RUTCOR
RESEARCH
REPORT

Augmented Lagrangian and Alternating Direction Methods for Convex Optimization: A Tutorial and Some Illustrative Computational Results

Jonathan Eckstein[a]

RRR 32-2012, December 2012

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]Department of Management Science and Information Systems (MSIS) and RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003

Rutcor Research Report
RRR 32-2012, December 2012

Augmented Lagrangian and Alternating Direction Methods for Convex Optimization: A Tutorial and Some Illustrative Computational Results

Jonathan Eckstein

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, and the relative ease with which it may be implemented in parallel and distributed computational environments. This chapter aims to provide an accessible introduction to the analytical underpinnings of the method, which are often obscured in treatments that do not assume knowledge of convex and set-valued analysis. In particular, it is tempting to view 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. This chapter, assuming as little prior knowledge of convex analysis as possible, shows that the actual convergence mechanism of the algorithm is quite different, and then underscores this observations with some new computational results in which we compare the ADMM to algorithms that do indeed work by approximately minimizing the augmented Lagrangian.

Acknowledgements: This material is based in part upon work supported by the National Science Foundation under Grant CCF-1115638. The author also thanks his student Wang Yao for his work on the computational experiments.Please insert the acknowledgement here.

1 Introduction

The *alternating direction method of multipliers* (ADMM) is a convex optimization algorithm dating back to the early 1980's [10, 11]; 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 adapted 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".
- Depending on the application, it is often is 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 [3] describes the ADMM from the perspective of machine learning applications; another, older survey is contained in the doctoral thesis [5].

The current literature of the ADMM presents its convergence theory in two different ways: some references [17, 7, 8] make extensive use of abstract concepts from convex analysis and set-valued operators, and are thus somewhat inaccessible to many readers, while others [2, 3] present proofs which use only elementary principles but do not attempt to convey the "deep structure" of the algorithm.

This paper has two goals: the first is to attempt to convey the "deep structure" of the ADMM without assuming the reader has extensive familiarity with convex or set-valued 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 wider audience. The second goal is to present some recent computational results which are interesting in their own right and serve to illustrate the practical impact of the convergence theory of the ADMM. In particular, these results suggest that it is not, despite superficial appearances, particularly insightful to think of the ADMM as an approximation to the classical augmented Lagrangian algorithm combined with a block coordinate minimization method for the subproblems.

The remainder of this paper is structured as follows: Section 2 describes the ADMM and a popular example application, the "lasso" or compressed sensing problem, and then Section 3 summarizes some necessary background material. Section 4 presents the classic augmented Lagrangian method for convex problems as an application of nonexpansive algorithmic mappings, and then Section 5 applies the same analytical techniqeus to the ADMM. Section 6 presents the computational results and Section 7 makes some concluding remarks

and presents some avenues for future research. The material in Sections 2-5 is not fundamentally new, and can be inferred from the references mentioned in each section; the only 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.

2 Statement of the ADMM and a Sample Problem

Although it can be developed in a slightly more general form — see for example [3] — the following problem formulation is sufficient for most applications of the ADMM:

$$\min_{x\in\mathbb{R}^n} f(x) + g(Mx). \tag{1}$$

Here, M is an $m \times n$ matrix, often 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 f and g, in the sense that if $f(x) = \infty$ or $g(Mx) = \infty$, then the point x is considered to be infeasible for (1).

By appropriate use of infinite values for f or g, a very wide range of convex problems may be modeled through (1). To make the discussion more concrete, however, we now describe a simple illustrative example that fits very readily into the form (1) without any use of infinite function values, and resembles in basic structure many of applications responsible for the resurgence of interest in the ADMM: the "lasso" or "compressed sensing" problem. This problem takes the form

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

where A is a $p \times n$ matrix, $b \in \mathbb{R}^p$, and $\nu > 0$ is a given scalar parameter. The idea of the 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 clearly has limitations in terms of finding sparse near-solutions to $Ax = b$, it serves as a good example application of the ADMM; 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). First, note that we can rewrite (1), introducing an additional decision variable $z \in \mathbb{R}^m$, as

$$\begin{array}{ll} \min & f(x) + g(z) \\ \text{ST} & Mx = z. \end{array} \tag{3}$$

The classical augmented Lagrangian algorithm, which we will discuss in more depth in Section 4, is described in the case of formulation (3) by the recursions

$$(x^{k+1}, z^{k+1}) \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \left\{ f(x) + g(z) + \langle \lambda^k, Mx - z\rangle + \tfrac{c_k}{2}\left\|Mx-z\right\|^2 \right\} \tag{4}$$

$$\lambda^{k+1} = \lambda^k + c_k(Mx^{k+1} - z^{k+1}). \tag{5}$$

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 paper, $\langle a, b\rangle$ denotes the usual Euclidean inner product $a^\top b$.

In this setting, the standard augmented Lagrangian algorithm (4)-(5) is not very attractive because the minimizations of f and g in the subproblem (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).

The alternating direction method of multipliers (ADMM) for (1) or (3) takes the form

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + g(z^k) + \langle \lambda^k, Mx - z^k\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{6}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{z\in\mathbb{R}^m} \left\{ f(x^{k+1}) + g(z) + \langle \lambda^k, Mx^{k+1} - z\rangle + \tfrac{c}{2}\left\|Mx^{k+1} - z\right\|^2 \right\} \tag{7}$$

$$\lambda^{k+1} = \lambda^k + c(Mx^{k+1} - z^{k+1}). \tag{8}$$

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 (6) and (7). Unlike the classical augmented Lagrangian method, the ADMM essentially decouples the functions f and g, since (6) requires only minimization of a quadratic perturbation of f, and (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 (6) and (7) may be computed in an efficient and perhaps highly parallel manner.

Given the form of the two algorithms, is natural to view the ADMM (6)-(8) as an approximate version of the classical augmented Lagrangian method (4)-(5) in which a single pass of "Gauss-Seidel" block minimization substitutes for full minimization of the augmented Lagrangian

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

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 (4); this viewpoint dates back to one of the earliest presentation of the method in [10]. However, this interpretation bears no resemblance to any known convergence theory for the ADMM, and there is no known general way of quantifying how close the sequence of calculations (6)-(7) approaches the joint minimization (4). The next two sections of this paper attempt to give the reader insight into the known convergence theory of the ADMM, without requiring extensive background in convex or set-valued analysis. But first, it is necessary to gain some insight into the convergence mechanism behind classical augmented Lagrangian method such as (4)-(5).

3 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 or omitted. Most of the material here can be readily found (often in more general form) in [21] or in textbooks such as [1]. The main goal insight of this section is that there is a strong relationship between convex functions and nonexpansive mappings, that is, maps $N : \mathbb{R}^n \to \mathbb{R}^n$ 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 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*

$$f(x') \geq f(x) + \langle v, x' - x' \rangle \qquad \forall\, x' \in \mathbb{R}^n. \tag{9}$$

The notation $\partial f(x)$ denotes the set of all subgradients of f at x.

Basically, v is a subgradient of f at x if the affine function $a(x') = f(x) + \langle v, x' - x \rangle$ 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 $\{\nabla f(x)\}$.

Lemma 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*

$$\langle x - x', v - v' \rangle \geq 0. \tag{10}$$

Proof. From the subgradient inequality (9), 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 (10). □

Lemma 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. This result is a simple consequence of Lemma 2. 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 asserts that $\langle x - x', v - v' \rangle \geq 0$, we must have $v = v'$ and hence $x = x'$. □

Figure 1 shows a simplified depiction of Lemmas 2 and 3 in $\mathbb{R}^1$. The solid line shows the set of possible pairs $(x, v) \in \mathbb{R}^n \times \mathbb{R}^n$ where $v \in \partial f(x)$; by Lemma 2, connecting any two such pairs must result in line segment that is either vertical or has nonnegative slope. The dashed line represents the set of points $(x, v) \in \mathbb{R}^n \times \mathbb{R}^n$ satisfying $x + cv = z$; since this line has negative slope, it can intersect the solid line at most once, illustrating Lemma 3. Next, we give conditions under which a decomposition of any $z \in \mathbb{R}^n$ into $x + cv$, where $v \in \partial f(x)$, must exist, in addition to having to be unique. First, we state a necessary background result:

