FAST ALTERNATING LINEARIZATION METHODS FOR MINIMIZING THE SUM OF TWO CONVEX FUNCTIONS

DONALD GOLDFARB*, SHIQIAN MA†, AND KATYA SCHEINBERG‡

Revised February 9, 2012

Abstract. We present in this paper alternating linearization algorithms based on an alternating direction augmented Lagrangian approach for minimizing the sum of two convex functions. Our basic methods require at most $O(1/\epsilon)$ iterations to obtain an ϵ-optimal solution, while our accelerated (i.e., fast) versions of them require at most $O(1/\sqrt{\epsilon})$ iterations, with little change in the computational effort required at each iteration. For both types of methods, we present one algorithm that requires both functions to be smooth with Lipschitz continuous gradients and one algorithm that needs only one of the functions to be so. Algorithms in this paper are Gauss-Seidel type methods, in contrast to the ones proposed by Goldfarb and Ma in [22] where the algorithms are Jacobi type methods. Numerical results are reported to support our theoretical conclusions and demonstrate the practical potential of our algorithms.

Key words. Convex Optimization, Variable Splitting, Alternating Linearization Method, Alternating Direction Method, Augmented Lagrangian Method, Optimal Gradient Method, Gauss-Seidel Method, Peaceman-Rachford Method

AMS subject classifications. Primary, 65K05; Secondary, 68Q25, 90C25

1. Introduction.

In this paper, we are interested in the following convex optimization problem:

$$\min_{x\in\mathbb{R}^n} F(x) \equiv f(x) + g(x) \tag{1.1}$$

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are both convex functions such that the following two problems are easy to solve for any $\mu > 0$ and $z \in \mathbb{R}^n$ relative to minimizing $F(x)$:

$$\min_{x\in\mathbb{R}^n} \left\{ f(x) + \frac{1}{2\mu}\|x - z\|^2 \right\} \tag{1.2}$$

and

$$\min_{x\in\mathbb{R}^n} \left\{ g(x) + \frac{1}{2\mu}\|x - z\|^2 \right\}. \tag{1.3}$$

Note that when $f(\cdot)$ ($g(\cdot)$, resp.) is a function of a matrix $X \in \mathbb{R}^{m\times n}$, Problem (1.2) ((1.3), resp.) is associated with a matrix $Z \in \mathbb{R}^{m\times n}$ instead of $z \in \mathbb{R}^n$ and the $\|\cdot\|$ norm is the Frobenius norm rather than the Euclidean norm. In particular, we are specially interested in cases where solving (1.2) (or (1.3)) takes roughly the same effort as computing the gradient (or a subgradient) of $f(x)$ (or $g(x)$, respectively). Problems of this type arise in many applications of practical interest. The following are some interesting examples.

Example 1. ℓ_1 minimization in compressed sensing (CS). Signal recovery problems in compressed sensing [9, 14] use the ℓ_1 norm $\|x\|_1 := \sum_{i=1}^n |x_i|$ as a regularization term to promote sparsity in the solution

*Department of Industrial Engineering and Operations Research, Columbia University, New York, 10027, USA. Email: goldfarb@columbia.edu. Research supported in part by NSF Grants DMS 06-06712 and DMS 10-16571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER25856.

†Institute for Mathematics and Its Applications, 400 Lind Hall, 207 Church Street SE, University of Minnesota, Minneapolis, MN 55455, USA. Email: maxxa007@ima.umn.edu

‡Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA. Email: katyas@lehigh.edu

$x \in \mathbb{R}^n$ of a linear system $Ax = b$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. This results in the unconstrained problem

$$\min_{x\in\mathbb{R}^n} \left\{ \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1 \right\}, \tag{1.4}$$

where $\rho > 0$, which is of the form of (1.1) with $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$. In this case, the two problems (1.2) and (1.3) are easy to solve. Specifically, (1.2) reduces to solving a linear system and (1.3) reduces to the vector shrinkage operation $x = T_{\mu\rho}(z)$, where componentwise

$$T_{\mu\rho}(z)_i = \mathrm{sgn}(z_i)(|z_i| - \mu\rho)_+, \forall i, \tag{1.5}$$

which requires $O(n)$ operations (see e.g., [26]). Depending on the size and structure of A, solving the system of linear equations required by (1.2) may be more expensive, less expensive or comparable to computing the gradient $A^\top(Ax-b)$ of $f(x)$. In the application we consider in Section 4, these computations are comparable due to the special structure of A.

Example 2. Nuclear norm minimization (NNM). The nuclear norm minimization problem, which seeks a low-rank solution of a linear system, can be cast as

$$\min_{X\in\mathbb{R}^{m\times n}} \left\{ \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \rho\|X\|_* \right\}, \tag{1.6}$$

where $\rho > 0$, $X \in \mathbb{R}^{m\times n}$, $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ is a linear operator, $b \in \mathbb{R}^p$ and the nuclear norm $\|X\|_*$ is defined as the sum of the singular values of the matrix X. Problem (1.6) and a special case of it, the so-called matrix completion problem, have many applications in optimal control, online recommendation systems, computer vision, etc. (see e.g., [41, 8, 10, 30]). In Problem (1.6), if we let $f(X) = \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2$ and $g(X) = \rho\|X\|_*$, then Problem (1.2) reduces to solving a linear system. The solution of Problem (1.3) is given by the matrix shrinkage operation (see e.g., [33, 23]), $X = U\mathrm{Diag}(\bar{\sigma})V^\top$, where $U\mathrm{Diag}(\sigma)V^\top$ is the singular value decomposition of Z and $\bar{\sigma}_i = (\sigma_i - \mu\rho)_+$, for all i. Note that the matrix shrinkage operation is comparable in cost to computing a subgradient of $\|X\|_*$, or the gradient of the smoothed version of this function, which also require an SVD.

Example 3. Robust principal component analysis (RPCA). The RPCA problem seeks to recover a low-rank matrix X from a corrupted matrix M. This problem has many applications in computer vision, image processing and web data ranking (see e.g., [7]), and can be formulated as

$$\min\{\|X\|_* + \rho\|Y\|_1 : X + Y = M\}, \tag{1.7}$$

where $\rho > 0$, $M \in \mathbb{R}^{m\times n}$ and the ℓ_1 norm $\|Y\|_1 := \sum_{i,j} |Y_{ij}|$. Note that by defining $f(X) = \|X\|_*$ and $g(X) = \rho\|M - X\|_1$, (1.7) can be rewritten in the form of (1.1). Moreover, the two problems (1.2) and (1.3) corresponding to (1.7) have solutions given respectively by a matrix shrinkage operation and a vector shrinkage operation.

Example 4. Sparse inverse covariance selection (SICS). Gaussian graphical models are of great interest in statistical learning. Because conditional independence between different nodes correspond to zero entries in the inverse covariance matrix of the Gaussian distribution, one can learn the structure of the graph by estimating a sparse inverse covariance matrix from sample data by solving the following maximum

likelihood problem with an ℓ_1-regularization term, (see e.g., [51, 19, 48, 4]).

$$\max \left\{ \log\det(X) - \langle \Sigma, X\rangle - \rho\|X\|_1 \right\},$$

or equivalently,

$$\min \left\{ -\log\det(X) + \langle \Sigma, X\rangle + \rho\|X\|_1 \right\}, \tag{1.8}$$

where $\rho > 0$ and $\Sigma \in S_+^n$ (the set of symmetric positive semidefinite matrices) is the given sample covariance matrix. Note that by defining $f(X) := -\log\det(X) + \langle \Sigma, X\rangle$ and $g(X) := \rho\|X\|_1$, (1.8) is of the form of (1.1). Moreover, the solution to Problem (1.2), a comparable effort to computing the gradient of $f(X)$, is given by $X = U\mathrm{Diag}(\bar{\sigma})U^\top$, where $U\mathrm{Diag}(\sigma)U^\top$ is the spectral decomposition of $\mu\Sigma - Z$ and $\bar{\sigma}_i = (-\sigma_i + \sqrt{\sigma_i^2 + 4\mu})/2$ for all i. The solution of Problem (1.3) corresponds to a vector shrinkage operation. (See [42] for details.)

Algorithms for solving Problem (1.1) have been studied extensively in the literature. For large-scale problems, for which problems (1.2) and (1.3) are relatively easy to solve, the class of alternating direction methods that are based on variable splitting combined with the augmented Lagrangian method are particularly important. In these methods, one splits the variable x into two variables, i.e., one introduces a new variable y and rewrites Problem (1.1) as

$$\min\{f(x) + g(y) : x - y = 0\}. \tag{1.9}$$

Since Problem (1.9) is an equality constrained problem, the augmented Lagrangian method can be used to solve it. Given a penalty parameter $1/\mu$, at the k-th iteration, the augmented Lagrangian method minimizes the augmented Lagrangian function

$$\mathcal{L}_\mu(x, y; \lambda) := f(x) + g(y) - \langle \lambda, x - y\rangle + \frac{1}{2\mu}\|x - y\|^2, \tag{1.10}$$

with respect to x and y, i.e., it solves the subproblem

$$(x^k, y^k) := \arg\min_{x,y} \mathcal{L}_\mu(x, y; \lambda^k) \tag{1.11}$$

and then updates the Lagrange multiplier λ^k via:

$$\lambda^{k+1} := \lambda^k - \frac{1}{\mu}(x^k - y^k). \tag{1.12}$$

Minimizing $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y jointly is often not easy. In fact, it generally is not any easier than solving the original Problem (1.1). However, if one minimizes $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y alternatingly, one needs to solve problems of the form (1.2) and (1.3), which as we have already discussed, is often easy to do. Such an alternating direction augmented Lagrangian method (ADAL) for solving (1.9) is given below as Algorithm 1.

The history of alternating direction methods (ADMs) goes back to the 1950s for solving PDEs [15, 40] and to the 1970s for solving variational problems associated with PDEs [20, 21]. ADMs have also been applied to solving variational inequality problems by Tseng [46, 45] and He et al. [27, 29]. Recently, with the emergence

Algorithm 1: Alternating Direction Augmented Lagrangian Method (ADAL)

Choose μ, λ^0 and $x^0 = y^0$.
for $k = 0, 1, \cdots$ **do**
  $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$
  $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^k)$
  $\lambda^{k+1} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1})$

of compressed sensing and subsequent great interest in ℓ_1 minimization [9, 14], ADMs have been applied to ℓ_1 and total variation regularized problems arising from signal processing and image processing. The papers of Goldstein and Osher [25], Afonso et al. [1] and Yang and Zhang [50] are based on the alternating direction augmented Lagrangian framework (Algorithm 1), and demonstrate that ADMs are very efficient for solving ℓ_1 and TV regularized problems. The work of Yuan [52] and Yuan and Yang [53] showed that ADMs can also efficiently solve ℓ_1-regularized problems arising from statistics and data analysis. More recently, Wen, Goldfarb and Yin [49] and Malick et al. [34] applied alternating direction augmented Lagrangian methods to solve semidefinite programming (SDP) problems. The results in [49] show that these methods greatly outperform interior point methods on several classes of well-structured SDP problems. Furthermore, He et al. proposed an alternating direction based contraction method for solving separable linearly constrained convex problems [28].

Another important and related class of algorithms for solving (1.1) is based on operator splitting. The aim of these algorithms is to find an x such that

$$0 \in T_1(x) + T_2(x), \tag{1.13}$$

where T_1 and T_2 are maximal monotone operators. This is a more general problem than (1.1) and ADMs for it have been the focus of a substantial amount of research; e.g., see [12, 16, 11, 17, 13, 32, 43]. Since the first-order optimality conditions for (1.1) are:

$$0 \in \partial f(x) + \partial g(x), \tag{1.14}$$

where $\partial f(x)$ denotes the subdifferential of $f(x)$ at the point x, a solution to Problem (1.1) can be obtained by solving Problem (1.14). For example, see [16, 32, 43] and references therein for more information on this class of algorithms.

While global convergence results for various splitting and alternating direction algorithms have been established under appropriate conditions, our interest here is on iteration complexity bounds for such algorithms. By an iteration complexity bound we mean a bound on the number of iterations needed to obtain an ϵ-optimal solution which is defined as follows.

DEFINITION 1.1. *$x_\epsilon \in \mathbb{R}^n$ is called an ϵ-optimal solution to* (1.1) *if $F(x_\epsilon) - F(x^*) \leq \epsilon$, where x^* is an optimal solution to* (1.1).

