A FIRST-ORDER SMOOTHED PENALTY METHOD FOR COMPRESSED SENSING

N. S. AYBAT* AND G. IYENGAR†

Abstract. We propose a first-order smoothed penalty algorithm (SPA) to solve the sparse recovery problem $\min\{\|x\|_1 : Ax = b\}$. SPA is efficient as long as the matrix-vector product Ax and $A^T y$ can be computed efficiently; in particular, A need not have orthogonal rows. SPA converges to the target signal by solving a sequence of penalized optimization sub-problems, and each sub-problem is solved using Nesterov's optimal algorithm for simple sets [18, 19]. We show that the SPA iterates x_k are ϵ-feasible, i.e. $\|Ax_k - b\|_2 \leq \epsilon$ and ϵ-optimal, i.e. $|\,\|x_k\|_1 - \|x^*\|_1| \leq \epsilon$ after $\tilde{\mathcal{O}}(\epsilon^{-\frac{3}{2}})$ iterations. SPA is able to work with ℓ_1, ℓ_2 or ℓ_∞ penalty on the infeasibility, and SPA can be easily extended to solve the relaxed recovery problem $\min\{\|x\|_1 : \|Ax - b\|_2 \leq \epsilon\}$.

1. Introduction. In this paper we design a new first-order penalty-based algorithm for solving the ℓ_1-minimization problem

$$\min_{x \in \Re^n} \|x\|_1 \text{ subject to } Ax = b, \tag{1.1}$$

where ℓ_1-norm $\|x\|_1 = \sum_{i=1}^n |x(i)|$, $x(i)$ denotes the i-th component of the vector x, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and the number of equations $m \ll n$. This problem can be reformulated into a linear program (LP) and therefore, can, in theory, be solved efficiently.

LPs of the form (1.1) have recently attracted a lot of attention since they serve as the basis for a new signal processing paradigm known as *compressive sensing* (CS) [4, 5, 6, 8]. The goal in CS is to recover a sparse signal x from a small set of linear measurements or transform values $b = Ax$. Ordinarily the sparse signal would have to be recovered by solving the NP-hard ℓ_0-minimization problem

$$\min_{x \in \Re^n} \|x\|_0 \text{ subject to } Ax = b, \tag{1.2}$$

where the ℓ_0 norm $\|x\|_0 = \sum_{i=1}^n \mathbf{1}(x(i) \neq 0)$. Recently, Candes, Romberg and Tao [4, 5, 6] and Donoho [8] have shown that when the target signal x is s-sparse, i.e. only s of the n components are non-zeros, and the measurement matrix A satisfies some regularity conditions, the sparse signal can be recovered by solving the LP (1.1) with probability $1 - \mathcal{O}(e^{-\gamma n})$ for some $\gamma > 0$ when the number of measurements $m = \mathcal{O}(s \ln(n))$. Thus, in theory, the sparse signal can be recovered very efficiently.

However, in practice, solving the LP (1.1) is hard. This is because the constraint matrix A is large and dense, and the LPs are often ill-conditioned. Thus, general purpose simplex-based LP solvers are not able to efficiently solve (1.1). In typical CS applications, the problem dimension is large – $n \approx 10^6$; therefore, general purpose interior point methods that require factorization of an $m \times n$ matrix are not practical for solving LPs that arise in CS applications.

The measurement matrix A in CS applications has a lot of structure that can be exploited by special purpose algorithms. In many applications A is a partial discrete cosine transform matrix, i.e. measurement $b = Ax$ is the value of the discrete cosine transform (DCT) of the signal x for a small set of frequencies. Consequently, Ax and $A^T y$ can be computed very efficiently using the Fast Fourier Transform (FFT). In other applications A is a partial wavelet matrix; once again Ax and $A^T y$ can be computed efficiently by using forward and inverse wavelet transforms. A number of different recently proposed algorithms exploit this structural fact to efficiently solve (1.1). One class of these algorithms solve (1.1) in the Lagrangian form:

$$\min_{x \in \mathbb{R}^n} \lambda \|x\|_1 + \|Ax - b\|_2^2. \tag{1.3}$$

Figueiredo, Nowak and Wright [17] propose the GPSR algorithm that uses gradient projection method with Barzilai-Borwein steps to solve (1.3). Hale, Yin and Zhang [9, 10] propose solving (1.3) using fixed point continuation (FPC) algorithm that embeds soft-thresholding (IST) algorithm [13] in a continuation strategy. Wen, Yin, Goldfarb and Zhang [21] improve the performance of FPC by adding an active set (AS) step.

*IEOR Department, Columbia University. Email: nsa2106@columbia.edu

†IEOR Department, Columbia University. Email: gi10@columbia.edu

Yin, Osher, Goldfarb and Darbon [22] solve (1.3) using Bregman iterative regularization. In this algorithm one solves a sequence of problems of the form

$$\min_{x\in\mathbb{R}^n} \lambda\|x\|_1 + \frac{1}{2}\|Ax - f_k\|_2^2, \tag{1.4}$$

where f_k are obtained by suitably updating the measurement vector b. This method utilizes FPC for solving the unconstrained subproblems (1.4). GPSR, FPC and FPC-AS only converge to the optimal solution of (1.3). There are no known continuation schemes that ensure that these algorithm converge to the solution of (1.1). Bregman iteration based methods [22] provably converges to the optimal solution of the basis pursuit problem (1.1); however, the convergence rate is unknown.

Other algorithms for the ℓ_1-minimization problem include an iterative solver in an interior-point framework [16], and an accelerated projected gradient method [14]. In [7], Van den Berg and Friedlander adapt the nonmonotone spectral projected gradient algorithm to efficiently solve the LASSO subproblem $\Psi(t) = \{\|Ax - b\|_2^2 : \|x\|_1 \leq t\}$ and then update the LASSO parameter t using a Newton step to solve the relaxed ℓ_1-minimization problem

$$\begin{array}{ll} \min & \|x\|_1, \\ \text{s.t.} & \|Ax - b\|_2 \leq \epsilon. \end{array} \tag{1.5}$$

The algorithm in [7] provably converges to the optimal solution of the relaxed problem (1.5); however, the convergence rate is unknown.

In this paper we propose a new first-order smoothed penalty algorithm (SPA) to solve the sparse recovery problem $\min\{\|x\|_1 : Ax = b\}$. SPA employs Nesterov's optimal gradient method for non-smooth convex optimization [19] to solve the penalized subproblems. While this paper was being prepared for submission, we became aware of a technical report by Becker, Bobin and Candès [20] where they independently propose a new algorithm NESTA for solving the the relaxed ℓ_1-minimization problem (1.5) and, by setting $\epsilon = 0$, the ℓ_1-minimization problem (1.1), which is also an adaptation of Nesterov's optimal gradient method for non-smooth convex functions [19] to solve (1.5). NESTA computes an ϵ-optimal solution for (1.5) in $\mathcal{O}(\frac{1}{\epsilon})$ Nesterov updates, where each update involves computing the gradient of suitably smoothed version of the ℓ_1 norm $\|x\|_1$ and solving an optimization problem of the form

$$\min_{\{x:\|Ax-b\|_2\leq\epsilon\}} \Big\{c^T x + \frac{L}{2}\|x - z\|_2^2\Big\}. \tag{1.6}$$

See [19] and Section 2 for details on the smoothing and the update optimization problem. When $A^T A$ is an orthogonal projector, i.e. the rows of A are orthonormal (as is the case when A is a partial Fourier or DCT matrix), solving (1.6) requires one to compute one matrix-vector multiplication of the form Ax and one of the form $A^T y$, and is, therefore, very efficient in the CS context. However, when $A^T A$ is *not* orthogonal projector (as is the case when the measurement matrix A corresponds to a partial non-orthogonal wavelets transform or the partial pseudo-polar Fourier transform that arises in the context of CT imaging [1]) the complexity of the update step is $\mathcal{O}(n^3)$ and is, therefore, prohibitive for practical applications. NESTA can be embedded in a continuation scheme that allows one to compute a solution with any desired accuracy.

In this paper we propose a new first-order sequential penalty algorithm (SPA) to solve the CS decoding problem (1.1). This algorithm was, in part, motivated by the fact that a direct application of the Nesterov non-smooth optimization to (1.1) (as in NESTA) results in a very expensive update step. SPA solves (1.1) by iteratively solving a sequence of optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \big\{\lambda_k\|x\|_1 + \|Ax - b\|_2\big\},$$

where $\lambda_k \searrow 0$. The updates in Nesterov algorithm for solving the sub-problem involves computing the gradient of a suitably smoothed version of the function $\lambda_k\|x\|_1 + \|Ax - b\|_2$ and solving an *unconstrained* sub-problem of the form $\min_x \big\{c^T x + \frac{L}{2}\|x - z\|_2^2\big\}$. Since each sub-problem is *unconstrained*, optimally solving them is as simple as taking a simple gradient step $x = z - \frac{1}{L}c$. (In Section 4 we present a slightly modified version of SPA where the update step has $O(n\ln(n))$ complexity but the overall performance is superior).

We show in Section 2 that the complexity of computing gradient of a smoothed version of $\lambda\|x\|_1 + \|Ax-b\|_2$ is dominated by the time of computing Ax, and can, therefore, be computed efficiently in the CS context. Since we penalize the infeasibility by the appropriately smoothed version of $\|Ax-b\|_2$, the iterates with small infeasibility are penalized harsher in SPA as compared to algorithms employing the smooth penalty $\|Ax-b\|_2^2$, and, therefore, we expect SPA to converge faster, especially when the tolerance on feasibility is small.

The main contributions of this paper are as follows.

(a) We show that SPA converges to an optimal solution x^* of (1.1), i.e. $x^* \in \operatorname{argmin}\{\|x\|_1 : Ax = b\}$. See Theorem 2.1 and Corollary 2.2 for details. In order for the algorithm to be efficient, we only require that the matrix-vector product Ax and $A^T y$ be computed efficiently; in particular, we do not require that A has orthonormal rows. This implies that our algorithm can be used to recover compressed CT scans [1] where $A^T A$ is *not* an orthogonal projector.

(b) We show an explicit bound on the degree of sub-optimality $|\ \|x_k\|_1 - \|x^*\|_1|$ for *any* iterate x_k. Thus, the user can stop the algorithm at any iteration k with guarantee on the sub-optimality. See Theorem 2.4 for details. Using this result we also establish a convergence rate for the algorithm. We show that there exist a priori fixed parameter settings such that, for *all* small enough ϵ, the iterates x_k computed by our algorithm are ϵ-feasible, i.e. $\|Ax_k - b\|_2 \leq \epsilon$, and ϵ-optimal, $|\ \|x_k\|_1 - \|x^*\|_1| \leq \epsilon$, after $\tilde{\mathcal{O}}(\epsilon^{-\frac{3}{2}})$ iterations, where the complexity of each iteration is $\mathcal{O}(n\ln(n))$. See Theorem 2.5 for details.

(c) The SPA algorithmic framework is very flexible. One can change the penalty $\|Ax-b\|_2$ to either $\|Ax-b\|_1$ or $\|Ax-b\|_\infty$ without affecting any aspect of the theoretical or practical performance. The framework easily extends to the relaxed recovery problem $\min\{\|x\|_1 : \|Ax-b\|_p \leq \epsilon\}$, where $p = 1, 2, \infty$.

As noted earlier, NESTA [20] can be embedded in a continuation scheme that computes feasible ϵ-optimal iterate in $\mathcal{O}(\epsilon^{-1})$ iteration, where the the complexity of each iteration is $\mathcal{O}(n\ln(n))$ when $A^T A$ is orthogonal projector and $\mathcal{O}(n^3)$ otherwise. Thus, the worst case complexity of NESTA is superior to SPA when $A^T A$ is orthogonal projector. However, since the Nesterov update in SPA is a simple gradient step, we expect that in practice SPA will be competitive with NESTA even in the special case. Our numerical results reported in Section 5 do lend credence to this hypothesis.

The rest of the paper is organized as follows. In Section 2 we motivate SPA and discuss its convergence properties. In Section 3 we discuss extensions of the algorithm to the related optimization problems. In Section 4 we discuss some implementation details that significantly improve the practical performance of algorithm. In Section 5 we discuss results of our numerical experiments.

2. A smoothed penalty method for ℓ_1-minimization. We assume that A has full row rank. Consequently, A^T has full column rank. We propose solving (1.1) by solving a sequence of penalized problems of the form

$$\min\left\{\lambda\|x\|_1 + \|Ax-b\|_2\right\} \tag{2.1}$$

with $\lambda \searrow 0$. Since $P(x) = \|Ax-b\|_2$ is an *exact* penalty function for the feasible region of (1.1), there exists $\lambda^* > 0$ such that the optimal solution x^* of (1.1) is optimal for (2.1) for all $\lambda \leq \lambda^* < \infty$. However, both $\|x\|_1$ and $P(x)$ are non-smooth convex functions of x; consequently, sub-gradient based optimization methods for (2.1) are likely to perform poorly. We propose an algorithm that computes an optimal solution for ℓ_1-minimization problem (1.1) by solving an appropriately "smoothed" version of (2.1). The smoothing and the algorithm builds on the work of Nesterov [19]. Since we solve a smoothed version of the penalized optimization problem (2.1), we are not guaranteed that the optimal solution x^* of (1.1) is a solution of the smoothed optimization problem for some $\lambda > 0$.

