A General Analysis of the Convergence of ADMM

Robert Nishihara RKN@EECS.BERKELEY.EDU
Laurent Lessard LESSARD@BERKELEY.EDU
Benjamin Recht BRECHT@EECS.BERKELEY.EDU
Andrew Packard APACKARD@BERKELEY.EDU
Michael I. Jordan JORDAN@EECS.BERKELEY.EDU
University of California, Berkeley, CA 94720 USA

Abstract

We provide a new proof of the linear convergence of the alternating direction method of multipliers (ADMM) when one of the objective terms is strongly convex. Our proof is based on a framework for analyzing optimization algorithms introduced in Lessard et al. (2014), reducing algorithm convergence to verifying the stability of a dynamical system. This approach generalizes a number of existing results and obviates any assumptions about specific choices of algorithm parameters. On a numerical example, we demonstrate that minimizing the derived bound on the convergence rate provides a practical approach to selecting algorithm parameters for particular ADMM instances. We complement our upper bound by constructing a nearly-matching lower bound on the worst-case rate of convergence.

1. Introduction

The alternating direction method of multipliers (ADMM) seeks to solve the problem

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array} \tag{1}$$

with variables $x \in \mathbb{R}^p$ and $z \in \mathbb{R}^q$ and constants $A \in \mathbb{R}^{r \times p}$, $B \in \mathbb{R}^{r \times q}$, and $c \in \mathbb{R}^r$. ADMM was introduced in Glowinski & Marroco (1975) and Gabay & Mercier (1976). More recently, it has found applications in a variety of distributed settings such as model fitting, resource allocation, and classification. A partial list of examples includes Bioucas-Dias & Figueiredo (2010); Wahlberg et al. (2012); Bird (2014); Forero et al. (2010); Sedghi et al. (2014); Li et al. (2014); Wang & Banerjee (2012); Zhang et al. (2012); Meshi & Globerson (2011); Wang et al. (2013); Aslan et al. (2013); Forouzan & Ihler (2013); Romera-Paredes & Pontil (2013); Behmardi et al. (2014); Zhang & Kwok (2014). See Boyd et al. (2011) for an overview.

Part of the appeal of ADMM is the fact that, in many contexts, the algorithm updates lend themselves to parallel implementations. The algorithm is given in Algorithm 1. We refer to $\rho > 0$ as the step-size parameter.

Algorithm 1 Alternating Direction Method of Multipliers

```
1: Input: functions $f$ and $g$, matrices $A$ and $B$, vector $c$, parameter $\rho$
2: Initialize $x_0, z_0, u_0$
3: repeat
4:     $x_{k+1} = \arg\min_x f(x) + \frac{\rho}{2}\|Ax + Bz_k - c + u_k\|^2$
5:     $z_{k+1} = \arg\min_z g(z) + \frac{\rho}{2}\|Ax_{k+1} + Bz - c + u_k\|^2$
6:     $u_{k+1} = u_k + Ax_{k+1} + Bz_{k+1} - c.$
7: until meet stopping criterion
```

A popular variant of Algorithm 1 is over-relaxed ADMM, which introduces an additional parameter α and replaces each instance of Ax_{k+1} in the z and u updates in Algorithm 1 with

$$\alpha A x_{k+1} - (1 - \alpha)(Bz_k - c).$$

The parameter α is typically chosen to lie in the interval $(0, 2]$, but we demonstrate in Section 8 that a larger set of choices can lead to convergence. Over-relaxed ADMM is described in Algorithm 2. When $\alpha = 1$, Algorithm 2 and Algorithm 1 coincide. We will analyze Algorithm 2.

The conventional wisdom that ADMM works well without any tuning (Boyd et al., 2011), for instance by setting $\rho = 1$, is often not borne out in practice. Algorithm 1 can be challenging to tune, and Algorithm 2 is even harder. We use the machinery developed in this paper to make reasonable recommendations for setting ρ and α when some information about f is available (Section 8).

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Algorithm 2 Over-Relaxed Alternating Direction Method of Multipliers

```
1: Input: functions $f$ and $g$, matrices $A$ and $B$, vector $c$,
   parameters $\rho$ and $\alpha$
2: Initialize $x_0, z_0, u_0$
3: repeat
4:     $x_{k+1} = \arg\min_x f(x) + \frac{\rho}{2}\|Ax + Bz_k - c + u_k\|^2$
5:     $z_{k+1} = \arg\min_z g(z) + \frac{\rho}{2}\|\alpha Ax_{k+1} - (1-\alpha)Bz_k + Bz - \alpha c + u_k\|^2$
6:     $u_{k+1} = u_k + \alpha Ax_{k+1} - (1-\alpha)Bz_k + Bz_{k+1} - \alpha c$
7: until meet stopping criterion
```

In this paper, we give an upper bound on the linear rate of convergence of Algorithm 2 for all ρ and α (Theorem 7), and we give a nearly-matching lower bound (Theorem 8).

Importantly, we show that we can prove convergence rates for Algorithm 2 by numerically solving a 4×4 semidefinite program (Theorem 6). When we change the parameters of Algorithm 2, the semidefinite program changes. Whereas prior work requires a new proof of convergence for every change to the algorithm, our work automates that process.

Our work builds on the integral quadratic constraint framework introduced in Lessard et al. (2014), which uses ideas from robust control to analyze optimization algorithms that can be cast as discrete-time linear dynamical systems. Related ideas, in the context of feedback control, appear in Corless (1990); D'Alto & Corless (2013). Our work provides a flexible framework for analyzing variants of Algorithm 1, including those like Algorithm 2 created by the introduction of additional parameters. In Section 7, we compare our results to prior work.

2. Preliminaries and Notation

Let $\overline{\mathbb{R}}$ denote the extended real numbers $\mathbb{R} \cup \{+\infty\}$. Suppose that $f\colon \mathbb{R}^d \to \overline{\mathbb{R}}$ is convex and differentiable, and let ∇f denote the gradient of f. We say that f is strongly convex with parameter $m > 0$ if for all $x, y \in \mathbb{R}^d$, we have

$$f(x) \geq f(y) + \nabla f(y)^\top (x-y) + \tfrac{m}{2}\|x-y\|^2.$$

When ∇f is Lipschitz continuous with parameter L, then

$$f(x) \leq f(y) + \nabla f(y)^\top (x-y) + \tfrac{L}{2}\|x-y\|^2.$$

For $0 < m \leq L < \infty$, let $S_d(m, L)$ denote the set of differentiable convex functions $f\colon \mathbb{R}^d \to \overline{\mathbb{R}}$ that are strongly convex with parameter m and whose gradients are Lipschitz continuous with parameter L. We let $S_d(0, \infty)$ denote the set of convex functions $\mathbb{R}^d \to \overline{\mathbb{R}}$. In general, we let ∂f denote the subdifferential of f. We denote the d-dimensional identity matrix by I_d and the d-dimensional zero matrix by 0_d. We will use the following results.

