Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 4, pp. 619–644, October 2015

UNDERSTANDING THE CONVERGENCE OF THE ALTERNATING DIRECTION METHOD OF MULTIPLIERS: THEORETICAL AND COMPUTATIONAL PERSPECTIVES*

Jonathan Eckstein and Wang Yao

Abstract: The alternating direction of multipliers (ADMM) is a form of augmented Lagrangian algorithm that has experienced a renaissance in recent years due to its applicability to optimization problems arising from "big data" and image processing applications, as well as the relative ease with which it may be implemented in parallel and distributed computational environments. While it is easiest to describe the method as an approximate version of the classical augmented Lagrangian algorithm, using one pass of block coordinate minimization to approximately minimize the augmented Lagrangian at each iteration, the known convergence proofs bear no discernible relationship to this description. In this largely tutorial paper, we try to give an accessible version of the "operator splitting" version of the ADMM convergence proof, first developing some techniques that we use to analyze a simple variant of the classical augmented Lagrangian method. We assume relatively little prior knowledge of convex analysis. Using two dissimilar classes of application problems, we also computationally compare the ADMM to some algorithms that do indeed work by approximately minimizing the augmented Lagrangian. The results suggest that the ADMM is different from such methods not only in its convergence analysis but also in its computational behavior.

Key words: *alternating direction method of multipliers (ADMM)*

Mathematics Subject Classification: *90C25, 49M27*

1 Introduction

The *alternating direction method of multipliers* (ADMM) is a convex optimization algorithm first proposed in 1975 [16, page 69] and first analyzed in the early 1980's [14, 15]. It has attracted renewed attention recently due to its applicability to various machine learning and image processing problems. In particular,

- It appears to perform reasonably well on these relatively recent applications, better than when applied to traditional operations research problems such as minimum-cost network flows.
- The ADMM can take advantage of the structure of these problems, which involve optimizing sums of fairly simple but sometimes nonsmooth convex functions.
- Extremely high accuracy is not usually a requirement for these applications, reducing the impact of the ADMM's tendency toward slow "tail convergence".

*This work was partially supported by National Science Foundation grant CCF-1115638.

- Depending on the application, it is often relatively easy to implement the ADMM in a distributed-memory, parallel manner. This property is important for "big data" problems in which the entire problem dataset may not fit readily into the memory of a single processor.

The recent survey article [6] describes the ADMM from the perspective of machine learning applications; another, older survey is contained in the doctoral thesis [8].

Although it can be developed in a slightly more general form — see for example [6] — the following problem formulation is sufficient for most applications of the ADMM:

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

Here, M is an $m \times n$ matrix, sometimes assumed to have full column rank, and f and g are convex functions on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. We let f and g take not only values in $\mathbb{R}$ but also the value $+\infty$, so that constraints may be "embedded" in them, in the sense that if $f(x) = \infty$ or $g(Mx) = \infty$, then the point x is considered to be infeasible for (1.1).

By appropriate use of infinite values for f or g, a very wide range of convex problems may be modeled through (1.1). To make the discussion more concrete, however, we now describe a simple illustrative example that fits readily into the form (1.1) without use of infinite function values, and resembles in basic structure many of the applications responsible for the resurgence of interest in the ADMM: the "lasso" or "compressed sensing" problem. This problem takes the form

$$\min_{x\in\mathbb{R}^n} \tfrac{1}{2}\|Ax - b\|^2 + \nu \|x\|_1 , \tag{1.2}$$

where A is a $p \times n$ matrix, $b \in \mathbb{R}^p$, and $\nu > 0$ is a given scalar parameter. The idea of this model is find an approximate solution to the linear equations $Ax = b$, but with a preference for making the solution vector $x \in \mathbb{R}^n$ sparse; the larger the value of the parameter ν, the more the model prefers sparsity of the solution versus accuracy of solving $Ax = b$. While this model has some limitations in terms of finding sparse near-solutions to $Ax = b$, it serves as a good example application for the ADMM, simply by taking $f(x) = \frac{1}{2}\|Ax - b\|^2$, $M = I$, and $g(x) = \nu \|x\|_1$. Many other now-popular applications have a similar general form, but may use more complicated norms in place of $\|\cdot\|_1$; for example, in some applications, x is treated as a matrix, and one uses the nuclear norm (the sum of singular values) in the objective to try to induce x to have low rank.

We now describe the classical augmented Lagrangian method and the ADMM for (1.1). First, note that we can rewrite (1.1), introducing an additional decision variable vector $z \in \mathbb{R}^m$, as the following problem over $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$:

$$\begin{array}{ll} \min & f(x) + g(z) \\ \text{ST} & Mx = z. \end{array} \tag{1.3}$$

For this formulation, the classical augmented Lagrangian algorithm, which we will discuss in more depth in Section 3, takes the form

$$(x^{k+1}, z^{k+1}) \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \left\{ f(x) + g(z) + \langle \lambda^k, Mx - z\rangle + \tfrac{c_k}{2}\|Mx - z\|^2 \right\} \tag{1.4}$$

$$\lambda^{k+1} = \lambda^k + c_k(Mx^{k+1} - z^{k+1}). \tag{1.5}$$

Here, $\{\lambda^k\}$ is a sequence of estimates of the Lagrange multipliers of the constraints $Mx = z$, while $\{(x^k, z^k)\}$ is a sequence of estimates of the solution vectors x and z, and $\{c_k\}$ is a

sequence of positive scalar parameters bounded away from 0. Throughout this article, $\langle a, b\rangle$ denotes the usual Euclidean inner product $a^\top b$.

In this setting, the standard augmented Lagrangian algorithm (1.4)-(1.5) is not very attractive because the minimizations of f and g in the subproblem (1.4) are strongly coupled through the term $\frac{c_k}{2}\|Mx - z\|^2$, and hence the subproblems are not likely to be easier to solve than the original problem (1.1).

The alternating direction method of multipliers (ADMM) for (1.1) or (1.3) takes the following form, for some scalar parameter $c > 0$:

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + g(z^k) + \langle \lambda^k, Mx - z^k\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{1.6}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{z\in\mathbb{R}^m} \left\{ f(x^{k+1}) + g(z) + \langle \lambda^k, Mx^{k+1} - z\rangle + \tfrac{c}{2}\left\|Mx^{k+1} - z\right\|^2 \right\} \tag{1.7}$$

$$\lambda^{k+1} = \lambda^k + c(Mx^{k+1} - z^{k+1}). \tag{1.8}$$

Clearly, the constant terms $g(z^k)$ and $f(x^{k+1})$, as well as some other constants, may be dropped from the respective minimands of (1.6) and (1.7). Unlike the classical augmented Lagrangian method, the ADMM essentially decouples the functions f and g, since (1.6) requires only minimization of a quadratic perturbation of f, and (1.7) requires only minimization of a quadratic perturbation of g. In many situations, this decoupling makes it possible to exploit the individual structure of the f and g so that each of (1.6) and (1.7) may be computed in an efficient and perhaps highly parallel manner.

Given the form of the two algorithms, it is natural to view the ADMM (1.6)-(1.8) as an approximate version of the classical augmented Lagrangian method (1.4)-(1.5) in which a single pass of "Gauss-Seidel" block minimization substitutes for full minimization of the augmented Lagrangian $L_c(x, z, \lambda^k)$, where we define

$$L_c(x, z, \lambda) = f(x) + g(z) + \langle \lambda, Mx - z\rangle + \tfrac{c}{2}\|Mx - z\|^2. \tag{1.9}$$

That is, we substitute minimization with respect to x followed by minimization with respect to z for the joint minimization with respect to x and z required by (1.4). This viewpoint was in fact the motivation for the original proposal for the ADMM in [16]. Curiously, however, this interpretation does not seem to play a role in any known convergence proof for the ADMM, and there is no known general way of quantifying how closely one iteration of the two calculations (1.6)-(1.7) approaches the joint minimization (1.4).

There are two fundamental approaches to proving the convergence of the ADMM, each based on a different form of two-way *splitting*, that is, expressing a mapping as the sum of two simpler mappings. One approach is at its core based on a splitting of the classical Lagrangian function

$$L(x, z, \lambda^k) = f(x) + g(z) + \langle \lambda^k, Mx - z\rangle \tag{1.10}$$

of the problem (1.3). This approach dates back to [14], and essentially equivalent analyses were given later, for example, in the textbook [5] and the appendix to the survey article [6]. The drawback of this approach is that the analysis is quite lengthy and its structure can be hard to discern.

The second convergence proof approach for the ADMM dates back to [15], and is based on combining a splitting of the dual functional of (1.3) with some operator splitting theory originating in [21]. This approach is elaborated and expanded in the popular reference [10], and it can be made simpler and more intuitive by using ideas from [20]. Once understood, the operator-splitting perspective yields considerable insight into the convergence of the ADMM.

Its existing presentations, however, require considerable background in convex analysis and can thus be somewhat unapproachable.

This article has two goals: the first is to attempt to convey the understanding inherent in the second, operator-splitting approach to proving convergence of the ADMM, but assuming only basic knowledge of convex analysis. To this end, some mathematical rigor will be sacrificed in the interest of clarifying the key concepts. The intention is to make most of the benefits of a deeper understanding of the method available to a widest possible audience — in essence, the idea is to simplify the analysis of [10, 15] into a form that does not require extensive convex analysis background, developing the required concepts in the simplest required form as the analysis proceeds.

The second goal of this article is to present some recent computational results which illustrate that the convergence theory of the ADMM also seems to have a practical dimension. In particular, these results suggest that, despite outward appearances and its original motivation, significant insight is missing if we primarily think of the ADMM as an approximate version of the the classical augmented Lagrangian algorithm using block coordinate minimization for the subproblems. In particular, using two different problems classes, we computationally compare the ADMM to methods that are in fact approximate augmented Lagrangian algorithms with block-coordinate-minimization subproblem solvers, and show that they have significantly different behavior.

The remainder of this article is structured as follows: Section 2 summarizes some necessary background material, which may be largely skipped by readers familiar with convex analysis. Section 3 presents the classic augmented Lagrangian method for convex problems as an application of nonexpansive algorithmic mappings, and then Section 4 applies the same analytical techniques to the ADMM. Section 5 presents the computational experiments, while Section 6 offers some concluding remarks and presents some avenues for future research. The material in Sections 2-4 is not fundamentally new, and can be inferred from the references mentioned in each section; the only intended contribution is in the manner of presentation. The new computational work underscores the ideas found in the theory and raises some issues for future investigation.

A longer, online companion version of this article containing omitted proofs and additional figures may be found in [13]. This expanded version also contains a section describing a product-space extension of the ADMM to more than two blocks of variables. This extension is also described in Section 5.2 of the textbook [4].

2 Background Material from Convex Analysis

This section summarizes some basic analytical results that will help to structure and clarify the following analysis. Proofs are only included when they are simple and insightful. More complicated proofs will be either sketched, referred to the online companion version [13], or omitted. Most of the material here can be readily found (often in more general form) in [24] or in textbooks such as [2, 3]. The main insight meant to be provided by this section is that there is a strong relationship between convex functions and *nonexpansive mappings*, that is, functions $N : \mathbb{R}^n \to \mathbb{R}^n$ with $\|N(x) - N(x')\| \leq \|x - x'\|$ for all $x, x' \in \mathbb{R}^n$. Furthermore, in the case of particular kinds of convex functions d arising from dual formulations of optimization problems, there is a relatively simple way to evaluate the nonexpansive map N that corresponds to d.

