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, as well as others. We present and investigate two classes of algorithms derived from either the primal or the dual forms of the ℓ_1-problems. The construction of the algorithms consists of two main steps: (1) to reformulate an ℓ_1-problem into one having partially separable objective functions by adding new variables and constraints; and (2) to apply an exact or inexact alternating direction method to 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 each and every iteration. Convergence properties of these algorithms are established or restated when they already exist. Extensive numerical results in comparison with several state-of-the-art algorithms are given to demonstrate that the proposed algorithms are efficient, stable and robust. Moreover, we present numerical results to emphasize two practically important but perhaps overlooked points. One point is that algorithm speed should always be evaluated relative to appropriate solution accuracy; another is that whenever erroneous measurements possibly exist, the ℓ_1-norm fidelity should be the fidelity of choice in compressive sensing.

Key words. Sparse solution recovery, compressive sensing, ℓ_1-minimization, primal, dual, alternating direction method

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

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 of an underdetermined linear system. What makes such a scheme work is sparsity; i.e., the original signal must have a sparse or nearly sparse 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. 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 certain reconstruction technique. Basic CS theory presented in [7, 9, 15] states that it is extremely probable to reconstruct $\bar{x}$ accurately or even exactly from b provided that $\bar{x}$ is sufficiently sparse (or nearly sparse) 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 CS decoding process and some existing methods for the relevant optimization problems, summarize our main contributions in this paper,

*Department of Mathematics, Nanjing University, 22 Hankou Road, Nanjing, 210093, P.R. China (jfyang@nju.edu.cn)

†Department of Computational and Applied Mathematics, Rice University, 6100 Main Street, MS-134, Houston, Texas, 77005, U.S.A. (yzhang@rice.edu)

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 only concentrate 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., [1]). 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 [12]:

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

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, [16]). When b contains noise, or when $\bar{x}$ is not exactly sparse but only compressible, as are the cases 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 [12]:

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

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 ie 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., [46]) which, however, are not a subject of concern in this paper.

1.2. Some existing methods. In the last few years, quite a number of 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 basis pursuit 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 on 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.

One of the earliest first-order methods for solving (1.5) is the gradient projection method suggested in [22], 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 first-order method for solving (1.5) is the iterative shrinkage/thresholding (IST) method, which is first proposed in [21, 37, 14] for wavelet-based image deconvolution and then independently discovered and analyzed by many others [19, 43, 44, 13]. In [29, 30], 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 non-monotone line search with Barzilai-Borwein steplength [2]. A similar sparse reconstruction algorithm called SpaRSA was also studied by Wright, Nowak and Figueiredo in [51]. Recently, Beck and Teboulle proposed a fast IST algorithm (FISTA) in [3], which attains the same optimal convergence in function values as Nesterov's multi-step gradient method [36] for minimizing composite functions. Lately, Yun and Toh also studied a block coordinate gradient descent (CGD) method in [56] for solving (1.5).

There exist also algorithms for solving constrained ℓ_1-problems (1.3) and (1.4). The Bregman iteration [38] was applied to the basis pursuit problem in [53]. In the same paper, a linearized Bregman method was also suggested and analyzed subsequently in [5, 6, 54]. In [23], Friedlander and Van den Berg proposed a spectral projection gradient method (SPGL1), where (1.4) is solved by a root-finding framework applied to a sequence of LASSO problems [45]. Moreover, based on a smoothing technique studied in [35], a fast and accurate first-order algorithm called NESTA was proposed in [4] for solving (1.4).

In Section 4, we present extensive comparison results with several state-of-the-art algorithms including FPC, SpaRSA, FISTA and CGD for solving (1.5), and SPGL1 and NESTA for solving (1.3) and (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, re-discovered. Yet interestingly, until the writing of the present paper, 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 ADM technique to the area of solving ℓ_1-problems in CS and other applications, and to demonstrate its usefulness as a versatile and powerful algorithmic tool. Indeed, based on ADM technique we have derived first-order primal-dual algorithms for (1.3), (1.4), (1.5) and (1.6). Furthermore, the versatility of ADM approach has allowed us to develop a Matlab package called `YALL1` [57] that can solve eight different ℓ_1-minimization models, i.e., (1.3), (1.4), (1.5), (1.6) and their nonnegative counterparts, where local weights are also permitted in the ℓ_1-norm.

In this paper, we present extensive computational results to document the numerical performance of the proposed algorithms in comparison to several state-of-the-art algorithms for solving ℓ_1-problems under various situations. As by-products, we also address a couple of 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 a projection operator onto a 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 component-wise 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 technique 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. In Section 3, we present numerical results to compare the behavior of model (1.6) to that of models (1.4) and (1.5) under various scenarios of noise. In Section 4, we first re-emphasize the sometimes overlooked common sense on appropriate evaluations of algorithm speed, and then present extensive comparison results with several state-of-the-art algorithms to demonstrate the performance of the proposed algorithms. Finally, we conclude the paper in Section 5 and discuss several extensions of 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 on a 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 optimization problem

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

where variables x and y are separate in the objective, and 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 the Lagrangian multiplier and $\beta>0$ is a penalty parameter. The classic augmented Lagrangian method [33, 39] iterates as follows: given $\lambda^k\in\mathbb{R}^p$,

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

where $\gamma\in(0,2)$ guarantees convergence, as long as the subproblem is solved to an increasingly high accuracy at every iteration [41]. However, an accurate, joint minimization with respect to (x,y) can become costly without taking advantage of the separable form of the objective function $f(x)+g(y)$. 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

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

The basic idea of ADM goes back to the work of Glowinski and Marocco [28] and Gabay and Mercier [25]. Let $\theta_1(\cdot)$ and $\theta_2(\cdot)$ be convex functionals, and A be a continuous linear operator. The authors of [25] 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 [27], ADM is interpreted as the Douglas-Rachford splitting method [17] applied to a dual problem. The equivalence between ADM and a proximal point method is shown in [18]. ADM is also studied in convex programming [24] and variational inequalities [47, 32]. Moreover, ADM has been extended to allowing inexact subproblem minimization [18, 31].

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 [26, 27] 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. Recently, ADM has been applied to total variation based image restoration and reconstruction in [20, 52]. In the following, we apply ADM technique to (1.4) and (1.5), while the application to (1.3) and (1.6) will be a by-product.

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 into 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

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

that 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

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

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), β not included, at $x = x^k$. The solution of (2.9) is given explicitly by

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

where "$\circ$" represents component-wise multiplication. The 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:

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

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 [18] where each ADM subproblem is required to be solved more and more accurately as the algorithm proceeds. On the other hand, the analysis in [31] does cover the convergence of (2.13) but only for the case $\gamma = 1$. A more general convergence result for (2.13) is established below that allows $\gamma > 1$.

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 the Appendix. □

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

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

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

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

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

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

REMARK 1. *We point out that, when $\mu = \delta = 0$, both (2.13) and (2.16) reduce to*

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

which is an algorithm for solving the basis pursuit problem (1.3). Using similar techniques as in the proof for Theorem 2.1, we can establish the convergence of (2.17) to a solution of (1.3). The only difference between (2.17) and the linearized Bregman method proposed in [53] lies in the updating of 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., [54]).

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 in 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), which results to equally simple yet more efficient algorithms. First, we assume that the rows of A are orthonormal, i.e., A satisfies $AA^* = I$. We will remark on how to extend the treatment to a non-orthonormal matrix A at the end of this section.

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

$$\begin{aligned} &\max_{y\in\mathbb{C}^m} \min_{x\in\mathbb{C}^n, r\in\mathbb{C}^m} \left\{ \|x\|_1 + \frac{1}{2\mu}\|r\|^2 - \mathbf{Re}(y^*(Ax + r - b)) \right\} \\ &= \max_{y\in\mathbb{C}^m} \left\{ \mathbf{Re}(b^*y) - \frac{\mu}{2}\|y\|^2 + \min_{x\in\mathbb{C}^n} (\|x\|_1 - \mathbf{Re}(y^*Ax)) + \frac{1}{2\mu} \min_{r\in\mathbb{C}^m} \|r - \mu y\|^2 \right\} \\ &= \max_{y\in\mathbb{C}^m} \left\{ \mathbf{Re}(b^*y) - \frac{\mu}{2}\|y\|^2 : A^*y \in \mathbf{B}_1^\infty \right\}, \end{aligned} \tag{2.18}$$

where $\mathbf{B}_1^\infty \triangleq \{\xi \in \mathbb{C}^n : \|\xi\|_\infty \le 1\}$. By introducing $z \in \mathbb{C}^n$, (2.18) is equivalently transformed to

$$\max_{y\in\mathbb{C}^m} \left\{ f_d(y) \triangleq \mathbf{Re}(b^*y) - \frac{\mu}{2}\|y\|^2 : z - A^*y = 0, z \in \mathbf{B}_1^\infty \right\}, \tag{2.19}$$

which has an augmented Lagrangian subproblem of the form

$$\min_{y\in\mathbb{C}^m, z\in\mathbb{C}^n} \left\{ -\mathbf{Re}(b^*y) + \frac{\mu}{2}\|y\|^2 - \mathbf{Re}(x^*(z - A^*y)) + \frac{\beta}{2}\|z - A^*y\|^2, z \in \mathbf{B}_1^\infty \right\}, \tag{2.20}$$