Lemma 1. *Suppose that $f \in S_d(m, L)$, where $0 < m \leq L < \infty$. Suppose that $b_1 = \nabla f(a_1)$ and $b_2 = \nabla f(a_2)$. Then*

$$\begin{bmatrix} a_1 - a_2 \\ b_1 - b_2 \end{bmatrix}^\top \begin{bmatrix} -2mLI_d & (m+L)I_d \\ (m+L)I_d & -2I_d \end{bmatrix} \begin{bmatrix} a_1 - a_2 \\ b_1 - b_2 \end{bmatrix} \geq 0.$$

Proof. The Lipschitz continuity of ∇f implies the co-coercivity of ∇f, that is

$$(a_1 - a_2)^\top (b_1 - b_2) \geq \tfrac{1}{L}\|b_1 - b_2\|^2.$$

Note that $f(x) - \frac{m}{2}\|x\|^2$ is convex and its gradient is Lipschitz continuous with parameter $L - m$. Applying the co-coercivity condition to this function and rearranging gives

$$(m+L)(a_1-a_2)^\top(b_1-b_2) \geq mL\|a_1-a_2\|^2 + \|b_1-b_2\|^2,$$

which can be put in matrix form to complete the proof. □

Lemma 2. *Suppose that $f \in S_d(0, \infty)$, and suppose that $b_1 \in \partial f(a_1)$ and $b_2 \in \partial f(a_2)$. Then*

$$\begin{bmatrix} a_1 - a_2 \\ b_1 - b_2 \end{bmatrix}^\top \begin{bmatrix} 0_d & I_d \\ I_d & 0_d \end{bmatrix} \begin{bmatrix} a_1 - a_2 \\ b_1 - b_2 \end{bmatrix} \geq 0.$$

Lemma 2 is simply the statement that the subdifferential of a convex function is a monotone operator.

When M is a matrix, we use κ_M to denote the condition number of M. For example, $\kappa_A = \sigma_1(A)/\sigma_p(A)$, where $\sigma_1(A)$ and $\sigma_p(A)$ denote the largest and smallest singular values of the matrix A. When $f \in S_d(m, L)$, we let $\kappa_f = \frac{L}{m}$ denote the condition number of f. We denote the Kronecker product of matrices M and N by $M \otimes N$.

3. ADMM as a Dynamical System

We group our assumptions together in Assumption 3.

Assumption 3. *We assume that f and g are convex, closed, and proper. We assume that for some $0 < m \leq L < \infty$, we have $f \in S_p(m, L)$ and $g \in S_q(0, \infty)$. We assume that A is invertible and that B has full column rank.*

The assumption that f and g are closed (their sublevel sets are closed) and proper (they neither take on the value $-\infty$ nor are they uniformly equal to $+\infty$) is standard.

We begin by casting over-relaxed ADMM as a discrete-time dynamical system with state sequence (ξ_k), input sequence (ν_k), and output sequences (w_k^1) and (w_k^2) satisfying the recursions

$$\xi_{k+1} = (\hat{A} \otimes I_r)\xi_k + (\hat{B} \otimes I_r)\nu_k \quad (2a)$$

$$w_k^1 = (\hat{C}^1 \otimes I_r)\xi_k + (\hat{D}^1 \otimes I_r)\nu_k \quad (2b)$$

$$w_k^2 = (\hat{C}^2 \otimes I_r)\xi_k + (\hat{D}^2 \otimes I_r)\nu_k \quad (2c)$$

for particular matrices $\hat{A}$, $\hat{B}$, $\hat{C}^1$, $\hat{D}^1$, $\hat{C}^2$, and $\hat{D}^2$ (whose dimensions do not depend on any problem parameters).

First define the functions $\hat{f}, \hat{g} \colon \mathbb{R}^r \to \overline{\mathbb{R}}$ via

$$\begin{aligned}\hat{f} &= (\rho^{-1} f) \circ A^{-1} \\ \hat{g} &= (\rho^{-1} g) \circ B^{\dagger} + \mathbb{I}_{\operatorname{im} B},\end{aligned} \tag{3}$$

where $B^{\dagger}$ is any left inverse of B and where $\mathbb{I}_{\operatorname{im} B}$ is the $\{0, \infty\}$-indicator function of the image of B. We define $\kappa = \kappa_f \kappa_A^2$ and to normalize we define

$$\hat{m} = \frac{m}{\sigma_1^2(A)} \quad \hat{L} = \frac{L}{\sigma_p^2(A)} \quad \rho = (\hat{m}\hat{L})^{\frac{1}{2}} \rho_0. \tag{4}$$

Note that under Assumption 3,

$$\hat{f} \in S_p(\rho_0^{-1} \kappa^{-\frac{1}{2}}, \rho_0^{-1} \kappa^{\frac{1}{2}}) \tag{5a}$$
$$\hat{g} \in S_q(0, \infty). \tag{5b}$$

To define the relevant sequences, let the sequences (x_k), (z_k), and (u_k) be generated by Algorithm 2 with parameters α and ρ. Define the sequences (r_k) and (s_k) by $r_k = Ax_k$ and $s_k = Bz_k$ and the sequence (ξ_k) by

$$\xi_k = \begin{bmatrix} s_k \\ u_k \end{bmatrix}.$$

We define the sequence (ν_k) as in Proposition 4.

Proposition 4. *There exist sequences (β_k) and (γ_k) with $\beta_k = \nabla \hat{f}(r_k)$ and $\gamma_k \in \partial \hat{g}(s_k)$ such that when we define the sequence (ν_k) by*

$$\nu_k = \begin{bmatrix} \beta_{k+1} \\ \gamma_{k+1} \end{bmatrix},$$

then (ξ_k) and (ν_k) satisfy (2a) with the matrices

$$\hat{A} = \begin{bmatrix} 1 & \alpha - 1 \\ 0 & 0 \end{bmatrix} \qquad \hat{B} = \begin{bmatrix} \alpha & -1 \\ 0 & -1 \end{bmatrix}. \tag{6}$$

Proof. Using the fact that A has full rank, we rewrite the update rule for x from Algorithm 2 as

$$x_{k+1} = A^{-1} \operatorname*{arg\,min}_r f(A^{-1} r) + \tfrac{\rho}{2} \|r + s_k - c + u_k\|^2.$$

Multiplying through by A, we can write

$$r_{k+1} = \operatorname*{arg\,min}_r \hat{f}(r) + \tfrac{1}{2} \|r + s_k - c + u_k\|^2.$$

