

A Flexible ADMM Algorithm for Big Data Applications

Daniel P. Robinson*

Rachael Tappenden†

Abstract

We present a flexible Alternating Direction Method of Multipliers (F-ADMM) algorithm for solving optimization problems involving a strongly convex objective function that is separable into $n \geq 2$ blocks, subject to (non-separable) linear equality constraints. The F-ADMM algorithm uses a *Gauss-Seidel* scheme to update blocks of variables, and a regularization term is added to each of the subproblems arising within F-ADMM. We prove, under common assumptions, that F-ADMM is globally convergent.

We also present a special case of F-ADMM that is *partially parallelizable*, which makes it attractive in a big data setting. In particular, we partition the data into groups, so that each group consists of multiple blocks of variables. By applying F-ADMM to this partitioning of the data, and using a specific regularization matrix, we obtain a hybrid ADMM (H-ADMM) algorithm: the grouped data is updated in a Gauss-Seidel fashion, and the blocks within each group are updated in a Jacobi manner. Convergence of H-ADMM follows directly from the convergence properties of F-ADMM. Also, a special case of H-ADMM can be applied to functions that are convex, rather than strongly convex. We present numerical experiments to demonstrate the practical advantages of this algorithm.

Keywords: Alternating Direction Method of Multipliers; convex optimization; Gauss-Seidel; Jacobi; regularization; separable function;

AMS Classification: 49M15; 49M37; 58C15; 65K05; 65K10; 65Y20; 68Q25; 90C30; 90C60

1 Introduction

In this work we study the optimization problem

$$\underset{x_1, \dots, x_n}{\text{minimize}} \quad \sum_{i=1}^n f_i(x_i) \tag{1a}$$

$$\text{subject to} \quad \sum_{i=1}^n A_i x_i = b, \tag{1b}$$

where, for each $i = 1, \dots, n$, the function $f_i : \mathbf{R}^{N_i} \rightarrow \mathbf{R} \cup \{\infty\}$ is strongly convex, closed, and extended real valued, and the vector $b \in \mathbf{R}^m$ and matrix $A_i \in \mathbf{R}^{m \times N_i}$ represent problem data. Note that the objective function (1a) is separable in the decision vectors $x_1, \dots, x_n$, but that the linear constraint (1b) links them together, which makes problem (1) non-separable overall.

We can think of the decision vectors $\{x_i\}$ as “blocks” of a single decision vector $x \in \mathbf{R}^N$, where $N = \sum_{i=1}^n N_i$. This can be achieved by partitioning the $N \times N$ identity matrix I column-wise into n submatrices $\{U_i \in \mathbf{R}^{N \times N_i}\}_{i=1}^n$, so that $I = [U_1, \dots, U_n]$, and then setting $x = \sum_{i=1}^n U_i x_i$. That is, x is the vector formed by stacking the vectors $\{x_i\}_{i=1}^n$ on top of each other. It is easy to see that $x_i = U_i^T x \in \mathbf{R}^{N_i}$, and that if we

*D. P. Robinson is with the School of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218, USA. e-mail: daniel.p.robinson@jhu.edu. This author has received support from the U.S. National Science Foundation under Grant No. DMS-1217153.

†R. Tappenden (corresponding author) is with the School of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218, USA. e-mail: rtappen1@jhu.edu.

let $A := \sum_{i=1}^n A_i U_i^T \in \mathbf{R}^{m \times N}$, then (1b) is equivalent to $Ax = b$. Note that $A_i = AU_i$ for $i = 1, 2, \dots, n$, and that we can write $A = [A_1, \dots, A_n]$. If we now let $f(x) := \sum_{i=1}^n f_i(x_i)$, problem (1) is equivalent to

$$\underset{x \in \mathbf{R}^N}{\text{minimize}} \quad f(x) \tag{2a}$$

$$\text{subject to} \quad Ax = b. \tag{2b}$$

Although problems (1) and (2) are mathematically equivalent, it is important to note that the best algorithms for solving them take advantage of the block structure that is made explicit in formulation (1).

1.1 Relevant Previous Work

Many popular algorithms for solving (1) (equivalently, for solving (2)) are based on the Augmented Lagrangian function. In the remainder of this section, we describe several algorithms that are closely related to our proposed framework.

The Augmented Lagrangian Method of Multipliers (ALMM)

The ALMM (e.g., see [1]) is based on the augmented Lagrangian function

$$\mathcal{L}_\rho(x; y) := f(x) - \langle y, Ax - b \rangle + \frac{\rho}{2} \|Ax - b\|_2^2, \tag{3}$$

where $\rho > 0$ is called the penalty parameter, $y \in \mathbf{R}^m$ is a dual vector that estimates a Lagrange multiplier vector, and $\langle p, q \rangle = p^T q$ is the standard inner product in $\mathbf{R}^n$. The most basic variant of ALMM (see Algorithm 1), involves two key steps during each iteration. First, for a fixed dual estimate, the augmented Lagrangian (3) is minimized with respect to the primal vector x . Second, using the minimizer computed in the first step, a simple update is made to the dual vector that is equivalent to a dual ascent step for maximizing an associated dual function. In practice, computing the minimizer in the first step is the computational bottleneck. This is especially true for large-scale problems that arise in big data applications, and therefore extensive research has focused on reducing its cost (e.g., decomposition techniques [16, 19, 20]).

Algorithm 1 A basic variant of ALMM for solving problem (2).

- 1: **Initialization:** $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, and penalty parameter $\rho > 0$.
- 2: **while** the stopping condition has not been met **do**
- 3: Update the primal variables by minimizing the augmented Lagrangian:

$$x^{(k+1)} \leftarrow \arg \min_x \mathcal{L}_\rho(x; y^{(k)}) \tag{4}$$

- 4: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \rho(Ax^{(k+1)} - b)$$

- 5: Set $k \leftarrow k + 1$.

- 6: **end while**
-

Although sophisticated variants of ALMM are successfully used in many important application areas (e.g., optimal control in natural gas networks [24]), generally they are unable to directly take advantage of the block separability described in formulation (1), when it exists. Nonetheless, ALMM serves as the basis for many related and powerful methods, as we now discuss.

The Alternating Direction Method of Multipliers (ADMM)

The ADMM has been a widely used algorithm for solving problems of the form (1) when $n = 2$, for convex functions. Global convergence of ADMM was established in the early 1990's by Eckstein and Bertsekas [9]

while studying the algorithm as a particular instance of a Douglas-Rachford splitting method. This relationship allowed them to use monotone operator theory to obtain their global convergence guarantees. (An introduction to ADMM and its convergence theory can be found in the tutorial style paper by Eckstein [8]. See also [3].) Pseudocode for ADMM when $n = 2$ is given below as Algorithm 2.

Algorithm 2 ADMM for solving problem (1) when $n = 2$.

- 1: **Initialization:** $x^{(0)} \in \mathbf{R}^N$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, and penalty parameter $\rho > 0$.
- 2: **while** the stopping condition has not been met **do**
- 3: Update the primal variables in a Gauss-Seidel fashion:

$$x_1^{(k+1)} \leftarrow \arg \min_x \mathcal{L}_\rho(x, x_2^{(k)}; y^{(k)}) \quad (5a)$$

$$x_2^{(k+1)} \leftarrow \arg \min_x \mathcal{L}_\rho(x_1^{(k+1)}, x; y^{(k)}) \quad (5b)$$

- 4: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \rho(Ax^{(k+1)} - b)$$

- 5: Set $k \leftarrow k + 1$.
 - 6: **end while**
-

In words, ADMM works as follows. At iteration k , for a fixed multiplier $y^{(k)}$ and fixed block $x_2^{(k)}$, the new point $x_1^{(k+1)}$ is defined as the minimizer (for simplicity, we assume throughout that this minimizer exists and that it is unique) of the augmented Lagrangian with respect to the first block of variables x_1 . Then, in a similar fashion, the first (updated) block $x_1^{(k+1)}$ is fixed, and the augmented Lagrangian is minimized with respect to the second block of variables x_2 to obtain $x_2^{(k+1)}$. Finally, the dual variables are updated in the same manner as for the basic ALMM (see Algorithm 1), and the process is repeated. Notice that a key feature of ADMM is that the blocks of variables x_1 and x_2 are updated in a *Gauss-Seidel* fashion, i.e., the *updated* values for the first block of variables are used to define the subproblem used to obtain the updated values for the second block of variables. The motivation for the design of ADMM is that each subproblem (see (5a) and (5b)) should be substantially easier to solve than the subproblem (see (4)) used by ALMM. For many important applications, this is indeed the case.

The interest in ADMM has exploded in recent years because of applications in signal and image processing, compressed sensing [22], matrix completion [23], distributed optimization and statistical and machine learning [3], and quadratic and linear programming [2]. Convergence of ADMM has even been studied for specific instances of nonconvex functions, namely consensus and sharing problems [15].

A natural question to ask is whether ADMM converges when there are more than two blocks, i.e., when $n \geq 3$. The authors in [5] show via a counterexample that ADMM is not necessarily convergent if $n = 3$. However, they also show that if $n = 3$ and at least two of the matrices that define the linking constraint (1b) are orthogonal, then ADMM will converge. In a different paper [4], the authors show that ADMM will converge when $n = 3$ if at least one of the functions f_i in (1a) is strongly convex.

Other works have considered the more general case of $n \geq 2$. For example, an ADMM-type algorithm for $n \geq 2$ blocks is introduced in [21], where during each iteration a randomly selected subset of blocks is updated in parallel. The method incorporates a “backward step” on the dual update to ensure convergence. Hong and Luo [14] present a convergence proof for the n block ADMM when the functions are convex, but under many assumptions that are difficult to verify in practice. Work in [11] shows that ADMM is convergent in the n block case when the functions f_i for $i = 1, \dots, n$ are strongly convex. Moreover, the work in [10] introduces an ADMM type method for convex functions of n blocks that incorporates a step of Gaussian back substitution at every iteration to ensure convergence. Finally, the works in [12, 13] propose a splitting version of ADMM, that separates the n blocks into 2 groups. The groups are updated in a Gauss-Seidel fashion, and the blocks within each group are updated in parallel, and the method uses a regularization term with a specific form to ensure convergence.

The Generalized ADMM (G-ADMM)

Deng and Yin [7] introduced G-ADMM, which is a variant of ADMM for solving problems of the form (1) when $n = 2$ and the functions f_i are convex. They proposed the addition of a (general) regularization term to the augmented Lagrangian function during the minimization subproblem within ADMM and the addition of a relaxation parameter γ to the dual variable update. Their motivation for the inclusion of a regularization term was twofold. First, for certain applications, a careful choice of that regularizer lead to subproblems that were significantly easier to solve. Second, the regularization stabilized the iterates, which has theoretical and numerical advantages.

Their method is stated below as Algorithm 3. It uses, for any symmetric positive-definite matrix M and vector z , the ellipsoidal norm

$$\|z\|_M^2 := z^T M z. \quad (6)$$

Algorithm 3 G-ADMM for solving problem (1) when $n = 2$.

- 1: **Initialization:** $x_1^{(0)} \in \mathbf{R}^{N_1}$, $x_2^{(0)} \in \mathbf{R}^{N_2}$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, parameters $\rho > 0$ and $\gamma \in (0, 2)$, and regularization matrices $P_1 \in \mathbf{R}^{N_1 \times N_1}$ and $P_2 \in \mathbf{R}^{N_2 \times N_2}$.
- 2: **while** the stopping condition has not been met **do**
- 3: Update the primal variables in a Gauss-Seidel fashion:

$$x_1^{(k+1)} \leftarrow \arg \min_x \mathcal{L}_\rho(x, x_2^{(k)}; y^{(k)}) + \frac{1}{2} \|x - x_1^{(k)}\|_{P_1}^2 \quad (7a)$$

$$x_2^{(k+1)} \leftarrow \arg \min_x \mathcal{L}_\rho(x_1^{(k+1)}, x; y^{(k)}) + \frac{1}{2} \|x - x_2^{(k)}\|_{P_2}^2 \quad (7b)$$

- 4: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \gamma \rho (Ax^{(k+1)} - b)$$

- 5: Set $k \leftarrow k + 1$.
 - 6: **end while**
-

The authors prove [7] that Algorithm 3 converges to a solution from an arbitrary starting point as long as the regularization matrices P_1 and P_2 in (7a) and (7b) satisfy certain properties. We stress that the convergence analysis for G-ADMM only applies to the $n = 2$ case.

The Jacobi ADMM (J-ADMM)

Deng et al. [6] have extended the ideas first presented in G-ADMM [7]. Their new J-ADMM strategy (stated below as Algorithm 4) may be used to solve problem (1) in the general case of $n \geq 2$ blocks. Note that (8a) is equivalent to the update

$$x_i^{(k+1)} \leftarrow \arg \min_{x_i} \mathcal{L}_\rho(x_1^{(k)}, \dots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k)}, \dots, x_n^{(k)}; y^{(k)}) + \frac{1}{2} \|x_i - x_i^{(k)}\|_{P_i}^2,$$

(where $P_i \in \mathbf{R}^{N_i \times N_i}$ is a regularization matrix) which we state in order to highlight the relationship of their method to the previous ones. We also comment that the form of the update used in (8a) motivates why their algorithm is of the proximal type.