Definition 2.1. Given any function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a vector $v \in \mathbb{R}^n$ is said to be a

subgradient of f at $x \in \mathbb{R}^n$ if

$$f(x') \geq f(x) + \langle v, x' - x \rangle \qquad \forall\, x' \in \mathbb{R}^n. \tag{2.1}$$

We use the notation $\partial f(x)$ for set of all subgradients of f at x.

Essentially, v is a subgradient of f at x if the affine function $a(\,\cdot\,)$ given by $a(x') = f(x) + \langle v, x' - x \rangle$ underestimates f throughout $\mathbb{R}^n$. Furthermore, it can be seen immediately from the definition that x^* is a global minimizer of f if and only if $0 \in \partial f(x^*)$. If f is differentiable at x and convex, then $\partial f(x)$ is the singleton set $\{\nabla f(x)\}$ (this fact is intuitive to visualize, but the proof is nontrivial; see for example [3, 24]). Convergence proofs for convex optimization algorithms very frequently use the following property of the subgradient:

Lemma 2.2. *The subgradient mapping of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ has the following monotonicity property: given any $x, x', v, v' \in \mathbb{R}^n$ such that $v \in \partial f(x)$ and $v' \in \partial f(x')$, we have*

$$\langle x - x', v - v' \rangle \geq 0. \tag{2.2}$$

Proof. From the subgradient inequality (2.1), we have $f(x') \geq f(x) + \langle v, x' - x \rangle$ and $f(x) \geq f(x') + \langle v', x - x' \rangle$, which we may add to obtain $f(x) + f(x') \geq f(x) + f(x') + \langle v - v', x' - x \rangle$. Canceling the identical terms $f(x) + f(x')$ from both sides and rearranging yields (2.2). □

From this monotonicity property, it is straightforward to derive a basic "decomposition" property of subgradients, a basic building block of the ADMM convergence proof we will explore in the next section:

Lemma 2.3. *Given any function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, a vector $z \in \mathbb{R}^n$, and a scalar $c > 0$ there is at most one way to write $z = x + cv$, where $v \in \partial f(x)$.*

Proof. Suppose that $y = x + cv = x' + cv'$ where $v \in \partial f(x)$ and $v' \in \partial f(x')$. Then simple algebra yields $x - x' = c(v' - v)$ and hence $\langle x - x', v - v' \rangle = -c\|v - v'\|^2$. But since Lemma 2.2 asserts that $\langle x - x', v - v' \rangle \geq 0$, we must have $v = v'$ and hence $x = x'$. □

Next, we give conditions under which a decomposition of any $z \in \mathbb{R}^n$ into $x + cv$, $v \in \partial f(x)$ must exist, in addition to having to be unique. First, we state a necessary background result:

Lemma 2.4. *Suppose $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^n \to \mathbb{R}$ are convex, and g is continuously differentiable. Then $\partial(f + g)(x) = \partial f(x) + \nabla g(x) = \{ y + \nabla g(x) \mid y \in \partial f(x) \}$ for all $x \in \mathbb{R}^n$.*

The proof of this result is somewhat more involved than one might assume, so we omit it; see for example [24, Theorem 23.8 and 25.1] or [3, Propositions 4.2.2 and 4.2.4]. However, the result itself should be reasonably intuitive: adding a differentiable convex function g to the convex function f simply translates the set of subgradients at each point x by $\nabla g(x)$.

Definition 2.5. A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called *proper* if it is not everywhere $+\infty$, that is, there exists $x \in \mathbb{R}^n$ such that $f(x) \in \mathbb{R}$. Such a function is called *closed* if the set $\{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid t \geq f(x)\}$ is closed.

By a straightforward argument, it may be seen that closedness of f is equivalent to the *lower semicontinuity* condition that $f(\lim_{k\to\infty} x^k) \leq \liminf_{k\to\infty} f(x^k)$ for all convergent sequences $\{x^k\} \subset \mathbb{R}^n$; see [24, Theorem 7.1] or [3, Proposition 1.2.2]. We are now ready to state a simple condition guaranteeing that a decomposition of the form given in Lemma 2.3 must exist for every $z \in \mathbb{R}^n$.

Proposition 2.6. *If $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, proper, and convex, then for any scalar $c > 0$, each point $z \in \mathbb{R}^n$ can be written in* exactly *one way as $x + cv$, where $v \in \partial f(x)$.*

A sketch of the proof of this proposition may be found in the online companion [13]. This result is a special case of Minty's theorem [23]. We now show that, because of the monotonicity of the subgradient map, we may construct a nonexpansive mapping on $\mathbb{R}^n$ corresponding to any closed proper convex function:

Proposition 2.7. *Given a closed proper convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and a scalar $c > 0$, the mapping $N_{cf} : \mathbb{R}^n \to \mathbb{R}^n$ given by*

$$N_{cf}(z) = x - cv \qquad \text{where } x, v \in \mathbb{R}^n \text{ are such that } v \in \partial f(x) \text{ and } x + cv = z \tag{2.3}$$

is everywhere uniquely defined and nonexpansive. The fixed points of N_{cf} are precisely the minimizers of f.

Proof. From Proposition 2.6, there is exactly one way to express any $z \in \mathbb{R}^n$ as $x + cv$, so $N_{cf}(z)$ exists and is uniquely defined for all $z \in \mathbb{R}^n$. To show that N_{cf} is nonxpansive, consider any any $z, z' \in \mathbb{R}^n$, respectively expressed as $z = x + cv$ and $z' = x' + cv'$ with $v \in \partial f(x)$ and $v' \in \partial f(x')$. Then we write

$$\begin{aligned} \|z - z'\|^2 &= \|x + cv - (x' + cv')\|^2 = \|x - x'\|^2 + 2c\langle x - x', v - v'\rangle + c^2 \|v - v'\|^2 \\ \|N_{cf}(z) - N_{cf}(z')\|^2 &= \|x - cv - (x' - cv')\|^2 = \|x - x'\|^2 - 2c\langle x - x', v - v'\rangle + c^2 \|v - v'\|^2. \end{aligned}$$

Therefore, $\|N_{cf}(z) - N_{cf}(z')\|^2 = \|z - z'\|^2 - 4c\langle x - x', v - v'\rangle$. By monotonicity of the subgradient, the inner product in the last term is nonnegative, leading to the conclusion that $\|N_{cf}(z) - N_{cf}(z')\| \leq \|z - z'\|$. With z, x, and v as above, we note that

$$N_{cf}(z) = z \quad \Leftrightarrow \quad x - cv = x + cv \quad \Leftrightarrow \quad v = 0 \quad \Leftrightarrow \quad x \text{ minimizes } f \text{ and } z = x.$$

This equivalence proves the assertion regarding fixed points. □

We also include one additional standard consequence of closedness which will be needed later. The proof is straightforward and is included in the online companion [13].

Lemma 2.8. *If f is a closed convex function and $\{x^k\}, \{v^k\} \subset \mathbb{R}^n$ are convergent sequences such that $v^k \in \partial f(x^k)$ for all k, then $\lim_{k\to\infty} v^k \in \partial f\big(\lim_{k\to\infty} x^k\big)$.*

3 The Classical Augmented Lagrangian Method and Nonexpansiveness

We now review the theory of the classical augmented Lagrangian method from the standpoint of nonexpansive mappings; although couched in slightly different language here, this analysis originated with [26, 27]. Consider an optimization problem

$$\begin{array}{ll} \min & h(x) \\ \text{ST} & Ax = b, \end{array} \tag{3.1}$$

where $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed proper convex, and $Ax = b$ represents an arbitrary set of linear inequality constraints. This formulation can subsume the problem (1.3) through the change of variables $(x, z) \to x$, letting $b = 0 \in \mathbb{R}^m$ and $A = [M \;\; -I]$, and defining h appropriately.

Using standard Lagrangian duality[1], the dual function of (3.1) is

$$q(\lambda) = \min_{x \in \mathbb{R}^n} \{h(x) + \langle \lambda, Ax - b \rangle\}. \tag{3.2}$$

The dual problem to (3.1) is to maximize $q(\lambda)$ over $\lambda \in \mathbb{R}^m$. Defining $d(\lambda) = -q(\lambda)$, we may equivalently formulate the dual problem as minimizing the function d over $\mathbb{R}^m$. We now consider the properties of the negative dual function d:

Lemma 3.1. *The function d is closed and convex. If the vector $x^* \in \mathbb{R}^n$ attains the minimum in* (3.2) *for some given $\lambda \in \mathbb{R}^m$, then $b - Ax^* \in \partial d(\lambda)$.*

Proof. We have that $d(\lambda) = \max_{x \in \mathbb{R}^n} \{-h(x) - \langle \lambda, Ax - b \rangle\}$, that is, that d is the pointwise maximum, over all $x \in \mathbb{R}^n$, of the affine (and hence convex) functions of λ given by $a_x(\lambda) = -h(x) - \langle \lambda, Ax - b \rangle$. Thus, d is convex, and

$$\{(\lambda, t) \in \mathbb{R}^m \times \mathbb{R} \mid t \geq d(\lambda)\} = \bigcap_{x \in \mathbb{R}^n} \{(\lambda, t) \in \mathbb{R}^m \times \mathbb{R} \mid t \geq -h(x) - \langle \lambda, Ax - b \rangle\}.$$

Each of the sets on the right of this relation is defined by a single non-strict linear inequality on (λ, t) and is thus closed. Since any intersection of closed sets is also closed, the set on the left is also closed and therefore d is closed.

Next, suppose that x^* attains the minimum in (3.2). Then, for any $\lambda' \in \mathbb{R}^m$,

$$\begin{aligned} q(\lambda') = \min_{x \in \mathbb{R}^n} \{h(x) + \langle \lambda', Ax - b \rangle\} &\leq h(x^*) + \langle \lambda', Ax^* - b \rangle \\ &= h(x^*) + \langle \lambda, Ax^* - b \rangle + \langle \lambda' - \lambda, Ax^* - b \rangle \\ &= q(\lambda) + \langle Ax^* - b, \lambda' - \lambda \rangle. \end{aligned}$$

Negating this relation yields $d(\lambda') \geq d(\lambda) + \langle b - Ax^*, \lambda' - \lambda \rangle$ for all $\lambda' \in \mathbb{R}^m$, meaning that $b - Ax^* \in \partial d(\lambda)$. □

Since d is closed, we may deduce that if it is also proper (finite for at least one choice of $\lambda \in \mathbb{R}^m$), then Proposition 2.6 guarantees that any $\mu \in \mathbb{R}^m$ can be decomposed into $\mu = \lambda + cv$, where $v \in \partial d(\lambda)$. A particularly interesting property of functions defined like d is that this decomposition may be computed in a manner not much more burdensome than evaluating d itself:

Proposition 3.2. *Given any $\mu \in \mathbb{R}^m$, consider the problem*

$$\min_{x \in \mathbb{R}^n} \{h(x) + \langle \mu, Ax - b \rangle + \tfrac{c}{2} \|Ax - b\|^2\}. \tag{3.3}$$

If $\bar{x}$ is an optimal solution to this problem, setting $\lambda = \mu + c(A\bar{x} - b)$ and $v = b - A\bar{x}$ yields $\lambda, v \in \mathbb{R}^m$ such that $\lambda + cv = \mu$ and $v \in \partial d(\lambda)$, where $d = -q$ is as defined above.