This implies that

$$0 = \nabla \hat{f}(r_{k+1}) + r_{k+1} + s_k - c + u_k,$$

and so

$$r_{k+1} = -s_k - u_k + c - \beta_{k+1}, \tag{7}$$

where $\beta_{k+1} = \nabla \hat{f}(r_{k+1})$. In the same spirit, we rewrite the update rule for z as

$$\begin{aligned} s_{k+1} = \operatorname*{arg\,min}_s \hat{g}(s) \\ + \tfrac{1}{2} \|\alpha r_{k+1} - (1-\alpha) s_k + s - \alpha c + u_k\|^2. \end{aligned}$$

It follows that there exists some $\gamma_{k+1} \in \partial \hat{g}(s_{k+1})$ such that

$$0 = \gamma_{k+1} + \alpha r_{k+1} - (1-\alpha) s_k + s_{k+1} - \alpha c + u_k.$$

It follows then that

$$\begin{aligned} s_{k+1} &= -\alpha r_{k+1} + (1-\alpha) s_k + \alpha c - u_k - \gamma_{k+1} \\ &= s_k - (1-\alpha) u_k + \alpha \beta_{k+1} - \gamma_{k+1}, \end{aligned} \tag{8}$$

where the second equality follows by substituting in (7). Combining (7) and (8) to simplify the u update, we have

$$\begin{aligned} u_{k+1} &= u_k + \alpha r_{k+1} - (1-\alpha) s_k + s_{k+1} - \alpha c \\ &= -\gamma_{k+1}. \end{aligned} \tag{9}$$

Together, (8) and (9) confirm the relation in (2a). □

Corollary 5. *Define the sequences (β_k) and (γ_k) as in Proposition 4. Define the sequences (w_k^1) and (w_k^2) via*

$$w_k^1 = \begin{bmatrix} r_{k+1} - c \\ \beta_{k+1} \end{bmatrix} \qquad w_k^2 = \begin{bmatrix} s_{k+1} \\ \gamma_{k+1} \end{bmatrix}.$$

Then the sequences (ξ_k), (ν_k), (w_k^1), and (w_k^2) satisfy (2b) and (2c) with the matrices

$$\begin{aligned} \hat{C}^1 &= \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} & \hat{D}^1 &= \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \\ \hat{C}^2 &= \begin{bmatrix} 1 & \alpha - 1 \\ 0 & 0 \end{bmatrix} & \hat{D}^2 &= \begin{bmatrix} \alpha & -1 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{10}$$

4. Convergence Rates from Semidefinite Programming

Now, in Theorem 6, we make use of the perspective developed in Section 3 to obtain convergence rates for Algorithm 2. This is essentially the same as the main result of Lessard et al. (2014), and we include it because it is simple and self-contained.

Theorem 6. *Suppose that Assumption 3 holds. Let the sequences (x_k), (z_k), and (u_k) be generated by running Algorithm 2 with step size $\rho = (\hat{m}\hat{L})^{\frac{1}{2}} \rho_0$ and with over-relaxation parameter α. Suppose that (x_*, z_*, u_*) is a fixed point of Algorithm 2, and define*

$$\varphi_k = \begin{bmatrix} z_k \\ u_k \end{bmatrix} \qquad \varphi_* = \begin{bmatrix} z_* \\ u_* \end{bmatrix}.$$

Fix $0 < \tau < 1$, and suppose that there exist a 2×2 positive definite matrix $P \succ 0$ and nonnegative constants $\lambda^1, \lambda^2 \geq 0$ such that the 4×4 linear matrix inequality

$$0 \succeq \begin{bmatrix} \hat{A}^\top P \hat{A} - \tau^2 P & \hat{A}^\top P \hat{B} \\ \hat{B}^\top P \hat{A} & \hat{B}^\top P \hat{B} \end{bmatrix} + \begin{bmatrix} \hat{C}^1 & \hat{D}^1 \\ \hat{C}^2 & \hat{D}^2 \end{bmatrix}^\top \begin{bmatrix} \lambda^1 M^1 & 0 \\ 0 & \lambda^2 M^2 \end{bmatrix} \begin{bmatrix} \hat{C}^1 & \hat{D}^1 \\ \hat{C}^2 & \hat{D}^2 \end{bmatrix} \tag{11}$$

is satisfied, where $\hat{A}$ and $\hat{B}$ are defined in (6), where $\hat{C}^1$, $\hat{D}^1$, $\hat{C}^2$, and $\hat{D}^2$ are defined in (10), and where M^1 and M^2 are given by

$$M^1 = \begin{bmatrix} -2\rho_0^{-2} & \rho_0^{-1}(\kappa^{-\frac{1}{2}} + \kappa^{\frac{1}{2}}) \\ \rho_0^{-1}(\kappa^{-\frac{1}{2}} + \kappa^{\frac{1}{2}}) & -2 \end{bmatrix}$$

$$M^2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Then for all $k \geq 0$, we have

$$\|\varphi_k - \varphi_*\| \leq \kappa_B \sqrt{\kappa_P} \|\varphi_0 - \varphi_*\| \tau^k.$$

Proof. Define r_k, s_k, β_k, γ_k, ξ_k, ν_k, w_k^1, and w_k^2 as before. Choose $r_* = Ax_*$, $s_* = Bz_*$, and

$$w_*^1 = \begin{bmatrix} r_* - c \\ \beta_* \end{bmatrix} \quad w_*^2 = \begin{bmatrix} s_* \\ \gamma_* \end{bmatrix} \quad \xi_* = \begin{bmatrix} s_* \\ u_* \end{bmatrix} \quad \nu_* = \begin{bmatrix} \beta_* \\ \gamma_* \end{bmatrix}$$

such that $(\xi_*, \nu_*, w_*^1, w_*^2)$ is a fixed point of the dynamics of (2) and satisfying $\beta_* = \nabla \hat{f}(r_*)$, $\gamma_* \in \partial \hat{g}(s_*)$. Now, consider the Kronecker product of the right hand side of (11) and I_r. Multiplying this on the left and on the right by $\begin{bmatrix} (\xi_j - \xi_*)^\top & (\nu_j - \nu_*)^\top \end{bmatrix}$ and its transpose, respectively, we find

$$\begin{aligned} 0 \geq\ & (\xi_{j+1} - \xi_*)^\top P (\xi_{j+1} - \xi_*) \\ & - \tau^2 (\xi_j - \xi_*)^\top P (\xi_j - \xi_*) \\ & + \lambda^1 (w_j^1 - w_*^1)^\top M^1 (w_j^1 - w_*^1) \\ & + \lambda^2 (w_j^2 - w_*^2)^\top M^2 (w_j^2 - w_*^2). \end{aligned} \tag{12}$$

