SIAM J. SCI. COMPUT.
Vol. 33, No. 1, pp. 250–278

ALTERNATING DIRECTION ALGORITHMS FOR ℓ_1-PROBLEMS IN COMPRESSIVE SENSING*

JUNFENG YANG† AND YIN ZHANG‡

Abstract. In this paper, we propose and study the use of alternating direction algorithms for several ℓ_1-norm minimization problems arising from sparse solution recovery in compressive sensing, including the basis pursuit problem, the basis pursuit denoising problems of both unconstrained and constrained forms, and others. We present and investigate two classes of algorithms derived from either the primal or the dual form of ℓ_1-problems. The construction of the algorithms consists of two main steps: (1) to reformulate an ℓ_1-problem into one having blockwise separable objective functions by adding new variables and constraints; and (2) to apply an exact or inexact alternating direction method to the augmented Lagrangian function of the resulting problem. The derived alternating direction algorithms can be regarded as first-order primal-dual algorithms because both primal and dual variables are updated at every iteration. Convergence properties of these algorithms are established or restated when they already exist. Extensive numerical experiments are performed, using randomized partial Walsh–Hadamard sensing matrices, to demonstrate the versatility and effectiveness of the proposed approach. Moreover, we present numerical results to emphasize two practically important but perhaps overlooked points: (i) that algorithm speed should be evaluated relative to appropriate solution accuracy; and (ii) that when erroneous measurements possibly exist, the ℓ_1-fidelity should generally be preferable to the ℓ_2-fidelity.

Key words. compressive sensing, ℓ_1-minimization, primal, dual, augmented Lagrangian function, alternating direction method

AMS subject classifications. 65F22, 65J22, 65K10, 90C25, 90C06

DOI. 10.1137/090777761

1. Introduction.

In the last few years, algorithms for finding sparse solutions of underdetermined linear systems have been intensively studied, largely because solving such problems constitutes a critical step in an emerging methodology in digital signal processing—compressive sensing or sampling (CS). In CS, a digital signal is encoded as inner products between the signal and a set of random (or random-like) vectors where the number of such inner products, or linear measurements, can be significantly fewer than the length of the signal. On the other hand, the decoding process requires finding a sparse solution, either exact or approximate, to an underdetermined linear system. What makes such a scheme work is sparsity; i.e., the original signal must have a sparse or compressible representation under some known basis. Throughout this paper we will allow all involved quantities (signals, acquired data, and encoding matrices) to be complex. Let $\bar{x} \in \mathbb{C}^n$ be an original signal that we wish to capture. Without loss of generality, we assume that $\bar{x}$ is sparse under the canonical basis; i.e., the number of nonzero components in $\bar{x}$, denoted by $\|\bar{x}\|_0$, is far fewer than its length.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 19, 2009; accepted for publication (in revised form) November 5, 2010; published electronically February 3, 2011.

http://www.siam.org/journals/sisc/33-1/77776.html

†Department of Mathematics, Nanjing University, 22 Hankou Road, Nanjing, 210093, People's Republic of China (jfyang@nju.edu.cn). This author was supported by the Natural Science Foundation of China NSFC-11001123, NSFC-10971095.

‡Department of Computational and Applied Mathematics, Rice University, 6100 Main Street, MS-134, Houston, TX 77005 (yzhang@rice.edu). This author was supported in part by NSF DMS-0811188 and ONR grant N00014-08-1-1101.

Instead of sampling $\bar{x}$ directly, in CS one first obtains a set of linear measurements

$$b = A\bar{x} \in \mathbb{C}^m, \tag{1.1}$$

where $A \in \mathbb{C}^{m\times n}$ $(m < n)$ is an encoding matrix. The original signal $\bar{x}$ is then reconstructed from the underdetermined linear system $Ax = b$ via a certain reconstruction technique. Basic CS theory presented in [9, 11, 17] states that it is extremely probable to reconstruct $\bar{x}$ accurately or even exactly from b provided that $\bar{x}$ is sufficiently sparse (or compressible) relative to the number of measurements, and the encoding matrix A possesses certain desirable attributes.

In the rest of this section, we briefly review the essential ingredients of the CS decoding process and some existing methods for the relevant optimization problems, summarize our main contributions in this paper, and describe the notation and organization of the paper.

1.1. Signal decoding in CS. To make CS successful, two ingredients must be addressed carefully. First, a sensing matrix A must be designed so that the compressed measurement $b = A\bar{x}$ contains enough information for a successful recovery of $\bar{x}$. Second, an efficient, stable, and robust reconstruction algorithm must be available for recovering $\bar{x}$ from A and b. In the present paper, we will concentrate only on the second aspect.

In order to recover the sparse signal $\bar{x}$ from the underdetermined system (1.1), one could naturally consider seeking among all solutions of (1.1) the sparsest one, i.e., solving

$$\min_{x\in\mathbb{C}^n} \{\|x\|_0 : Ax = b\}, \tag{1.2}$$

where $\|x\|_0$ is the number of nonzeros in x. Indeed, with overwhelming probability decoder (1.2) can recover sparse signals exactly from a very limited number of random measurements (see, e.g., [3]). Unfortunately, this ℓ_0-problem is combinatorial and generally computationally intractable. A fundamental decoding model in CS is the so-called basis pursuit (BP) problem [14]:

$$\min_{x\in\mathbb{C}^n} \{\|x\|_1 : Ax = b\}. \tag{1.3}$$

Minimizing the ℓ_1-norm in (1.3) plays a central role in promoting solution sparsity. In fact, problem (1.3) shares common solutions with (1.2) under some favorable conditions (see, for example, [18]). When b contains noise, or when $\bar{x}$ is not exactly sparse but only compressible, as is the case in most practical applications, certain relaxation to the equality constraint in (1.3) is desirable. In such situations, common relaxations to (1.3) include the constrained basis pursuit denoising (BP_δ) problem [14],

$$\min_{x\in\mathbb{C}^n} \{\|x\|_1 : \|Ax - b\|_2 \le \delta\}, \tag{1.4}$$

and its variants, including the unconstrained basis pursuit denoising (QP_μ) problem

$$\min_{x\in\mathbb{C}^n} \|x\|_1 + \frac{1}{2\mu}\|Ax - b\|_2^2, \tag{1.5}$$

where $\delta, \mu > 0$ are parameters. From optimization theory, it is well known that problems (1.4) and (1.5) are equivalent in the sense that solving one will determine

a parameter value in the other so that the two share the same solution. As δ and μ approach zero, both BP_δ and QP_μ converge to (1.3). In this paper, we also consider the use of an ℓ_1/ℓ_1 model of the form

$$\min_{x\in\mathbb{C}^n} \|x\|_1 + \frac{1}{\nu}\|Ax-b\|_1 \tag{1.6}$$

whenever b might contain erroneous measurements. It is well known that unlike (1.5), where squared ℓ_2-norm fidelity is used, the ℓ_1-norm fidelity term makes (1.6) an exact penalty method in the sense that it reduces to (1.3) when $\nu > 0$ is less than some threshold.

It is worth noting that problems (1.3), (1.4), (1.5), and (1.6) all have their "nonnegative counterparts" where the signal x is real and nonnegative. These nonnegative counterparts will be briefly considered later. Finally, we mention that aside from ℓ_1-related decoders, there exist alternative decoding techniques such as greedy algorithms (e.g., [52]) which, however, are not a subject of concern in this paper.

1.2. Some existing methods. In the last few years, numerous algorithms have been proposed and studied for solving the aforementioned ℓ_1-problems arising in CS. Although these problems are convex programs with relatively simple structures (e.g., the BP problem is a linear program when x is real), they do demand dedicated algorithms because standard methods, such as interior-point algorithms for linear and quadratic programming, are simply too inefficient for them. This is the consequence of several factors, most prominently the fact that the data matrix A is totally dense while the solution is sparse. Clearly, the existing standard algorithms were not designed to handle such a feature. Another noteworthy structure is that encoding matrices in CS are often formed by randomly taking a subset of rows from orthonormal transform matrices, such as DCT (discrete cosine transform), DFT (discrete Fourier transform), or DWHT (discrete Walsh–Hadamard transform) matrices. Such encoding matrices do not require storage and enable fast matrix-vector multiplications. As a result, first-order algorithms that are able to take advantage of such a special feature lead to better performance and are highly desirable. In this paper we derive algorithms that take advantage of the structure ($AA^* = I$), and our numerical experiments are focused on randomized partial transform sensing matrices.

One of the earliest first-order methods applied to solving (1.5) is the gradient projection method suggested in [24] by Figueiredo, Nowak, and Wright, where the authors reformulated (1.5) as a box-constrained quadratic program and implemented a gradient projection method with line search. To date, the most widely studied class of first-order methods for solving (1.5) is variants of the iterative shrinkage/thresholding (IST) method, which was first proposed for wavelet-based image deconvolution (see [41, 16, 23], for example) and then independently discovered and analyzed by many others (for example, [21, 48, 49, 15]). In [31] and [32], Hale, Yin, and Zhang derived the IST algorithm from an operator splitting framework and combined it with a continuation strategy. The resulting algorithm, which is named fixed-point continuation (FPC), is also accelerated via a nonmonotone line search with Barzilai–Borwein steplength [4]. A similar sparse reconstruction algorithm called SpaRSA was also studied by Wright, Nowak, and Figueiredo in [58]. Recently, Beck and Teboulle proposed a fast IST algorithm (FISTA) in [5] which attains the same optimal convergence in function values as Nesterov's multistep gradient method [40] for minimizing composite convex functions. Lately, Yun and Toh also studied a block coordinate gradient descent (CGD) method in [63] for solving (1.5).

There also exist algorithms for solving constrained ℓ_1-problems (1.3) and (1.4). Bregman iterations, proposed in [42] and now known to be equivalent to the augmented Lagrangian method, were applied to the BP problem by Yin et al. in [60]. In the same paper, a linearized Bregman method was also suggested and was analyzed subsequently in [7, 8, 61]. In [54], Van den Berg and Friedlander proposed a spectral projection gradient method (SPGL1), where (1.4) is solved by a root-finding framework applied to a sequence of LASSO problems [51]. Moreover, based on a smoothing technique studied by Nesterov in [39], a first-order algorithm called NESTA was proposed by Becker, Bobin, and Candès in [6] for solving (1.4).

1.3. Contributions. After years of intensive research on ℓ_1-problem solving, it would appear that most relevant algorithmic ideas have been either tried or, in many cases, rediscovered. Yet, interestingly, until very recently the classic idea of alternating direction method (ADM) had not, to the best of our knowledge, been seriously investigated.

The *main contributions of this paper* are to introduce the ADM approach to the area of solving ℓ_1-problems in CS (as well as solving similar problems in image and signal processing) and to demonstrate its usefulness as a versatile and powerful algorithmic approach. From the ADM framework we have derived first-order primal-dual algorithms for models (1.3)–(1.6) and their nonnegative counterparts where signals are real and nonnegative. For each model, an ADM algorithm can be derived based on either the primal or the dual. Since the dual-based algorithms appear to be slightly more efficient when sensing matrices are orthonormal, we have implemented them in a MATLAB package called `YALL1` (short for Your ALgorithm for L1). Currently, `YALL1` [64] can effectively solve eight different ℓ_1-problems: models (1.3)–(1.6) and their nonnegative counterparts, where signals can be real (and possibly nonnegative) or complex, and orthonormal sparsifying bases and weights are also permitted in the ℓ_1-regularization term, which takes the more general form $\|Wx\|_{w,1} \triangleq \sum_{i=1}^{n} w_i|(Wx)_i|$ for any $W \in \mathbb{C}^{n\times n}$ with $W^*W = I$ and $w \in \mathbb{R}^n$ with $w \geq 0$.

In this paper, we present extensive computational results to document the numerical performance of the proposed ADM algorithms in comparison to several state-of-the-art algorithms for solving ℓ_1-problems under various situations, including FPC, SpaRSA, FISTA, and CGD for solving (1.5), and SPGL1 and NESTA for solving (1.3) and (1.4). As by-products, we also address a couple of related issues of practical importance, i.e., choices of optimization models and proper evaluation of algorithm speed.

1.4. Notation. We let $\|\cdot\|$ be the ℓ_2-norm and $\mathcal{P}_\Omega(\cdot)$ be the orthogonal projection operator onto a closed convex set Ω under the ℓ_2-norm. Superscripts "$\top$" and "$*$" denote, respectively, the transpose and the conjugate transpose operators for real and complex quantities. We let $\mathbf{Re}(\cdot)$ and $|\cdot|$ be, respectively, the real part and the magnitude of a complex quantity, which are applied componentwise to complex vectors. Further notation will be introduced wherever it occurs.

1.5. Organization. This paper is organized as follows. In section 2, we first review the basic idea of the classic ADM framework and then derive alternating direction algorithms for solving (1.3), (1.4), and (1.5). We also establish convergence of the primal-based algorithms, while that of the dual-based algorithms follows from classic results in the literature when sensing matrices have orthonormal rows. In section 3, we illustrate how to reduce model (1.6) to (1.3) and present numerical results to compare the behavior of model (1.6) to that of models (1.4) and (1.5) under various scenarios

of data noise. In section 4, we first re-emphasize the sometimes overlooked common sense on appropriate evaluations of algorithm speed and then present extensive numerical results on the performance of the proposed ADM algorithms in comparison to several state-of-the-art algorithms. Finally, we conclude the paper in section 5 and discuss several extensions of the ADM approach to other ℓ_1-like problems.