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 (2.19), i.e., alternatingly update the multiplier (or the primal variable) $x \in \mathbb{C}^n$ and the dual variables $y \in \mathbb{C}^m$ and $z \in \mathbb{C}^n$. First, it is easy to show that, for $x = x^k$ and $y = y^k$, the minimizer z^{k+1} of (2.20) with respect to z is given explicitly by

$$z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \tag{2.21}$$

where, as in the rest of the paper, $\mathcal{P}$ represent a projection (in Euclidean norm) onto a convex set denoted as a subscript. Second, for $x = x^k$ and $z = z^{k+1}$, the minimization of (2.20) with respect to y is a least squares problem and the corresponding normal equations are

$$(\mu I + \beta AA^*)y = \beta Az^{k+1} - (Ax^k - b). \tag{2.22}$$

Under the assumption $AA^* = I$, the solution y^{k+1} of (2.22) is given by

$$y^{k+1} = \frac{\beta}{\mu + \beta}\left(Az^{k+1} - (Ax^k - b)/\beta\right). \tag{2.23}$$

Finally, we update x as follows

$$x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}), \tag{2.24}$$

where $\gamma \in (0, (\sqrt{5}+1)/2)$. Thus, the ADM scheme for (2.19) is as follows:

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = \frac{\beta}{\mu+\beta}\left(Az^{k+1} - (Ax^k - b)/\beta\right), \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}). \end{cases} \tag{2.25}$$

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

$$\max_{y\in\mathbb{C}^m}\left\{b^*y - \delta\|y\| : A^*y \in \mathbf{B}_1^\infty\right\}, \tag{2.26}$$

and produces the iteration scheme

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = \mathcal{S}\left(Az^{k+1} - (Ax^k - b)/\beta, \delta/\beta\right), \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}), \end{cases} \tag{2.27}$$

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

REMARK 2. *Under the assumption $AA^* = I$,* (2.25) *is an exact ADM in the sense that each subproblem is solved exactly. From convergence results in [26, 27], for any $\beta > 0$ and $\gamma \in (0, (\sqrt{5}+1)/2)$, the sequence $\{(x^k, y^k, z^k)\}$ generated by* (2.25) *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* (2.19)*. Similar arguments apply to* (2.27) *and the primal-dual pair* (1.4) *and* (2.26)*.*

Derived from the dual problems, we name the algorithms (2.25) and (2.27) 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

$$\max_{y\in\mathbb{C}^m}\left\{\mathbf{Re}(b^*y) : A^*y \in \mathbf{B}_1^\infty\right\}, \tag{2.28}$$

which is a special case of (2.18) and (2.26) with $\mu = \delta = 0$. Therefore, both (2.25) and (2.27) can be applied to solve (1.3). Specifically, when $\mu = \delta = 0$, both (2.25) and (2.27) reduce to

$$\begin{cases} z^{k+1} = \mathcal{P}_{B_1^\infty}(A^*y^k + x^k/\beta), \\ y^{k+1} = Az^{k+1} - (Ax^k - b)/\beta, \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^*y^{k+1}). \end{cases} \tag{2.29}$$

We note that the last equality in (2.18) holds if and only if $r = \mu y$. Therefore, the primal-dual residues and the duality gap between (2.5) and (2.19) can be defined by

$$\begin{cases} r_p \triangleq Ax + r - b \equiv Ax + \mu y - b, \\ r_d \triangleq A^* y - z, \\ \Delta \triangleq f_d(y) - f_p(x, r) \equiv \mathbf{Re}(b^* y) - \mu\|y\|^2 - \|x\|_1. \end{cases} \tag{2.30}$$

In computation, algorithm (2.25) can be terminated by

$$\text{Res} \triangleq \left\{ \|r_p\|/\|b\|,\ \|r_d\|/\sqrt{m},\ \Delta/f_p(x,r) \right\} \le \epsilon \tag{2.31}$$

where $\epsilon > 0$ measures residue in optimality.

When $AA^* \neq I$, the solution of (2.22) is costly. In this case, we take a steepest descent step in the y direction and obtain the following iteration scheme:

$$\begin{cases} z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}(A^* y^k + x^k/\beta), \\ y^{k+1} = y^k - \alpha_k^* g^k, \\ x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^* y^{k+1}), \end{cases} \tag{2.32}$$

where, at the current point (x^k, y^k, z^k), g^k and α_k^* are given by

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

In our experiments, algorithm (2.32) converges very well for random matrices where $AA^* \neq I$, although its convergence remains an issue of further research. Similar arguments apply to (2.26).

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:

$$\min_{x \in \mathbb{R}^n} \left\{ \|x\|_1 + \frac{1}{2\mu}\|Ax - b\|^2 : x \ge 0 \right\}, \tag{2.33}$$

where (A, b) can remain complex, e.g., A being a partial Fourier matrix. A similar derivation as for (2.18) shows that a dual problem of (2.33) is equivalent to

$$\max_{y \in \mathbb{C}^m} \left\{ \mathbf{Re}(b^* y) - \frac{\mu}{2}\|y\|^2 : z - A^* y = 0, z \in \mathcal{F} \right\}, \tag{2.34}$$

where $\mathcal{F} \triangleq \{z \in \mathbb{C}^n : \mathbf{Re}(z) \le 1\}$. The only difference between (2.34) and (2.19) 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 (2.34), i.e., alternately update the primal and dual variables, yields an iterative algorithm with the same updating formulae as (2.25) except the computation for z^{k+1} is replaced by

$$z^{k+1} = \mathcal{P}_{\mathcal{F}}(A^* y^k + x^k/\beta). \tag{2.35}$$

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` can be applied to solve (1.3), (1.4), (1.5), (1.6) and their nonnegative counterparts.

3. Choice of denoising models. In most practical applications, measurements are invariably contaminated by some noise. To date, the most widely used denoising models in CS include (1.5) and its variants, which are based on the ℓ_2-norm fidelity. An alternative approach is to penalize $Ax-b$ by the ℓ_1-norm, resulting the ℓ_1/ℓ_1 denoising model (1.6). Between the two types of models, which type should be preferred in general? So far, model (1.5), along with its variants, is widely used and appears to be the *de facto* model of choice. However, we provide supporting evidence to argue that perhaps in most practical situations the ℓ_1/ℓ_1 model (1.6) should be preferred unless there is an excessive amount of white noise in observed data, in which case one can only expect to obtain low-quality solutions. On the other hand, if there is any possibility that observed data may contain large measurement errors or impulsive noise, model (1.6) can potentially be dramatically better than (1.5) (see also, e.g., [50]).

We conducted a set of experiments comparing models (1.4), (1.5) with (1.6) on random problems with $n = 1000$, $m = 300$ and $k = 60$, using the solver YALL1 [57], which implements the dual ADMs described in Subsection 2.3 for eight different models, including model (1.6) that is reformulated and solved as a basis pursuit problem in the form of (1.3). The reformulation is as follows. Clearly, problem (1.6) is equivalent to

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

which, after a change of variables, can be rewritten as $\min_{\hat{x}\in\mathbb{C}^{n+m}}\{\|\hat{x}\|_1 : \hat{A}\hat{x} = \hat{b}\}$, where

$$\hat{A} = \frac{(A, \nu I)}{\sqrt{1+\nu^2}}, \quad \hat{b} = \frac{\nu}{\sqrt{1+\nu^2}} b \quad \text{and} \quad \hat{x} = \begin{pmatrix} \nu x \\ r \end{pmatrix}. \tag{3.1}$$

We note that $\hat{A}\hat{A}^* = I$ provided that $AA^* = I$. In our experiments, each model is solved for a sequence of parameter values varying from 0 to 1. The simulation of data acquisition is as follows

$$b = A\bar{x} + p \equiv A\bar{x} + p_W + p_I \equiv b_W + p_I, \tag{3.2}$$

where p, p_W, p_I represents, respectively, the total, the white and the impulsive noise, and b_W is the data containing white noise only (or noiseless whenever $p_W = 0$). White noise is generated by the `MATLAB` command `randn(n,1)` multiplied by an appropriately chosen constant to attain a desired signal-to-noise ratio, while impulsive noise values are set to be ± 1 at random positions of b that is always scaled so that $\|b\|_\infty = 1$. As is usually done, we measure the signal-to-noise ratio (SNR) of both b_W and b in terms of decibel (dB). In this paper, we use the following definition of SNR of b:

$$\mathrm{SNR}(b) = 20\log_{10}\left(\frac{\|b - \mathbf{E}(b)\|}{\|p\|}\right),$$

where $\mathbf{E}(b)$ represents the mean value of b. The SNR of b_W is defined similarly. Impulsive noise is measured by percentage, e.g., 5% impulsive noise means 5% of elements in b are erroneous. For a computed solution x from a data vector b defined in (3.2), the relative error in x is defined in terms of percentage:

$$\mathrm{RelErr}(x) = \frac{\|x - \bar{x}\|}{\|\bar{x}\|} \times 100\%. \tag{3.3}$$

Figure 3.1 presents results for $p_W = 0$ and p_I changes from 1% to 10%. From the results on the top 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 becomes increasingly severe), while model (1.4) 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). From the results on the bottom row of Figure 3.1, model (1.5) behaves similarly as (1.4), i.e., the quality of recovered signals is much worse than those from (1.6). Therefore, there should be no doubt that model (1.6) is superior to (1.4) and (1.5) provided that ν is chosen properly and should be the model of choice whenever there is a possibility of erroneous measurements in data no matter how small the percentage might be.