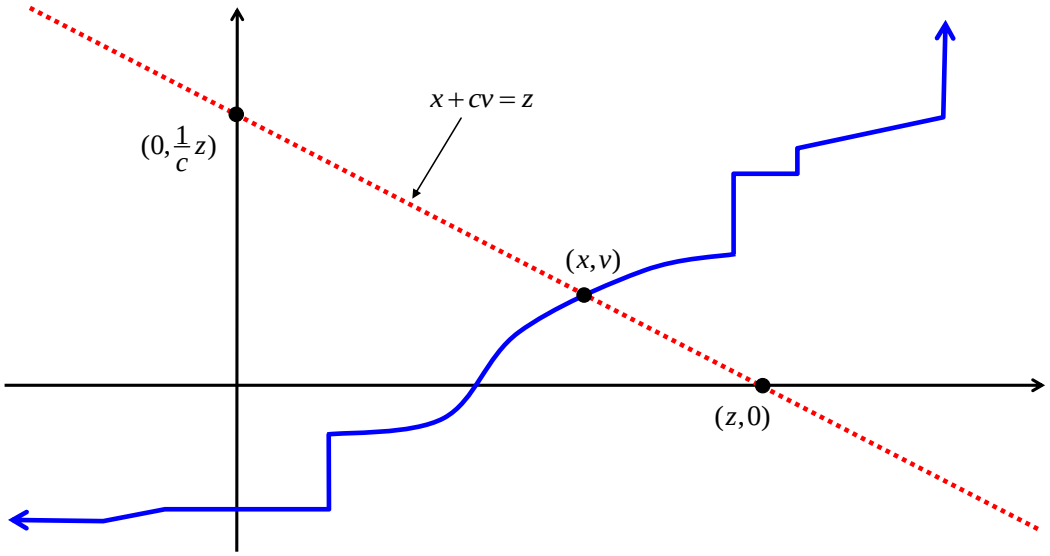

Figure 1: Graphical depiction of Lemmas 2 and 3 for dimension $n = 1$.

Lemma 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, at any $x \in \mathbb{R}^n$,*

$$\partial(f + g)(x) = \partial f(x) + \nabla g(x) = \{y + \nabla g(x) \mid y \in \partial f(x)\}.$$

The proof of this result is somewhat more involved than one might assume, so we omit it; see for example [21, Theorem 23.8 and 25.1] or [1, Propositions 4.2.2 and 4.2.4]. However, the result itself should be completely 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 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 very 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 [21, Theorem 7.1] or [1, Proposition 1.2.2].

Proposition 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 $v \in \partial f(x)$.*

Sketch of proof. Let $c > 0$ and $z \in \mathbb{R}^n$ be given. Consider now the function $\tilde{f} : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ given by $\tilde{f}(x) = f(x) + \frac{1}{2c}\|x - z\|^2$. Using the equivalence of closedness to lower semicontinuity, it is easily seen that f being closed and proper implies that $\tilde{f}$ is closed and proper. The convex function f can decrease with at most a linear rate in any direction, so the

presence of the quadratic term $\frac{1}{2c}\|x-z\|^2$ guarantees that $\tilde{f}$ is *coercive*, that is $\tilde{f}(x) \to +\infty$ as $\|x\| \to \infty$. Since $\tilde{f}$ is proper, closed, and coercive, one can then invoke a version of the classical Weirstrass theorem — see for example [1, Proposition 2.1.1] — to assert that $\tilde{f}$ must attain its minimum value at some point x, which implies that $0 \in \partial\tilde{f}(x)$. Since f and $x \mapsto \frac{1}{2c}\|x-z\|^2$ are both convex, and the latter is differentiable, Lemma 4 asserts that there must exist $v \in \partial f(x)$ such that $0 = v + \nabla_x\left(\frac{1}{2c}\|x-z\|^2\right) = v + \frac{1}{c}(x-z)$, which is easily rearranged into $z = x + cv$. Lemma 3 asserts that this representation must be unique. □

This result is a special case of Minty's theorem [20]. 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 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*

$$N_{cf}(z) = x - cv \qquad \text{where } x, v \in \mathbb{R}^n \text{ are such that } v \in \partial f(x) \text{ and } x + cv = z \tag{11}$$

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

Proof. From Proposition 6, there is exactly one way to express any $z \in \mathbb{R}^n$ as $x + cv$, so therefore the definition given for $N_{cf}(z)$ exists and is unique for all z. 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

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

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. □

4 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

essentially orginated with [23, 24]. Consider an optimization problem

$$\begin{array}{ll} \min & h(x) \\ \text{ST} & Ax = b, \end{array} \tag{12}$$

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 model the problem (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 (12) is

$$q(\lambda) = \min_{x \in \mathbb{R}^n} \{h(x) + \langle \lambda, Ax - b \rangle\}. \tag{13}$$

The dual problem to (12) 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 8 *The function d is closed and convex. If the vector $x^* \in \mathbb{R}^n$ attains the minimum in* (13) *for some given $\lambda \in \mathbb{R}^m$, then $b - Ax^* \in \partial d(\lambda)$.*

Proof. We have that $d(\lambda) = \max_{x \in \mathbb{R}^n} \{-f(x) - \langle \lambda, Ax - b \rangle\}$, 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 simple (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 (13). 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 $b - Ax^* \in \partial d(\lambda)$. □

[1] Rockafellar's parametric conjugate duality framework [22] 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.

Since d is closed, then if it is also proper, that is, it is finite for at least one choice of $\lambda \in \mathbb{R}^m$, Proposition 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 d is that this decomposition may be computed in a manner not much more burdensome than evaluating d itself:

Proposition 9 *Given any $\mu \in \mathbb{R}^m$, consider the problem*

$$\min_{x \in \mathbb{R}^n} \{h(x) + \langle \mu, Ax - b \rangle + \tfrac{c}{2} \|Ax - b\|^2\}. \tag{14}$$

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 4, $\bar{x}$ is optimal for (14) if there exists $w \in \partial h(\bar{x})$ such that

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

where $\lambda = \mu + c(A\bar{x} - b)$ as above. Using Lemma 4 once again, this condition means that $\bar{x}$ attains the minimum in (13), and hence Lemma 8 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 6 may be much more difficult than simply evaluating the function. The main message of Proposition 9 is that for functions d obtained from the duals of convex programming problems like (12), 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 the one that needs to be solved to evaluate d itself. To avoid getting bogged down in convex-analytic details that would distract from our main tutorial goals, we will not directly address here exactly when it is guaranteed that an optimal solution to (14) 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 (14), the same calculation allows us to evaluate the nonexpansive mapping N_{cd}, using the notation of Section 3: specifically, using the notation of Proposition 9, 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 (12). 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 10 *Let $T : \mathbb{R}^m \to \mathbb{R}^m$ be nonexpansive, that is, $\|T(y) - T(y')\| \leq \|y - y'\|$ 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*

$$y^{k+1} = \tfrac{\rho_k}{2} T(y^k) + (1 - \tfrac{\rho_k}{2}) y^k, \tag{15}$$

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

The case $\rho_k = 1$ 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 10 was proven, in a much more general setting, as long ago as 1955 [16], and the case of variable ρ_k is similar and implicit in many more recent results, such as those of [7].

Combining Proposition 9 and Theorem 10 yields a convergence proof for a form of augmented Lagrangian algorithm:

Proposition 11 *Consider a problem of the form (12) 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*

$$x^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^m} \left\{ h(x) + \langle \lambda^k, Ax - b \rangle + \tfrac{c}{2} \|Ax - b\|^2 \right\} \tag{16}$$

$$\lambda^{k+1} = \lambda^k + \rho_k c(Ax^{k+1} - b). \tag{17}$$

If the dual problem to (12) possesses an optimal solution, then $\{\lambda^k\}$ converges to one of them, and all limit points of $\{x^k\}$ are optimal solutions to (12).

Proof. The optimization problem solved in (16) is just the one in Proposition 9 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

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

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 (12) is a minimizer of d, and hence a fixed point of N_{cd} by Proposition 7. Theorem 10 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 (17) that $Ax^k - b \to 0$. If x^* is any optimal solution to (12), we have from the optimality of x^{k+1} in (16) that

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

that is, $h(x^k) + \langle \lambda^{k-1}, Ax^k - b\rangle + \frac{c}{2}\|Ax^k - b\| \leq h(x^*)$ 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

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

and so x^∞ is also optimal for (12). □

There are two differences between (16)-(17) and the augmented Lagrangian method as it is typical presented for (12). 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 10 because the operators $N_{c_k d}$ being used at each iteration may be different. However, essentially the same convergence proof needed for Theorem 10 may still be employed, because the operators $N_{c_k d}$ all have exactly the same fixed points, 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* [23, 24]; see also [7].