2. ADM-based first-order primal-dual algorithms. In this section, based on the classic ADM technique, we propose first-order primal-dual algorithms that update both primal and dual variables at each iteration for the solution of ℓ_1-problems. We start with a brief review of the general framework of ADM.

2.1. General framework of ADM. Let $f(x) : \mathbb{R}^m \to \mathbb{R}$ and $g(y) : \mathbb{R}^n \to \mathbb{R}$ be convex functions, $A \in \mathbb{R}^{p\times m}$, $B \in \mathbb{R}^{p\times n}$, and $b \in \mathbb{R}^p$. We consider the structured convex optimization problem

$$\min_{x,y} \{f(x) + g(y) : Ax + By = b\}, \tag{2.1}$$

where the variables x and y appear separately in the objective and are coupled only in the constraint. The augmented Lagrangian function of this problem is given by

$$\mathcal{L}_{\mathcal{A}}(x, y, \lambda) = f(x) + g(y) - \lambda^\top(Ax + By - b) + \frac{\beta}{2}\|Ax + By - b\|^2, \tag{2.2}$$

where $\lambda \in \mathbb{R}^p$ is a Lagrangian multiplier and $\beta > 0$ is a penalty parameter. The classic augmented Lagrangian method [36, 44] iterates as follows: given $\lambda^k \in \mathbb{R}^p$,

$$\begin{cases} (x^{k+1}, y^{k+1}) \leftarrow \arg\min_{x,y} \mathcal{L}_{\mathcal{A}}(x, y, \lambda^k), \\ \lambda^{k+1} \leftarrow \lambda^k - \gamma\beta(Ax^{k+1} + By^{k+1} - b), \end{cases} \tag{2.3}$$

where $\gamma \in (0, 2)$ guarantees convergence, as long as the subproblem is solved to an increasingly high accuracy at every iteration [46]. However, an accurate joint minimization with respect to both x and y can become costly. In contrast, ADM utilizes the separability structure in (2.1) and replaces the joint minimization by two simpler subproblems. Specifically, ADM minimizes $\mathcal{L}_{\mathcal{A}}(x, y, \lambda)$ with respect to x and y separately via a Gauss–Seidel-type iteration. After just one sweep of alternating minimization with respect to x and y, the multiplier λ is updated immediately. In short, given (y^k, λ^k), ADM iterates as follows:

$$\begin{cases} x^{k+1} \leftarrow \arg\min_x \mathcal{L}_{\mathcal{A}}(x, y^k, \lambda^k), \\ y^{k+1} \leftarrow \arg\min_y \mathcal{L}_{\mathcal{A}}(x^{k+1}, y, \lambda^k), \\ \lambda^{k+1} \leftarrow \lambda^k - \gamma\beta(Ax^{k+1} + By^{k+1} - b). \end{cases} \tag{2.4}$$

In the above, the domains for the variables x and y are assumed to be $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, but the derivation will be the same if these domains are replaced by closed convex sets $X \subset \mathbb{R}^m$ and $Y \subset \mathbb{R}^n$, respectively. In that case, the minimization problems in (2.4) will be over the sets X and Y, respectively.

The basic idea of ADM goes back to the work of Glowinski and Marocco [29] and Gabay and Mercier [26]. Let $\theta_1(\cdot)$ and $\theta_2(\cdot)$ be convex functionals, and let A be a continuous linear operator. The authors of [26] considered minimizing an energy function of the form

$$\min_u \theta_1(u) + \theta_2(Au).$$

By introducing an auxiliary variable v, the above problem was equivalently transformed to

$$\min_{u,v} \{\theta_1(u) + \theta_2(v) : Au - v = 0\},$$

which has the form of (2.1) and to which the ADM approach was applied. Subsequently, ADM was studied extensively in optimization and variational analysis. In [28], ADM is interpreted as the Douglas–Rachford splitting method [19] applied to a dual problem. The equivalence between ADM and a proximal point method is shown in [20]. The works applying ADM to convex programming and variational inequalities include [53, 25, 35], to mention just a few. Moreover, ADM has been extended to allowing inexact minimization (see [20, 34], for example).

In (2.4), a steplength $\gamma > 0$ is attached to the update of λ. Under certain technical assumptions, convergence of ADM with a steplength $\gamma \in (0, (\sqrt{5}+1)/2)$ was established in [27, 28] in the context of variational inequality. The shrinkage in the permitted range from $(0, 2)$ in the augmented Lagrangian method to $(0, (\sqrt{5}+1)/2)$ in ADM is related to relaxing the exact minimization of $\mathcal{L}_{\mathcal{A}}(x, y, \lambda^k)$ with respect to (x, y) to merely one round of alternating minimization.

Interestingly, the ADM approach was not widely utilized in the field of image and signal processing (including compressive sensing) until very recently when a burst of works applying ADM techniques appeared in 2009, including our ADM-based ℓ_1-solver package `YALL1` (see [64], published online in April 2009) and a number of ADM-related papers (see [22, 59, 30, 43, 1, 2], for example). The rest of the paper presents the derivation and performance of the proposed ADM algorithms for solving the ℓ_1-models (1.3)–(1.6) and their nonnegative counterparts, many of which have been implemented in `YALL1`.

2.2. Applying ADM to primal problems. In this subsection, we apply ADM to primal ℓ_1-problems (1.4) and (1.5). First, we introduce auxiliary variables to reformulate these problems in the form of (2.1). Then, we apply alternating minimization to the corresponding augmented Lagrangian functions, either exactly or approximately, to obtain ADM-like algorithms.

With an auxiliary variable $r \in \mathbb{C}^m$, problem (1.5) is clearly equivalent to

$$\min_{x\in\mathbb{C}^n,\, r\in\mathbb{C}^m} \left\{ \|x\|_1 + \frac{1}{2\mu}\|r\|^2 : Ax + r = b \right\}, \tag{2.5}$$

which has an augmented Lagrangian subproblem of the form

$$\min_{x\in\mathbb{C}^n,\, r\in\mathbb{C}^m} \left\{ \|x\|_1 + \frac{1}{2\mu}\|r\|^2 - \mathbf{Re}(y^*(Ax + r - b)) + \frac{\beta}{2}\|Ax + r - b\|^2 \right\}, \tag{2.6}$$

where $y \in \mathbb{C}^m$ is a multiplier and $\beta > 0$ is a penalty parameter. Given (x^k, y^k), we obtain $(r^{k+1}, x^{k+1}, y^{k+1})$ by applying alternating minimization to (2.6). First, it is easy to show that, for $x = x^k$ and $y = y^k$ fixed, the minimizer of (2.6) with respect to r is given by

$$r^{k+1} = \frac{\mu\beta}{1+\mu\beta}\left(y^k/\beta - (Ax^k - b)\right). \tag{2.7}$$

Second, for $r = r^{k+1}$ and $y = y^k$ fixed, simple manipulation shows that the minimization of (2.6) with respect to x is equivalent to

$$\min_{x\in\mathbb{C}^n} \|x\|_1 + \frac{\beta}{2}\|Ax + r^{k+1} - b - y^k/\beta\|^2, \tag{2.8}$$

which itself is in the form of (1.5). However, instead of solving (2.8) exactly, we approximate it by

$$\min_{x\in\mathbb{C}^n} \|x\|_1 + \beta\left(\mathbf{Re}((g^k)^*(x-x^k)) + \frac{1}{2\tau}\|x-x^k\|^2\right), \tag{2.9}$$

where $\tau > 0$ is a proximal parameter and

$$g^k \triangleq A^*(Ax^k + r^{k+1} - b - y^k/\beta) \tag{2.10}$$

is the gradient of the quadratic term in (2.8) at $x = x^k$ excluding the multiplication by β. The solution of (2.9) is given explicitly by (see, e.g., [15, 31])

$$x^{k+1} = \text{Shrink}\left(x^k - \tau g^k, \frac{\tau}{\beta}\right) \triangleq \max\left\{|x^k - \tau g^k| - \frac{\tau}{\beta}, 0\right\}\frac{x^k - \tau g^k}{|x^k - \tau g^k|}, \tag{2.11}$$

where all the operations are performed componentwise and $0 * \frac{0}{0} = 0$ is assumed. When the quantities involved are all real, the set of componentwise operation defined in (2.11) is well known as the one-dimensional shrinkage (or soft thresholding). Finally, we update the multiplier y by

$$y^{k+1} = y^k - \gamma\beta(Ax^{k+1} + r^{k+1} - b), \tag{2.12}$$

where $\gamma > 0$ is a constant. In short, ADM applied to (1.5) produces the iteration

$$\begin{cases} r^{k+1} = \frac{\mu\beta}{1+\mu\beta}\left(y^k/\beta - (Ax^k - b)\right), \\ x^{k+1} = \text{Shrink}(x^k - \tau g^k, \tau/\beta), \\ y^{k+1} = y^k - \gamma\beta(Ax^{k+1} + r^{k+1} - b). \end{cases} \tag{2.13}$$

We note that (2.13) is an inexact ADM because the x-subproblem is solved approximately. The convergence of (2.13) is not covered by the analysis given in [20], where each ADM subproblem is required to be solved more and more accurately as the algorithm proceeds. On the other hand, the analysis in [34] does cover the convergence of (2.13) but only for the case $\gamma = 1$. A more general convergence result for (2.13) that allows $\gamma > 1$ is established below. This is a meaningful extension since our experiments show that $\gamma > 1$ generally leads to faster convergence than $\gamma = 1$. Consequently, $\gamma > 1$ will always be used in our tests presented in section 4.

THEOREM 2.1. *Let $\tau, \gamma > 0$ satisfy $\tau\lambda_{\max} + \gamma < 2$, where $\lambda_{\max}$ denotes the maximum eigenvalue of A^*A. For any fixed $\beta > 0$, the sequence $\{(r^k, x^k, y^k)\}$ generated by (2.13) from any starting point (x^0, y^0) converges to $(\tilde{r}, \tilde{x}, \tilde{y})$, where $(\tilde{r}, \tilde{x})$ is a solution of (2.5).*

Proof. The proof is given in Appendix A. ∎

A similar alternating minimization idea can also be applied to problem (1.4), which is equivalent to

$$\min_{x\in\mathbb{C}^n,\, r\in\mathbb{C}^m} \{\|x\|_1 : Ax + r = b, \|r\| \le \delta\}, \tag{2.14}$$

and has an augmented Lagrangian subproblem of the form

$$\min_{x\in\mathbb{C}^n,\, r\in\mathbb{C}^m} \left\{\|x\|_1 - \mathbf{Re}(y^*(Ax + r - b)) + \frac{\beta}{2}\|Ax + r - b\|^2 : \|r\| \le \delta\right\}. \tag{2.15}$$

Similar to the derivation of (2.13), applying inexact alternating minimization to (2.15) yields the following iteration scheme:

$$\left\{\begin{array}{l} r^{k+1} = \mathcal{P}_{\mathbf{B}_\delta}\left(y^k/\beta - (Ax^k - b)\right), \\ x^{k+1} = \text{Shrink}(x^k - \tau g^k, \tau/\beta), \\ y^{k+1} = y^k - \gamma\beta(Ax^{k+1} + r^{k+1} - b), \end{array}\right. \tag{2.16}$$

where g^k is defined as in (2.10), and $\mathcal{P}_{\mathbf{B}_\delta}$ is the orthogonal projection (in the Euclidean norm) onto the set $\mathbf{B}_\delta \triangleq \{\xi \in \mathbb{C}^m : \|\xi\| \le \delta\}$. This algorithm also has a similar convergence result to that of (2.13).

THEOREM 2.2. *Let $\tau, \gamma > 0$ satisfy $\tau\lambda_{\max} + \gamma < 2$, where $\lambda_{\max}$ denotes the maximum eigenvalue of A^*A. For any fixed $\beta > 0$, the sequence $\{(r^k, x^k, y^k)\}$ generated by (2.16) from any starting point (x^0, y^0) converges to $(\tilde{r}, \tilde{x}, \tilde{y})$, where $(\tilde{r}, \tilde{x})$ solves (2.14).*

The proof of Theorem 2.2 is similar to that of Theorem 2.1 and thus is omitted.