Algorithm 4 J-ADMM for solving problem (1) for $n \geq 2$.

- 1: **Initialize:** $x^{(0)} \in \mathbf{R}^N$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, parameters $\rho > 0$ and $\gamma \in (0, 2)$, and regularization matrices $P_i \in \mathbf{R}^{N_i \times N_i}$ for $i = 1, \dots, n$.
- 2: **while** stopping condition has not been met **do**
- 3: **for** $i = 1, \dots, n$ (in parallel) **do**

$$x_i^{(k+1)} \leftarrow \arg \min_{x_i} \left\{ f_i(x_i) + \frac{\rho}{2} \|A_i x_i + \sum_{j \neq i}^n A_j x_j^{(k)} - b - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_i - x_i^{(k)}\|_{P_i}^2 \right\} \quad (8a)$$

- 4: **end for**

- 5: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \gamma \rho (Ax^{(k+1)} - b) \quad (8b)$$

- 6: Set $k \leftarrow k + 1$.

- 7: **end while**

In [6], the authors establish global convergence of J-ADMM for appropriately chosen regularization matrices P_i . Moreover, they showed that J-ADMM has a convergence rate of $\mathcal{O}(1/k)$.

1.2 Our Main Contributions

We now summarize the main contributions of this work.

1. We present a new flexible ADMM algorithm, called F-ADMM, that solves problems of the form (1) for strongly convex f_i , for general $n \geq 2$ based on a *Gauss-Seidel* updating scheme. The quadratic regularizer used in F-ADMM is a user defined matrix that must be sufficiently positive definite (see Assumption 4), which makes F-ADMM flexible. For some applications, a careful choice of the regularizer makes the subproblems arising within F-ADMM significantly easier to solve, e.g., see the discussion in [7, Section 1.2] and [6, Section 1.2]. We prove that F-ADMM is *globally convergent* in Section 2.
2. We introduce a hybrid Jacobi/Gauss-Seidel variant of F-ADMM, called H-ADMM, that is *partially parallelizable*. This is significant because it makes H-ADMM competitive in a *big data setting*. For H-ADMM, the blocks of variables are gathered into multiple groups, with a Gauss-Seidel updating scheme between groups, and a Jacobi updating scheme on the individual blocks within each group. We demonstrate that H-ADMM is simply F-ADMM with a particular choice of regularization matrix, and thus the convergence of H-ADMM follows directly from the convergence proof for F-ADMM.
3. We show that if the n blocks of data are partitioned into two groups, then H-ADMM can be applied to convex functions f_i , rather than strongly convex functions. In this special case, with carefully chosen regularization matrices, H-ADMM extends the algorithm in [7] from the $n = 2$ case, to the case with general n , and convergence follows directly from the results presented in [7].

1.3 Paper Outline

In Section 2 we present our new flexible ADMM framework and show that any instance of it is globally convergent. In Section 3 we consider a particular instance of our general framework, and proceed to show that it is a hybrid of Jacobi- and Gauss-Seidel-type updates. We also discuss the practical advantages of this hybrid algorithm, which includes the fact that it is *partially parallelizable*. Finally, in Section 4 we present numerical experiments that illustrate the advantages of our flexible ADMM framework.

2 A Flexible ADMM (F-ADMM)

In this section we present and analyze a new F-ADMM framework for solving problems of the form (1). For convenience, we define the vector

$$u^{(k)} := \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix}. \quad (9)$$

Our analysis requires several assumptions concerning problem (1) that are assumed to hold throughout. The first of which uses $\partial f(x)$ to denote the subdifferential of f at the point x , i.e.,

$$\partial f(x) := \{s \in \mathbf{R}^N \mid \langle s, w - x \rangle \leq f(w) - f(x), \forall w \in \text{dom} f\}, \quad (10)$$

where $\text{dom} f = \{x : f(x) < \infty\}$. Moreover,

$$\partial f_i(x_i) := \{s_i \in \mathbf{R}^{N_i} \mid \langle s_i, w_i - x_i \rangle \leq f_i(w_i) - f_i(x_i), \forall w_i \in \text{dom} f_i\}. \quad (11)$$

We also require the following definition of strong convexity. A function $f_i : \mathbf{R}^N \rightarrow \mathbf{R} \cup \{+\infty\}$ is strongly convex with convexity parameter $\mu_i > 0$ if for all $x_i, w_i \in \text{dom} f_i$,

$$f_i(w_i) \geq f_i(x_i) + \langle \partial f_i(x_i), w_i - x_i \rangle + \frac{\mu_i}{2} \|w_i - x_i\|_2^2. \quad (12)$$

We may now state our assumptions on problem (1).

Assumption 1. *The set of saddle points (equivalently, the set of KKT-points) for (1) is nonempty, i.e.,*

$$U^* := \{u^* \in \mathbf{R}^{N+m} : u^* = (x^*, y^*), A^T y^* \in \partial f(x^*), \text{ and } Ax^* - b = 0\} \neq \emptyset.$$

Assumption 2. *The function f_i is strongly convex with strong convexity constant $\mu_i > 0$ for $i = 1, \dots, n$.*

If Assumption 1 does not hold, then ADMM may have unsolvable or unbounded subproblems, or the sequence of Lagrange multiplier estimates may diverge. In particular, x^* is the solution to (1) and y^* is a solution to the associated dual problem. Assumption 2 allows us to define

$$\mu := \min_{1 \leq i \leq n} \mu_i > 0 \quad (13)$$

as the minimum strong convexity parameter for the functions $\{f_i\}_{i=1}^n$, as well as use the following lemma.

Lemma 3 (Strong monotonicity of the subdifferential, Theorem 12.53 and Exercise 12.59 in [18]). *Under Assumption 2, for any $x_i, w_i \in \text{dom} f_i$ we have*

$$\langle s_i - t_i, x_i - w_i \rangle \geq \mu_i \|x_i - w_i\|_2^2, \quad \forall s_i \in \partial f_i(x_i), t_i \in \partial f_i(w_i), \quad i = 1, \dots, n. \quad (14)$$

The following matrices will be important for defining the regularization matrices used in our algorithm, and will also be used in our convergence proof. In particular, we define the block diagonal matrix A_D , and the strictly upper triangular matrix A_Δ as

$$A_\Delta := \begin{bmatrix} A_2 & \dots & A_n \\ & \ddots & \vdots \\ & & A_n \end{bmatrix} \quad \text{and} \quad A_D := \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_n \end{bmatrix}, \quad (15)$$

where $\{A_\Delta, A_D\} \subset \mathbf{R}^{mn \times N}$. We then have the strictly (block) upper triangular matrix

$$A_D^T A_\Delta = \begin{bmatrix} A_1^T A_2 & \dots & A_1^T A_n \\ & \ddots & \vdots \\ & & A_{n-1}^T A_n \end{bmatrix} \in \mathbf{R}^{N \times N}. \quad (16)$$

Notice that $A_D^T A_\Delta$ is equivalent to $\text{triu}^+(A^T A)$, where $\text{triu}^+(X)$ denotes the strictly upper (block) triangular part of X . We are now in a position to describe the details of our F-ADMM method.

2.1 The Algorithm

Our F-ADMM method is stated formally as Algorithm 5. As for J-ADMM, F-ADMM requires the choice of a penalty parameter $\rho > 0$ and regularization matrices $\{P_i\}_{i=1}^n$. Our convergence analysis considered in Section 2.2 requires them to satisfy the following assumption that uses the definition of μ in (13).

Assumption 4. *The matrices P_i are symmetric and satisfy $P_i \succ \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 I$ for all $i = 1, \dots, n$.*

Algorithm 5 F-ADMM for solving problem (1).

- 1: **Initialize:** $x^{(0)} \in \mathbf{R}^N$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, parameters $\rho > 0$, $\gamma \in (0, 2)$, and matrices $\{P_i\}_{i=1}^n$ satisfying Assumption 4.
- 2: **while** stopping condition has not been met **do**
- 3: Update the primal variables in a Gauss-Seidel fashion:

$$\begin{aligned}
x_1^{(k+1)} &\leftarrow \arg \min_{x_1} \left\{ f_1(x_1) + \frac{\rho}{2} \|A_1 x_1 + \sum_{j=2}^n A_j x_j^{(k)} - b - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_1 - x_1^{(k)}\|_{P_1}^2 \right\} \\
&\vdots \\
x_i^{(k+1)} &\leftarrow \arg \min_{x_i} \left\{ f_i(x_i) + \frac{\rho}{2} \|A_i x_i + \sum_{j=1}^{i-1} A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - b - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_i - x_i^{(k)}\|_{P_i}^2 \right\} \\
&\vdots \\
x_n^{(k+1)} &\leftarrow \arg \min_{x_n} \left\{ f_n(x_n) + \frac{\rho}{2} \|A_n x_n + \sum_{j=1}^{n-1} A_j x_j^{(k+1)} - b - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_n - x_n^{(k)}\|_{P_n}^2 \right\}
\end{aligned}$$

- 4: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \gamma \rho (A x^{(k+1)} - b) \quad (17)$$

- 5: Set $k \leftarrow k + 1$.
 - 6: **end while**
-

We now describe the k th iteration of Algorithm 5 in more detail. For fixed dual vector $y^{(k)}$, the current point $x^{(k)}$ is updated in a Gauss-Seidel (i.e., a cyclic block-wise) fashion. To begin, decision vectors $x_2^{(k)}, \dots, x_n^{(k)}$ are fixed, and the first subproblem in Step 3 is minimized with respect to x_1 to give the new point $x^{(k+1)}$. Similar to before, we note that the i th subproblem in Step 3 is equivalent to

$$x_i^{(k+1)} \leftarrow \arg \min_{x_i} \mathcal{L}_\rho(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i, x_{i+1}^{(k)}, \dots, x_n^{(k)}; y^{(k)}) + \frac{1}{2} \|x_i - x_i^{(k)}\|_{P_i}^2. \quad (18)$$

Next, the second block x_2 is updated using the information obtained in the update of the first block x_1 . That is, the vectors $x_3^{(k)}, \dots, x_n^{(k)}$ remain fixed, as does $x_1^{(k+1)}$, and the regularized augmented Lagrangian is minimized with respect to x_2 to give the new point $x_2^{(k+1)}$. The process is repeated until all n blocks have been updated, giving the vector $x^{(k+1)}$. Finally, the dual vector $y^{(k)}$ is updated using the same formula as in J-ADMM (see Algorithm 4). Steps 3 and 4 are repeated until a stopping threshold has been reached.

Remark 5. *It is clear that Algorithm 5 uses a (serial) cyclic block coordinate descent (CD) type method to update the primal vector x . That is, in Step 3 of Algorithm 5, a single pass of block CD is applied to the current point $x^{(k)}$ to give the new point $x^{(k+1)}$, and then the dual vector is updated.*

2.2 Convergence

To analyze F-ADMM, we require the block diagonal matrices G_x and G defined as

$$G_x := \begin{bmatrix} P_1 & & \\ & \ddots & \\ & & P_n \end{bmatrix} \quad \text{and} \quad G := \begin{bmatrix} G_x & \\ & \frac{1}{\gamma\rho}I \end{bmatrix}, \quad (19)$$

where I is the (appropriately sized) identity matrix, and $\gamma \in (0, 2)$ and $\rho > 0$ are algorithm parameters. The following result gives sufficient conditions for declaring that a limit point of problem (1) is optimal.

Lemma 6. *If $\mathcal{K}$ is any subsequence of the natural numbers satisfying*

$$\lim_{k \in \mathcal{K}} u^{(k)} = u^L \quad \text{and} \quad \lim_{k \in \mathcal{K}} \|u^{(k)} - u^{(k+1)}\|_G = 0 \quad (20)$$

for some limit point u^L , then $u^L \in U^*$, i.e., u^L solves problem (1).

Proof. Let us first observe that the two limits in (20) jointly imply that

$$\lim_{k \in \mathcal{K}} u^{(k+1)} = u^L \equiv \begin{pmatrix} x^L \\ y^L \end{pmatrix}. \quad (21)$$

Also, it follows from (20) and the definitions of $u^{(k)}$ (see (9)) and G (see (19)), that $\lim_{k \in \mathcal{K}} (y^{(k)} - y^{(k+1)}) = 0$. Combining this with (17), (21), and (20) establishes that

$$b = \lim_{k \in \mathcal{K}} Ax^{(k+1)} = Ax^L = \lim_{k \in \mathcal{K}} Ax^{(k)} \quad (22)$$

so that, in particular, x^L is feasible for problem (1).