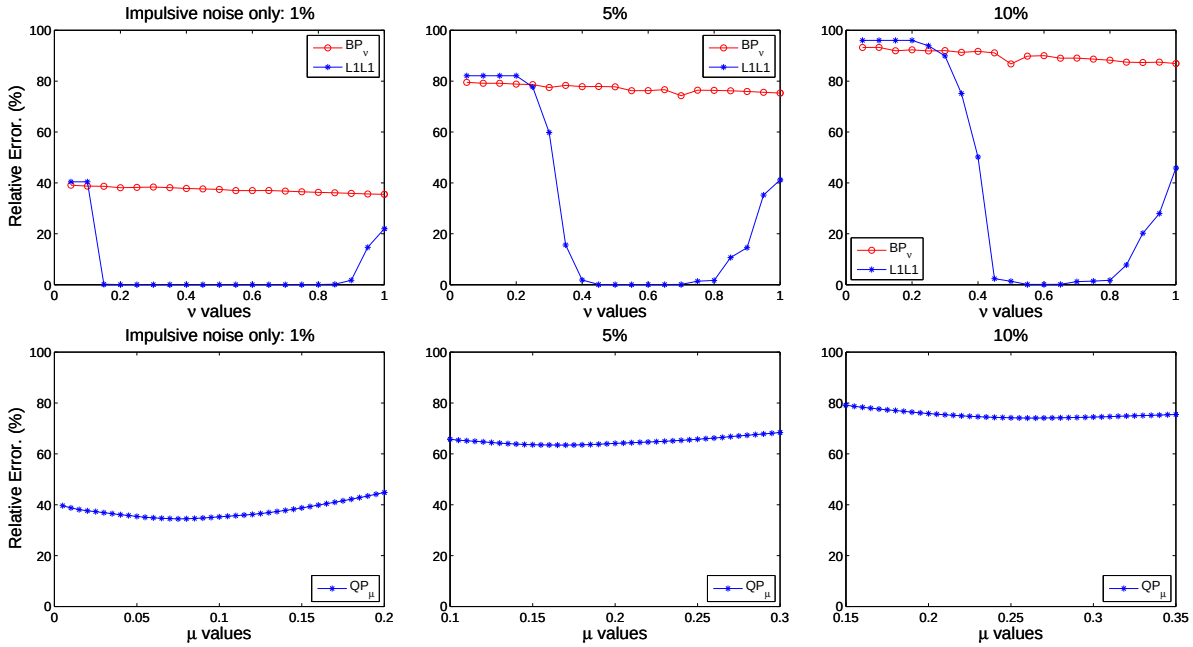

FIG. 3.1. *Recovered results from data corrupted by impulsive noise (from left to right in both rows: 1%, 5% and 10%). Top row: recovered solutions from BP_ν ($\delta \leftarrow \nu$ in (1.4)) and ℓ_1/ℓ_1 problem (1.6) (the x-axes represent ν values in both models); Bottom row: recovered solutions from QP_μ (the x-axes represent μ values in the model). In all plots, the y-axes represent relative errors of recovered solutions to the true sparse signal.*

Which model should be used when data contain both white and impulsive noise? Let us examine results given in Figure 3.2, where (1.6) is compared with (1.4) with data satisfying $\mathrm{SNR}(b_W) = 40$dB (first row) and 20dB (second row). In both cases p_I takes values of 1%, 5% and 10%. Similar to the noiseless 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).

Is model (1.6) still appropriate when there is no impulsive noise? Figure 3.3 contains results obtained from data with white noise only of $\mathrm{SNR}(b) = \mathrm{SNR}(b_W) = 30$dB, 20dB and 10dB, respectively. Qualitatively and loosely speaking, these three types of data can be characterized as good, fair and poor, respectively. As can be seen from the top left plot, on good data (1.4) offers no improvement whatsoever to the basis pursuit 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 top 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 top right plot), model (1.4) always offers better solution quality than that of model (1.6). However, for poor data the solution quality is always poor for $\nu \in [0, 1]$ (and beyond as well). At $\nu = 1$, the relative error of model (1.4) is about

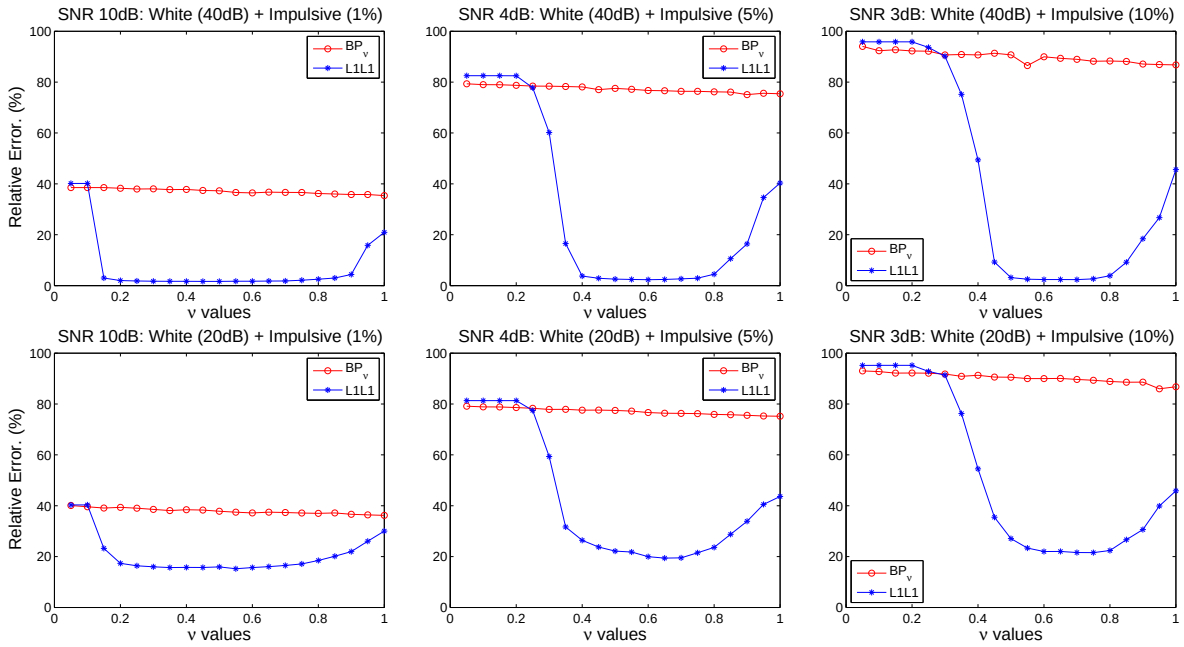

FIG. 3.2. *Recovered results from data corrupted by both white and impulsive noise. SNR of b_W is 40dB (top row) and 20dB (bottom row); In each row, the ratios of impulsive noise corruption are 1%, 5% and 10% from left to right; x-axes: model prameters in BP_ν ($\delta \leftarrow \nu$ in* (1.4)*) and ℓ_1/ℓ_1 model* (1.6)*; y-axes: relative errors of recovered solutions to the true signal.*

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, as is shown on the bottom row of Figure 3.3.

The sum of the computational evidence suggests the following three guidelines, at least for random problems of the type tested here:

1. Whenever data may contain large measurement errors, erroneous observations or in general impulsive noise, the ℓ_1-norm fidelity used by model (1.6) should naturally be preferred over the ℓ_2-norm fidelity used by model (1.4) and its variants.
2. Without impulsive noise, the ℓ_1-norm fidelity does no harm to solution quality, as long as data do not contain a large amount of white noise and ν remains reasonably small.
3. When data are contaminated by a large amount of white noise, the ℓ_2-norm fidelity should be preferred. In this case, however, high-quality recovery should not be expected regardless what model is used.

4. Numerical results. In this section, we compare the proposed ADMs for ℓ_1 problems, which will be referred as PADM and DADM, respectively, with several state-of-the-art algorithms to demonstrate their practical performance. The rest of this section is organized as follows. In Section 4.1, we present experimental 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 experimental settings, including parameter choices, stopping rules and generation of problem data under `MATLAB` environment. In Section 4.3, we compare PADM and DADM with FPC-BB [29, 30], a fixed-point continuation method with non-monotone line search based the Barzilai and Borwein (BB) steplength [2], SpaRSA [51] — a sparse reconstruction algorithm for more general regularizers, FISTA [3] — a fast IST algorithm that attains an optimal convergence rate in function values,