We point out that, when $\mu = \delta = 0$, both (2.13) and (2.16) reduce to

$$\left\{\begin{array}{l} x^{k+1} = \text{Shrink}\left(x^k - \tau A^*(Ax^k - b - y^k/\beta), \tau/\beta\right), \\ y^{k+1} = y^k - \gamma\beta(Ax^{k+1} - b). \end{array}\right. \tag{2.17}$$

It is easy to show that x^{k+1} given in (2.17) is a solution of

$$\min_x \|x\|_1 - \mathbf{Re}((y^k)^*(Ax - b)) + \frac{\beta}{2\tau}\|x - (x^k - \tau A^*(Ax^k - b))\|^2, \tag{2.18}$$

which approximates at x^k the augmented Lagrangian subproblem of (1.3),

$$\min_x \|x\|_1 - \mathbf{Re}((y^k)^*(Ax - b)) + \frac{\beta}{2}\|Ax - b\|^2,$$

by linearizing $\frac{1}{2}\|Ax - b\|^2$ and adding a proximal term. Therefore, (2.17) is an inexact augmented Lagrangian algorithm for the BP problem (1.3). The only difference between (2.17) and the linearized Bregman method proposed in [60] lies in the updating of the multiplier. The advantage of (2.17) is that it solves (1.3), while the linearized Bregman method solves a penalty approximation of (1.3); see, e.g., [61]. We have the following convergence result for the iteration scheme (2.17).

THEOREM 2.3. *Let $\tau, \gamma > 0$ satisfy $\tau\lambda_{\max} + \gamma < 2$, where $\lambda_{\max}$ denotes the maximum eigenvalue of A^*A. For any fixed $\beta > 0$, the sequence $\{(x^k, y^k)\}$ generated by (2.17) from any starting point (x^0, y^0) converges to $(\tilde{x}, \tilde{y})$, where $\tilde{x}$ is a solution of (1.3).*

Proof. A sketch of the proof of this theorem is given in Appendix B. ∎

Since we applied the ADM idea to the primal problems (1.3), (1.4), and (1.5), we name the resulting algorithms (2.13), (2.16), and (2.17) primal-based ADMs or PADMs for short. In fact, these algorithms are really of primal-dual nature because both the primal and the dual variables are updated at each and every iteration. In addition, these are obviously first-order algorithms.

2.3. Applying ADM to dual problems. Similarly, we can apply the ADM idea to the dual problems of (1.4) and (1.5), resulting in equally simple yet more efficient algorithms when the sensing matrix A has orthonormal rows. Throughout this subsection, we will make the assumption that the rows of A are orthonormal, i.e., $AA^* = I$. At the end of this section, we will extend the derived algorithms to matrices with nonorthonormal rows.

Simple computation shows that the dual of (1.5) or equivalently (2.5) is given by

$$\begin{aligned}
&\max_{y\in\mathbb{C}^m}\min_{x\in\mathbb{C}^n,r\in\mathbb{C}^m}\left\{\|x\|_1+\frac{1}{2\mu}\|r\|^2-\mathbf{Re}(y^*(Ax+r-b))\right\}\\
&=\max_{y\in\mathbb{C}^m}\left\{\mathbf{Re}(b^*y)-\frac{\mu}{2}\|y\|^2+\min_{x\in\mathbb{C}^n}(\|x\|_1-\mathbf{Re}(y^*Ax))+\frac{1}{2\mu}\min_{r\in\mathbb{C}^m}\|r-\mu y\|^2\right\}\\
&=\max_{y\in\mathbb{C}^m}\left\{\mathbf{Re}(b^*y)-\frac{\mu}{2}\|y\|^2:A^*y\in\mathbf{B}_1^\infty\right\},
\end{aligned}\tag{2.19}$$

where $\mathbf{B}_1^\infty\triangleq\{\xi\in\mathbb{C}^n:\|\xi\|_\infty\le 1\}$. By introducing $z\in\mathbb{C}^n$, (2.19) is equivalently transformed to

$$\max_{y\in\mathbb{C}^m}\left\{f_d(y)\triangleq\mathbf{Re}(b^*y)-\frac{\mu}{2}\|y\|^2:z-A^*y=0,z\in\mathbf{B}_1^\infty\right\},\tag{2.20}$$

which has an augmented Lagrangian subproblem of the form

$$\min_{y\in\mathbb{C}^m,z\in\mathbb{C}^n}\left\{-\mathbf{Re}(b^*y)+\frac{\mu}{2}\|y\|^2-\mathbf{Re}(x^*(z-A^*y))+\frac{\beta}{2}\|z-A^*y\|^2,z\in\mathbf{B}_1^\infty\right\},\tag{2.21}$$

where $x\in\mathbb{C}^n$ is a multiplier (in fact, the primal variable) and $\beta>0$ is a penalty parameter. Now we apply the ADM scheme to (2.20). First, it is easy to show that, for $x=x^k$ and $y=y^k$ fixed, the minimizer z^{k+1} of (2.21) with respect to z is given explicitly by

$$z^{k+1}=\mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k+x^k/\beta),\tag{2.22}$$

where, as in the rest of the paper, $\mathcal{P}$ represents an orthogonal projection (in the Euclidean norm) onto a closed convex set denoted by the subscript. Second, for $x=x^k$ and $z=z^{k+1}$ fixed, the minimization of (2.21) with respect to y is a least squares problem, and the corresponding normal equations are

$$(\mu I+\beta AA^*)y=\beta Az^{k+1}-(Ax^k-b).\tag{2.23}$$

Under the assumption $AA^*=I$, the solution y^{k+1} of (2.23) is given by

$$y^{k+1}=\frac{\beta}{\mu+\beta}\left(Az^{k+1}-(Ax^k-b)/\beta\right).\tag{2.24}$$

Finally, we update x as follows:

$$x^{k+1}=x^k-\gamma\beta(z^{k+1}-A^*y^{k+1}),\tag{2.25}$$

where $\gamma\in(0,(\sqrt{5}+1)/2)$. Thus, the ADM scheme for (2.20) is as follows:

$$\left\{\begin{array}{l}
z^{k+1}=\mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k+x^k/\beta),\\
y^{k+1}=\frac{\beta}{\mu+\beta}\left(Az^{k+1}-(Ax^k-b)/\beta\right),\\
x^{k+1}=x^k-\gamma\beta(z^{k+1}-A^*y^{k+1}).
\end{array}\right.\tag{2.26}$$

Similarly, the ADM technique can also be applied to the dual of (1.4) given by

$$\max_{y\in\mathbb{C}^m}\{b^*y-\delta\|y\|:A^*y\in\mathbf{B}_1^\infty\}\tag{2.27}$$

and produces the iteration scheme

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = \mathcal{S}\left(Az^{k+1} - (Ax^k - b)/\beta, \delta/\beta\right), \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}), \end{cases} \tag{2.28}$$

where $\mathcal{S}(v, \delta/\beta) \triangleq v - \mathcal{P}_{\mathbf{B}_{\delta/\beta}}(v)$ with $\mathbf{B}_{\delta/\beta}$ being the Euclidean ball in $\mathbb{C}^m$ with radius δ/β.

Under the assumption $AA^* = I$, (2.26) is an exact ADM in the sense that each subproblem is solved exactly. From convergence results in [27, 28], for any $\beta > 0$ and $\gamma \in (0, (\sqrt{5}+1)/2)$, the sequence $\{(x^k, y^k, z^k)\}$ generated by (2.26) from any starting point (x^0, y^0) converges to $(\tilde{x}, \tilde{y}, \tilde{z})$, which solves the primal-dual pair (1.5) and (2.20). Similar arguments apply to (2.28) and the primal-dual pair (1.4) and (2.27).

Derived from the dual problems, we name the algorithms (2.26) and (2.28) dual-based ADMs or simply DADMs. Again we note these are in fact first-order primal-dual algorithms.

It is easy to show that the dual of (1.3) is given by

$$\max_{y \in \mathbb{C}^m} \{\mathbf{Re}(b^*y) : A^*y \in \mathbf{B}_1^\infty\}, \tag{2.29}$$

which is a special case of (2.19) and (2.27) with $\mu = \delta = 0$. Therefore, both (2.26) and (2.28) can be applied to solve (1.3). Specifically, when $\mu = \delta = 0$, both (2.26) and (2.28) reduce to

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = Az^{k+1} - (Ax^k - b)/\beta, \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}). \end{cases} \tag{2.30}$$

We note that the last equality in (2.19) holds if and only if $r = \mu y$. Therefore, the primal-dual residues and the duality gap between (2.5) and (2.20) can be defined by

$$\begin{cases} r_p \triangleq Ax + r - b \equiv Ax + \mu y - b, \\ r_d \triangleq A^*y - z, \\ \Delta \triangleq f_d(y) - f_p(x, r) \equiv \mathbf{Re}(b^*y) - \mu\|y\|^2 - \|x\|_1. \end{cases} \tag{2.31}$$

In computation, algorithm (2.26) can be terminated by

$$\text{Res} \triangleq \max\left\{\|r_p\|/\|b\|,\ \|r_d\|/\sqrt{m},\ \Delta/f_p(x, r)\right\} \le \epsilon, \tag{2.32}$$

where $\epsilon > 0$ is a stopping tolerance for the relative optimality residue.

When $AA^* \neq I$, the solution of (2.23) could be costly. In this case, we take a steepest descent step in the y direction and obtain the following iteration scheme:

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = y^k - \alpha_k^* g^k, \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}), \end{cases} \tag{2.33}$$

where g^k and α_k^* are given by

$$g^k = \mu y^k + Ax^k - b + \beta A(A^*y^k - z^{k+1}) \quad \text{and} \quad \alpha_k^* = \frac{(g^k)^* g^k}{(g^k)^*\left(\mu I + \beta AA^*\right) g^k}. \tag{2.34}$$

In our experiments, algorithm (2.33) converges very well for random matrices where $AA^* \neq I$, although its convergence remains an issue worthy of further research. Similar arguments apply to (2.27).

The ADM idea can also be easily applied to ℓ_1-problems for recovering real and nonnegative signals. As an example, we consider model (1.5) plus nonnegativity constraints:

$$\min_{x\in\mathbb{R}^n} \left\{ \|x\|_1 + \frac{1}{2\mu}\|Ax-b\|^2 : x \geq 0 \right\}, \tag{2.35}$$

where (A, b) can remain complex, e.g., A being a partial Fourier matrix. A derivation similar to that for (2.19) shows that a dual problem of (2.35) is equivalent to

$$\max_{y\in\mathbb{C}^m} \left\{ \mathbf{Re}(b^*y) - \frac{\mu}{2}\|y\|^2 : z - A^*y = 0, z \in \mathcal{F} \right\}, \tag{2.36}$$

where $\mathcal{F} \triangleq \{z \in \mathbb{C}^n : \mathbf{Re}(z) \leq 1\}$. The only difference between (2.36) and (2.20) lies in the changing of constraints on z from $z \in \mathbf{B}_1^\infty$ to $z \in \mathcal{F}$. Applying the ADM idea to (2.36) yields an iterative algorithm with the same updating formulae as (2.26) except the computation for z^{k+1} is replaced by

$$z^{k+1} = \mathcal{P}_{\mathcal{F}}(A^*y^k + x^k/\beta). \tag{2.37}$$

It is clear that the projection onto $\mathcal{F}$ is trivial. The same procedure applies to the dual problems of other ℓ_1-problems with nonnegativity constraints as well. Currently, with simple optional parameter settings, our MATLAB package `YALL1` [64] can be applied to models (1.3)–(1.6) and their nonnegative counterparts.

3. Choice of denoising models. In this section, we make a digression to emphasize an important issue in choosing denoising models in CS. In practical applications, measured data are usually contaminated by noise of different kinds or combinations. To date, the most widely used denoising models in CS are (1.4) and its variants that use the ℓ_2-fidelity, implicitly assuming that the noise is Gaussian. In this section, we aim to demonstrate that the model (1.6) with ℓ_1-fidelity is capable of handling several noise scenarios.

First, it is easy to observe that (1.6) can be reformulated into the form of the basis pursuit model (1.3). Clearly, (1.6) is equivalent to $\min_{x,r}\{\nu\|x\|_1 + \|r\|_1 : Ax + r = b\}$, which can be rewritten as

$$\min_{\hat{x}}\{\|\hat{x}\|_1 : \hat{A}\hat{x} = \hat{b}\}, \quad \text{where } \hat{A} = \frac{[A\ \ \nu I]}{\sqrt{1+\nu^2}},\ \hat{b} = \frac{\nu b}{\sqrt{1+\nu^2}},\ \hat{x} = \begin{pmatrix} \nu x \\ r \end{pmatrix}.$$

Moreover, we note that $\hat{A}\hat{A}^* = I$ whenever $AA^* = I$, allowing model (1.6) to be effectively solved by the ADM scheme (2.17) or (2.30).

In the following we provide evidence to show that model (1.6) can potentially be dramatically better than (1.4) whenever the observed data may contain large measurement errors (see also [57]). We conducted a set of experiments comparing ℓ_2-fidelity based models with (1.6) on random problems with $n = 1000$, $m = 300$, and $\|\bar{x}\|_0 = 60$, using the solver YALL1 [64] that implements the dual ADMs described in subsection 2.3. In our experiments, each model is solved for a sequence of parameter values (δ, μ, and ν in (1.4), (1.5), and (1.6), respectively) varying in $(0, 1)$. The simulation of data acquisition is given by $b = A\bar{x} + p_W + p_I \equiv b_W + p_I$, where matrix A

is random Gaussian with its rows orthonormalized by QR factorization, p_W and p_I represent white and impulsive noise, respectively, and b_W is the data containing white noise only. White noise is generated appropriately so that data b attains a desired signal-to-noise ratio (SNR), while impulsive noise values are set to ± 1 at random positions of b, which is always scaled so that $\|b\|_\infty = 1$. The SNR of b_W is defined as $\mathrm{SNR}(b_W) = 20\log_{10}(\|b_W - \mathbf{E}(b_W)\|/\|p_W\|)$, where $\mathbf{E}(b_W)$ represents the mean value of b_W. The severity of impulsive noise is measured by percentage. For a computed solution x, its relative error to $\bar{x}$ is defined as $\mathrm{RelErr}(x) = \|x - \bar{x}\|/\|\bar{x}\| \times 100\%$. For notational convenience, we will use BP_ν to refer to model (1.4) with δ replaced by ν in the figures and discussions of this section.