Since $\|x\|_1 = \max_{\{u:\|u\|_\infty \leq 1\}}\{u^T x\}$, we smooth $\|x\|_1$ by setting

$$f_\mu(x) = \max_{\{u:\|u\|_\infty \leq 1\}} \left\{x^T u - \frac{\mu}{2}\|u\|_2^2\right\},$$

where $\mu > 0$. The optimal u for a particular x is given by

$$u_x(i) = \operatorname{sign}(x(i)) \min\left\{\frac{|x(i)|}{\mu}, 1\right\}, \quad i = 1, \ldots, n, \tag{2.2}$$

where

$$\text{sign}(x) = \begin{cases} 1 & x > 0, \\ 0 & x = 0, \\ -1 & x < 0. \end{cases}$$

The smoothed penalty function $f_\mu(x) = \sum_{i=1}^n H_\mu(x(i))$, where

$$H_\mu(y) = \begin{cases} \frac{y^2}{2\mu}, & 0 \le |y| \le \mu, \\ |y| - \frac{\mu}{2}, & \mu < |y|, \end{cases} \tag{2.3}$$

denotes the Hüber penalty function used in robust statistics [12]. The function $f_\mu(x)$ is convex with a Lipschitz continuous gradient $\nabla f_\mu(x) = u_x$ with the Lipschitz constant $L_\mu^f = \frac{1}{\mu}$.

We smooth the penalty function $P(x)$ by setting

$$P_\nu(x) = \max_{\{w:\|w\|_2 \le 1\}} \Big\{ (Ax-b)^T w - \frac{\nu}{2}(w^T w) \Big\} = H_\nu(\|Ax-b\|_2), \tag{2.4}$$

for $\nu > 0$. The optimal value of w for a particular x is given by

$$w_x = \begin{cases} \frac{Ax-b}{\nu}, & \|Ax-b\|_2 \le \nu, \\ \\ \frac{Ax-b}{\|Ax-b\|_2}, & \|Ax-b\|_2 > \nu. \end{cases} \tag{2.5}$$

Note that $\|w_x\|_2 = \min\{1, \nu^{-1}\|Ax-b\|_2\}$, and when $\|w_x\|_2 < 1$ we must have $w_x = \nu^{-1}(Ax-b)$.

The function $P_\nu(x)$ is convex and has Lipschitz continuous gradient $\nabla P_\nu(x) = A^T w_x$ with the Lipschitz constant $L_\nu^P = \frac{\|A\|_2^2}{\nu}$, where $\|A\|_2 = \max_{\{\|u\|_2 \le 1, \|v\|_2 \le 1\}}\{u^T A v\} = \sigma_{\max}(A)$, where $\sigma_{\max}(A)$ denotes the largest singular value of A.

One can smooth $f(x)$ (resp. $P(x)$) using any strongly convex function $h(u)$ (resp. $g(w)$) for which the maximization problem defining $f_\mu(x)$ (resp. $P_\nu(x)$) can be solved in closed form. We chose $h(u) = \frac{1}{2}\|u\|_2^2$ (resp. $g(w) = \frac{1}{2}\|w\|_2^2$) because u_x (resp. w_x) has a very simple structure which allows us to establish convergence results easily. See [19] for details of the smoothing.

We propose to solve (1.1) by solving a sequence of smoothed penalized on problems of the form

$$\min_{x \in \mathbb{R}^n} \Big\{ \lambda f_\mu(x) + P_\nu(x) \Big\}. \tag{2.6}$$

The outline of the Smoothed Penalty Algorithm (SPA) is displayed in Figure 2.1 (see Figure 4.1 for more implementation details). This is the version we use for the establishing the theoretical properties of the algorithm. The algorithm takes as input the sequence of multipliers $\{(\mu_k, \nu_k, \lambda_k, \tau_k)\}_{k \in \mathbb{Z}_+}$. In Section 4 we describe how we set these multipliers in practice.

THEOREM 2.1. *Let $\{x_k \in \mathbb{R}^n : k \in \mathbb{Z}_+\}$ denote the sequence of iterates generated by the Smoothed Penalty Algorithm (SPA) displayed in Figure 2.1 when*

(i) Smoothing parameter for $\|x\|_1$: $\mu_k \searrow 0$
(ii) Smoothing parameter for $P(x)$: $\nu_k \searrow \alpha \ge 0$
(iii) Penalty multiplier: $\lambda_k \searrow 0$
(iv) Approximate optimality parameter: $\tau_k \searrow 0$ such that $\frac{\tau_k}{\lambda_k} \to 0$.

Then, $\{x_k \in \mathbb{R}^n : k \in \mathbb{Z}_+\}$ is a bounded sequence. Let $\bar{x}$ denote any limit point of $\{x_k : k \in \mathbb{Z}_+\}$. Then $\bar{x}$ is an optimal solution of the ℓ_1-minimization problem (1.1).

Remark 2.1. *The notation $\gamma_k \searrow \eta$ (resp. $\gamma_k \nearrow \eta$) denotes that the sequence $\{\gamma_k\}$ is monotonically decreasing (resp. increasing).*

Proof. Let $x_k^* = \text{argmin}_{x \in \mathbb{R}^n} Q_k(x)$ denote the unconstrained minimizer of $Q_k(x)$ and let v denote any vector satisfying $Av = b$. Then

$$Q_k(x_k^*) \le Q_k(v) = \lambda_k f_{\mu_k}(v) \le \lambda_k \|v\|_1, \tag{2.7}$$

Outline of Smoothed Penalty Algorithm

input: multipliers $\left\{\left(\mu_k, \nu_k, \lambda_k, \tau_k\right)\right\}_{k\in\mathbb{Z}_+}$

$k \leftarrow 0$

while (Stopping Criterion not true)

do

$k \leftarrow k+1$

Let $Q_k(x) = \lambda_k f_{\mu_k}(x) + P_{\nu_k}(x)$, $\quad x_k^* = \operatorname{argmin}_{x\in\mathbb{R}^n}\left\{Q_k(x)\right\}$ and $L_k = \frac{\lambda_k}{\mu_k} + \frac{\|A\|_2^2}{\nu_k}$

1 Starting from x_{k-1} use Nesterov optimal gradient algorithm to compute x_k such that $Q_k(x_k) \le Q_k(x_k^*) + \frac{\tau_k^2}{2L_k}$

return x_k

Fig. 2.1. *Outline of Smoothed Penalty Algorithm (SPA)*

where the first inequality follows from the fact that $x_k^* = \operatorname{argmin}_{x\in\mathbb{R}^n} Q_k(x)$, the equality follows from the fact that when $Av = b$, $P_\nu(v) = 0$ for all $\nu > 0$, and last inequality follows from the fact that $f_\mu(x) \le \|x\|_1$ for all $\mu > 0$. Since the smoothed function $f_{\mu_k}(x) \ge \|x\|_1 - \frac{\mu_k n}{2}$, the smoothed penalty function $P_\nu(x) \ge 0$ for all $\nu > 0$, and Step 1 ensures $Q_k(x_k) \le Q_k(x_k^*) + \frac{\tau_k^2}{2L_k}$, it follows that

$$\|x_k\|_1 \le f_{\mu_k}(x_k) + \frac{\mu_k n}{2} \le \lambda_k^{-1} Q_k(x_k) + \frac{\mu_k n}{2} \le \lambda_k^{-1}\Big(Q_k(x_k^*) + \frac{\tau_k^2}{2L_k}\Big) + \frac{\mu_k n}{2} \le \|v\|_1 + \frac{\tau_k^2}{2\lambda_k L_k} + \frac{\mu_k n}{2}, \quad (2.8)$$

where the last inequality follows from the bound in (2.7). Since $\mu_k \searrow 0$, $\lambda_k \searrow 0$, $\tau_k/\lambda_k \to 0$, and $L_k \to \infty$, (2.8) implies that the sequence $\{x_k\}_{k\ge1}$ has a limit point. Let $\bar{x}$ denote any limit point of this sequence and let $\mathcal{K}$ denote a subsequence such that $\lim_{k\in\mathcal{K}} x_k = \bar{x}$.

It is easy to check that Q_k is convex with a Lipschitz continuous gradient with the Lipschitz constant L_k. For any such function

$$\frac{1}{2L_k}\|\nabla Q_k(x)\|_2^2 \le Q_k(x) - Q_k(x_k^*).$$

Therefore, Step 1 implies that $\|\nabla Q_k(x_k)\|_2 \le \tau_k$. The gradient $\nabla Q_k(x_k) = \lambda_k u_k + A^T w_k$ where u_k satisfies (2.2) with $x = x_k$ and $\mu = \mu_k$, and w_k satisfies (2.5) with $x = x_k$ and $\nu = \nu_k$. Therefore,

$$\|\nabla Q_k(x_k)\|_2 = \|\lambda_k u_k + A^T w_k\|_2 \le \tau_k.$$

Thus, it follows that

$$\|A^T w_k\|_2 \le (\tau_k + \lambda_k\|u_k\|_2) \le (\tau_k + \lambda_k\sqrt{n}),$$

where the last inequality follows from the fact $\|u_k\|_2 \le \sqrt{n}$. Hence,

$$\lim_{k\in\mathcal{K}} \|A^T w_k\|_2 = 0.$$

Since $\|\cdot\|_2$ is a continuous function, and A^T is assumed to have a full column rank, it follows that

$$\lim_{k\in\mathcal{K}} \|A^T w_k\|_2 = 0 \Rightarrow \|A^T \lim_{k\in\mathcal{K}} w_k\|_2 = 0 \Rightarrow \lim_{k\in\mathcal{K}} w_k = 0.$$

Thus, $\exists B > 0$ such that $\|w_k\|_2 < 1$ for $k \in \mathcal{K} \cap \{l : l \ge B\}$. Recall that (2.5) implies that $\|w_k\|_2 = \min\{1, \nu_k^{-1}\|Ax_k - b\|_2\}$ and when $\|w\|_k < 1$, we must have $w_k = \frac{1}{\nu_k}(Ax_k - b)$, i.e. $Ax_k - b = \nu_k w_k$. Consequently, $\lim_{k\in\mathcal{K}} w_k = 0$ and $\lim_k \nu_k = \alpha \ge 0$ together imply

$$A\bar{x} = b, \quad (2.9)$$

i.e. every limit point of the iterate sequence $\{x_k\}_{k\ge1}$ is feasible. Note that this proof works for any finite $\alpha \ge 0$, i.e. we don't need to force the penalty parameter $\nu_k \to 0$.

Next, we show that $\bar{x}$ is a Karush-Kuhn-Tucker (KKT) point, and is, therefore, optimal. For all $k \geq 1$, $\|\nabla f_{\mu_k}(x_k)\|_\infty = \|u_k\|_\infty \leq 1$. Therefore, there exists a vector $\bar{g} \in \mathbb{R}^n$ and a subsequence $\mathcal{K}_1 \subset \mathcal{K}$ such that

$$\lim_{k \in \mathcal{K}_1} u_k = \bar{g}. \tag{2.10}$$

Since $\lim_{k\in\mathcal{K}_1} x_k = \bar{x}$ and (2.10) holds, it follows that

$$\bar{g}(i) = \begin{cases} \text{sign}(\bar{x}(i)) & |\bar{x}(i)| \neq 0, \\ \in [-1, 1] & \bar{x}(i) = 0. \end{cases}$$

Therefore, $\bar{g} \in \partial\|x\|_1 \mid_{x=\bar{x}}$. Let $\theta_k = \frac{-w_k}{\lambda_k}$. Since the gradient $\nabla Q_k(x) = \lambda_k u_k + A^T w_k$ we have that $A^T\theta_k = u_k - \frac{1}{\lambda_k}\nabla Q_k(x_k)$. Since A^T has full column rank, $\theta_k = (AA^T)^{-1}A(u_k - \frac{1}{\lambda_k}\nabla Q_k(x_k))$. From Step 1 it follows that $\|\nabla Q_k(x_k)\|_2 \leq \tau_k$ and $\frac{\tau_k}{\lambda_k} \to 0$; therefore, $\lim_k \frac{\nabla Q_k(x_k)}{\lambda_k} = 0$. Therefore, (2.10) implies that $\bar{\theta} = \lim_{k\in\mathcal{K}_1} \theta_k$ exists and

$$\bar{g} = A^T\bar{\theta}. \tag{2.11}$$

From (2.9) and (2.11), it follows that $\bar{x}$ is a KKT point for the ℓ_1-minimization problem (1.1). Since $\|x\|_1$ is convex, the optimization problem (1.1) is a convex programming problem with equality constraints. Hence KKT conditions are sufficient for optimality and we can conclude that $\bar{x}$ is an optimal solution for (1.1). □

In compressive sensing exact recovery occurs only when $\min\{\|x\|_1 : Ax = b\}$ has a *unique* solution. The following Corollary establishes that SPA converges to this solution.

COROLLARY 2.2. *Suppose the ℓ_1-minimization problem* $\min\{\|x\|_1 : Ax = b\}$ *has a* unique *optimal solution. Let $\{x_k : k \in \mathbb{Z}_+\}$ denote the sequence of iterates generated by the Smoothed Penalty Algorithm (SPA) displayed in Figure 2.1 when*

(i) Smoothing parameter for $\|x\|_1$: $\mu_k \searrow 0$
(ii) Smoothing parameter for $P(x)$: $\nu_k \searrow \alpha \geq 0$
(iii) Penalty multiplier: $\lambda_k \searrow 0$
(iv) Approximate optimality parameter: $\tau_k \searrow 0$ such that $\frac{\tau_k}{\lambda_k} \to 0$.

Then $\lim_{k\to\infty} x_k = x^*$ where $x^* = \arg\min\{\|x\|_1 : Ax = b\}$.

Note that Theorem 2.1 and Corollary 2.2 will continue to hold even if we only ensure $\|\nabla Q_k(x_k)\|_2 \leq \tau_k$, provided the iterates $\{x_k\}_{k\in\mathbb{Z}_+}$ are eventually bounded. In Section 4 we use this fact to develop a modified version of SPA that has superior practical performance. Theorem 2.1 and Corollary 2.2 continue to hold when we penalize the infeasibility by the ℓ_1 or the ℓ_∞ norm. Therefore, the version of SPA that uses ℓ_1 or ℓ_∞ penalty also recovers the optimal solution.