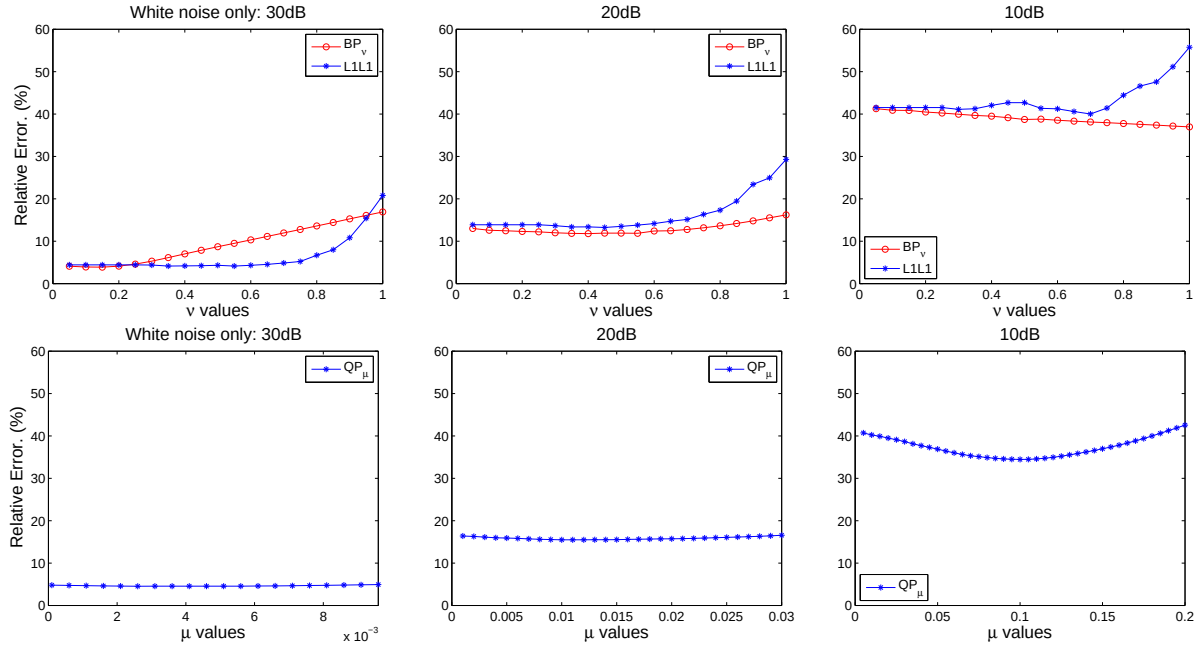

FIG. 3.3. *Recovered results from data contaminated by white noise only. In both rows, SNR of b_W: 30dB, 20dB and 10dB from left to right. Top row: results of BP_ν ($\delta \leftarrow \nu$ in* (1.4)*) and ℓ_1/ℓ_1 model* (1.6)*; Bottom row: results of QP_μ; In all plots, the x-axes represent model parameters and the y-axes represent relative errors of recovered solutions to the true signal.*

and CGD [56] — 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 [23] — a spectral projected gradient algorithm for (1.4), and NESTA [4] — a fast and accurate fast first-order algorithm based on Nesterov's smoothing technique [35]. 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, and rightfully so. However, speed is a relative concept and ought to be used within context, which unfortunately has not always been the case. In track and field, human speed is measured relative to the distance of a race. The fastest short distance runner may very well be a rather slow long distance runner and vice versa. Similarly, algorithm speed should be measured relative to accuracy. Using a single accuracy to judge algorithm speed could be as misleading as comparing the speeds of short and long distance runners.

Clearly, the definition for a good accuracy can vary with situation and application. Consequently, algorithm speed should be judged within concrete context. A relevant question here is what kind of accuracy is reasonable for solving compressive sensing problems, especially when data are noisy as is the case in most real applications. To answer this question, we solved (1.3) with noiseless data and solved (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 random Gaussian 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. The results on relative error defined in (3.3) and optimality measured in residue defined in (2.31) are given in Figure 4.1.

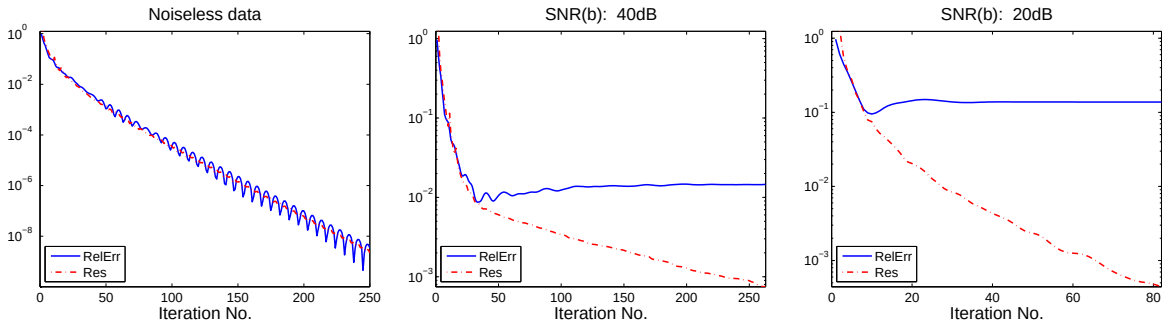

FIG. 4.1. *Relative error versus optimality for noiseless and noisy data. The x-axes represent number of iterations, and y-axes represent relative error (RelErr) defined in* (3.3) *and residue (Res) defined in* (2.31).

It is clear from Figure 4.1 that solving ℓ_1-problems to 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 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 anymore after the residue is reduced to about 10^{-2} in about 40 iterations. This phenomenon becomes more and more obvious for higher level noisy data, as is shown in the right plot of Figure 4.1. Based on these observations, we conclude that solving ℓ_1-problems to high accuracy is not necessary whenever observed data are contaminated by noise. We choose to emphasize this rather mundane point because too often such a common sense has been ignored in algorithmic studies. In our numerical comparison, whenever noisy data are used we will not compare 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. Experimental settings. Now we describe experimental settings, including selection of parameters, stopping rules and generation of data in `MATLAB`. In our experiments, we mainly tested randomized partial transform matrices as encoding matrices for reasons given below. First, partial transform matrices do not need explicit storage and thus are suitable for large scale tests. Second, fast matrix-vector multiplications are available, which are the main computation load of all algorithms to be compared with. Finally, for this class of matrices there hold $AA^* = I$, which is not only useful in parameter choices in PADM but also required in DADM for exact minimization of subproblems. We note that condition $AA^* = I$ implies $\lambda_{\max}(A^*A) = 1$. Therefore, for convergence of PADM, in our experiments we set $\tau = 0.8$, $\gamma = 1.199$ and $\beta = 2m/\|b\|_1$ in (2.13) and (2.16), which work quite well in practice, although suitably larger τ and γ seem work better 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 especially for noisy data. Thus, when comparing with other algorithms, we simply terminated PADM and DADM when relative change of two consecutive iterates becomes 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 (2.31), are possible. Parametric settings of FPC-BB, SpaRSA, FISTA, CGD, SPGL1 and NESTA will be specified when we make comparison with them. 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, `xbar` represents a sparse signal that we wish to recover, and `sigma` is the standard deviation of additive Gaussian noise. The partial Walsh-Hadamard transform is implemented in the 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 k (the number of nonzero components in `xbar`). We initialized x to the zero vector in 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 [29, 30], SpaRSA [51], FISTA [3] and CGD [56], all of which were developed in the last two years for solving (1.5). In the comparison, we used random Gaussian spikes as `xbar`, i.e., the location of nonzeros are selected uniformly at random while the values of nonzero components are *iid* Gaussian entries. The standard deviation of additive noise is set to be 10^{-3}, while model parameter μ in (1.5) is set to be either 10^{-3} or 10^{-4}, both of which are suitable values for small noise. We aimed to compare both the efficiency and robustness of the algorithms.

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 PADM and DADM for a prescribed number of iterations, and choose proper stopping tolerance values for other algorithms so that their iteration numbers are roughly euqal to the prescribed iteration number. 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.

For the purpose of clarity, we first compare PADM and DADM with FPC-BB and SpaRSA, and then with FISTA and CGD. Since FPC-BB implements continuation on regularization parameter but not on stopping tolerance, we set all parameters as default except 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 be 20 and terminated it when relative change in function values falls below 10^{-7}. Since the per-iteration cost is roughly two matrix-vector multiplications for all compared algorithms, it is helpful to examine the decreasing behavior of relative errors and function values as functions of iteration numbers. Figure 4.2 presents the results of two different combinations of m and k. Each result is the average of 50 runs on randomly generated data.

As can be seen from Figure 4.2 that, compared with FPC-BB and SpaRSA, PADM and DADM usually decrease relative errors and function values faster. Specifically, the relative error and function value curves of both PADM and DADM fall below those of FPC-BB and SpaRSA almost throughout the entire iteration process. With no more than 100 iterations, PADM and DADM reached lowest relative errors and then started to increase, which is a problem of the model (1.5) rather than the algorithm since function values keep decreasing. It is also clear that all algorithms attain nearly equal relative errors and function values.