Proof. Appealing to Lemma 2.4, $\bar{x}$ is optimal for (3.3) if there exists $w \in \partial h(\bar{x})$ such that

$$\begin{aligned} 0 &= w + \tfrac{\partial}{\partial x} \left[\langle \mu, Ax - b \rangle + \tfrac{c}{2} \|Ax - b\|^2 \right]_{x = \bar{x}} \\ &= w + A^\top \mu + cA^\top (A\bar{x} - b) \end{aligned}$$

[1] Rockafellar's parametric conjugate duality framework [25] gives deeper insight into the material presented here, but in order to focus on the main topic at hand, we use a form of duality that should be more familiar to most readers.

$$= w + A^\top\big(\mu + c(A\bar{x} - b)\big) = w + A^\top\lambda,$$

where $\lambda = \mu + c(A\bar{x} - b)$ as above. Using Lemma 2.4 once again, this condition means that $\bar{x}$ attains the minimum in (3.2), and hence Lemma 3.1 implies that $v = b - A\bar{x} \in \partial d(\lambda)$. Finally, we note that $\lambda + cv = \mu + c(A\bar{x} - b) + c(b - A\bar{x}) = \mu$. □

For a general closed proper convex function, computing the decomposition guaranteed to exist by Proposition 2.6 may be much more difficult than simply evaluating the function. The main message of Proposition 3.2 is that for functions d obtained from the duals of convex programming problems like (3.1), this may not be the case — it may be possible to compute the decomposition by solving a minimization problem that could well be little more difficult than required to evaluate d itself. To avoid getting bogged down in convex-analytic details that would distract from our main goals, we will not directly address here exactly when it is guaranteed that an optimal solution to (3.3) exists; we will simply assume that the problem has a solution. This situation is also easily verified in most applications.

Since the decomposition $\mu = \lambda + cv$, $v \in \partial d(\lambda)$, can be evaluated by solving (3.3), the same calculation allows us to evaluate the nonexpansive mapping N_{cd}, using the notation of Section 2: specifically, using the notation of Proposition 3.2, we have

$$N_{cd}(\mu) = \lambda - cv = \mu + c(A\bar{x} - b) - c(b - A\bar{x}) = \mu + 2c(A\bar{x} - b).$$

We know that N_{cd} is a nonexpansive map, and that any fixed point of N_{cd} is a minimizer of d, that is, an optimal solution to the dual problem of (3.1). It is thus tempting to consider simply iterating the map $\lambda^{k+1} = N_{cd}(\lambda^k)$ recursively starting from some arbitrary $\lambda^0 \in \mathbb{R}^m$ in the hope that $\{\lambda^k\}$ would converge to a fixed point. Now, if we knew that the Lipschitz modulus of N_{cd} were less than 1, linear convergence of $\lambda^{k+1} = N_{cd}(\lambda^k)$ to such a fixed point would be elementary to establish. However, we only know that the Lipschitz modulus of N_{cd} is *at most* 1. Thus, if we were to iterate the mapping $\lambda^{k+1} = N_{cd}(\lambda^k)$, it is possible the iterates could simply "orbit" at fixed distance from the set of fixed points without converging.

We now appeal to a long-established result from fixed point theory, which asserts that if one "blends" a small amount of the identity with a nonexpansive map, convergence is guaranteed and such "orbiting" cannot occur:

Theorem 3.3. *Let $T : \mathbb{R}^m \to \mathbb{R}^m$ be nonexpansive, that is, $\|T(y) - T(y')\| \le \|y - y'\|$ for all $y, y' \in \mathbb{R}^m$. Let the sequence $\{\rho_k\} \subset (0, 2)$ be such that $\inf_k\{\rho_k\} > 0$ and $\sup_k\{\rho_k\} < 2$. If the mapping T has any fixed points and $\{y^k\}$ conforms to the recursion*

$$y^{k+1} = \tfrac{\rho_k}{2}T(y^k) + (1 - \tfrac{\rho_k}{2})y^k, \tag{3.4}$$

then $\{y^k\}$ converges to a fixed point of T.

A proof of this theorem may be found in the online companion [13], and is a simplification of the proof of the slightly stronger (and also infinite-dimensional) result given in Theorem 5.14 of [2]. Note that the case $\rho_k = 1$ in the above result corresponds to taking the simple average $\frac{1}{2}T + \frac{1}{2}I$ of T with the identity map. The $\rho_k \equiv 1$ case of Theorem 3.3 was proven, in an infinite-dimensional setting, as long ago as 1955 [19].

Combining Proposition 3.2 and Theorem 3.3 yields a convergence proof for a form of augmented Lagrangian algorithm:

Proposition 3.4. *Consider a problem of the form* (3.1) *and a scalar* $c > 0$. *Suppose, for some scalar sequence* $\{\rho_k\} \subset (0, 2)$ *with the property that* $\inf_k\{\rho_k\} > 0$ *and* $\sup_k\{\rho_k\} < 2$, *the sequences* $\{x^k\} \subset \mathbb{R}^n$ *and* $\{\lambda^k\} \subset \mathbb{R}^m$ *evolve according to the recursions*

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^m} \left\{ h(x) + \langle \lambda^k, Ax - b\rangle + \tfrac{c}{2}\|Ax - b\|^2 \right\} \tag{3.5}$$

$$\lambda^{k+1} = \lambda^k + \rho_k c(Ax^{k+1} - b). \tag{3.6}$$

If the dual problem to (3.1) *possesses an optimal solution, then* $\{\lambda^k\}$ *converges to one of them, and all limit points of* $\{x^k\}$ *are optimal solutions to* (3.1).

Proof. The optimization problem solved in (3.5) is just the one in Proposition 3.2 with λ^k substituted for μ. Therefore,

$$N_{cd}(\lambda^k) = \lambda^k + c(Ax^{k+1} - b) - c(b - Ax^{k+1}) = \lambda^k + 2c(Ax^{k+1} - b),$$

and consequently

$$\tfrac{\rho_k}{2}N_{cd}(\lambda^k) + (1 - \tfrac{\rho_k}{2})\lambda^k = \tfrac{\rho_k}{2}\big(\lambda^k + 2c(Ax^{k+1} - b)\big) + (1 - \tfrac{\rho_k}{2})\lambda^k = \lambda^k + \rho_k c(Ax^{k+1} - b),$$

meaning that $\lambda^{k+1} = \frac{\rho_k}{2}N_{cd}(\lambda^k) + (1 - \frac{\rho_k}{2})\lambda^k$. An optimal solution to the dual of (3.1) is a minimizer of d, and hence a fixed point of N_{cd} by Proposition 2.7. Theorem 3.3 then implies that $\{\lambda^k\}$ converges to such a fixed point.

Since λ^k is converging and $\{\rho_k\}$ is bounded away from 0, we may infer from (3.6) that $Ax^k - b \to 0$. If x^* is any optimal solution to (3.1), we have from $Ax^* = b$ and the optimality of x^{k+1} in (3.5) that

$$\begin{aligned} h(x^{k+1}) + \langle \lambda^{k+1}, Ax^{k+1} - b\rangle + \tfrac{c}{2}\left\|Ax^{k+1} - b\right\|^2 &\le h(x^*) + \langle \lambda^{k+1}, Ax^* - b\rangle + \tfrac{c}{2}\|Ax^* - b\|^2 \\ &= h(x^*), \end{aligned}$$

that is, $h(x^k) + \langle \lambda^{k-1}, Ax^k - b\rangle + \frac{c}{2}\|Ax^k - b\|^2 \le h(x^*)$ for all k. Let x^∞ be any limit point of $\{x^k\}$, and $\mathcal{K}$ denote a sequence of indices such that $x^k \to_{\mathcal{K}} x^\infty$. Since $Ax^k - b \to 0$, we have $Ax^\infty = b$. Since h is closed, it is lower semicontinuous, so

$$h(x^\infty) \le \liminf_{\substack{k\to\infty\\ k\in\mathcal{K}}} h(x^k) = \liminf_{\substack{k\to\infty\\ k\in\mathcal{K}}} \left\{ h(x^k) + \langle \lambda^{k-1}, Ax^k - b\rangle + \tfrac{c}{2}\|Ax^k - b\| \right\} \le h(x^*),$$

and so x^∞ is also optimal for (3.1). □

There are two differences between (3.5)-(3.6) and the augmented Lagrangian method as it is typical presented for (3.1). First, many versions of the augmented Lagrangian method omit the parameters $\{\rho_k\}$, and are thus equivalent to the special case $\rho_k \equiv 1$. Second, it is customary to let the parameter c vary from iteration to iteration, replacing it with a sequence of parameters $\{c_k\}$, with $\inf_k\{c_k\} > 0$. In this case, one cannot appeal directly to the classical fixed-point result of Theorem 3.3 because the operators $N_{c_k d}$ being used at each iteration may be different. However, essentially the same convergence proof as for Theorem 3.3 may still be employed, because the operators $N_{c_k d}$ all have exactly the same fixed points, namely the set of optimal dual solutions. This observation, in a more general form, is the essence of the convergence analysis of the *proximal point algorithm* [26, 27]; see also [10].

4 Analyzing the ADMM through Compositions of Nonexpansive Mappings

We now describe the convergence theory of the ADMM (1.6)-(1.8) using the same basic tools described in the previous section. Essentially, this material is a combination and simplified overview of analyses of the ADMM and related methods in such references as [21], [15], [20], [8], and [11, Section 1]; a similar treatment may be found in [10].

To begin, we consider the dual of the problem (1.3), namely to maximize the function

$$q(\lambda) = \min_{\substack{x\in\mathbb{R}^n \\ z\in\mathbb{R}^m}} \{f(x) + g(z) + \langle \lambda, Mx - z\rangle\} \tag{4.1}$$

$$= \min_{x\in\mathbb{R}^n} \{f(x) + \langle \lambda, Mx\rangle\} + \min_{z\in\mathbb{R}^m} \{g(z) - \langle \lambda, z\rangle\} \tag{4.2}$$

$$= q_1(\lambda) + q_2(\lambda), \tag{4.3}$$

where we define

$$q_1(\lambda) = \min_{x\in\mathbb{R}^n} \{f(x) + \langle \lambda, Mx\rangle\} \qquad q_2(\lambda) = \min_{z\in\mathbb{R}^m} \{g(z) - \langle \lambda, z\rangle\}. \tag{4.4}$$

Defining $d_1(\lambda) = -q_1(\lambda)$ and $d_2(\lambda) = -q_2(\lambda)$, the dual of (1.3) is thus equivalent to minimizing the sum of the two convex functions d_1 and d_2:

$$\min_{\lambda\in\mathbb{R}^m} d_1(\lambda) + d_2(\lambda) \tag{4.5}$$

The functions d_1 and d_2 have the same general form as the dual function d of the previous section; therefore, we can apply the same analysis:

Lemma 4.1. *The functions d_1 and d_2 are closed and convex. Given some $\lambda \in \mathbb{R}^m$, suppose $x^* \in \mathbb{R}^n$ attains the first minimum in* (4.4)*. Then $-Mx^* \in \partial d_1(\lambda)$. Similarly, if $z^* \in \mathbb{R}^p$ attains the second minimum in* (4.4)*, then $z^* \in \partial d_2(\lambda)$.*

Proof. We simply apply Lemma 3.1 twice. In the case of d_1, we set $h = f$, $A = M$, and $b = 0$, while in the case of d_2, we set $h = g$, $A = -I$, and $b = 0$. □

We now consider some general properties of problems of the form (4.5), that is, minimizing the sum of two convex functions.