5 Analyzing the ADMM through Compositions of Nonexpansive Mappings

We now describe the convergence theory of the ADMM (6)-(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 [18], [10], [17], [5], and [8, Section 1]; a similar treatment may be found in [7].

To begin, we consider the dual problem of the problem (3), namely to maximize the function

$$q(\lambda) = \min_{\substack{x\in\mathbb{R}^n \\ z\in\mathbb{R}^p}} \{f(x) + g(z) + \langle \lambda, Mx - z\rangle\} \tag{18}$$

$$= \min_{x\in\mathbb{R}^n} \{f(x) + \langle \lambda, Mx\rangle\} + \min_{z\in\mathbb{R}^p} \{g(z) - \langle \lambda, z\rangle\} \tag{19}$$

$$= q_1(\lambda) + q_2(\lambda), \tag{20}$$

where we define

$$q_1(\lambda) = \min_{x \in \mathbb{R}^n} \{f(x) + \langle \lambda, Mx \rangle\} \qquad q_2(\lambda) = \min_{z \in \mathbb{R}^p} \{g(z) - \langle \lambda, z \rangle\}. \tag{21}$$

Defining $d_1(\lambda) = -q_1(\lambda)$ and $d_2(\lambda) = -q_2(\lambda)$, the dual of (3) is thus equivalent to minimizing the sum of the two convex functions d_1 and d_2:

$$\min_{\lambda \in \mathbb{R}^m} d_1(\lambda) + d_2(\lambda) \tag{22}$$

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 12 *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 (21). Then $-Mx^* \in \partial d_1(\lambda)$. Similarly, if $z^* \in \mathbb{R}^p$ attains the second minimum in (21), then $z^* \in \partial d_2(\lambda)$.*

Proof. We simply apply Lemma 8 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 (22), that is, minimizing the sum of two convex functions.

Lemma 13 *A sufficient condition for $\lambda^* \in \mathbb{R}^m$ to solve (22) is*

$$\exists\, v^* \in \partial d_1(\lambda^*) \;\; : \;\; -v^* \in \partial d_2(\lambda^*). \tag{23}$$

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 (22). □

Under some mild regularity conditions that are met in most practical applications, (23) 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 (22).

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 3. These maps are given by

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

are both nonexpansive by exactly the same analysis given in Section 4. It follows that their functional composition $N_{cd_1} \circ N_{cd_2}$ is also nonexpansive, that is,

$$\big\| N_{cd_1}\big(N_{cd_2}(\lambda)\big) - N_{cd_1}\big(N_{cd_2}(\lambda')\big) \big\| \leq \|N_{cd_2}(\lambda) - N_{cd_2}(\lambda')\| \leq \|\lambda - \lambda'\|$$

for all $\lambda, \lambda' \in \mathbb{R}^m$.

Lemma 14 $\mathrm{fix}(N_{cd_1} \circ N_{cd_2}) = \{\lambda + cv \mid v \in \partial d_2(\lambda), -v \in \partial d_1(\lambda)\}$.

Proof. Take any $y \in \mathbb{R}^m$ and write it as $y = \lambda + cv$, where $v \in \partial d_2(\lambda)$. Then $N_{cd_2}(y) = \lambda - cv$. Now, $N_{cd_2}(y) = \lambda - cv$ can be written in at 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

$$\mu - cw = y = \lambda + cv. \tag{24}$$

Adding $\mu + cw = N_{cd_2}(y) = \lambda - cv$ to (24), we obtain $\mu = \lambda$. Substituting $\mu = \lambda$ into (24), we also obtain $w = -v$. □

Thus, finding a fixed point of $N_{cd_1} \circ N_{cd_2}$, is essentially the same as finding $\lambda, v \in \mathbb{R}^m$ satisfying $v \in \partial d_2(\lambda)$, $-v \in \partial d_1(\lambda)$, and thus an optimal solution to the dual problem (22).

At this point, an obvious strategy is to try to apply Theorem 10 to finding a fixed point of $N_{cd_1} \circ N_{cd_2}$, leading essentially immediately to a solution of (22), that is, we perform the iteration

$$y^{k+1} = \tfrac{\rho_k}{2} N_{cd_1}\big(N_{cd_2}(y^k)\big) + (1 - \tfrac{\rho_k}{2}) y^k \tag{25}$$

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

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

Finally, the last expression for y^{k+1} above may be simplified as follows: since $\lambda^k - cv^k = \mu^k + cw^k$, 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 implemented as follows, starting from some arbitrary $\lambda^0, v^0 \in \mathbb{R}^m$:

1. Find μ^k, w^k such that $w^k \in \partial d_1(\mu^k)$ and

$$\mu^k + cw^k = \lambda^k - cv^k. \tag{26}$$

2. Find λ^{k+1}, v^{k+1} such that $v^{k+1} \in \partial d_2(\lambda^{k+1})$ and

$$\lambda^{k+1} + cv^{k+1} = \rho_k \mu^k + (1-\rho_k)\lambda^k + cv^k. \tag{27}$$

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 (22). As an aside, this pattern of computations is an example of (generalized) *Douglas-Rachford splitting* [18, 5, 7], whose relationship to the ADMM was first shown in [11]. It is the same basic "engine" behind the convergence of several other popular algorithms, including the progressive hedging method for stochastic programming [25].

We now consider how to implement the scheme above for the particular functions $d_1 = -q_1$ and $d_2 = -q_2$, where q_1 and q_2 are defined by (21). Conveniently, we can simply apply Proposition 9: in the case of (26), applying Proposition 9 with $h = f$, $A = M$, $b = 0$, and $\mu = \lambda^k - cv^k$ yields the computation

$$x^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ f(x) + \langle \lambda^k - cv^k, Mx \rangle + \tfrac{c}{2} \|Mx\|^2 \right\} \tag{28}$$

$$\mu^k = \lambda^k - cv^k + cMx^{k+1} \tag{29}$$

$$w^k = -Mx^{k+1}. \tag{30}$$

Now let us consider (27): again applying Proposition 9, 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

$$z^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ g(z) + \langle \rho_k \mu^k + (1-\rho_k)\lambda^k + cv^k, -z \rangle + \tfrac{c}{2} \|-z\|^2 \right\} \tag{31}$$

$$\lambda^{k+1} = \rho_k \mu^k + (1-\rho_k)\lambda^k + cv^k - cz^{k+1} \tag{32}$$

$$v^{k+1} = z^{k+1}. \tag{33}$$

We now simplify this system of recursions. First, we note that the sequence $\{w^k\}$ does not appear in any recursion except (30), so we may simply eliminate it. Second, using (33), we may substitute $v^k = z^k$ throughout, yielding

$$x^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ f(x) + \langle \lambda^k - cz^k, Mx \rangle + \tfrac{c}{2} \|Mx\|^2 \right\} \tag{34}$$

$$\mu^k = \lambda^k - cz^k + cMx^{k+1} \tag{35}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{x \in \mathbb{R}^n} \left\{ g(z) - \langle \rho_k \mu^k + (1-\rho_k)\lambda^k + cz^k, z \rangle + \tfrac{c}{2} \|z\|^2 \right\} \tag{36}$$

$$\lambda^{k+1} = \rho_k \mu^k + (1-\rho_k)\lambda^k + cz^k - cz^{k+1}. \tag{37}$$

Next, consider (34). Adding the constant $\frac{c}{2}\|z^k\|^2$ to the minimand and completing the square yields the equivalent computation

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k, Mx\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{38}$$

From (35), we have $\mu^k = \lambda^k + c(Mx^{k+1} - z^k)$, hence

$$\begin{aligned} \rho_k\mu^k + (1-\rho_k)\lambda^k + cz^k &= \rho_k\big(\lambda^k + c(Mx^{k+1} - z^k)\big) + (1-\rho_k)\lambda^k + cz^k \\ &= \lambda^k + \rho_k cMx^{k+1} + (1-\rho_k)cz^k \\ &= \lambda^k + c\big(\rho_k Mx^{k+1} + (1-\rho_k)z^k\big). \end{aligned} \tag{39}$$

Thus, the minimand in (36) may be written

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

Similarly to the derivation of (38), 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 (39) into (37) 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 collapse to

$$x^{k+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ f(x) + \langle \lambda^k, Mx\rangle + \tfrac{c}{2}\left\|Mx - z^k\right\|^2 \right\} \tag{40}$$

$$z^{k+1} \in \operatorname*{Arg\,min}_{z\in\mathbb{R}^p} \left\{ g(z) - \langle \lambda^k, z\rangle + \tfrac{c}{2}\left\|\rho_k Mx^{k+1} + (1-\rho_k)z^k - z\right\|^2 \right\} \tag{41}$$

$$\lambda^{k+1} = \lambda^k + c\big(\rho_k Mx^{k+1} + (1-\rho_k)z^k - z^{k+1}\big). \tag{42}$$

In the special case $\rho_k \equiv 1$, this sequence of recursions reduces exactly to the ADMM (6)-(8), except for some immaterial constant terms in the minimands.