Complexity bounds for first-order methods for solving convex optimization problems have been given by Nesterov and many others. In [36, 37], Nesterov gave first-order algorithms for solving smooth unconstrained convex minimization problems with an iteration complexity of $O(\sqrt{L/\epsilon})$, where L is the Lipschitz constant of the gradient of the objective function, and showed that this is the best complexity that is obtainable when only first-order information is used. These methods can be viewed as accelerated gradient methods

where a combination of past iterates is used to compute the next iterate. Similar techniques were then applied to nonsmooth problems [38, 47, 5, 39] and corresponding optimal complexity results were obtained. The ISTA (Iterative Shrinkage/Thresholding Algorithm) and FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) algorithms proposed by Beck and Teboulle in [5] are designed for solving (1.1) when one of the functions (say $f(x)$) is smooth and the other is not. It is proved in [5] that the number of iterations required by ISTA and FISTA to get an ϵ-optimal solution to Problem (1.1) are respectively $O(L(f)/\epsilon)$ and $O(\sqrt{L(f)/\epsilon})$, under the assumption that $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant $L(f)$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L(f)\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

ISTA computes a sequence $\{x^k\}$ via the iteration

$$x^{k+1} := \arg\min_x Q_\mu^f(x, x^k), \tag{1.15}$$

where

$$Q_\mu^f(u, v) := g(u) + f(v) + \langle \nabla f(v), u - v\rangle + \frac{1}{2\mu}\|u - v\|^2, \tag{1.16}$$

while FISTA computes $\{x^k\}$ via the iteration

$$\left\{ \begin{array}{lcl} x^k & := & \arg\min_x Q_\mu^f(x, y^k) \\ t_{k+1} & := & \left(1 + \sqrt{1 + 4t_k^2}\right)/2 \\ y^{k+1} & := & x^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x^k - x^{k-1}) \end{array} \right. \tag{1.17}$$

starting with $t_1 = 1$, $y^1 = x^0 \in \mathbb{R}^n$ and $k = 1$.

Note that ISTA and FISTA treat the functions $f(x)$ and $g(x)$ very differently. At each iteration they both linearize the function $f(x)$ but never directly minimize it, while they do minimize the function $g(x)$ in conjunction with the linearization of $f(x)$ and a proximal (penalty) term. These two methods have proved to be efficient for solving the CS Problem (1.4) (see e.g., [5, 26]) and the NNM Problem (1.6) (see e.g., [33, 44]). ISTA and FISTA work well in these areas because $f(x)$ is quadratic and is well approximated by linearization. However, for the RPCA Problem (1.7) where two complicated functions are involved, ISTA and FISTA do not work well. For the SICS Problem (1.8), intermediate iterates X^k may not be positive definite, and hence the gradient of $f(X) = -\log\det(X) + \langle \Sigma, X\rangle$ may not be well defined at X^k. Therefore, ISTA and FISTA cannot be used to solve the SICS Problem (1.8). In [42], it is shown that SICS problems can be very efficiently solved by our ADM approach.

Our contribution. In this paper, we propose both basic and accelerated (fast) versions of alternating linearization methods (ALMs) based on an alternating direction augmented Lagrangian approach for solving (1.1), and analyze their iteration complexities. Our basic methods require at most $O(L/\epsilon)$ iterations to obtain an ϵ-optimal solution, while our fast methods require at most $O(\sqrt{L/\epsilon})$ iterations with only a very small increase in the computational effort required at each iteration, where L is a constant that depends on the Lipschitz constants of the functions involved. Thus, our fast methods have the same iteration complexity as *optimal* first-order methods. For both types of methods, we present an algorithm that requires both functions to be continuously differentiable with Lipschitz constants for the gradients denoted by $L(f)$ and $L(g)$. In this case $L = L(f)L(g)/(L(f) + L(g))$. We also present for each type of method, an algorithm

that only needs one of the functions, say $f(x)$, to be smooth, in which case $L = L(f)$[1] The algorithms that require both functions to be smooth are related to the multiple splitting algorithms in a recent paper by Goldfarb and Ma [22]. The algorithms in [22] are Jacobi type methods since they do not use information from the current iteration to solve succeeding subproblems in that iteration, while the algorithms proposed in this paper are Gauss-Seidel type methods since information from the current iteration is used later in the same iteration. These algorithms can also be viewed as extensions of the ISTA and FISTA algorithms in [5]. The complexity bounds we obtain for our algorithms are similar to those in [5].

At each iteration, our algorithms alternatively minimize two different approximations to the original objective function, obtained by keeping one function unchanged and linearizing the other one. Our basic algorithm is similar in many ways to the alternating linearization method proposed by Kiwiel et al. [31]. In particular, the approximate functions minimized at each step of Algorithm 3.1 in [31] have the same form as those minimized in our algorithm. However, our basic algorithm differs from the one in [31] in the way that the proximal terms are chosen, and our accelerated algorithms are very different. Moreover, no complexity bounds have been given for the algorithm in [31]. To the best of our knowledge, the complexity results in this paper are the first ones that have been given for a Gauss-Seidel type alternating direction method [2]. Complexity results for related Jacobi type alternating direction methods are given in [22].

Organization. The rest of this paper is organized as follows. In Sections 2 and 3 we propose our alternating linearization methods based on alternating direction augmented Lagrangian methods and give convergence/complexity bounds for them. We compare the performance of our ALMs to other competing algorithms using an image deblurring problem in Section 4. Finally, we offer some conclusion in Section 5.

2. Alternating Linearization Methods. In each iteration of the ADAL method, Algorithm 1, the Lagrange multiplier λ is updated just once, immediately after the augmented Lagrangian is minimized with respect to y. It seems natural to also update λ after solving the subproblem with respect to x. By doing this, we get a symmetric version of the ADAL method. This algorithm is given below as Algorithm 2.

Algorithm 2: Symmetric Alternating Direction Augmented Lagrangian Method (SADAL)

1 Choose μ, λ^0 and $x^0 = y^0$.
2 **for** $k = 0, 1, \cdots$ **do**
3 $\quad x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$
4 $\quad \lambda^{k+\frac{1}{2}} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^k)$
5 $\quad y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}})$
6 $\quad \lambda^{k+1} := \lambda^{k+\frac{1}{2}} - \frac{1}{\mu}(x^{k+1} - y^{k+1})$

This ADAL variant is described and analyzed in [21]. Moreover, it is shown in [21] that Algorithms 1 and 2 are equivalent to the Douglas-Rachford [15] and Peaceman-Rachford [40] methods, respectively, applied to the optimality condition (1.14) for Problem (1.1). If we assume that both $f(x)$ and $g(x)$ are differentiable, it follows from the first-order optimality conditions for the two subproblems in lines 3 and 5 of Algorithm 2 that

$$\lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1}) \quad \text{and} \quad \lambda^{k+1} = -\nabla g(y^{k+1}). \tag{2.1}$$

[1] Note that if $L(g) \to \infty$ then $L(f)L(g)/(L(f) + L(g)) \to L(f)$.

[2] After completion of an earlier version of the present paper, which is available on http://arxiv.org/abs/0912.4571, Monteiro and Svaiter [35] gave an iteration complexity bound to achieve a desired closeness of the current iterate to the solution, rather than for obtaining ϵ optimality, for ADMs for solving the more general Problem (1.13).

Substituting (2.1) into Algorithm 2, we get the following alternating linearization method (ALM) which is equivalent to the SADAL method (Algorithm 2) when both f and g are differentiable. In Algorithm 3,

Algorithm 3: Alternating Linearization Method (ALM)

1 Choose μ and $x^0 = y^0$.
2 **for** $k = 0, 1, \cdots$ **do**
3 $\quad x^{k+1} := \arg\min_x Q^g_\mu(x, y^k)$
4 $\quad y^{k+1} := \arg\min_y Q^f_\mu(y, x^{k+1})$

$Q_f(u, v)$ is defined by (1.16) and

$$Q^g_\mu(u, v) := f(u) + g(v) + \langle \nabla g(v), u - v\rangle + \frac{1}{2\mu}\|u - v\|_2^2. \tag{2.2}$$

In Algorithm 3, we alternatively replace the functions g and f by their linearizations plus a proximal regularization term to get an approximation to the original function F.

A drawback of Algorithm 3 is that it requires both f and g to be continuously differentiable. In many applications, however, one of these functions is nonsmooth, as in the examples given in Section 1. Although Algorithm 2 can be applied when $f(x)$ and $g(x)$ are nonsmooth, we are unable to provide a complexity bound in this case without smoothing. However, when only one of the functions of f and g is nonsmooth (say g is nonsmooth), the following variant of Algorithm 3 applies, and for this algorithm, we have a complexity result. To give a more general algorithm, we allow different penalty parameters μ to be used in the two subproblems. Note that these μ act like step lengths for the steps $x^{k+1} - y^k$ and $y^{k+1} - x^{k+1}$ that are taken.

Algorithm 4: Alternating Linearization Method with Skipping Steps (ALM-S)

1 Choose $\mu_f > 0, \mu_g > 0$, λ^0 and $x^0 = y^0$.
2 **for** $k = 0, 1, \cdots$ **do**
3 $\quad x^{k+1} := \arg\min_x \mathcal{L}_{\mu_g}(x, y^k; \lambda^k)$
4 $\quad$ **If** $F(x^{k+1}) > \mathcal{L}_{\mu_g}(x^{k+1}, y^k; \lambda^k)$, **then** $x^{k+1} := y^k$
5 $\quad y^{k+1} := \arg\min_y Q^f_{\mu_f}(y, x^{k+1})$
6 $\quad \lambda^{k+1} := \nabla f(x^{k+1}) - (x^{k+1} - y^{k+1})/\mu_f$

We call Algorithm 4, ALM with skipping steps (ALM-S) because in line 4 of Algorithm 4, if

$$F(x^{k+1}) > \mathcal{L}_{\mu_g}(x^{k+1}, y^k; \lambda^k) \tag{2.3}$$

holds, we let $x^{k+1} := y^k$, i.e., we do not set x^{k+1} to the value computed in line 3. As we shall see in the proof of Theorem 2.3 below, $F(x^{k+1}) \le \mathcal{L}_{\mu_g}(x^{k+1}, y^k; \lambda^k)$ is necessary for obtaining the iteration-complexity bounds for Algorithm 4. An alternative version of Algorithm 4 that in general requires less work when the computation of x^{k+1} is not skipped is the following Algorithm 5.

Note that in Algorithm 5, when (2.3) does not hold, we update λ instead of computing $\nabla f(x^{k+1})$ as is done in the SADAL algorithm. Algorithm 5 is usually faster than Algorithm 4 when computing $\nabla f(x^{k+1})$ is costly. Note also that when (2.3) holds, then the steps of the algorithm reduce to those of ISTA, and when (2.3) does not hold, then the steps of the algorithm reduce to those of ALM. Thus, ALM-S is a hybrid of ISTA and ALM.

Algorithm 5: Alternating Linearization Method with Skipping Steps (equivalent version)

1 Choose $\mu_f > 0, \mu_g > 0$, λ^0 and $x^0 = y^0$.
2 **for** $k = 0, 1, \cdots$ **do**
3 $\quad x^{k+1} := \arg\min_x \mathcal{L}_{\mu_g}(x, y^k; \lambda^k)$
4 $\quad$ **if** $F(x^{k+1}) > \mathcal{L}_{\mu_g}(x^{k+1}, y^k; \lambda^k)$ **then**
5 $\quad\quad x^{k+1} := y^k$
6 $\quad\quad y^{k+1} := \arg\min_y Q^f_{\mu_f}(y, x^{k+1})$
7 $\quad\quad \lambda^{k+1} := \nabla f(x^{k+1}) - (x^{k+1} - y^{k+1})/\mu_f$
8 $\quad$ **else**
9 $\quad\quad \lambda^{k+\frac{1}{2}} := \lambda^k - (x^{k+1} - y^k)/\mu_g$
10 $\quad\quad y^{k+1} := \arg\min_y \mathcal{L}_{\mu_f}(x^{k+1}, y; \lambda^{k+\frac{1}{2}})$
11 $\quad\quad \lambda^{k+1} := \lambda^{k+\frac{1}{2}} - (x^{k+1} - y^{k+1})/\mu_f$

The following theorem gives conditions under which Algorithms 2, 3, 4 and 5 are equivalent.

THEOREM 2.1. *(i) If both f and g are differentiable, and λ_0 is set to $-\nabla g(y^0)$, then Algorithms 2 and 3 are equivalent. (ii) If in addition g is Lipschitz continuous with Lipschitz constant $L(g)$, and $\mu = \mu_f = \mu_g \leq 1/L(g)$, then Algorithms 3 and 4 are equivalent. (iii) If f is differentiable, then Algorithms 4 and 5 are equivalent.*

Proof. When both f and g are differentiable and $\lambda_0 = -\nabla g(y^0)$, (2.1) holds for all $k \geq 0$, and it follows that $\mathcal{L}_\mu(x, y^k; \lambda^k) \equiv Q^g_\mu(x, y^k)$ and $\mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}}) \equiv Q^f_\mu(y, x^{k+1})$. This proves part (i). If $\nabla g(x)$ is Lipschitz continuous and $\mu_g \leq 1/L(g)$,

$$g(x^{k+1}) \leq g(y^k) + \langle \nabla g(y^k), x^{k+1} - y^k \rangle + \frac{1}{2\mu_g} \left\| x^{k+1} - y^k \right\|_2^2,$$

holds (see e.g., [6]). This implies that (2.3) does not hold and hence, $x^{k+1} := \arg\min_x \mathcal{L}_{\mu_g}(x, y^k; \lambda^k)$, and the equivalence of Algorithms 3 and 4 follows. This proves part (ii). The optimality of x^{k+1} in line 3 of Algorithm 5 implies that $\lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1})$ when (2.3) does not hold and hence that $\mathcal{L}_{\mu_f}(x^{k+1}, y; \lambda^{k+\frac{1}{2}}) \equiv Q^f_{\mu_f}(y, x^{k+1})$. This proves part (iii). □

We show in the following that the iteration complexities of Algorithms 3 and 4 for appropriate choices of μ and μ_f and μ_g are both $O(1/\epsilon)$ for obtaining an ϵ-optimal solution for (1.1). First, we need the following generalization of Lemma 2.3 in [5].