Next, the optimality condition for the i th subproblem in Step 3 of Algorithm 5 ensures the existence of a vector $g_i(x_i^{(k+1)}) \in \partial f_i(x_i^{(k+1)})$ satisfying

$$\begin{aligned} 0 &= g_i(x_i^{(k+1)}) + \rho A_i^T \left(A_i x_i^{(k+1)} + \sum_{j=1}^{i-1} A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - b - \frac{y^{(k)}}{\rho} \right) + P_i(x_i^{(k+1)} - x_i^{(k)}) \\ &= g_i(x_i^{(k+1)}) - A_i^T y^{(k)} + \rho A_i^T \left(A_i x_i^{(k+1)} + \sum_{j=1}^{i-1} A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - b \right) + P_i(x_i^{(k+1)} - x_i^{(k)}) \\ &= g_i(x_i^{(k+1)}) - A_i^T y^{(k)} + \rho A_i^T \left(\sum_{j=1}^i A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - Ax^L \right) + P_i(x_i^{(k+1)} - x_i^{(k)}), \end{aligned}$$

where we also used (22) to substitute for b in the last equation. Using $Ax^L = \sum_{j=1}^n A_j x_j^L$ and rearranging the previous equation gives

$$g_i(x_i^{(k+1)}) = A_i^T y^{(k)} - \rho A_i^T \left(\sum_{j=1}^i A_j (x_j^{(k+1)} - x_j^L) + \sum_{l=i+1}^n A_l (x_l^{(k)} - x_l^L) \right) - P_i(x_i^{(k+1)} - x_i^{(k)}).$$

By taking limits over the subsequence $\mathcal{K}$ of the previous equation, and using (20) and (21), we know that

$$\lim_{k \in \mathcal{K}} g_i(x_i^{(k+1)}) = A_i^T y^L. \quad (23)$$

We may then use $g_i(x_i^{(k+1)}) \in \partial f_i(x_i^{(k+1)})$, (21), (23), and [17, Theorem 24.4] to conclude that

$$A_i^T y^L \in \partial f_i(x_i^L).$$

Combining this inclusion, which holds for all $1 \leq i \leq n$, with (22) shows that u^L is a KKT point for problem (1), and thus is a solution as claimed. $\square$

Our aim is to combine Lemma 6 with the next result, which shows that the sequence $\{\|u_k - u^*\|_G\}$ is nonexpansive with respect to any $u^* \in U^*$. We note that the proof is inspired by that for J-ADMM [6].

Theorem 7. *Let Assumptions 1, 2, and 4 hold. Then, for any $u^* \in U^*$ and all $k \geq 1$, there exists a constant $\eta > 0$ such that*

$$\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 \geq \eta \|u^{(k)} - u^{(k+1)}\|_G^2 \quad (24)$$

with $u^{(k)}$ defined in (9) and G defined in (19).

Proof. At each iteration of Algorithm 5, a subproblem of the following form is solved for x_i :

$$x_i^{(k+1)} = \arg \min_{x_i} \left\{ f_i(x_i) + \frac{\rho}{2} \|A_i x_i + \sum_{j=1}^{i-1} A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - b - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_i - x_i^{(k)}\|_{P_i} \right\}. \quad (25)$$

The first order optimality condition for (25) is

$$0 \in \partial f_i(x_i^{(k+1)}) + \rho A_i^T \left(\sum_{j=1}^i A_j x_j^{(k+1)} + \sum_{l=i+1}^n A_l x_l^{(k)} - b - \frac{y^{(k)}}{\rho} \right) + P_i(x_i^{(k+1)} - x_i^{(k)}),$$

and rearranging gives

$$\rho A_i^T \left(b + \frac{y^{(k)}}{\rho} - \sum_{j=1}^i A_j x_j^{(k+1)} - \sum_{l=i+1}^n A_l x_l^{(k)} \right) + P_i(x_i^{(k)} - x_i^{(k+1)}) \in \partial f_i(x_i^{(k+1)}).$$

Noting that $\sum_{l=i+1}^n A_l x_l^{(k+1)} - A x^{(k+1)} = -\sum_{j=1}^i A_j x_j^{(k+1)}$ and defining $\hat{y} := y^{(k)} - \rho(Ax^{(k+1)} - b)$ gives

$$A_i^T \hat{y} - \rho A_i^T \left(\sum_{j=i+1}^n A_j (x_j^{(k)} - x_j^{(k+1)}) \right) + P_i(x_i^{(k)} - x_i^{(k+1)}) \in \partial f_i(x_i^{(k+1)}).$$

Using Lemma 3 with $A_i^T y^* \in \partial f_i(x_i^*)$, we have

$$\mu_i \|x_i^{(k+1)} - x_i^*\|_2^2 \leq \left\langle x_i^{(k+1)} - x_i^*, A_i^T (\hat{y} - y^*) - \rho A_i^T \sum_{j=i+1}^n A_j (x_j^{(k)} - x_j^{(k+1)}) + P_i(x_i^{(k)} - x_i^{(k+1)}) \right\rangle.$$

Now, summing the previous inequality over all blocks i gives

$$\begin{aligned} \mu \|x^{(k+1)} - x^*\|_2^2 &\leq \left\langle A(x^{(k+1)} - x^*), \hat{y} - y^* \right\rangle - \rho \sum_{i=1}^n \left\langle A_i(x_i^{(k+1)} - x_i^*), \sum_{j=i+1}^n A_j(x_j^{(k)} - x_j^{(k+1)}) \right\rangle \\ &\quad + \sum_{i=1}^n (x_i^{(k+1)} - x_i^*)^T P_i(x_i^{(k)} - x_i^{(k+1)}), \end{aligned} \quad (26)$$

where $\mu > 0$ is defined in (13). Notice that, by (17) the following relation holds

$$A(x^{(k+1)} - x^*) = \frac{1}{\gamma \rho} (y^{(k)} - y^{(k+1)}), \quad (27)$$

and we also have that

$$\hat{y} - y^* = (\hat{y} - y^{(k+1)}) + (y^{(k+1)} - y^*) = \frac{\gamma - 1}{\gamma} (y^{(k)} - y^{(k+1)}) + (y^{(k+1)} - y^*). \quad (28)$$

Then (26) becomes

$$\begin{aligned}
& \sum_{i=1}^n (x_i^{(k+1)} - x_i^*)^T P_i (x_i^{(k)} - x_i^{(k+1)}) - \rho \sum_{i=1}^n \left\langle A_i (x_i^{(k+1)} - x_i^*), \sum_{j=i+1}^n A_j (x_j^{(k)} - x_j^{(k+1)}) \right\rangle \\
& \stackrel{(27)}{\geq} \mu \|x^{(k+1)} - x^*\|_2^2 - \frac{1}{\gamma\rho} \langle y^{(k)} - y^{(k+1)}, \hat{y} - y^* \rangle \\
& \stackrel{(28)}{=} \mu \|x^{(k+1)} - x^*\|_2^2 - \frac{1}{\gamma\rho} \langle y^{(k)} - y^{(k+1)}, y^{(k+1)} - y^* \rangle + \frac{1-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2.
\end{aligned} \tag{29}$$

Using the identity

$$\sum_{i=1}^n \left\langle A_i (x_i^{(k+1)} - x_i^*), \sum_{j=i+1}^n A_j (x_j^{(k)} - x_j^{(k+1)}) \right\rangle \stackrel{(16)}{=} \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle, \tag{30}$$

we may deduce from (29) and the definition of G_x that

$$\begin{aligned}
& \sum_{i=1}^n (x_i^{(k+1)} - x_i^*)^T P_i (x_i^{(k)} - x_i^{(k+1)}) \\
& = \left\langle x^{(k+1)} - x^*, G_x (x^{(k)} - x^{(k+1)}) \right\rangle \\
& \geq \mu \|x^{(k+1)} - x^*\|_2^2 - \frac{1}{\gamma\rho} \langle y^{(k)} - y^{(k+1)}, y^{(k+1)} - y^* \rangle \\
& \quad + \frac{1-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \rho \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle.
\end{aligned} \tag{31}$$

Then, by rearranging (31) we have

$$\begin{aligned}
(u^{(k+1)} - u^*) G (u^{(k)} - u^{(k+1)}) & \geq \mu \|x^{(k+1)} - x^*\|_2^2 + \frac{1-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 \\
& \quad + \rho \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle,
\end{aligned} \tag{32}$$

where G is defined in (19). Combining the relation

$$\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 = 2(u^{(k+1)} - u^*) G (u^{(k)} - u^{(k+1)}) + \|u^{(k)} - u^{(k+1)}\|_G^2$$

with (32) gives

$$\begin{aligned}
\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 & \geq 2\mu \|x^{(k+1)} - x^*\|_2^2 + 2\frac{1-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \|u^{(k)} - u^{(k+1)}\|_G^2 \\
& \quad + 2\rho \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle \\
& = 2\mu \|x^{(k+1)} - x^*\|_2^2 + \frac{2-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \|x^{(k)} - x^{(k+1)}\|_{G_x}^2 \\
& \quad + 2\rho \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle.
\end{aligned} \tag{33}$$

Notice that, because $\mu > 0$, the following holds:

$$2\rho \left\langle x^{(k+1)} - x^*, A_D^T A_\Delta (x^{(k)} - x^{(k+1)}) \right\rangle \geq -2\mu \|x^{(k+1)} - x^*\|_2^2 - \frac{\rho^2}{2\mu} \|A_D^T A_\Delta (x^{(k)} - x^{(k+1)})\|_2^2. \tag{34}$$

Now, combining (33) and (34) gives

$$\begin{aligned}
\|u^{(k)} - u^*\|_G^2 &= \|u^{(k+1)} - u^*\|_G^2 \\
&\geq \frac{2-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \|x^{(k)} - x^{(k+1)}\|_{G_x}^2 - \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 \|x^{(k)} - x^{(k+1)}\|_2^2 \\
&= \frac{2-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \sum_{i=1}^n \|x_i^{(k)} - x_i^{(k+1)}\|_{P_i - \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 I}^2
\end{aligned} \tag{35}$$

and note that Assumption 4 guarantees that

$$T_i := P_i - \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 I \succ 0.$$

If we then let $\eta_i := \lambda_{\min}(T_i)/\|P_i\|_2 > 0$, we have from the definition of T_i and standard norm inequalities

$$\begin{aligned}
\|x_i^{(k)} - x_i^{(k+1)}\|_{P_i - \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 I}^2 &= \|x_i^{(k)} - x_i^{(k+1)}\|_{T_i}^2 \\
&\geq \lambda_{\min}(T_i) \|x_i^{(k)} - x_i^{(k+1)}\|_2^2 \\
&= \eta_i \|P_i\|_2 \|x_i^{(k)} - x_i^{(k+1)}\|_2^2 \\
&\geq \eta_i (x_i^{(k)} - x_i^{(k+1)})^T P_i (x_i^{(k)} - x_i^{(k+1)}) = \eta_i \|x_i^{(k)} - x_i^{(k+1)}\|_{P_i}^2.
\end{aligned}$$

Combining this with (35) gives

$$\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 \geq \frac{2-\gamma}{\gamma^2\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \sum_{i=1}^n \eta_i \|x_i^{(k)} - x_i^{(k+1)}\|_{P_i}^2.$$

From the previous inequality and the definition

$$\eta := \min \left\{ \frac{2-\gamma}{\gamma}, \min_{1 \leq i \leq n} \eta_i \right\} > 0,$$

we have

$$\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 \geq \eta \left(\frac{1}{\gamma\rho} \|y^{(k)} - y^{(k+1)}\|_2^2 + \sum_{i=1}^n \|x_i^{(k)} - x_i^{(k+1)}\|_{P_i}^2 \right) = \eta \|u^{(k)} - u^{(k+1)}\|_G^2,$$

which is the desired result. $\square$

We we may now state our main convergence result for Algorithm 5.

Theorem 8. *If the conditions of Theorem 7 hold, then the sequence $\{u^{(k)}\}_{k \geq 0}$ generated by Algorithm 5 converges to some vector u^L that is a solution to problem (1).*

Proof. Let u^* be any solution in U^* . It then follows from Theorem 7 that

$$\|u^{(k)} - u^*\|_G \leq \|u^{(0)} - u^*\|_G \text{ for all } k \geq 1, \tag{36}$$

so that $\{u^{(k)}\}_{k \geq 0}$ is a bounded sequence. Moreover, for any integer $p \geq 1$, it follows from (7) that

$$\sum_{k=1}^p \eta \|u^{(k)} - u^{(k+1)}\|_G^2 \leq \sum_{k=1}^p \left(\|u^{(k)} - u^*\|_G^2 - \|u^{(k+1)} - u^*\|_G^2 \right) = \|u^{(0)} - u^*\|_G^2 - \|u^{(p+1)} - u^*\|_G^2 \leq \|u^{(0)} - u^*\|_G^2.$$

Taking limits of both sides of the previous inequality as $p \rightarrow \infty$ shows that the sum is finite, and since all the summands are nonnegative that

$$\lim_{k \rightarrow \infty} \|u^{(k)} - u^{(k+1)}\|_G = 0. \quad (37)$$

Next, using the boundedness of $\{u^{(k)}\}_{k \geq 0}$, we may conclude the existence of a subsequence $\mathcal{K} \subseteq \{1, 2, \dots\}$ and a vector $u^L \in \mathbf{R}^{N+m}$ such that

$$\lim_{k \in \mathcal{K}} u^{(k)} = u^L. \quad (38)$$

It follows from (38), (37), and Lemma 6 that u^L is a solution to problem (1). Finally, since (24) held for *any* $u^* \in U^*$ and we have proved that $u^L \in U^*$, it follows that $\lim_{k \rightarrow \infty} u^{(k)} = u^L$, as desired. $\square$

3 A Hybrid ADMM (H-ADMM)

One of the disadvantages of a Gauss-Seidel type updating scheme within ADMM is that it is inherently serial. With problem dimension growing ever larger in this era of big data, and the ubiquity of parallel processing power, a Jacobi type updating scheme may be preferable in many real-world instances of problem (1). The purpose of this section is to show that if F-ADMM is applied to “grouped data”, and a special choice of regularization matrix is employed for each group, then Algorithm 5 becomes a *hybrid* Gauss-Seidel/Jacobi ADMM-type method. Therefore, Algorithm 5 is *partially parallelizable*.