Lemma 1 and (5a) show that the third term on the right hand side of (12) is nonnegative. Lemma 2 and (5b) show that the fourth term on the right hand side of (12) is nonnegative. It follows that

$$(\xi_{j+1} - \xi_*)^\top P (\xi_{j+1} - \xi_*) \leq \tau^2 (\xi_j - \xi_*)^\top P (\xi_j - \xi_*).$$

Inducting from $j = 0$ to $k - 1$, we see that

$$(\xi_k - \xi_*)^\top P (\xi_k - \xi_*) \leq \tau^{2k} (\xi_0 - \xi_*)^\top P (\xi_0 - \xi_*),$$

for all k. It follows that

$$\|\xi_k - \xi_*\| \leq \sqrt{\kappa_P} \|\xi_0 - \xi_*\| \tau^k.$$

The conclusion follows. □

For fixed values of α, ρ_0, $\hat{m}$, $\hat{L}$, and τ, the feasibility of (11) is a semidefinite program with variables P, λ^1, and λ^2. We perform a binary search over τ to find the minimal rate τ such that the linear matrix inequality in (11) is satisfied. The results are shown in Figure 1 for a wide range of condition numbers κ, for $\alpha = 1.5$, and for several choices of ρ_0. In Figure 2, we plot the values $-1/\log \tau$ to show the number of iterations required to achieve a desired accuracy.

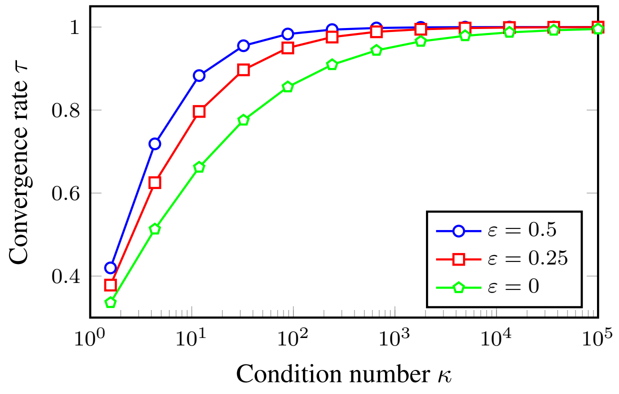

Figure 1. For $\alpha = 1.5$ and for several choices of ε in $\rho_0 = \kappa^\varepsilon$, we plot the minimal rate τ for which the linear matrix inequality in (11) is satisfied as a function of κ.

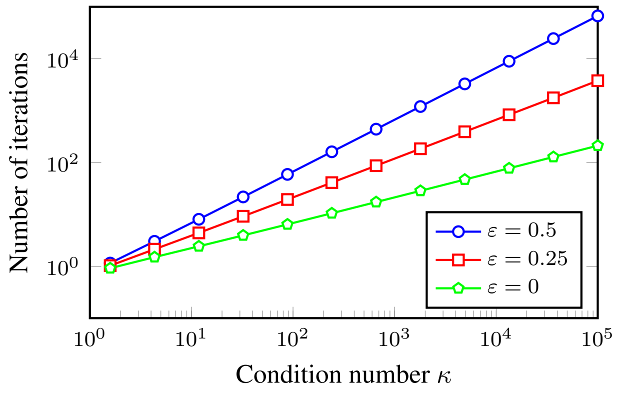

Figure 2. For $\alpha = 1.5$ and for several choices of ε in $\rho_0 = \kappa^\varepsilon$, we compute the minimal rate τ such that the linear matrix inequality in (11) is satisfied, and we plot $-1/\log \tau$ as a function of κ.

Note that when $\rho_0 = \kappa^\varepsilon$, the matrix M^1 is given by

$$M^1 = \begin{bmatrix} -2\kappa^{-2\varepsilon} & \kappa^{-\frac{1}{2}-\varepsilon} + \kappa^{\frac{1}{2}-\varepsilon} \\ \kappa^{-\frac{1}{2}-\varepsilon} + \kappa^{\frac{1}{2}-\varepsilon} & -2 \end{bmatrix},$$

and so the linear matrix inequality in (11) depends only on κ and not on $\hat{m}$ and $\hat{L}$. Therefore, we will consider step sizes of this form (recall from (4) that $\rho = (\hat{m}\hat{L})^{\frac{1}{2}} \rho_0$). The choice $\varepsilon = 0$ is common in the literature (Giselsson & Boyd, 2014), but requires the user to know the strong-convexity parameter $\hat{m}$. We also consider the choice $\varepsilon =$

0.5, which produces worse guarantees, but does not require knowledge of $\hat{m}$.

One weakness of Theorem 6 is the fact that the rate we produce is not given as a function of κ. To use Theorem 6 as stated, we first specify the condition number (for example, $\kappa = 1000$). Then we search for the minimal τ such that (11) is feasible. This produces an upper bound on the convergence rate of Algorithm 2 (for example, $\tau = 0.9$). To remedy this problem, in Section 5, we demonstrate how Theorem 6 can be used to obtain the convergence rate of Algorithm 2 as a symbolic function of the step size ρ and the over-relaxation parameter α.

5. Symbolic Rates for Various ρ and α

In Section 4, we demonstrated how to use semidefinite programming to produce numerical convergence rates. That is, given a choice of algorithm parameters and the condition number κ, we could determine the convergence rate of Algorithm 2. In this section, we show how Theorem 6 can be used to prove symbolic convergence rates. That is, we describe the convergence rate of Algorithm 2 as a function of ρ, α, and κ. In Theorem 7, we prove the linear convergence of Algorithm 2 for all choices $\alpha \in (0, 2)$ and $\rho = (\hat{m}\hat{L})^{\frac{1}{2}}\kappa^{\varepsilon}$, with $\varepsilon \in (-\infty, \infty)$. This result generalizes a number of results in the literature. As two examples, Giselsson & Boyd (2014) consider the case $\varepsilon = 0$ and Deng & Yin (2012) consider the case $\alpha = 1$ and $\varepsilon = 0.5$.

The rate given in Theorem 7 is loose by a factor of four relative to the lower bound given in Theorem 8. However, weakening the rate by a constant factor eases the proof by making it easier to find a certificate for use in (11).