This derivation contains the essence of the convergence theory of the ADMM method: it is an application of the fixed-point algorithm of Theorem 10 to the nonexpansive mapping $N_{cd_1} \circ N_{cd_2}$. We summarize the flow of analysis above in the proposition that follows; to avoid getting sidetracked in convex-analytic details, we make an additional assumption about subgradients of the function d_1; this assumption is guaranteed by a variety of simple regularity conditions that are met in most practical applications.

Proposition 15 *Consider the problem* (1)*, and let the constant* $c > 0$ *be given. Suppose there exists some optimal primal-dual solution pair* $\big((x^*, z^*), \lambda^*\big)$ *to* (1) *with the following properties:*

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. *$Mx^* = z^*$.*

Assume also that 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. 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* (40)-(42)*, where $\inf_k\{\rho_k\} > 0$ and $\sup_k\{\rho_k\} < 2$, then $\lambda^k \to \lambda^\infty$, $z^k \to z^\infty$, and $Mx^k \to Mx^\infty = z^\infty$, where $\big((x^\infty, z^\infty), \lambda^\infty\big)$ is some triple conforming to conditions 1-3 above (substituting "∞" for "$*$").*

Proof. The development above shows that the recursions (40)-(42) are equivalent to the sequence $y^k = \lambda^k + cv^k = \lambda^k + cz^k$ being produced by the recursion (25), which by Theorem 10 converges to a fixed point of the operator $N_{cd_1} \circ N_{cd_2}$, if one exists. If the triple $\big((x^*, z^*), \lambda^*\big)$ satisfies conditions 1-3, then $\lambda^* + cz^*$ is just such a fixed point, so such a point does exist. Therefore y^k is convergent to some fixed point y^∞ of $N_{cd_1} \circ N_{cd_2}$, which by Lemma 14 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 $R_{cd_2} = \frac{1}{2}N_{cd_2} + \frac{1}{2}I$. Since N_{cd_2} is nonexpansive, R_{cd_2} is certainly continuous. We have $R_{cd_2}(y^\infty) = \frac{1}{2}(\lambda^\infty - cz^\infty) + \frac{1}{2}(\lambda^\infty + cz^\infty) = \lambda^\infty$ and similarly $R_{cd_2}(y^k) = \lambda^k$ since $z^k = v^k \in \partial d_2(\lambda^k)$, and so by the continuity of R_{cd_2} we must have $\lambda^k = R_{cd_2}(y^k) \to R_{cd_2}(y^\infty) = \lambda^\infty$ 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 (42) as

$$\lambda^{k+1} - \lambda^k = c\rho_k(Mx^{k+1} - z^k) + c(z^k - z^{k+1}). \tag{43}$$

The quantity on the left of (43) 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 several observations worth making at this point:

1. In most results regarding the convergence of the ADMM, the assumption above about the form of subgradients of d_1 is typically replaced by more natural, verifiable assumptions regarding f and M; these assumptions imply the assumption on d_1 holds, and are met in almost all practical applications. The version above was presented to minimize the amount of convex-analytical background required.

2. The resemblance of the ADMM to an approximate version of the method of multipliers (4)-(5) is in some sense coincidental; the convergence theory is not based on approximating the nonexpansive mappings $N_{c_k d}$ underlying the classical method of multipliers as discussed in Section 4, but on the exact evaluation of the fundamentally different nonexpansive mapping $N_{cd_1} \circ N_{cd_2}$. Note that approximate versions of the ADMM are possible, as described for example in [7].

3. The underlying convergence-mechanism differences between the ADMM and the classical augmented Lagrangian method (4)-(5) are underscored by the different forms that the parameters ρ_k of the fixed point iteration (15) appear. Applying the augmented Lagrangian method (16)-(17) to the problem formulation (3) would result in a version of the method (4)-(5) in which (5) is amended to

$$\lambda^{k+1} = \lambda_k + \rho_k c_k (Mx^{k+1} - z^{k+1}). \tag{44}$$

   On the other hand, ρ_k is used in a different way in (40)-(42): the overrelaxation appears in the "target" value $\rho_k Mx^{k+1} + (1 - \rho_k)z^k$ for z in (41) and similarly in the following multiplier update (42).[2]

4. Varying the parameter c from one iteration in the ADMM to another is more problematic for the ADMM than in 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 analysis of the convergence of the ADMM with nonconstant c much more difficult, although some partial results have been obtained; see for example [14].

5. 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. This situation explains the relative difficulty of proving the convergence of the ADMM when extended in a direct manner to more than two blocks of variables. This topic is an area of recent active research — see for example [12, 13] — but so far the results obtained do not have the simplicity of the ADMM, and they often require "corrector" steps that prevent them from being direct generalizations of the ADMM.

6 Computational Comparison of ADMM and Approximate Augmented Lagrangian Methods

6.1 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 using a single pass of a block-minimization method to approximately minimize the augmented Lagrangian. We now explore the same topic from a computational perspective

[2]Note, however, that [10] proves convergence of a variant of the ADMM consisting of (6), (7), and (44), where ρ_k is constant and in the range $\big(0, (1+\sqrt{5})/2\big)$. The scheme (40)-(42) has a conceptually simpler convergence proof, allows larger ρ_k, and appears to work at least as well in practice.

by comparing the ADMM to methods that really do perform 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 in (4) or (16) — that can be hard to test in practice. Notable exceptions are the methods proposed in [6, 9]. Here, we will focus on the "relative error" method of [9], which has been shown to have better practical performance; we will not review or explain the analysis of these methods there.

In the context of problem (12), each iteration of the algorithmic template of [9] 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

$$y^k \in \partial_x \big[f(x) + \langle \lambda^k, Ax - b \rangle + \tfrac{c_k}{2} \|Ax - b\|^2 \big]_{x=x^{k+1}} \tag{45}$$

$$\tfrac{2}{c_k} \left| \langle w^k - x^{k+1}, y^k \rangle \right| + \|y^k\|^2 \leq \sigma \left\| Ax^{k+1} - b \right\|^2 \tag{46}$$

2. Perform the updates

$$\lambda^{k+1} = \lambda^k + c_k(Ax^{k+1} - b) \tag{47}$$

$$w^{k+1} = w^k - c_k y^k. \tag{48}$$

In (45), ∂_x denotes the set of subgradients with respect to x, with λ treated as a fixed parameter. The conditions (45)-(46) constitute an approximate minimization with respect to x, in the sense that if x^{k+1} exactly minimizes the augmented Lagrangian as in (16), then we can take $y^k = 0$ and (46) is immediately satisfied because its left-hand side is zero. However, (45)-(46) 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 not needed in the exact form of the algorithm.

Adapting this pattern to the specific problem (1), we define the notation

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

We also 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 recursive conditions

$$(y_x^k, y_z^k) \in \partial_{(x,z)} L_c(x^{k+1}, z^{k+1}, \lambda^k) \tag{49}$$

$$\tfrac{2}{c_k} \left| \langle w_x^k - x^{k+1}, y_x^k \rangle + \langle w_z^k - z^{k+1}, y_z^k \rangle \right| + \|y_x^k\|^2 + \|y_z^k\|^2 \leq \sigma \left\| Mx^{k+1} - z^{k+1} \right\|^2 \tag{50}$$

$$\lambda^{k+1} = \lambda^k + c_k(Mx^{k+1} - z^{k+1}) \tag{51}$$

$$w_x^{k+1} = w_x^k - c_k y_x^k \tag{52}$$

$$w_z^{k+1} = w_z^k - c_k y_z^k. \tag{53}$$

We now consider how to perform the approximate minimization embodied in (49)-(50), keeping in mind that f or g (or both) might be nonsmooth; for example, in problem (2), the function f is smooth, but g is nonsmooth.