3.1 Notation and Assumptions

Suppose that the function $f(x)$ is separable into n blocks, as in (1a). Then, we can (implicitly) partition the variables x_i and functions $f_i(x_i)$ together into $\ell < n$ groups. For simplicity of exposition, we will assume that n is divisible by some p , so that $\ell p = n$, which means that we form ℓ groups of p blocks. Then, problem (1) is equivalent to the following partitioned problem:

$$\underset{x \in \mathbf{R}^N}{\text{minimize}} \quad f(x) \equiv \sum_{j=1}^{\ell} \mathbf{f}_j(\mathbf{x}_j) \quad (39a)$$

$$\text{subject to} \quad \sum_{j=1}^{\ell} \mathcal{A}_j \mathbf{x}_j = b \quad (39b)$$

with

$$\mathbf{x}_1 := \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}, \quad \mathbf{x}_2 := \begin{bmatrix} x_{p+1} \\ \vdots \\ x_{2p} \end{bmatrix}, \quad \dots \quad \mathbf{x}_\ell := \begin{bmatrix} x_{(\ell-1)p+1} \\ \vdots \\ x_n \end{bmatrix}, \quad (40)$$

$$\mathbf{f}_1(\mathbf{x}_1) := \sum_{i=1}^p f_i(x_i), \quad \mathbf{f}_2(\mathbf{x}_2) := \sum_{i=p+1}^{2p} f_i(x_i), \quad \dots \quad \mathbf{f}_\ell(\mathbf{x}_\ell) := \sum_{i=(\ell-1)p+1}^n f_i(x_i),$$

and

$$\mathcal{A}_1 := [A_1 \quad \dots \quad A_p], \quad \mathcal{A}_2 := [A_{p+1} \quad \dots \quad A_{2p}], \quad \dots \quad \mathcal{A}_\ell := [A_{(\ell-1)p+1} \quad \dots \quad A_n].$$

Notice that $A = [\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_\ell] \equiv [A_1, A_2, \dots, A_n]$, and $x = [\mathbf{x}_1^T, \dots, \mathbf{x}_\ell^T]^T \equiv [x_1^T, \dots, x_n^T]^T$. Furthermore, it will be useful to define the index sets

$$\mathcal{S}_1 = \{1, \dots, p\}, \quad \mathcal{S}_2 = \{p+1, \dots, 2p\}, \quad \dots \quad \mathcal{S}_\ell = \{(\ell-1)p+1, \dots, n\} \quad (41)$$

associated with the partition described above, and to use the notation $\mathcal{S}_{i,j}$ to denote the j th element of $\mathcal{S}_i$.

We now think of applying Algorithm 5 to the ℓ groups of data. That is, in Step 3 of Algorithm 5 we have ℓ minimization problems, one for each of the grouped data points $\mathbf{x}_j$ (rather than n minimization problems, one for each of the individual data blocks x_i). For the grouped data, we require regularization matrices $\mathcal{P}_1, \dots, \mathcal{P}_\ell$, for each of the ℓ groups; these matrices will be crucial in our upcoming derivation.

To motivate the idea of “grouped data”, and to make the ideas that will be discussed in this rest of this section more concrete, we give a specific example that shows how our hybrid algorithm will work.

Example 9. Suppose there are $n = 12$ blocks and we have access to a parallel computer with $p = 4$ processors. We make a formal partition of the data into $\ell = 3$ groups. That is, we set $\mathbf{f}_1(\mathbf{x}_1) = \sum_{i=1}^4 f_i(x_i)$, $\mathbf{f}_2(\mathbf{x}_2) = \sum_{i=5}^8 f_i(x_i)$ and $\mathbf{f}_3(\mathbf{x}_3) = \sum_{i=9}^{12} f_i(x_i)$, $\mathbf{x}_1 = [x_1^T, \dots, x_4^T]^T$, $\mathbf{x}_2 = [x_5^T, \dots, x_8^T]^T$, and $\mathbf{x}_3 = [x_9^T, \dots, x_{12}^T]^T$, partition the matrix A accordingly, and initialize index sets $\mathcal{S}_1 = \{1, \dots, 4\}$, $\mathcal{S}_2 = \{5, \dots, 8\}$ and $\mathcal{S}_3 = \{9, \dots, 12\}$. Then, a single iteration of H-ADMM (see Steps 3–7 of Algorithm 6) will run in the following way. The Lagrange multiplier estimate $y^{(k)}$ and (group) variables $\mathbf{x}_2^{(k)}$ and $\mathbf{x}_3^{(k)}$ are fixed. Group variable $\mathbf{x}_1$ is updated by solving a subproblem of the form (52) for each of $x_1, \dots, x_4$ in parallel. This gives the new point $\mathbf{x}_1^{(k+1)}$. Then, $\mathbf{x}_1^{(k+1)}$ and $\mathbf{x}_3^{(k)}$ are fixed, and four subproblems of the form (52) are solved for each of $x_5, \dots, x_8$ in parallel, giving $\mathbf{x}_2^{(k+1)}$. Next, $\mathbf{x}_1^{(k+1)}$ and $\mathbf{x}_2^{(k+1)}$ are fixed, and four subproblems of the form (52) are solved for each of $x_9, \dots, x_{12}$ in parallel, giving $\mathbf{x}_3^{(k+1)}$. Finally, $y^{(k+1)}$ is updated in (53).

Example 9 shows that Algorithm 6 is running a Gauss-Seidel process on the group variables, but running a Jacobi process to update the individual blocks within each group. This example shows an efficient implementation in the sense that, by ensuring that the group size p matches the number of processors, all processors are always engaged, and that updated information is utilized when it is available.

In the rest of this section we explain how H-ADMM (Algorithm 6) is obtained from F-ADMM.

3.2 Separability Via Regularization

We show that, if the regularization matrices $\{\mathcal{P}_i\}_{i=1}^\ell$ are chosen appropriately, F-ADMM can be partially parallelized, and forms the hybrid algorithm H-ADMM. In particular, for the i th subproblem in F-ADMM (applied to the grouped data in (39)), the p blocks within the i th group can be solved for in parallel.

In what follows, we use the relationships

$$\left\| \sum_{j=1}^n A_j x_j \right\|_2^2 = \sum_{j=1}^n \langle A_j x_j, A_j x_j \rangle + \sum_{j=1}^n \sum_{\substack{l=1 \\ l \neq j}}^n \langle A_j x_j, A_l x_l \rangle \quad \text{and} \quad (42)$$

$$\sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l, A_j x_j \rangle = \sum_{j=1}^p \sum_{\substack{l=1 \\ l \neq j}}^p \langle A_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}}, A_{\mathcal{S}_{i,l}} x_{\mathcal{S}_{i,l}} \rangle, \quad (43)$$

which can easily be verified. Using the definition of $\mathcal{A}_i$ and a similar reasoning as for (42), it follows that

$$\|\mathcal{A}_i \mathbf{x}_i\|_2^2 = \left\| \sum_{j \in \mathcal{S}_i} A_j x_j \right\|_2^2 = \sum_{j \in \mathcal{S}_i} \|A_j x_j\|_2^2 + \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_j x_j, A_l x_l \rangle. \quad (44)$$

We now define $\mathbf{b}_i := b - \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} - \sum_{s=i+1}^\ell \mathcal{A}_s \mathbf{x}_s^{(k)}$, and notice that $\mathbf{b}_i$ is fixed when minimizing the augmented Lagrangian with respect to group $\mathbf{x}_i$. Recalling Algorithm 5 and (18), and using (44), the update

for the i th subproblem for our grouped data problem *without* the regularization term is equivalent to

$$\begin{aligned}
\mathbf{x}_i^{(k+1)} &= \arg \min_{\mathbf{x}_i} \mathcal{L}_\rho(\mathbf{x}_1^{(k+1)}, \dots, \mathbf{x}_{i-1}^{(k+1)}, \mathbf{x}_i, \mathbf{x}_{i+1}^{(k)}, \dots, \mathbf{x}_\ell^{(k)}; y^{(k)}) \\
&= \arg \min_{\mathbf{x}_i} \left\{ \mathbf{f}_i(\mathbf{x}_i) - \langle y^{(k)}, \mathcal{A}_i \mathbf{x}_i - \mathbf{b}_i \rangle + \frac{\rho}{2} \|\mathcal{A}_i \mathbf{x}_i - \mathbf{b}_i\|_2^2 \right\} \\
&= \arg \min_{\mathbf{x}_i} \left\{ \mathbf{f}_i(\mathbf{x}_i) - \langle y^{(k)}, \mathcal{A}_i \mathbf{x}_i - \mathbf{b}_i \rangle + \frac{\rho}{2} \|\mathbf{b}_i\|_2^2 + \frac{\rho}{2} \|\mathcal{A}_i \mathbf{x}_i\|_2^2 - \rho \langle \mathcal{A}_i \mathbf{x}_i, \mathbf{b}_i \rangle \right\} \\
&= \arg \min_{\mathbf{x}_i} \left\{ \mathbf{f}_i(\mathbf{x}_i) - \langle y^{(k)}, \mathcal{A}_i \mathbf{x}_i - \mathbf{b}_i \rangle - \rho \langle \mathcal{A}_i \mathbf{x}_i, \mathbf{b}_i \rangle + \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \|A_j x_j\|_2^2 + \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l, A_j x_j \rangle \right\}. \quad (45)
\end{aligned}$$

Notice that it is the final term in (45) that makes the minimization of the augmented Lagrangian (with respect to the group $\mathbf{x}_i$) non-separable; it contains a cross product term, which shows interaction between different blocks of variables within the i th group indexed by $\mathcal{S}_i$.

3.2.1 Defining the group regularization matrices

We eliminate the non-separability in (45) by carefully choosing the regularization matrices $\{\mathcal{P}_i\}_{i=1}^\ell$. From a practical perspective, if the problem is made separable, then the individual blocks within the i th group can be *updated in parallel*. To this end, we choose the matrix that defines our regularizer to be

$$\mathcal{P}_i := \begin{bmatrix} P_{\mathcal{S}_{i,1}} & -\rho A_{\mathcal{S}_{i,1}}^T A_{\mathcal{S}_{i,2}} & \dots & -\rho A_{\mathcal{S}_{i,1}}^T A_{\mathcal{S}_{i,p}} \\ -\rho A_{\mathcal{S}_{i,2}}^T A_{\mathcal{S}_{i,1}} & P_{\mathcal{S}_{i,2}} & & \vdots \\ \vdots & & \ddots & \\ \vdots & & & P_{\mathcal{S}_{i,p-1}} & -\rho A_{\mathcal{S}_{i,p-1}}^T A_{\mathcal{S}_{i,p}} \\ -\rho A_{\mathcal{S}_{i,p}}^T A_{\mathcal{S}_{i,1}} & \dots & -\rho A_{\mathcal{S}_{i,p}}^T A_{\mathcal{S}_{i,p-1}} & P_{\mathcal{S}_{i,p}} \end{bmatrix}. \quad (46)$$

We remind the reader that the matrices $\{P_{\mathcal{S}_{i,j}}\}_{j=1}^p$ used to define $\mathcal{P}_i$ are user defined symmetric matrices that must be chosen to be sufficiently positive definite, to ensure that convergence of F-ADMM on the grouped data is guaranteed. Before we formalize our assumption, we require the definitions

$$\mathcal{A}_\Delta := \begin{bmatrix} \mathcal{A}_2 & \dots & \mathcal{A}_\ell \\ & \ddots & \vdots \\ & & \mathcal{A}_\ell \end{bmatrix} \quad \text{and} \quad \mathcal{A}_D := \begin{bmatrix} \mathcal{A}_1 & & \\ & \ddots & \\ & & \mathcal{A}_\ell \end{bmatrix}, \quad (47)$$

where $\{\mathcal{A}_\Delta, \mathcal{A}_D\} \subset \mathbf{R}^{m\ell \times N}$. We then have the strictly (block) upper triangular matrix

$$\mathcal{A}_D^T \mathcal{A}_\Delta = \begin{bmatrix} \mathcal{A}_1^T \mathcal{A}_2 & \dots & \mathcal{A}_1^T \mathcal{A}_\ell \\ & \ddots & \vdots \\ & & \mathcal{A}_{\ell-1}^T \mathcal{A}_\ell \end{bmatrix} \in \mathbf{R}^{N \times N}. \quad (48)$$

Notice that the definitions of $\mathcal{A}_D$, $\mathcal{A}_\Delta$ and $\mathcal{A}_D^T \mathcal{A}_\Delta$ in (47) and (48), are analogues to A_D , A_Δ and $A_D^T A_\Delta$ defined in (15) and (16). We are now ready to state our assumption on $\{\mathcal{P}_i\}_{i=1}^\ell$, which is actually Assumption 4 applied to the grouped data problem (39).