We use the Nesterov optimal gradient algorithm [18] to compute the iterates x_k. To minimize a convex function g with a Lipschitz continuous gradient, in every iteration of the Nesterov optimal gradient algorithm we need to solve two problems of the form

$$\min_{x\in\mathbb{R}^n} \Big\{ c^T x + \frac{L}{\sigma} D(x, z) \Big\},$$

where c is a function of the gradients of the function g computed in all the previous iterates, L denotes the Lipschitz constant of the gradient of the function g, $D(x, z)$ denotes the Bregman function corresponding to a strongly convex *prox* function $h(y)$ with convexity parameter σ. We use $h(y) = \frac{1}{2}\|z\|^2$ with $D(x, z) = \frac{1}{2}\|x-z\|_2^2$ and $\sigma = 1$. Therefore, the Nesterov update reduces to $x = z - \frac{1}{L}c$. Consequently, the most expensive step in the Nesterov algorithm is computing the gradient $\nabla Q_k(x)$ which involves matrix multiplications of the form $A^T(Ax - b)$. In Section 4 we describe some algorithmic modification that improves the practical performance of SPA.

LEMMA 2.3. *Nesterov's optimal algorithm for simple sets [18] computes an x_k satisfying Step 1 in SPA in*

$$N_k = \Big\lceil \frac{2L_k(\|x^*\|_1 + \mu_k n/2)}{\tau_k} \Big\rceil \tag{2.12}$$

iterations, where $L_k = \frac{\lambda_k}{\mu_k} + \frac{\|A\|_2^2}{\nu_k}$ *denotes the Lipschitz constant for* ∇Q_k*, and* x^* *denotes the optimal solution of (1.1).*

Remark 2.2. *Nesterov optimal algorithm guarantees the bound (2.12) for* all *initial starting points for the k-th subproblem. We are not able to take advantage of the fact that the particular initial point* $x_{(k-1)}$ *for the k-th subproblem is close to an optimal solution for the k-th subproblem since* $Q_{k-1}(x) \approx Q_k(x)$ *for all x.*

Proof.

For any convex function $g : \mathbb{R}^n \to \mathbb{R}$ that has a Lipschitz continuous gradient with the Lipschitz constant L with respect to the ℓ_2 norm, we are guaranteed that l-th iterate x_l computed by Nesterov optimal algorithm satisfies [18, 19]

$$g(x_l) - g(x^*) \leq \frac{4Lh(x^*)}{\sigma(l+1)(l+2)},$$

where $x^* = \operatorname{argmin}_{x\in\Re^n}\{g(x)\}$ and $h(x)$ denotes the particular prox function with convexity parameter σ used in the Nesterov algorithm.

In the proof of Lemma 2.1 we showed that $x_k^* = \operatorname{argmin}_{x\in\mathbb{R}^n}\{Q_k(x)\}$ satisfies $\|x_k^*\|_1 \leq \|v\|_1 + \frac{\mu_k n}{2}$, for any vector v such that $Av = b$. In particular, $\|x_k^*\|_1 \leq \|x^*\|_1 + \frac{\mu_k n}{2}$.

Since $Q_k(x)$ is a convex function with a Lipschitz continuous gradient with Lipschitz constant $L_k = \frac{\lambda_k}{\mu_k} + \frac{\|A\|_2^2}{\nu_k}$ and we use the prox function $h(x) = \frac{1}{2}\|x\|_2^2 \leq \frac{1}{2}\|x\|_1^2$ with $\sigma = 1$, it follows that the l-th iterate x_l computed by Nesterov optimal algorithm applied to the problem $\min_{x\in\mathbb{R}^n}\{Q_k(x)\}$ satisfies $Q_k(x_l) - Q_k(x_k^*) \leq \frac{\tau_k^2}{2L_k}$ for all $l \geq \left\lceil \frac{2L_k(\|x^*\|_1 + \mu_k n/2)}{\tau_k} \right\rceil \triangleq N_k$. □

Next, we characterize the finite iteration performance of SPA. This analysis will lead to a convergence rate result in Theorem 2.5.

THEOREM 2.4. *Let* $\{x_k\}_{k\in\mathbb{Z}_+}$ *denote the sequence of iterates generated by Algorithm SPA and* $\sigma_{\min}(A)$ *denote the smallest non-zero singular value of A. Then, for all* $k \geq 1$*,*

$$\|x_k\|_1 - \|x^*\|_1 \leq \frac{\tau_k^2}{2\lambda_k L_k} + \frac{\mu_k n}{2}, \tag{2.13}$$

and for all k such that $\tau_k + \lambda_k\sqrt{n} < \sigma_{min}(A)$*,*

$$\begin{aligned} \|x_k\|_1 - \|x^*\|_1 &\geq -\frac{\mu_k n}{2} - \frac{\nu_k}{\lambda_k}, \\ \|Ax_k - b\|_2 &\leq \nu_k. \end{aligned} \tag{2.14}$$

Proof. Since the penalty function $P_\nu(x) \geq 0$, $f_\mu(x) \geq \|x\|_1 - \frac{\mu n}{2}$ for all $x \in \mathbb{R}^n$ and $\mu, \nu > 0$, it follows that

$$\begin{aligned} \lambda_k\|x_k\|_1 &\leq \lambda_k f_{\mu_k}(x_k) + \frac{\lambda_k\mu_k n}{2}, \\ &\leq Q_k(x_k) + \frac{\lambda_k\mu_k n}{2}, \\ &\leq Q_k(x_k^*) + \frac{\tau_k^2}{2L_k} + \frac{\lambda_k\mu_k n}{2}, \\ &\leq Q_k(x^*) + \frac{\tau_k^2}{2L_k} + \frac{\lambda_k\mu_k n}{2}, \\ &\leq \lambda_k\|x^*\|_1 + \frac{\tau_k^2}{2L_k} + \frac{\lambda_k\mu_k n}{2}, \end{aligned}$$

where the last inequality follows from the fact that $Ax^* = b$. Thus, (2.13) follows.

Next, we establish the bound on the infeasibility of the iterate x_k for k satisfying $\tau_k + \lambda_k\sqrt{n} < \sigma_{\min}(A)$. Recall that $Q_k(x_k) \leq Q_k(x_k^*) + \frac{\tau_k^2}{2L_k}$ implies that $\|\nabla Q_k(x_k)\|_2 \leq \tau_k$, and this, in turn, implies that

$$\|\nabla P_{\nu_k}(x_k)\|_2 = \|A^T w_k\|_2 \leq (\tau_k + \lambda_k\|\nabla f_{\mu_k}(x)\|_2) \leq \tau_k + \lambda_k\sqrt{n},$$

where the last inequality follows from the fact that $\|\nabla f_{\mu_k}(x_k)\|_\infty = \|u_k\|_\infty \leq 1$. Since A^T is assumed to have a full column rank, it follows that

$$\|w_k\|_2 \leq \frac{\|A^T w_k\|_2}{\sigma_{\min}(A)} \leq \frac{\tau_k + \lambda_k \sqrt{n}}{\sigma_{\min}(A)} < 1.$$

Since $\|w_k\| < 1$, (2.5) implies that $w_k = \frac{Ax_k - b}{\nu_k}$, and therefore, $\|Ax_k - b\|_2 = \nu_k \|w_k\|_2 \leq \nu_k$.

The final step in the proof is to establish the lower bound (2.14). Fix an iterate k such that $\tau_k + \lambda_k \sqrt{n} < \sigma_{\min}(A)$. We establish a lower bound for $\|x_k\|_1$ using the linear programming duality:

$$\begin{array}{lll} \text{minimize} & \|x\|_1, & = \\ \text{subject to} & Ax = b. & \end{array} \qquad \begin{array}{lll} \text{maximize} & b^T w, \\ \text{subject to} & \|A^T w\|_\infty \leq 1. \end{array}$$

Let w^* denote the optimal dual solution. Then $\|x^*\|_1 = b^T w^*$ and $\|A^T w^*\|_2 \leq \sqrt{n}$ and $\|w^*\|_2 \leq \frac{\sqrt{n}}{\sigma_{\min}(A)}$. Linear programming duality also implies that

$$\begin{array}{lll} \underset{x \in \Re^n}{\text{minimize}} & \lambda \|x\|_1 + \|Ax - b\|_2 & = \\ & & \end{array} \qquad \begin{array}{lll} \text{maximize} & \lambda b^T w, \\ \text{subject to} & \|A^T w\|_\infty \leq 1, \\ & \|w\|_2 \leq \lambda^{-1}. \end{array} \tag{2.15}$$

Since $\|w^*\|_2 \leq \frac{\sqrt{n}}{\sigma_{\min}(A)}$, it follows that w^* is feasible for the dual program in (2.15) whenever $\lambda \sqrt{n} \leq \sigma_{\min}(A)$. Since $\tau_k + \lambda_k \sqrt{n} \leq \sigma_{\min}(A)$, weak duality implies that $\min_x \{\lambda_k \|x\|_1 + \|Ax - b\|_2\} \geq \lambda_k b^T w^*$. Next, we relate the exact penalty objective function to the smoothed penalty function $Q_k(x)$.

$$\begin{aligned} \min_x Q_k(x) &\geq \min_x \left\{\lambda_k \|x\|_1 + \|Ax - b\|_2\right\} - \frac{\lambda_k \mu_k n}{2} - \frac{\nu_k}{2}, \\ &\geq \lambda_k b^T w^* - \frac{\lambda_k \mu_k n}{2} - \frac{\nu_k}{2}, \\ &= \lambda_k \|x^*\|_1 - \frac{\lambda_k \mu_k n}{2} - \frac{\nu_k}{2}. \end{aligned} \tag{2.16}$$

Also,

$$\begin{aligned} Q_k(x_k) &= \lambda_k f_{\mu_k}(x_k) + P_{\nu_k}(x_k), \\ &\leq \lambda_k \|x_k\|_1 + \frac{\|Ax_k - b\|_2^2}{2\nu_k}, \qquad (2.17) \\ &\leq \lambda_k \|x_k\|_1 + \frac{\nu_k}{2}, \qquad (2.18) \end{aligned}$$

where (2.17) and (2.18) follow from the fact that $\frac{\tau_k + \sqrt{n}\lambda_k}{\sigma_{\min}(A)} < 1$ implies that $w_k = \frac{Ax_k - b}{\nu_k}$ and $\|Ax_k - b\|_2 \leq \nu_k$. The lower bound follows from (2.16) and (2.18). □

Theorem 2.1 and 2.4 reveal the relationship between four multipliers $(\mu_k, \nu_k, \tau_k, \lambda_k)$ used in SPA. For the convergence proof for SPA in Theorem 2.1 we required that $\tau_k/\lambda_k \to 0$ as $k \to \infty$. The upper bound (2.13) in Theorem 2.4 implies that all the terms in the multiplier sequence $\{\mu_k\}_{k \in \mathbb{Z}_+}$ should be scaled as $\frac{1}{n}$ as a function of the the target signal dimension n to ensure that the error is $\mathcal{O}(1)$. The lower bound (2.14) in Theorem 2.4 implies that $\nu_k/\lambda_k \to 0$ as $k \to \infty$. Since lower bound holds only if $\tau_k + \lambda_k \sqrt{n} \leq \sigma_{\min}(A)$; therefore, the terms in the sequence $\{\lambda_k\}_{k \in \mathbb{Z}_+}$ should scale as $\frac{\sigma_{\min}(A)}{\sqrt{n}}$. We show in Appendix A.1 that in compressive sensing applications with Discrete Cosine Transform (DCT) (resp. Discrete Fourier Transform (DFT) measurements) $\sigma_{\min}(A) = 1$ (resp. $\sigma_{\min}(A) = \frac{1}{2}$); thus, in this setting $\lambda_k = \mathcal{O}(\frac{1}{\sqrt{n}})$ for all $k \in \mathbb{Z}_+$. In the result below we fix $(\lambda_0, \mu_0, \nu_0) > 0$ (independent of the problem dimension n) and then use these scaling rules to construct a sequence of multiplier that ensure a very good convergence rate for SPA.

THEOREM 2.5. *Fix $0 < \delta < 1$, $\frac{1}{4} \leq \alpha \leq \frac{3}{4}$ and strictly positive parameters $(\lambda_0, \tau_0, u_0, \nu_0)$. Select the sequence multipliers as follows:*

$$\mu_k = \begin{cases} \frac{\mu_0}{n}, & k = 1, \\ \alpha^{(1+\delta)}\mu_{k-1} & k > 1, \end{cases} \qquad \nu_k = \begin{cases} \frac{\nu_0}{\sqrt{n}}, & k = 1, \\ \alpha^{(1+\delta)}\nu_{k-1} & k > 1, \end{cases}$$
$$\tau_k = \begin{cases} \tau_0, & k = 1, \\ \alpha^{\frac{1}{2}(1+\delta)}\tau_{k-1}, & k > 1, \end{cases} \qquad \lambda_k = \begin{cases} \frac{\lambda_0}{\sqrt{n}} & k = 1, \\ \alpha^{\delta}\lambda_{k-1} & k > 1. \end{cases} \tag{2.19}$$

Let $\{\bar{x}_k : k \geq 1\}$ denote sequence of iterates computed by the SPA algorithm corresponding to this set of multipliers.