LEMMA 2.2. *Let $\psi : \mathbb{R}^n \to \mathbb{R}$ and $\phi : \mathbb{R}^n \to \mathbb{R}$ be convex functions and for a given point $v \in \mathbb{R}^n$, define*

$$Q^\psi_\mu(u, v) := \phi(u) + \psi(v) + \langle \gamma_\psi, u - v \rangle + \frac{1}{2\mu} \|u - v\|_2^2,$$

and

$$p_\psi(v) := \arg\min_u Q^\psi_\mu(u, v), \tag{2.4}$$

where γ_ψ is any subgradient of ψ at v. Let $\Phi(\cdot) = \phi(\cdot) + \psi(\cdot)$. If

$$\Phi(p_\psi(v)) \leq Q^\psi_\mu(p_\psi(v), v), \tag{2.5}$$

then for any u,

$$2\mu(\Phi(u) - \Phi(p_\psi(v))) \geq \|p_\psi(v) - u\|^2 - \|v - u\|^2. \tag{2.6}$$

Proof. From (2.5), we have

$$\begin{aligned} \Phi(u) - \Phi(p_\psi(v)) &\geq \Phi(u) - Q_\mu^\psi(p_\psi(v), v) \\ &= \Phi(u) - \left(\phi(p_\psi(v)) + \psi(v) + \langle \gamma_\psi, p_\psi(v) - v\rangle + \tfrac{1}{2\mu}\|p_\psi(v) - v\|_2^2\right). \end{aligned} \tag{2.7}$$

Since ϕ and ψ are convex we have

$$\phi(u) \geq \phi(p_\psi(v)) + \langle \gamma_\phi, u - p_\psi(v)\rangle, \tag{2.8}$$

and

$$\psi(u) \geq \psi(v) + \langle \gamma_\psi, u - v\rangle, \tag{2.9}$$

where γ_ϕ is a subgradient of ϕ at $p_\psi(v)$. Summing (2.8) and (2.9) yields

$$\Phi(u) \geq \phi(p_\psi(v)) + \psi(v) + \langle \gamma_\phi, u - p_\psi(v)\rangle + \langle \gamma_\psi(v), u - v\rangle,$$

which when combined with (2.7) yields

$$\Phi(u) - \Phi(p_\psi(v)) \geq \langle \gamma_\psi + \gamma_\phi, u - p_\psi(v)\rangle - \frac{1}{2\mu}\|p_\psi(v) - v\|_2^2. \tag{2.10}$$

Now using the fact that $p_\psi(v)$ satisfies the first-order optimality conditions for (2.4), i.e.,

$$\gamma_\phi + \gamma_\psi + \frac{1}{\mu}(p_\psi(v) - v) = 0, \tag{2.11}$$

it follows that (2.10) is equivalent to

$$\Phi(u) - \Phi(p_\psi(v)) \geq \langle -\frac{1}{\mu}(p_\psi(v) - v), u - p_\psi(v)\rangle - \frac{1}{2\mu}\|p_\psi(v) - v\|_2^2 = \frac{1}{2\mu}\left(\|p_\psi(v) - u\|^2 - \|v - u\|^2\right). \square$$

We now analyze the iteration complexity of a slightly generalized version of Algorithm 4, where instead of keeping μ_f and μ_g fixed throughout the algorithm, we allow for these penalty parameters (step lengths) to vary from one iteration to the next. That is, at iteration k we use μ_f^k and μ_g^k.

THEOREM 2.3. *In Algorithm 4, suppose that at iteration k, μ_f is set to $\mu_f^k \geq \tau$ for some fixed $\tau > 0$ and μ_g is set to $\mu_g^k > 0$, and μ_f^k is chosen so that*

$$F(y^{k+1}) \leq Q_{\mu_f^k}^f(x^{k+1}, y^{k+1}). \tag{2.12}$$

Then the iterates $\{y^k\}$ in Algorithm 4 satisfy

$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{2\sum_{j=0}^{k-1}(\mu_f^j + \delta_j \mu_g^j)}, \quad \forall k, \tag{2.13}$$

where x^* is an optimal solution of (1.1) and $\delta_k = 1$ if $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$, i.e., iteration k was not a skipping step, and $\delta_k = 0$ otherwise. Thus, the sequence $\{F(y^k)\}$ produced by Algorithm 4 converges to $F(x^*)$. Moreover, if $\nabla f(\cdot)$ is Lipschitz continuous with Lipschitz constant $L(f)$ and $1/(\beta L(f)) \leq \mu_f \leq 1/L(f)$ where $\beta \geq 1$, the number of iterations needed to obtain an ϵ-optimal solution is at most $\lceil C/\epsilon \rceil$, where $C = \beta L(f)\|x^0 - x^*\|^2/2$.

Proof. By letting $\psi = f$, $\phi = g$, $u = x^*$ and $v = x^{j+1}$, in (2.6) of Lemma 2.2, and using (2.12), we get $p_\psi(v) = y^{j+1}$, $\Phi = F$ and

$$2\mu_f^j(F(x^*) - F(y^{j+1})) \geq \|y^{j+1} - x^*\|^2 - \|x^{j+1} - x^*\|^2. \tag{2.14}$$

Similarly, by letting $\psi = g$, $\phi = f$, $u = x^*$ and $v = y^j$ in (2.6) we get $p_g(v) = x^{j+1}$, $\Phi = F$ and

$$2\delta_j\mu_g^j(F(x^*) - F(x^{j+1})) \geq \|x^{j+1} - x^*\|^2 - \|y^j - x^*\|^2. \tag{2.15}$$

Note that (2.15) also holds when $\delta_j = 0$ since $x^{j+1} = y^j$ in this case. Taking the summation of (2.14) and (2.15) we get

$$2(\mu_f^j + \delta_j\mu_g^j)F(x^*) - 2\delta_j\mu_g^j F(x^{j+1}) - 2\mu_f^j F(y^{j+1}) \geq \|y^{j+1} - x^*\|^2 - \|y^j - x^*\|^2. \tag{2.16}$$

Summing (2.16) over $j = 0, 1, \ldots, k-1$ we get

$$\begin{aligned} &2\sum_{j=0}^{k-1}(\mu_f^j + \delta_j\mu_g^j)F(x^*) - 2\sum_{j=0}^{k-1}\delta_j\mu_g^j F(x^{j+1}) - 2\sum_{j=0}^{k-1}\mu_f^j F(y^{j+1}) \\ \geq &\sum_{j=0}^{k-1}\left(\|y^{j+1} - x^*\|^2 - \|y^j - x^*\|^2\right) \\ = &\|y^k - x^*\|^2 - \|y^0 - x^*\|^2 \\ \geq &-\|x^0 - x^*\|^2. \end{aligned} \tag{2.17}$$

For any j, since Lemma 2.2 holds for any u, letting $u = x^{j+1}$ instead of x^* we obtain similarly to (2.14) that

$$2\mu_f^j(F(x^{j+1}) - F(y^{j+1})) \geq \|y^{j+1} - x^{j+1}\|^2 \geq 0. \tag{2.18}$$

Similarly, letting $u = y^j$ instead of x^* we get from (2.15) that

$$2\delta_j\mu_g^j(F(y^j) - F(x^{j+1})) \geq \|x^{j+1} - y^j\|^2 \geq 0. \tag{2.19}$$

Hence we have

$$F(y^j) \geq F(y^{j+1}) \quad \text{and} \quad F(x^j) \geq F(x^{j+1}), \text{ for all } j. \tag{2.20}$$

The inequalities (2.20) show that the sequences of function values $F(y^j)$ and $F(x^j)$ are non-increasing. Thus we have,

$$\sum_{j=0}^{k-1} \mu_f^j F(y^{j+1}) \geq \sum_{j=0}^{k-1} \mu_f^j F(y^k) \quad \text{and} \quad \sum_{j=0}^{k-1} \delta_j \mu_g^j F(x^{j+1}) \geq \sum_{j=0}^{k-1} \delta_j \mu_g^j F(x^k). \tag{2.21}$$

Combining (2.17) and (2.21) yields

$$2\sum_{j=0}^{k-1} (\mu_f^j + \delta_j \mu_g^j) F(x^*) - 2\sum_{j=0}^{k-1} \mu_f^j F(y^k) - 2\sum_{j=0}^{k-1} \delta_j \mu_g^j F(x^k) \geq -\|x^0 - x^*\|^2. \tag{2.22}$$

Hence, since $F(y^k) \leq F(x^k)$,

$$2\sum_{j=0}^{k-1} (\mu_f^j + \delta_j \mu_g^j)(F(x^*) - F(y^k)) \geq -\|x^0 - x^*\|^2,$$

which gives us the desired result (2.13). The last two statements of the theorem follow immediately from (2.13) and the facts that $\mu_f^j \geq \tau > 0$, $\mu_g^j > 0$ and $\delta_j \geq 0$ for all j. □

COROLLARY 2.4. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and $L(g)$, respectively. For $0 < \mu_f \leq 1/L(f)$ and $0 < \mu_g \leq 1/L(g)$, Algorithm 3 satisfies*

$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{2k(\mu_f + \mu_g)}, \quad \forall k, \tag{2.23}$$

where x^* is an optimal solution of (1.1). Thus sequence $\{F(y^k)\}$ produced by Algorithm 3 converges to $F(x^*)$. Moreover, if $1/(\beta L(f)) \leq \mu_f \leq 1/L(f)$ and $1/(\beta L(g)) \leq \mu_g \leq 1/L(g)$ where $\beta \geq 1$, the number of iterations needed to get an ϵ-optimal solution is at most $\lceil C/\epsilon \rceil$, where $C = \beta\|x^0 - x^*\|^2 L(f)L(g)/(2(L(f) + L(g)))$.

Proof. The conclusion follows from Theorems 2.1 and 2.3 and the fact that there are no skipping steps.□

REMARK 2.5. *The complexity bound in Corollary 2.4 is smaller than the analogous bound for ISTA in [5] by a factor of (L(f)+L(g))/L(g). It is also interesting to observe that if $L(f) \sim L(g)$ then the gain in the number of iterations needed to obtain an ϵ-optimal solution is roughly 2. On the other hand, if $L(f) \gg L(g)$, the gain over ISTA can be very large, while if $L(g) \gg L(f)$ the theoretical gain is negligible. As our computational results show, the gain in practice is often substantial. It is easy to see that the bound in Theorem 2.3 is also an improvement over the bound in [5] as long as $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$ holds for at least one value of k. It is reasonable then to ask if the per-iteration costs of Algorithms 3 and 4 are comparable to that of ISTA. This is indeed the case when the assumption holds that minimizing $\mathcal{L}_\mu(x, y^k; \lambda^k)$ has a cost comparable to (and often involves the same computations as) computing the gradient $\nabla f(y^k)$.*

REMARK 2.6. *Our algorithms can be applied to more general problems of the form*

$$\begin{aligned} \min \quad & f(x) + g(y) \\ s.t. \quad & Ax + y = b, \end{aligned} \tag{2.24}$$

since one can express (2.24) as $\min \quad f(x) + g(b - Ax)$. While the iteration complexity bounds in Theorem 2.3 and Corollary 2.4 still apply, the subproblems that must be solved at each iteration may not be "easy". For example, even if Problem (1.3) is easy to solve, the problem $\min_{x \in \mathbb{R}^n} \{g(b - Ax) + \frac{1}{2\mu}\|x - z\|^2\}$ may not be.

REMARK 2.7. *If a convex constraint $x \in \mathcal{C}$, where $\mathcal{C}$ is a convex set is added to Problem* (1.1)*, and we impose this constraint in the two subproblems in Algorithms 3 and 4, i.e., we impose $x \in \mathcal{C}$ in the subproblems with respect to x and $y \in \mathcal{C}$ in the subproblems with respect to y, the complexity results in Theorem 2.3 and Corollary 2.4 continue to hold. The only changes in the proof are in Lemma 2.2. In the statement both u and v are restricted to lie in C. Also in the proof of Lemma 2.2, the first equality in* (2.10) *becomes a "$\geq$" inequality due to the fact that the optimality conditions* (2.11) *become*

$$\langle \gamma_\phi(p_\psi(v)) + \gamma_\psi(v) + \frac{1}{\mu}(p_\psi(v) - v), u - p_\psi(v)\rangle \geq 0, \forall u \in \mathcal{C}.$$

As in Remark 2.6, for Algorithm 3 and 4 to be effective, one must be able to solve the subproblems at each iteration efficiently. Clearly, this will only be the case for fairly simple sets $\mathcal{C}$. (See, for example [3]).

