad, *Why simple shrinkage is still relevant for redundant representations?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5559–5569.

[25] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, *A wide-angle view at iterated shrinkage algorithms*, in Proceedings of the SPIE (Wavelets XII), Vol. 6701, San Diego, 2007, 670102.

[26] M. Elad, B. Matalon, and M. Zibulevsky, *Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization*, Appl. Comput. Harmon. Anal., 23 (2006), pp. 346–367.

[27] F. Facchinei, A. Fischer, and C. Kanzow, *On the accurate identification of active constraints*, SIAM J. Optim., 9 (1998), pp. 14–32.

[28] M. Figueiredo and R. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Trans. Image Process., 12 (2003), pp. 906–916.

[29] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE J. Sel. Topics Signal Process., 1 (2007), pp. 586–598.

[30] N. I. M. Gould and P. L. Toint, *An iterative working-set method for large-scale nonconvex quadratic programming*, Appl. Numer. Math., 43 (2002), pp. 109–128.

[31] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.

[32] W. W. Hager and H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.

[33] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for l_1-minimization: Methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.

[34] E. T. Hale, W. Yin, and Y. Zhang, *A numerical study of fixed-point continuation applied to compressed sensing*, J. Comput. Math., 28 (2010), pp. 170–194.

[35] Ilog, Inc., *ILOG CPLEX* 10.0, 2006, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[36] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large-scale l_1-regularized least squares*, IEEE J. Sel. Topics Signal Process., 1 (2007), pp. 606–617.

[37] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk, *Analog-to-information conversion via random demodulation*, in Proceedings of the IEEE Dallas Circuits and Systems Workshop (DCAS), Dallas, 2006.

[38] J. Laska, S. Kirolos, M. Duarte, T. Ragheb, R. Baraniuk, and Y. Massoud, *Theory and implementation of an analog-to-information converter using random demodulation*, in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), New Orleans, 2007.

[39] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. Gilbert, M. Iwen, and M. Strauss, *Random sampling for analog-to-information conversion of wideband signals*, in Proceedings of the IEEE Dallas Circuits and Systems Workshop (DCAS), Dallas, 2006.

[40] H. Lee, A. Battle, R. Raina, and A. Y. Ng, *Efficient sparse coding algorithms*, in Advances in Neural Information Processing Systems, Vol. 19, B. Schölkopf, J. Platt, and T. Hoffman, eds., MIT Press, Cambridge, MA, 2007, pp. 801–808.

[41] M. Lustig, D. Donoho, and J. M. Pauly, *Sparse MRI: The application of compressed sensing for rapid MR imaging*, Magn. Reson. Med., 58 (2007), pp. 1182–1195.

[42] S. G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Trans. Signal Process., 41 (1993), pp. 3397–3415.

[43] J. J. Moré and G. Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 377–400.

[44] J. J. Moré and G. Toraldo, *On the solution of large quadratic programming problems with bound constraints*, SIAM J. Optim., 1 (1991), pp. 93–113.

[45] D. Needell and J. A. Tropp, *Cosamp: Iterative signal recovery from incomplete and inaccurate samples*, Appl. Comput. Harmon. Anal., 26 (2008), pp. 301–321.

[46] Y. Nesterov, *Gradient Methods for Minimizing Composite Objective Function*, CORE Discussion Paper 2007/76, 2007, http://www.optimization-online.org.

[47] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., 2nd ed., Springer, New York, 2006.

[48] R. Nowak and M. Figueiredo, *Fast wavelet-based image deconvolution using the EM algorithm*, in Proceedings of the 35th IEEE Asilomar Conference on Signals, Systems, and Computers, Monterey, CA, 2001, pp. 371–375.

[49] S. Osher, Y. Mao, B. Dong, and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, Commun. Math. Sci., 8 (2010), pp. 93–111.

[50] W. Sun and Y.-X. Yuan, *Optimization Theory and Methods*, Nonlinear Programming, Springer Optimization and Its Applications 1, Springer, New York, 2006.

[51] D. Takhar, J. Laska, M. Wakin, M. Duarte, D. Baron, S. Sarvotham, K. Kelly, and R. Baraniuk, *A new compressive imaging camera architecture using optical-domain compression*, in Proceedings of Computational Imaging IV at SPIE Electronic Imaging, San Jose, CA, 2006, pp. 43–52.

[52] P. L. Toint, *An assessment of nonmonotone linesearch techniques for unconstrained optimization*, SIAM J. Sci. Comput., 17 (1996), pp. 725–739.

[53] J. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. Inform. Theory, 50 (2006), pp. 2231–2342.

[54] J. Tropp, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Trans. Inform. Theory, 51 (2006), pp. 1030–1051.

[55] J. Tropp, M. Wakin, M. Duarte, D. Baron, and R. Baraniuk, *Random filters for compressive sampling and reconstruction*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, 2006, pp. 872–875.

[56] P. Tseng and S. Yun, *Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization*, J. Optim. Theory Appl., 140 (2009), pp. 513–535.

[57] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Program., 117 (2009), pp. 387–423.

[58] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, SIAM J. Sci. Comput., 31 (2008), pp. 890–912.

[59] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. J. Herrmann, R. Saab, and O. Yilmaz, *Algorithm* 890: *Sparco: A testing framework for sparse reconstruction*, ACM Trans. Math. Software, 35 (2009), pp. 1–16.

[60] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. Kelly, and R. Baraniuk, *An architecture for compressive imaging*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), Atlanta, 2006, pp. 1273–1276.

[61] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. Kelly, and R. Baraniuk, *Compressive imaging for video representation and coding*, in Proceedings of the Picture Coding Symposium (PCS), Beijing, China, 2006.

[62] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *On the Convergence of an Active Set Method for L_1 Minimization*, Technical report, Department of Industrial Engineering and Operations Research, Columbia University, New York, 2008.

[63] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Proc., 57 (2009), pp. 2479–2493.

[64] J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for $TVL1$-$L2$ signal reconstruction from partial Fourier data*, IEEE J. Sel. Top. Signal Process., 4 (2007), pp. 288–297.

[65] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci., 1 (2008), pp. 143–168.

[66] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., 14 (2004), pp. 1043–1056.

[67] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *Algorithm* 778: *L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization*, ACM Trans. Math. Software, 23 (1997), pp. 550–560.