Lemma 4.2. *A sufficient condition for $\lambda^* \in \mathbb{R}^m$ to solve* (4.5) *is that*

$$\text{there exists } v^* \in \partial d_1(\lambda^*) \text{ such that } -v^* \in \partial d_2(\lambda^*). \tag{4.6}$$

Proof. For all $\lambda \in \mathbb{R}^m$ the subgradient inequality gives

$$d_1(\lambda) \geq d_1(\lambda^*) + \langle v^*, \lambda - \lambda^*\rangle$$
$$d_2(\lambda) \geq d_2(\lambda^*) + \langle -v^*, \lambda - \lambda^*\rangle.$$

Adding these two inequalities produces the relation $d_1(\lambda) + d_2(\lambda) \geq d_1(\lambda^*) + d_2(\lambda^*)$ for all $\lambda \in \mathbb{R}^m$, so λ^* must be optimal for (4.5). □

Under some mild regularity conditions that are met in most practical applications, (4.6) is also *necessary* for optimality; however, to avoid a detour into subdifferential calculus, we will simply assume that this condition holds for at least one optimal solution λ^* of (4.5).

Fix any constant $c > 0$ and assume that d_1 and d_2 are proper, that is, each is finite for at least one value of the argument λ. Since d_1 and d_2 are closed and convex, we may conclude that they correspond to nonexpansive maps as shown in Section 2. These maps are given by

$$\begin{aligned} N_{cd_1}(y) &= \lambda - cv \qquad \text{where } \lambda, v \in \mathbb{R}^m \text{ are such that } v \in \partial d_1(\lambda) \text{ and } \lambda + cv = y \\ N_{cd_2}(y) &= \mu - cw \qquad \text{where } \mu, w \in \mathbb{R}^m \text{ are such that } w \in \partial d_2(\mu) \text{ and } \mu + cw = y. \end{aligned}$$

N_{cd_1} and N_{cd_2} are both nonexpansive by the analysis given in Section 3. It follows that their functional composition is also nonexpansive, that is, $\left\|N_{cd_1}\big(N_{cd_2}(y)\big) - N_{cd_1}\big(N_{cd_2}(y')\big)\right\| \leq \|N_{cd_2}(y) - N_{cd_2}(y')\| \leq \|y - y'\|$ for all $y, y' \in \mathbb{R}^m$. We now consider the set of fixed points of this composed mapping:

Lemma 4.3. *The set of fixed points of the composition $N_{cd_1} \circ N_{cd_2}$ of N_{cd_1} and N_{cd_2} is*

$$\operatorname{fix}(N_{cd_1} \circ N_{cd_2}) = \{\lambda + cv \mid v \in \partial d_2(\lambda), -v \in \partial d_1(\lambda)\}.$$

Proof. Take any $y \in \mathbb{R}^m$ and write it as $y = \lambda + cv$, where $v \in \partial d_2(\lambda)$; Lemma 4.1 and Proposition 2.6 imply that this must be possible. Then $N_{cd_2}(y) = \lambda - cv$. Now, again using Lemma 4.1 and Proposition 2.6, we know that $N_{cd_2}(y) = \lambda - cv$ can be written in exactly one way as $\mu + cw$, where $w \in \partial d_1(\mu)$. Then $N_{cd_1}(N_{cd_2}(y)) = \mu - cw$, and thus y is a fixed point of $N_{cd_1} \circ N_{cd_2}$ if and only if

$$\mu - cw = y = \lambda + cv. \tag{4.7}$$

Adding $\mu + cw = N_{cd_2}(y) = \lambda - cv$ to (4.7), we obtain $\mu = \lambda$. Substituting $\mu = \lambda$ into (4.7), we also obtain $w = -v$. □

Thus, finding a fixed point of $N_{cd_1} \circ N_{cd_2}$ is essentially the same as finding two vectors $\lambda, v \in \mathbb{R}^m$ satisfying $v \in \partial d_2(\lambda)$, $-v \in \partial d_1(\lambda)$, and thus an optimal solution to the dual problem (4.5). Finding such a point essentially results in finding a solution to the problem (1.3), as we now show. We first make one standard definition regarding the solution, and a regularity assumption that will play the role of standard constraint qualification (a condition guaranteeing the existence of a Lagrange multiplier).

Definition 4.4. A *KKT point* for problem (1.3) is some $\big((x^*, z^*), \lambda^*\big) \in (\mathbb{R}^n \times \mathbb{R}^n) \times \mathbb{R}^m$ such that

1. x^* minimizes $f(x) + \langle \lambda^*, Mx \rangle$ with respect to x
2. z^* minimizes $g(z) - \langle \lambda^*, z \rangle$ with respect to z
3. $Mx^* = z^*$.

Assumption 4.1. All subgradients of the function $d_1(\lambda) = \min_{x \in \mathbb{R}^n}\{f(x) + \langle \lambda, Mx \rangle\}$ at each point $\lambda \in \mathbb{R}^m$ take the form $-M\bar{x}$, where $\bar{x}$ attains the stated minimum over x in (4.4).

We note that Lemma 3.1 guarantees that $-M\bar{x}$ is subgradient of d_1 at λ whenever $\bar{x}$ minimizes $f(x) + \langle \lambda^*, Mx \rangle$ over x, so the assumption merely says that there are no other subgradients. This assumption is guaranteed by a variety of more standard regularity conditions that are met in most practical applications. However, to avoid getting sidetracked in convex-analytic details, we keep the assumption as stated.

Lemma 4.5. *If* $\big((x^*, z^*), \lambda^*\big)$ *is a KKT point, then* (x^*, z^*) *is an optimal solution of* (1.3) *and* x^* *is an optimal solution of* (1.1). *If Assumption 4.1 holds and* $z^* \in \partial d_2(\lambda^*)$ *and* $-z^* \in \partial d_1(\lambda^*)$, *then there exists* $x^* \in \mathbb{R}^n$ *such that* $\big((x^*, z^*), \lambda^*\big)$ *is a KKT point.*

A proof of this lemma may be found in the online companion [13].

At this point, an obvious strategy is to try to apply Theorem 3.3 to finding a fixed point of $N_{cd_1} \circ N_{cd_2}$, leading essentially immediately to a solution of (4.5). That is, we perform the iteration

$$y^{k+1} = \tfrac{\rho_k}{2} N_{cd_1}\big(N_{cd_2}(y^k)\big) + (1 - \tfrac{\rho_k}{2}) y^k \tag{4.8}$$

for some sequence of parameters $\{\rho_k\}$ with $\inf_k\{\rho_k\} > 0$ and $\sup_k\{\rho_k\} < 2$.

We now show that this procedure turns out to be equivalent to the ADMM. We show this equivalence in two stages, first by converting (4.8) into a more computationally explicit but still generic form, and then specializing this form to our particular form of d_1 and d_2. To begin the first stage of this analysis, consider the point y^k at the beginning of such an iteration, and express it as $\lambda^k + cv^k$, for $v^k \in \partial d_2(\lambda^k)$. Then, to apply the recursion (4.8), we proceed as follows:

1. Applying the map N_{cd_2} produces the point $\lambda^k - cv^k$.
2. Next, we express $\lambda^k - cv^k$ as $\mu^k + cw^k$, for $w^k \in \partial d_1(\mu^k)$.
3. We then calculate

$$\begin{aligned} y^{k+1} = \tfrac{\rho_k}{2} N_{cd_1}(N_{cd_2}(y^k)) + (1 - \tfrac{\rho_k}{2}) y^k &= \tfrac{\rho_k}{2} N_{cd_1}(\mu^k + cw^k) + (1 - \tfrac{\rho_k}{2})(\lambda^k + cv^k) \\ &= \tfrac{\rho_k}{2}(\mu^k - cw^k) + (1 - \tfrac{\rho_k}{2})(\lambda^k + cv^k). \end{aligned}$$

The last expression for y^{k+1} above may be simplified as follows: since we know $\lambda^k - cv^k = \mu^k + cw^k$, we have by simple rearrangement that $\lambda^k - \mu^k = c(v^k + w^k)$. Thus,

$$\begin{aligned} y^{k+1} &= \tfrac{\rho_k}{2}(\mu^k - cw^k) + (1 - \tfrac{\rho_k}{2})(\lambda^k + cv^k) \\ &= \tfrac{\rho_k}{2}\big(\mu^k - cw^k - (\lambda^k + cv^k)\big) + \lambda^k + cv^k \\ &= \tfrac{\rho_k}{2}\big(\mu^k - \lambda^k - c(v^k + w^k)\big) + \lambda^k + cv^k \\ &= \tfrac{\rho_k}{2}\big(2(\mu^k - \lambda^k)\big) + \lambda^k + cv^k = \rho_k \mu^k + (1 - \rho_k)\lambda^k + cv^k, \end{aligned}$$

where the second-to-last equality follows from $\lambda^k - \mu^k = c(v^k + w^k)$. Thus, in terms of the sequences $\{\lambda^k\}$, $\{\mu^k\}$, $\{v^k\}$, and $\{w^k\}$, the algorithm may be expressed as follows, starting from some arbitrary $\lambda^0, v^0 \in \mathbb{R}^m$:

$$\text{Find } \mu^k, w^k: \qquad w^k \in \partial d_1(\mu^k) \qquad \mu^k + cw^k = \lambda^k - cv^k \tag{4.9}$$

$$\text{Find } \lambda^{k+1}, v^{k+1}: \quad v^{k+1} \in \partial d_2(\lambda^{k+1}) \quad \lambda^{k+1} + cv^{k+1} = \rho_k \mu^k + (1 - \rho_k)\lambda^k + cv^k. \tag{4.10}$$

We know in this situation that the vectors $y^k = \lambda^{k+1} + cv^{k+1}$ should converge to a point of the form $y^* = \lambda^* + cv^*$, where $v^* \in \partial d_2(\lambda^*)$ and $-v^* \in \partial d_1(\lambda)$, meaning that λ^* solves (4.5). As an aside, this pattern of computations is an example of (generalized) *Douglas-Rachford splitting* [8, 10, 21], whose relationship to the ADMM was first shown in [15]. Douglas-Rachford splitting is the same basic "engine" behind the convergence of several other popular algorithms, including the progressive hedging method for stochastic programming [28].