3. Fast Alternating Linearization Methods. In this section, we propose a fast alternating linearization method (FALM) which computes an ϵ-optimal solution to Problem (1.1) in $O(\sqrt{L/\epsilon})$ iterations, while keeping the work at each iteration almost the same as that required by ALM, where L is a constant that depends on the Lipschitz constants of the functions involved.

FALM is an accelerated version of ALM for solving (1.1), or equivalently (1.9), when $f(x)$ and $g(x)$ are both differentiable, and is given below as Algorithm 6. Clearly, FALM is also a Gauss-Seidel type algorithm. In fact, it is a successive over-relaxation type algorithm since $(t_k - 1)/t_{k+1} > 0, \forall k \geq 2$.

Algorithm 6: Fast Alternating Linearization Method (FALM)

Choose $\mu_f > 0$ and $\mu_g > 0$ and $x^0 = y^0 = z^1$, set $t_1 = 1$.
for $k = 1, 2, \cdots$ **do**
  $x^k := \arg\min_x Q^g_{\mu_g}(x, z^k)$
  $y^k := \arg\min_y Q^f_{\mu_f}(y, x^k)$
  $t_{k+1} := (1 + \sqrt{1 + 4t_k^2})/2$
  $z^{k+1} := y^k + \frac{t_k - 1}{t_{k+1}}(y^k - y^{k-1})$

Algorithm 6 assumes that both f and g are continuously differentiable. This assumption is needed to construct $Q^g_{\mu_g}(x, z^k)$ and $Q^f_{\mu_f}(x, z^k)$ and provide a theoretical complexity bound for Algorithm 6. To develop an algorithm and the corresponding complexity results for the case when one of the functions is non-differentiable, we use a skipping technique as in Algorithm 4. FALM with skipping steps (FALM-S), which does not require $g(x)$ to be smooth, is given below as Algorithm 7.

The following theorem gives conditions under which Algorithms 6 and 7 are equivalent.

THEOREM 3.1. *If both $f(x)$ and $g(x)$ are differentiable and $\nabla g(x)$ is Lipschitz continuous with Lipschitz constant $L(g)$, and $\mu_g \leq 1/L(g)$, then Algorithms 6 and 7 are equivalent.*

Proof. As in Theorem 2.1, if f and g are differentiable, $\mathcal{L}_{\mu_g}(x, z^k; \lambda^k) \equiv Q^g_{\mu_g}(x, z^k)$. The conclusion then follows from the fact that $F(x^k) \leq \mathcal{L}_{\mu_g}(x^k, z^k; \lambda^k)$ always holds when $\mu_g \leq 1/L(g)$ and thus there are no skipped steps. □

Note that as in Algorithm 4, Algorithm 7 keeps μ_g and μ_f fixed in all iterations. This is done to simplify the presentation. In fact it is possible to adjust μ_f and μ_g at each iteration in both Algorithms 4 and 7. A full backtracking strategy for selecting step parameters μ_f and μ_g, which includes Algorithm 7 as a special case, can be found in [24], however, the algorithmic setting in [24] is substantially more complicated, and it is yet to be shown that such strategy is efficient in practice. Hence, we present here only the much simpler

Algorithm 7: FALM with Skipping Steps (FALM-S)

Choose $\mu_f > 0$ and $\mu_g > 0$, $x^0 = y^0 = z^1$ and $\lambda^1 \in -\partial g(z^1)$, set $t_1 = 1$.
for $k = 1, 2, \cdots$ **do**
  $x^k := \arg\min_x \mathcal{L}_{\mu_g}(x, z^k; \lambda^k)$
  if $F(x^k) > \mathcal{L}_{\mu_g}(x^k, z^k; \lambda^k)$ **then**
    if *x-step was not skipped at iteration* $k-1$ **then**
      $\theta_k = \frac{\mu_f + \mu_g}{\mu_f}$
    else
      $\theta_k = 1$
    $t_k := \left(1 + \sqrt{1 + 4\theta_{k-1} t_{k-1}^2}\right)/2$
    $x^k := z^k := y^{k-1} + \frac{t_{k-1}-1}{t_k}\left(y^{k-1} - y^{k-2}\right)$; i.e. x-step is skipped at iteration k
  $y^k := \arg\min_y Q^f_{\mu_f}(y, x^k)$
  if *x-step was skipped at iteration* k **then**
    $\theta_k = \frac{\mu_f}{\mu_f + \mu_g}$
  else
    $\theta_k = 1$
  $t_{k+1} := \left(1 + \sqrt{1 + 4\theta_k t_k^2}\right)/2$
  $z^{k+1} := y^k + \frac{t_k - 1}{t_{k+1}}\left(y^k - y^{k-1}\right)$
  Choose $\lambda^{k+1} \in -\partial g(z^{k+1})$

Algorithm 7. We present for completeness the following result from [24], which we specialize for the case of Algorithm 7.

LEMMA 3.2. *If at every step of Algorithm 7,* $F(y^k) \le Q^f_{\mu_f}(x^k, y^k)$*, then*

$$2(\bar{\mu}_k t_k^2 v_k - \bar{\mu}_{k+1} t_{k+1}^2 v_{k+1}) \ge \|u^{k+1}\|^2 - \|u^k\|^2, \tag{3.1}$$

where $u^k := t_k y^k - (t_k - 1)y^{k-1} - x^*$, $v_k := 2F(y^k) - 2F(x^*)$ *and* $\bar{\mu}_{k+1} = \mu_f/2$ *if the x-step was skipped at iteration* k *and* $\bar{\mu}_{k+1} = (\mu_f + \mu_g)/2$*, otherwise.*

Proof. We will prove that the following inequality holds:

$$\begin{aligned} &2(\bar{\mu}_k t_k^2 v_k - \bar{\mu}_{k+1} t_{k+1}^2 v_{k+1}) \\ \ge\, &t_{k+1}(t_{k+1} - 1)\left(\|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2\right) + t_{k+1}\left(\|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2\right). \end{aligned} \tag{3.2}$$

The proof of (3.1) and hence, the lemma, then follows from the fact that the right hand side of inequality (3.2) equals

$$\|t_{k+1} y^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 - \|t_{k+1} z^{k+1} - (t_{k+1} - 1)y^k - x^*\|^2 = \|u^{k+1}\|^2 - \|u^k\|^2,$$

where we have used the fact that $t_{k+1} z^{k+1} := t_{k+1} y^k + t_k(y^k - y^{k-1}) - (y^k - y^{k-1})$.

First, noting that $\theta_k = \frac{\bar{\mu}_k}{\bar{\mu}_{k+1}}$ for all k, it follows from the expression for t_{k+1} that

$$\bar{\mu}_k t_k^2 = \bar{\mu}_{k+1} t_{k+1}(t_{k+1} - 1). \tag{3.3}$$

Then by applying Lemma 2.2 to the function $Q^f_{\mu_f}(u, v)$ with $u = y^k$ and $v = x^{k+1}$, we get $p^f_{\mu_f}(v) = y^{k+1}$, and

$$2\mu_f(F(y^k) - F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|x^{k+1} - y^k\|^2. \tag{3.4}$$

Now let us define $\mu_g^{k+1} := 0$ if the x-step was skipped on the $k+1$-st iteration, and to $\mu_g^{k+1} = \mu_g$ otherwise. Therefore if iteration $k+1$ is a regular iteration (i.e., $\mu_g^{k+1} = \mu_g$) applying Lemma 2.2 to the function $Q^g_{\mu_g^{k+1}}(u, v)$ with $u = y^k$, $v = z^{k+1}$, we get $p^g_{\mu_g^{k+1}}(v) = x^{k+1}$ and

$$2\mu_g^{k+1}(F(y^k) - F(x^{k+1})) \geq \|x^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2. \tag{3.5}$$

In the x-step was skipped on the $k+1$-st iteration (i.e., $\mu_g^{k+1} = 0$), (3.5) still holds since $x^{k+1} = z^{k+1}$.

Summing (3.4) and (3.5), and using the definition of $\bar{\mu}_{k+1}$ and the fact that $F(y^{k+1}) \leq F(x^{k+1})$, we obtain,

$$2\bar{\mu}_{k+1}(v_k - v_{k+1}) = 2\bar{\mu}_{k+1}(2F(y^k) - 2F(y^{k+1})) \geq \|y^{k+1} - y^k\|^2 - \|z^{k+1} - y^k\|^2 \tag{3.6}$$

Furthermore, if $\mu_g^{k+1} = \mu_g$, applying Lemma 2.2 to the function $Q^g_{\mu_g^{k+1}}(u, v)$ with $u = x^*$, $v = z^{k+1}$, we get $p^g_{\mu_g^{k+1}}(v) = x^{k+1}$ and

$$2\mu_g^{k+1}(F(x^*) - F(x^{k+1})) \geq \|x^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{3.7}$$

On the other hand, if $\mu_g^{k+1} = 0$ then (3.7) still holds, since $x^{k+1} := z^{k+1}$ Lemma 2.2 applied to the function $Q^f_{\mu_f}(u, v)$ with $u = x^*$, $v = x^{k+1}$ and $p^f_{\mu_f}(v) = y^{k+1}$, gives

$$2\mu_f(F(x^*) - F(y^{k+1}) \geq \|y^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2. \tag{3.8}$$

Summing (3.7) and (3.8), and again using the fact that $F(y^{k+1}) \leq F(x^{k+1})$ and the definition of $\bar{\mu}_{k+1}$, we obtain

$$-2\bar{\mu}_{k+1} v_{k+1} = 2\bar{\mu}_{k+1}(2F(x^*) - 2F(y^{k+1})) \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{3.9}$$

If we multiply (3.6) by $t_{k+1}(t_{k+1}-1)$ and (3.9) by t_{k+1}, and take the sum of the resulting two inequalities, we get (3.2) by using (3.3). □

Let $r(k)$ be the number of iterations among the first k iterations generated by Algorithm 7 that are regular iterations (hence, $k - r(k)$ is the number of skipping iterations). We now prove the following specialized version of Lemma 3.5 in [24].

LEMMA 3.3. *For all $k \geq 1$ the sequence of scalars $\{\bar{\mu}_k t_k^2\}$ generated by Algorithm 7 satisfies:*

$$\bar{\mu}_k t_k^2 \geq \frac{1}{4}((k - r(k))\sqrt{\mu_f} + r(k)\sqrt{\mu_f + \mu_g})^2 \tag{3.10}$$

Proof. Our proof is by induction. Since $t_1 = 1$ and $\bar{\mu}_1 = \frac{\mu_f+\mu_g}{2}$ if $r(1) = 1$ and $\bar{\mu}_1 = \frac{\mu_f}{2}$ if $r(1) = 0$, (3.10) holds for $k = 1$. Now assuming that (3.10) holds for a given $k \geq 1$ from

$$t_{k+1} := \left(1 + \sqrt{1 + 4\frac{\bar{\mu}_k}{\bar{\mu}_{k+1}} t_k^2}\right)/2$$

we have

$$\begin{aligned}
\sqrt{\bar{\mu}_{k+1}} t_{k+1} &\geq \frac{\sqrt{\mu_{k+1}}}{2} + \sqrt{\bar{\mu}_k} t_k \\
&\geq \frac{1}{2}(\sqrt{\mu_{k+1}} + (k - r(k))\sqrt{\mu_f} + r(k)\sqrt{\mu_f + \mu_g}) \\
&= \frac{1}{2}((k + 1 - r(k+1))\sqrt{\mu_f} + r(k+1)\sqrt{\mu_f + \mu_g}).
\end{aligned}$$

The last equation is derived by considering two different possible values of μ_{k+1} and the corresponding value of $r(k+1)$. □

Now we are ready to give the iteration complexity of Algorithm 7.

THEOREM 3.4. *Let $\alpha = \sqrt{\frac{\mu_f+\mu_g}{\mu_f}} - 1$. Assuming $\nabla f(\cdot)$ is Lipschitz continuous with Lipschitz constant $L(f)$, if $0 < \mu_f \leq 1/L(f)$, the sequence $\{y^k\}$ generated by Algorithm 7 satisfies*

$$F(y^k) - F(x^*) \leq \frac{2\|x^0 - x^*\|^2}{\mu_f(k + \alpha r(k))^2}. \tag{3.11}$$

Hence, the sequence $\{F(y^k)\}$ produced by Algorithm 4 converges to $F(x^*)$. Moreover, if $1/(\beta L(f)) \leq \mu_f \leq 1/L(f)$, where $\beta \geq 1$, the number of iterations required by Algorithm 7 to obtain an ϵ-optimal solution to (1.1) is at most $\lfloor\sqrt{C/\epsilon}\rfloor$, where $C = 2\beta L(f)\|x^0 - x^*\|^2$.