Figure 3.1 presents three types of results, i.e., impulsive noise only (first row), both white and impulsive noise (second row), and white noise only (third row). From the first row of Figure 3.1, it is quite clear that model (1.6) is able to recover the exact solution $\bar{x}$ to a high accuracy for a range of ν values (although the range for high quality recovery shrinks when the corruption rate increases), while model (1.4) with the same parameter values is not, even though in all cases it reduces relative errors by about 5% when ν is close to 1 (we tried even larger ν values but the achievable improvement soon saturates at that level). The results from model (1.5) (μ varies from 10^{-3} to 0.3) are generally similar to those of (1.4) and thus are omitted. Therefore, it is evident that in the presence of erroneous measurements, no matter how small the percentage might be, model (1.6) can be potentially much better than models (1.4) and (1.5) provided that ν is chosen properly.

For the case when data contain both white and impulsive noise, let us examine results given in the second row of Figure 3.1, where (1.6) is compared with (1.4) with data satisfying $\mathrm{SNR}(b_W) = 40$dB and p_I varies from 1% to 10%. Similar to the case free of white noise, evidence strongly suggests that (1.6) should be the model of choice whenever there might be erroneous measurements or impulsive noise in data even in the presence of white noise. We did not present the results of (1.5) since they are similar to those of (1.4). We also tested higher white noise levels and obtained similar results, except the quality of reconstruction deteriorates.

Is model (1.6) still appropriate without impulsive noise? The bottom row of Figure 3.1 contains results obtained from data with only white noise of $\mathrm{SNR}(b_W) =$ 30 dB, 20 dB, and 10 dB, respectively. Loosely speaking, these three types of data can be characterized as good, fair, and poor, respectively. As can be seen from the left plot, on good data (1.4) offers no improvement whatsoever to the BP model ($\nu = 0$) as ν decreases. On the contrary, it starts to degrade the quality of solution once $\nu > 0.25$. On the other hand, model (1.6) essentially does no harm until $\nu > 0.7$. From the middle plot, it can be seen that on fair data both models start to degrade the quality of solution after $\nu > 0.7$, while the rate of degradation is faster for model (1.6). Only in the case of poor data (the right plot), model (1.4) always offers better solution quality than model (1.6). However, for poor data the recovered solution quality is always poor. At $\nu = 1$, the relative error for model (1.4) is about 38%, representing a less than 5% improvement over the relative error 42% at $\nu = 0.05$, while the best error attained from model (1.6) is about 40%. The results of (1.5) are generally similar to those of (1.4) provided that model parameters are selected properly.

The sum of the computational evidence suggests the following three guidelines, at least for random problems of the type tested here: (i) whenever data may contain erroneous measurements or impulsive noise, ℓ_1-fidelity used by model (1.6) should

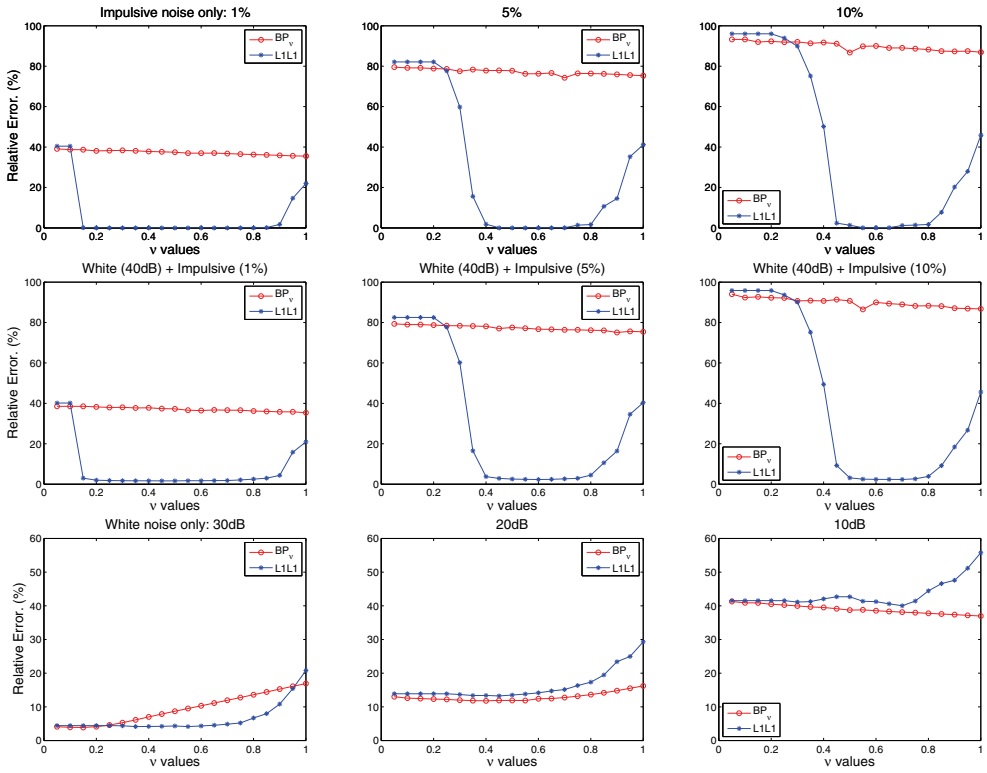

FIG. 3.1. *Comparison results of models* (1.4) *and* (1.6). *First row: Results from data corrupted by impulsive noise only (from left to right, the percentage of impulsive noise is* 1%, 5%, *and* 10%*). Second row: Results from data corrupted by both white and impulsive noise. Third row: Results from data contaminated by white noise only (from left to right, the SNR of* b_W *is* 30 *dB,* 20 *dB, and* 10 *dB). In all plots, the* x*-axes represent the parameter value in* (1.4) *and* (1.6)*, and the* y*-axes represent relative errors of recovered solutions to the true sparse signals.*

naturally be preferred over the ℓ_2-one used by model (1.4) and its variants; (ii) without impulsive noise, ℓ_1-fidelity basically does no harm to solution quality, as long as data do not contain a large amount of white noise and ν remains reasonably small; and (iii) when data are contaminated by a large amount of white noise, then ℓ_2-fidelity should be preferred. In the last case, however, high-quality recovery should not be expected regardless of what model is used.

4. Numerical results. In this section, we compare the proposed ADM algorithms, referred to as PADM and DADM, corresponding to the primal- and dual-based algorithms, respectively, with several state-of-the-art algorithms. In section 4.1, we give numerical results to emphasize a simple yet often overlooked point that algorithm speed should be evaluated relative to solution accuracy. In section 4.2, we describe our experiment settings, including parameter choices, stopping rules, and the generation of problem data under the MATLAB environment. In section 4.3, we compare PADM and DADM with FPC-BB [31, 32]—a fixed-point continuation method with a nonmonotone line search based on the Barzilai and Borwein (BB) steplength [4], SpaRSA [58]—a reconstruction algorithm designed for more general regularizers than the ℓ_1-regularizer, FISTA [5]—a fast IST algorithm that attains an optimal convergence rate in function values, and CGD [63]—a block coordinate gradient descent method for

minimizing ℓ_1-regularized convex smooth function. In section 4.4 we compare PADM and DADM with SPGL1 [54] — a spectral projected gradient algorithm for (1.4), and NESTA [6]—a first-order algorithm based on Nesterov's smoothing technique [39]. We also compare DADM with SPGL1 on the BP problem in section 4.5. All experiments were performed under Windows Vista Premium and MATLAB v7.8 (R2009a) running on a Lenovo laptop with an Intel Core 2 Duo CPU at 1.8 GHz and 2 GB of memory.

4.1. Relative error versus optimality. In algorithm assessment, the speed of an algorithm is often taken as an important criterion. However, speed is a relative concept and should be measured in company with appropriate accuracy, which clearly varies with situation and application. A relevant question here is what accuracy is reasonable for solving compressive sensing problems, especially when data are noisy, as is the case in most real applications. To address this question, we solved (1.3) with noiseless data and (1.5) with data contaminated by white noise of small to moderate levels. In this experiment, the measurement matrix was constructed by orthogonalizing and normalizing the rows of a 330 by 1000 standard Gaussian random matrix. The true signal $\bar{x}$ has 60 nonzeros whose positions are determined at random, and the nonzero values are random Gaussian. Both problems (1.3) and (1.5) were solved by DADM to a relative high accuracy (from an optimization rather than recovery standpoint). The results of relative error and optimality residue (defined in (2.32)) are given in Figure 4.1.

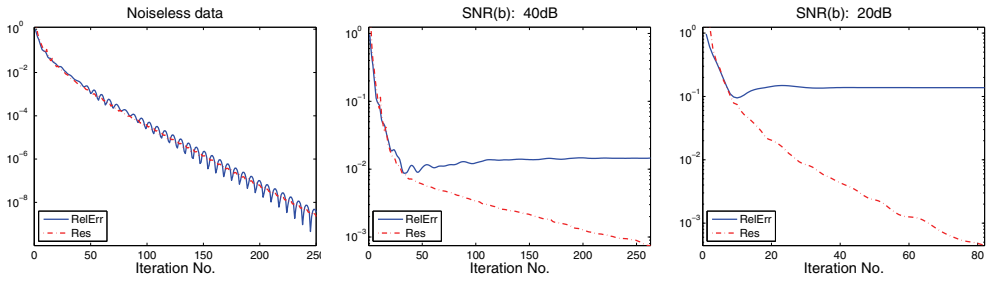

FIG. 4.1. *Relative error versus optimality for noiseless and noisy data. The x-axes represent the number of iterations, and y-axes represent the magnitude of the relative error (RelErr) and the optimality residue (Res) as defined in* (2.32).

It is clear from Figure 4.1 that solving ℓ_1-problems to increasingly high accuracy improves solution quality only when the observed data are free of noise. In the left plot of Figure 4.1 where noiseless data were used in (1.3), both relative error and optimality measured by residue decrease as DADM proceeds. For noisy data, a relatively low accuracy is sufficient to give the best relative error that an ℓ_1-denoising model can possibly reach, e.g., in the middle plot of Figure 4.1, where low level noisy data were used in (1.5), relative error does not decrease further after the residue is reduced to about 10^{-2} in about 40 iterations. This phenomenon becomes more obvious for noisy data of higher levels, as is shown in the right plot of Figure 4.1. These experiments clearly demonstrate that when observed data are noisy (which is common in practice), solving ℓ_1-problems to excessively high accuracy is unnecessary. This well-known fact is often refereed to as the Morozov discrepancy principle [38] in solving inverse and ill-posed problems (also see [33] for a discussion on solving ill-posed linear inverse problems). We choose to emphasize this rather mundane point because such common

sense has sometimes been ignored in algorithmic studies in the applications of CS. In our numerical comparison below, whenever noisy data are used we will compare not how fast algorithms achieve a *high* accuracy, but how fast they achieve an *appropriate* accuracy that is consistent with the noise level of data.

4.2. Experiment settings. Now we describe parameter selections, stopping rules, and generation of data in our numerical experiments. In order to test the proposed algorithms under conditions as realistic as practically feasible, which requires us to test sufficiently large problems and perform multiple runs for each test case, we chose to use randomized partial transform matrices in our experiments. Partial transform matrices do not require explicit storage and permit fast matrix-vector multiplications, which are the main computational tasks in all algorithms compared, allowing us to do more realistic experiments with relatively large-scale signals. Because of their low demands in computer resources and comparable recoverability with random matrices, randomized partial-transform matrices, such as DFT, DCT, and DWHT matrices, are arguably the most suitable for large-scale applications, as is pointed out in [6]. The proposed ADMs have the ability to take advantage of the orthonormality condition $AA^* = I$, which allows easy choice of parameters in PADM and exact minimization of subproblems in DADM. In our experiments we set $\tau = 0.8$, $\gamma = 1.199$, and $\beta = 2m/\|b\|_1$ in (2.13) and (2.16), which guarantee the convergence of PADM given that $\lambda_{\max}(A^*A) = 1$ and also work quite well in practice (although suitably larger τ and γ seem to accelerate convergence most of the time). For DADM, we used the default settings in YALL1, i.e., $\gamma = 1.618$ and $\beta = \|b\|_1/m$. As described in subsection 4.1, high accuracy is not always necessary in CS problems with noisy data. Thus, when comparing with other algorithms, we simply terminated PADM and DADM when the relative change of two consecutive iterates became small, i.e.,

$$\text{RelChg} \triangleq \frac{\|x^{k+1} - x^k\|}{\|x^k\|} < \epsilon, \tag{4.1}$$

where $\epsilon > 0$ is a tolerance, although more complicated stopping rules, such as the one based on optimality conditions defined in (2.32), are possible. Parametric settings of FPC-BB, SpaRSA, FISTA, CGD, SPGL1, and NESTA will be specified when we discuss individual experiments.