Inspired by the ADMM, 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),

$$x^{k,i+1} \in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ L_{c_k}(x, z^{k,i}, p^k) \right\} \tag{54}$$

$$z^{k,i+1} \in \operatorname*{Arg\,min}_{z\in\mathbb{R}^m} \left\{ L_{c_k}(x^{k,i+1}, z, p^k) \right\} \tag{55}$$

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 [27] establishes subsequential convergence of such a scheme in most cases of interest, including the example problem (2). In integrating (54)-(55) into the overall scheme (49)-(53), we note that (55) 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 computational scheme (56) shown in Figure 2. The last two steps of this method are not strictly 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 literature, another suggested approach to approximately minimizing the augmented Lagrangian is the diagonal-quadratic approximation method, or DQA [26], which uses the following inner loop, where $\tau \in (0, 1)$ is a scalar parameter:

$$\begin{aligned}
x^{k,i+1} &\in \operatorname*{Arg\,min}_{x\in\mathbb{R}^n} \left\{ L_{c_k}(x, \bar{z}^{k,i}, \lambda^k) \right\} \\
z^{k,i+1} &\in \operatorname*{Arg\,min}_{z\in\mathbb{R}^m} \left\{ L_{c_k}(\bar{x}^{k,i}, z, \lambda^k) \right\} \\
\bar{x}^{k,i+1} &= \tau x^{k,i+1} + (1-\tau)\bar{x}^{k,i} \\
\bar{z}^{k,i+1} &= \tau x^{k,i+1} + (1-\tau)\bar{z}^{k,i}.
\end{aligned}$$

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 more complicated algorithm (57) shown in Figure 3.

Using prototype MATLAB implementations, we compared algorithms (56) and (57) to the standard ADMM with overrelaxation (40)-(42). We also included "exact" versions of the Gauss-Seidel and DQA methods in which the inner loop is iterated until it obtains a very small augmented Lagrangian subgradient (with norm not exceeding some small fixed parameter δ) or a large iteration limit is reached.

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

$$\tfrac{2}{c_k} \left| \langle w^k - x^{k,i}, y^k \rangle \right| + \|y^k\|^2 \leq \sigma \|Mx^{k,i} - z^{k,i}\|^2 \tag{56}$$

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

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

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

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

Figure 2: GS-RE algorithm.

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

$$\tfrac{2}{c_k} \left| \langle w_x^k - x^{k,i}, y_x^k \rangle + \langle w_z^k - \bar{z}^{k,i}, y_z^k \rangle \right| + \|y_x^k\|^2 + \|y_z^k\|^2 \leq \sigma \left\| Mx^{k,i} - z^{k,i} \right\|^2 \tag{57}$$

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

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

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

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

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

Figure 3: DQA-RE algorithm.

6.2 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 (2) derived from standard cancer DNA microarray datasets [4]. 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.

The lasso problem (2) may be reduced to the form (1) by taking

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

The x-minimization step (40) of the ADMM — or equivalently the first minimization in the inner loop of (56) or (57) — reduces to solving a system of linear equations involving the matrix $A^\top A + cI$, namely (in the ADMM case)

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

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 of [4], 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. The z-minimization (41) — or equivalently the second minimization in the inner loop of (56) or (57) — reduces to a very simple componentwise "vector shrinkage" operation. In the ADMM case, this shrinkage operation is

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

Since $M = I$, the multiplier update (42) is also a very simple componentwise calculation, so the backsolves required to implement the x-minimization dominate the time required for each ADMM iteration. The inner-loop termination tests of algorithms (56) or (57) require a few inner-product calculations, so the x-minimization also dominates the per-iteration time for the other algorithms tested. For a more detailed discussion of applying the ADMM to the Lasso problem (2), see for example [3].

The computational tests used a standard method for normalizing the matrix A, scaling each row to have norm 1 and scaling each element of b accordingly; 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: 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 primal variables were also initialized to zero. For all algorithms, the overall termination criterion was

$$\operatorname{dist}_\infty\left(0, \partial_x\left[\tfrac{1}{2}\|Ax - b\|^2 + \nu\|x\|_1\right]_{x=x^k}\right) \le \epsilon, \tag{58}$$

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

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

where $\text{dist}_\infty(t, S) = \inf\{\|t - s\|_\infty \mid s \in S\}$, and ϵ is a tolerance parameter we set 10^{-6}. In the relative-error augmented Lagrangian methods (56) and (57), we set the parameter σ to 0.99. For the "exact" versions of these methods, we terminated the inner loop when

$$\text{dist}_\infty\big(0, \partial_{(x,z)}\big[L_{c_k}(x^{k,i}, z^{k,i}, \lambda^k)\big]\big) \leq \frac{\epsilon}{10}, \tag{59}$$

or after 20,000 inner loop iterations, whichever came first. Note that the overall termination condition (58) is in practice much stronger than the "relative error" termination conditions proposed in [3] (this reference uses the term "relative error" in a different sense than in this paper) . For example, we found that using an "accuracy" of 10^{-4} with those conditions could still result in errors as great as 5% in the objective function, so we use the more stringent and stable condition (58) instead.

Table 1 and Figure 4 show the number of multiplier adjustment steps for each combination of the six problem instance in the test set of [4] 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 (58) was met. In all figures and tables of computational results, "GS-RE" means the relative-error Gauss-Seidel algorithm (56) and "DQA-RE" means the relative-error DQA algorithm (57), while "GS" and "DQA" mean the respective "exact" versions of this method with the inner loop instead terminated by the condition (59).

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 was made toward minimizing the augmented Lagrangian.

A different picture arises when we examine total number of inner iterations for each