Proof. Using the same notation as in Lemmas 3.2 and 3.3, (3.9) implies that

$$-2\bar{\mu}_1 v_1 \geq \|y^1 - x^*\|^2 - \|z^1 - x^*\|^2.$$

Thus we have

$$2\bar{\mu}_1 v_1 + \|y^1 - x^*\|^2 \leq \|z^1 - x^*\|^2 = \|x^0 - x^*\|^2. \tag{3.12}$$

From Lemma 3.2 we know that the sequence $\{2\bar{\mu}_k t_k^2 v_k + \|u^k\|^2\}$ is non-increasing. Therefore, we have

$$2\bar{\mu}_k t_k^2 v_k \leq 2\bar{\mu}_k t_k^2 v_k + \|u^k\|^2 \leq 2\mu_1 t_1^2 v_1 + \|u^1\|^2 = 2\bar{\mu}_1 v_1 + \|y^1 - x^*\|^2 \leq \|x^0 - x^*\|^2, \tag{3.13}$$

where the equality follows from the facts that $t_1 = 1$ and $u^1 = y^1 - x^*$, and the last inequality is from (3.12).

Recalling the definition of v_k in Lemma 3.2 and the lower bound on $\mu_k t_k^2$ from Lemma 3.3, we get from (3.13) that

$$F(y^k) - F(x^*) \leq \frac{2\|x^0 - x^*\|^2}{((k - r(k))\sqrt{\mu_f} + r(k)\sqrt{\mu_f + \mu_g})^2}.$$

It is easy to check that this bound is equivalent to (3.11), and that the worst-case bound on the number of iterations follows from (3.11). □

COROLLARY 3.5. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and*

$L(g)$, respectively. For $0 < \mu_f \leq 1/L(f)$ and $0 < \mu_g \leq 1/L(g)$, Algorithm 6 satisfies

$$F(y^k) - F(x^*) \leq \frac{2\|x^0 - x^*\|^2}{(\mu_f + \mu_g)k^2}, \quad \forall k, \tag{3.14}$$

where x^* is an optimal solution of (1.1). Hence, the sequence $\{F(y^k)\}$ produced by Algorithm 6 converges to $F(x^*)$. Moreover, if $1/(\beta L(f)) \leq \mu_f \leq 1/L(f)$ and $1/(\beta L(g)) \leq \mu_g \leq 1/L(g)$, where $\beta \geq 1$, the number of iterations needed to get an ϵ-optimal solution is at most $\lceil \sqrt{C/\epsilon} - 1 \rceil$, where $C = \frac{2\beta L(f)L(g)\|x^0 - x^*\|^2}{L(f)+L(g)}$.

Proof. Note that since $\mu_f \leq 1/L(f)$ and $\mu_g \leq 1/L(g)$, from Theorem 3.1 we know that Algorithms 6 and 7 are equivalent; i.e., every step in Algorithm 7 is a regular step. Therefore, Theorem 3.4 holds and $r(k) = k$, which leads to (3.14). □

REMARK 3.6. *The complexity bound in Corollary 3.5 is smaller than the analogous bound for FISTA in [5] by a factor of $\sqrt{\frac{L(f)+L(g)}{L(g)}}$. Similarly to the case of the basic algorithms in the previous section, if $L(f) \sim L(g)$ then the gain in the number of iterations needed to obtain an ϵ-optimal solution is roughly $\sqrt{2}$. On the other hand, if $L(f) \gg L(g)$, the gain over FISTA can be very large. while if $L(g) \gg L(f)$, the gain is negligible. Our computational results show that the gain is substantial in many cases, but is smaller than the gain observed in the case of the basic algorithm, in accordance with the theory. It is easy to see that the bound in Theorem 3.4 is also an improvement over the bound in [5] as long as $F(x^{k+1}) \leq \mathcal{L}_\mu(x^{k+1}, y^k; \lambda^k)$ holds for at least one value of k. As in the case of Algorithms 3 and 4, the per-iteration costs of Algorithms 6 and 7 are comparable to that of FISTA.*

REMARK 3.7. *Line 6 in Algorithm 6 and Line 15 in Algorithm 7 can be changed to:*

$$z^{k+1} := w^k + \frac{1}{t_{k+1}}[t_k(y^k - w^{k-1}) - (w^k - w^{k-1})],$$

where $w^k := \alpha x^k + (1-\alpha)y^k, \alpha \in (0,1)$, and Theorem 3.4 and Corollary 3.5 still hold.

4. Comparison of ALM, FALM, ISTA, FISTA, ADAL and SADAL. In this section we focus on comparing the performance of our basic and fast ALMs, with and without skipping steps, against ISTA, FISTA, ADAL (Algorithm 1) and SADAL (Algorithm 2) on benchmark wavelet-based image deblurring problems from [1, 18]. In our first set of tests, we used the well-known Cameraman image [1] of size 256×256 after a uniform blur of size 9×9 (denoted by the operator R) was applied to it and Gaussian noise (generated by the function *randn* in MATLAB with a seed of 0 and a standard deviation of 0.56) was added. We also tested another blurring kernel and image, which we will describe later. Since the vector of coefficients of the wavelet transform of the image is sparse in this problem, one can try to reconstruct the image u from the observed blurred image b by solving the problem:

$$\bar{x} := \arg\min_x \quad \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1, \tag{4.1}$$

and setting $u := W\bar{x}$, where $A := RW$ and W is the inverse discrete Haar wavelet transform with four levels. By defining $f(x) := \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$, it is clear that (4.1) can be expressed in the form of (1.1) and can be solved by ALM-S (Algorithms 4 and 5), FALM-S (Algorithm 7), ISTA, FISTA, ADAL and SADAL. However, in order to use ALM (Algorithm 3) and FALM (Algorithm 6), we need to smooth $g(x)$ first, since these two algorithms require both f and g to be smooth to ensure the existence of the gradients. Here we apply the smoothing technique introduced by Nesterov [38] since this technique guarantees that the gradient of the smoothed function is Lipschitz continuous. A smoothed approximation to the ℓ_1 function

$g(x) := \rho\|x\|_1$ with smoothness parameter $\sigma > 0$ is

$$g_\sigma(x) := \max\{\langle x, z\rangle - \frac{\sigma}{2}\|z\|_2^2 : \|z\|_\infty \le \rho\}. \tag{4.2}$$

It is easy to show that the optimal solution $z_\sigma(x)$ of (4.2) is

$$z_\sigma(x)_i = \min\{\rho, \max\{x_i/\sigma, -\rho\}\}, \forall i. \tag{4.3}$$

According to Theorem 1 in [38], the gradient of g_σ is given by $\nabla g_\sigma(x) = z_\sigma(x)$ and is Lipschitz continuous with Lipschitz constant $L(g_\sigma) = 1/\sigma$. After smoothing g, we can apply Algorithms 3 and 6 to solve the smoothed problem:

$$\min_x f(x) + g_\sigma(x). \tag{4.4}$$

We have the following theorem about the ϵ-optimal solutions of problems (4.1) and (4.4).

THEOREM 4.1. *Let $\sigma = \frac{\epsilon}{n\rho^2}$ and $\epsilon > 0$. If $x(\sigma)$ is an $\epsilon/2$-optimal solution to (4.4), then $x(\sigma)$ is an ϵ-optimal solution to (4.1).*

Proof. Let $D_g := \max\{\frac{1}{2}\|z\|_2^2 : \|z\|_\infty \le \rho\} = \frac{1}{2}n\rho^2$ and x^* and $x^*(\sigma)$ be optimal solution to problems (4.1) and (4.4), respectively. From $\sigma D_g = \frac{\epsilon}{2}$ and the facts that $x(\sigma)$ is an $\epsilon/2$-optimal solution to (4.4) and

$$g_\sigma(x) \le g(x) \le g_\sigma(x) + \sigma D_g, \forall x \in \mathbb{R}^n, \tag{4.5}$$

we have

$$\begin{aligned} f(x(\sigma)) + g(x(\sigma)) - f(x^*) - g(x^*) &\le f(x(\sigma)) + g_\sigma(x(\sigma)) + \sigma D_g - f(x^*(\sigma)) - g_\sigma(x^*(\sigma)) \\ &\le \epsilon/2 + \sigma D_g = \epsilon. \qquad \square \end{aligned}$$

Thus, to find an ϵ-optimal solution to (4.1), we can apply Algorithms 3 and 6 to find an $\epsilon/2$-optimal solution to (4.4) with $\sigma = \frac{\epsilon}{n\rho^2}$. The results in Corollaries 2.4 and 3.5 hold since ∇g_σ is Lipschitz continuous. However, the numbers of iterations needed by ALM and FALM to obtain an ϵ-optimal solution to (4.1) become $O(1/\epsilon^2)$ and $O(1/\epsilon)$, respectively, since the Lipschitz constant $L(g_\sigma) = 1/\sigma = \frac{n\rho^2}{\epsilon} = O(1/\epsilon)$.

When Algorithms 4 and 7 are applied to solve (4.1), the subproblems (1.2) and (1.3) are easy to solve. Specifically, (1.2) corresponds to solving a linear system which requires $O(n \log n)$ operations as does the computation of $\nabla f(x)$, due to the special structures of R and W (see [1, 2]); (1.3) corresponds to the vector shrinkage operation (1.5). When Algorithms 3 and 6 are applied to solve (4.4), (1.3) with g replaced by g_σ is also easy to solve; its optimal solution is

$$x := z - \tau \min\{\rho, \max\{-\rho, \frac{z}{\tau + \sigma}\}\}.$$

We now list the algorithms that we tested and some of their implementational details.

- Our ADAL, FISTA, and ISTA implementations were based on the MATLAB codes for SALSA and FISTA that we downloaded from http://cascais.lx.it.pt/∼mafonso/salsa.html. For ADAL we used the SALSA code with one alternating direction minimization pass per iteration as described in [1]. For FISTA we used the MATLAB FISTA code's default inputs and for ISTA we simply deleted the second and third steps in (1.17) from the FISTA code.

- Since ALM is equivalent to SADAL when both functions are smooth, we implemented ALM as SADAL when we solved (4.4), which means that no gradient computations were performed in ALM.
- SADAL was applied directly to the nonsmooth Problem (4.1) (without smoothing); hence its iterates were slightly different from those of ALM.
- We implemented ALM-S as Algorithm 5 since the latter was usually faster than Algorithm 4. The main difference between ALM-S and SADAL is that ALM-S allows for different values of μ_g and μ_f and it allows steps to be skipped, hence our theory applies to all our ALM-S variants.
- Analogously the difference between FALM and FALM-S is the ability of the latter to use different values for μ_f and μ_g and that due to the implementation of the step skipping mechanism FALM-S variants satisfy our convergence theory.
- In our tests for ALM-S and FALM-S, μ_f was always set to 1, and we tested several different values for μ_g for each problem that were chosen proportional to $1/\rho$, since ρ is proportional to the change in the subgradient of $g(x)$. We present results for four values of μ_g for each ρ that were most indicative of the effects of the choice of μ_g.
- In all algorithms, we set the initial points $x^0 = y^0 = \mathbf{0}$, and in FALM and FALM-S we set $z^1 = \mathbf{0}$. Moreover, λ^0 was set to $\mathbf{0}$ in Algorithms 1, 2, 5 and 7 since $\nabla g_\sigma(x^0) = \mathbf{0}$ and $\mathbf{0} \in -\partial g(x^0)$ when $x^0 = \mathbf{0}$. Also, whenever $g(x)$ was smoothed, we set the smoothing parameter $\sigma = 10^{-6}$. μ was set to 1 in ADAL and SADAL since the Lipschitz constant of the gradient of function $\frac{1}{2}\|RW(\cdot) - b\|_2^2$ was known to be less than or equal to 1 in all of our tests. We set μ to 1 even for ALM and FALM for solving the smoothed problems. Although this violates the requirement $\mu \leq \frac{1}{L(g_\sigma)}$ in Corollaries 2.4 and 3.5, as we see from our numerical results reported below, ALM and FALM still worked very well.
- All of our codes were written in MATLAB and run in MATLAB 7.12.0 on a laptop with Intel Core I5 2.5 GHz CPU and 4GB of RAM.

We compared the performance of the different algorithms based on the results produced by FISTA. We first ran FISTA for 1000 iterations and recorded the objective function value FISTA(j) obtained by FISTA at the j-th iteration. Then for all other algorithms, we recorded when they first reached an objective function value that was smaller than FISTA(100), FISTA(500) and FISTA(1000), or when the maximum number of iterations, which was set to 5000 in our tests, was reached. We report the number of iterations, CPU times and the number of skipping steps (in parentheses), needed by each algorithm at each recording point in Tables 4.1, 4.2 and 4.3, corresponding to $\rho = 0.001, 0.0075$ and 0.1, respectively. Note that skipping steps only occur in Algorithm ALM-S and FALM-S. Also note that $\rho = 0.0075$ was suggested in [1] to give a good image recovery.