Let

$$\epsilon_0 = \frac{\max\{\mu_0, \frac{2\nu_0}{\lambda_0}, \frac{2\nu_0\tau_0^2}{\lambda_0\|A\|_2^2}, \frac{\nu_0}{\sqrt{n}}\}\sigma_{min}(A)^{\frac{2+\delta}{\delta}}\alpha^{2+\delta}}{\big(2\max\{\lambda_0, \tau_0\}\big)^{\frac{2+\delta}{\delta}}}.$$

Then for $\epsilon < \epsilon_0$ there exists an $k_\epsilon \leq \mathcal{O}\big(\frac{\ln(\frac{1}{\epsilon})}{\ln(\frac{1}{\alpha})}\big)$ such that

$$\|A\bar{x}_{k_\epsilon} - b\|_2 \leq \epsilon, \qquad |\|\bar{x}_{k_\epsilon}\|_1 - \|x^*\|_1| \leq \epsilon,$$

and the running time to compute such an iterate is $\mathcal{O}\Big(\big(\|x^*\|_1 + \frac{\mu_0}{2}\big)n^{\frac{3}{2}}\ln(n)\epsilon^{-\frac{3}{2}(1+\delta)}\Big)$, where x^* denotes the optimal solution of (1.1).

Remark 2.3. *Note that we* fix *a choice of parameters and compute the sequence of iterates $\{\bar{x}_k : k \geq 1\}$ a priori. Theorem 2.5 establishes that for each $\epsilon < \epsilon_0$ we can find an iterate k_ϵ that is ϵ-feasible and ϵ-optimal and the running time to compute such an iterate grows as $\epsilon^{-\frac{3}{2}(1+\delta)}$. In Section 4 we show how to efficiently compute a bound on $\|x^*\|_1$.*

Proof. For the update scheme in (2.19),

$$\tau_k + \lambda_k\sqrt{n} \leq \tau_0\alpha^{\frac{1}{2}(1+\delta)(k-1)} + \lambda_0\alpha^{\delta(k-1)} \leq 2\max\{\lambda_0, \tau_0\}\alpha^{\delta(k-1)}.$$

Thus, $\tau_k + \lambda_k\sqrt{n} \leq \sigma_{\min}(A)$ for all

$$k \geq K_1 + 1 \triangleq \frac{\ln(\frac{2\max\{\lambda_0,\tau_0\}}{\sigma_{\min}(A)})}{\delta\ln(\frac{1}{\alpha})} + 1.$$

Then Theorem 2.4 and the fact that $L_k \geq \frac{\|A\|_2^2}{\nu_k}$ implies that for all $k > K_1 + 1$,

$$\begin{aligned} |\|x_k\|_1 - \|x^*\|_1| &\leq \frac{\mu_k n}{2} + \max\left\{\frac{\nu_k}{\lambda_k}, \frac{\tau_k^2}{2\lambda_k L_k}\right\}, \\ &\leq \frac{\mu_0}{2}\cdot\alpha^{(1+\delta)(k-1)} + \max\left\{\frac{\nu_0}{\lambda_0}\cdot\alpha^{(k-1)}, \frac{\nu_0\tau_0^2}{\lambda_0\|A\|_2^2}\cdot\alpha^{(2+\delta)(k-1)}\right\}, \end{aligned}$$

Thus,

$$|\|x_k\|_1 - \|x^*\|_1| \leq \epsilon,$$

for all

$$k \geq K_2 + 1 \triangleq \max\left\{K_1, \frac{\ln(\frac{\mu_0}{\epsilon})}{(1+\delta)\ln(\frac{1}{\alpha})}, \frac{\ln(\frac{2\nu_0}{\lambda_0\epsilon})}{\ln(\frac{1}{\alpha})}, \frac{\ln(\frac{2\nu_0\tau_0^2}{\lambda_0\|A\|_2^2\epsilon})}{(2+\delta)\ln(\frac{1}{\alpha})}\right\} + 1. \tag{2.20}$$

From Theorem 2.4 we also have that for $k > K_1 + 1$,

$$\|Ax - b\|_2 \leq \nu_k = \frac{\nu_0}{\sqrt{n}}\alpha^{(1+\delta)(k-1)}.$$

Consequently, $\|Ax-b\|_2 \leq \epsilon$ for all

$$k \geq K_3 + 1 \triangleq \left\{ K_1, \frac{\ln(\frac{\nu_0}{\epsilon\sqrt{n}})}{(1+\delta)\ln(\frac{1}{\alpha})} \right\} + 1. \tag{2.21}$$

From (2.20) and (2.21), it follows that that for all $\epsilon < \frac{B\sigma_{min}(A)^{\frac{2+\delta}{\delta}}\alpha^{2+\delta}}{\left(2\max\{\lambda_0,\tau_0\}\right)^{\frac{2+\delta}{\delta}}}$, the number of iterations $K = \max\{K_2, K_3\} + 1$ required to compute an ϵ-feasible and ϵ-optimal point satisfies

$$K \leq \frac{\ln(\frac{B}{\epsilon})}{\ln(\frac{1}{\alpha})}, \tag{2.22}$$

where

$$B = \max\left\{ \mu_0, \frac{2\nu_0}{\lambda_0}, \frac{2\nu_0\tau_0^2}{\lambda_0\|A\|_2^2}, \frac{\nu_0}{\sqrt{n}} \right\}.$$

Lemma 2.3 implies that K iterations of Algorithm SPA require a total of

$$N_{in} \leq \sum_{k=1}^{K} \lceil \frac{2L_k(\|x^*\|_1 + \mu_k n/2)}{\tau_k} \rceil$$

steps of the Nesterov optimal algorithm for simple sets. Since $\mu_k \leq \frac{\mu_0}{2}$ for all $k \geq 1$, and

$$\begin{aligned}\sum_{k=1}^{K} \frac{L_k}{\tau_k} &= \sum_{k=1}^{K} \Big(\frac{\lambda_k}{\tau_k\mu_k} + \frac{\|A\|_2^2}{\tau_k\nu_k} \Big), \\ &= \sum_{k=0}^{K-1} \left(\Big(\frac{\lambda_0\sqrt{n}}{\tau_0\mu_0} \Big) \cdot \alpha^{-\frac{1}{2}(3+\delta)k} + \Big(\frac{\|A\|_2^2\sqrt{n}}{\tau_0\nu_0} \Big) \alpha^{-\frac{3}{2}(1+\delta)k} \right).\end{aligned}$$

Thus,

$$N_{in} = \mathcal{O}\left(\sqrt{n}\Big(\|x^*\|_1 + \frac{\mu_0}{2} \Big) \max\left\{ \frac{\lambda_0}{\tau_0\mu_0}, \frac{\|A\|_2^2}{\tau_0\nu_0} \right\} \alpha^{-\frac{3}{2}(1+\delta)K} \right).$$

From (2.22) it follows that for all $\epsilon < \frac{B\sigma_{min}(A)^{\frac{2+\delta}{\delta}}\alpha^{2+\delta}}{\left(2\max\{\lambda_0,\tau_0\}\right)^{\frac{2+\delta}{\delta}}}$,

$$N_{in} = \mathcal{O}\left(\sqrt{n}\Big(\|x^*\|_1 + \frac{\mu_0}{2} \Big) \max\left\{ \frac{\lambda_0}{\tau_0\mu_0}, \frac{\|A\|_2^2}{\tau_0\nu_0} \right\} \cdot B^{\frac{3}{2}(1+\delta)} \cdot \epsilon^{-\frac{3}{2}(1+\delta)} \right).$$

Since each inner iteration requires $\mathcal{O}(n\ln(n))$ operations, it follows that algorithm SPA computes an ϵ-infeasible and ϵ-optimal solution in $\mathcal{O}\Big(\Big(\|x^*\|_1 + \frac{\mu_0}{2} \Big) n^{\frac{3}{2}} \ln(n) \epsilon^{-\frac{3}{2}(1+\delta)} \Big)$ operations. □

Theorem 2.4 and Theorem 2.5 require that the iterates x_k satisfy the stronger condition $Q_k(x_k) \leq \min_x\{Q_k(x)\} + \frac{\tau_k^2}{2L_k}$; it is not sufficient to only ensure the weaker condition $\|\nabla Q_k(x_k)\|_2 \leq \tau_k$. Theorem 2.5 establishes SPA computes an ϵ-optimal solution in $\mathcal{O}(\epsilon^{-\frac{3}{2}})$ operations. On the other hand Nesterov algorithm requires only $\mathcal{O}(\epsilon^{-1})$ operations to compute an ϵ-optimal solution; however, in order to achieve this rate the Nesterov updates must be feasible, i.e. the iterates must satisfy $Ax = b$. This requires a projection; thus, the complexity of a feasible Nesterov update is $\mathcal{O}(mn)$ operations (see the last paragraph of Appendix A.2). SPA does not require projections and computes a Nesterov update in $\mathcal{O}(n)$ operations; however, since the SPA updates are infeasible, the number of Nesterov updates increase by a factor $\epsilon^{\frac{1}{2}}$. Comparing $\mathcal{O}(mn\epsilon^{-1})$ operations for Nesterov-type algorithms with feasible updates with $\mathcal{O}(n\epsilon^{-\frac{3}{2}})$ operations for SPA, it follows that the penalty approach is superior when the required accuracy ϵ is not too small $\epsilon \geq \mathcal{O}(\frac{1}{m^2})$. Since, in practice, $m = \mathcal{O}(n)$, and problem dimension n is large, the lower bound on ϵ is quite small. In the next section, we show that SPA is superior to a feasible Nesterov-type algorithms for noisy recovery for all $\epsilon \geq \mathcal{O}(\frac{1}{n^4})$, i.e. for almost all practical instances.

3. Extensions of the SPA to noisy recovery. A simple modification of SPA solves the noisy signal recovery problem

$$\begin{array}{ll} \text{minimize} & \|x\|_1, \\ \text{subject to} & \|Ax-b\|_2 \leq \epsilon. \end{array} \tag{3.1}$$

To solve this problem we use the exact penalty function

$$\phi(x) = \max\Big\{0, \|Ax-b\|_2 - \epsilon\Big\} = \max_{0\leq t\leq 1}\Big\{t\big(\|Ax-b\|_2-\epsilon\big)\Big\} = \max_{\{(w,t):\|w\|_2\leq t, t\in[0,1]\}}\Big\{w^T(Ax-b)_2 - t\epsilon\Big\}.$$

Next, we smooth this function to get the smoothed penalty function

$$\phi_\nu(x) = \max_{\{(w,t):\|w\|_2\leq t, t\in[0,1]\}}\Big\{w^T(Ax-b)_2 - t\epsilon - \frac{\nu}{2}\Big(t^2 + t\|\frac{w}{t}\|_2^2\Big)\Big\}$$

Using results in Hoda et al [11] one can show that the function $h(t,w) = \frac{1}{2}\big(t^2 + t\|\frac{1}{t}w\|_2^2\big)$ is strongly convex over the truncated cone $\{(t,w) : \|w\|_2 \leq t, t\in[0,1]\}$, and consequently, $\phi_\nu(x)$ is a convex function with a Lipschitz continuous gradient. Given the structure of $h(t,w)$, one can rewrite $P_\nu(x)$ as follows:

$$\phi_\nu(x) = \max_{t\in[0,1]}\Big\{t\big(\max_{\|\hat{w}\|_2\leq 1}\big\{\hat{w}^T(Ax-b) - \frac{\nu}{2}\|\hat{w}\|_2^2\big\}\big) - t\epsilon - \frac{\nu}{2}t^2\Big\}.$$

Recall that the smoothed ℓ_2-penalty function $P_\nu(x) = \max_{\|\hat{w}\|_2\leq 1}\big\{\hat{w}^T(Ax-b) - \frac{1}{2}\|\hat{w}\|_2^2\big\}$ with the optimal $\hat{w}_x$ given by (2.5). Thus,

$$\phi_\nu(x) = \max_{t\in[0,1]}\Big\{t\big(P_\nu(x)-\epsilon\big) - \frac{\nu}{2}t^2\Big\},$$

with the optimal $t_x = \min\big\{\frac{(P_\nu(x)-\epsilon)^+}{\nu}, 1\big\}$. Thus, the gradient $\nabla\phi_\nu(x) = t_x\hat{w}_x$, where $\hat{w}_x$ is given by (2.5).

Theorem 2.1, Corollary 2.2, Theorem 2.4 and Theorem 2.5 all remain valid for SPA applied to the penalized objective function $\lambda f_\mu(x) + \phi_\nu(x)$ (see [2]). Thus, SPA efficiently computes a solution for the noisy recovery problem (3.1). Note that unlike NESTA [20], SPA does not require A^TA to be orthogonal projector, i.e. A does not need to have orthonormal rows. For general A, the complexity of computing a feasible Nesterov update is $\mathcal{O}(n^3)$; thus, comparing $\mathcal{O}(n^3\epsilon^{-1})$ operations for Nesterov-type algorithms with feasible updates with $\mathcal{O}(n\epsilon^{-\frac{3}{2}})$ operations for SPA, it follows that the complexity bound for SPA is superior to Nesterov-type algorithms that compute feasible iterates as long as $\epsilon \geq \mathcal{O}(\frac{1}{n^4})$.

The analysis in this section can be extended to solve noisy recovery problems $\min\{\|x\|_1 : \|Ax-b\|_1 \leq \epsilon\}$ and $\min\{\|x\|_1 : \|Ax-b\|_\infty \leq \epsilon\}$ [2]. These formulations are interesting when the measurement noise has a Laplacian or Extreme Value distributions.

4. Implementation details of Algorithm SPA. In this section we describe some algorithmic modifications that significantly improve the practical performance of SPA. The SPA with all these modifications is shown in Figure 4.1.

4.1. Bounds on iterates and modified Nesterov updates. Recall that Theorem 2.1 and Corollary 2.2 continue to hold even when the Nesterov update steps that compute the iterates are terminated when $\|\nabla Q_k(x_k)\| \leq \tau_k$, provided we can ensure that the iterates are uniformly bounded. We found that in practice terminating the Nesterov updates using the gradient condition was significantly faster and incorporated this in the practical version of SPA.