In all experiments, we generated data b by MATLAB scripts `b = A*xbar + sigma* randn(m,1)`, where `A` is a randomized partial Walsh–Hadamard transform matrix whose rows are randomly chosen and columns randomly permuted, `xbar` represents a sparse signal that we wish to recover, and `sigma` is the standard deviation of additive Gaussian noise. Specifically, the Walsh–Hadamard transform matrix of order 2^j is defined recursively by

$$H_{2^0} = [1]\,,\ H_{2^1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \ldots, H_{2^j} = \begin{bmatrix} H_{2^{j-1}} & H_{2^{j-1}} \\ H_{2^{j-1}} & -H_{2^{j-1}} \end{bmatrix}.$$

It can be shown that $H_{2^j} H_{2^j}^\top = 2^j I$. In our experiments, encoding matrix A contains random selected rows from $2^{j/2} H_{2^j}$, where $2^{j/2}$ is a normalization factor. A fast Walsh–Hadamard transform is implemented in C language with a MATLAB mex-interface available to all codes compared. In all tests, we set $n = 8192$ and tested various combinations of m and p (the number of nonzero components in `xbar`). In all the test results given below, we used the zero vector as the starting point for all algorithms unless otherwise specified.

4.3. Comparison with FPC-BB, SpaRSA, FISTA, and CGD. In this subsection, we present comparison results of PADM and DADM with FPC-BB [31, 32], SpaRSA [58], FISTA [5], and CGD [63], all of which were developed in the last two years for solving (1.5). In this test, we used random Gaussian spikes as `xbar`; i.e., the location of nonzeros are selected uniformly at random while the values of the nonzero components are i.i.d. standard Gaussian. The standard deviations of additive noise `sigma` and the model parameter μ in (1.5) are, respectively, set to 10^{-3} and 10^{-4}. Since different algorithms use different stopping criteria, it is rather difficult to compare their relative performance completely fairly. Therefore, we present two classes of comparison results. In the first class of results, we run all the algorithms for about 1000 iterations by adjusting their stopping rules. Then we examine how relative errors and function values decrease as each algorithm proceeds. In the second class of results, we terminate the ADM algorithms by (4.1), while the stopping rules used for other algorithms in comparison will be specified below.

Since FPC-BB implements continuation on the regularization parameter but not on stopping tolerance, we set all parameters as default, except that in the last step of continuation we let `xtol` $= 10^{-5}$ and `gtol` $= 0.02$, which is more stringent than the default setting `xtol` $= 10^{-4}$ and `gtol` $= 0.2$ because the latter usually produces solutions of lower quality than that of other algorithms in comparison. For SpaRSA, we used its monotonic variant, set continuation steps to 20, and terminated it when the relative change in function value fell below 10^{-7}. The FISTA algorithm [5] is a modification of the well-known IST algorithm [23, 41, 16]. Started at x^0, FISTA iterates as follows: $x^{k+1} = \text{Shrink}\left(y^k - \tau A^*(Ay^k - b), \tau/\mu\right)$, where $\tau > 0$ is a parameter, and

$$y^k = \begin{cases} x^0 & \text{if } k = 0, \\ x^k + \frac{t_{k-1}-1}{t_k}(x^k - x^{k-1}) & \text{otherwise,} \end{cases}$$

where

$$t_k = \begin{cases} 1 & \text{if } k = 0, \\ \frac{1+\sqrt{1+4t_{k-1}^2}}{2} & \text{otherwise.} \end{cases}$$

It is shown in [5] that FISTA attains an optimal convergence rate $O(1/k^2)$ in decreasing the function value, where k is the iteration counter. We set $\tau \equiv 1$ in the implementation of FISTA. For the comparison with CGD, we used its continuation variant (the code `CGD_cont` in the MATLAB package of CGD) and set all parameters as default, except we set the initial μ value to be $\max(0.01\|A^\top b\|_\infty, 2\mu)$, which works better than the default setting in our tests when μ is small.

Since the per-iteration cost is roughly two matrix-vector multiplications for all compared algorithms, it is meaningful to examine the decreasing behavior of relative errors and function values as functions of the iteration number. Figure 4.2 presents the results of two cases of m and p. Each result is the average of 50 runs on randomly generated data.

As can be seen from Figure 4.2, PADM and DADM usually decrease relative errors and function values faster than both FPC-BB and SpaRSA throughout the entire iteration process. Without using continuation and line search techniques as in FPC-BB and SpaRSA, FISTA is generally much slower than the others. In this set of experiments FISTA decreased function values faster at the very beginning but fell behind eventually. On the other hand, it was the slowest in decreasing relative

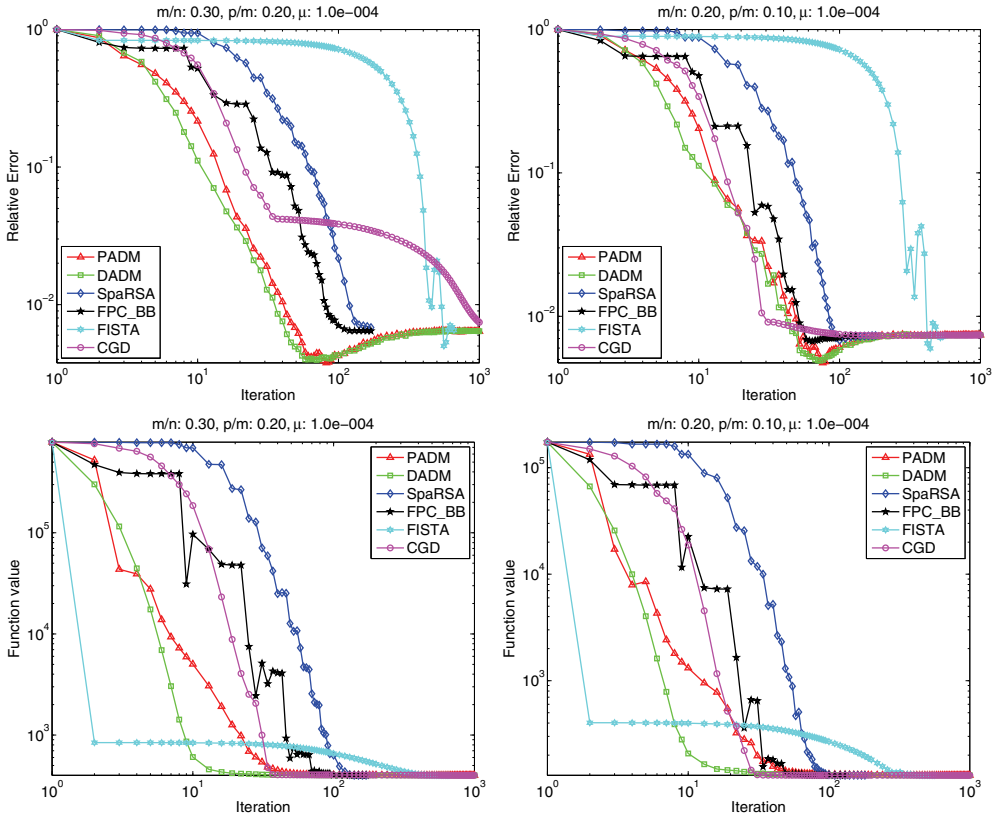

FIG. 4.2. *Comparison results of PADM, DADM, SpaRSA, FPC-BB, FISTA, and CGD on* (1.5) *(average of* 50 *random runs; standard deviation of Gaussian noise is* 10^{-3}*). The* x*-axes represent the number of iterations, and the* y*-axes represent relative errors (plots at the top) or function values (plots at the bottom), both in logarithmic scale.*

errors almost throughout the entire iteration process. We have found that the slow convergence of FISTA becomes even more pronounced when μ is smaller. On the first test set represented by the first column of Figure 4.2, both ADMs converge faster than CGD in decreasing both the relative error and the function value throughout the iteration process. On the second test set represented by the second column of Figure 4.2, CGD performed more competitively. However, CGD appeared to be sensitive to the choice of starting points. To demonstrate this, we tested the algorithms with another starting point $x^0 = A^\top b$ with all the other settings unchanged. The results for relative errors are given in Figure 4.3. By comparing Figure 4.3 with the first row of Figure 4.2, we observe that all algorithms exhibited consistent patterns of convergence except CGD, whose convergence is slower for $x^0 = A^\top b$ than for $x^0 = 0$.

It is worth noting that within no more than 100 iterations PADM and DADM reached lowest relative errors and then started to increase them, which reflects a property of model (1.5) rather than that of the algorithms given the fact that the function value kept decreasing. It is also clear that all algorithms eventually attained nearly equal relative errors and function values at the end. We point out that the performance of SpaRSA, FPC-BB, FISTA, and CGD is significantly affected by the value of μ. In general, model (1.5) becomes more and more difficult for continuation-based

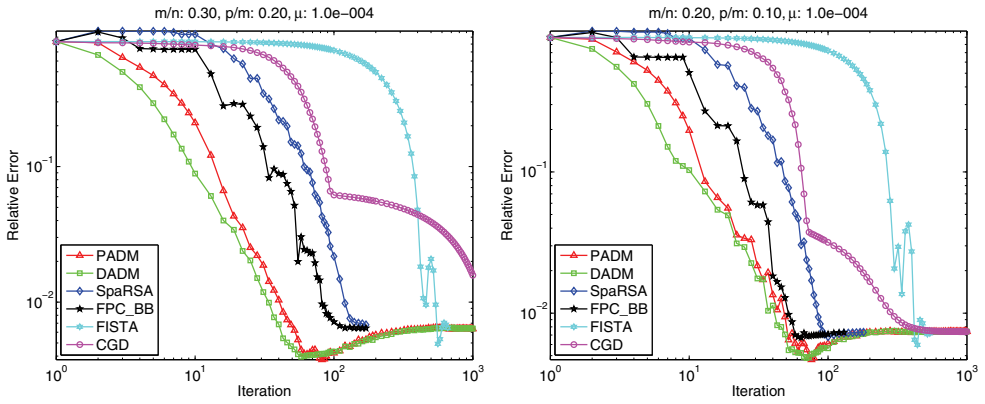

FIG. 4.3. *Comparison results of PADM, DADM, SpaRSA, FPC-BB, FISTA, and CGD on* (1.5) *(average of* 50 *random runs; standard deviation of Gaussian noise is* 10^{-3}*; and the common initial point for all algorithms is* $A^\top b$*). The x-axes represent the number of iterations, and the y-axes represent relative errors, both in logarithmic scale.*

algorithms (SpaRSA, FPC-BB, and CGD) as μ decreases, while the performance of ADMs is essentially unaffected, which can be well justified by the fact that μ can be set to 0 in both (2.13) and (2.26), in which case both algorithms solve the basis pursuit model (1.3).

In addition to the results presented in Figure 4.2, we also experimented on other combinations of (m, p) with noisy data and observed similar phenomena. As is the case in Figure 4.2, the relative error produced by the ADM algorithms tends to eventually increase after the initial decrease when problem (1.5) is solved to high accuracy. This implies, as suggested in section 4.1, that it is unnecessary to run the ADMs to a higher accuracy than what is warranted by the accuracy of the underlying data, though this is a difficult issue in practice since data accuracy is usually not precisely known.

Next we compare PADM and DADM with FPC-BB and SpaRSA for various combinations of (m, p), while keeping the noise level at `sigma` = 1e-3 and the model parameter at $\mu = 10^{-4}$. Here we do not include results for FISTA and CGD because they have been found to be less competitive on this set of tests. As is mentioned earlier, without continuation and line search techniques, FISTA is much slower than ADM algorithms. On the other hand, most of time CGD is slower in terms of decreasing relative errors, as is indicated by Figures 4.2 and 4.3.

We set all parameters as default in FPC-BB and use the same setting as before for SpaRSA, except it is terminated when relative change in function values falls below 10^{-4}. We set the stopping tolerance to $\epsilon = 2 \times 10^{-3}$ in (4.1) for PADM and DADM. The above stopping rules were selected so that all four algorithms attain more or less the same level of relative errors upon termination. For each fixed pair (m, p), we take the average of 50 runs on random instances. Detailed results including iteration number (Iter) and relative error to the true sparse signal (RelErr) are given in Table 4.1.