In Table 4.1, we report the results for $\rho = 0.001$ using four different values of μ_g $(0.1, 1, 10, 100)$ for ALM-S and FALM-S. Specifically, for ALM-S1 we used $\mu_g = 0.1$, for ALM-S2 we used $\mu_g = 1$, etc. From Table 4.1 we see that the fast algorithms (FISTA, FALM and FALM-S) took much fewer iterations and were much faster than the basic algorithms (ISTA, ADAL, SADAL, ALM and ALM-S) for solving Problem (4.1) with $\rho = 0.001$. For example, to reduce the objective function value to below FISTA(100) = 16221.24, the basic algorithms ISTA and ADAL needed more than 1300 iterations, and SADAL and ALM needed 689 iterations, while all the fast algorithms needed less than 100 iterations. In particular, FALM-S4 (i.e., $\mu_g = 100$) needed only 9 iterations for this purpose. To reduce the objective function value to below FISTA(1000) = 12868.14, all the basic algorithms needed more than 5000 iterations, while the fast algorithms needed less than 1000 iterations, except for FALM-S2, which needed 1440 iterations. We also see that FALM-S was much faster

TABLE 4.1

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.001$ *and uniform kernel on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 0.1, 1, 10, 100$. $FISTA(100) = 16221.24$, $FISTA(500) = 13427.78$, $FISTA(1000) = 12868.14$. $ISTA(500) = 17993.24$, $ISTA(2500) = 15466.98$, $ISTA(5000) = 14735.47$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	13	500	60	1000	124
ISTA	1374	166	>5000	>591	>5000	>591
ADAL	1375	282	>5000	>1047	>5000	>1047
SADAL	689	154	>5000	>1123	>5000	>1123
ALM	689	154	>5000	>1147	>5000	>1147
FALM	70	17	351	89	701	180
ALM-S1	1251 (0)	338	>5000 (0)	>1363	>5000 (0)	>1363
ALM-S2	689 (0)	187	>5000 (1)	>1349	>5000 (1)	>1349
ALM-S3	127 (0)	34	4456 (2028)	1244	>5000 (2300)	>1396
ALM-S4	15 (0)	4	3360 (3104)	988	>5000 (4670)	>1456
FALM-S1	96 (0)	28	477 (0)	138	970 (343)	281
FALM-S2	70 (0)	20	351 (0)	101	1440 (616)	418
FALM-S3	29 (0)	8	161 (21)	47	670 (530)	196
FALM-S4	9 (0)	3	179 (146)	52	687 (654)	200
ISTA	500	61	2500	298	5000	591
ADAL	501	102	2501	515	5001	1047
SADAL	252	56	1252	280	2502	561
ALM	252	54	1252	285	2502	578
ALM-S1	456 (0)	121	2274 (0)	616	4547 (0)	1239
ALM-S2	252 (0)	68	1252 (0)	336	2502 (0)	674
ALM-S3	47 (0)	13	229 (0)	62	497 (44)	136
ALM-S4	7 (0)	2	27 (0)	7	87 (36)	24

than ALM-S for the same choice of μ_g. We see from Table 4.1 that the larger μ_g was in ALM-S, the faster the algorithm was. To further compare the performance of the basic algorithms, we report the number of iterations, CPU times, etc., needed by them to first obtain a lower objective function value than that obtained by ISTA at its 500th, 2500th and 5000th iterations. From the second part of Table 4.1 we see that ADAL needed almost the same number of iterations as ISTA to achieve the same objective function value, but took more CPU time, since instead of computing the gradient, it required solving a specially structured system of linear equations. We also see that SADAL and ALM needed the same number of iterations, and they both took fewer iterations and less CPU time than ISTA and ADAL to achieve a given objective function value. We also see that ALM-S1, ALM-S2, ALM-S3 and ALM-S4 were all faster than ISTA in terms of number of iterations, and ALM-S3 and ALM-S4 were both faster than ISTA in terms of CPU times. This indicates that the alternating linearization technique helps for this particular image deblurring problem, especially when a larger step size μ_g was chosen.

We plot in Figure 4.1 the objective function value versus the number of iterations taken by ISTA, FISTA, ALM-S3 and FALM-S3 (i.e., $\mu_g = 10$) for solving (4.1) with $\rho = 0.001$. We see from Figure 4.1 that, for this particular problem, the performance of the accelerated algorithm FALM-S3 was much better than that of ALM-S3. We also see that ALM-S3 performed much better than ISTA, and FALM-S3 performed much better than FISTA, further illustrating the benefit of our alternating linearization approach.

We report results in Table 4.2 for $\rho = 0.0075$ and four different values of μ_g in ALM-S and FALM-S: $1/7.5, 1/0.75, 1/0.075, 1/0.0075$. Table 4.2 shows that the fast algorithms were much faster than the basic

TABLE 4.2

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.0075$ *and uniform kernel on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 1/7.5, 1/0.75, 1/0.075, 1/0.0075$. $FISTA(100) = 76909.27$, $FISTA(500) = 71006.83$, $FISTA(1000) = 69685.65$. $ISTA(500) = 79532.22$, $ISTA(2500) = 75099.91$, $ISTA(5000) = 73630.42$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	13	500	60	1000	119
ISTA	1208	147	>5000	>606	>5000	>606
ADAL	1208	236	>5000	>975	>5000	>975
SADAL	607	122	>5000	>1018	>5000	>1018
ALM	607	128	>5000	>1064	>5000	>1064
FALM	70	18	353	90	712	182
ALM-S1	1067 (0)	276	>5000 (0)	>1271	>5000 (0)	>1271
ALM-S2	535 (27)	136	>5000 (300)	>1264	>5000 (300)	>1264
ALM-S3	155 (73)	40	>5000 (3421)	>1310	>5000 (3421)	>1310
ALM-S4	409 (401)	113	>5000 (4852)	>1324	>5000 (4852)	>1324
FALM-S1	94 (0)	26	494 (392)	138	994 (892)	275
FALM-S2	84 (17)	24	548 (481)	154	1057 (990)	296
FALM-S3	49 (32)	14	451 (434)	127	951 (934)	267
FALM-S4	71 (68)	20	469 (466)	131	968 (965)	269
ISTA	500	61	2500	304	5000	606
ADAL	501	98	2500	488	5000	975
SADAL	252	51	1256	253	2511	506
ALM	252	53	1256	266	2511	536
ALM-S1	443 (0)	114	2207 (0)	569	4413 (0)	1125
ALM-S2	221 (8)	57	1109 (62)	282	2219 (129)	560
ALM-S3	53 (17)	13	355 (192)	93	774 (455)	202
ALM-S4	200 (196)	55	754 (738)	205	1282 (1252)	346

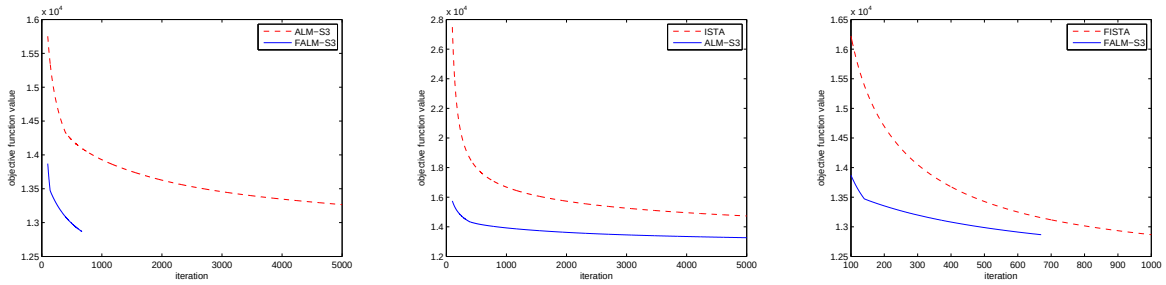

FIG. 4.1. *The comparison of objective function values versus number of iterations for Algorithms ISTA, FISTA, ALM-S3, FALM-S3 for* $\rho = 0.001$

algorithms. In particular, FALM took the fewest iterations. We again observe that FALM-S was much faster than ALM-S for the same choice of μ_g, and ALM-S took fewer iterations and was much faster than ISTA, when μ_g was set to a large value (as in ALM-S2, ALM-S3 and ALM-S4). Again, this indicates that the alternating linearization technique helped for this particular problem. We also note that although the performance of ALM-S1, ALM-S2 and ALM-S3 improved as μ_g increased, the performance of ALM-S4 was worse than ALM-S3. The reason for this is that μ_g was too large in ALM-S4, causing it to take more skipping steps, which resulted in ALM-S4 being more like ISTA and hence slower. As suggested in [1], $\rho = 0.0075$ gives the best image recovery. Therefore, we exhibit the original and blurred images, and those recovered by ALM-S1 and FALM-S1 (i.e., $\mu_g = 1/7.5$) in Figure 4.2. We see that both ALM-S1 and FALM-S1 recovered

TABLE 4.3

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.1$ *and uniform kernel on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 0.01, 0.1, 1, 10$. $FISTA(100) = 834675.05$, $FISTA(500) = 807350.97$, $FISTA(1000) = 803361.57$. $ISTA(500) = 850979.64$, $ISTA(2500) = 825934.44$, $ISTA(5000) = 818918.90$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	12	500	60	1000	120
ISTA	1272	151	>5000	>588	>5000	>588
ADAL	1272	287	>5000	>1128	>5000	>1128
SADAL	711	165	>5000	>1125	>5000	>1125
ALM	711	169	>5000	>1187	>5000	>1187
FALM	83	21	963	250	>5000	>1441
ALM-S1	1261 (0)	342	>5000 (0)	>1372	>5000 (0)	>1372
ALM-S2	1158 (0)	311	>5000 (0)	>1331	>5000 (0)	>1331
ALM-S3	645 (13)	169	>5000 (50)	>1386	>5000 (50)	>1386
ALM-S4	338 (242)	96	>5000 (3611)	>1437	>5000 (3611)	>1437
FALM-S1	101 (80)	32	501 (480)	158	1001 (980)	313
FALM-S2	100 (80)	31	500 (480)	155	1000 (980)	312
FALM-S3	97 (88)	30	497 (488)	154	997 (988)	310
FALM-S4	94 (91)	29	494 (491)	154	994 (991)	309
ISTA	500	61	2500	296	5000	588
ADAL	500	110	2501	564	5001	1128
SADAL	272	63	1439	331	3000	682
ALM	272	65	1438	339	2996	709
ALM-S1	497 (0)	134	2477 (0)	669	4952 (0)	1359
ALM-S2	456 (0)	121	2274 (0)	609	4547 (0)	1211
ALM-S3	256 (8)	68	1262 (20)	343	2518 (31)	698
ALM-S4	156 (119)	44	626 (436)	178	1215 (834)	348

the image very well. It is interesting to note that, although FALM-S reduces the objective function value faster, the image recovered by ALM-S was better.

We report results in Table 4.3 for $\rho = 0.1$ and four different values of μ_g in ALM-S and FALM-S: $0.01, 0.1, 1, 10$. Conclusions can be drawn from Table 4.3 that are similar to those can be drawn from Tables 4.1 and 4.2, except that in this case the performance of the theoretically unsupported FALM algorithm was worse than all of the theoretically supported FALM-S variants.

We also illustrate our comparisons graphically by plotting in Figures 4.3, 4.4 and 4.5 the number of iterations needed to reduce the objective function value to FISTA(100), FISTA(500) and FISTA(1000) for the fast algorithms, and to ISTA(500), ISTA(2500) and ISTA(5000) for the basic algorithms. In the graphs in the left portion of Figures 4.3, 4.4 and 4.5, each group of three bars gives the number of iterations needed by each algorithm to reduce the objective function value to FISTA(100), FISTA(500) and FISTA(1000) in that order. In the graphs in the right portion of Figures 4.3, 4.4 and 4.5, each group of three bars gives the number of iterations needed by each algorithm to reduce the objective function value to ISTA(500), ISTA(2500) and ISTA(5000) in that order.