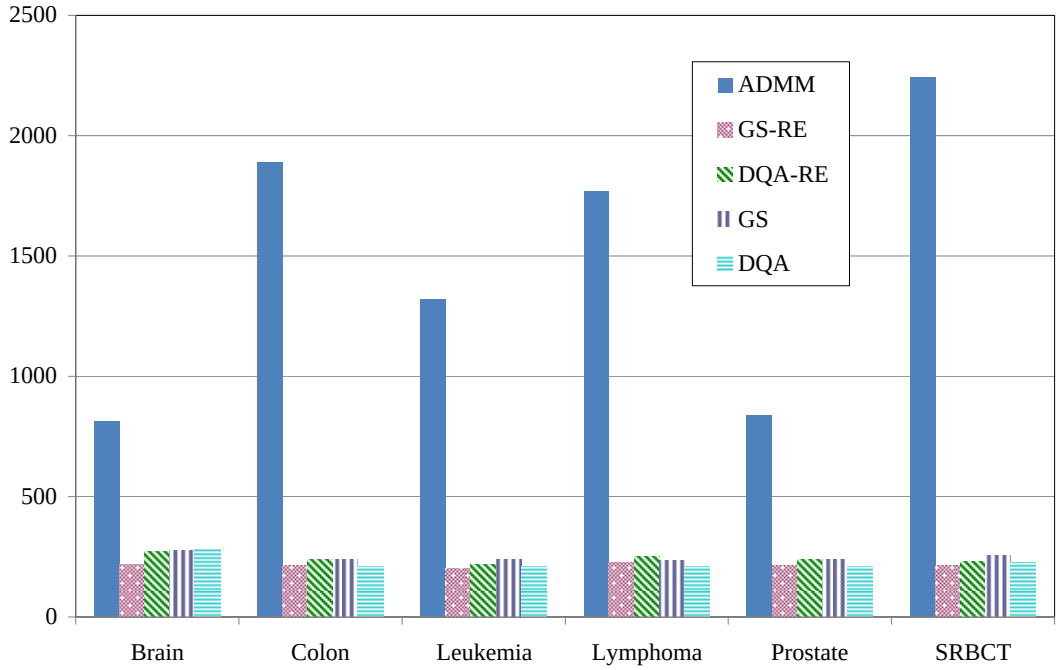

Figure 4: Lasso problems: number of outer iterations / multiplier adjustments

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

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

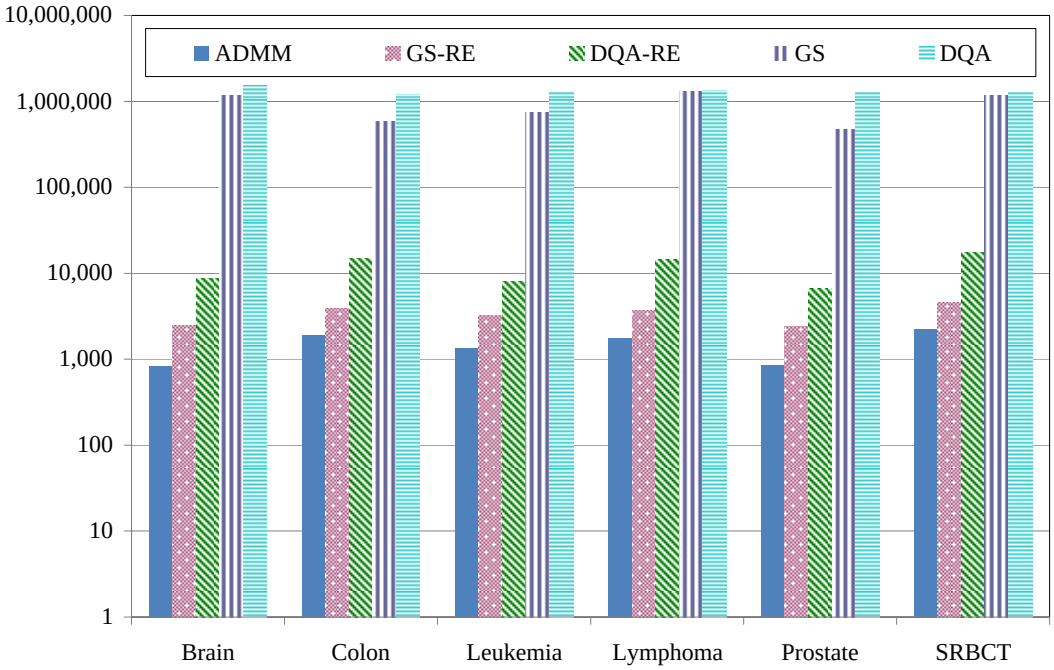

Figure 5: Cumulative number of inner iterations / x- or z-minimizations for the lasso test problems

algorithm, that is, the total number of x- or z-minimizations required; this information is shown in Table 2 and Figure 5, which uses a logarithmic vertical scale. 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 inner iterations per outer iteration was often reached for both the GS and DQA methods. Thus, the augmented Lagrangian was not truly exactly minimized in a consistent manner in any of the algorithms, hence the difference in the number of outer iterations between the GS and DQA methods in Table 1 and Figure 4 (if the augmented Lagrangian was truly exactly minimized, one would expect the same number of outer iterations for these two methods).

Figure 6 shows the same information as Figure 5, but without the very inefficient GS and DQA methods, and with a linear instead of logarithmic vertical scale. It is clear that the GS-RE method is far more efficient than DQA-RE, but ADMM requires significantly less computation than GS-RE. Figure 7 displays MATLAB run times on a Core i3 M330 2.16GHz laptop running Windows 7, again omitting the slow "exact" methods: DQA-RE is by far the slowest method, but the difference between ADMM and GS-RE is less pronounced for some problems than in Figure 6. However, MATLAB results tend to be unreliable predictors of comparative run times in other programming environments, so one should not read too much into these results.

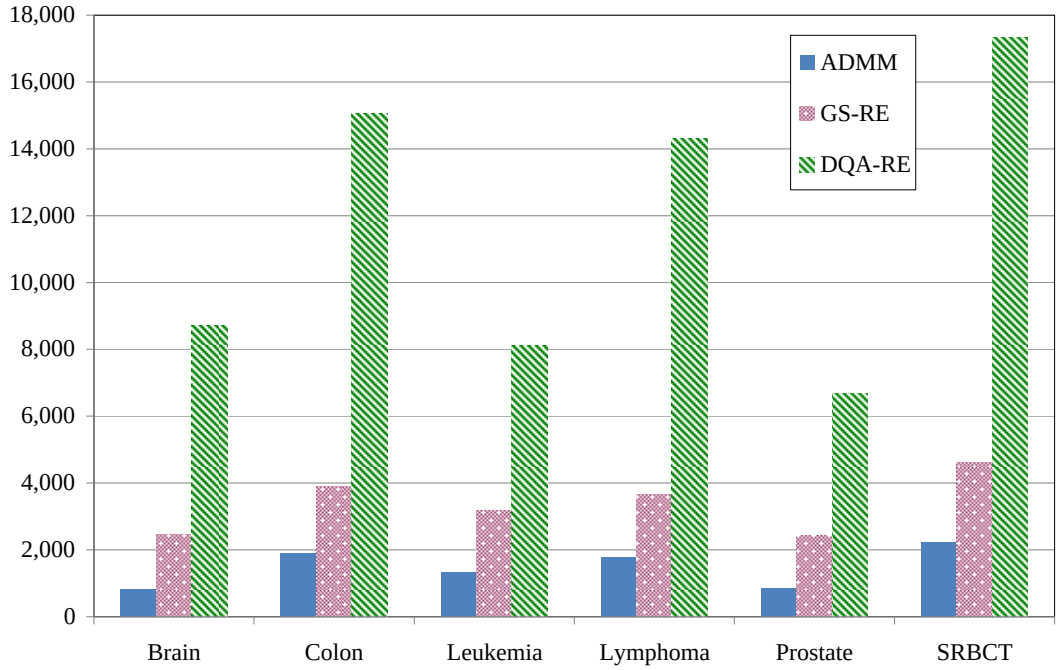

Figure 6: Cumulative number of inner iterations / x- or z-minimizations for the lasso test problems, omitting the "exact" methods

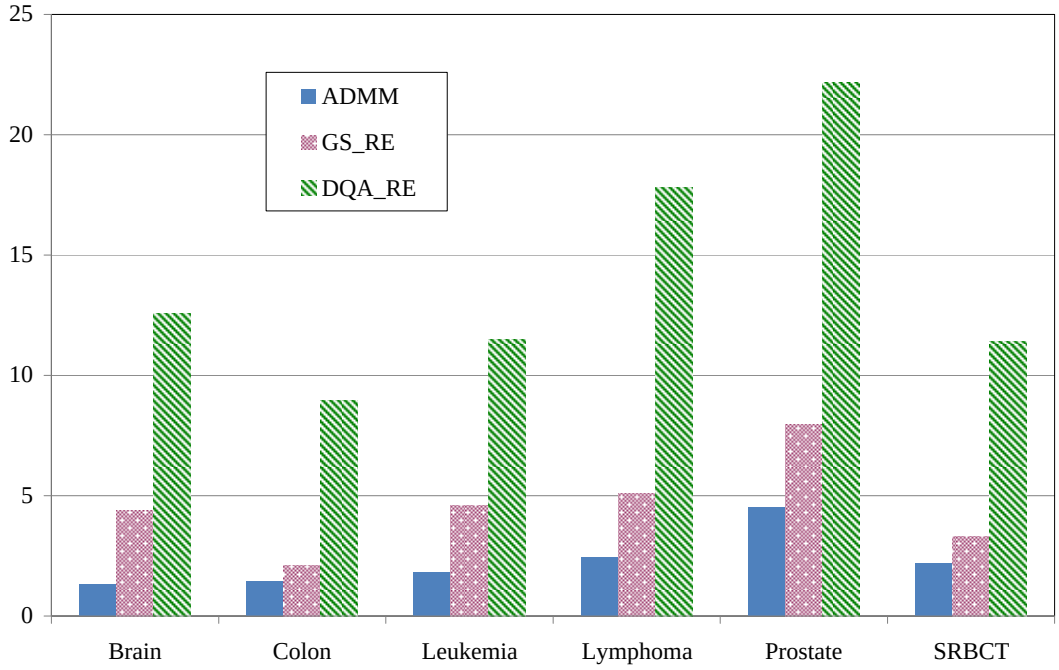

Figure 7: Run times in seconds for the lasso test problems on a Core i3 M330 2.16GHz processor, omitting the "exact" methods

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.

In these results, one notable phenomenon is that the total number of iterations the ADMM requires to optimize a problem instance is significantly below the number of iterations the Gauss-Seidel block optimization inner loop requires 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. Block-coordinate minimization is a very inefficient algorithm for the lasso problem; the ADMM is not an application of block Gauss-Seidel within the augmented Lagrangian framework, but a fundamentally different algorithm bearing only a superficial resemblance to Gauss-Seidel block-coordinate minimization.