We now move to the second stage of our derivation, in which we specialize (4.9)-(4.10) above to the particular functions $d_1 = -q_1$ and $d_2 = -q_2$, where q_1 and q_2 are defined

by (4.4). Conveniently, we can simply apply Proposition 3.2: in the case of (4.9) with $d_1 = -q_1$, applying Proposition 3.2 with $h = f$, $A = M$, $b = 0$, and $\mu = \lambda^k - cv^k$ yields the computation

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k - cv^k, Mx\rangle + \tfrac{c}{2}\left\|Mx\right\|^2 \right\} \tag{4.11}$$

$$\mu^k = \lambda^k - cv^k + cMx^{k+1} \tag{4.12}$$

$$w^k = -Mx^{k+1}. \tag{4.13}$$

Now let us consider (4.10), with $d_2 = -q_2$: again applying Proposition 3.2, but now with $h = g$, $A = -I$, $b = 0$, and $\mu = \rho_k\mu^k + (1-\rho_k)\lambda^k + cv^k$, yields the computation

$$z^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ g(z) + \langle \rho_k\mu^k + (1-\rho_k)\lambda^k + cv^k, -z\rangle + \tfrac{c}{2}\left\|-z\right\|^2 \right\} \tag{4.14}$$

$$\lambda^{k+1} = \rho_k\mu^k + (1-\rho_k)\lambda^k + cv^k - cz^{k+1} \tag{4.15}$$

$$v^{k+1} = z^{k+1}. \tag{4.16}$$

We now simplify the system of recursions (4.11)-(4.16). First, we note that the sequence $\{w^k\}$ does not appear explicitly except in (4.13), so we may simply eliminate it. Second, using (4.16), we may substitute $v^k = z^k$ throughout, yielding

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k - cz^k, Mx\rangle + \tfrac{c}{2}\left\|Mx\right\|^2 \right\} \tag{4.17}$$

$$\mu^k = \lambda^k - cz^k + cMx^{k+1} \tag{4.18}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ g(z) - \langle \rho_k\mu^k + (1-\rho_k)\lambda^k + cz^k, z\rangle + \tfrac{c}{2}\left\|z\right\|^2 \right\} \tag{4.19}$$

$$\lambda^{k+1} = \rho_k\mu^k + (1-\rho_k)\lambda^k + cz^k - cz^{k+1}. \tag{4.20}$$

Next, consider (4.17). Adding the constant $\frac{c}{2}\|z^k\|^2$ to the minimand and completing the square yields the equivalent computation

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k, Mx\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{4.21}$$

From (4.18), we have $\mu^k = \lambda^k + c(Mx^{k+1} - z^k)$, hence

$$\begin{aligned} \rho_k\mu^k + (1-\rho_k)\lambda^k + cz^k &= \rho_k\big(\lambda^k + c(Mx^{k+1} - z^k)\big) + (1-\rho_k)\lambda^k + cz^k \\ &= \lambda^k + \rho_k cMx^{k+1} + (1-\rho_k)cz^k \\ &= \lambda^k + c\big(\rho_k Mx^{k+1} + (1-\rho_k)z^k\big). \end{aligned} \tag{4.22}$$

Thus, the minimand in (4.19) may be written $g(z) - \langle \lambda^k + c\big(\rho_k Mx^{k+1} + (1-\rho_k)z^k\big), z\rangle + \frac{c}{2}\left\|z\right\|^2$. Similarly to the derivation of (4.21), adding the constant $\frac{c}{2}\|\rho_k Mx^{k+1} + (1-\rho_k)z^k\|^2$ and completing the square yields the equivalent minimand

$$g(z) - \langle \lambda^k, z\rangle + \tfrac{c}{2}\left\|\rho_k Mx^{k+1} + (1-\rho_k)z^k - z\right\|^2,$$

and substituting (4.22) into (4.20) produces $\lambda^{k+1} = \lambda^k + c\big(\rho_k Mx^{k+1} + (1-\rho_k)z^k - z^{k+1}\big)$. Summarizing, the recursions (4.11)-(4.16) collapse to

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k, Mx\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{4.23}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{z \in \mathbb{R}^p} \left\{ g(z) - \langle \lambda^k, z \rangle + \tfrac{c}{2} \left\| \rho_k M x^{k+1} + (1-\rho_k) z^k - z \right\|^2 \right\} \tag{4.24}$$

$$\lambda^{k+1} = \lambda^k + c\big(\rho_k M x^{k+1} + (1-\rho_k) z^k - z^{k+1}\big). \tag{4.25}$$

In the special case $\rho_k \equiv 1$, these recursions reduce exactly to the ADMM (1.6)-(1.8), except for some immaterial constant terms in the minimands.

This derivation contains the essence of the operator-splitting convergence theory of the ADMM method: it is an application of the fixed-point algorithm of Theorem 3.3 to the nonexpansive mapping $N_{cd_1} \circ N_{cd_2}$. We take advantage of this observation in the following proposition:

Proposition 4.6. *Let the constant $c > 0$ be given and suppose that Assumption 4.1 holds and there exists a KKT point for problem* (1.3)*. Then if the sequences $\{x^k\} \subset \mathbb{R}^n$, $\{z^k\} \subset \mathbb{R}^m$, and $\{\lambda^k\} \subset \mathbb{R}^m$ conform to the recursions* (4.23)-(4.25)*, where $\inf_k\{\rho_k\} > 0$ and $\sup_k\{\rho_k\} < 2$, we have that $\lambda^k \to \lambda^*$, $z^k \to z^*$, and $Mx^k \to Mx^* = z^*$, where $\big((x^*, z^*), \lambda^*\big)$ is some KKT point.*

Proof. The development above shows that the recursions (4.23)-(4.25) are equivalent to the sequence $y^k = \lambda^k + cv^k = \lambda^k + cz^k$ being produced by the recursion (4.8), which by Theorem 3.3 converges to a fixed point of the operator $N_{cd_1} \circ N_{cd_2}$, if one exists. The existence of such a fixed point is implied by Lemma 4.5 and the hypothesis that a KKT point exists. Therefore y^k is convergent to some fixed point y^∞ of $N_{cd_1} \circ N_{cd_2}$, which by Lemma 4.3 is of the form $y^\infty = \lambda^\infty + cz^\infty$, where $z^\infty \in \partial d_2(\lambda^\infty)$ and $-z^\infty \in \partial d_1(\lambda^\infty)$. By the assumption regarding d_1, there exists some x^∞ such that $-Mx^\infty = -z^\infty$, that is, $Mx^\infty = z^\infty$.

Consider the mapping $P_{cd_2} = \frac{1}{2} N_{cd_2} + \frac{1}{2} I$. Since N_{cd_2} is nonexpansive, P_{cd_2} is certainly continuous. We have $P_{cd_2}(y^\infty) = \frac{1}{2}(\lambda^\infty - cz^\infty) + \frac{1}{2}(\lambda^\infty + cz^\infty) = \lambda^\infty$ and similarly $P_{cd_2}(y^k) = \lambda^k$ since $z^k = v^k \in \partial d_2(\lambda^k)$, and so by the continuity of P_{cd_2} we must have $\lambda^k = P_{cd_2}(y^k) \to P_{cd_2}(y^\infty) = \lambda^\infty$ and thus $z^k = \frac{1}{c}(y^k - \lambda^k) \to \frac{1}{c}(y^\infty - \lambda^\infty) = z^\infty$. We may also rewrite (4.25) as

$$\lambda^{k+1} - \lambda^k = c\rho_k(Mx^{k+1} - z^k) + c(z^k - z^{k+1}). \tag{4.26}$$

The quantity on the left of (4.26) converges to 0 since $\{\lambda^k\}$ is convergent, while the last term on the right converges to 0 since $\{z^k\}$ is convergent. Since ρ_k is bounded away from 0, it follows that $Mx^{k+1} \to z^\infty = Mx^\infty$. □

There are a number of observations worth making at this point:

1. In most results regarding the convergence of the ADMM, Assumption 4.1 is typically replaced by more natural, verifiable assumptions regarding f and M; these assumptions imply that Assumption 4.1 holds, and are met in almost all practical applications. The version of the analysis given above was chosen to minimize the amount of convex-analytical background required.

2. The ADMM's convergence analysis is not based on approximating the nonexpansive mapping $N_{cd} = N_{c(d_1+d_2)}$ underlying the classical method of multipliers as applied to problem (1.3), but on exact evaluation of the related but fundamentally different nonexpansive mapping $N_{cd_1} \circ N_{cd_2}$. From a theoretical standpoint, therefore, the ADMM is not an approximate version of the classical augmented Lagrangian method. Note that approximate versions of the ADMM are possible, as described for example in [10].

3. Varying the parameter c from one iteration to another is more problematic for the ADMM than for the classical augmented Lagrangian method, because changing c shifts the set of fixed points of the map $N_{cd_1} \circ N_{cd_2}$. This phenomenon makes proving the convergence of the ADMM with nonconstant c much more difficult, although some partial results have been obtained; see for example [17] (or [1], for a specific application).

4. The technique of composing N_{cd_1} with N_{cd_2} to obtain a nonexpansive mapping whose fixed points are closely related to the minimizers of $d_1 + d_2$ does not have an obvious extension to the case of more than two functions: for example, if we consider three convex functions d_1, d_2, d_3, the fixed points of $N_{cd_1} \circ N_{cd_2} \circ N_{cd_3}$ are not closely related to the minima of $d_1 + d_2 + d_3$. This phenomenon helps explain the relative difficulty of proving the convergence of the ADMM when extended in a direct manner to more than two blocks of variables.

5. Here, we are essentially following the operator splitting approach to analyzing the convergence of the ADMM, as first explored in [15]. When one follows this approach, the overrelaxation parameters ρ_k appear in a different way than in the classical augmented Lagrangian method. Applying the augmented Lagrangian method (3.5)-(3.6) to the problem formulation (1.3) results in a version of the method (1.4)-(1.5) in which (1.5) is amended to
$$\lambda^{k+1} = \lambda_k + \rho_k c_k (Mx^{k+1} - z^{k+1}). \tag{4.27}$$
On the other hand, ρ_k is used in a different way in (4.23)-(4.25): the overrelaxation parameter appears in the "target" value $\rho_k Mx^{k+1} + (1 - \rho_k)z^k$ for z in (4.24) and similarly in the following multiplier update (4.25).