Let $q(v) = \text{argmin}\{\|x-v\|_2 : Ax = b\}$. Computing $q(v)$ requires a projection onto the affine space $Ax = b$. We show in Appendix A.2 that by pre-computing the eigenvalue decomposition of the matrix $AA^T \in \mathbb{R}^{m\times m}$ one can reduce the running time of this projection to $\mathcal{O}(m(m+n)+k_f(m,n))$ where $k_f(m,n)$ denotes the running time for compute Ax for a general x.

Using an analysis very similar to that in the proof of Theorem 2.1, we can show that $\|x_k^*\|_1 \leq f_{\mu_k}(q(v)) + (\mu_k n)/2$ for all $v \in \Re^n$. Let $\mathcal{V} \subset \mathbb{R}^n$ denote any finite collection of vectors. Then $\min_{v\in\mathcal{V}}\{f_{\mu_k}(q(v))\} + (\mu_k n)/2$

SMOOTHED PENALTY ALGORITHM $\left(c_\tau^{(0)}, c_\tau^{(1)}, c_\mu, c_\nu, c_\eta, c_\lambda, c_\sigma\right)$

$q(0) \leftarrow \arg\min\{\|x\|_2 |\ Ax = b\}, \quad x_0 \leftarrow q_0, \quad k \leftarrow 0$

while $\big((k == 0)$ **or** $(\|x_k - x_{k-1}\|_\infty > \gamma)\big)$

do

$k \leftarrow k + 1$

$Q_k(.) \leftarrow \lambda_k f_{\mu_k}(.) + P_{\nu_k}(.)$

/* Update parameter values */

if $(k == 1)$

then $\lambda_1 \leftarrow \frac{2}{\sqrt{n}}$, $\tau_1 \leftarrow c_\tau^{(0)} \|\nabla Q_1(x_0)\|_2$, $\eta_1 = \lambda_1 \|x_0\|_1$,

$\mu_1 = \frac{c_\eta \eta_1}{(\sqrt{n}\lambda_1 + 1)\sqrt{n}}$, $\nu_1 = \frac{c_\eta \eta_1}{(\sqrt{n}\lambda_1 + 1)}$, $\beta = f_{\mu_1}(x_0)$

else $\lambda_k = c_\lambda \lambda_{k-1}$, $\tau_k = \min\left\{c_\tau^{(1)} \tau_{k-1}, c_\tau^{(0)} \|\nabla Q_k(x_{k-1})\|_2\right\}$

$\mu_k \leftarrow \min\left\{c_\mu \mu_{k-1}, \frac{c_\eta \eta_{k-1}}{(\sqrt{n}\lambda_k + 1)\sqrt{n}}\right\}$, $\nu_k \leftarrow \min\left\{c_\nu \nu_{k-1}, \frac{c_\eta \eta_{k-1}}{\sqrt{n}\lambda_k + 1}\right\}$

$L_k \leftarrow \frac{\lambda_k}{\mu_k} + \frac{\|A\|_2^2}{\nu_k}$

/* Start Nesterov Update */

$l \leftarrow 0, \quad x_{k,l} \leftarrow x_{(k-1)}, \quad \sigma_k \leftarrow \beta + \frac{\mu_k n}{2}$

compute $\nabla Q_k(x_{k,l})$

while $\left(\|\nabla Q_k(x_{k,l})\|_2 > \tau_k\right)$

do

1 $y_{k,l} \leftarrow \left(\frac{2}{l+2}\right) \Pi_{\ell_1}\left(\sigma_k, \frac{(k+2)(x_{k,l} - \nabla Q_k(x_{k,l})/L_k) - k y_{k,l-1}}{2}\right) + \left(\frac{l}{l+2}\right) y_{k,l-1}$

2 $z_{k,l} \leftarrow \Pi_{\ell_1}\left(\sigma_k, x_{k,0} - \frac{\sum_{i=0}^{l} (\frac{i+1}{2}) \nabla Q_k(x_{k,i})}{L_k}\right)$

$x_{k,l+1} \leftarrow \left(\frac{2}{l+3}\right) z_{k,l} + \left(\frac{l+1}{l+3}\right) y_{k,l}$

$l \leftarrow l + 1$

if $\big((l \mod L) == 0\big)$

then $\pi \leftarrow \text{argmin}\{\|x - x_{k,l}\|_2 : Ax = b\}$

$\beta \leftarrow \min\{\beta, f_{\mu_k}(\pi)\}$

$\sigma_k \leftarrow \beta + \frac{\mu_k n}{2}$

compute $\nabla Q_k(x_{k,l})$

/* End Nesterov Updates */

$x_k \leftarrow x_{k,l}$

$\eta_k \leftarrow$ approximately duality gap from Nesterov algorithm

$q_k \leftarrow \text{argmin}\{\|x - x_k\|_2 : Ax = b\}$

$\beta \leftarrow \min\{\beta, f_{\mu_k}(q_k)\}$

return x_k

FIG. 4.1. *Implemented version of Smoothed Penalty Algorithm (SPA)*

is also a valid, and possibly improved, bound for $\|x_k^*\|_1$. Consequently, we can restrict the iterate x_k to the set $\left\{x : \|x\|_1 \leq \min_{v \in \mathcal{V}}\{f_{\mu_k}(q(v))\} + (\mu_k n)/2\right\}$.

In our implementation of SPA, we begin with $\mathcal{V} = \{0\}$ and we insert x_k iterates to $\mathcal{V}$ as they become available and compute $q(x_k)$ in every SPA iteration (and not every Nesterov update step), so that we can bound the updates to ensure that the Nesterov updates terminate quickly. Moreover, we also compute a new bound whenever the number of Nesterov updates for solving a given subproblem exceeds $L = 50$.

In our numerical experiments, we compute at most $15 - 25$ projections in order to solve the problem to optimality; therefore, the cost of the projection does not add much to the overall cost of the algorithm. Note that we only need *one* projection to ensure that the SPA iterates are uniformly bounded, and therefore, converge to the optimal solution; the additional projections are added only to improve practical performance.

We compute the SPA iterates $\{x_k : k \geq 1\}$ using a slightly modified version of Nesterov's optimal algorithm for simple sets [18, 19]. We solve the k-th subproblem

$$\begin{array}{ll} \text{minimize} & Q_k(x), \\ \text{subject to} & \|x\|_1 \leq \sigma_k, \end{array}$$

by iteratively computing three sets of iterates$\{(x_{kl}, y_{kl}, z_{kl}) : l \geq 0\}$:

1. $y_{k,l}$ iterate in Step 1 of Figure 4.1 is computed using $x_{k,l}$, $y_{k,l-1}$ and the gradient $\nabla Q_k(x_{k,l})$:

$$\begin{aligned} y_{k,l} &= \text{argmin}\{\frac{L_k}{2}\|y - x_{k,l}\|_2^2 + \nabla Q_k(x_{k,l})^T y : \ y \in \left(\frac{2}{l+2}\right) S_k + \left(\frac{l}{l+2}\right) y_{k,l-1}\}, \\ &= \left(\frac{2}{l+2}\right) \Pi_{\ell_1}\left(\sigma_k, \frac{(k+2)(x_{k,l} - \nabla Q_k(x_{k,l})/L_k) - k y_{k,l-1}}{2}\right) + \left(\frac{l}{l+2}\right) y_{k,l-1}, \end{aligned}$$

   where $S_k = \{y \in \Re^n : \ \|y\|_1 \leq \sigma_k\}$ and the projection $\Pi_{\ell_1}(\sigma, \hat{y}) = \text{argmin}\{\|y - \hat{y}\|_2^2 : \|y\|_1 \leq \sigma\}$. In Appendix A.3 we show that the projection Π_{ℓ_1} can be computed with a $O(n \ln(n))$ worst case complexity and $\mathcal{O}(n)$ randomized complexity. This update scheme is *not* the standard Nesterov y-update [19]; however, one can show that this is a valid, and possibly an improved, update using the last paragraph of Lemma 1 in [19].
2. The $z_{k,l}$ iterate in Step 2 is computed using the initial point $x_{k,0}$ and the gradients $\nabla Q_k(x_{k,i})$ for all the iterates $i \leq l$:

$$\begin{aligned} z_{k,l} &= \text{argmin}\left\{\sum_{i=0}^{l} \Big(\frac{i+1}{2}\Big)\nabla Q_k(x_{k,i})^T z + \frac{L}{2}\|z - x_{k,0}\|_2^2 : \|z\|_1 \leq \sigma_k\right\}, \\ &= \Pi_{\ell_1}\left(\sigma_k, x_{k,0} - \frac{1}{L}\sum_{i=0}^{n} \big(\frac{i+1}{2}\big)\nabla Q_k(x_{k,i})\right). \end{aligned}$$

   Note that in the k-th sub-problem, we use the *prox* function $h(x) = \frac{1}{2}\|x - x_{k,0}\|_2^2$ to compute the iterates $\{z_{k,l}\}$. A bound of the form (2.12) is valid for this prox function if the SPA iterates are uniformly bounded.
3. The $\{x_{k,l} : l \geq 0\}$ iterates are computed by initializing $x_{k,0} = x_{k-1}$, the previous SPA iterate, and setting $x_{k,l} = \Big(\frac{2}{l+3}\Big) z_{k,l-1} + \Big(\frac{l+1}{l+3}\Big) y_{k,l-1}$, for all $l \geq 1$.

Note that we terminate the Nesterov updates when $\|\nabla Q(x_k)\|_2 \leq \tau_k$. This ensures that the conclusions of Theorem 2.1 and Corollary 2.2 remain valid; however, the finite iteration results in Theorem 2.4 and Theorem 2.5 may not be valid.

4.2. Stopping criterion for SPA. We terminate when the ℓ_∞ difference between successive iterates are below a threshold γ, i.e. $\|x_k - x_{k-1}\|_\infty \leq \gamma$. In our numerical experiments we set γ by experimenting with a small instance of the problem.

4.3. Multiplier selection. The approximate optimality parameter τ_k is set as follows:

$$\begin{array}{rcl} \tau_1 & \leftarrow & c_\tau^{(0)} \|\nabla Q_1(x_0)\|_2, \\ \tau_{k+1} & \leftarrow & \min\left\{c_\tau^{(1)}\tau_k, c_\tau^{(0)}\|\nabla Q_{k+1}(x_k)\|_2\right\}, \quad \text{for all } k \geq 1. \end{array}$$

Guided by the scaling result implicit in Theorem 2.5 we set $\lambda_1 = \frac{2}{\sqrt{n}}$, and then set $\lambda_k = c_\lambda \lambda_{k-1}$ for all $k \geq 2$.

In the k-th iteration of SPA, we solve a smoothed version of the penalized optimization problem

$$\begin{array}{ll} \text{minimize} & \lambda_k\|x\|_1 + \|Ax - b\|_2, \\ \text{subject to} & \|x\|_1 \leq \sigma_k \end{array} \tag{4.1}$$

The dual of this optimization problem is given by

$$\begin{array}{ll} \text{maximize} & -b^T w - \sigma_k\|A^T w + \lambda_k u\|_\infty, \\ \text{subject to} & \|u\|_\infty \leq 1, \\ & \|w\|_2 \leq 1. \end{array} \tag{4.2}$$

Noting that x_0 and $(u_0, w_0) = (0, 0)$ are, respectively, primal and dual feasible, we initialize the duality gap $\eta_0 = \lambda_1 \|x_0\|_1$. Nesterov's non-smooth optimization algorithm [19] also returns approximately optimal dual variables. Let η_k denote the duality gap between the k-th primal iterate x_k and the dual iterates (u_k, w_k) returned by the Nesterov algorithm applied to the k-th subproblem. We set $\delta_{k+1} = c_\eta \eta_k$ as the target approximation error for the next subproblem. The Nesterov non-smooth optimization algorithm then dictates that the smoothing parameter (μ, ν) should be set to $\mu = \frac{\delta_{k+1}}{(\sqrt{n}\lambda_{k+1}+1)\sqrt{n}}$ and $\nu = \frac{\delta_{k+1}}{\sqrt{n}\lambda_{k+1}+1}$ in order to minimize the number of Nesterov updates required to compute a δ_{k+1}-optimal solution. Since we require that μ_k and ν_k be monotonically decreasing we modify this parameter update as follows:

$$\begin{aligned} \mu_k &\leftarrow \min\left\{c_\mu \mu_{k-1}, \tfrac{\delta_k}{(\sqrt{n}\lambda_k+1)\sqrt{n}}\right\}, \quad k \geq 1, \quad \mu_0 = \infty, \\ \nu_k &\leftarrow \min\left\{c_\nu \nu_{k-1}, \tfrac{\delta_k}{\sqrt{n}\lambda_k+1}\right\}, \quad k \geq 1, \quad \nu_0 = \infty. \end{aligned}$$

We "tune" the constants $(c_\tau^{(0)}, c_\tau^{(1)}, c_\mu, c_\nu, c_\eta, c_\lambda, c_\sigma)$ on the smallest $n = 64 \times 64$ problem and then used the values for all the other problems.

Since the parameter sequence $\{(\lambda_k, \tau_k, \mu_k, \nu_k) : k \geq 1\}$ follow the scaling in Theorem 2.5, we should expect that the optimal choice of initial multipliers $(\lambda_0, \mu_0, \nu_0)$ should be independent of n for a given measurement ratio m/n and sparsity ratio s/n. In our numerical experiments we found that this to approximately true. We exploit this fact by tuning the parameters $(\lambda_0, \mu_0, \nu_0)$ on the smallest problem (with $n = 64 \times 64$) and then use these parameters for all larger problems.

5. Numerical experiments. We conducted two sets of numerical experiments with SPA. The goal in the first set of experiments was to investigate how the complexity of SPA grows with the problem dimension. The second set of experiments compares the performance of SPA with another Nesterov-type algorithm NESTA [20] and a fixed point continuation algorithm FPC [9, 10].