6.3 Computational Tests for the Transportation Problem

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

$$\begin{array}{lll} \min & r^\top x & \\ \text{ST} & \sum_{j:(i,j)\in E} x_{ij} = s_i & \forall\, i \in S \\ & \sum_{i:(i,j)\in E} x_{ij} = d_j & \forall\, j \in D \\ & x_{ij} \geq 0 & \forall\, (i,j) \in E. \end{array} \tag{60}$$

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) or (3) is as follows: let $m = n = |E|$ and define

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

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

Again taking $M = I$, problem (3) is now equivalent to (60). 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 (40)-(42) to this choice of f, g, and M results in a highly parallelizable algorithm. The x-minimization (40) reduces to solving the following optimization problem independently for each source node $i \in S$:

$$\begin{array}{ll} \min & \frac{1}{2}\sum\limits_{j:\,(i,j)\in E} r_{ij}x_{ij} \; + \sum\limits_{j:\,(i,j)\in E} \lambda_{ij}x_{ij} + \frac{c}{2}\sum\limits_{j:\,(i,j)\in E} (x_{ij} - z_{ij})^2 \\ \text{ST} & \sum\limits_{j:\,(i,j)\in E} x_{ij} = s_i \\ & x_{ij} \geq 0 \qquad\qquad \forall\, j : (i,j) \in E. \end{array} \tag{61}$$

This problem is equivalent to projection onto a simplex, and a simple Lagrange multiplier analysis shows that it may be solved by sorting the breakpoints and then finding a zero crossing of a monotone piecewise-linear function. With careful use of a linear-time median-finding algorithm, the sorting operation may be eliminated, and the entire operation may be performed in linear time [19].

The z-minimization (41) reduces to a very similar set of $|D|$ independent minimizations, one for each $j \in D$:

$$\begin{array}{ll} \min & \frac{1}{2}\sum\limits_{i:\,(i,j)\in E} r_{ij}z_{ij} - \sum\limits_{i:\,(i,j)\in E} \lambda_{ij}z_{ij} + \frac{c}{2}\sum\limits_{i:\,(i,j)\in E} \left(z_{ij} - (\rho_k x_{ij}^{k+1} + (1-\rho_k)z_{ij}^k)\right)^2 \\ \text{ST} & \sum\limits_{i:\,(i,j)\in E} z_{ij} = d_j \\ & z_{ij} \geq 0 \qquad\qquad \forall\, i : (i,j) \in E. \end{array} \tag{62}$$

Finally, because $M = I$, the muliplier update (42) decomposes into $|E|$ independent tasks, one for each $(i,j) \in E$:

$$\lambda_{ij}^{k+1} = \lambda_{ij}^k + c\left(\rho_k x_{ij}^{k+1} + (1-\rho_k)z_{ij}^k - z_{ij}^{k+1}\right). \tag{63}$$

We note that this algorithm is highly parallelizable, with the x-minimization step separating into $|S|$ independent tasks, the z-minimization separating into $|D|$ independent tasks, and the multiplier update separating into $|E|$ independent tasks. 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 (40)-(42), the relative-error augmented Lagrangian methods (56) and (57), and their "exact" counterparts in a setting different from the lasso problems of the previous subsection. Note that while a few details are different, the x-minimization, z-minimization, and multiplier updates of (56), (57), and their

Problem Instance	ADMM	GS-RE	DQA-RE	GS	DQA
20×20	,1633	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 3: Transportation problems: number of outer iterations / multiplier adjustments

"exact" counterparts may be implemented very similarly in this setting to their respective counterparts (61), (62), and (63) above.

To test the algorithm, we randomly generated dense problems 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 uniformly distributed independent edge costs, as in problem generators like NETGEN [15]. 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.

In MATLAB, we implemented the same five algorithms as in Section 6.2, using a simple sorting procedure for the simplex projections; because the problems we generated did not have nodes of very high degree and because MATLAB intrinsic functions tend to be far faster than user-written code, this approach is likely to be more efficient in practice than implementing the specialized simplex projection algorithm of [19]. After a modicum of experimentation, a reasonable choice 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 limit was only encountered by the two "exact" algorithms GS and DQA, and less often than in the lasso setting.

Table 3 and Figure 8 show the number of outer iterations 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.

Table 4 and Figure 9 show the cumulative inner iteration counts, with Figure 9 employing a logarithmic vertical axis. Figure 10 showns the same data with a linear vertical axis, omitting the two slowest algorithms, GS and DQA. 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 faster for one problem. Finally, Figure 11 shows

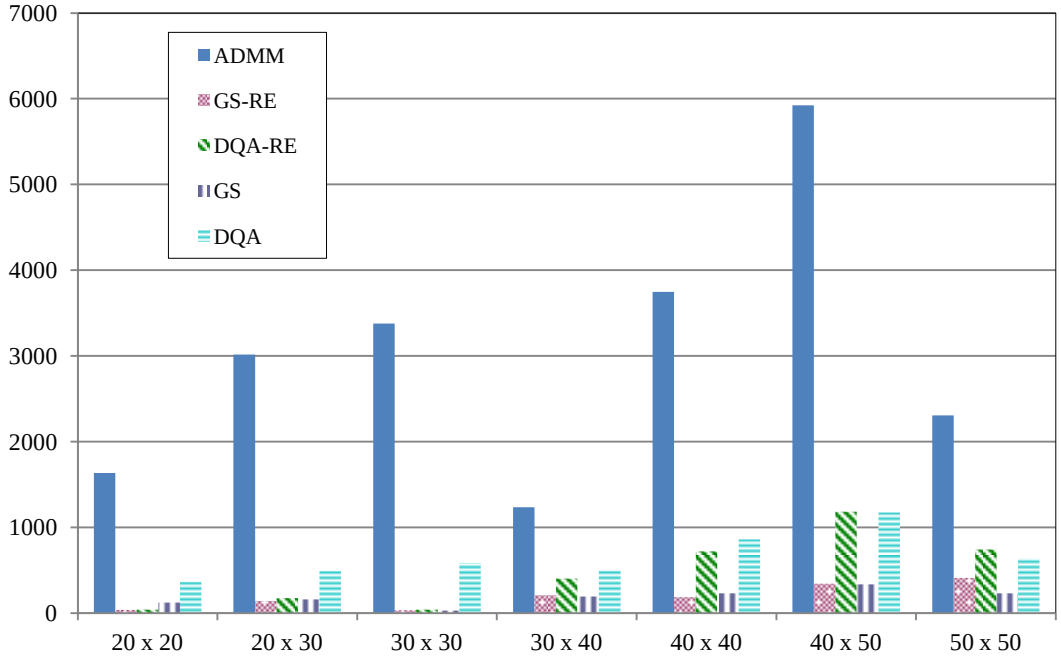

Figure 8: Transportation problems: number of outer iterations / multiplier adjustments

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

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

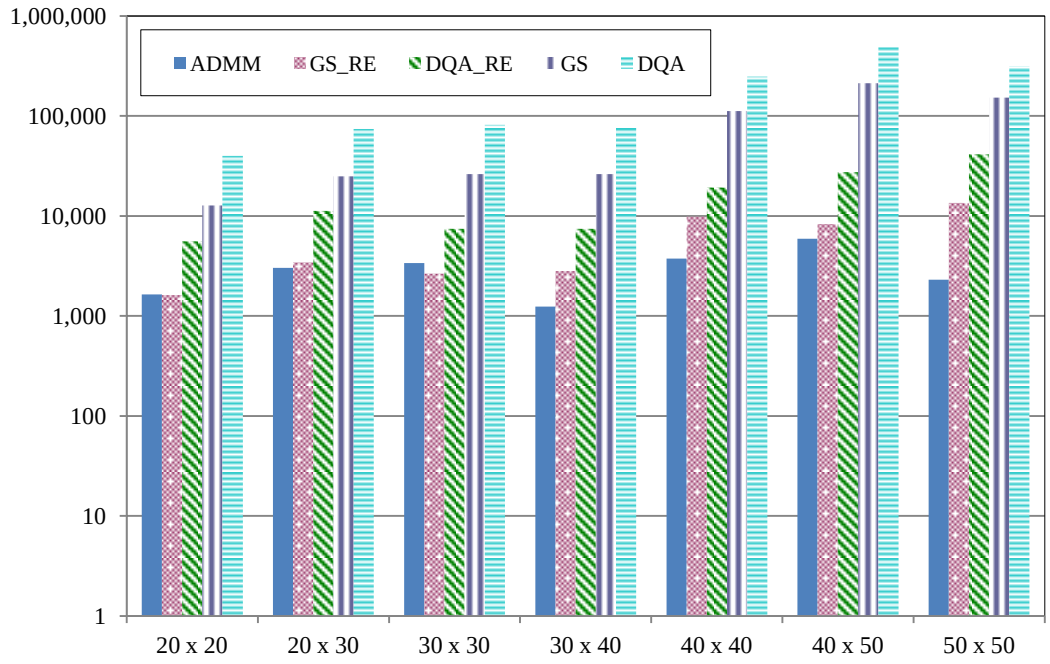

Figure 9: Cumulative number of inner iterations / x- or z-minimizations for Transportation problems