6. As mentioned earlier, there is another approach to proving the convergence of the ADMM, which dates back to [14] and is based on a splitting of the Lagrangian function (1.10), rather than a splitting of the dual function. This path of analysis leads to a convergence proof which, instead of being based on the Fejér monotonicity of $\{y^k\} = \{\lambda^k + cz^k\}$ to the set $\operatorname{fix}(N_{cd_1} \circ N_{cd_2}) = \{\lambda + cv \mid v \in \partial d_2(\lambda), -v \in \partial d_1(\lambda)\}$, establishes Fejér monotonicity of $\{(\lambda^k, cz^k)\}$ to the set $\{(\lambda, cv) \mid v \in \partial d_2(\lambda), -v \in \partial d_1(\lambda)\}$, which is a slightly different condition. When this approach is taken, the overrelaxation parameters ρ_k naturally appear only in the multiplier update, and one obtains a method consisting of (1.6), (1.7), and (4.27), that is,
$$x^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ f(x) + \langle \lambda^k, Mx \rangle + \tfrac{c}{2} \left\| Mx - z^k \right\|^2 \right\} \tag{4.28}$$
$$z^{k+1} \in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{ g(z) - \langle \lambda^k, z \rangle + \tfrac{c}{2} \left\| Mx^{k+1} - z \right\|^2 \right\} \tag{4.29}$$
$$\lambda^{k+1} = \lambda_k + \rho_k c_k (Mx^{k+1} - z^{k+1}). \tag{4.30}$$
The analysis in [14] proves convergence of this method, where ρ_k is constant and in the range $\big(0, (1 + \sqrt{5})/2\big)$; the same proof can be easily generalized to the case $0 < \liminf_{k\to\infty} \rho_k \le \limsup_{k\to\infty} \rho_k < (1 + \sqrt{5})/2$. This is a narrower range for $\{\rho_k\}$ than one obtains for the operator-splitting-based analysis, and furthermore it does not appear that there is any way to derive the method (4.28)-(4.30) from the operator-splitting approach presented in this paper.[2] Technically, this result means

[2]Specifically, to derive (4.28) from the operator-splitting framework, one needs $z^k \in \partial d_2(\lambda^k)$, but choosing $\rho_k \neq 1$ in the previous iteration's execution of (4.30) violates this condition.

there are actually two distinct families of ADMM algorithms, one derived from the operator-splitting framework and the other derived from Lagrangian splitting. These two families coincide in the case $\rho_k \equiv 1$, in which case (4.23)-(4.25) and (4.28)-(4.30) become identical sequences of calculations. Which form of overrelaxation is preferable in practice appears to depend on the application. This paper has explored the operator-splitting framework, because its conceptual structure is easier to understand and yields more intuition about the algorithm. However, some of the same insights can be gleaned from the Lagranian-splitting framework. For example, in the Lagrangian-splitting approach, it is apparent that changing the value of c shifts the set of fixed points $\{(\lambda, cv) \mid v \in \partial d_2(\lambda), -v \in \partial d_1(\lambda)\}$ that attracts the iterates, which helps explain the difficulties of proving convergence for variable c. And again, the proof is not based on one pass of the computations (1.6)-(1.7) measurably approximating overall minimization of the augmented Lagrangian, but on establishing Fejér monotonicity to some set related to the optimality conditions $v \in \partial d_2(\lambda)$, $-v \in \partial d_1(\lambda)$.

5 Computational Comparison of the ADMM and Approximate Augmented Lagrangian Methods

5.1 Comparison Algorithms

The preceding sections have attempted to give some theoretical insight into the the differences between the classical augmented Lagrangian algorithm and the ADMM. Both may be viewed as applications of the same basic result about convergence of iterated nonexpansive mappings, but in significantly different ways: despite outward appearances, the ADMM is not from the existing theoretical perspective an approximate version of the augmented Lagrangian method. We now explore the same topic from a computational perspective by comparing the ADMM to methods that really do perform measurable approximate minimization of the augmented Lagrangian.

One difficulty with approximate augmented Lagrangian algorithms is that they can involve approximation criteria — that is, conditions for deciding whether one is sufficiently close to having found a minimum of subproblems such as (1.4) or (3.5) — that can be hard to test in practice. Notable exceptions are the methods proposed in [9,12]. Here, we will focus on the "relative error" method of [12], which appears to have the best practical performance in the prior literature; we will not review or explain the analysis of these methods here.

In the context of problem (3.1), each iteration of the algorithmic template of [12] takes the following general form, where $\sigma \in [0, 1)$ is a fixed parameter:

1. Find $x^{k+1}, y^{k+1} \in \mathbb{R}^n$ such that

$$y^k \in \partial_x \left[h(x) + \langle \lambda^k, Ax - b \rangle + \tfrac{c_k}{2} \|Ax - b\|^2 \right]_{x=x^{k+1}} \tag{5.1}$$

$$\tfrac{2}{c_k} \left| \langle w^k - x^{k+1}, y^k \rangle \right| + \|y^k\|^2 \leq \sigma \left\| Ax^{k+1} - b \right\|^2 \tag{5.2}$$

2. Perform the updates

$$\lambda^{k+1} = \lambda^k + c_k(Ax^{k+1} - b) \tag{5.3}$$

$$w^{k+1} = w^k - c_k y^k. \tag{5.4}$$

In (5.1), ∂_x denotes the set of subgradients with respect to x, with λ treated as a fixed parameter. The conditions (5.1)-(5.2) constitute an approximate minimization with respect

to x, in the sense that if x^{k+1} exactly minimizes the augmented Lagrangian as in (3.5), then we can take $y^k = 0$ and (5.2) is immediately satisfied because its left-hand side is zero and its right-hand side must be nonnegative. However, (5.1)-(5.2) can tolerate a less exact minimization, with a subgradient that is not exactly 0. Note that $\{y^k\}, \{w^k\} \subset \mathbb{R}^n$ are auxiliary sequences that are not needed in the exact form of the algorithm.

Adapting this pattern to the specific problem (1.1), we split y^k into $(y_x^k, y_z^k) \in \mathbb{R}^n \times \mathbb{R}^m$ and similarly split w^k into $(w_x^k, w_z^k) \in \mathbb{R}^n \times \mathbb{R}^m$, obtaining the following recursive conditions, where we use the augmented Lagrangian notation L_c defined in (1.9):

$$(y_x^k, y_z^k) \in \partial_{(x,z)} L_c(x^{k+1}, z^{k+1}, \lambda^k) \tag{5.5}$$

$$\tfrac{2}{c_k}\left|\langle w_x^k - x^{k+1}, y_x^k\rangle + \langle w_z^k - z^{k+1}, y_z^k\rangle\right| + \|y_x^k\|^2 + \|y_z^k\|^2 \le \sigma\left\|Mx^{k+1} - z^{k+1}\right\|^2 \tag{5.6}$$

$$\lambda^{k+1} = \lambda^k + c_k(Mx^{k+1} - z^{k+1}) \tag{5.7}$$

$$w_x^{k+1} = w_x^k - c_k y_x^k \tag{5.8}$$

$$w_z^{k+1} = w_z^k - c_k y_z^k. \tag{5.9}$$

We now consider how to perform the approximate minimization embodied in (5.5)-(5.6), keeping in mind that f or g (or both) might be nonsmooth; for example, in the lasso problem (1.2), the function f is smooth, but g is nonsmooth.

As suggested at least as long ago as in [16], one way to attempt to approximately minimize the augmented Lagrangian is to use a block Gauss-Seidel algorithm which alternately minimizes with respect to x and then with respect to z, that is (looping over i),

$$x^{k,i+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{L_{c_k}(x, z^{k,i}, p^k)\right\} \tag{5.10}$$

$$z^{k,i+1} \in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{L_{c_k}(x^{k,i+1}, z, p^k)\right\} \tag{5.11}$$

We choose this approach because of its resemblance to the ADMM. Convergence of such a scheme is well known for smooth convex functions, but for nonsmooth functions the relevant literature is quite limited. Fortunately, a result of Tseng [30] establishes subsequential convergence of such a scheme in most cases of interest, including the example problem (1.2). In integrating (5.10)-(5.11) into the overall scheme (5.5)-(5.9), we note that (5.11) guarantees a zero subgradient exists with respect to z, so we may take $y_z^k = 0$ for all k. If we take $w_z^0 = 0$, then $w_z^k = 0$ for all k, and we may drop the sequences $\{y_z^k\}$ and $\{w_z^k\}$ from the algorithm and omit the subscripts x from $\{y_x^k\}$ and $\{w_x^k\}$. Thus, we obtain the algorithm shown in Figure 1, which we call "GS-RE" (for Gauss-Seidel Relative Error). The last two steps of this method are not theoretically necessary, but make the method more efficient in practice: rather than starting the inner loop from an arbitrary point, we initialize it from the result of the prior outer iteration.

In the prior literature, another suggested approach to approximately minimizing the augmented Lagrangian is the diagonal-quadratic approximation method, or DQA [29], which uses the following inner loop, where $\tau \in (0, 1)$ is a scalar parameter:

$$x^{k,i+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{L_{c_k}(x, \bar{z}^{k,i}, \lambda^k)\right\}$$

$$z^{k,i+1} \in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{L_{c_k}(\bar{x}^{k,i}, z, \lambda^k)\right\}$$

$$\bar{x}^{k,i+1} = \tau x^{k,i+1} + (1-\tau)\bar{x}^{k,i}$$

$$\bar{z}^{k,i+1} = \tau x^{k,i+1} + (1-\tau)\bar{z}^{k,i}.$$

Repeat for $i = 0, 1, \ldots$

$$x^{k,i+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ L_{c_k}(x, z^{k,i}, \lambda^k) \right\}$$

$$z^{k,i+1} \in \operatorname*{Arg\,min}_{z \in \mathbb{R}^m} \left\{ L_{c_k}(x^{k,i+1}, z, \lambda^k) \right\}$$

Until there exists $y^k \in \partial_x L_{c_k}(x^{k,i}, z^{k,i}, \lambda^k)$ with

$$\tfrac{2}{c_k} \left| \langle w^k - x^{k,i}, y^k \rangle \right| + \|y^k\|^2 \leq \sigma \|M x^{k,i} - z^{k,i}\|^2$$

$$\lambda^{k+1} = \lambda^k + \rho_k c_k (M x^{k,i} - z^{k,i})$$

$$w^{k+1} = w^k - c_k y^k$$

$$x^{k+1,0} = x^{k,i}$$

$$z^{k+1,0} = z^{k,i}$$

Figure 1: The GS-RE algorithm.

In this case, convergence is obtained for the sequence $\{(\bar{x}^{k,i}, \bar{z}^{k,i})\}$ (over the inner index i), and it is not possible to streamline the algorithm by asserting that some subcomponent of the subgradient must be zero. We therefore obtain the slightly more complicated algorithm shown in Figure 2, which we refer to as "DQA-RE".

Using prototype MATLAB implementations, we compared the GS-RE and DQA-RE algorithms to the standard ADMM with overrelaxation (4.23)-(4.25). We also included "exact" versions of the Gauss-Seidel and DQA methods in which the respective inner loops of GS-RE and DQA-RE are iterated until they obtain a very small augmented Lagrangian subgradient (with norm not exceeding some small fixed parameter δ) or a large iteration limit is reached. We call these methods "GS" and "DQA", respectively. The exact Gauss-Seidel approach was described, along with the original proposal for the ADMM, in [16, pages 68-69], while the exact DQA method was proposed in [29].