As can be seen from Table 4.1, in most cases PADM and DADM obtained smaller or comparable relative errors in fewer numbers of iterations than FPC-BB and SpaRSA. This is particularly evident for the case $(m/n, p/m) = (0.2, 0.2)$, where both PADM and DADM obtained notably smaller relative errors, while taking far

TABLE 4.1
Comparison results on (1.5) *(*`sigma` $= 10^{-3}$, $\mu = 10^{-4}$, *average of* 50 *runs).*

$n = 8192$		PADM		DADM		SpaRSA		FPC-BB	
m/n	p/m	Iter	RelErr	Iter	RelErr	Iter	RelErr	Iter	RelErr
0.3	0.1	38.9	6.70E-3	36.4	5.91E-3	103.6	5.61E-3	56.1	5.80E-3
0.3	0.2	50.2	6.52E-3	46.6	5.49E-3	141.3	7.25E-3	94.3	7.66E-3
0.2	0.1	57.2	7.17E-3	54.3	6.25E-3	114.5	7.53E-3	70.5	7.64E-3
0.2	0.2	63.1	8.54E-3	56.1	8.43E-3	180.0	1.68E-2	124.4	2.52E-2
0.1	0.1	85.5	1.17E-2	81.3	1.10E-2	135.6	1.27E-2	84.1	1.35E-2
0.1	0.2	125.4	9.70E-2	105.1	8.99E-2	214.4	1.60E-1	126.4	2.00E-1
Average		70.0	—	63.3	—	148.2	—	92.6	—

fewer iterations than FPC-BB and SpaRSA. At the bottom of Table 4.1, we calculate the average numbers of iterations required by the four algorithms.

We also tried more stringent stopping rules for the algorithms compared. Specifically, we tried `xtol`=10^{-5} and `gtol`=0.02 in FPC-BB and terminated SpaRSA when the relative change in the function value fell below 10^{-7}. The resulting relative errors either remained roughly the same as those presented in Table 4.1 or were just slightly better, while the iteration number required by FPC-BB increased about 50% and that required by SpaRSA increased more than 100%. For the ADMs, we have found that smaller tolerance values (say, $\epsilon = 5 \times 10^{-4}$) do not necessarily or consistently improve relative error results, while also increasing the required number of iterations.

4.4. Comparison with SPGL1 and NESTA. In this subsection, we compare PADM and DADM with SPGL1 and NESTA for solving model (1.4). As before, `xbar` consists of random Gaussian spikes, and the standard deviation of additive noise is `sigma` $= 10^{-3}$. The model parameter δ in (1.4) was set to be the 2-norm of additive noise (the ideal case). As in the previous experiment, we performed two sets of tests. In the first set, we ran all compared algorithms for about 400 iterations by adjusting their stopping tolerance values, while leaving all other parameters to their default values. Figure 4.4 presents average results on 50 random problems, where two combinations of m and p are used. The resulting relative error and residue in fidelity (i.e., $\|Ax - b\|$) are plotted as functions of iterations.

As can be seen from the first row of Figure 4.4, compared with SPGL1 and NESTA, both PADM and DADM attained smaller relative errors throughout most of the iteration process (with the exception of at the very beginning). With no more than 100 iterations, both PADM and DADM reached lowest relative errors and then started to increase them slightly. It is interesting to observe that NESTA is the slowest to decrease the relative error to the solution, although it is the fastest to attain the feasibility ($\|Ax - b\| \leq \delta$). In fact, NESTA is a feasible method that attains the feasibility after one iteration (see [6] for its algorithm construction), as can be seen from the second row of Figure 4.3.

In the second set of tests, we terminated PADM and DADM again with $\epsilon = 2 \times 10^{-3}$ in (4.1). For SPGL1 and NESTA, we set all parameters as default, except `TolVar` is set to be 10^{-6} in NESTA (where the default value is 10^{-5}) to obtain solutions of comparable quality. The average results on 50 random problems are given in Table 4.2. As mentioned, matrix-vector multiplications are the main computational load for all the first-order algorithms, but SPGL1 also requires substantial calculations of other kinds. For orthonormal sensing matrices, the number of matrix-vector multiplications required by PADM, DADM, and NESTA is two per iteration. When the sensing matrices are not orthonormal, the number remains at two for PADM (though it

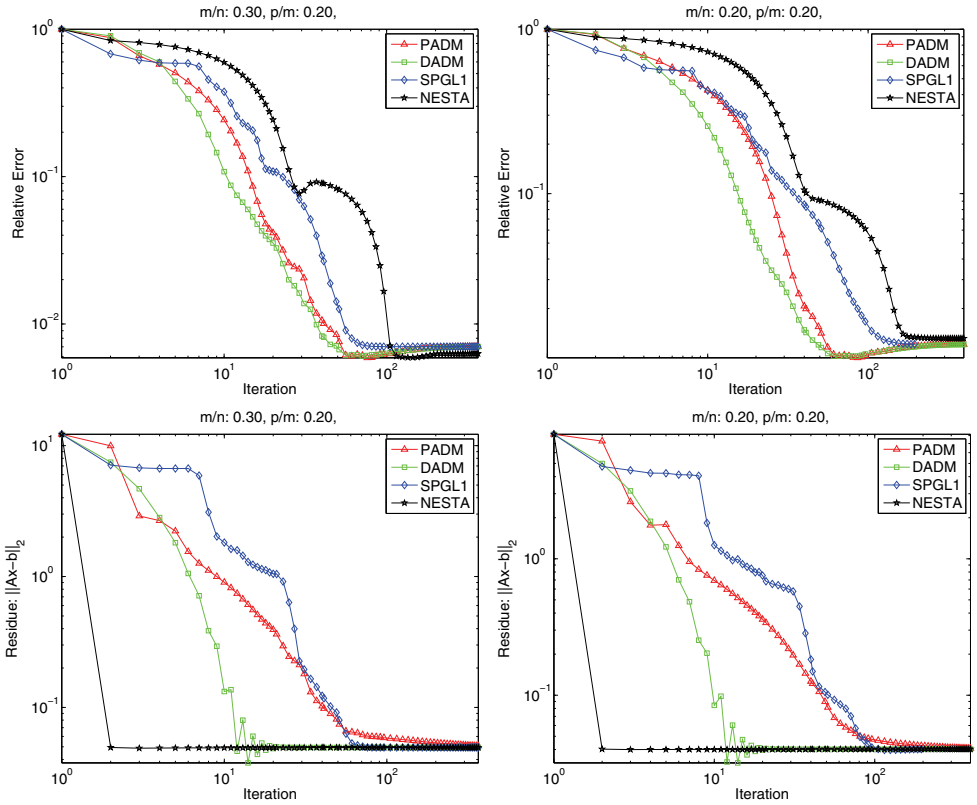

FIG. 4.4. *Comparison results of PADM, DADM, SPGL*1*, and NESTA on* (1.4). *The x-axes represent the number of iterations, and the y-axes represent the relative error (plots at the top) or the fidelity residue (plots at the bottom), both in logarithmic scale. The standard deviation of Gaussian noise is* 10^{-3}. *The results are average of* 50 *random runs.*

requires computing the maximum eigenvalue of AA^* to guarantee convergence), while the number increases to three for DADM (one extra required by computing α^* in (2.34)) and to six for NESTA (see [6]). On the other hand, the number required by SPGL1 can vary from iteration to iteration. To accurately reflect the computational costs consumed by the three algorithms, instead of iteration numbers we present in Table 4.2 the number of matrix-vector multiplications, denoted by #AAt, which includes both `A*x` and `A'*y`.

TABLE 4.2
Comparison results on (1.4) *(*`sigma` $= 10^{-3}$, $\delta =$ `norm(noise)`, *average of* 50 *runs).*

$n = 8192$		PADM		DADM		SPGL1		NESTA	
m/n	p/m	#AAt	RelErr	#AAt	RelErr	#AAt	RelErr	#AAt	RelErr
0.3	0.1	82.0	5.82E-3	74.6	7.64E-3	97.7	5.31E-3	297.2	5.74E-3
0.3	0.2	95.6	8.28E-3	90.0	7.36E-3	199.3	7.07E-3	304.3	8.18E-3
0.2	0.1	108.4	7.59E-3	101.0	8.76E-3	149.7	7.46E-3	332.5	6.99E-3
0.2	0.2	120.8	1.04E-2	108.6	1.06E-2	168.2	1.21E-2	336.9	5.77E-2
0.1	0.1	155.7	1.52E-2	149.4	1.42E-2	171.9	1.29E-2	340.6	1.72E-2
0.1	0.2	181.2	8.96E-2	187.8	8.22E-2	184.0	1.13E-1	363.4	3.10E-1
Average		124.0	—	118.6	—	161.8	—	329.2	—

TABLE 4.3

Comparison results on (1.3) *(b is noiseless; stopping rule is* $\epsilon = 10^{-6}$ *in* (4.1)*; average of* 50 *runs).*

$n = 8192$		DADM				SPGL1			
m/n	p/m	RelErr	RelRes	CPU	#AAt	RelErr	RelRes	CPU	#AAt
0.3	0.1	7.29E-5	4.41E-16	0.44	258.8	1.55E-5	9.19E-6	0.39	114.9
0.3	0.2	7.70E-5	4.65E-16	0.78	431.4	2.50E-5	6.77E-6	1.11	333.4
0.2	0.1	4.26E-5	4.54E-16	0.66	388.2	3.39E-5	1.51E-5	0.45	146.7
0.2	0.2	7.04E-5	4.85E-16	1.15	681.8	1.40E-4	1.03E-5	2.50	791.0
0.1	0.1	4.17E-5	4.86E-16	1.11	698.2	1.25E-4	3.26E-5	0.64	207.9
Average		—	—	0.83	491.7	—	—	1.02	318.8

As can be seen from Table 4.2, compared with SGPL1 and NESTA, both PADM and DADM obtained solutions of comparable quality within smaller numbers of matrix-vector multiplications. At the bottom of Table 4.2, we present the average numbers of matrix-vector multiplications required by the four algorithms.

4.5. Comparison with SPGL1 on BP problems. In this subsection, we compare DADM with SPGL1 on the BP problem (1.3). The relative performance of PADM and DADM has been illustrated in the previous comparisons, and DADM is slightly more efficient than PADM. Therefore, we present only results of DADM. We point out that NESTA can also solve (1.3) by setting $\delta = 0$. However, as is observable from results in Figure 4.4 and Table 4.2, NESTA is the slowest in decreasing the relative error. Thus, we compare DADM only with SPGL1. For BP problems data b is supposed to be noiseless, and higher accuracy optimization should lead to higher-quality solutions. Thus, we terminated DADM with a stringent stopping tolerance of $\epsilon = 10^{-6}$ in (4.1). All parameters in SPGL1 are set to be default values. Detailed comparison results are given in Table 4.3, where, besides relative error (`RelErr`) and the number of matrix-vector multiplications (#AAt), the relative residue `RelRes` $= \|Ax - b\|/\|b\|$ and CPU time in seconds are also given. The results for $m/n = 0.1$ and $p/m = 0.2$ are not included in Table 4.3 since both algorithms failed to recover accurate solutions.

We observe that when measurements are noiseless and a highly accurate solution is demanded, the ADM algorithms can sometimes be slower than SPGL1. Indeed, Table 4.3 shows that DADM is slower than SPGL1 in two cases (i.e., $(m/n, p/m) = (0.3, 0.1)$ and $(0.2, 0.1)$) while at the same time getting lower accuracy. On the other hand, it is considerably faster than SPGL1 in the case $(m/n, p/m) = (0.2, 0.2)$ while getting higher accuracy. The average CPU time and number of matrix-vector multiplications required by the two algorithms are presented in the last row of Table 4.3. We note that since SPGL1 requires some nontrivial calculations other than matrix-vector multiplications, a smaller #AAt number by SPGL1 does not necessarily lead to a shorter CPU time. We also comment that the relative residue results of DADM are always numerically zero because when $AA^* = I$ the sequence $\{x^k\}$ generated by DADM, applied to (1.3), satisfies $Ax^{k+1} - b = (1 - \gamma)(Ax^k - b)$, and thus $\|Ax - b\|$ decreases fairly quickly for $\gamma = 1.618$.

4.6. Summary. We provided supporting evidence to emphasize that algorithm speed should be evaluated relative to solution accuracy. With noisy measurements, solving ℓ_1-problems to excessively high accuracy is generally unnecessary. In practice, it is more relevant to evaluate the speed of an algorithm based on how fast it achieves an appropriate accuracy consistent with noise levels in data.

We presented extensive experimental results to compare the proposed ADM algorithms with state-of-the-art algorithms FPC-BB, SpaRSA, FISTA, CGD, SPGL1, and NESTA, using partial Walsh–Hadamard sensing matrices. Our numerical results show that the proposed algorithms are efficient and robust. In particular, the ADM algorithms can generally reduce relative errors faster than all other tested algorithms. This observation is based not only on results presented here using partial Walsh–Hadamard sensing matrices and Gaussian spike signals, but also on unreported results using other types of sensing matrices (partial DFT, DCT, and Gaussian random matrices) and sparse signals. In practice, however, since relative errors cannot be measured directly and do not seem to have predictable correlations with observable quantities such as fidelity residue, it remains practically elusive to take full advantage of such a favorable property of the ADM algorithms. Nevertheless, even with unnecessarily stringent tolerance values, the proposed ADM algorithms are still competitive with other state-of-the-art algorithms.

Our test results also indicate that the dual-based ADMs are generally more efficient than the primal-based ones. One plausible explanation is that when A is orthonormal, the dual-based algorithms are exact ADMs, while the primal-based ones, which solve some subproblems approximately, are inexact. The dual-based ADMs have been implemented in a MATLAB package called `YALL1` [64], which can solve eight different ℓ_1-models including (1.3)–(1.6) and their nonnegative counterparts.