Theorem 7. *Suppose that Assumption 3 holds. Let the sequences (x_k), (z_k), and (u_k) be generated by running Algorithm 2 with parameter $\alpha \in (0, 2)$ and with step size $\rho = (\hat{m}\hat{L})^{\frac{1}{2}}\kappa^{\varepsilon}$, where $\varepsilon \in (-\infty, \infty)$. Define x_*, z_*, u_*, φ_k, and φ_* as in Theorem 6. Then for all sufficiently large κ, we have*

$$\|\varphi_k - \varphi_*\| \leq C\|\varphi_0 - \varphi_*\| \left(1 - \frac{\alpha}{2\kappa^{0.5+|\varepsilon|}}\right)^k,$$

where

$$C = \kappa_B \sqrt{\max\left\{\tfrac{\alpha}{2-\alpha}, \tfrac{2-\alpha}{\alpha}\right\}}.$$

Proof. We claim that for all sufficiently large κ, the linear matrix inequality in (11) is satisfied with the rate $\tau = 1 - \frac{\alpha}{2\kappa^{0.5+|\varepsilon|}}$ and with certificate

$$\lambda^1 = \alpha\kappa^{\varepsilon-0.5} \quad \lambda^2 = \alpha \quad P = \begin{bmatrix} 1 & \alpha - 1 \\ \alpha - 1 & 1 \end{bmatrix}.$$

The matrix on the right hand side of (11) can be expressed as $-\frac{1}{4}\alpha\kappa^{-2}M$, where M is a symmetric 4×4 matrix whose last row and column consist of zeros. We wish to prove that M is positive semidefinite for all sufficiently large κ. To do so, we consider the cases $\varepsilon \geq 0$ and $\varepsilon < 0$ separately, though the two cases will be nearly identical. First suppose that $\varepsilon \geq 0$. In this case, the nonzero entries of M are specified by

$$\begin{aligned}
M_{11} &= \alpha\kappa^{1-2\varepsilon} + 4\kappa^{\frac{3}{2}-\varepsilon} \\
M_{12} &= \alpha^2\kappa^{1-2\varepsilon} - \alpha\kappa^{1-2\varepsilon} + 12\kappa^{\frac{3}{2}-\varepsilon} - 4\alpha\kappa^{\frac{3}{2}-\varepsilon} \\
M_{13} &= 4\kappa + 8\kappa^{\frac{3}{2}-\varepsilon} \\
M_{22} &= 8\kappa^2 - 4\alpha\kappa^2 + \alpha\kappa^{1-2\varepsilon} + 4\kappa^{\frac{3}{2}-\varepsilon} \\
M_{23} &= 4\kappa + 8\kappa^2 - 4\alpha\kappa^2 + 8\kappa^{\frac{3}{2}-\varepsilon} \\
M_{33} &= 8\kappa + 8\kappa^2 - 4\alpha\kappa^2 + 8\kappa^{\frac{3}{2}-\varepsilon} + 8\kappa^{\frac{3}{2}+\varepsilon}.
\end{aligned}$$

We show that each of the first three leading principal minors of M is positive for sufficiently large κ. To understand the behavior of the leading principal minors, it suffices to look at their leading terms. For large κ, the first leading principal minor (which is simple M_{11}) is dominated by the term $4\kappa^{\frac{3}{2}-\varepsilon}$, which is positive. Similarly, the second leading principal minor is dominated by the term $16(2 - \alpha)\kappa^{\frac{7}{2}-\varepsilon}$, which is positive. When $\varepsilon > 0$, the third leading principal minor is dominated by the term $128(2 - \alpha)\kappa^5$, which is positive. When $\varepsilon = 0$, the third leading principal minor is dominated by the term $64\alpha(2 - \alpha)^2\kappa^5$, which is positive. Since these leading coefficients are all positive, it follows that for all sufficiently large κ, the matrix M is positive semidefinite.

Now suppose that $\varepsilon < 0$. In this case, the nonzero entries of M are specified by

$$\begin{aligned}
M_{11} &= 8\kappa^{\frac{3}{2}-\varepsilon} - 4\kappa^{\frac{3}{2}+\varepsilon} + \alpha\kappa^{1+2\varepsilon} \\
M_{12} &= 8\kappa^{\frac{3}{2}-\varepsilon} + 4\kappa^{\frac{3}{2}+\varepsilon} - 4\alpha\kappa^{\frac{3}{2}+\varepsilon} - \alpha\kappa^{1+2\varepsilon} + \alpha^2\kappa^{1+2\varepsilon} \\
M_{13} &= 4\kappa + 8\kappa^{\frac{3}{2}-\varepsilon} \\
M_{22} &= 8\kappa^2 - 4\alpha\kappa^2 + 8\kappa^{\frac{3}{2}-\varepsilon} - 4\kappa^{\frac{3}{2}+\varepsilon} + \alpha\kappa^{1+2\varepsilon} \\
M_{23} &= 4\kappa + 8\kappa^2 - 4\alpha\kappa^2 + 8\kappa^{\frac{3}{2}-\varepsilon} \\
M_{33} &= 8\kappa + 8\kappa^2 - 4\alpha\kappa^2 + 8\kappa^{\frac{3}{2}-\varepsilon} + 8\kappa^{\frac{3}{2}+\varepsilon}.
\end{aligned}$$

As before, we show that each of the first three leading principal minors of M is positive. For large κ, the first leading principal minor (which is simple M_{11}) is dominated by the term $8\kappa^{\frac{3}{2}-\varepsilon}$, which is positive. Similarly, the second leading principal minor is dominated by the term $32(2 - \alpha)\kappa^{\frac{7}{2}-\varepsilon}$, which is positive. The third leading principal minor is dominated by the term $128(2 - \alpha)\kappa^5$, which is positive. Since these leading coefficients are all positive, it follows that for all sufficiently large κ, the matrix M is positive semidefinite.

The result now follows from Theorem 6 by noting that P has eigenvalues α and $2 - \alpha$. $\square$

Note that since the matrix P doesn't depend on ρ, the proof holds even when the step size changes at each iteration.

6. Lower Bounds

In this section, we probe the tightness of the upper bounds on the convergence rate of Algorithm 2 given by Theorem 6. The construction of the lower bound in this section is similar to a construction given in Ghadimi et al. (2015).