5.2 Computational Tests for the Lasso Problem

We performed tests on two very different problem classes. The first class consisted of six instances of the lasso problem (1.2) derived from standard cancer DNA microarray datasets [7]. These instances have very "wide" observation matrices A, with the number of rows p ranging from 42 to 102, but the number of columns n ranging from 2000 to 6033. As mentioned in the introduction, the lasso problem (1.2) may be reduced to the form (1.1) by taking

$$f(x) = \tfrac{1}{2} \|Ax - b\|^2 \qquad g(z) = \nu \|z\|_1 \qquad M = I.$$

The x-minimization step (4.23) of the ADMM — or equivalently the first minimization in the inner loop of GS-RE or DQA-RE — then reduces to solving a system of linear equations involving the matrix $A^\top A + cI$, namely (in the ADMM case)

$$x^{k+1} = (A^\top A + cI)^{-1} \left(A^\top b + c z^k - \lambda^k \right). \tag{5.12}$$

As long as the parameter c remains constant throughout the algorithm, the matrix $A^\top A + cI$ need only be factored once at the outset, and each iteration only involves backsolving the system using this factorization. Furthermore, for "wide" matrices such as those in the dataset

Repeat for $i = 0, 1, \ldots$

$x^{k,i+1} \in \operatorname{Arg\,min}_{x \in \mathbb{R}^n} \left\{ L_{c_k}(x, \bar{z}^{k,i}, \lambda^k) \right\}$

$z^{k,i+1} \in \operatorname{Arg\,min}_{z \in \mathbb{R}^m} \left\{ L_{c_k}(\bar{x}^{k,i}, z, \lambda^k) \right\}$

$\bar{x}^{k,i+1} = \tau x^{k,i+1} + (1-\tau)\bar{x}^{k,i}$

$\bar{z}^{k,i+1} = \tau x^{k,i+1} + (1-\tau)\bar{z}^{k,i}$

Until there exists $(y_x^k, y_z^k) \in \partial_{(x,z)} L_{c_k}(\bar{x}^{k,i}, \bar{z}^{k,i}, \lambda^k)$ with

$\frac{2}{c_k} \left| \langle w_x^k - \bar{x}^{k,i}, y_x^k \rangle + \langle w_z^k - \bar{z}^{k,i}, y_z^k \rangle \right| + \|y_x^k\|^2 + \|y_z^k\|^2 \leq \sigma \left\| M\bar{x}^{k,i} - \bar{z}^{k,i} \right\|^2$

$\lambda^{k+1} = \lambda^k + \rho_k c_k (M\bar{x}^{k,i} - \bar{z}^{k,i})$

$w_x^{k+1} = w_x^k - c_k y_x^k$

$w_z^{k+1} = w_z^k - c_k y_z^k$

$\bar{x}^{k+1,0} = \bar{x}^{k,i}$

$\bar{z}^{k+1,0} = \bar{z}^{k,i}$.

Figure 2: The DQA-RE algorithm.

of [7], one may use the Sherman-Morrison-Woodbury inversion formula to substitute a factorization of the much smaller matrix $I + \frac{1}{c} AA^\top$ for the factorization of $A^\top A + cI$; each iteration then requires a much faster, lower-dimensional backsolve operation, along with some additional matrix multiplications and simple vector arithmetic. The z-minimization (4.24) — or equivalently the second minimization in the inner loop of the GS-RE or DQA-RE — reduces to a very simple componentwise "vector shrinkage" operation. In the ADMM case, this shrinkage operation is

$$z_i^{k+1} = \operatorname{sgn}(x_i^{k+1} + \tfrac{1}{c}\lambda_i^k) \max\left\{0, \left|x_i^{k+1} + \tfrac{1}{c}\lambda_i^k\right| - \tfrac{\nu}{c}\right\} \qquad i = 1, \ldots, n. \tag{5.13}$$

Since $M = I$, the multiplier update (4.25) is also a very simple componentwise calculation, so the backsolves required to implement the x-minimization dominate the time required for each ADMM iteration.

Applying the GS-RE and DQA-RE algorithms to the lasso problem results in x- and z-minimizations respectively very similar to (5.12) and (5.13), these algorithms' multiplier updates are nearly identical to the ADMM, and their inner-loop termination tests require only a few inner-product calculations. Thus, the x-minimization step also dominates the per-iteration time for all the algorithms tested. For a more detailed discussion of applying the ADMM to the lasso problem (1.2), see for example [6].

Our computational tests used a standard method for normalizing the matrix A, scaling each column to have norm 1, and also scaling b to have norm 1; after performing this normalization, the problem scaling parameter ν was set to $0.1\|A^\top b\|_\infty$.

After some experimentation, we settled on the following algorithm parameter settings, which seemed to work well for all the algorithms tested: regarding the penalty parameters, we chose $c = 10$ for the ADMM and $c_k \equiv 10$ for the other algorithms. For overrelaxation, we chose $\rho_k \equiv \rho = 1.95$ for all the algorithms. For the DQA inner loop, we set $\tau = 0.5$.

For all algorithms, the initial Lagrange muliplier estimate λ^0 was the zero vector; all

Table 1: Lasso problems: number of outer iterations / multiplier adjustments

Problem Instance	**ADMM**	**GS-RE**	**DQA-RE**	**GS**	**DQA**
Brain	814	219	272	277	281
Colon	1,889	213	237	237	211
Leukemia	1,321	200	218	239	212
Lymphoma	1,769	227	253	236	209
Prostate	838	213	238	237	210
SRBCT	2,244	216	232	254	226

primal variables were also initialized to zero. For all algorithms, the overall termination criterion was

$$\operatorname{dist}_\infty\Big(0, \partial_x\Big[\tfrac{1}{2}\,\|Ax-b\|^2 + \nu\,\|x\|_1\Big]_{x=x^k}\Big) \le \epsilon, \tag{5.14}$$

where $\operatorname{dist}_\infty(t,S) = \inf\{\|t-s\|_\infty \mid s \in S\}$, and ϵ is a tolerance parameter we set 10^{-6}. In the relative-error augmented Lagrangian methods, GS-RE and DQA-RE, we set the parameter σ to 0.99. For the "exact" versions of these methods, GS and DQA respectively, we terminated the inner loop when

$$\operatorname{dist}_\infty\big(0, \partial_{(x,z)}\big[L_{c_k}(x^{k,i}, z^{k,i}, \lambda^k)\big]\big) \le \frac{\epsilon}{10}, \tag{5.15}$$

or after 20,000 inner loop iterations, whichever came first. Note that the overall termination condition (5.14) is in practice much stronger than the "relative error" termination conditions described in [6] (where the term "relative error" is used in a different sense than in this paper and [12]). Empirically, we found that using an "accuracy" of 10^{-4} with the relative error conditions of [6] could still result in errors as great as 5% in the objective function, so we used the more stringent and stable condition (5.14) instead.

Table 1 shows the number of multiplier adjustment steps for each combination of the six problem instance in the test set of [7] and each of the five algorithms tested; for the ADMM, the number of multiplier adjustments is simply the total number of iterations, whereas for the other algorithms it is the number of times the outer (over k) loop was executed until the convergence criterion (5.14) was met.

Generally speaking, the ADMM method requires between 4 and 10 times as many multiplier adjustments as the other algorithms. Note that even with highly approximate minimization of the augmented Lagrangian as implied by the choice of $\sigma = 0.99$ in the GS-RE and DQA-RE methods, their number of multiplier adjustments is far smaller than for the ADMM method. Comparing the four non-ADMM methods, there is very little variation in the number of multiplier adjustments, so there does not seem to be much penalty for minimizing the augmented Lagrangian approximately rather than almost exactly; however, there is a penalty in terms of outer iterations for the ADMM's strategy of updating the multipliers as often as possible, regardless of how little progress may have been made toward minimizing the augmented Lagrangian.

A different picture emerges when we examine total number of inner iterations for each algorithm, that is, the total number of x- or z-minimizations required; this information is shown in Table 2. First, it is clear that exactly minizing the augmented Lagrangian by either the Gauss-Seidel or DQA method is extremely inefficient, increasing the number of inner iterations by over two orders of magnitude as compared to the respective relative-error versions of the algorithms. This estimate is in fact conservative, because the limit of 20,000

Table 2: Cumulative number of inner iterations / x- or z-minimizations for the lasso test problems

Problem Instance	ADMM	GS-RE	DQA-RE	GS	DQA
Brain	814	2,452	8,721	1,186,402	1,547,424
Colon	1,889	3,911	15,072	590,183	1,214,258
Leukemia	1,321	3,182	8,105	735,858	1,286,989
Lymphoma	1,769	3,665	14,312	1,289,728	1,342,941
Prostate	838	2,421	6,691	475,004	1,253,180
SRBCT	2,244	4,629	17,333	1,183,933	1,266,221

inner iterations per outer iteration was often reached for both the GS and DQA methods. The augmented Lagrangian was thus not truly exactly minimized in a consistent manner in any of the algorithms, explaining the difference in the number of outer iterations between the GS and DQA methods in Table 1 (if the augmented Lagrangian were truly exactly minimized, one would expect the same number of outer iterations for these two methods). In summary, it is clear that the GS-RE method is far more efficient than DQA-RE, but ADMM requires significantly less computation than GS-RE.

We also performed some experiments with varying the parameter τ in the DQA-RE method. Employing a larger value of τ, such as 0.8, tended to increase the number of outer iterations, but modestly decrease the cumulative number of inner iterations. However, the difference was not sufficient to make DQA-RE competitive with GS-RE or the ADMM.

In these results, it is noteworthy that the total number of iterations that the ADMM requires to optimize a problem instance is significantly below the number of iterations that the Gauss-Seidel block optimization inner loop needs to exactly optimize a single augmented Lagrangian. In the results presented above, the average number of inner iterations per outer iteration of GS is on the order of 5,000, but this average includes many cases in which the inner loop was cut off at 20,000 iterations, so the true average number of inner iterations is actually higher. On the other hand, the ADMM requires only 2,244 iterations to optimize the most difficult problem instance. The form of block-coordinate minimization we have described is a very inefficient algorithm for the lasso problem, and while the first two steps of the ADMM bear an outward resemblance to this form of block-coordinate descent, the ADMM seems to be a fundamentally different and more efficient algorithm.

5.3 Computational Tests for the Transportation Problem

We also conducted computational tests on classical linear transportation problem, a problem class in which constraints play a much stronger role. Given a bipartite graph (S, D, E), we formulate this problem class as

$$\begin{array}{lll} \min & r^\top x & \\ \text{ST} & \displaystyle\sum_{j:(i,j)\in E} x_{ij} = s_i & \forall\, i \in S \\ & \displaystyle\sum_{i:(i,j)\in E} x_{ij} = d_j & \forall\, j \in D \\ & x_{ij} \geq 0 & \forall\, (i,j) \in E. \end{array} \tag{5.16}$$

Here, x_{ij} is the flow on edge $(i,j) \in E$, s_i is the supply at source $i \in S$, and d_j is the demand at destination node $j \in D$. Also, $x \in \mathbb{R}^{|E|}$ is the vector of all the flow variables x_{ij},

$(i,j) \in E$, and $r \in \mathbb{R}^{|E|}$, whose coefficients are similarly indexed as r_{ij}, is the vector of unit tranportation costs on the edges.

One way this problem may be formulated in the form (1.1) or (1.3) is as follows: let $m = n = |E|$ and define

$$f(x) = \begin{cases} \frac{1}{2} r^\top x, & \text{if } x \geq 0 \text{ and } \sum_{j:(i,j)\in E} x_{ij} = s_i \ \ \forall\, i \in S \\ +\infty & \text{otherwise} \end{cases} \tag{5.17}$$

$$g(z) = \begin{cases} \frac{1}{2} r^\top z, & \text{if } z \geq 0 \text{ and } \sum_{i:(i,j)\in E} z_{ij} = d_j \ \ \forall\, j \in D \\ +\infty & \text{otherwise.} \end{cases} \tag{5.18}$$

Again taking $M = I$, problem (1.3) is now equivalent to (5.16). Essentially, the $+\infty$ values in f enforce flow balance at all source nodes and the $+\infty$ values in g enforce flow balance at the destination nodes. Both f and g include the edge costs and the constraint that flows are nonnegative, and the constraints $Mx = z$, reducing to $x = z$, require that the flow into each edge at its source is equal to the flow out of the edge at its destination.