5.1. Experimental setup. We tested SPA on randomly generated target signals. The target signal $x^* \in \Re^n$ was chosen to be s-sparse, i.e. exactly s out of n components were nonzero. Following the experimental setup in a recent paper of Becker, et al. [20] we set

$$x^*(i) = \mathbf{1}(i \in \Lambda)\ \eta_1(i) 10^{5\eta_2(i)} \tag{5.1}$$

where

(i) the set Λ was constructed by randomly selecting s indices from the set $\{1, \ldots, n\}$,
(ii) $\eta_1(i)$, $i \in \Lambda$, were independently, and identically distributed Bernoulli random variables taking values $+1$ or -1 with equal probability,
(iii) $\eta_2(i)$, $i \in \Lambda$, were independently, and identically distributed uniform$[0, 1]$ random variables.

The signals x^* were created in this manner have a dynamic range of 100dB.

The measurement matrix A and the measurement vector b were constructed as follows. We randomly selected $m = \frac{n}{4}$ frequencies from the set $\{0, \ldots, n\}$. Let $A \in \Re^{m \times n}$ denote a $m \times n$ partial Discrete Cosine matrix constructed from these randomly selected frequencies and $b = Ax^*$ denote the Discrete Cosine transform of the signal x^* evaluated at the chosen frequencies.

We found that for a fixed measurement ratio m/n, sparsity ratio s/n, and the accuracy tolerance γ, the total number of Nesterov updates is effectively independent of the dimension n of the target signal. In our experiment we exploit this empirical result by first tuning the constants controlling the parameter updates for a smallest sized problem and subsequently using these fixed parameters for solving all larger problems. In our numerical experiments the constants controlling the parameter update were set as follows:

$$c_\tau^{(0)} = 0.2, \quad c_\tau^{(1)} = 0.855, \quad c_\mu = 0.1, \quad c_\nu = 0.1, \quad c_\eta = 0.8, \quad c_\lambda = 0.9, \quad c_\sigma = 0.9. \tag{5.2}$$

5.2. Algorithm scaling results. We tested the algorithm for s-sparse signals with

(i) three different sizes: small $n = 64 \times 64$, medium $n = 256 \times 256$, and large $n = 512 \times 512$,
(ii) two sparsity levels: high $s = \lceil n/400 \rceil$, and low $s = \lceil n/40 \rceil$.

Sparsity	ϵ	Table
$s = n/400$	1	Table 5.3
$s = n/400$	0.1	Table 5.5
$s = n/400$	0.01	Table 5.7
$s = n/40$	1	Table 5.2
$s = n/40$	0.1	Table 5.4
$s = n/40$	0.01	Table 5.6

TABLE 5.1
Summary of numerical experiments

In order to assess the convergence properties of the SPA we replaced the stopping criterion $\|x_k - x_{k-1}\|_\infty \leq \epsilon$ with

$$\|x_k - x^*\|_\infty \leq \gamma. \tag{5.3}$$

We report results for $\gamma = 1$, 10^{-1} and 10^{-2}. The signal model in (5.1) and the stopping criterion implies that the algorithm produces x_k with $5 + \log_{10}(1/\gamma)$ digits of accuracy. Note that the stopping criterion (5.3) is only used to test the convergence properties the algorithm in this simulation study.

The Table 5.1 summarizes the sparsity conditions and the parameter settings that were investigated in the numerical experiments. The column marked `Table` lists the table where we display the results corresponding to the parameter setting of the particular row, e.g. the results for $s = n/40$ and $\gamma = 0.1$ are displayed in Table 5.4. In Tables 5.2– 5.7, the row labeled `Nesterov Updates` lists the total number of Nesterov update iterations during the course of SPA, the row labeled `SPA Iter.` lists the number of SPA iterations, the row labeled `Mat-mult updates` lists the number of matrix-vector multiplications required to compute the Nesterov updates and the row labeled `Mat-mult proj` lists the number of matrix-vector multiplications required to compute the projections. All other rows are self-explanatory. We generated $N = 10$ random instances for each of the experimental conditions. The column labeled `average` lists the average taken over the $N = 10$ random instances, the columns labeled `min` (resp. `max`) list the minimum (resp. maximum) over the 10 instances.

The experiment results support the following conclusions:

(a) SPA is very efficient in computing a solution to (1.1) – the algorithm requires anywhere from 7 to 14 iterations to converge.
(b) For a given sparsity type (high or low) and stopping criterion ϵ, the total number of SPA iterations and the number of Nesterov updates is a very slowly growing function of the dimension n of the target signal.
(c) The number of Nesterov updates (and also the overall running time of SPA) increases with the number of non-zero elements in the target signal x^*. Increasing the number of non-zero elements from $s = n/800$ to $s = n/40$ nearly doubled the total number of Nesterov updates.

As remarked in Section 4.3 we used the fixed set of constants in (5.2) to update the parameter sequence for all the experiments.

TABLE 5.2
Experiment Results for $m = n/4$, $s = m/10$ and $\|x^{sol} - x^*\|_\infty \leq 1$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	191.1	186	196	187.6	184	192	179.9	167	199
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	2.158E-05	8.671E-06	2.860E-05	1.988E-05	4.364E-06	2.824E-05	7.751E-06	1.350E-06	2.421E-05
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} > \mathbf{0}$	9.828E-01	9.675E-01	9.932E-01	9.821E-01	9.703E-01	9.981E-01	9.699E-01	9.035E-01	9.982E-01
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} = \mathbf{0}$	4.338E-01	2.596E-01	5.540E-01	3.500E-01	2.477E-01	5.196E-01	1.929E-01	5.254E-02	5.075E-01
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	2.915	2.095	6.133	2.007	0.999	3.833	1.185	0.343	1.614
$\|\mathbf{x^{sol}}\|_1$	56633019.9	55369477.2	59671255.5	14250930.8	13356266.4	15031117.1	1033580.7	836966.7	1289395.7
$\|\mathbf{x^*}\|_1$	56631797.7	55368213.9	59669841.3	14250648.3	13355937.4	15030813.1	1033574.2	836965.1	1289377.9
Time	114.2	110.9	116.8	26.4	25.8	27.2	1.5	1.4	1.6
SPA Iter. #	6.0	6.0	6.0	6.0	6.0	6.0	5.6	5.0	6.0
Mat-mult updates	383.2	373.0	393.0	376.2	369.0	385.0	360.8	335.0	399.0
Mat-mult proj	14.0	14.0	14.0	14.0	14.0	14.0	13.2	12.0	14.0
Mat-mult total	397.2	387	407	390.2	383	399	374	347	413

TABLE 5.3
Experiment Results for $m = n/4$, $s = m/100$ and $\|x^{sol} - x^*\|_\infty \leq 1$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	116.1	112	125	115.0	110	119	121.8	106	145
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	1.152E-04	6.023E-05	1.950E-04	1.418E-04	9.075E-05	2.010E-04	9.243E-05	3.745E-05	1.697E-04
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} > \mathbf{0}$	8.554E-01	5.139E-01	9.900E-01	8.639E-01	7.060E-01	9.798E-01	8.474E-01	5.931E-01	9.993E-01
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} = \mathbf{0}$	2.385E-01	9.342E-02	3.899E-01	2.522E-01	1.588E-01	4.040E-01	1.756E-01	5.006E-02	2.689E-01
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	2.598	1.563	3.301	1.546	1.353	1.706	0.536	0.442	0.692
$\|\mathbf{x^{sol}}\|_1$	5588869.7	4424350.4	7000986.9	1508227.7	1171948.5	1838361.6	193844.0	108724.1	311458.8
$\|\mathbf{x^*}\|_1$	5588239.3	4423488.0	7000565.2	1508021.0	1171713.0	1838194.7	193828.2	108705.6	311447.1
Time	68.3	65.8	73.7	15.9	15.3	16.3	1.2	1.0	1.4
SPA Iter. #	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
Mat-mult updates	233.2	225.0	251.0	231.0	221.0	239.0	244.6	213.0	291.0
Mat-mult proj	8.0	8.0	8.0	8.0	8.0	8.0	9.6	8.0	12.0
Mat-mult total	241.2	233	259	239	229	247	254.2	221	303

TABLE 5.4
Experiment Results for $m = n/4$, $s = m/10$ and $\|x^{sol} - x^*\|_\infty \leq 0.1$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	216.5	215	217	215.4	212	217	213.9	207	222
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	5.125E-07	4.156E-07	5.817E-07	5.285E-07	4.310E-07	6.059E-07	7.269E-07	4.826E-07	1.721E-06
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} > \mathbf{0}$	8.834E-02	7.672E-02	9.920E-02	8.838E-02	8.332E-02	9.452E-02	9.284E-02	8.504E-02	9.991E-02
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} = \mathbf{0}$	1.819E-02	1.145E-02	3.205E-02	1.850E-02	8.355E-03	2.799E-02	1.943E-02	8.087E-03	3.319E-02
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	0.947	0.906	1.029	0.496	0.483	0.520	0.138	0.073	0.159
$\|\mathbf{x^{sol}}\|_1$	56631826.7	55368245.1	59669869.9	14250655.8	13355944.7	15030820.1	1033575.0	836965.6	1289378.6
$\|\mathbf{x^*}\|_1$	56631797.7	55368213.9	59669841.3	14250648.3	13355937.4	15030813.1	1033574.2	836965.1	1289377.9
Time	129.2	128.1	130.2	30.3	29.8	30.7	1.8	1.7	2.0
SPA Iter. #	6.0	6.0	6.0	6.0	6.0	6.0	6.1	6.0	7.0
Mat-mult updates	434.0	431.0	435.0	431.8	425.0	435.0	428.8	415.0	445.0
Mat-mult proj	14.0	14.0	14.0	14.0	14.0	14.0	14.6	14.0	18.0
Mat-mult total	448	445	449	445.8	439	449	443.4	429	461

TABLE 5.5
Experiment Results for $m = n/4$, $s = m/100$ and $\|x^{sol} - x^*\|_\infty \leq 0.1$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	137.2	135	142	135.4	128	140	142.6	129	168
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	1.491E-05	7.940E-06	2.246E-05	1.414E-05	9.903E-06	1.926E-05	1.089E-05	5.200E-06	1.837E-05
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} > \mathbf{0}$	8.873E-02	7.052E-02	9.791E-02	9.024E-02	7.090E-02	9.967E-02	8.659E-02	6.325E-02	9.869E-02
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i)} = \mathbf{0}$	2.648E-02	1.545E-02	3.684E-02	2.417E-02	1.038E-02	4.421E-02	1.564E-02	5.112E-03	2.438E-02
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	0.262	0.172	0.349	0.129	0.094	0.200	0.052	0.040	0.066
$\|\mathbf{x^{sol}}\|_1$	5588321.0	4423587.4	7000662.6	1508041.6	1171732.9	1838213.3	193830.1	108707.6	311448.8
$\|\mathbf{x^*}\|_1$	5588239.3	4423488.0	7000565.2	1508021.0	1171713.0	1838194.7	193828.2	108705.6	311447.1
Time	81.1	79.9	84.2	18.8	17.9	19.4	1.4	1.3	1.7
SPA Iter. #	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0
Mat-mult updates	275.4	271.0	285.0	271.8	257.0	281.0	286.2	259.0	337.0
Mat-mult proj	10.0	10.0	10.0	10.0	10.0	10.0	11.6	10.0	14.0
Mat-mult total	285.4	281	295	281.8	267	291	297.8	269	351

TABLE 5.6
Experiment Results for $m = n/4$, $s = m/10$ and $\|x^{sol} - x^*\|_\infty \leq 0.01$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	265.9	255	273	258.4	251	269	254.7	245	274
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	2.153E-07	7.013E-08	3.162E-07	1.711E-07	6.319E-08	3.697E-07	1.314E-07	6.587E-08	3.730E-07
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i) > 0}$	9.726E-03	8.470E-03	9.988E-03	9.476E-03	8.663E-03	9.929E-03	9.412E-03	8.993E-03	9.917E-03
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i) = 0}$	3.999E-03	1.572E-03	5.407E-03	2.842E-03	1.349E-03	4.926E-03	2.501E-03	9.708E-04	4.786E-03
$\|\mathbf{Ax^{sol} - b}\|_2$	0.037	0.019	0.089	0.031	0.009	0.049	0.011	0.003	0.015
$\|\mathbf{x^{sol}}\|_1$	56631809.9	55368228.1	59669858.5	14250650.7	13355938.4	15030814.2	1033574.4	836965.2	1289378.0
$\|\mathbf{x^*}\|_1$	56631797.7	55368213.9	59669841.3	14250648.3	13355937.4	15030813.1	1033574.2	836965.1	1289377.9
Time	158.6	152.0	162.9	36.4	35.3	38.1	2.2	2.1	2.3
SPA Iter. #	7.8	7.0	8.0	7.5	7.0	8.0	7.3	7.0	8.0
Mat-mult updates	532.8	511.0	547.0	517.8	503.0	539.0	510.4	491.0	549.0
Mat-mult proj	17.6	16.0	18.0	17.0	16.0	18.0	17.0	16.0	22.0
Mat-mult total	550.4	527	565	534.8	519	557	527.4	507	571

TABLE 5.7
Experiment Results for $m = n/4$, $s = m/100$ and $\|x^{sol} - x^*\|_\infty \leq 0.01$

	n=512×512			**n=256×256**			**n=64×64**		
	Average	**Min**	**Max**	**Average**	**Min**	**Max**	**Average**	**Min**	**Max**
Nesterov updates #	155.4	143	162	154.3	139	160	161.3	145	189
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	1.464E-06	7.299E-07	3.310E-06	1.617E-06	7.140E-07	2.708E-06	1.206E-06	5.335E-07	1.964E-06
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i) > 0}$	8.525E-03	6.447E-03	9.731E-03	9.165E-03	6.617E-03	9.982E-03	8.967E-03	7.543E-03	9.992E-03
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty :\ \mathbf{x^*(i) = 0}$	2.631E-03	1.359E-03	5.592E-03	2.826E-03	1.251E-03	5.688E-03	1.461E-03	3.080E-04	3.141E-03
$\|\mathbf{Ax^{sol} - b}\|_2$	0.021	0.012	0.031	0.012	0.009	0.020	0.005	0.004	0.006
$\|\mathbf{x^{sol}}\|_1$	5588247.2	4423502.7	7000575.2	1508023.3	1171714.9	1838196.5	193828.4	108705.9	311447.3
$\|\mathbf{x^*}\|_1$	5588239.3	4423488.0	7000565.2	1508021.0	1171713.0	1838194.7	193828.2	108705.6	311447.1
Time	92.2	84.6	96.3	21.6	19.5	22.2	1.6	1.5	1.9
SPA Iter. #	6.9	6.0	7.0	6.9	6.0	7.0	7.0	7.0	7.0
Mat-mult updates	311.8	287.0	325.0	309.6	279.0	321.0	323.6	291.0	379.0
Mat-mult proj	11.8	10.0	12.0	11.8	10.0	12.0	13.6	12.0	16.0
Mat-mult total	323.6	297	337	321.4	289	333	337.2	303	395

5.3. Comparison of SPA with other solvers. In this section we report the result of our numerical experiments comparing SPA with NESTA [20] and FPC [10]. We created 10 random problems of size $n = 512 \times 512$ using the procedure described in Section 5.1.