We also solved Problem (4.1) with a different blurring kernel, the kernel 3A in [1] (see Table 1 in [1] for details). We report results for this deblurring problem with $\rho = 0.001, 0.04$ and 0.1 in Tables 4.4, 4.5 and 4.6, respectively. Note that $\rho = 0.04$ is given as the default value for the regularization parameter for this problem in the SALSA code since this value approximately gives the best recovery [1]. We now compare the results reported in Tables 4.1 and 4.4 corresponding to solving Problem (4.1) with the same ρ ($\rho = 0.001$),

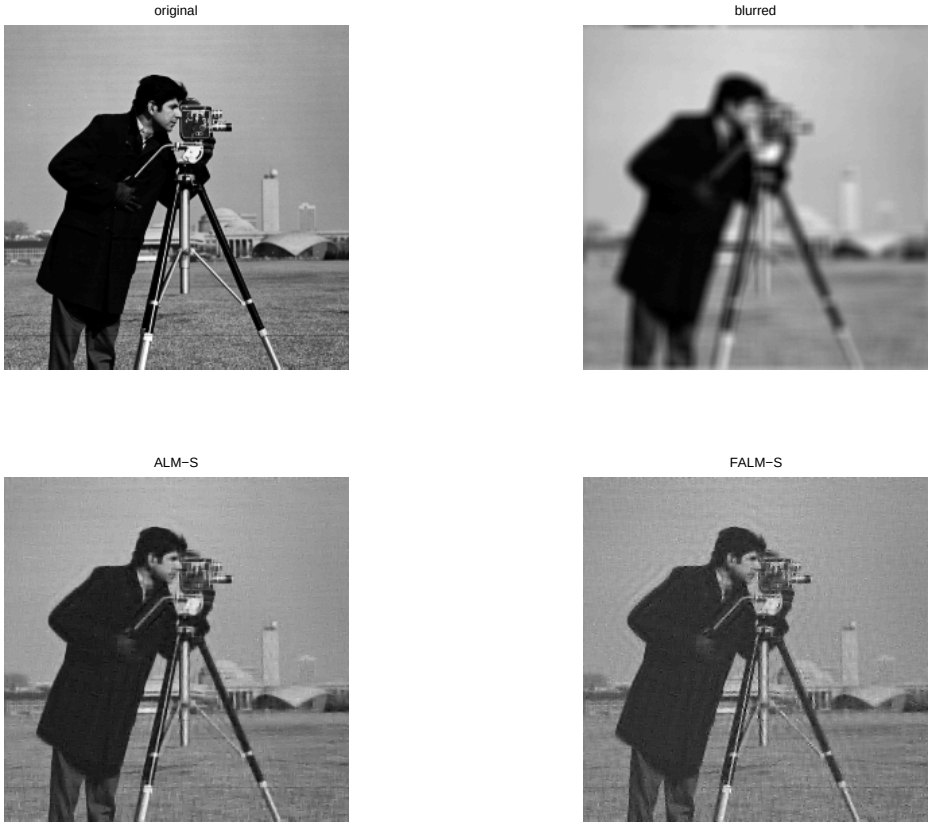

FIG. 4.2. *The original, blurred, and recovered images by ALM-S1 and FALM-S1 with* $\rho = 0.0075$

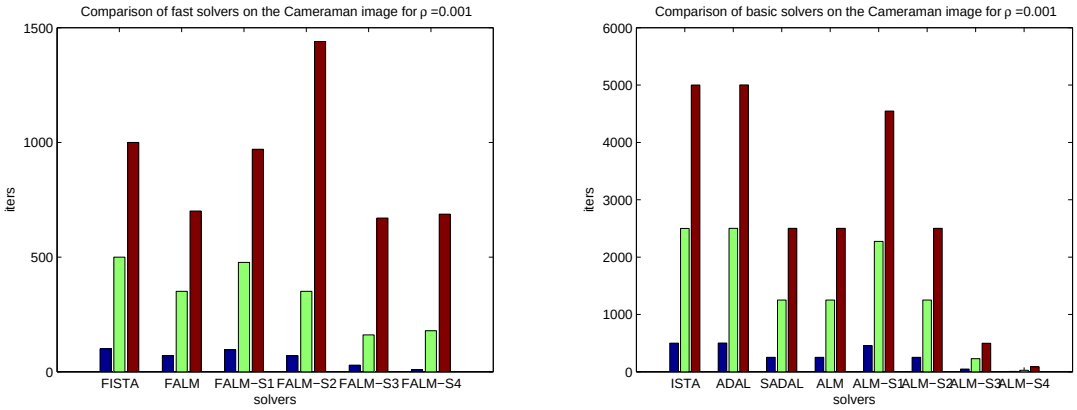

FIG. 4.3. *Comparison of the number of iterations needed for the Cameraman image for* $\rho = 0.001$

but with a different operator A and blurred image b. We see that, to reduce the objective function to FISTA(100), the number of iterations needed by the basic algorithms in these two tables differ by about 10%, while the difference between the number of iterations needed by the fast algorithms for the two blurring kernels is negligible. These tables show that to obtain higher accuracy, i.e., objective values corresponding to FISTA(500) and FISTA(1000), the fast algorithms FISTA-S3 and FISTA-S4 that used large step sizes (i.e., a

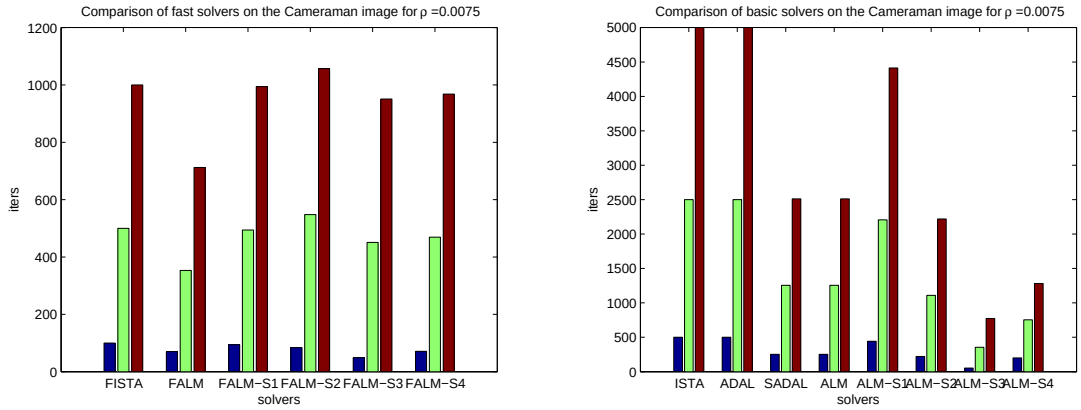

FIG. 4.4. *Comparison of the number of iterations needed for the Cameraman image for* $\rho = 0.0075$

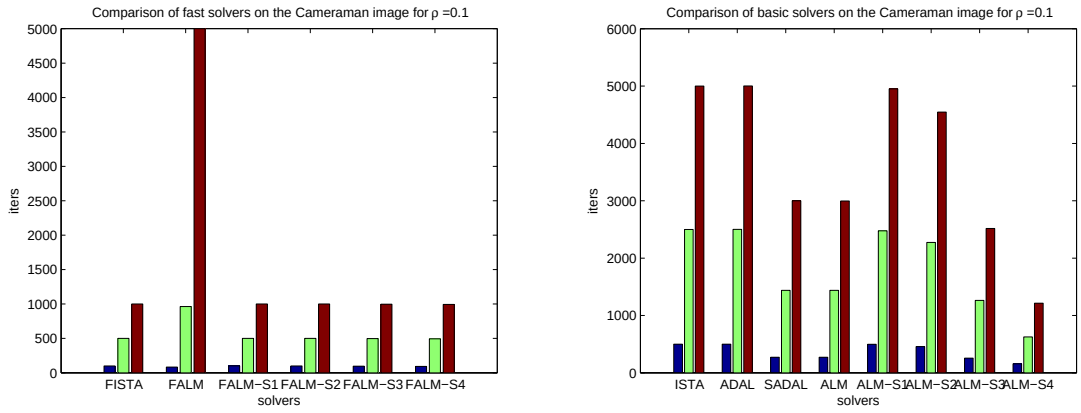

FIG. 4.5. *Comparison of the number of iterations needed for the Cameraman image for* $\rho = 0.1$

μ_g of 10 and 100), behaved quite differently for the two blurring kernels. To reduce the objective function to ISTA(500) and ISTA(2500), the numbers of iterations needed by all of the basic algorithms were essentially the same. To reduce the objective function to ISTA(5000), the number of iterations needed by ALM-S3 and ALM-S4 (i.e., those variants that used a large μ_g) differed by about 8.7% and 67%, respectively, while the numbers of iterations needed by other algorithms were almost the same. Similarly, comparing the results reported in Table 4.3 with those in Table 4.6 corresponding to $\rho = 0.1$, we again observe that to reduce the objective function value to FISTA(100), the numbers of iterations needed by the basic algorithms were slightly different, while the numbers of iterations needed by the fast algorithms were almost identical. Also, to reduce the objective function to ISTA(5000), the numbers of iterations needed by the basic algorithms were almost the same, except for ALM-S4 ($\mu_g = 10$) where there was about a 3.5% difference.

We also solved Problem (4.1) with the uniform kernel on a different image (the Lena image) with a higher resolution with 512×512 pixels. The results for $\rho = 0.001$ are reported in Table 4.7. Comparing the results reported in Tables 4.1 and 4.7, we can see the effects of using different images with different resolutions since the two problems have the same blurring kernel and ρ. We see from Tables 4.1 and 4.7 that, to reduce the objective function value to FISTA(100) and ISTA(500), the numbers of iterations needed by the algorithms are almost the same. To reduce the objective function value to FISTA(1000), the number of iterations needed by FALM-S2, FALM-S3 and FALM-S4 differ about 6%, 12% and 11%, respectively. To

reduce the objective function value to ISTA(5000), the number of iterations needed by ALM-S3 and ALM-S4 differ greatly.

TABLE 4.4

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.001$ *and kernel 3A on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 0.1, 1, 10, 100$. $FISTA(100) = 16822.89$, $FISTA(500) = 11850.74$, $FISTA(1000) = 11310.35$. $ISTA(500) = 27300.27$, $ISTA(2500) = 14447.61$, $ISTA(5000) = 12946.92$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	14	500	65	1000	130
ISTA	1522	202	>5000	>656	>5000	>656
ADAL	1523	347	>5000	>1157	>5000	>1157
SADAL	763	185	>5000	>1161	>5000	>1161
ALM	763	188	>5000	>1235	>5000	>1235
FALM	71	21	351	106	701	209
ALM-S1	1385 (0)	375	>5000 (0)	>1333	>5000 (0)	>1333
ALM-S2	763 (0)	203	>5000 (1)	>1326	>5000 (1)	>1326
ALM-S3	140 (0)	37	4361 (2172)	1201	>5000 (2561)	>1377
ALM-S4	17 (0)	4	3463 (3233)	990	>5000 (4707)	>1423
FALM-S1	96 (0)	29	477 (0)	144	965 (237)	289
FALM-S2	71 (0)	21	351 (0)	105	1455 (460)	442
FALM-S3	30 (0)	9	145 (0)	43	624 (466)	187
FALM-S4	10 (0)	3	120 (82)	37	637 (599)	194
ISTA	500	66	2500	328	5000	656
ADAL	501	114	2502	575	5000	1157
SADAL	252	61	1252	302	2502	593
ALM	252	63	1252	307	2502	617
ALM-S1	456 (0)	126	2274 (0)	610	4547 (0)	1213
ALM-S2	252 (0)	66	1252 (0)	332	2502 (0)	665
ALM-S3	47 (0)	12	229 (0)	61	457 (0)	121
ALM-S4	7 (0)	2	27 (0)	7	52 (0)	14

The results of our tests on the image deblurring Problem (4.1) show the following, at least with regard to the limited set of image blurring kernels and regularization parameters that we tried.

- The accelerated (i.e., fast) algorithms take far fewer iterations and are indeed much faster than the basic algorithms. Note that all S (skipping) variants satisfy the requirements for our complexity theory.
- ALM (SADAL) and its skipping variants substantially outperform ISTA and ADAL; hence, the extra linearization/minimization step pays off.
- FALM and its skipping variants usually outperform FISTA in terms of number of iterations, but the degree to which this is true is not as great as in the ALM/ISTA comparison.
- Choosing a suitable μ_g for the x-minimization subproblem different from μ_f can result in significant improvements in performance for both ALM-S and FALM-S. When appropriately chosen these algorithms are substantially faster than ISTA and FISTA in reducing the objective function initially; as higher accuracy is demanded these differences fade. It is clear from the results that a continuation strategy for choosing μ_g might work well; clearly, at the beginning choosing the step length μ_g large works best, while towards the later iterations choosing μ_g large can lead to erratic behavior. We note however, that due to the fact that the problem data is noisy, once the level of noise is reached, further iterations do not produced meaningful results; hence, the results for achieving an objective

TABLE 4.5

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.04$ *and kernel 3A on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 1/40, 1/4, 1/0.4, 1/0.04$. $FISTA(100) = 397184.14$, $FISTA(500) = 376921.35$, $FISTA(1000) = 372110.29$. $ISTA(500) = 407885.99$, $ISTA(2500) = 391039.34$, $ISTA(5000) = 386204.47$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	13	500	64	1000	127
ISTA	1252	162	>5000	>636	>5000	>636
ADAL	1252	270	>5000	>1058	>5000	>1058
SADAL	639	139	>5000	>1088	>5000	>1088
ALM	639	148	>5000	>1140	>5000	>1140
FALM	72	20	375	103	784	218
ALM-S1	1222 (0)	321	>5000 (0)	>1314	>5000 (0)	>1314
ALM-S2	1002 (0)	268	>5000 (0)	>1320	>5000 (0)	>1320
ALM-S3	437 (108)	117	>5000 (1255)	>1340	>5000 (1255)	>1340
ALM-S4	328 (289)	92	>5000 (4285)	>1398	>5000 (4285)	>1398
FALM-S1	100 (69)	30	500 (469)	149	1000 (969)	295
FALM-S2	98 (72)	30	498 (472)	149	998 (972)	297
FALM-S3	91 (80)	27	491 (480)	146	991 (980)	293
FALM-S4	89 (86)	28	488 (485)	148	988 (985)	296
ISTA	500	64	2500	321	5000	636
ADAL	501	108	2500	536	5000	1058
SADAL	255	56	1283	280	2580	567
ALM	255	58	1282	293	2579	588
ALM-S1	489 (0)	129	2441 (0)	641	4880 (0)	1283
ALM-S2	402 (0)	107	2002 (0)	533	4002 (0)	1059
ALM-S3	174 (41)	47	872 (217)	234	1742 (436)	467
ALM-S4	150 (134)	42	605 (527)	170	1133 (976)	317

value of FISTA(1000) may not be very significant.