Assumption 10. *The matrices $\mathcal{P}_i$ are symmetric and satisfy $\mathcal{P}_i \succ \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 I$ for all $i = 1, \dots, \ell$.*

Importantly, if F-ADMM is applied to the grouped data problem (39) and Assumption 10 holds, then convergence is automatic, i.e., convergence of F-ADMM equipped with Assumption 10 applied to problem (39) follows directly from the convergence results presented in Section 2.

3.2.2 Incorporating the regularization term

Now that the regularization matrices $\{\mathcal{P}_i\}_{i=1}^\ell$ are defined, we return to the non-separability encountered in (45). Recall that the subproblem in Step 3 of F-ADMM (Algorithm 5) is equivalent to (18), which in turn is equivalent to (45) + $\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^{(k)}\|_{\mathcal{P}_i}^2$. We concentrate on the regularization term, and notice that

$$\begin{aligned}
\frac{1}{2}\mathbf{x}_i^T \mathcal{P}_i \mathbf{x}_i &\stackrel{(40)+(41)}{=} \frac{1}{2} \begin{bmatrix} x_{\mathcal{S}_{i,1}}^T & \dots & x_{\mathcal{S}_{i,p}}^T \end{bmatrix} \begin{bmatrix} P_{\mathcal{S}_{i,1}} x_{\mathcal{S}_{i,1}} - \rho A_{\mathcal{S}_{i,1}}^T \left(\sum_{\substack{j=1 \\ j \neq 1}}^p A_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}} \right) \\ \vdots \\ P_{\mathcal{S}_{i,p}} x_{\mathcal{S}_{i,p}} - \rho A_{\mathcal{S}_{i,p}}^T \left(\sum_{\substack{j=1 \\ j \neq p}}^p A_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}} \right) \end{bmatrix} \\
&= \frac{1}{2} \sum_{j=1}^p \|x_{\mathcal{S}_{i,j}}\|_{P_{\mathcal{S}_{i,j}}}^2 - \frac{\rho}{2} \sum_{j=1}^p \sum_{\substack{l=1 \\ l \neq j}}^p \langle A_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}}, A_{\mathcal{S}_{i,l}} x_{\mathcal{S}_{i,l}} \rangle \\
&\stackrel{(43)}{=} \frac{1}{2} \sum_{j \in \mathcal{S}_i} \|x_j\|_{P_j}^2 - \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l, A_j x_j \rangle. \tag{49}
\end{aligned}$$

Following a similar argument, we can write

$$\begin{aligned}
\mathbf{x}_i^T \mathcal{P}_i \mathbf{x}_i^{(k)} &= \sum_{j=1}^p x_{\mathcal{S}_{i,j}}^T P_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}}^{(k)} - \rho \sum_{j=1}^p \sum_{\substack{l=1 \\ l \neq j}}^p \langle A_{\mathcal{S}_{i,j}} x_{\mathcal{S}_{i,j}}, A_{\mathcal{S}_{i,l}} x_{\mathcal{S}_{i,l}}^{(k)} \rangle \\
&= \sum_{j \in \mathcal{S}_i} x_j^T P_j x_j^{(k)} - \rho \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j \rangle. \tag{50}
\end{aligned}$$

We may now use (49) and (50) to write

$$\begin{aligned}
\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^{(k)}\|_{\mathcal{P}_i}^2 &= \frac{1}{2} \sum_{j \in \mathcal{S}_i} \|x_j\|_{P_j}^2 - \sum_{j \in \mathcal{S}_i} x_j^T P_j x_j^{(k)} + \frac{1}{2} \sum_{j \in \mathcal{S}_i} \|x_j^{(k)}\|_{P_j}^2 \\
&\quad + \rho \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j \rangle - \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l, A_j x_j \rangle - \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j^{(k)} \rangle.
\end{aligned}$$

This may be equivalently written as

$$\begin{aligned}
&\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^{(k)}\|_{\mathcal{P}_i}^2 \\
&= \frac{1}{2} \sum_{j \in \mathcal{S}_i} \|x_j - x_j^{(k)}\|_{P_j}^2 + \rho \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j \rangle - \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l, A_j x_j \rangle - \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j^{(k)} \rangle.
\end{aligned}$$

By adding this regularization term, i.e., $\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^{(k)}\|_{\mathcal{P}_i}^2$, to the objective function in (45), we obtain (ignoring terms independent of $\mathbf{x}_i$) the F-ADMM update

$$\begin{aligned}
\mathbf{x}_i^{(k+1)} &= \arg \min_{\mathbf{x}_i} \left\{ \mathbf{f}_i(\mathbf{x}_i) - \langle y^{(k)}, \mathcal{A}_i \mathbf{x}_i - \mathbf{b}_i \rangle - \rho \langle \mathcal{A}_i \mathbf{x}_i, \mathbf{b}_i \rangle \right. \\
&\quad \left. + \frac{\rho}{2} \sum_{j \in \mathcal{S}_i} \|A_j x_j\|_2^2 + \rho \sum_{j \in \mathcal{S}_i} \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j \rangle + \frac{1}{2} \sum_{j \in \mathcal{S}_i} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\},
\end{aligned}$$

which is equivalent (again ignoring constant terms) to

$$\begin{aligned} \mathbf{x}_i^{(k+1)} &= \arg \min_{\mathbf{x}_i} \sum_{j \in \mathcal{S}_i} \left\{ f_j(x_j) - \langle y^{(k)}, A_j x_j \rangle - \rho \langle A_j x_j, \mathbf{b}_i \rangle + \frac{\rho}{2} \|A_j x_j\|_2^2 + \rho \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} \langle A_l x_l^{(k)}, A_j x_j \rangle + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\} \\ &= \arg \min_{\mathbf{x}_i} \sum_{j \in \mathcal{S}_i} \left\{ f_j(x_j) + \frac{\rho}{2} \|A_j x_j + \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} A_l x_l^{(k)} - \mathbf{b}_i - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\}. \end{aligned} \quad (51)$$

The regularization matrix P_i , defined in (46), has caused the cross-product term to be eliminated from (45) (recall that (45) was the update *without* using the regularization term), and subsequently the subproblem for updating $\mathbf{x}_i^{(k+1)}$ is separable into p blocks (one solve for each $j \in \mathcal{S}_i$). That is, the decision variables x_j for $j \in \mathcal{S}_i$ can be solved for *in parallel*. This updating strategy forms our hybrid algorithm H-ADMM, which is able to use a combination of *both* Jacobi and Gauss-Seidel updates. We emphasize that H-ADMM is a special case of Algorithm 5, where the blocks of variables have been (implicitly) grouped together as in (39), and the regularization matrices have the form (46).

3.2.3 The H-ADMM Algorithm

The following is a formal statement of our H-ADMM algorithm. Recall that H-ADMM is a special case of F-ADMM, and convergence of H-ADMM follows directly from the convergence theory for F-ADMM.

Algorithm 6 H-ADMM for solving problem (1).

- 1: **Initialize:** $x^{(0)} \in \mathbf{R}^N$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, parameters $\rho > 0$ and $\gamma \in (0, 2)$, data partition index sets $\{\mathcal{S}_i\}_{i=1}^\ell$, and regularization matrices $\{P_i\}_{i=1}^n$ satisfying Assumption 10.
- 2: **while** stopping condition has not been met **do**
- 3: **for** $i = 1, \dots, \ell$ in a Gauss-Seidel fashion solve **do**
- 4: Set $\mathbf{b}_i \leftarrow b - \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} - \sum_{s=i+1}^\ell \mathcal{A}_s \mathbf{x}_s^{(k)}$.
- 5: **for** $j \in \mathcal{S}_i$ (in parallel) **do**

$$x_j^{(k+1)} \leftarrow \arg \min_{x_j} \sum_{j \in \mathcal{S}_i} \left\{ f_j(x_j) + \frac{\rho}{2} \|A_j x_j + \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} A_l x_l^{(k)} - \mathbf{b}_i - \frac{y^{(k)}}{\rho}\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\} \quad (52)$$

- 6: **end for**
- 7: **end for**
- 8: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \gamma \rho (Ax^{(k+1)} - b). \quad (53)$$

- 9: Set $k \leftarrow k + 1$.

10: **end while**

The groups of data are updated in a Gauss-Seidel scheme (see the **for** loop in Step 3), while the individual blocks within each group are updated in a Jacobi (parallel) scheme (see the inner **for** loop in Step 5).

We have presented H-ADMM as a (serial) Gauss-Seidel algorithm that has an inner loop which can be executed in parallel, i.e., H-ADMM is partially parallel. However, we can also view H-ADMM as a fully parallel method that occasionally inserts updated information during the update from $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k+1)}$. This shows that H-ADMM is extremely flexible.

Remark 11. Notice that the regularization matrix (46) is not explicitly formed in H-ADMM (Algorithm 6). Specifically, H-ADMM only uses the matrices $P_{\mathcal{S}_{i,1}}, \dots, P_{\mathcal{S}_{i,p}}$, which lie on the main (block) diagonal of P_i . Therefore, as for F-ADMM, only one $N_i \times N_i$ regularization matrix P_i is required by H-ADMM for each of the $i = 1, \dots, n$ (individual) blocks.

Remark 12. *The update (52) in H-ADMM has the same form as the update (8a) in J-ADMM (Algorithm 4) for all $j \in S_i$. Therefore, a single iteration of H-ADMM has essentially the same cost as that of J-ADMM. However, H-ADMM has the advantage of using the most recent updates when updating the groups of data.*

3.3 Computational Considerations

Parallel algorithms are imperative on modern computer architectures, which is why, at face value, Jacobi-type methods seem to have significant advantages over Gauss-Seidel-type competitors. The H-ADMM (Algorithm 6) bridges the gap between purely Jacobi or purely Gauss-Seidel updates, finding a balance between ensuring algorithm speed via parallelization and allowing up-to-date information to be fed back into the algorithm. In this section we describe how to choose the number of groups ℓ and group size p to “optimize” H-ADMM from a computational perspective. Moreover, we show that H-ADMM is competitive compared with J-ADMM.

Consider a big data application where the number of blocks n is very large. Moreover, suppose we have access to a parallel machine with p processors, where $p < n$ (or even $p \ll n$). Again we will assume that $n = \ell p$, and the n blocks are organized into ℓ groups of p blocks. We stress that *the number of blocks in each group is the same as the number of processors*.

To implement H-ADMM we first initialize $\mathbf{b}_1$. Then, take the first group of p blocks and send one block to each of the p processors. These p blocks are updated in parallel as in (52) (Step 5 of Algorithm 6). Once these p blocks have been updated, we have the updated group variable $\mathbf{x}_1^{(k+1)}$ consisting of individual blocks $x_1^{(k+1)}, \dots, x_p^{(k+1)}$. We then form $\mathbf{b}_2$ as in Step 4 of Algorithm 6. Notice that $\mathbf{b}_2$ incorporates the new information from the updated block $\mathbf{x}_1^{(k+1)}$ via the term $\mathcal{A}_1 \mathbf{x}_1^{(k+1)}$, i.e., we feed the updated information back into the algorithm. Now, the next group of p blocks are sent to the p processors to be updated, giving $\mathbf{x}_2^{(k+1)}$ consisting of individual blocks $x_{p+1}^{(k+1)}, \dots, x_{2p}^{(k+1)}$. This new information is then fed back into H-ADMM via the vector $\mathbf{b}_3$. The process is repeated until a full sweep of the data has been completed, i.e., all n blocks have been updated.

In this way, our H-ADMM algorithm has (essentially) the same computational cost as J-ADMM, because the data blocks have been grouped in an intelligent way that takes advantage of the processors available. (For J-ADMM, the data blocks also need to be sent to processors in groups of p , it is just that, for J-ADMM, there is no need to update the vector $\mathbf{b}_i$ between the ℓ sweeps of the processors.) We note that for J-ADMM, the matrix-vector multiplication $Ax^{(k+1)}$ is computed once *all n blocks of x have been updated* (i.e., once $x^{(k+1)}$ is available), whereas for H-ADMM, the computation of $Ax^{(k+1)}$ has been split and performed in stages with the vectors $\mathcal{A}_i \mathbf{x}_i^{(k+1)}$ (for $i = 1, \dots, \ell$) computed after *each group of data has been updated* and the sum taken just before the dual variables are updated. Again, this shows that H-ADMM and J-ADMM have approximately the same computational cost, but H-ADMM has the advantage of new information becoming available to the algorithm, which has the potential for H-ADMM to be more efficient.