5. Concluding remarks. We proposed solving ℓ_1-problems arising from compressive sensing by first-order primal-dual algorithms derived from the classic ADM framework utilizing the augmented Lagrangian function and alternating minimization idea. This ADM approach is applicable to numerous ℓ_1-problems including, but not limited to, the models (1.3)–(1.6) and their nonnegative counterparts. When applied to the ℓ_1-problems, the per-iteration cost of these algorithms is dominated by two matrix-vector multiplications. Extensive experimental results show that the proposed ADM algorithms, especially the dual-based ones, perform at least competitively with several state-of-the-art algorithms. On various classes of test problems with noisy data, the proposed ADM algorithms have unmistakably exhibited the following advantages over competing algorithms in comparison: (i) they converge well without the help of a continuation or a line search technique; (ii) their performance is insensitive to changes in model, starting point, and algorithm parameters; and (iii) they demonstrate a notable ability to quickly decrease the relative error to true solutions. Although the ADM algorithms are not necessarily the fastest in reaching an extremely high accuracy when observed data are noiseless, they are arguably the fastest in reaching the best achievable level of accuracy when data contain a nontrivial level of noise. However, to take full advantage of the ADMs one needs appropriate stopping tolerance values, which can be difficult to estimate in practice.

The most influential feature of the ADM approach is perhaps its great versatility and its seemingly universal effectiveness for a wide range of optimization problems in signal, image, and data analysis, particular those involving ℓ_1-like regularizations such as nuclear-norm (sum of singular values) regularization in matrix rank minimization like the matrix completion problem [45, 10, 12], or the total variation (TV) regularization [47] widely used in image processing. While the nuclear-norm is just an extension of the ℓ_1-norm to the matrix case, the TV regularization can be converted to ℓ_1-regularization after introducing a splitting variable [55, 59]. Therefore, the ADM approach is applicable to both nuclear-norm and TV regularized problems (in either

primal or dual form) in a rather straightforward manner so that the derivations and discussions are largely analogous to those for ℓ_1-problems as presented in this paper. Recently, the ADM has also been applied to TV-based image reconstruction in [22, 59, 50, 37] and to semidefinite programming in [56]. A more recent application of the ADM approach is to the problem of decomposing a given matrix into a sum of a low-rank matrix and a sparse matrix simultaneously using ℓ_1-norm and nuclear-norm regularizations (see [13]). An ADM scheme has been proposed and studied for this problem in [62].

Although the ADM approach is classic and its convergence properties have been well studied, its remarkable effectiveness in signal and image reconstruction problems involving ℓ_1-like regularizations has just been recognized very recently. These fruitful new applications bring new research issues, such as convergence of certain inexact ADM schemes and optimal choices of algorithm parameters, that should be interesting for further investigations.

Appendix A. Proof of Theorem 2.1.

Proof. Let $(\tilde{r}, \tilde{x})$ be any solution of (2.5). From optimization theory, there exists $\tilde{y} \in \mathbb{C}^m$ such that the following conditions are satisfied:

$$\tilde{r}/\mu - \tilde{y} = 0,\ A^*\tilde{y} \in \partial\|\tilde{x}\|_1,\ \text{ and }\ A\tilde{x} + \tilde{r} = b. \tag{A.1}$$

For convenience, we let $\hat{r} \triangleq r^{k+1}$, $\hat{x} \triangleq x^{k+1}$, and $\hat{y} \triangleq y^k - \beta(Ax^{k+1} + r^{k+1} - b)$. For $x = x^k$ and $y = y^k$ fixed, the minimizer r^{k+1} of (2.6) with respect to r satisfies

$$r^{k+1}/\mu + \beta(Ax^k + r^{k+1} - b - y^k/\beta) = 0. \tag{A.2}$$

Following the definitions of $\hat{r}$, $\hat{x}$, and $\hat{y}$, (A.2) can be rewritten as $\hat{r}/\mu - \hat{y} + \beta A(x^k - \hat{x}) = 0$. Further considering $\tilde{r}/\mu - \tilde{y} = 0$, we have $\hat{y} - \tilde{y} - \beta A(x^k - \hat{x}) = (\hat{r} - \tilde{r})/\mu$, and thus

$$(\hat{r} - \tilde{r})^* \left(\hat{y} - \tilde{y} - \beta A(x^k - \hat{x})\right) = \|\hat{r} - \tilde{r}\|^2/\mu \geq 0. \tag{A.3}$$

Similarly, the optimality condition for (2.9) takes the form of

$$\frac{\beta}{\tau}\left(x^k - \tau A^*(Ax^k + r^{k+1} - b - y^k/\beta) - x^{k+1}\right) \in \partial\|x^{k+1}\|_1. \tag{A.4}$$

From the definitions of $\hat{r}$, $\hat{x}$, and $\hat{y}$, (A.4) can be rewritten as $A^*\hat{y} - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x}) \in \partial\|\hat{x}\|_1$. Further considering $A^*\tilde{y} \in \partial\|\tilde{x}\|_1$ and the convexity of $\|\cdot\|_1$, it follows that

$$(\hat{x} - \tilde{x})^* \left(A^*\hat{y} - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x}) - A^*\tilde{y}\right) \geq 0,$$

or, equivalently,

$$(A\hat{x} - A\tilde{x})^* \left(\hat{y} - \tilde{y} - \beta A(x^k - \hat{x})\right) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq 0. \tag{A.5}$$

The addition of (A.3) and (A.5) yields

$$((A\hat{x} + \hat{r}) - (A\tilde{x} + \tilde{r}))^* \left(\hat{y} - \tilde{y} - \beta A(x^k - \hat{x})\right) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq 0. \tag{A.6}$$

Further considering $A\tilde{x} + \tilde{r} = b$ and $A\hat{x} + \hat{r} - b = (y^k - \hat{y})/\beta$, (A.6) can be represented as

$$\frac{1}{\beta}(\hat{y} - \tilde{y})^*(y^k - \hat{y}) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq (y^k - \hat{y})^* A(x^k - \hat{x}). \tag{A.7}$$

Let I_n be the identity matrix of order n. For convenience, we define

$$\text{(A.8)}\qquad \begin{aligned} G_0 &= \begin{pmatrix} I_n & \\ & \gamma I_m \end{pmatrix}, & G_1 &= \begin{pmatrix} \frac{\beta}{\tau} I_n & \\ & \frac{1}{\beta} I_m \end{pmatrix}, \\ G &= \begin{pmatrix} \frac{\beta}{\tau} I_n & \\ & \frac{1}{\beta\gamma} I_m \end{pmatrix}, & \text{and} \quad u &= \begin{pmatrix} x \\ y \end{pmatrix}. \end{aligned}$$

By using this notation and considering equality $\hat{u} - \tilde{u} = (\hat{u} - u^k) + (u^k - \tilde{u})$, (A.7) implies

$$\text{(A.9)}\qquad (u^k - \tilde{u})^* G_1 (u^k - \hat{u}) \geq \|u^k - \hat{u}\|^2_{G_1} + (y^k - \hat{y})^* A (x^k - \hat{x}).$$

From the definition of $\hat{y}$ and the formula for y^{k+1} in (2.13), we have $y^{k+1} = y^k - \gamma(y^k - \hat{y})$. Therefore, the iteration of x and y in (2.13) can be written as $u^{k+1} = u^k - G_0(u^k - \hat{u})$, and thus

$$\begin{aligned} \|u^{k+1} - \tilde{u}\|^2_G &= \|u^k - \tilde{u} - G_0(u^k - \hat{u})\|^2_G \\ &= \|u^k - \tilde{u}\|^2_G - 2(u^k - \tilde{u})^* G_0 G (u^k - \hat{u}) + \|G_0(u^k - \hat{u})\|^2_G. \end{aligned}$$

Considering the fact that $G_0 G = G_1$, the above equality implies

$$\begin{aligned} \|u^k - \tilde{u}\|^2_G - \|u^{k+1} - \tilde{u}\|^2_G &= 2(u^k - \tilde{u})^* G_1 (u^k - \hat{u}) - \|G_0(u^k - \hat{u})\|^2_G \\ \text{(from (A.9))} \quad &\geq 2\|u^k - \hat{u}\|^2_{G_1} + 2(y^k - \hat{y})^* A (x^k - \hat{x}) - \|u^k - \hat{u}\|^2_{G_0 G G_0} \\ \text{(from (A.8))} \quad &= \frac{\beta}{\tau}\|x^k - \hat{x}\|^2 + \frac{2-\gamma}{\beta}\|y^k - \hat{y}\|^2 + 2(y^k - \hat{y})^* A (x^k - \hat{x}). \end{aligned} \tag{A.10}$$

From condition $\tau\lambda_{\max} + \gamma < 2$, it holds that $\delta \triangleq 1 - \tau\lambda_{\max}/(2-\gamma) > 0$. Let $\rho \triangleq (2-\gamma)/(\beta + \beta\delta) > 0$. From the Cauchy–Schwarz inequality $2a^*b \geq -\rho\|a\|^2 - \|b\|^2/\rho$, (A.10) implies

$$\begin{aligned} \|u^k - \tilde{u}\|^2_G - \|u^{k+1} - \tilde{u}\|^2_G &\geq \frac{\beta}{\tau}\|x^k - \hat{x}\|^2 + \frac{2-\gamma}{\beta}\|y^k - \hat{y}\|^2 \\ &\quad - \rho\|y^k - \hat{y}\|^2 - \frac{1}{\rho}\|A(x^k - \hat{x})\|^2 \\ \text{(from } \lambda_{\max} = \lambda_{\max}(A^*A)) \quad &\geq \left(\frac{\beta}{\tau} - \frac{\lambda_{\max}}{\rho}\right)\|x^k - \hat{x}\|^2 \\ &\quad + \left(\frac{2-\gamma}{\beta} - \rho\right)\|y^k - \hat{y}\|^2 \\ \text{(from definitions of } \delta \text{ and } \rho) \quad &= \frac{\beta\delta^2}{\tau}\|x^k - \hat{x}\|^2 + \frac{2-\gamma}{\beta}\frac{\delta}{1+\delta}\|y^k - \hat{y}\|^2 \\ \text{(from definitions of } \hat{x} \text{ and } \hat{y}) \quad &= \frac{\beta\delta^2}{\tau}\|x^k - x^{k+1}\|^2 \\ &\quad + \frac{2-\gamma}{\beta\gamma^2}\frac{\delta}{1+\delta}\|y^k - y^{k+1}\|^2 \\ \text{(from definitions of } \hat{x}, \hat{y} \text{ and } G) \quad &\geq \eta\|u^k - u^{k+1}\|^2_G, \end{aligned} \tag{A.11}$$

where $\eta \triangleq \min(\delta^2, \frac{\delta(2-\gamma)}{\gamma(1+\delta)}) > 0$. It follows from (A.11) that

(a) $\|u^k - u^{k+1}\|_G \to 0$;

(b) $\{u^k\}$ lies in a compact region; and
(c) $\|u^k - \tilde{u}\|_G^2$ is monotonically nonincreasing and thus converges.