Next we compare DADM with FISTA and CGD. 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

$$y^k = \begin{cases} x^0, & \text{if } k = 0; \\ x^k + \frac{t_{k-1}-1}{t_k}(x^k - x^{k-1}), & \text{otherwise,} \end{cases} \quad \text{where} \quad t_k = \begin{cases} 1, & \text{if } k = 0; \\ \frac{1+\sqrt{1+4t_{k-1}^2}}{2}, & \text{otherwise.} \end{cases}$$

It is shown in [3] that FISTA attains an optimal convergence rate $O(1/k^2)$ in decreasing function values,

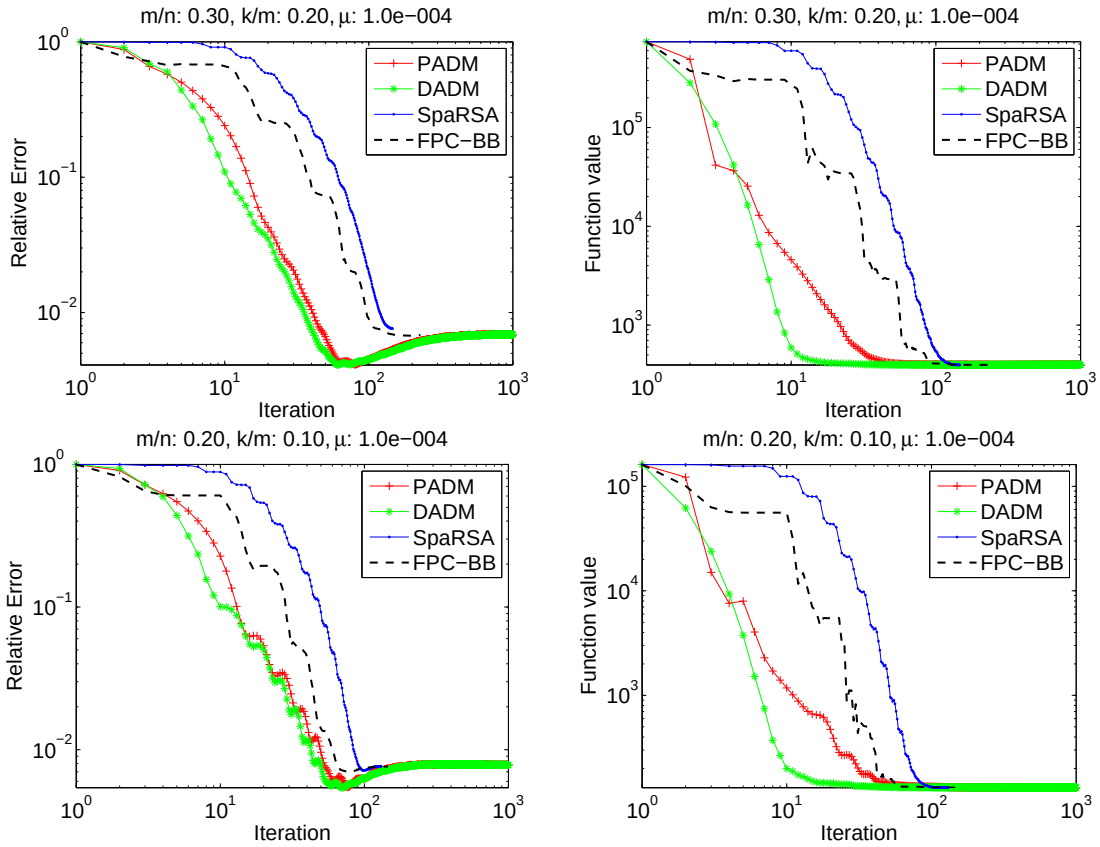

FIG. 4.2. *Comparison results of PADM, DADM, FPC-BB and SpaRSA on* (1.5). *The x-axes represent number of iterations, and y-axes represent relative errors (plots on the left) and function values (plots on the right). The standard deviation of Gaussian noise is* `sigma` $= 10^{-3}$. *The results are average of 50 runs.*

where k is the iteration counter. By letting $t_k \equiv 1$, FISTA reduces to the well-known primary iterative-soft-thresholding (IST) algorithm. Figure 4.3 shows the comparison results of DADM and FISTA, along with those of the primary IST algorithm in order to illustrate the advantage of FISTA over IST (after a small modification). Since the relative performance of PADM and DADM on problem (1.5) has been illustrated in Figure 4.2 and the dual approach seems to be slightly more efficient than the primal one, here we merely presented the results of DADM for simplicity. In this experiment, we set $\tau = 1$ for both FISTA and IST and initialized all algorithms at $x = A^*b$ rather than at the origin in order to make function values fall into a relatively small range.

It can be seen from Figure 4.3 that FISTA converges much faster than IST. However, without heuristic techniques such as continuation and line search as incorporated in many algorithms, FISTA is generally slower than DADM. The slow convergence of FISTA and IST as compared with DADM becomes more pronounced when μ becomes smaller in which case (1.5) becomes more difficult. From Figure 4.3, when $\mu = 10^{-3}$, FISTA converges to a nearly optimal function value within no more than 200 iterations, while IST consumes more than 1000 iterations. When μ is decreased from 10^{-3} to 10^{-4}, both FISTA and IST become slower, while the convergence of DADM is not affected.

Similarly, the comparison results of DADM with CGD are given in Figure 4.4, along with those of FPC-BB in order to evaluate the relative performance of CGD. In this experiment, we used the continuation variant of CGD (the code `CGD_cont` in the `MATLAB` package of CGD) and set all parameters as default except

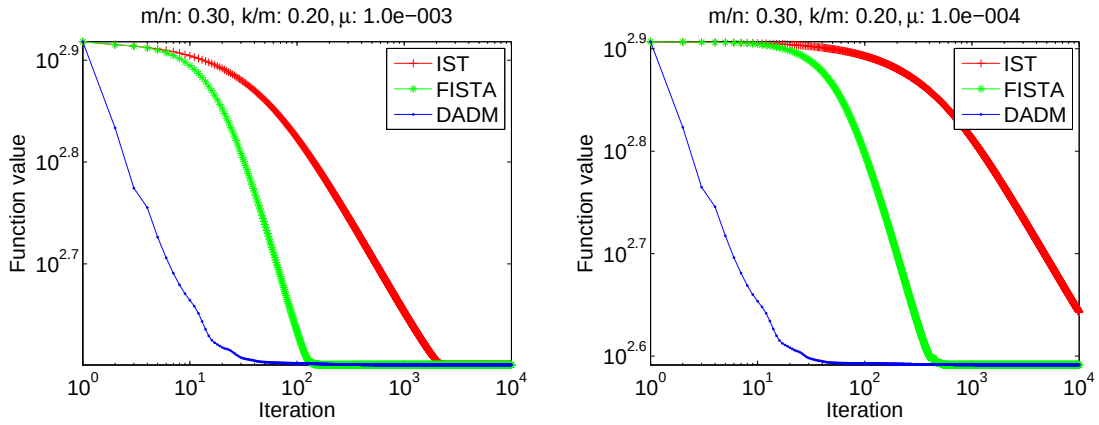

FIG. 4.3. *Comparison results of IST, FISTA and DADM on* (1.5). *The x-axes represent number of iterations, and y-axes represent function values, both in logarithmic scale. The standard deviation of Gaussian noise is* 10^{-3}. *Left plot:* $\mu = 10^{-3}$; *Right plot:* $\mu = 10^{-4}$. *The results are average of 50 runs.*

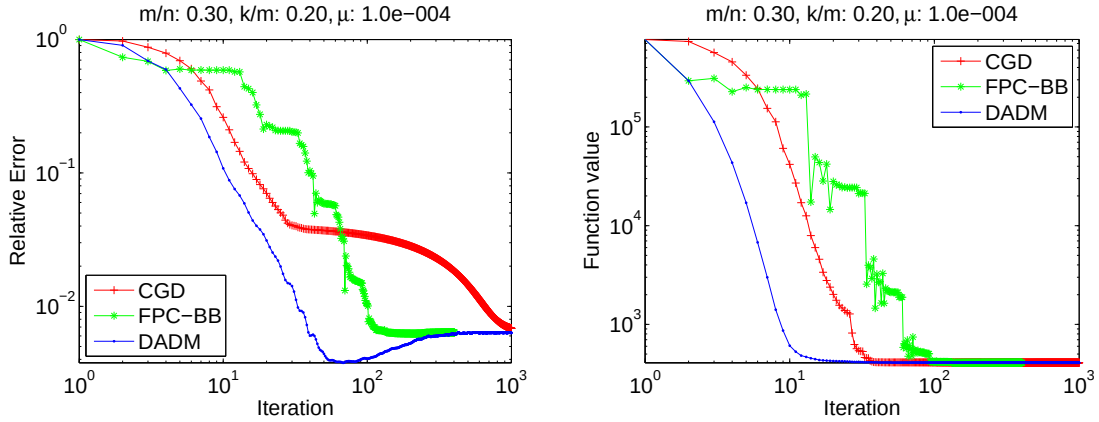

FIG. 4.4. *Comparison results of FPC-BB, CGD and DADM on* (1.5). *The x-axes represent number of iterations, and y-axes represent relative errors (plots on the left) and function values (plots on the right), both in logarithmic scale. The standard deviation of Gaussian noise is* 10^{-3}. *The results are average of 50 runs.*

setting the initial μ value to be $\max(0.01\|A^\top x^0\|_\infty, 2\mu)$ which works better than the default setting in our tests when μ is small. It can be seen from the plot on the right of Figure 4.4 that CGD decreases the function value faster than FPC-BB. However, from the plot on the left, CGD does not necessarily decrease the relative error to the true signal faster than FPC-BB. In comparison, DADM converges faster in both the function value and relative error.

We experimented on various combinations of (m, k) with noisy data and observed similar phenomenon. As is the case in Figures 4.2 and 4.4, the relative error produced by the ADM algorithms tends to eventually increase, after the initial decrease when problem (1.5) is solved to a high accuracy. This implies, as suggested in Section 4.1, that it is unnecessary to solve ℓ_1-problems to a higher accuracy than what is warranted by the accuracy of the underlying data. Therefore, in the next set of tests we terminate PADM and DADM whenever (4.1) is satisfied with $\epsilon = 5 \times 10^{-4}$ instead of using much smaller tolerances.