Remark 13. *Notice that if $n \leq p$, then H-ADMM is essentially equivalent to J-ADMM (applied to strongly convex functions) if we take $\ell = 1$ and replace Step 4 with $\mathbf{b}_1 \equiv b$.*

3.3.1 An efficient implementation of Steps 4–6 in Algorithm 6

Algorithm 6 was written to match our presentation in the text. However, in practice, it is computationally advantageous to perform Steps 4–6 in a different, but equivalent way. To that end, consider the middle term

in the minimization subproblem (52), and using the definition of $\mathbf{b}_i$ (Step 4 in Algorithm 6) we have

$$\begin{aligned}
A_j x_j + \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} A_l x_l^{(k)} - \mathbf{b}_i - \frac{y^{(k)}}{\rho} &= A_j x_j + \sum_{\substack{l \in \mathcal{S}_i \\ l \neq j}} A_l x_l^{(k)} + \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} + \sum_{s=i+1}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} - b - \frac{y^{(k)}}{\rho} \\
&= A_j x_j - A_j x_j^{(k)} + \sum_{l \in \mathcal{S}_i} A_l x_l^{(k)} + \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} + \sum_{s=i+1}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} - b - \frac{y^{(k)}}{\rho} \\
&= A_j x_j - A_j x_j^{(k)} + \mathcal{A}_i \mathbf{x}_i^{(k)} + \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} + \sum_{s=i+1}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} - b - \frac{y^{(k)}}{\rho} \\
&= A_j x_j - A_j x_j^{(k)} + \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} + \sum_{s=i}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} - b - \frac{y^{(k)}}{\rho}. \tag{54}
\end{aligned}$$

Notice that the last 4 terms in (54) are fixed with respect to $j \in \mathcal{S}_i$, so we can combine them into a single vector v_i say, and rewrite Steps 4-6 in Algorithm 6 as follows.

Algorithm 7 An efficient implementation to replace Steps 4-6 in H-ADMM.

- 1: Set $v_i \leftarrow \sum_{q=1}^{i-1} \mathcal{A}_q \mathbf{x}_q^{(k+1)} + \sum_{s=i}^{\ell} \mathcal{A}_s \mathbf{x}_s^{(k)} - b - \frac{y^{(k)}}{\rho}$.
- 2: **for** $j \in \mathcal{S}_i$ (in parallel) **do**

$$x_j^{(k+1)} \leftarrow \arg \min_{x_j} \sum_{j \in \mathcal{S}_i} \left\{ f_j(x_j) + \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_i\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\}. \tag{55}$$

- 3: **end for**

3.3.2 Practical considerations regarding Assumption 10

As discussed in Section 3.2.1, choosing the regularization matrices to have the form (46) for $i = 1, \dots, \ell$ in a manner that satisfies Assumption 10, ensures that H-ADMM is globally convergent. However, we have remarked that an implementation of H-ADMM only needs the individual (diagonal) block matrices $\{P_j\}_{j=1}^n$. The purpose of this section is to translate Assumption 10, which is an assumption on the group regularization matrices $\{\mathcal{P}_i\}_{i=1}^{\ell}$, into a *practical condition* on the matrices $\{P_j\}_{j=1}^n$.

To this end, recall the definition of $\mathcal{P}_i$ in (46). If we define

$$\mathcal{P}_i^D := \begin{bmatrix} P_{\mathcal{S}_{i,1}} + \rho \mathcal{A}_{\mathcal{S}_{i,1}}^T \mathcal{A}_{\mathcal{S}_{i,1}} & & \\ & \ddots & \\ & & P_{\mathcal{S}_{i,p}} + \rho \mathcal{A}_{\mathcal{S}_{i,p}}^T \mathcal{A}_{\mathcal{S}_{i,p}} \end{bmatrix}, \tag{56}$$

then Assumption 10 can be written equivalently as $\mathcal{P}_i \equiv \mathcal{P}_i^D - \rho \mathcal{A}_i^T \mathcal{A}_i \succ \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 I$, which holds if and only if $\mathcal{P}_i^D \succ \rho \mathcal{A}_i^T \mathcal{A}_i + \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 I$. Using $\rho \|\mathcal{A}_i\|_2^2 I \succeq \rho \mathcal{A}_i^T \mathcal{A}_i$, a sufficient condition for Assumption 10 to hold is that

$$\mathcal{P}_i^D \succ \rho \|\mathcal{A}_i\|_2^2 I + \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 I. \tag{57}$$

It then follows from the definition of $\mathcal{P}_i^D$ that (57) will hold (equivalently, Assumption 10 will be satisfied) if the matrices $\{P_j\}_{j \in \mathcal{S}_i}$ are chosen to satisfy

$$P_j + \rho \mathcal{A}_j^T \mathcal{A}_j \succ \rho \|\mathcal{A}_i\|_2^2 I + \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 I \quad \forall j \in \mathcal{S}_i. \tag{58}$$

That is, if (58) is satisfied for all $1 \leq i \leq \ell$, then H-ADMM is globally convergent.

3.4 A Special Case for Convex Functions

In this section, we describe how our hybrid algorithm is appropriate for convex functions (i.e., we do not need strong convexity) in the case of 2 groups. In particular, it is based on Algorithm 3, which was first introduced in [7] and shown to be globally convergent if the regularization matrices P_1 and P_2 in (7) are chosen appropriately. In particular, the convergence theory introduced in [7] holds when $P_1 \succ 0$ and $P_2 \succ 0$.

During the remainder of this section, we demonstrate that by choosing the regularization matrix appropriately, Algorithm 3 can be extended to handle the n block case while maintaining all existing convergence theory. This is done by following the hybridization scheme introduced previously in this section.

So, suppose that we have an optimization problem of the form (1), and that we partition the n blocks into 2 groups, i.e., we have $\ell = 2$ groups and, for simplicity, assume that $p = n/2$.¹ We can then equivalently write our problem in the form (39) where

$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ \vdots \\ x_{n/2} \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} x_{n/2+1} \\ \vdots \\ x_n \end{bmatrix}, \quad (59a)$$

$$\mathbf{f}_1(\mathbf{x}_1) = \sum_{i=1}^{n/2} f_i(x_i), \quad \mathbf{f}_2(\mathbf{x}_2) = \sum_{i=n/2+1}^n f_i(x_i), \quad (59b)$$

and

$$\mathcal{A}_1 = [A_1 \quad \dots \quad A_{n/2}] \quad \text{and} \quad \mathcal{A}_2 = [A_{n/2+1} \quad \dots \quad A_n]. \quad (59c)$$

It is clear that we can apply Algorithm 3 to the grouped data (59). If we now choose the regularization matrices $\mathcal{P}_1$ and $\mathcal{P}_2$ to have the same form as in (46), we have

$$\mathcal{P}_1 = \begin{bmatrix} P_1 & -\rho A_1^T A_2 & \dots & -\rho A_1^T A_{n/2} \\ -\rho A_2^T A_1 & P_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ -\rho A_{n/2}^T A_1 & \dots & \dots & P_{n/2} \end{bmatrix} \quad (60)$$

and

$$\mathcal{P}_2 = \begin{bmatrix} P_{n/2+1} & -\rho A_{n/2+1}^T A_{n/2+2} & \dots & -\rho A_{n/2+1}^T A_n \\ -\rho A_{n/2+2}^T A_{n/2+1} & P_{n/2+2} & & \vdots \\ \vdots & & \ddots & \vdots \\ -\rho A_n^T A_{n/2+1} & \dots & \dots & P_n \end{bmatrix}. \quad (61)$$

Moreover, by letting

$$\mathcal{P}_1^D = \begin{bmatrix} P_1 + \rho A_1^T A_1 & & \\ & \ddots & \\ & & P_{n/2} + \rho A_{n/2}^T A_{n/2} \end{bmatrix} \quad \text{and} \quad \mathcal{P}_2^D = \begin{bmatrix} P_{n/2+1} + \rho A_{n/2+1}^T A_{n/2+1} & & \\ & \ddots & \\ & & P_n + \rho A_n^T A_n \end{bmatrix}$$

we have that $\mathcal{P}_1 = \mathcal{P}_1^D - \rho \mathcal{A}_1^T \mathcal{A}_1$ and $\mathcal{P}_2 = \mathcal{P}_2^D - \rho \mathcal{A}_2^T \mathcal{A}_2$. Thus, a sufficient condition for the matrices $\mathcal{P}_1$ and $\mathcal{P}_2$ to be positive definite is that

$$\mathcal{P}_1^D \succ \rho \|\mathcal{A}_1\|_2^2 I \quad \text{and} \quad \mathcal{P}_2^D \succ \rho \|\mathcal{A}_2\|_2^2 I. \quad (62)$$

¹From a practical perspective it is sensible to let the first group have a cardinality that is a multiple of the number of processors, and then the second group would contain the remaining blocks.

We can then see that a sufficient condition for (62) to hold is to choose the matrices $\{P_j\}_{j=1}^n$ to satisfy

$$P_j \succ \rho \|\mathcal{A}_1\|_2^2 I \text{ for } 1 \leq j \leq n/2, \text{ and } P_j \succ \rho \|\mathcal{A}_2\|_2^2 I \text{ for } n/2 + 1 \leq j \leq n. \quad (63)$$

In summary, if the matrices $\{P_j\}_{j=1}^n$ are chosen to satisfy (63), then Algorithm 8 is guaranteed to converge for *convex functions*. We also comment that $\mathcal{P}_1$ and $\mathcal{P}_2$ in (60) and (61) need not be formed explicitly, since Algorithm 8 only requires the block diagonal regularization matrices $\{P_j\}_{j=1}^n$ be chosen to satisfy (63).

Algorithm 8 H-ADMM($\ell = 2$) for solving problem (1) with a convex objective.

1: **Initialization:** $x^{(0)} \in \mathbf{R}^N$, $y^{(0)} \in \mathbf{R}^m$, iteration counter $k = 0$, parameters $\rho > 0$ and $\gamma \in (0, 2)$, and regularization matrices $\{P_j\}_{j=1}^n$.

2: **while** the stopping condition has not been met **do**

3: Set $v_1 = \sum_{i=1}^2 \mathcal{A}_i \mathbf{x}_i^{(k)} - b - \frac{y^{(k)}}{\rho}$.

4: **for** $j \in \{1, \dots, n/2\}$ in parallel **do**

$$x_j^{(k+1)} \leftarrow \min_{x_j} \left\{ f_j(x_j) + \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_1\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\} \quad (64a)$$

5: **end for**

6: Set $v_2 = \mathcal{A}_1 \mathbf{x}_1^{(k+1)} + \mathcal{A}_2 \mathbf{x}_2^{(k)} - b - \frac{y^{(k)}}{\rho}$.

7: **for** $j \in \{n/2 + 1, \dots, n\}$ in parallel **do**

$$x_j^{(k+1)} \leftarrow \min_{x_j} \left\{ f_j(x_j) + \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_2\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\} \quad (64b)$$

8: **end for**

9: Update the dual variables:

$$y^{(k+1)} \leftarrow y^{(k)} - \gamma \rho (Ax^{(k+1)} - b).$$

10: Set $k \leftarrow k + 1$.

11: **end while**

Remark 14. An algorithm similar to Algorithm 8 is presented in [12, 13]. However, Algorithm 8 is more general because the only restriction on the matrices $\{P_j\}_{j=1}^n$ are that they are “positive definite enough”, i.e., they satisfy (63). On the other hand, the algorithm in [12, 13] requires the regularization matrices to take the specific form $c_i \rho A_i^T A_i$, where A_i has full rank and $c_i > n/2$ for all $i = 1, \dots, n$ (assuming that the individual blocks are partitioned evenly into 2 groups). These latter conditions are more restrictive, and also do not necessarily mean that the subproblems arising within their algorithm are easier to solve. For the example of l_1 -minimization subject to equality constraints, the regularization matrices $\{P_j\}_{j=1}^n$ in Algorithm 8, can be chosen to have the form $P_i = \tau_j I - \rho A_i^T A_i$ for some τ_j , which means that subproblems (64a) and (64b) can be solved using soft-thresholding. This is not possible for the algorithm presented in [12, 13]. For further details, see the numerical experiments in Section 4.2.

4 Numerical Experiments

In this section we present numerical experiments to demonstrate the computational performance of F-ADMM (Algorithm 5) and H-ADMM (Algorithm 6), and compare them with J-ADMM [6]. All numerical experiments were conducted using MATLAB on a PC with an Intel i5-3317U, 1.70GHz processor, and 6Gb RAM.