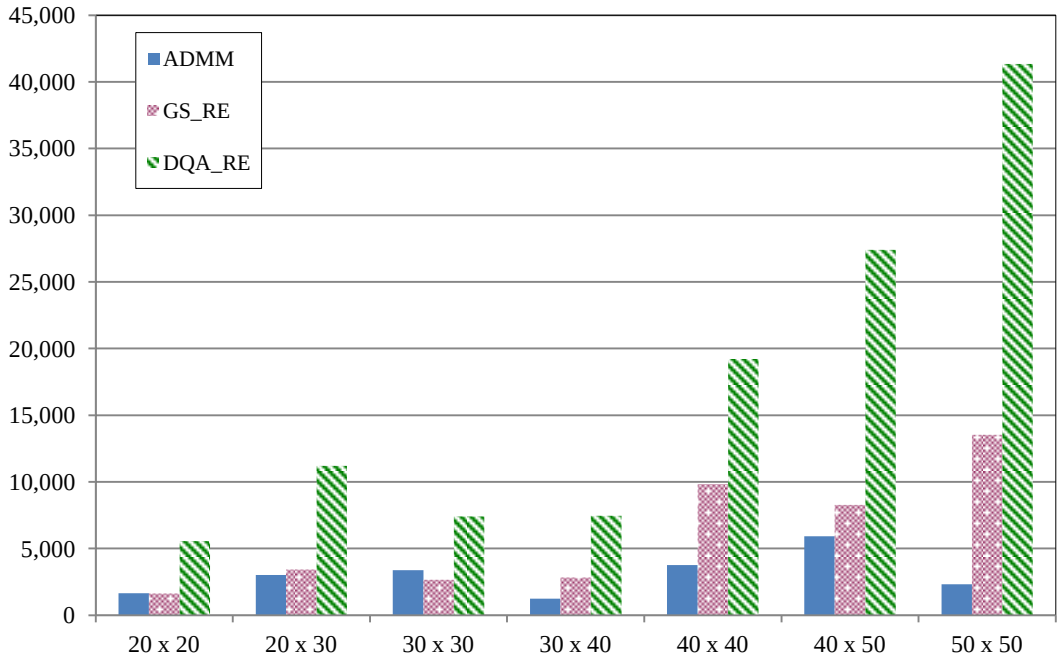

Figure 10: Cumulative number of inner iterations / x- or z-minimizations for tranportation problems, omitting the GS and DQA methods

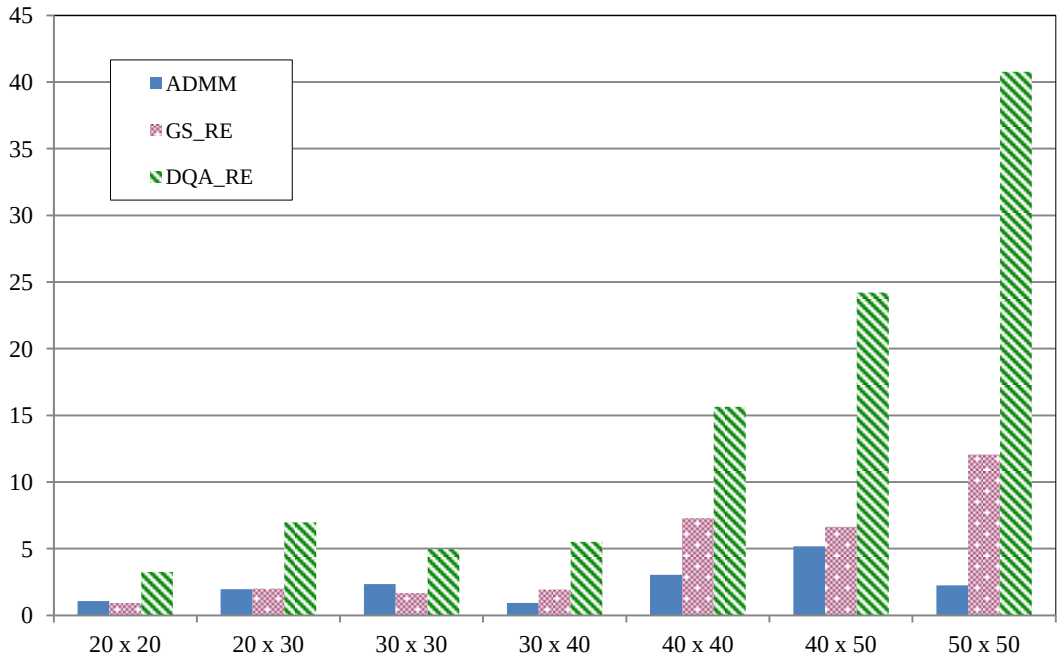

Figure 11: Transportation problems: run times in seconds on a Xeon X5472 3.00 GHz processor, omitting the GS and DQA methods

corresponding MATLAB run times on a Xeon X5472 3.00 GHz processor running Ubuntu Linux 10.04. These run times closely track the inner iteration counts of Figure 10.

7 Concluding Remarks

The central point of this paper is that the ADMM algorithm, despite superficial appearances, is not simply an approximate version of the standard augmented Lagrangian method applied to problems with a specific two-block structure. Sections 4 and 5 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 6, 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 then 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 superficially appealing notion that the ADMM is a kind of hybrid of the augmented Lagrangian and Gauss-Seidel block minimization algorithms is fundamentally misleading.

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 6, yet this phenomenon does not 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 is partially 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.

References

[1] D. P. Bertsekas. *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, 2003. With A. Nedić and A. E. Ozdaglar.

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

[3] 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(1):1–122, 2011.

[4] M. Dettling and P. Bühlmann. Finding predictive gene groups from microarray data. *Journal of Multivariate Analysis*, 90(1):106–131, 2004.

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

[6] J. Eckstein. A practical general approximation criterion for methods of multipliers based on Bregman distances. *Mathematical Programming.*, 96(1):61–86, 2003.

[7] J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.

[8] J. Eckstein and M. C. Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 10(2):218–235, 1998.

[9] J. Eckstein and P. J. S. Silva. A practical relative error criterion for augmented Lagrangians. *Mathematical Programming*, 2012. Online version DOI 10.1007/s10107-012-0528-9.

[10] M. Fortin and R. Glowinski. On decomposition-coordination methods using an augmented Lagrangian. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland: Amsterdam, 1983.

[11] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland: Amsterdam, 1983.

[12] B.-S. He. Parallel splitting augmented Lagrangian methods for monotone structured variational inequalities. *Computational Optimization and Applications*, 42(2):195–212, 2009.

[13] B.-S. He, Z. Peng, and X.-F. Wang. Proximal alternating direction-based contraction methods for separable linearly constrained convex optimization. *Frontiers of Mathematics in China*, 6(1):79–114, 2011.

[14] 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(2):337–356, 2000.

[15] D. Klingman, A. Napier, and J. Stutz. Netgen: A program for generating large-scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[16] M. A. Krasnosel'skiĭ. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1):123–127, 1955.

[17] J. Lawrence and J. E. Spingarn. On fixed points of non-expansive piecewise isometric mappings. *Proceedings of the London Mathematical Society*, 3(3):605, 1987.

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

[19] N. Maculan and G. G. de Paula, Jr. A linear-time median-finding algorithm for projecting a vector on the simplex of $\mathbb{R}^n$. *Operations Research Letters*, 8(4):219–222, 1989.

[20] G. J. Minty. Monotone (nonlinear) operators in Hilbert space. *Duke Mathematical Journal*, 29:341–346, 1962.

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

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

[23] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[24] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[25] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

[26] A. Ruszczyński. On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. *Mathematics of Operations Research*, 20(3):634–656, 1995.

[27] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.