From (a), there hold $x^k - x^{k+1} \to 0$ and $y^k - y^{k+1} \to 0$. From $y^k = y^{k-1} - \gamma\beta(Ax^k + r^k - b)$ and $y^{k-1} - y^k \to 0$, it follows that $Ax^k + r^k - b \to 0$. From (b), $\{u^k\}$ has a subsequence $\{u^{k_j}\}$ that converges to $u^\star = (x^\star; y^\star)$; i.e., $x^{k_j} \to x^\star$ and $y^{k_j} \to y^\star$. From the iteration formula (2.7), we have

$$r^k = \frac{\mu\beta}{1+\mu\beta}\left(\frac{y^k}{\beta} - \left(Ax^k - b\right) + \frac{y^{k-1} - y^k}{\beta} + A\left(x^k - x^{k-1}\right)\right).$$

Taking into consideration $x^{k_j} \to x^\star$, $y^{k_j} \to y^\star$, $x^k - x^{k+1} \to 0$, and $y^k - y^{k+1} \to 0$, the above equality implies

$$r^{k_j} \to r^\star \triangleq \frac{\mu\beta}{1+\mu\beta}\left(\frac{y^\star}{\beta} - (Ax^\star - b)\right), \quad j \to \infty.$$

Therefore, $(r^\star, x^\star, y^\star)$ is a limit point of $\{(r^k, x^k, y^k)\}$. Next we show that $(r^\star, x^\star, y^\star)$ satisfies the optimality conditions for (2.5). First, from (2.12) and (2.7), we have

$$\begin{aligned} y^{k+1} &= y^k - \gamma\beta(Ax^{k+1} + r^{k+1} - b) \\ &= y^k - \gamma\beta\left(Ax^{k+1} - b + \frac{\mu\beta}{1+\mu\beta}\left(\frac{y^k}{\beta} - (Ax^k - b)\right)\right), \end{aligned}$$

which is equivalent to

$$\frac{y^k - y^{k+1}}{\gamma\beta} + A(x^k - x^{k+1}) = \frac{1}{1+\mu\beta}\left(Ax^k - b + \mu y^k\right). \tag{A.12}$$

By taking the limit of (A.12) over k_j, it follows that

$$Ax^\star - b + \mu y^\star = 0. \tag{A.13}$$

Second, from the definition of $r^\star$, it holds that

$$r^\star/\mu - y^\star = \frac{\beta}{1+\mu\beta}\left(\frac{y^\star}{\beta} - (Ax^\star - b)\right) - y^\star = \frac{\beta}{1+\mu\beta}\left(b - Ax^\star - \mu y^\star\right) = 0, \tag{A.14}$$

where the last equality comes from (A.13). Finally, (A.4) can be represented as

$$\frac{\beta}{\tau}\left((I - \tau A^*A)(x^k - x^{k+1}) - \tau A^*(Ax^{k+1} + r^{k+1} - b - y^k/\beta)\right) \in \partial\|x^{k+1}\|_1. \tag{A.15}$$

Since $x^{k_j} \to x^\star$ and $x^{k+1} - x^k \to 0$, we have $x^{k_j+1} = x^{k_j} + (x^{k_j+1} - x^{k_j}) \to x^\star$. By taking the limit of (A.15) over k_j and further considering $Ax^k + r^k - b \to 0$, it follows that $A^*y^\star \in \partial\|x^\star\|_1$, which together with (A.13) and (A.14) implies that $(r^\star, x^\star, y^\star)$ satisfies the optimality conditions for (2.5), i.e.,

$$r^\star/\mu - y^\star = 0, A^*y^\star \in \partial\|x^\star\|_1, \text{ and } Ax^\star + r^\star = b. \tag{A.16}$$

Therefore, we have shown that any limit point of $\{(r^k, x^k, y^k)\}$ is an optimal solution of (2.5). Since (A.11) holds for any optimal solution of (2.5), by letting $\tilde{u} = (\tilde{x}, \tilde{y}) = (x^\star, y^\star)$ at the beginning and considering (c), we get the convergence of $\{u^k\}$ and thus that of $\{(r^k, x^k, y^k)\}$. ∎

Appendix B. Proof of Theorem 2.3.

Proof. Suppose $(\tilde{x}, \tilde{y})$ satisfies the optimality conditions for (1.3), i.e., $A^*\tilde{y} \in \partial\|\tilde{x}\|_1$ and $A\tilde{x} = b$. Let $\hat{x} \triangleq x^{k+1}$ and $\hat{y} \triangleq y^k - \beta(Ax^{k+1} - b)$. The optimality condition for (2.18) takes the form

$$\frac{\beta}{\tau}\left(x^k - \tau A^*(Ax^k - b - y^k/\beta) - x^{k+1}\right) \in \partial\|x^{k+1}\|_1. \tag{B.1}$$

From the definitions of $\hat{x}$ and $\hat{y}$, (B.1) can be rewritten as $A^*\hat{y} - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x}) \in \partial\|\hat{x}\|_1$. Further considering $A^*\tilde{y} \in \partial\|\tilde{x}\|_1$ and the convexity of $\|\cdot\|_1$, it follows that

$$(\hat{x} - \tilde{x})^*\left(A^*\hat{y} - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x}) - A^*\tilde{y}\right) \geq 0,$$

or, equivalently,

$$(A\hat{x} - A\tilde{x})^*\left(\hat{y} - \tilde{y} - \beta A(x^k - \hat{x})\right) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq 0. \tag{B.2}$$

Further considering $A\tilde{x} = b$ and $A\hat{x} - b = (y^k - \hat{y})/\beta$, (B.2) can be represented as

$$\frac{1}{\beta}(\hat{y} - \tilde{y})^*(y^k - \hat{y}) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq (y^k - \hat{y})^*A(x^k - \hat{x}),$$

which is the key inequality for the convergence of $\{(x^k, y^k)\}$. By using the same notation as defined in (A.8) and similar discussions as in Appendix A, we can prove that

(a) $\|u^k - u^{k+1}\|_G \to 0$;
(b) $\{u^k\}$ lies in a compact region;
(c) $\|u^k - \tilde{u}\|_G^2$ is monotonically nonincreasing and thus converges.

From (a), there hold $x^k - x^{k+1} \to 0$ and $y^k - y^{k+1} \to 0$. From (b), $\{u^k\}$ has a subsequence $\{u^{k_j}\}$ that converges to $u^\star = (x^\star; y^\star)$, i.e., $x^{k_j} \to x^\star$ and $y^{k_j} \to y^\star$. Since $x^{k_j} \to x^\star$ and $y^{k_j - 1} - y^{k_j} = \gamma\beta(Ax^{k_j} - b) \to 0$, it holds that $Ax^\star = b$. By taking the limit of (B.1) over k_j and considering $x^{k_j+1} = (x^{k_j+1} - x^{k_j}) + x^{k_j} \to x^\star$, it follows that $A^*y^\star \in \partial\|x^\star\|_1$. Therefore, we have shown that $(x^\star, y^\star)$ satisfies the optimality conditions for (1.3). Since the above discussions apply to any solution $\tilde{u}$ of (1.3), by letting $\tilde{u} = (\tilde{x}, \tilde{y}) = (x^\star, y^\star)$ at the beginning and considering (c), we get the convergence of $\{u^k\}$ and thus that of $\{(x^k, y^k)\}$. ∎

Acknowledgments. We are grateful to two anonymous referees for their valuable comments and suggestions which have helped improve the paper. The first author would like to thank Prof. Bingsheng He of Nanjing University and Prof. Wotao Yin of Rice University for their helpful discussions.

REFERENCES

[1] M. V. Afonso, J. Bioucas-Dias, and M. Figueiredo, *Fast image recovery using variable splitting and constrained optimization*, IEEE Trans. Image Process., 19 (2010), pp. 2345–2356.

[2] M. V. Afonso, J. Bioucas-Dias, and M. Figueiredo, *A fast algorithm for the constrained formulation of compressive image reconstruction and other linear inverse problems*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE, Washington, DC, 2010, pp. 4034–4037.

[3] D. Baron, M. Duarte, S. Sarvotham, M. B. Wakin, and R. G. Baraniuk, *Distributed compressed sensing*, available at http://dsp.rice.edu/cs/DCS112005.pdf, 2009.
[4] J. Barzilai and J. Borwein, *Two point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
[5] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.
[6] 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, CA, 2009.
[7] J. Cai, S. Osher, and Z. Shen, *Linearized Bregman Iterations for Compressive Sensing*, UCLA CAM TR08–06, UCLA, Los Angeles, CA, 2008.
[8] J. Cai, S. Osher, and Z. Shen, *Convergence of the Linearized Bregman Iteration for ℓ_1-Norm Minimization*, UCLA CAM TR08–52, UCLA, Los Angeles, CA, 2008.
[9] E. Candès, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate information*, Commun. Pure Appl. Math., 59 (2005), pp. 1207–1233.
[10] E. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2008), pp. 717–772.
[11] 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.
[12] E. Candès and T. Tao, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Trans. Inform. Theory, 56 (2009), pp. 2053–2080.
[13] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank Sparsity Incoherence for Matrix Decomposition*, http://arxiv.org/abs/0906.2220, 2009.
[14] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.
[15] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Commun. Pure Appl. Math., 57 (2004), pp. 1413–1457.
[16] C. De Mol and M. Defrise, *A note on wavelet-based inversion algorithms*, Contemp. Math., 313 (2002), pp. 85–96.
[17] D. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.
[18] D. Donoho, *For most large underdetermined systems of linear equations, the minimal ℓ_1-norm solution is also the sparsest solution*, Commun. Pure Appl. Math., 59 (2006), pp. 907–934.
[19] J. Douglas and H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Trans. Amer. Math. Soc., 82 (1956), pp. 421–439.
[20] J. Eckstein and D. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.
[21] M. Elad, *Why simple shrinkage is still relevant for redundant representations?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5559–5569.
[22] E. Esser, *Applications of Lagrangian-Based Alternating Direction Methods and Connections to Split Bregman*, CAM report TR09–31, UCLA, Los Angeles, CA, 2009.
[23] M. Figueiredo and R. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Trans. Image Process., 12 (2003), pp. 906–916.
[24] M. Figueiredo, R. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE J. Sel. Top. Signa., 1 (2007), pp. 586–597.
[25] M. Fukushima, *Application of the alternating direction method of multipliers to separable convex programming*, Comput. Optim. Appl., 1 (1992), pp. 93–111.
[26] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Comp. Math. Appl., 2 (1976), pp. 17–40.
[27] R. Glowinski, *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, New York, 1984.
[28] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator Splitting Methods in Nonlinear Mechanics*, SIAM Stud. Appl. Math. 9, SIAM, Philadelphia, 1989.
[29] R. Glowinski and A. Marrocco, *Sur l'approximation par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, Rev. Française Automat. Informat. Rech. Opér., 9 (1975), pp. 41–76.
[30] T. Goldstein and S. Osher, *The split Bregman method for* L1*-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

[31] E. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.
[32] E. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation applied to compressed sensing: Implementation and numerical experiments*, J. Comput. Math., 28 (2010), pp. 170–194.
[33] M. Hanke, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Longman Scientific & Technical, Harlow, UK, 1995.
[34] B. He, L. Liao, D. Han, and H. Yang, *A new inexact alternating directions method for monotone variational inequalities*, Math. Progam. Ser. A, 92 (2002), pp. 103–118.
[35] B. He and H. Yang, *Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities*, Oper. Res. Lett., 23 (1998), pp. 151–161.
[36] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.
[37] C. Li, W. Yin, and Y. Zhang, *User's Guide for TVAL3: TV Minimization by Augmented Lagrangian and Alternating Direction Algorithms*, CAAM report, Rice University, Houston, TX, 2009.
[38] V. A. Morozov, *On the solution of functional equations by the method of regularization*, Soviet Math. Dokl., 7 (1966), pp. 414–417.
[39] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program. Ser. A, 103 (2005), pp. 127–152.
[40] Y. Nesterov, *Gradient methods for minimizing composite objective function*, CORE discussion paper 2007/76, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 2007.
[41] R. Nowak and M. Figueiredo, *Fast wavelet-based image deconvolution using the EM algorithm*, in Proceedings of the 35th Asilomar Conference on Signals, Systems and Computers, Vol. 1, IEEE, Washington, DC, 2001, pp. 371–375.
[42] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul., 4 (2005), pp. 460–489.
[43] G. Plonka and J. Ma, *Curvelet-wavelet regularized split Bregman iteration for compressed sensing*, Int. J. Wavelets Multiresolut. Inf. Process., to appear.
[44] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 283–298.
[45] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.
[46] R. T. Rockafellar, *The multiplier method of Hestenes and Powell applied to convex programming*, J. Optim. Theory Appl., 12 (1973), pp. 555–562.
[47] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[48] J.-L. Starck, E. Candès, and D. Donoho, *Astronomical image representation by the curvelet transform*, Astron. Astrophys., 398 (2003), pp. 785–800.
[49] J.-L. Starck, M. Nguyen, and F. Murtagh, *Wavelets and curvelets for image deconvolution: A combined approach*, Signal Process., 83 (2003), pp. 2279–2283.
[50] M. Tao, J.-F. Yang, and B. He, *Alternating direction algorithms for total variation deconvolution in image reconstruction*, available from http://www.optimization-online.org/DB_FILE/2009/11/2463.pdf., 2009.
[51] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. Roy. Statist. Soc. Ser. B., 58 (1996), pp. 267–288.
[52] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inform. Theory, 53 (2007), pp. 4655–4666.
[53] P. Tseng, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control Optim., 29 (1991), pp. 119–138.
[54] E. Van den Berg and M. Friedlander, *Probing the pareto frontier for basis pursuit solutions*, SIAM J. Sci. Comput., 31 (2008), pp. 890–912.
[55] Y. Wang, J.-F. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imaging Sci., 1 (2008), pp. 248–272.
[56] Z. Wen, W. Yin, and D. Goldfarb, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Math. Prog. Comp., 2 (2010), pp. 203–230.
[57] J. Wright and Y. Ma, *Dense error correction via ℓ_1-minimization*, IEEE Trans. Inform. Theory, to appear.
[58] S. Wright, R. Nowak, and M. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[59] J.-F. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for TVL1-L2 signal reconstruction from partial Fourier data*, IEEE J. Sel. Top. Signa., 4 (2010), pp. 288–297.

[60] 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.
[61] W. Yin, *The linearized Bregman method: Reviews, analysis, and generalizations*, CAAM TR09–02, Rice University, Houston, TX, 2009.
[62] X. Yuan and J.-F. Yang, *Sparse and low-rank matrix decomposition via alternating direction methods*, available from http://www.optimization-online.org/DB_FILE/2009/11/2447.pdf., 2009.
[63] S. Yun and K.-C. Toh, *A coordinate gradient descent method for ℓ_1-regularized convex minimization*, Comput. Optim. Appl., to appear.
[64] Y. Zhang, J.-F. Yang, and W. Yin, *Your ALgorithm for L*1, available from http://yall1.blogs.rice.edu/, 2009.