- Ignoring the step skipping mechanism as is done is ALM, SADAL and FALM can result in erratic behavior.
- The progress of both ALM-S and FALM-S eventually slows down if μ_g is chosen too large, but it is clear that FALM-S is more sensitive to a poor choice of μ_g. This is not surprising, since the FALM algorithms use over-relaxation and do not generate monotonically decreasing objective function values.

For other problem classes, choosing values for μ (μ_f and μ_g) that may violate the requirements of our theory, such as using continuation on μ, may be necessary to obtain competitive results in practice.

5. Conclusion. In this paper, we proposed both basic and accelerated versions of alternating linearization methods for minimizing the sum of two convex functions. Our basic methods require at most $O(L/\epsilon)$ iterations to obtain an ϵ-optimal solution, if the penalty parameters $1/(\beta L(f)) \leq \mu_f \leq 1/L(f)$ and $1/(\beta L(g)) \leq \mu_g \leq 1/L(g)$ where $\beta \geq 1$, $L(f)$, $L(g)$ and L are Lipschitz constants for ∇f, ∇g and an appropriate combination of them, while our accelerated methods require at most $O(\sqrt{L/\epsilon})$ iterations with only a small additional amount of computational effort at each iteration. Algorithms with complexities $O(L(f)/\epsilon)$ and $O(\sqrt{L(f)/\epsilon})$ for requiring only f to be smooth, were also analyzed. Numerical results on wavelet-based image deblurring problems are reported. These results demonstrate the efficiency and the practical potential of our algorithms.

TABLE 4.6

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.1$ *and kernel 3A on the Cameraman image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 0.01, 0.1, 1, 10$. $FISTA(100) = 911837.36$, $FISTA(500) = 877332.50$, $FISTA(1000) = 870941.70$. $ISTA(500) = 929486.61$, $ISTA(2500) = 902074.88$, $ISTA(5000) = 893555.94$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	14	500	64	1000	127
ISTA	1283	163	>5000	>637	>5000	>637
ADAL	1283	269	>5000	>1048	>5000	>1048
SADAL	699	153	>5000	>1126	>5000	>1126
ALM	699	157	>5000	>1126	>5000	>1126
FALM	78	21	485	133	>5000	>1378
ALM-S1	1271 (0)	332	>5000 (0)	>1302	>5000 (0)	>1302
ALM-S2	1167 (0)	307	>5000 (0)	>1316	>5000 (0)	>1316
ALM-S3	651 (15)	171	>5000 (54)	>1318	>5000 (54)	>1318
ALM-S4	345 (249)	95	>5000 (3530)	>1383	>5000 (3530)	>1383
FALM-S1	101 (85)	30	501 (485)	149	1001 (985)	298
FALM-S2	100 (85)	30	500 (485)	148	1000 (985)	296
FALM-S3	97 (88)	29	497 (488)	148	997 (988)	295
FALM-S4	94 (91)	28	494 (491)	147	994 (991)	297
ISTA	500	63	2500	319	5000	637
ADAL	501	105	2501	523	5001	1048
SADAL	268	58	1386	303	2820	613
ALM	268	60	1385	311	2818	635
ALM-S1	497 (0)	130	2477 (0)	645	4952 (0)	1289
ALM-S2	456 (0)	120	2274 (0)	600	4547 (0)	1197
ALM-S3	256 (8)	68	1264 (23)	334	2520 (35)	664
ALM-S4	157 (120)	43	635 (446)	175	1256 (879)	347

Acknowledgement. We would like to thank Zaiwen Wen for insightful discussions on the topic of this paper, Mario Figueiredo for providing code for the blurring kernels used in our numerical experiments, and the two anonymous referees for their constructive suggestions that improved the presentation of the paper greatly.

REFERENCES

[1] M. AFONSO, J. BIOUCAS-DIAS, AND M. FIGUEIREDO, *Fast image recovery using variable splitting and constrained optimization*, IEEE Transactions on Image Processing, 19 (2010), pp. 2345–2356.

[2] ———, *An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems*, IEEE Transactions on Image Processing, 20 (2011), pp. 681–695.

[3] N. S. AYBAT, D. GOLDFARB, AND G. IYENGAR, *Fast first-order methods for stable principal component pursuit*, preprint available at http://arxiv.org/abs/1105.2126, (2011).

[4] O. BANERJEE, L. EL GHAOUI, AND A. D'ASPREMONT, *Model selection through sparse maximum likelihood estimation for multivariate gaussian for binary data*, Journal of Machine Learning Research, 9 (2008), pp. 485–516.

[5] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sciences, 2 (2009), pp. 183–202.

[6] D. P. BERTSEKAS, *Nonlinear Programming, 2nd Ed*, Athena Scientific, Belmont, Massachusetts, 1999.

[7] E. J. CANDÈS, X. LI, Y. MA, AND J. WRIGHT, *Robust principal component analysis?*, Journal of ACM, 58 (2011), pp. 1–37.

[8] E. J. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, 9 (2009), pp. 717–772.

[9] E. J. CANDÈS, J. ROMBERG, AND T. TAO, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, 52 (2006), pp. 489–509.

TABLE 4.7

Comparison of the algorithms for solving (4.1) *with* $\rho = 0.001$ *and uniform kernel on the* 512×512 *Lena image. Four different* μ_g*'s were used:* $S1, S2, S3, S4 : 0.1, 1, 10, 100$*.* $FISTA(100) = 53737.69$*,* $FISTA(500) = 47173.53$*,* $FISTA(1000) = 45808.14$*.* $ISTA(500) = 56901.31$*,* $ISTA(2500) = 52154.67$*,* $ISTA(5000) = 50455.10$

solver	iter	CPU	iter	CPU	iter	CPU
FISTA	100	57	500	286	1000	574
ISTA	1377	783	>5000	>2858	>5000	>2858
ADAL	1377	909	>5000	>3319	>5000	>3319
SADAL	690	473	>5000	>3428	>5000	>3428
ALM	690	485	>5000	>3510	>5000	>3510
FALM	71	59	351	299	701	602
ALM-S1	1252 (0)	1045	>5000 (0)	>4174	>5000 (0)	>4174
ALM-S2	690 (0)	577	>5000 (2)	>4180	>5000 (2)	>4180
ALM-S3	127 (0)	105	4547 (2152)	3927	>5000 (2389)	>4319
ALM-S4	16 (0)	13	3860 (3612)	3498	>5000 (4703)	>4527
FALM-S1	96 (0)	89	477 (2)	453	977 (502)	934
FALM-S2	71 (0)	66	699 (155)	669	1319 (775)	1260
FALM-S3	29 (0)	26	246 (140)	234	750 (644)	718
FALM-S4	9 (0)	8	259 (234)	246	763 (738)	730
ISTA	500	283	2500	1425	5000	2858
ADAL	501	327	2501	1653	5001	3319
SADAL	252	172	1252	858	2502	1718
ALM	252	176	1252	880	2502	1758
ALM-S1	456 (0)	380	2274 (0)	1901	4547 (0)	3796
ALM-S2	252 (0)	210	1252 (0)	1047	2502 (1)	2093
ALM-S3	47 (0)	38	229 (0)	191	609 (167)	521
ALM-S4	7 (0)	6	27 (0)	22	227 (177)	206

[10] E. J. CANDÈS AND T. TAO, *The power of convex relaxation: near-optimal matrix completion*, IEEE Trans. Inform. Theory, 56 (2009), pp. 2053–2080.

[11] P. L. COMBETTES, *Solving monotone inclusions via compositions of nonexpansive averaged operators*, Optimization, 53 (2004), pp. 475–504.

[12] P. L. COMBETTES AND JEAN-CHRISTOPHE PESQUET, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 564–574.

[13] P. L. COMBETTES AND V. R. WAJS, *Signal recovery by proximal forward-backward splitting*, SIAM Journal on Multiscale Modeling and Simulation, 4 (2005), pp. 1168–1200.

[14] D. DONOHO, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.

[15] J. DOUGLAS AND H. H. RACHFORD, *On the numerical solution of the heat conduction problem in 2 and 3 space variables*, Transactions of the American Mathematical Society, 82 (1956), pp. 421–439.

[16] J. ECKSTEIN AND D. P. BERTSEKAS, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.

[17] J. ECKSTEIN AND B. F. SVAITER, *A family of projective splitting methods for sum of two maximal monotone operators*, Math. Program. Ser. B, 111 (2008), pp. 173–199.

[18] M. FIGUEIREDO AND R. NOWAK, *An EM algorithm for wavelet-based image restoration*, IEEE Transactions on Image Processing, 12 (2003), pp. 906–916.

[19] J. FRIEDMAN, T. HASTIE, AND R. TIBSHIRANI, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, (2007).

[20] 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.

[21] R. GLOWINSKI AND P. LE TALLEC, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia, Pennsylvania, 1989.

[22] D. GOLDFARB AND S. MA, *Fast multiple splitting algorithms for convex optimization*, tech. report, Department of IEOR,

Columbia University. Preprint available at http://arxiv.org/abs/0912.4570, 2009.

[23] ———, *Convergence of fixed point continuation algorithms for matrix rank minimization*, Foundations of Computational Mathematics, 11 (2011), pp. 183–210.

[24] D. Goldfarb and K. Scheinberg, *Fast first-order methods for composite convex optimization with line search*, preprint, (2011).

[25] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

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

[27] B. S. He, L.-Z. Liao, D. Han, and H. Yang, *A new inexact alternating direction method for monotone variational inequalities*, Math. Program., 92 (2002), pp. 103–118.

[28] B. S. He, M. Tao, M. Xu, and X. Yuan, *Alternating direction based contraction method for generally separable linearly constrained convex programming problems*, Optimization-online: http://www.optimization-online.org/DB_HTML/2009/11/2465.html, (2009).

[29] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, Journal of optimization theory and applications, 106 (2000), pp. 337–356.

[30] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from a few entries*, IEEE Trans. on Info. Theory, 56 (2010), pp. 2980–2998.

[31] K. C. Kiwiel, C. H. Rosa, and A. Ruszczynski, *Proximal decomposition via alternating linearization*, SIAM J. Optimization, 9 (1999), pp. 668–689.

[32] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[33] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Mathematical Programming Series A, 128 (2011), pp. 321–353.

[34] J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization, 20 (2009), pp. 336–356.

[35] R. D. C. Monteiro and B. F. Svaiter, *Iteration-complexity of block-decomposition algorithms and the alternating minimization augmented Lagrangian method*, Preprint, (2010).

[36] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$*, Dokl. Akad. Nauk SSSR, 269 (1983), pp. 543–547.

[37] ———, *Introductory lectures on convex optimization*, 87 (2004), pp. xviii+236. A basic course.

[38] ———, *Smooth minimization for non-smooth functions*, Math. Program. Ser. A, 103 (2005), pp. 127–152.

[39] ———, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 2007/76, (2007).

[40] D. H. Peaceman and H. H. Rachford, *The numerical solution of parabolic elliptic differential equations*, SIAM Journal on Applied Mathematics, 3 (1955), pp. 28–41.

[41] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Review, 52 (2010), pp. 471–501.

[42] K. Scheinberg, S. Ma, and D. Goldfarb, *Sparse inverse covariance selection via alternating linearization methods*, in Proceedings of the Neural Information Processing Systems (NIPS), 2010.

[43] J. E. Spingarn, *Partial inverse of a monotone operator*, Appl. Math. Optim., 10 (1983), pp. 247–265.

[44] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optimization, 6 (2010), pp. 615–640.

[45] P. Tseng, *Further applications of a splitting algorithm to decomposition in variational inequalities and convex programming*, Mathematical Programming, 48 (1990), pp. 249–263.

[46] ———, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control and Optimization, 29 (1991), pp. 119–138.

[47] ———, *On accelerated proximal gradient methods for convex-concave optimization*, submitted to SIAM J. Optim., (2008).

[48] M. Wainwright, P. Ravikumar, and J. Lafferty, *High-dimensional graphical model selection using ℓ_1-regularized logistic regression*, NIPS, 19 (2007), pp. 1465–1472.

[49] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Mathematical Programming Computation, 2 (2010), pp. 203–230.

[50] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1 problems in compressive sensing*, SIAM Journal on Scientific Computing, 33 (2011), pp. 250–278.

[51] M. Yuan and Y. Lin, *Model selection and estimation in the Gaussian graphical model*, Biometrika, 94 (2007), pp. 19–35.

[52] X. Yuan, *Alternating direction methods for sparse covariance selection*, (2009). Preprint available at http://www.optimization-online.org/DB_HTML/2009/09/2390.html.

[53] X. Yuan and J. Yang, *Sparse and low rank matrix decomposition via alternating direction methods*, preprint, (2009).