Next we compare PADM and DADM with FPC-BB and SpaRSA for various combinations of (m, k). Here we do not present results for FISTA and CGD because they have been found to be less competitive in this set of tests. As is already mentioned earlier, without heuristic continuation and line search techniques,

FISTA is slower than ADM algorithms. On the other hand, CGD is slower in terms of decreasing relative error. 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}. For each fixed pair (m, k), 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.

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	k/m	Iter	RelErr	Iter	RelErr	Iter	RelErr	Iter	RelErr
0.3	0.1	67.4	3.93E-3	55.6	3.96E-3	103.3	5.70E-3	55.1	5.88E-3
0.3	0.2	76.4	4.03E-3	68.3	4.09E-3	139.4	7.12E-3	93.1	7.37E-3
0.2	0.1	102.2	6.34E-3	100.4	6.17E-3	114.2	7.47E-3	69.8	7.56E-3
0.2	0.2	103.3	7.90E-3	96.2	7.72E-3	178.4	1.59E-2	121.3	2.32E-2
0.1	0.1	154.2	1.23E-2	167.5	1.22E-2	136.8	1.30E-2	83.5	1.37E-2
0.1	0.2	252.4	1.09E-1	239.4	1.10E-1	210.2	1.65E-1	126.2	2.00E-1

As can be seen from Table 4.1, in all cases PADM and DADM obtained smaller relative errors in comparable numbers of iterations as FPC-BB and SpaRSA. This is particularly true for more difficult problems, e.g., for $m/n = 0.1$ and $k/m = 0.2$ where the resulting relative errors of PADM and DADM are considerably smaller than those of FPC-BB and SpaRSA. We also tried to terminated SpaRSA and FPC-BB using more stringent stopping rules. Specifically, we set `xtol`=10^{-5} and `gtol`=0.02 in FPC-BB and terminated SpaRSA when relative change in function values falls below 10^{-7}. The relative error results either remain roughly the same as those presented in Table 4.1 or were just slightly better, while the iteration numbers required by both FPC-BB and SpaRSA were increased from around 50% to more than 100%. We also tested partial DCT and sparse signals of different dynamic ranges and observed similar phenomenon.

4.4. Comparison with SPGL1 and NESTA. In this subsection, we compare PADM and DADM with SPGL1 and NESTA for solving (1.4). The same as before, `xbar` is random Gaussian spikes, and the standard deviation `sigma` of additive noise is 10^{-3}. 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 PADM and DADM to a prescribed number of iterations and terminated SPGL1 and NESTA with suitably chosen tolerances so that the iteration numbers consumed are roughly identical. Specifically, we set `optTol` $= 10^{-8}$ in SPGL1, which is a tolerance for optimality, and `TolVar` $= 10^{-8}$ in NESTA, which is a tolerance in relative change in objective function. All other parameters are set to their default values. Figure 4.5 presents average results of 50 random problems, where two combinations of m and k are used. The resulting relative error and fidelity residue (i.e., $\|Ax - b\|$) are plotted as functions of iterations.

As can be seen from the left column of Figure 4.5, compared with SPGL1 and NESTA, both PADM and DADM attained smaller relative errors throughout most of the iteration process (with the exception at the very beginning). With no more than 100 iterations, both PADM and DADM reached lowest relative errors and then started to increase slightly. It is interesting to note that, as can be seen from the right column of Figure 4.5, NESTA is the fastest in terms of decreasing fidelity residue but slowest in terms of decreasing the relative error. In the applications of compressive sensing, we are interested in recovering the true signal, which means that the smaller the relative error is, the better.

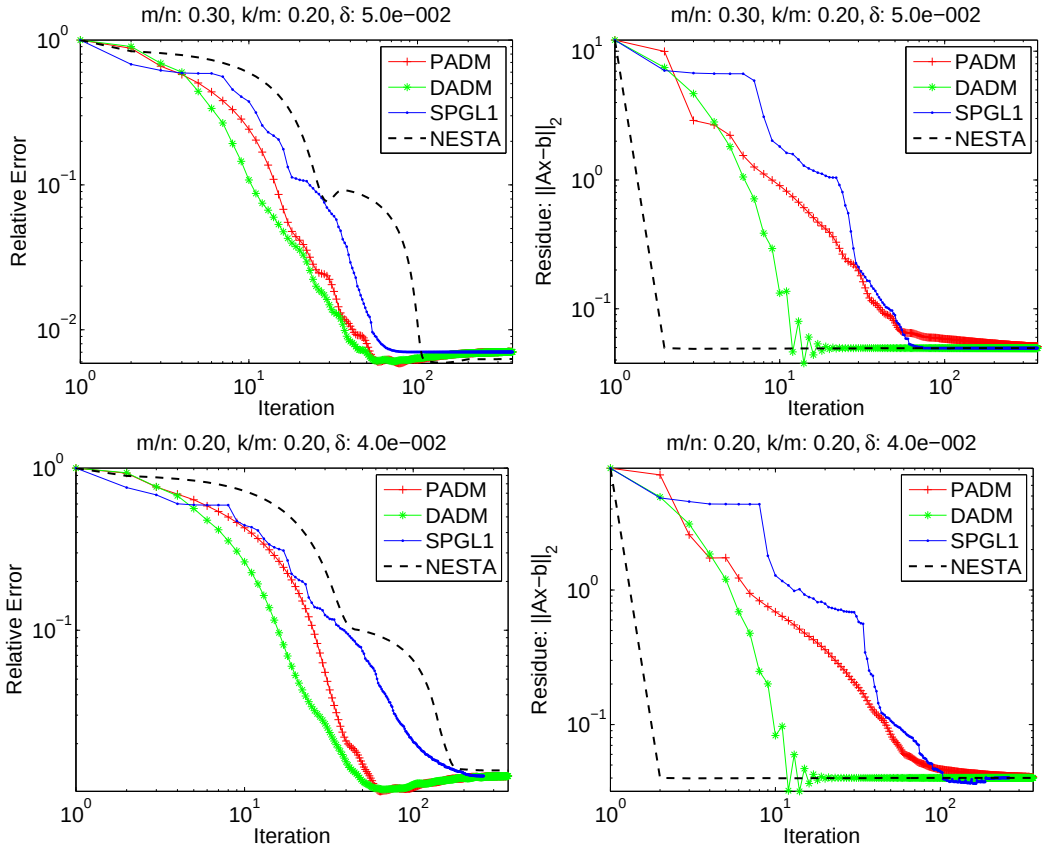

FIG. 4.5. *Comparison results of PADM, DADM, SPGL1 and NESTA on* (1.4). *The x-axes represent number of iterations; y-axes represent relative error (plots on the left) and residue (plots on the right). The standard deviation of Gaussian noise is* 10^{-3}. *The results are average of 50 runs.*

After extensive experiments under various scenarios, including different sensing matrices and sparse signals of different dynamic ranges, we have found that when data are noisy the proposed ADM algorithms decrease relative error faster than all other tested algorithms.

In the second set of tests, we terminated PADM and DADM by (4.1) with $\epsilon = 5 \times 10^{-4}$. 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 before, matrix-vector multiplications are the main computational load for all compared algorithms. The number of matrix-vector multiplications consumed by PADM and DADM is two times the number of iterations. However, the number used by SPGL1 is not always proportional to the number of iterations. As for NESTA, its per-iteration cost is also two matrix-vector multiplications for partial orthonormal matrices, although this number increases to six for general matrices. Therefore, instead of iteration numbers, we present in Table 4.2 the number of matrix-vector multiplications, denoted by #AAt that includes both `A*x` and `A'*y`.

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.

4.5. Comparison with SPGL1 on the basis pursuit problem. In this subsection, we compare DADM with SPGL1 on the basis pursuit problem (1.3), in which case b is noiseless and solving problems

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	k/m	#AAt	RelErr	#AAt	RelErr	#AAt	RelErr	#AAt	RelErr
0.3	0.1	107.9	6.40E-3	97.8	6.35E-3	120.8	5.39E-3	300.6	5.83E-3
0.3	0.2	118.1	6.05E-3	108.2	6.26E-3	160.1	7.26E-3	303.6	8.33E-3
0.2	0.1	124.9	8.01E-3	112.7	7.92E-3	149.7	7.35E-3	316.9	6.85E-3
0.2	0.2	122.6	1.05E-2	110.0	1.05E-2	168.2	1.62E-2	311.5	6.41E-2
0.1	0.1	167.3	1.29E-2	152.2	1.26E-2	171.9	1.29E-2	340.6	1.73E-2
0.1	0.2	161.2	9.86E-2	145.0	1.06E-1	184.0	1.49E-1	326.4	2.96E-1

TABLE 4.3
Comparison results on (1.3) *(b is noiseless; stopping rule:* $\epsilon = 10^{-6}$ *in* (4.1)*; average of 50 runs).*

n = 8192		DADM				SPGL1			
m/n	k/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

to a higher accuracy should result in 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.