Applying the ADMM (4.23)-(4.25) to this choice of f, g, and M results in a highly parallelizable algorithm; details may be found in the online companion article [13]. In short, the x minimization breaks down into $|S|$ independent minimizations, one for each source node. Each of these subproblems is equivalent to projection onto a simplex. The z minimization similarly breaks down into $|D|$ independent simplex-projection subproblems, one for each destination node. Since $M = I$, the multiplier update breaks down into $|E|$ independent scalar calculations of the form $\lambda_{ij}^{k+1} = \lambda_{ij}^k + c\left(\rho_k x_{ij}^{k+1} + (1-\rho_k) z_{ij}^k - z_{ij}^{k+1}\right)$.

Our goal here not to investigate the competitiveness of this approach with classical algorithms for the transportation problem, but simply to explore the relative performance of the ADMM (4.23)-(4.25), the relative-error augmented Lagrangian methods GS-RE and DQA-RE, and their "exact" counterparts GS and DQA in a setting different from the lasso problems of the previous subsection.

For our tests, we generated dense transportation problem instances by locating the source and destination nodes uniformly randomly on the unit square, and using the resulting Euclidean distances as the edge costs r_{ij}; this procedure creates more challenging and realistic problems than using independent uniformly distributed edge costs, as in problem generators like NETGEN [18]. The supply amounts s_i were generated from a normal distribution with mean 50 and standard deviation 20, rounded to an integer, and the demand amounts d_j were generated similarly, followed by an adjustment to force $\sum_{i\in S} s_i = \sum_{j\in D} d_j$. We generated problems of size 20×20 (20 sources and 20 destinations), 20×30, 30×30, 30×40, 40×40, 40×50, and 50×50.

We took the same five fundamental algorithms tested in Section 5.2, specialized them to transportation problems using (5.17)-(5.18) and $M = I$, and implemented them in MATLAB. To implement the simplex projections required by the subproblems, we used a a simple sorting procedure based on MATLAB's intrinsic `sort` function. Because the problems we generated did not have nodes of very high degree and MATLAB intrinsic functions tend to be far faster than user-written code, this approach is likely to be more efficient in practice for our problem instances than attempting to implement the specialized, theoretically more efficient simplex projection algorithm of [22]. After a modicum of experimentation, reasonable choices of parameter values appeared to be $c = 0.005$ and $\rho = 1.0$; the settings $\sigma = 0.99$, $\tau = 0.5$ were the same as for the lasso problems. Again, the overall tolerance was $\epsilon = 10^{-6}$, but the inner iteration limit was set to $10,000$ instead of $20,000$ iterations; this

Table 3: Transportation problems: number of outer iterations / multiplier adjustments

Problem Instance	ADMM	GS-RE	DQA-RE	GS	DQA
20×20	1,633	37	40	123	363
20×30	3,016	141	175	160	490
30×30	3,375	31	41	28	585
30×40	1,234	207	401	194	496
40×40	3,747	183	717	231	862
40×50	5,923	341	1181	335	1,176
50×50	2,307	407	740	230	631

Table 4: Cumulative inner iterations / x- or z-minimizations for transportation problems

Problem Instance	ADMM	GS-RE	DQA-RE	GS	DQA
20×20	1,633	1,618	5,565	12,731	39,951
20×30	3,016	3,431	11,208	24,919	74,010
30×30	3,375	2,654	7,394	26,139	81,187
30×40	1,234	2,813	7,446	26,241	76,433
40×40	3,747	9,839	19,206	111,500	247,187
40×50	5,923	8,263	27,385	212,774	486,617
50×50	2,307	13,519	41,346	152,260	311,860

limit was only encountered by the two "exact" algorithms GS and DQA, and less frequently than in the lasso setting.

Table 3 shows the number of multiplier adjustments for each algorithm. Qualitatively, the results are fairly similar to those for lasso problem, although there is more variability between problem instances and between the the four non-ADMM methods. As before, the ADMM methods requires by far the most outer iterations, but this phenomenon is even more pronounced. For example, for the 20×20 problem, the ADMM requires over 40 times as many multiplier updates as GS-RE.

Table 4 shows the cumulative inner iteration counts. The results are quite similar to those for lasso problems, except that for the three smallest problems ADMM and GS-RE have nearly the same performance, with GS-RE being faster for one problem.

6 Concluding Remarks

A central point of this article is that, despite outward appearances and its original motivation, there is something more to the ADMM than just using a block-coordinate-descent method to approximately minimize augmented Lagrangians. Sections 3 and 4 show that while the theoretical convergence analysis of the two methods can be performed with similar tools, there are fundamental differences: the convergence of standard augmented Lagrangian method derives from a nonexpansive mapping derived from the entire dual function, whereas the ADMM analysis uses the composition of two nonexpansive maps respectively obtained from the two additive components q_1 and q_2 of the dual function.

These theoretical differences are underscored by the computational results in Section 5, which compares the ADMM to both approximate and "exact" versions of the classical

augmented Lagrangian method, using block-coordinate minimization methods to (approximately) optimize the augmented Lagrangian. Over two very different problem classes, the results are fairly consistent: the ADMM makes far more multiplier adjustments than the methods derived from the classical augmented Lagrangian method, but in most cases is more computationally efficient overall. In particular, the total number of iterations of the ADMM is considerably less than the number of block-coordinate minimization steps needed to exactly optimize even a single augmented Lagrangian. In summary, both theoretically and computationally, the original motivating viewpoint of the ADMM as a kind of hybrid of the augmented Lagrangian and Gauss-Seidel block minimization algorithms does not give full insight into the true character of the method.

It is apparent from the results for the GS and DQA algorithms that block-coordinate minimization is not an efficient algorithm for minimizing the nonsmooth augmented Lagrangian functions arising in either of the application classes discussed in Section 5, yet this phenomenon does not seem to affect the performance of the ADMM. In general, the ADMM seems a superior algorithm to approximate minimization of augmented Lagrangians by block-coordinate approaches; this conclusion was somewhat unexpected, in that the ADMM has a reputation for fairly slow convergence, especially in the "tail", whereas classical augmented Lagrangian method are generally considered competitive or near-competitive methods, and form the basis for a number of state-of-the-art nonlinear optimization codes. However, the results here do not necessarily establish the superiority of the ADMM to classical augmented Lagrangian methods using more sophisticated methods to optimize the subproblems.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant CCF-1115638. This material was presented in abridged form as a plenary presentation at the INFORMS Computing Society Conference in Santa Fe, NM in January 2013. It is also intended as a more theoretically oriented companion to [6].

References

[1] N.S. Aybat and G. Iyengar, An alternating direction method with increasing penalty for stable principal component pursuit, *Comput. Optim. Appl.* 61 (2015) 632–668.

[2] H.H. Bauschke and P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[3] D.P. Bertsekas, *Convex Analysis and Optimization*, Athena Scientific, Belmont, MA, 2003.

[4] D.P. Bertsekas, *Convex Optimization Algorithms*, Athena Scientific, 2015.

[5] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, 1989.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends in Machine Learning.* 3 (2011) 1–122.

[7] M. Dettling and P. Bühlmann, Finding predictive gene groups from microarray data, *J. Multivariate Anal.* 90 (2004) 106–131.

[8] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*, PhD thesis, MIT, 1989.

[9] J. Eckstein, A practical general approximation criterion for methods of multipliers based on Bregman distances, *Math. Program.* 96 (2003) 61–86.

[10] 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) 293–318.

[11] J. Eckstein and M.C. Ferris, Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control, *INFORMS J. Comput.* 10 (1998) 218–235.

[12] J. Eckstein and P.J.S. Silva, A practical relative error criterion for augmented Lagrangians, *Math. Program.* 141 (2013) 319–348.

[13] J. Eckstein and W. Yao, Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives, Preprint 2015-05-4935, Optimization Online, 2015.

[14] M. Fortin and R. Glowinski, On decomposition-coordination methods using an augmented Lagrangian, in *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*, M. Fortin and R. Glowinski (eds.), North-Holland, Amsterdam, 1983, pp. 97–146.

[15] D. Gabay, Applications of the method of multipliers to variational inequalities, in *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*, M. Fortin and R. Glowinski (eds.), North-Holland, Amsterdam, 1983, pp. 299–331.

[16] R. Glowinski and A. Marroco, Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualité, d'une classe de problems de Dirichlet non lineares, *Rev. Française Informat. Recherche Opérationnelle.* 9 (1975) 41–76.

[17] B.-S. He, H. Yang and S.-L. Wang, Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities, *J. Optim. Theory Appl.* 106 (2000) 337–356.

[18] D. Klingman, A. Napier and J. Stutz, NETGEN: A program for generating large-scale capacitated assignment, transportation, and minimum cost flow network problems, *Manage. Sci.* 20 (1974) 814–821.

[19] M.A. Krasnosel'skiĭ, Two remarks on the method of successive approximations, *Uspekhi Mat. Nauk.* 10 (1955) 123–127.

[20] J. Lawrence and J.E. Spingarn, On fixed points of non-expansive piecewise isometric mappings, *Proc. Lond. Math. Soc.* 3 (1987) 605.

[21] P.L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, *SIAM J. Numer. Anal.* 16 (1979) 964–979.

[22] N. Maculan and G.G. de Paula, Jr., A linear-time median-finding algorithm for projecting a vector on the simplex of $\mathbb{R}^n$, *Oper. Res. Lett.* 8 (1989) 219–222.

[23] G.J. Minty, Monotone (nonlinear) operators in Hilbert space, *Duke Math. J.* 29 (1962) 341–346.

[24] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970.

[25] R.T. Rockafellar, *Conjugate Duality and Optimization*, SIAM, Philadelphia, 1974.

[26] R.T. Rockafellar, Augmented Lagrangians and applications of the proximal point algorithm in convex programming, *Math. Oper. Res.* 1 (1976) 97–116.

[27] R.T. Rockafellar, Monotone operators and the proximal point algorithm, *SIAM J. Control Optim.* 14 (1976) 877–898.

[28] R.T. Rockafellar and R.J.-B. Wets, Scenarios and policy aggregation in optimization under uncertainty, *Math. Oper. Res.* 16 (1991) 119–147.

[29] A. Ruszczyński, On convergence of an augmented Lagrangian decomposition method for sparse convex optimization, *Math. Oper. Res.* 20 (1995) 634–656.

[30] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, *J. Optim. Theory Appl.* 109 (2001) 475–494.

Manuscript received 24 April 2014
revised 27 November 2014
accepted for publication 27 November 2014

Jonathan Eckstein
Department of Management Science and
Information Systems and RUTCOR, Rutgers University
100 Rockafeller Road, Piscataway, NJ
E-mail address: jeckstei@rci.rutgers.edu

Wang Yao
RUTCOR, Rutgers University
100 Rockafeller Road, Piscataway, NJ
E-mail address: wang.yao09@rutgers.edu