4.1 l_2 -Minimization with Linear Constraints

In this numerical experiment, we consider the problem of determining the solution to an underdetermined system of equations with the smallest 2-norm. Specifically, we aim to solve

$$\underset{x \in \mathbf{R}^N}{\text{minimize}} \quad \frac{1}{2} \|x\|_2^2 \quad \text{subject to} \quad Ax = b. \quad (65)$$

We assume that there are $p = 10$ processors, and then divide the data into $\ell = 10$ groups, each group containing $p = 10$ blocks, with each block of size $N_i = 100$, which results in $N = 10^4$ total variables. We also note that the objective function in (65) is (block) separable and can be written as $f(x) = \sum_{i=1}^n f_i(x_i)$, with $n = 100$ and $x_i \in N_i$ and f_i is strongly convex with convexity parameter $\mu_i = 1$ for all $1 \leq i \leq 100$. The constraint matrix $A \in \mathbf{R}^{m \times N}$ with $m = 3 \cdot 10^3$ is chosen to be sparse, with approximately 20 nonzeros per row, where the nonzeros are taken from a Gaussian distribution. To ensure that $b \in \text{range}(A)$, we randomly generate a vector $z \in \mathbf{R}^N$ with Gaussian entries, and set $b := Az$ so that the constraints in (65) are feasible.

Notice that for problem (65), the subproblem for the j th block of x in H-ADMM can be written as

$$x_j^{(k+1)} \leftarrow \arg \min_{x_j} \left\{ \frac{1}{2} \|x_j\|_2^2 + \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_i\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\} \quad (66)$$

for $j \in \mathcal{S}_i$, where v_i is defined in Step 1 of Algorithm 7. (For F-ADMM the subproblems are solved for all $1 \leq j \leq n$.) Notice that the update $x_j^{(k+1)}$ can be found by solving the system of equations

$$(P_j + I + \rho A_j^T A_j) x_j^{(k+1)} = (P_j + \rho A_j^T A_j) x_j^{(k)} - \rho A_j^T v_i. \quad (67)$$

This linear system motivates us to choose the regularization matrix P_j to be of the form

$$P_j = \tau_j I - \rho A_j^T A_j \quad (68)$$

for some value τ_j , because it may then be combined with (66) to give the simple and inexpensive update

$$x_j^{(k+1)} = \frac{\tau_j}{\tau_j + 1} x_j^{(k)} - \frac{\rho}{\tau_j + 1} A_j^T v_i. \quad (69)$$

For computational reasons, the fraction $\tau_j/(\tau_j + 1)$ should be computed *before* multiplication with $x_j^{(k)}$.

For all of the numerical results reported below, we give the number of epochs required by J-ADMM, F-ADMM, and H-ADMM, and use algorithm parameters $\rho = 0.1$ and $\gamma = 1$. The terminology “epoch” refers to one sweep of the data, i.e. that all n blocks of x are updated once. All reported results are averages over one hundred runs. The stopping condition used in all experiments is $\frac{1}{2} \|Ax - b\|_2^2 \leq 10^{-10}$.

4.1.1 Results using the values of τ that satisfy the theory

In this section, we give the results of our numerical experiments when τ_j that defines P_j in (68) is chosen as dictated by theory. Specifically, we have

- F-ADMM: $\tau_j = \frac{\rho^2}{2\mu} \|A_D^T A_\Delta\|_2^2 + \rho \|A_j\|_2^2$ for all $1 \leq j \leq n$ (see Assumption 4)
- H-ADMM: $\tau_j = \frac{\rho^2}{2\mu} \|\mathcal{A}_D^T \mathcal{A}_\Delta\|_2^2 + \rho \|\mathcal{A}_i\|_2^2$ for all $j \in \mathcal{S}_i$ and $1 \leq i \leq \ell$ (see Assumption 10 and (58))
- J-ADMM: $\tau_j = \rho(n-1) \|A_j\|_2^2$ for all $1 \leq j \leq n$ (see [6])

for the three methods. To get a sense of the size of these choices for τ_j , we plot their magnitudes in Figure 1. The x-axis represents the block number and the y-axis the value of τ_j . For example, a blue point at the value (20, 180) means that $\tau_{20} = 180$. We can clearly see that the τ_j values are much smaller for H-ADMM and F-ADMM, than for J-ADMM. Moreover, the τ_j values used for F-ADMM and H-ADMM are similar in

magnitude. This is, perhaps, a disadvantage since a large value for τ_j translates into stronger regularization in each subproblem (66), which in turn translates into smaller steps and potentially slower convergence. For our test problem (65), this turns out to be the case, as we now discuss.

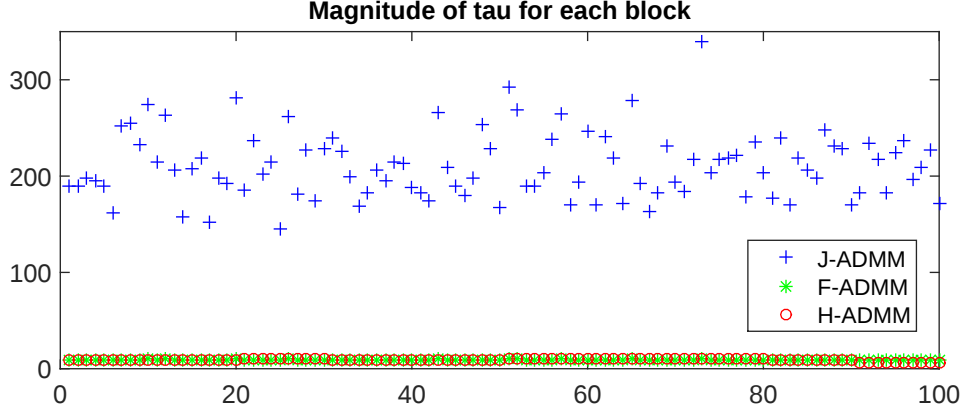


Figure 1: A plot of the magnitude of τ_i for each block $1 \leq i \leq n$ for problem (65).

In Table 1, we present the number of epochs needed by each method (averaged over 100 runs). They show that H-ADMM and F-ADMM require significantly fewer epochs than J-ADMM to determine the solution of problem (65) when the theoretical values of τ_j are chosen. As discussed in the previous paragraph, we can see that the larger values for τ_j needed by J-ADMM lead to poor numerical performance compared with F-ADMM and H-ADMM. However, we remind the reader that H-ADMM and J-ADMM are essentially the same cost per epoch (see Remark 12), while F-ADMM is generally more costly due to its sequential nature.

J-ADMM	H-ADMM	F-ADMM
4358.2	214.1	211.3

Table 1: We present the number of epochs required by J-ADMM, F-ADMM, and H-ADMM for the l_2 -minimization problem (65) using theoretical values of τ_j for $j = 1, \dots, n$.

4.1.2 Results using values for τ obtained by parameter tuning

In [6], it was mentioned that J-ADMM displays better practical performance for smaller values of τ_j than those required by the convergence theory. In this section, we compare the number of epochs required by H-ADMM, F-ADMM, and J-ADMM when τ_j is allowed to be obtained through parameter tuning. In this experiment, for simplicity, we assign the same value τ_j for all blocks $j = 1, \dots, n$. (i.e., $\tau_1 = \tau_2 = \dots = \tau_n$.) Moreover, we picked the starting value to be $\tau_j = \frac{\rho^2}{2} \|A\|^4$ because it approximates the values of τ_j given by theory, in the sense that: $\|A_D^T A_\Delta\|^2 \leq \|A_D\|^2 \|A_\Delta\|^2 \approx \|A\|^2 \|A\|^2$.²

Table 2 presents the number of epochs required by J-ADMM, F-ADMM, and H-ADMM on problem (65) as τ_j varies. For each τ_j we run each algorithm (J-ADMM, F-ADMM and H-ADMM) on 100 random instances of the problem formulation described in Section 4.1. It is clear that all algorithms require fewer epochs to satisfy the stopping tolerance as τ_j decreases. Moreover, for fixed τ_j , F-ADMM and H-ADMM require slightly fewer epochs than J-ADMM. Table 2 also shows that F-ADMM and H-ADMM will converge, in practice, for smaller values of τ_j than J-ADMM. In particular, J-ADMM diverged when we set $\tau_j = 0.2 \cdot \frac{\rho^2}{2} \|A\|^4$, whereas F-ADMM and H-ADMM converged for τ_j as small as $0.1 \cdot \frac{\rho^2}{2} \|A\|^4$; both diverged for $\tau_j = 0.09 \cdot \frac{\rho^2}{2} \|A\|^4$.

²There are many other ways that parameter tuning could be implemented, and we have simply implemented one possibility.

τ_j	J-ADMM	H-ADMM	F-ADMM
$\frac{\rho^2}{2} \ A\ ^4$	530.0	526.3	526.2
$0.6 \cdot \frac{\rho^2}{2} \ A\ ^4$	324.0	320.1	319.9
$0.4 \cdot \frac{\rho^2}{2} \ A\ ^4$	217.7	214.5	214.1
$0.22 \cdot \frac{\rho^2}{2} \ A\ ^4$	123.1	119.3	119.0
$0.2 \cdot \frac{\rho^2}{2} \ A\ ^4$	—	95.8	95.5
$0.1 \cdot \frac{\rho^2}{2} \ A\ ^4$	—	75.3	73.0

Table 2: We present the number of epochs required by J-ADMM, F-ADMM, and H-ADMM for the ℓ_2 -minimization problem (65) for varying values of τ_j . Here, τ_j takes the same value for all blocks $j = 1, \dots, n$.

It is clear that, when the parameter τ_j is tuned, F-ADMM and H-ADMM outperform J-ADMM, when performance is measured in terms of the number of epochs.

4.2 l_1 -Minimization with Linear Constraints

We now consider the problem of l_1 -minimization subject to equality constraints as given by

$$\underset{x}{\text{minimize}} \quad \|x\|_1 \quad \text{subject to} \quad Ax = b, \quad (70)$$

which arises frequently in the compressed sensing and machine learning literature. The one norm promotes sparse solutions, while the linear constraints ensure data fidelity. Note that the one norm is separable.

Problem (70) is convex and not strongly convex, which means that H-ADMM($\ell = 2$) (Algorithm 8) is guaranteed to converge, while convergence for F-ADMM and H-ADMM has not yet been established. Nonetheless, we include them in the numerical experiments to study their practical performance.

For ease of comparison, we follow the experiment setup given in [6]. In particular, suppose that the data is partitioned into $n = 100$ blocks of size $N_i = 10$ for all $1 \leq i \leq n$, so that $N = \sum_{i=1}^n N_i = 1000$. We suppose that $A = [A_1, \dots, A_n]$ is randomly generated with Gaussian entries, and that $A_i \in \mathbf{R}^{m \times N_i}$ for each $1 \leq i \leq n$ and $m = 300$, which means that $A \in \mathbf{R}^{m \times N}$. The sparse signal x^* has $k = 60$ randomly located nonzero entries, the nonzero entries are Gaussian, and the vector b is defined by $b := Ax^*$.

For every algorithm we set $\gamma = 1$ and $\rho = 10/\|b\|_1$. For H-ADMM we let $n = p\ell$ with $p = 4$ and $\ell = 25$, and for H-ADMM($\ell = 2$) we set $\ell = 2$ with both groups containing $p = 50$ blocks. All algorithms were terminated when $\|x - x^*\|_2/\|x^*\|_2 \leq 10^{-10}$. We report on the number of epochs (as in the previous section) and the final constraint residual $\frac{1}{2}\|r\|_2^2$, where $r = Ax - b$ for J-ADMM, F-ADMM, H-ADMM, and H-ADMM($\ell = 2$). All reported results are averages over 100 runs.

For problem (70), the subproblem for the j th block of x in H-ADMM can be written as

$$x_j^{(k+1)} \leftarrow \arg \min_{x_j} \left\{ \|x_j\|_1 + \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_i\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \right\}, \quad (71)$$

for $j \in \mathcal{S}_i$, where v_i is defined in Step 1 of Algorithm 7. (For F-ADMM the subproblems are solved for all

$1 \leq j \leq n$.) Next, using a similar choice for P_j as given by (68), the latter two terms in (71) become

$$\begin{aligned}
& \frac{\rho}{2} \|A_j(x_j - x_j^{(k)}) + v_i\|_2^2 + \frac{1}{2} \|x_j - x_j^{(k)}\|_{P_j}^2 \\
&= \frac{\rho}{2} (x_j - x_j^{(k)})^T A_j^T A_j (x_j - x_j^{(k)}) + \rho (x_j - x_j^{(k)})^T A_j^T v_i + \frac{1}{2} (x_j - x_j^{(k)})^T P_j (x_j - x_j^{(k)}) + \frac{\rho}{2} \|v_i\|_2^2 \\
&= \frac{1}{2} (x_j - x_j^{(k)})^T (P_j + \rho A_j^T A_j) (x_j - x_j^{(k)}) + \rho (x_j - x_j^{(k)})^T A_j^T v_i + \frac{\rho}{2} \|v_i\|_2^2 \\
&= \frac{\tau_j}{2} (x_j - x_j^{(k)})^T (x_j - x_j^{(k)}) + \rho (x_j - x_j^{(k)})^T A_j^T v_i + \frac{\rho}{2} \|v_i\|_2^2 \\
&= \tau_j \left[\frac{1}{2} \|x_j - d_j\|_2^2 - \frac{1}{2} \|x_j^{(k)} - d_j\|_2^2 + \frac{\rho}{2\tau_j} \|v_j\|_2^2 \right],
\end{aligned}$$

where we have defined $d_j := x_j^{(k)} - \frac{\rho}{\tau_j} A_j^T v_i$ to derive the last equality. Using the previous equality, the solution to subproblem (71) is the same as that given by

$$x_j^{(k+1)} \leftarrow \arg \min_{x_j} \left\{ \frac{1}{\tau_j} \|x_j\|_1 + \frac{1}{2} \|x_j - d_j\|_2^2 \right\}, \quad (72)$$

which is separable, so that soft thresholding can be used to solve for $x_j^{(k+1)}$.