As can be observed from previous experimental results, the proposed ADM algorithms may slow down after a fast convergence stage at the beginning. When measurements are free of noise and a highly accurate solution is demanded, the ADM algorithms can sometimes be slower than SPGL1. Table 4.3 shows that DADM is slower than SPGL1 in three of the five test cases in terms of CPU seconds while getting slightly lower accuracy (the results for $m/n = 0.1$ and $k/m = 0.2$ are omitted since both algorithms failed to recover an accurate solution). We note that since SPGL1 requires some non-trivial calculations other than matrix-vector multiplications, a larger #AAt number by DADM does not necessarily lead to a longer 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 geometrically.

4.6. Summary. We provided supporting evidence to emphasize the important point that algorithm speed should be evaluated relative to solution accuracy. Since more often than not measurements are noisy applications, solving ℓ_1-problems to a very high accuracy is generally not warranted and unnecessary. It is more practically relevant to evaluate the speed of an algorithm based on how fast it achieves an appropriate accuracy that is consistent with noise levels in data.

We presented extensive experimental results on ℓ_1-problems and compared the proposed first-order, primal-dual algorithms, derived from the ADM approach, with state-of-the-art algorithms FPC-BB, SpaRSA,

FISTA, CGD, SPGL1 and NESTA. Our numerical results show that on the tested cases the proposed ADM algorithms are efficient and stable. In particular, under practically relevant conditions where data contain noise and stopping tolerances are set to appropriate values (where a more stringent tolerance would not lead to a more accurate solution), the proposed ADM algorithms generally achieve lower relative errors within fewer or comparable number of iterations in comparison to other tested algorithms.

The experimental 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 ones are inexact which solve some subproblem approximately. The dual-based ADMs have been implemented in a `MATLAB` package called YALL1 [57] (short for Your ALgorithm for L1), which is applicable to eight different ℓ_1-models including (1.3), (1.4), (1.5), (1.6) and the corresponding nonnegative counterparts.

5. Concluding remarks. We proposed to solve ℓ_1-problems arising from compressive sensing by first-order, primal-dual algorithms derived from the Alternating Direction Method (ADM) framework which is based on the classic augmented Lagrangian function and alternating minimization idea for structured optimization. This ADM approach is applicable to numerious ℓ_1-problems including (1.3), (1.4), (1.5), (1.6) and their nonnegative counterparts, as well as others. 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 competitively with several state-of-the-art algorithms. On various classes of test problems with synthetic, noisy data, the proposed ADM algorithms have unmistakably exhibited the following advantages over competing algorithms in the comparison: (i) they converge relatively fast without the help of a continuation or a line search technique; (ii) their performance is relatively insensitive to changes in model and algorithmic parameters; and (iii) they demonstrate a notable ability to quickly decrease 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 most effective in obtaining the best achievable level of accuracy whenever data contain a nontrivial level of noise, which is the case most relevant to practical applications.

The most influential impact of the ADM approach is perhaps its great versatility and its seemingly universal effectiveness in a wide array 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 [40, 8, 10], or the total variation (TV) regularization [42] widely used in image processing. While the nuclear-norm is just an extension of ℓ_1-norm to the matrix case, the TV regularization can be converted to ℓ_1-regularization after introducing a splitting variable [48, 52]. 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 total variation based image reconstruction in [20, 52, 34] and to semi-definite programming in [49]. 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 [11]). An ADM scheme has been proposed and studied for this problem in [55].

Although the ADM approach is classic and its convergence properties have been well studied, its surprising 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, that should be interesting for further investigations.

Acknowledgments. The first author would like to thank Prof. Bingsheng He of Nanjing University and Dr. Wotao Yin of Rice University for helpful discussions.

REFERENCES

[1] 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.

[2] J. Barzilai, and J. Borwein, *Two point step size gradient methods*, IMA J Numer. Anal., vol. 8, pp. 141–148, 1988.

[3] A. Beck, and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imag. Sci., vol. 2, no. 1, pp. 183–202, 2009.

[4] S. Becker, J. Bobin, and E. Candès, *NESTA: A fast and accurate first-order method for sparse recovery*, Technical Report, California Institute of Technology, April, 2009.

[5] J. Cai, S. Osher, and Z. Shen, *Linearized Bregman iterations for compressive sensing*, UCLA CAM TR08–06.

[6] J. Cai, S. Osher, and Z. Shen, *Convergence of the linearized Bregman iteration for ℓ_1-norm minimization*, UCLA CAM TR08–52.

[7] E. Candès, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate information*, Commun. Pure Appl. Math., vol. 59, pp. 1207-1233, 2005.

[8] E. Candés and B. Recht, *Exact matrix completion via convex optimization*, Submitted, 2008.

[9] E. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, vol. 52, no. 2, pp. 489–509, 2006.

[10] E. Candés and T. Tao, *The power of convex relaxation: Near-optimal matrix completion*, Submitted, 2009.

[11] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank Sparsity incoherence for matrix decomposition*, `http://arxiv.org/abs/0906.2220`.

[12] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., vol. 20, pp. 33–61, 1998.

[13] I. Daubechies, M. Defriese and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Commun. Pure Appl. Math., vol. LVII, pp. 1413–1457, 2004.

[14] C. De Mol, and M. Defrise, *A note on wavelet-based inversion algorithms*, Contemp. Math., 313, pp. 85–96, 2002.

[15] D. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, vol. 52, no. 4, pp. 1289–1306, 2006.

[16] 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., vol. 59, no. 7, pp. 907–934, 2006.

[17] J. Douglas, and H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Trans. Am. Math. Soc., vol. 82, pp. 421439, 1956.

[18] J. Eckstein, and D. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., vol. 55, pp. 293–318, 1992.

[19] M. Elad, *Why simple shrinkage is still relevant for redundant representations?* IEEE Trans. Inform. Theory, vol. 52, no. 12, pp. 5559–5569, 2006.

[20] E. Esser, *Applications of Lagrangian-Based Alternating Direction Methods and Connections to Split Bregman*, CAM Report TR09-31, UCLA, 2009.

[21] M. Figueiredo and R. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Trans. Imag. Process., vol. 12, no. 8, pp. 906–916, 2003.

[22] 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., vol. 1, pp. 586–597, 2007.

[23] M. Friedlander, and E. Van den Berg, *Probing the pareto frontier for basis pursuit solutions*, SIAM J. Sci. Comput., vol. 31, no. 2, pp. 890–912, 2008.

[24] M. Fukushima, *Application of the alternating direction method of multipliers to separable convex programming*, Comput. Optim. Appl., vol. 1, pp. 93–111, 1992.

[25] D. Gabay, and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Comp. Math. Appl., vol. 2, pp. 17-40, 1976.

[26] R. Glowinski, *Numerical methods for nonlinear variational problems*, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1984.

[27] R. Glowinski, and P. Le Tallec, *Augmented Lagrangian and operator splitting methods in nonlinear mechanics*, SIAM 1989.

[28] R. Glowinski, and A. Marrocco, *Sur lapproximation par elements finis dordre un, et la resolution par penalisation-dualite*

dune classe de problemes de Dirichlet nonlineaires, Rev. Francaise dAut. Inf. Rech. Oper., R-2, pp. 41-76, 1975.

[29] E. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: methodology and convergence*, SIAM J. Optim., vol. 19, no. 3, pp. 1107–1130, 2008.

[30] E. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation applied to compressed sensing: implementation and numerical experiments*, J Comput. Math., to appear.

[31] B. He, L. Liao, D. Han, and H. Yang, *A new inexact alternating directions method for monotone variational inequalities*, Math. Progam., Ser. A, vol. 92, pp. 103–118, 2002.

[32] B. He, and H. Yang, *Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities*, Oper. Res. Lett., vol. 23, pp. 151–161, 1998.

[33] M. R. Hestenes, *Multiplier and gradient methods*, J. Optimiz. Theory App., vol. 4, pp. 303–320, 1969.

[34] C. Li, W. Yin, and Y. Zhang, *Users Guide for TVAL3: TV Minimization by Augmented Lagrangian and Alternating Direction Algorithms*, CAAM Report, 2009.

[35] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program., Serie A, vol. 103, pp. 127–152, 2005.

[36] Y. Nesterov, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 2007/76, 2007.

[37] 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, pp. 371–375, 2001.

[38] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterated regularization method for total variation-based image restoration*, Multiscale Model. Sim., vol. 4, pp. 460–489, 2005.

[39] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, NY, pp. 283–298, 1969.

[40] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization*, Submitted to SIAM Rev., 2007.

[41] R. T. Rockafellar, *The multiplier method of Hestenes and Powell applied to convex programming*, J. Optimiz. Theory App., vol. 12, pp. 555–562, 1973.

[42] L. Rudin, S. Osher and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, vol. 60, pp. 259–268, 1992.

[43] J.-L. Starck, E. Candès, and D. Donoho, *Astronomical image representation by the curvelet transform*, Astron. Astrophys., vol. 398, pp. 785–800, 2003.

[44] J.-L. Starck, M. Nguyen, and F. Murtagh, *Wavelets and curvelets for image deconvolution: a combined approach*, Signal Process., vol. 83, pp. 2279–2283, 2003.

[45] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. Roy. Statist. Soc., Ser. B., vol. 58, pp. 267-288, 1996.

[46] J. A. Tropp, and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inform. Theory, vol. 53, no. 12, pp. 4655-4666, 2007.

[47] P. Tseng, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control Optim., vol. 29, no. 1, pp. 119–138, 1991.