Let Q be a d-dimensional symmetric positive-definite matrix whose largest and smallest eigenvalues are L and m respectively. Let $f(x) = \frac{1}{2}x^\top Qx$ be a quadratic and let $g(z) = \frac{\delta}{2}\|z\|^2$ for some $\delta \geq 0$. Let $A = I_d$, $B = -I_d$, and $c = 0$. With these definitions, the optimization problem in (1) is solved by $x = z = 0$. The updates for Algorithm 2 are given by

$$x_{k+1} = \rho(Q + \rho I)^{-1}(z_k - u_k) \tag{13a}$$

$$z_{k+1} = \frac{\rho}{\delta + \rho}(\alpha x_{k+1} + (1-\alpha)z_k + u_k) \tag{13b}$$

$$u_{k+1} = u_k + \alpha x_{k+1} + (1-\alpha)z_k - z_{k+1}. \tag{13c}$$

Solving for z_k in (13b) and substituting the result into (13c) gives $u_{k+1} = \frac{\delta}{\rho}z_{k+1}$. Then eliminating x_{k+1} and u_k from (13b) using (13a) and the fact that $u_k = \frac{\delta}{\rho}z_k$ allows us to express the update rule purely in terms of z as

$$z_{k+1} = \underbrace{\left(\frac{\alpha\rho(\rho-\delta)}{\rho+\delta}(Q+\rho I)^{-1} + \frac{\rho - \alpha\rho + \delta}{\rho+\delta}I\right)}_{T} z_k.$$

Note that the eigenvalues of T are given by

$$1 - \frac{\alpha\rho(\lambda+\delta)}{(\rho+\delta)(\lambda+\rho)}, \tag{14}$$

where λ is an eigenvalue of Q. We will use this setup to construct a lower bound on the worst-case convergence rate of Algorithm 2 in Theorem 8.

Theorem 8. *Suppose that Assumption 3 holds. The worst-case convergence rate of Algorithm 2, when run with step size $\rho = (\hat{m}\hat{L})^{\frac{1}{2}}\kappa^\varepsilon$ and over-relaxation parameter α, is lower-bounded by*

$$1 - \frac{2\alpha}{1 + \kappa^{0.5+|\varepsilon|}}. \tag{15}$$

Proof. First consider the case $\varepsilon \geq 0$. Choosing $\delta = 0$ and $\lambda = m$, from (14), we see that T has eigenvalue

$$1 - \frac{\alpha}{1 + \kappa^{0.5+\varepsilon}}. \tag{16}$$

When initialized with z as the eigenvector corresponding to this eigenvalue, Algorithm 2 will converge linearly with rate given exactly by (16), which is lower bounded by the expression in (15) when $\varepsilon \geq 0$.

Now suppose that $\varepsilon < 0$. Choosing $\delta = L$ and $\lambda = L$, after multiplying the numerator and denominator of (14) by $\kappa^{0.5-\varepsilon}$, we see that T has eigenvalue

$$1 - \frac{2\alpha}{(1+\kappa^{0.5-\varepsilon})(\kappa^{-0.5+\varepsilon}+1)} \geq 1 - \frac{2\alpha}{1+\kappa^{0.5-\varepsilon}}. \tag{17}$$

When initialized with z as the eigenvector corresponding to this eigenvalue, Algorithm 2 will converge linearly with rate given exactly by the left hand side of (17), which is lower bounded by the expression in (15) when $\varepsilon < 0$. □

Figure 3 compares the lower bounds given by (16) with the upper bounds given by Theorem 6 for $\alpha = 1.5$ and for several choices of $\rho = (\hat{m}\hat{L})^{\frac{1}{2}}\kappa^\varepsilon$ satisfying $\varepsilon \geq 0$. The upper and lower bounds agree visually on the range of choices ε depicted, demonstrating the practical tightness of the upper bounds given by Theorem 6 for a large range of choices of parameter values.

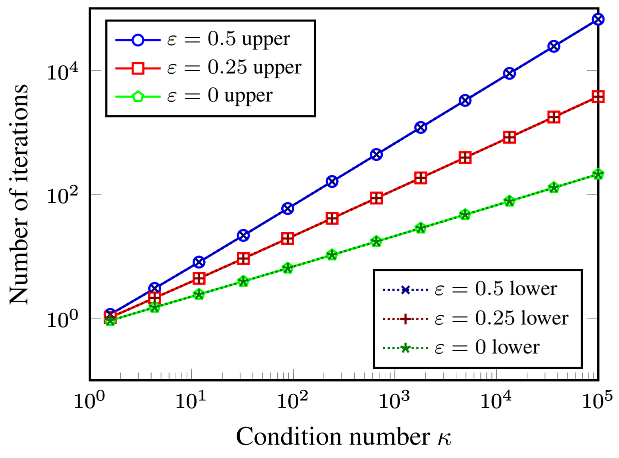

Figure 3. For $\alpha = 1.5$ and for several choices ε in $\rho_0 = \kappa^\varepsilon$, we plot $-1/\log\tau$ as a function of κ, both for the lower bound on τ given by (16) and the upper bound on τ given by Theorem 6. For each choice of ε in $\{0.5, 0.25, 0\}$, the lower and upper bounds agree visually. This agreement demonstrates the practical tightness of the upper bounds given by Theorem 6 for a large range of choices of parameter values.

7. Related Work

Several recent papers have studied the linear convergence of Algorithm 1 but do not extend to Algorithm 2. Deng & Yin (2012) prove a linear rate of convergence for ADMM in the strongly convex case. Iutzeler et al. (2014) prove the linear convergence of a specialization of ADMM to a class of distributed optimization problems under a local strong-convexity condition. Hong & Luo (2012) prove the linear convergence of a generalization of ADMM to a multiterm objective in the setting where each term can be decomposed as a strictly convex function and a polyhedral function. In particular, this result does not require strong convexity.

More generally, there are a number of results for operator splitting methods in the literature. Lions & Mercier (1979) and Eckstein & Ferris (1998) analyze the convergence of several operator splitting schemes. More recently, Patrinos et al. (2014a;b) prove the equivalence of forward-backward splitting and Douglas–Rachford splitting with a scaled version of the gradient method applied to unconstrained nonconvex surrogate functions (called the forward-backward envelope and the Douglas–Rachford envelope respectively). Goldstein et al. (2012) propose an accelerated version of ADMM in the spirit of Nesterov, and prove a $O(1/k^2)$ convergence rate in the case where f and g are both strongly convex and g is quadratic.

The theory of over-relaxed ADMM is more limited. Eckstein & Bertsekas (1992) prove the convergence of over-relaxed ADMM but do not give a rate. More recently, Davis & Yin (2014a;b) analyze the convergence rates of ADMM in a variety of settings. Giselsson & Boyd (2014) prove the linear convergence of Douglas–Rachford splitting in the strongly-convex setting. They use the fact that ADMM is Douglas–Rachford splitting applied to the dual problem (Eckstein & Bertsekas, 1992) to derive a linear convergence rate for over-relaxed ADMM with a specific choice of step size ρ. Eckstein (1994) gives convergence results for several specializations of ADMM, and found that over-relaxation with $\alpha = 1.5$ empirically sped up convergence. Ghadimi et al. (2015) give some guidance on tuning over-relaxed ADMM in the quadratic case.

Unlike prior work, our framework requires no assumptions on the parameter choices in Algorithm 2. For example, Theorem 6 certifies the linear convergence of Algorithm 2 even for values $\alpha > 2$. In our framework, certifying a convergence rate for an arbitrary choice of parameters amounts to checking the feasibility of a 4×4 semidefinite program, which is essentially instantaneous, as opposed to formulating a proof.