We chose parameter values for each of the three algorithms so that they produced a solution x^{sol} with ℓ_∞-error approximately equal to 5×10^{-4}, i.e. $\|x^{sol} - x^*\|_\infty \approx 5 \times 10^{-4}$. We set the parameter values for each algorithm by solving a set of small size problems and these parameter values were fixed throughout the experiments.

1. For SPA, we set $\epsilon = 5 \times 10^{-5}$.
2. NESTA solves $\min_{\|Ax-b\|_2 \le \sigma} f_\mu(x)$, where $f_\mu(x) = \max_{\{u:\|u\|_\infty \le 1\}} \left\{ x^T u - \frac{\mu}{2}\|u\|_2^2 \right\}$, using continuation on μ. When σ is set to 0, NESTA handles $\|Ax - b\|_2 \le \sigma$ constraint as $Ax = b$ and since $AA^T = I$ is assumed, projections on to $\{x \in \Re^n : Ax = b\}$ affine space can be done efficiently. NESTA stops when $\frac{|f_\mu(x_k) - \bar{f}_\mu(x_k)|}{f_\mu(x_k)} < \delta$, for some $\delta > 0$, where $\bar{f}_\mu(x_k) = \frac{1}{min\{10,k\}} \sum_{\ell=1}^{min\{10,k\}} f_\mu(x_{k-\ell})$. For NESTA, we set $\mu = 1 \times 10^{-3}$ and $\delta = 1 \times 10^{-10}$.
3. FPC solves $\min_{x \in \Re^n} \|x\|_1 + \frac{1}{\lambda}\|Ax - b\|_2^2$. For FPC, we set $\frac{1}{\lambda} = 1.5 \times 10^4$.

The experimental results in Table 5.8-5.10, show that all three algorithms produce similar results with comparable running times. While SPA and NESTA required approximately the same number of matrix-vector multiplications, FPC required far fewer matrix-vector multiplications to produce a result of similar ℓ_∞-error.

6. Conclusion. We propose a smoothed penalty algorithm (SPA) for the sparse recovery problem. The SPA recovers the target signal by solving a sequence of smoothed penalized sub-problems, and each sub-problem is solved using Nesterov's optimal method for simple sets [18, 19]. We show that the continuation scheme used in SPA provably converges to the target signal and we are also able to compute a convergence rate. Since we penalize infeasibility by the exact penalty function $\|Ax - b\|$, where $\|\cdot\|$ can be ℓ_1, ℓ_2 or ℓ_∞ norm, an accurate solution is obtained before penalty parameter takes on arbitrarily small value; consequently, our proposed algorithm is numerically stable. We found that for a fixed measurement ratio m/n, sparsity ratio s/n, and solution accuracy ϵ, the total number of Nesterov iterations is effectively independent of the dimension n of the target signal; thus, one can tune the parameters on the smallest problem and use these parameters for all larger problems. The numerical results reported in this paper show that SPA required very few iterations to accurately recover the target signal.

SPA is a very general algorithmic framework that can be used for ℓ_1-minimization, relaxed ℓ_1-minimization, ℓ_1-minimization problems with linear side constraints, and also for convex optimization problems of the form

$$\min_{X=[x_1,\ldots,x_q]} \Big\{ \sum_{i \ne j} \|x_i - x_j\|_1 + \sum_{i=1}^{q} f(x_i) \Big\}, \tag{6.1}$$

that arise in the context of maximum likelihood estimation for sparse graphical networks. The cost of this flexibility is that SPA is not as efficient as algorithms such as FPC that explicitly utilize the ℓ_1 objective term. In [3] we propose a new penalty and augment Lagrangian based algorithm that explicitly uses the ℓ_1 structure and is competitive with other specialized algorithms for ℓ_1-minimization.

TABLE 5.8
SPA: Experiment Results for $m = n/4$, $s = m/10$

Run #	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Nesterov updates #	307	298	307	300	300	305	305
$\|\|\mathbf{x}^{\mathbf{sol}}\|_1 - \|\mathbf{x}^*\|_1\|/\|\mathbf{x}^*\|_1$	1.30E-08	1.08E-08	1.31E-08	1.02E-08	1.08E-08	1.21E-08	1.18E-08
$\|\mathbf{x}^{\mathbf{sol}} - \mathbf{x}^*\|_\infty : \mathbf{x}^*(\mathbf{i}) > \mathbf{0}$	7.37E-04	7.52E-04	6.75E-04	5.11E-04	6.12E-04	5.69E-04	6.74E-04
$\|\mathbf{x}^{\mathbf{sol}} - \mathbf{x}^*\|_\infty : \mathbf{x}^*(\mathbf{i}) = \mathbf{0}$	7.37E-05	1.72E-04	8.02E-05	5.84E-05	6.25E-05	7.21E-05	8.77E-05
$\|\mathbf{A}\mathbf{x}^{\mathbf{sol}} - \mathbf{b}\|_2$	5.54E-03	6.63E-03	5.39E-03	5.55E-03	6.10E-03	5.53E-03	5.70E-03
$\|\mathbf{x}^{\mathbf{sol}}\|_1$	57158337.83	57143128.23	55730561.39	55936883.68	55704724.67	57130433.69	59669841.99
$\|\mathbf{x}^*\|_1$	57158337.09	57143127.62	55730560.66	55936883.11	55704724.06	57130433	59669841.29
Time	182.2	175.3	181.1	176.4	176.2	179.4	179.3
SPA Iter. #	9	9	9	9	9	9	9
Mat-mult updates	615	597	615	601	601	611	611
Mat-mult proj	24	20	24	20	20	20	20
Mat-mult total	639	617	639	621	621	631	631

Run #	**8**	**9**	**10**	**Average**	**Min**	**Max**
Nesterov updates #	306	307	303	303.8	298	307
$\|\|\mathbf{x}^{\mathbf{sol}}\|_1 - \|\mathbf{x}^*\|_1\|/\|\mathbf{x}^*\|_1$	1.32E-08	1.35E-08	1.31E-08	1.22E-08	1.02E-08	1.35E-08
$\|\mathbf{x}^{\mathbf{sol}} - \mathbf{x}^*\|_\infty : \mathbf{x}^*(\mathbf{i}) > \mathbf{0}$	6.18E-04	6.64E-04	6.45E-04	6.46E-04	5.11E-04	7.52E-04
$\|\mathbf{x}^{\mathbf{sol}} - \mathbf{x}^*\|_\infty : \mathbf{x}^*(\mathbf{i}) = \mathbf{0}$	7.54E-05	7.91E-05	6.94E-05	8.30E-05	5.84E-05	1.72E-04
$\|\mathbf{A}\mathbf{x}^{\mathbf{sol}} - \mathbf{b}\|_2$	5.59E-03	5.36E-03	5.45E-03	5.68E-03	5.36E-03	6.63E-03
$\|\mathbf{x}^{\mathbf{sol}}\|_1$	55794694.99	55368214.63	56681162.57	56631798.37	55368214.63	59669841.99
$\|\mathbf{x}^*\|_1$	55794694.25	55368213.89	56681161.83	56631797.68	55368213.89	59669841.29
Time	180.8	181.1	178.8	179.0	175.3	182.2
SPA Iter. #	9	9	9	9	9	9
Mat-mult updates	613	615	607	608.6	597	615
Mat-mult proj	24	24	20	21.6	20	24
Mat-mult total	637	639	627	630.2	617	639

TABLE 5.9
NESTA: Experiment Results for $m = n/4$, $s = m/10$

Run #	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Nesterov updates #	313	314	314	313	313	315	312
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	6.48E-08	6.46E-08	6.65E-08	6.63E-08	6.66E-08	6.49E-08	6.19E-08
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} > \mathbf{0}$	7.44E-04	7.92E-04	7.24E-04	7.55E-04	8.23E-04	7.14E-04	7.57E-04
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} = \mathbf{0}$	2.08E-04	2.07E-04	1.93E-04	2.69E-04	1.94E-04	2.77E-04	2.04E-04
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	4.1188E-10	4.16459E-10	4.07487E-10	4.093E-10	4.10809E-10	4.18116E-10	4.27474E-10
$\|\mathbf{x^{sol}}\|_1$	57158340.79	57143131.31	55730564.37	55936886.82	55704727.78	57130436.71	59669844.98
$\|\mathbf{x^*}\|_1$	57158337.09	57143127.62	55730560.66	55936883.11	55704724.06	57130433	59669841.29
Time	152.7	153.5	153.6	153.2	153.1	154.1	152.5
Mat-mult total	627	629	629	627	627	631	625

Run #	**8**	**9**	**10**	**Average**	**Min**	**Max**
Nesterov updates #	313	313	317	313.7	312	317
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	6.64E-08	6.69E-08	6.56E-08	6.55E-08	6.19E-08	6.69E-08
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} > \mathbf{0}$	6.90E-04	7.78E-04	7.72E-04	7.55E-04	6.90E-04	8.23E-04
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} = \mathbf{0}$	2.62E-04	2.06E-04	2.60E-04	2.28E-04	1.93E-04	2.77E-04
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	4.12915E-10	4.06957E-10	4.2648E-10	4.14788E-10	4.06957E-10	4.27474E-10
$\|\mathbf{x^{sol}}\|_1$	55794697.96	55368217.59	56681165.54	56631801.38	55368217.59	59669844.98
$\|\mathbf{x^*}\|_1$	55794694.25	55368213.89	56681161.83	56631797.68	55368213.89	59669841.29
Time	153.4	153.2	155.1	153.4	152.5	155.1
Mat-mult total	627	627	635	628.4	625	635

TABLE 5.10
FPC: Experiment Results for $m = n/4$, $s = m/10$

Run #	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Nesterov updates #	189	189	189	191	192	190	191
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	3.43E-08	3.44E-08	3.53E-08	3.50E-08	3.49E-08	3.44E-08	3.27E-08
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} > \mathbf{0}$	6.71E-04	6.72E-04	6.72E-04	6.88E-04	6.72E-04	7.00E-04	6.67E-04
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} = \mathbf{0}$	1.37E-04	1.74E-04	1.94E-04	1.64E-04	1.30E-04	1.87E-04	1.50E-04
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	1.16E-02	1.17E-02	1.17E-02	1.16E-02	1.15E-02	1.17E-02	1.16E-02
$\|\mathbf{x^{sol}}\|_1$	57158335.13	57143125.65	55730558.7	55936881.15	55704722.12	57130431.03	59669839.33
$\|\mathbf{x^*}\|_1$	57158337.09	57143127.62	55730560.66	55936883.11	55704724.06	57130433	59669841.29
Time	97.8	94.7	99.4	103.4	91.2	90.0	90.4
Mat-mult total	379	379	379	383	385	381	383

Run #	**8**	**9**	**10**	**Average**	**Min**	**Max**
Nesterov updates #	191	189	189	190	189	192
$\|\|\mathbf{x^{sol}}\|_1 - \|\mathbf{x^*}\|_1\|/\|\mathbf{x^*}\|_1$	3.51E-08	3.54E-08	3.41E-08	3.46E-08	3.27E-08	3.54E-08
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} > \mathbf{0}$	7.34E-04	6.81E-04	6.31E-04	6.79E-04	6.31E-04	7.34E-04
$\|\mathbf{x^{sol}} - \mathbf{x^*}\|_\infty : \mathbf{x^*(i)} = \mathbf{0}$	1.61E-04	1.54E-04	1.16E-04	1.57E-04	1.16E-04	1.94E-04
$\|\mathbf{Ax^{sol}} - \mathbf{b}\|_2$	1.16E-02	1.16E-02	1.15E-02	1.16E-02	1.15E-02	1.17E-02
$\|\mathbf{x^{sol}}\|_1$	55794692.29	55368211.92	56681159.89	56631795.72	55368211.92	59669839.33
$\|\mathbf{x^*}\|_1$	55794694.25	55368213.89	56681161.83	56631797.68	55368213.89	59669841.29
Time	90.5	89.4	89.4	93.6	89.4	103.4
Mat-mult total	383	379	379	381	379	385

REFERENCES

[1] N. S. Aybat and A. Chakraborty, *Reconstruction of CT images from parsimonious angular measurements via compressed sensing*, tech. report, Siemens Corp. Research, 2009.

[2] N. S. Aybat and G. Iyengar, *Extended analysis of SPA algorithm*, tech. report, IEOR Department, Columbia University, 2009.

[3] ———, *A new first-order augmented Lagrangian algorithm for ℓ_1-recovery*, tech. report, IEOR Department, Columbia University, 2009.