[48] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alteranting minimization alrorithm for total variation image reconstruction*, SIAM J. Imag. Sci., vol. 1, pp. 948–951, 2008.

[49] Z. Wen, W. Yin, and D. Goldfarb, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Technical Report, IEOR, Columbia University, 2009.

[50] J. Wright, and Y. Ma, *Dense error correction via ℓ_1-minimization*, submitted to IEEE Trans. Inform. theory, 2008.

[51] S. Wright, R. Nowak, and M. Figueiredo, *Sparse reconstruction by separable approximation*, in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, October 2008.

[52] J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for TVL1-L2 signal reconstruction from partial Fourier data*, submitted to IEEE J. Sel. Top. Signa., Special Issue on Compressive Sensing, 2009.

[53] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imag. Sci., vol. 1, no. 1, pp. 143–168, 2008.

[54] W. Yin, *The linearized Bregman method: reviews, analysis, and generalizations*, CAAM TR09–02, Rice University.

[55] X. Yuan, and J. Yang, *Sparse and low-rank matrix decomposition via alternating direction methods*, submitted, 2009.

[56] S. Yun, and K-C Toh, *A coordinate gradient descent method for ℓ_1-regularized convex minimization*, submitted, 2009.

[57] Y. Zhang, Your ALgorithm for L1, Available at: `http://www.caam.rice.edu/~yzhang/YALL1/`

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 $\hat{y} \triangleq y^k - \beta(A\hat{x} + \hat{r} - b)$. As such, $y^{k+1} = y^k - \gamma(y^k - \hat{y})$. From the definition of $\hat{r}$, $\hat{x}$ and $\hat{y}$, equation (2.7) can be reformulated 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

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

Similarly, equation (2.11) is equivalent to $A^*\hat{y} - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x}) \in \partial\|\hat{x}\|_1$. Further considering $A^*\tilde{y} \in \partial\|\tilde{x}\|_1$ and the convexity of $\|\cdot\|_1$, it can be shown that

$$(\hat{x} - \tilde{x})^* \left(A^*(\hat{y} - \tilde{y}) - \beta A^*A(x^k - \hat{x}) + \frac{\beta}{\tau}(x^k - \hat{x})\right) \geq 0. \tag{A.3}$$

From $A\tilde{x} + \tilde{r} = b$ and $\beta(A\hat{x} + \hat{r} - b) = y^k - \hat{y}$, the addition of (A.2) and (A.3) gives

$$\frac{1}{\beta}(\hat{y} - \tilde{y})^*(y^k - \hat{y}) + \frac{\beta}{\tau}(\hat{x} - \tilde{x})^*(x^k - \hat{x}) \geq (y^k - \hat{y})^*A(x^k - \hat{x}). \tag{A.4}$$

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

$$G_0 = \begin{pmatrix} I_n & \\ & \gamma I_m \end{pmatrix},\ G_1 = \begin{pmatrix} \frac{\beta}{\tau} I_n & \\ & \frac{1}{\beta} I_m \end{pmatrix},\ G = \begin{pmatrix} \frac{\beta}{\tau} I_n & \\ & \frac{1}{\beta\gamma} I_m \end{pmatrix} \text{ and } u = \begin{pmatrix} x \\ y \end{pmatrix}. \tag{A.5}$$

By using this notation and considering equality $\hat{u} - \tilde{u} = (\hat{u} - u^k) + (u^k - \tilde{u})$, (A.4) implies

$$(u^k - \tilde{u})^* G_1 (u^k - \hat{u}) \geq \|u^k - \hat{u}\|_{G_1}^2 + (y^k - \hat{y})^* A(x^k - \hat{x}). \tag{A.6}$$

The iteration of u in (2.13) can be written as $u^{k+1} = u^k - G_0(u^k - \hat{u})$, from which it can be shown

$$\begin{aligned} \|u^k - \tilde{u}\|_G^2 - \|u^{k+1} - \tilde{u}\|_G^2 &= 2(u^k - \tilde{u})^* G_1 (u^k - \hat{u}) - \|G_0(u^k - \hat{u})\|_G^2 \\ \text{(from (A.6))} \ &\geq 2\|u^k - \hat{u}\|_{G_1}^2 + 2(y^k - \hat{y})^* A(x^k - \hat{x}) - \|u^k - \hat{u}\|_{G_0 G G_0}^2 \\ \text{(from (A.5))} \ &= \frac{\beta}{\tau}\|x^k - \hat{x}\|^2 + \frac{2 - \gamma}{\beta}\|y^k - \hat{y}\|^2 + 2(y^k - \hat{y})^* A(x^k - \hat{x}) \end{aligned} \tag{A.7}$$

From condition $\tau\lambda_{\max} + \gamma < 2$, it holds that $\delta \triangleq 1 - \tau\lambda_{\max}/(2 - \gamma) > 0$. Let $\rho \triangleq (2 - \gamma)/(\beta + \beta\delta) > 0$, inequality (A.7) implies

$$\begin{aligned} \|u^k - \tilde{u}\|_G^2 - \|u^{k+1} - \tilde{u}\|_G^2 &\geq \frac{\beta}{\tau}\|x^k - \hat{x}\|^2 + \frac{2 - \gamma}{\beta}\|y^k - \hat{y}\|^2 - \rho\|y^k - \hat{y}\|^2 - \frac{1}{\rho}\|A(x^k - \hat{x})\|^2 \\ &\geq \left(\frac{\beta}{\tau} - \frac{\lambda_{\max}}{\rho}\right)\|x^k - \hat{x}\|^2 + \left(\frac{2 - \gamma}{\beta} - \rho\right)\|y^k - \hat{y}\|^2 \\ \text{(from definitions of } \delta \text{ and } \rho) \ &= \frac{\beta\delta^2}{\tau}\|x^k - \hat{x}\|^2 + \frac{2 - \gamma}{\beta}\frac{\delta}{1 + \delta}\|y^k - \hat{y}\|^2 \\ \text{(from definitions of } \hat{x}, \hat{y} \text{ and } G) \ &= \eta\|u^k - u^{k+1}\|_G^2, \end{aligned} \tag{A.8}$$

where $\eta \triangleq \min\left(\delta^2, \frac{\delta(2-\gamma)}{\gamma(1+\delta)}\right) > 0$. It follows from (A.8) that

(a) $\|u^k - u^{k+1}\|_G$ converges to 0, and thus $\lim_{k\to\infty} Ax^k + r^k = b$;

(b) $\{u^k\}$ lies in a compact region;

(c) $\|u^k - \tilde{u}\|_G^2$ is monotonically non-increasing and thus converges.

From (b), $\{u^k\}$ has a subsequence $\{u^{k_j}\}$ converges to $u^\star = (x^\star; y^\star)$. From $y^{k_j} \to y^\star$, $\lim_{k\to\infty} Ax^k + r^k = b$ and the iterative formula for y, we have $y^k \to y^\star$. From (c), $\|u^k - \tilde{u}\|_G^2 \to \|u^\star - \tilde{u}\|_G^2$, which, by further considering $y^k \to y^\star$, implies that any limit point of $\{x^k\}$, if more than one, must have an equal distance to $\tilde{x}$. For convenience, we let $z(x, y) \triangleq x - \tau A^*(Ax - b - y/\beta)/(1 + \mu\beta)$. By eliminating r^{k+1}, the second equation in (2.13) becomes $x^{k+1} = \text{Shrink}(z(x^k, y^k), \tau/\beta)$, which implies

$$x^{k_j+1} = \text{Shrink}\left(z(x^{k_j}, y^{k_j}), \tau/\beta\right) \to \text{Shrink}\left(z(x^\star, y^\star), \tau/\beta\right) \triangleq x^{\star\star}.$$

Thus, $x^{\star\star}$ is also a limit point of $\{x^k\}$ and must have an equal distance to $\tilde{x}$ as $x^\star$ does, i.e.,

$$\|x^\star - \tilde{x}\| = \|x^{\star\star} - \tilde{x}\| = \|\text{Shrink}\left(z(x^\star, y^\star), \tau/\beta\right) - \text{Shrink}\left(z(\tilde{x}, \tilde{y}), \tau/\beta\right)\|, \tag{A.9}$$

where the second equality is because $\tilde{x} = \text{Shrink}\left(z(\tilde{x}, \tilde{y}), \tau/\beta\right)$. From the property of $\text{Shrink}(\cdot, \tau/\beta)$, equality (A.9) implies (see e.g., [29])

$$x^\star - \tilde{x} = \text{Shrink}\left(z(x^\star, y^\star), \tau/\beta\right) - \text{Shrink}\left(z(\tilde{x}, \tilde{y}), \tau/\beta\right) = \text{Shrink}\left(z(x^\star, y^\star), \tau/\beta\right) - \tilde{x}.$$

Thus, $x^\star = \text{Shrink}\left(z(x^\star, y^\star), \tau/\beta\right)$. By letting $r^\star = b - Ax^\star$, it is easy to show from the above discussions that $(r^\star, x^\star, y^\star)$ is a solution to (2.5). By letting $\tilde{u} = (\tilde{x}, \tilde{y}) = (x^\star, y^\star) = u^\star$ and $\tilde{r} = r^\star$ at the beginning, we get the convergence of $\{u^k\}$ and thus that of $\{r^k, x^k, y^k\}$. □