8. Selecting Algorithm Parameters

In this section, we show how to use the results of Section 4 to select the parameters α and ρ in Algorithm 2 and we show the effect on a numerical example.

Recall that given a choice of parameters α and ρ and given the condition number κ, Theorem 6 gives an upper bound on the convergence rate of Algorithm 2. Therefore, one approach to parameter selection is to do a grid search over the space of parameters for the choice that minimizes the upper bound provided by Theorem 6. We demonstrate this approach numerically for a distributed Lasso problem, but first we demonstrate that the usual range of $(0, 2)$ for the over-relaxation parameter α is too limited, that more choices of α lead to linear convergence. In Figure 4, we plot the largest value of α found through binary search such that (11) is satisfied for some $\tau < 1$ as a function of κ. Proof techniques in prior work do not extend as easily to values of $\alpha > 2$. In our framework, we simply change some constants in a small semidefinite program.

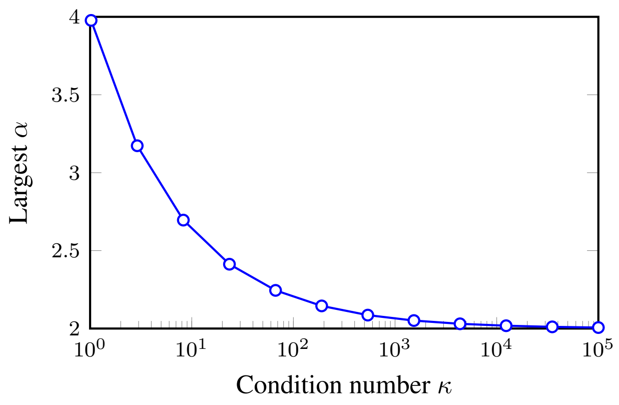

Figure 4. As a function of κ, we plot the largest value of α such that (11) is satisfied for some $\tau < 1$. In this figure, we set $\varepsilon = 0$ in $\rho_0 = \kappa^\varepsilon$.

8.1. Distributed Lasso

Following Deng & Yin (2012), we give a numerical demonstration with a distributed Lasso problem of the form

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{N} \frac{1}{2\mu}\|A_i x_i - b_i\|^2 + \|z\|_1 \\ \text{subject to} & x_i - z = 0 \quad \text{for all} \quad i = 1, \ldots, N. \end{array}$$

Each A_i is a tall matrix with full column rank, and so the first term in the objective will be strongly convex and its gradient will be Lipschitz continuous. As in Deng & Yin (2012), we choose $N = 5$ and $\mu = 0.1$. Each A_i is generated by populating a 600×500 matrix with independent standard normal entries and normalizing the columns. We generate each b_i via $b_i = A_i x^0 + \varepsilon_i$, where x^0 is a sparse 500-dimensional vector with 250 independent standard normal entries, and $\varepsilon_i \sim \mathcal{N}(0, 10^{-3} I)$.

In Figure 5, we compute the upper bounds on the convergence rate given by Theorem 6 for a grid of values of α and ρ. Each line corresponds to a fixed choice of α, and we plot only a subset of the values of α to keep the plot manageable. We omit points corresponding to parameter values for which the linear matrix inequality in (11) was not feasible for any value of $\tau < 1$.

In Figure 6, we run Algorithm 2 for the same values of α

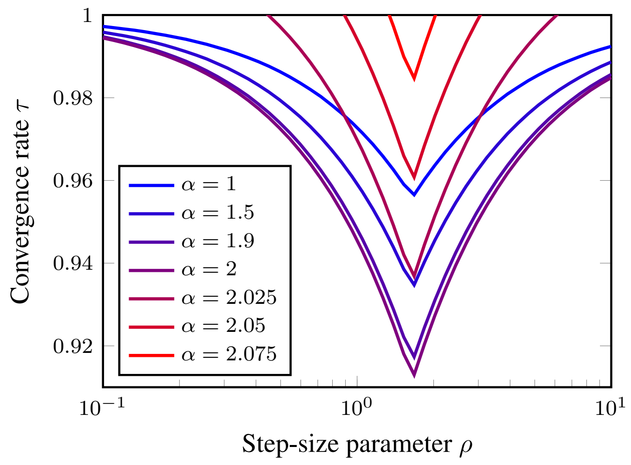

Figure 5. We compute the upper bounds on the convergence rate given by Theorem 6 for eighty-five values of α evenly spaced between 0.1 and 2.2 and fifty values of ρ geometrically spaced between 0.1 and 10. Each line corresponds to a fixed choice of α, and we show only a subset of the values of α to keep the plot manageable. We omit points corresponding to parameter values for which (11) is not feasible for any value of $\tau < 1$. This analysis suggests choosing $\alpha = 2.0$ and $\rho = 1.7$.

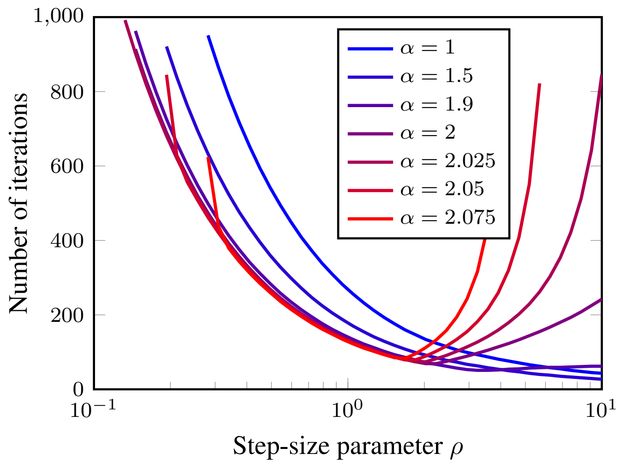

Figure 6. We run Algorithm 2 for eighty-five values of α evenly spaced between 0.1 and 2.2 and fifty value of ρ geometrically spaced between 0.1 and 10. We plot the number of iterations required for z_k to reach within 10^{-6} of a precomputed reference solution. We show only a subset of the values of α to keep the plot manageable. We omit points corresponding to parameter values for which Algorithm 2 exceeded 1000 iterations.

and ρ. We then plot the number of iterations needed for z_k to reach within 10^{-6} of a precomputed reference solution. We plot lines corresponding to only a subset of the values of α to keep the plot manageable, and we omit points corresponding to parameter values for which Algorithm 2 exceeded 1000 iterations. For the most part, the performance of Algorithm 2 as a function of ρ closely tracked the performance predicted by the upper bounds in Figure 5. Notably, smaller values of α seem more robust to poor choices of ρ. The parameters suggested by our analysis perform close to the best of any parameter choices.