[4] E. Candès and J. Romberg, *Quantitative robust uncertainty principles and optimally sparse decompositions*, Foundations of Computational Mathematics, 6 (2006), pp. 227–254.

[5] E. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Info. Th., 52 (2006).

[6] E. Candès and T. Tao, *Near optimal signal recovery from random projections: universal encoding strategies?*, IEEE Trans. Info. Th., 52 (2006), pp. 5406–5425.

[7] E. Van den Berg and M. P. Friedlander, *Probing the pareto frontier for basis pursuit solutions*, SIAM Journal on Scientific Computing, 31 (2008), pp. 890–912.

[8] D. Donoho, *Compressed sensing*, IEEE Trans. Info. Th., 52 (2006), pp. 1289–1306.

[9] E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation for $\ell 1$-regularized minimization with applications to compressed sensing*, tech. report, Rice University, 2007.

[10] ———, *Fixed-point continuation for $\ell 1$-minimization: Methodology and convergence*, SIAM Journal on Optimization, 19 (2008), pp. 1107–1130.

[11] S. Hoda, A. Gilpin, and J. Pena, *Smoothing techniques for computing nash equilibria of sequential games*, tech. report, Technical report, Carnegie Mellon University, 2008.

[12] P. J. Hüber, *Robust Statistics*, New York: Wiley, 1981.

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

[14] M. Fornasier I. Daubechies and I. Loris, *Accelerated projected gradient method for linear inverse problems with sparsity constraints*, Journal of Fourier Analysis and Applications, 14 (2008), pp. 764–792.

[15] Y. Singer J. Duchi, S. Shalew-Shwartz and T. Chandra, *Efficient projections onto the ℓ_1-ball for learning in high dimensions*, in Proceedings, Twenty-Fifth International Conference on Machine Learning, Andrew McCallum and Sam Roweis, eds., Helsinki, Finland, 2008, pp. 272–279.

[16] S. J. Kim K. Koh and S. Boyd, *Solver for $\ell 1$-regularized least squares problems*, tech. report, Stanford University, 2007.

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

[18] Yu. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, 2004.

[19] Yu. Nesterov, *Smooth minimization of nonsmooth functions*, Mathematical Programming, 103 (2005), pp. 127–152.

[20] J. Bobin S. Becker and E. Candès, *Nesta: a fast and accurate first-order method for sparse recovery*. Submitted for publication, April 2009.

[21] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation*, To appear in SIAM Journal on Scientific Computing, (2009).

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

Appendix A. Details of the steps in SPA. In this section, we collect together results that show that SPA is very efficient as long as one can compute the matrix-vector products Ax and $A^T y$ efficiently.

A.1. Bounds on $\|A\|_2$ and $\sigma_{\min}(A)$.

A.1.1. $\|A\|_2$ when measurements are discrete Fourier transforms. Let $C \in \mathbb{C}^{m\times n}$ be a partial fourier matrix, where rows of C are chosen randomly among the rows of n dimensional Fourier matrix. Without loss of generality, assume that n is an odd number. Since the target signal x^* takes real values, we can restrict the set of m randomly selected frequencies $\Gamma \subset \{0, 1, ..., \frac{n-1}{2}\}$ without any loss of generality.

Let $A_R = \Re(C)$, $A_I = \Im(C)$ and define

$$\bar{A} = \begin{bmatrix} A_R \\ A_I \end{bmatrix}.$$

Let $a^R(k)$ and $a^I(k)$ that denote the rows in A_R and in A_I, respectively, corresponding to the frequency index $k \in \Gamma$. Then

$$\begin{aligned} a^R(k) &= \left[\tfrac{1}{\sqrt{n}} \cos\left(\tfrac{2\pi jk}{n}\right)\right]_{j=0,\ldots,n-1} \\ a^I(k) &= -\left[\tfrac{1}{\sqrt{n}} \sin\left(\tfrac{2\pi jk}{n}\right)\right]_{j=0,\ldots,n-1}. \end{aligned}$$

Using simple properties of trigonometric sequences it is easy to establish that for all $k, l \in \Gamma$,

$$\begin{aligned} a^R(k)a^I(l)^T &= 0 \\ a^R(k)a^R(l)^T &= \begin{cases} 0, & \text{if } l \neq k, \\ \frac{1}{2}, & \text{if } l = k > 0; \\ 1, & \text{if } l = k = 0; \end{cases} \\ a^I(k)a^I(l)^T &= \begin{cases} 0, & \text{if } l \neq k, \\ \frac{1}{2}, & \text{if } l = k > 0; \\ 0, & \text{if } l = k = 0; \end{cases} \end{aligned} \tag{A.1}$$

The measurement matrix A is obtained by removing the row $a^I(0)$ from $\bar{A}$ if $0 \in \Gamma$; otherwise A is set to $\bar{A}$. Since the vector $a^I(0) = 0$, removing $a^I(0)$ does result in any loss of generality. Furthermore, (A.1) implies that AA^T is a diagonal matrix with entries taking values in the set $\{\frac{1}{2}, 1\}$. Thus, $\sigma_{\min}(A) = \frac{1}{\sqrt{2}}$ and $\|A\|_2 = \sigma_{\max}(A) = 1$.

A.1.2. $\|A\|_2$ when measurements are discrete Cosine transforms (DCT). A partial DCT matrix A satisfies $AA^T = I$. Therefore, $\|A\|_2 = \sigma_{min}(A) = \sigma_{max}(A) = 1$.

A.2. ℓ_2 or Least squares projection. In this section, we show that the ℓ_2-projection

$$\begin{aligned} \text{minimize} \quad & \|x - \hat{x}\|_2^2, \\ \text{subject to} \quad & Ax = b. \end{aligned} \tag{A.2}$$

can be computed efficiently. We compute this projection several times during the course of SPA. We initialize SPA by setting $x_0 = \text{argmin}\{\|x\|_2 \mid Ax = b\}$. Along the course of the algorithm we update β by solving $\min\{\|x - x_k\|_2 \mid Ax = b\}$. We show that both these problems can be solved efficiently. Note that in each instance of projection problem encountered during SPA, only the vector $\hat{x}$ changes but the matrix A remains constant.

In the Fourier case, the measurement matrix A is constructed as follows. Without loss of generality, assume that n is an odd number. Since the target signal x^* only takes real values, the set of m randomly selected frequencies $\Gamma \subset \{0, 1, ..., \frac{n-1}{2}\}$ without any loss of generality. Let $A_R = \Re(C)$ and $b_R = \Re(b)$ denote the real part of the matrix C and the vector b, respectively, and $A_I = \Im(C)$, $\bar{b}_I = \Im(b)$ denote the imaginary part of the matrix C and the vector b, respectively. Let

$$\bar{A} = \begin{bmatrix} A_R \\ A_I \end{bmatrix}, \qquad \bar{b} = \begin{bmatrix} b_R \\ b_I \end{bmatrix} \tag{A.3}$$

The measurement matrix A and the righthand-side vector b are constructed by removing redundant rows from the matrix $\bar{A}$ and the corresponding components from the vector $\bar{b}$. Since redundant equations do not alter the feasible region, i.e. $X = \{x \in \Re^n | Ax = b\} = \{x \in \Re^n | \overline{A}x = \bar{b}\}$, it follows that

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}\|x - \hat{x}\|_2^2, \\ \text{subject to} & Ax = b, \end{array}$$

is equivalent to

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}\|x - \hat{x}\|_2^2, \\ \text{subject to} & A_R x = b_R, \\ & A_I x = b_I. \end{array} \tag{A.4}$$

The optimal solution x^* to (A.4) satisfies the KKT conditions

$$\begin{array}{rcl} A_R^T\lambda_R + A_I^T\lambda_I - x^* & = & -\hat{x}, \\ A_R A_R^T\lambda_R + A_R A_I^T\lambda_I & = & b_R - A_R\hat{x}, \\ A_I A_R^T\lambda_R + A_I A_I^T\lambda_I & = & b_I - A_I\hat{x}, \end{array} \tag{A.5}$$

for some $\lambda_R, \lambda_I \in \mathbb{R}^m$. From Section A.1, $a^R(k)a^I(l)^T = 0$ for all k and l, i.e. $A_I A_R^T = A_R A_I^T = 0$. Thus, the KKT conditions (A.5) simply to

$$\begin{array}{rcl} A_R^T\lambda_R + A_I^T\lambda_I - x^* & = & -\hat{x}, \\ A_R A_R^T\lambda_R & = & b_R - A_R\hat{x}, \\ A_I A_I^T\lambda_I & = & b_I - A_I\hat{x}. \end{array} \tag{A.6}$$

The vectors $A_R\hat{x}$ and $A_I\hat{x}$ can be computed via a single FFT of $\hat{x}$ requiring $\mathcal{O}(n\log(n))$ operations. Since the matrix products $A_R A_R^T$ and $A_I A_I^T$ are diagonal (see Section A.1), we can compute λ_R and λ_I in $\mathcal{O}(m)$ operations. Next , $x^* = \hat{x} + A_R\lambda_R + A_I\lambda_I$ can be computed by one inverse FFT using $\mathcal{O}(n\log(n))$ operations. Thus, computing x requires $2\mathcal{O}(n\log(n)) + \mathcal{O}(m)$ operations.

Next, we consider the special case when A is a real matrix with orthonormal rows. This is the case when the measurement vector b corresponds to a sampled Discrete Cosine Transforms. Since the rows of A are orthonormal,

$$x^* = \text{argmin}\{\|x - \hat{x}\|_2 \mid Ax = b\} = \hat{x} + A^T(b - A\hat{x}).$$

Hence, if multiplications with A and A^T can be computed efficiently, the optimal projection x^* can be computed efficiently. When A is a partial DCT matrix, x^* can be computed by solving one forward and one inverse DCT, i.e. in $2\mathcal{O}(n\log(n))$ operations.

Next, consider the case when A is a real matrix with full row rank such that matrix-vector multiplications with A and A^T can be computed efficiently. Suppose Ax can computed in $\kappa_f(m,n)$ operations and $A^T y$ can computed in $\kappa_r(m,n)$ operations. In this case the optimal solution x^* satisfy the KKT conditions

$$\begin{array}{rcl} x^* & = & \hat{x} + A^T\lambda, \\ AA^T\lambda & = & b - A\hat{x}, \end{array}$$

for some $\lambda \in \mathbb{R}^m$. Since A is assumed to have full row rank, $(AA^T)^{-1} \in \mathbb{R}^{m\times m}$ exists. Hence, $x^* = \hat{x} + A^T(AA^T)^{-1}(b - A\hat{x})$. We compute x^* efficiently as follows. Let $A = U\Sigma V^T$ denote the singular value decomposition (SVD) of A where $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_m)$, with $\sigma_i > 0$ for all $i = 1, ..., m$, and $U, V \in \mathbb{R}^{n\times m}$ such that $U^TU = I$, $V^TV = I$. Computing the SVD takes $\mathcal{O}(m^2 n)$ operations.

The KKT multipliers $\lambda = \sum_{i=1}^m \left(\frac{u_i^T(b - A\hat{x})}{\sigma_i^2}\right)u_i$. Thus,

$$x^* = \hat{x} + \sum_{i=1}^m \left(\frac{u_i^T(b - A\hat{x})}{\sigma_i^2}\right)A^T u_i.$$

We compute and store the values $\{A^T u_i\}_{\{i=1,...,m\}}$ at the beginning of the algorithm, which requires $\mathcal{O}(m\kappa_r(m,n))$ operations. Given the precomputed values of $\{A^T u_i\}_{\{i=1,...,m\}}$, x^* can be computed in $\mathcal{O}(m(m+n) + \kappa_f(m,n))$ operations.

ℓ_1-PROJECTION $(\hat{x}, \sigma)$

$x_s \leftarrow \hat{x}$ sorted in increasing order

Compute $\rho = \max\left\{j \in \{1, ..., n\} : x_s(j) - \frac{1}{j}\left(\sum_{r=1}^{j} x_s(r) - \sigma\right) > 0\right\}$

$\xi \leftarrow \frac{1}{\rho}\left(\sum_{i=1}^{\rho} x_s(i) - \sigma\right)$

$x^*(i) \leftarrow \max\{\hat{x}(i) - \xi,\ 0\},\ i = 1, \ldots, n$

return x^*

FIG. A.1. *Projection onto the simplex*

A.3. Projection onto the ℓ_1-ball. In this section we show that the Euclidean projection problem

$$\begin{array}{rl} \text{minimize} & \|x - \hat{x}\|_2^2, \\ \text{subject to} & \|x\|_1 \leq \sigma, \end{array} \tag{A.7}$$

can be computed efficiently.

Define $\hat{y} = |\hat{x}|$ and consider the following optimization problem.

$$\begin{array}{rl} \text{minimize} & \|y - \hat{y}\|_2^2, \\ \text{subject to} & \sum_{i=1}^{n} y_i \leq \sigma, \\ & y \geq 0. \end{array} \tag{A.8}$$

Let x^* denote the projection of $\hat{x}$ onto the ℓ_1-ball, i.e. the optimal solution of (A.7). Then it is easy to check that $x^*(i)\hat{x}(i) \geq 0$ for all $i = 1, \ldots, n$, i.e. $x^*(i) = \text{sign}(\hat{x}(i))\,|x^*(i)|$, for all $i = 1, \ldots, n$. This implies that

$$x^*(i) = \text{sign}(\hat{x}(i))y^*(i), \quad i = 1, \ldots, n,$$

where y^* is the projection of $\hat{y}$ onto the simplex, i.e. the optimal solution of (A.8). Thus, the optimal solution of (A.7) can be recovered from the optimal solution of (A.8). Singer et al [15] show that the algorithm in Figure A.1 computes an optimal solution to (A.8) in $\mathcal{O}(n \log(n))$ operations.