9. Discussion

We showed that a framework based on semidefinite programming can be used to prove convergence rates for the alternating direction method of multipliers and allows a unified treatment of the algorithm's many variants, which arise through the introduction of additional parameters. We showed how to use this framework for establishing convergence rates, as in Theorem 6 and Theorem 7, and how to use this framework for parameter selection in practice, as in Section 8. The potential uses are numerous. This framework makes it straightforward to propose new algorithmic variants, for example, by introducing new parameters into Algorithm 2 and using Theorem 6 to see if various settings of these new parameters give rise to improved guarantees.

In the case that Assumption 3 does not hold, the most likely cause is that we lack the strong convexity of f. One approach to handling this is to run Algorithm 2 on the modified function $f(x) + \frac{\delta}{2}\|x\|^2$. By completing the square in the x update, we see that this amounts to an extremely minor algorithmic modification (it only affects the x update).

It should be clear that other operator splitting methods such as Douglas–Rachford splitting and forward-backward splitting can be cast in this framework and analyzed using the tools presented here.

Acknowledgments

This research is supported in part by NSF CISE Expeditions award CCF-1139158, LBNL award 7076018, DARPA XData award FA8750-12-2-0331, AFOSR award FA9550-12-1-0339, NASA grant NRA NNX12AM55A, ONR grants N00014-11-1-0688 and N00014-14-1-0024, US ARL and US ARO grant W911NF-11-1-0391, NSF awards CCF-1359814 and CCF-1217058, NSF grant DGE-1106400, a Sloan Research Fellowship, and gifts from Amazon Web Services, Google, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple, Blue Goji, Bosch, C3Energy, Cisco, Cray, Cloudera, EMC, Ericsson, Facebook, Guavus, Huawei, Informatica, Intel, Microsoft, NetApp, Pivotal, Samsung, Splunk, Virdata, VMware, and Yahoo!.

References

Aslan, Ö., Cheng, H., Zhang, X., and Schuurmans, D. Convex two-layer modeling. In *Advances in Neural Informa-*

tion Processing Systems, pp. 2985–2993, 2013.

Behmardi, B., Archambeau, C., and Bouchard, G. Overlapping trace norms in multi-view learning. *arXiv preprint arXiv:1404.6163*, 2014.

Bioucas-Dias, J. M. and Figueiredo, M. A. T. Alternating direction algorithms for constrained sparse regression: application to hyperspectral unmixing. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pp. 1–4. IEEE, 2010.

Bird, S. *Optimizing Resource Allocations for Dynamic Interactive Applications*. PhD thesis, University of California, Berkeley, 2014.

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

Corless, M. Guaranteed rates of exponential convergence for uncertain systems. *Journal of Optimization Theory and Applications*, 64(3):481–494, 1990.

D'Alto, L. and Corless, M. Incremental quadratic stability. *Numerical Algebra, Control and Optimization*, 3(1): 175–201, 2013.

Davis, D. and Yin, W. Convergence rate analysis of several splitting schemes. *arXiv preprint arXiv:1406.4834*, 2014a.

Davis, D. and Yin, W. Convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions. *arXiv preprint arXiv:1407.5210*, 2014b.

Deng, W. and Yin, W. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical report, DTIC Document, 2012.

Eckstein, J. Parallel alternating direction multiplier decomposition of convex programs. *Journal of Optimization Theory and Applications*, 80(1):39–62, 1994.

Eckstein, J. and Bertsekas, D. P. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

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

Forero, P. A., Cano, A., and Giannakis, G. B. Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 11:1663–1707, 2010.

Forouzan, S. and Ihler, A. Linear approximation to ADMM for MAP inference. In *Asian Conference on Machine Learning*, pp. 48–61, 2013.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3): 644–658, 2015.

Giselsson, P. and Boyd, S. Diagonal scaling in Douglas–Rachford splitting and ADMM. In *IEEE Conference on Decision and Control*, pp. 5033–5039, 2014.

Glowinski, R. and Marroco, A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 9(R2):41–76, 1975.

Goldstein, T., O'Donoghue, B., and Setzer, S. Fast alternating direction optimization methods. *CAM report*, pp. 12–35, 2012.

Hong, M. and Luo, Z.-Q. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012.

Iutzeler, F., Bianchi, P., Ciblat, Ph., and Hachem, W. Linear convergence rate for distributed optimization with the alternating direction method of multipliers. In *IEEE Conference on Decision and Control*, pp. 5046–5051, 2014.

Lessard, L., Recht, B., and Packard, A. Analysis and design of optimization algorithms via integral quadratic constraints. *arXiv preprint arXiv:1408.3595*, 2014.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014.

Lions, P.-L. and Mercier, B. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

Meshi, O. and Globerson, A. An alternating direction method for dual MAP LP relaxation. In *Machine Learning and Knowledge Discovery in Databases*, pp. 470–483. Springer, 2011.

Patrinos, P., Stella, L., and Bemporad, A. Douglas–Rachford splitting: complexity estimates and accelerated variants. In *IEEE Conference on Decision and Control*, pp. 4234–4239, 2014a.

Patrinos, P., Stella, L., and Bemporad, A. Forward-backward truncated Newton methods for large-scale convex composite optimization. *arXiv preprint arXiv:1402.6655*, 2014b.

Romera-Paredes, B. and Pontil, M. A new convex relaxation for tensor completion. In *Advances in Neural Information Processing Systems*, pp. 2967–2975, 2013.

Sedghi, H., Anandkumar, A., and Jonckheere, E. Multi-step stochastic ADMM in high dimensions: Applications to sparse optimization and matrix decomposition. In *Advances in Neural Information Processing Systems*, pp. 2771–2779, 2014.

Wahlberg, B., Boyd, S., Annergren, M., and Wang, Y. An ADMM algorithm for a class of total variation regularized estimation problems. In *IFAC Symposium on System Identification*, pp. 83–88, 2012.

Wang, H. and Banerjee, A. Online alternating direction method. In *International Conference on Machine Learning*, pp. 1119–1126, 2012.

Wang, H., Banerjee, A., Hsieh, C.-J., Ravikumar, P. K., and Dhillon, I. S. Large scale distributed sparse precision estimation. In *Advances in Neural Information Processing Systems*, pp. 584–592, 2013.

Zhang, C., Lee, H., and Shin, K. G. Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In *International Conference on Artificial Intelligence and Statistics*, pp. 1398–1406, 2012.

Zhang, R. and Kwok, J. Asynchronous distributed ADMM for consensus optimization. In *International Conference on Machine Learning*, pp. 1701–1709, 2014.