4.2.1 Results using the values of τ that satisfy the theory

Here we present the results of the above stated experiment setup when the values of τ_j required by the theory are used. We recall that the convergence theory for F-ADMM and H-ADMM has not been established in the convex case, so here we simply use the values of τ_j that are needed in the strongly convex case. We also recall that convergence of H-ADMM($\ell = 2$) is guaranteed in the convex case (see Section 3.4). Thus, in addition to the τ_j values for F-ADMM, H-ADMM, and J-ADMM given in Section 4.1.1, we also use

- H-ADMM($\ell = 2$): $\tau_j = \rho \|A_i\|_2^2$ for all $j \in \mathcal{S}_i$ and $1 \leq i \leq 2$ (see (62)).

Figure 2 shows typical τ_j values for each algorithm for the l_1 -minimization experiment. (For the meaning of each plotted point, see Section 4.1.1.) As it was for the l_2 -minimization problem (65), we again see that the τ_j values are much smaller for F-ADMM and H-ADMM, when compared to J-ADMM. In addition, the τ_j values for H-ADMM($\ell = 2$) are the smallest overall. Consequently, the regularization matrices used in H-ADMM($\ell = 2$) are significantly less positive definite than all other algorithms, and the regularization matrices for F-ADMM and H-ADMM are less positive definite than for J-ADMM.

In Table 3 we give the number of epochs required by each algorithm, as well as the final constraint residual $\frac{1}{2} \|r\|_2^2$, where $r = Ax - b$. Table 3 shows that H-ADMM($\ell = 2$) requires far fewer epochs than the other algorithm, with F-ADMM and H-ADMM requiring about one-sixth the number of epochs compared with J-ADMM, for the τ_j values stated above. Although there is no convergence theory for F-ADMM and H-ADMM, they both converge in practice for this setup, and are very competitive with J-ADMM.

J-ADMM		H-ADMM($\ell = 2$)		H-ADMM		F-ADMM	
Epochs	$\frac{1}{2} \ r\ _2^2$	Epochs	$\frac{1}{2} \ r\ _2^2$	Epochs	$\frac{1}{2} \ r\ _2^2$	Epochs	$\frac{1}{2} \ r\ _2^2$
12610.1	0.27e-16	605.8	0.19e-16	1882.1	0.24e-16	1879.4	0.24e-16

Table 3: We present the number of epochs required and final constraint violation by J-ADMM, F-ADMM, H-ADMM, and H-ADMM($\ell = 2$) for the l_1 -minimization problem (70) for varying values of τ_j .

4.2.2 Results using values for τ obtained by parameter tuning

While theory dictates the values of τ_j needed to guarantee convergence, experimental performance can often be improved by selecting better parameter values. In this section, we compare the performance of the

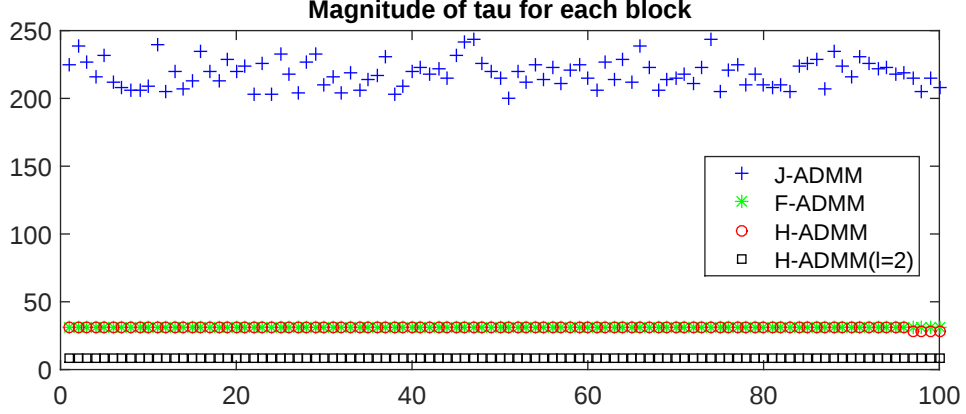


Figure 2: A plot of the magnitude of τ_i for each block $1 \leq i \leq n$ for problem (70).

algorithms from the previous section using smaller values of τ_j than those used in Section 4.2.1. We repeat the experiments described in Section 4.2, but now use the same value τ_j for all blocks $j = 1, \dots, n$ and for all algorithms. The results are presented in Table 4.

Table 4 shows that for the l_1 -minimization problem, the number of epochs needed by each of the algorithms to reach the stopping tolerance decreases as τ_j decreases. Also, for each fixed τ_j , J-ADMM requires the most epochs followed by H-ADMM($\ell = 2$) and H-ADMM, while F-ADMM requires the smallest number of epochs. This makes intuitive sense because, during every epoch, F-ADMM incorporates new information after every block has been updated, H-ADMM incorporates new information after every group of $p = 4$ blocks have been updated, H-ADMM($\ell = 2$) only incorporates new information after half of the blocks have been updated ($p = n/2 = 50$), while J-ADMM does not use any updated information within each epoch.

τ_j	J-ADMM		H-ADMM($\ell = 2$)		H-ADMM		F-ADMM	
	Epochs	$\frac{1}{2}\ r\ _2^2$	Epochs	$\frac{1}{2}\ r\ _2^2$	Epochs	$\frac{1}{2}\ r\ _2^2$	Epochs	$\frac{1}{2}\ r\ _2^2$
$0.2 \cdot \frac{\rho^2}{2}\ A\ ^4$	1078.3	0.23e-16	1066.2	0.23e-16	1051.3	0.21e-16	1053.3	0.21e-16
$0.1 \cdot \frac{\rho^2}{2}\ A\ ^4$	600.3	0.19e-16	581.5	0.19e-16	570.0	0.20e-16	567.8	0.22e-16
$0.05 \cdot \frac{\rho^2}{2}\ A\ ^4$	344.4	0.19e-16	328.8	0.20e-16	325.9	0.21e-16	326.0	0.21e-16
$0.03 \cdot \frac{\rho^2}{2}\ A\ ^4$	—	inf	—	inf	246.3	0.16e-16	244.4	0.15e-16
$0.02 \cdot \frac{\rho^2}{2}\ A\ ^4$	—	inf	—	inf	162.2	0.34e-16	150.3	0.29e-16

Table 4: We present the number of epochs required and final constraint violation by J-ADMM, F-ADMM, H-ADMM, and H-ADMM($\ell = 2$) for the l_1 -minimization problem (70) for varying values of τ_j .

Next, we can also see that J-ADMM and H-ADMM($\ell = 2$) perform well until $\tau_j = 0.05 \frac{\rho^2}{2}\|A\|^4$. However, for smaller values of τ_j , J-ADMM and H-ADMM($\ell = 2$) diverged. On the other hand, F-ADMM and H-ADMM still converge (in practice) for $\tau_j = 0.02 \frac{\rho^2}{2}\|A\|^4$, but diverged when $\tau = 0.019 \frac{\rho^2}{2}\|A\|^4$. Thus, we can conclude that if the parameter τ_j is hand-tuned for each algorithm, then practical performance is greatly improved for all methods, and that both F-ADMM and H-ADMM perform the best *in practice* on this convex optimization problem.

Remark 15. An adaptive parameter tuning scheme is presented in [6, Section 2.3], which ensures that the convergence theory developed for J-ADMM still holds, i.e., convergence of J-ADMM is guaranteed if their adaptive parameter tuning scheme is followed. Unfortunately, we were unable to replicate the numerical results presented in that paper, because there was not enough information regarding the tuning parameters

that they used. However, we implemented the adaptive parameter tuning scheme for J-ADMM using the following parameters: $\eta = 0.1$, $\alpha_i = 1.1$, $\beta_i = 0.1$, $Q_i = I$, for all $i = 1, \dots, n$, and on average over 100 runs on the l_1 -minimization experiment, J-ADMM required 442.6 epochs. This is more than the ≈ 220 epochs reported in that paper. In either case, by hand tuning τ_j , F-ADMM, H-ADMM, and H-ADMM($\ell = 2$) all outperform J-ADMM.

Remark 16. Following the same ideas as in [6, Section 2.3], it may be possible to develop adaptive parameter updating schemes for F-ADMM and H-ADMM that still ensure convergence. In this way, it may be possible to achieve additional computational gains for both of them.

5 Conclusion

We presented an algorithm for minimizing block-separable strongly convex objective functions subject to linear equality constraints. Our method, called F-ADMM, may be viewed as a flexible version of the popular ADMM algorithm. In particular, F-ADMM is provably convergent for any number of blocks, and contains popular methods such as ADMM, J-ADMM, and G-ADMM as special cases.

Our work was motivated by big data applications. We showed, via numerical experiments, that F-ADMM is especially effective when the number of blocks is larger than the number of available machines. In this case, unlike Jacobi methods, our method allows for updated variables to be used when updating the blocks within subsequent groups, all while maintaining essentially the same cost of a fully Jacobi method. Our numerical experiments indicate that this approach is more efficient and stable than the fully Jacobi method.

6 Acknowledgements

The authors would like to thank Professor Wotao Yin and Mr Damek Davis for their helpful discussion and feedback on an earlier version of this work.

References

- [1] Dimitri Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996.
- [2] Daniel Boley. Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs. *SIAM Journal on Optimization*, 23(4):2183–2207, November 2013.
- [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, 2010.
- [4] Xingju Caia, Deren Han, and Xiaoming Yuan. The direct extension of ADMM for three-block separable convex minimization models is convergent when one function is strongly convex. Technical report, School of Mathematical Sciences, Nanjing Normal University, and Department of Mathematics, Hong Kong Baptist University, Nanjing 210023, P.R. China and Hong Kong, P.R. China, November 2014.
- [5] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Technical report, Department of Management Science and Engineering, Stanford University, Huang Engineering Center 308, 475 Via Ortega, CA 94305-4121, October 2013. To appear in *Mathematical Programming*.
- [6] Wei Deng, Ming-Jun Lai, Zhimin Peng, and Wotao Yin. Parallel multi-block ADMM with $o(1/k)$ convergence. Technical report, Department of Mathematics, UCLA, Los Angeles, CA 90095-1555, USA, March 2014.

- [7] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. Report 12–14, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005-1892, 2012.
- [8] Jonathan Eckstein. Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. Technical Report Report RRR 32-2012, Center for Operations Research, Rutgers University, 640 Bartholomew Road, Piscataway, New Jersey, December 2012.
- [9] Jonathan Eckstein and Dimitri P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.
- [10] Xiaoling Fu, Bingsheng He, Xiangfeng Wang, and Xiaoming Yuan. Block-wise alternating direction method of multipliers with Gaussian back substitution for multiple-block convex programming. Technical report, Nanyang Technological University; East China Normal University; Hong Kong Baptist University, September 2014.
- [11] Deren Han and Xiaoming Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.
- [12] Bingsheng He, Minghua Xu, and Xiaoming Yuan. Block-wise ADMM with a relaxation factor for multiple-block convex programming. Technical report, Nanyang Technological University; Hong Kong Baptist University, August 2014.
- [13] Bingsheng He and Xiaoming Yuan. Block-wise alternating direction method of multipliers for multiple-block convex programming and beyond. Technical report, Nanyang Technological University; Hong Kong Baptist University, August 2014.
- [14] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. Technical report, Department of Electrical and Computer Engineering, University of Minnesota, 200 Union ST SE, Minneapolis, MN, 55455, March 2012.
- [15] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. Technical report, Department of Industrial and Manufacturing Systems Engineering, Iowa State University; The Chinese University of Hong Kong; Department of Electrical Engineering, Stanford University, Ames, IA 50011, USA; Shenzhen, China; 350 Serra Mall, Stanford, CA 94305, October 2014.
- [16] John M. Mulvey and Andrzej Ruszczyński. A diagonal quadratic approximation method for large scale linear programs. *Operations Research Letters*, 12:205–215, 1992.
- [17] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
- [18] R. Tyrrell Rockafellar and Roger J-B. Wets. *Variational Analysis*. Springer-Verlag, 3 edition, 2009.
- [19] Andrzej Ruszczyński. On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. *Mathematics of Operations Research*, 20(3):634–656, 1995.
- [20] Rachael Tappenden, Peter Richtárik, and Burak Büke. Separable approximations and decomposition methods for the augmented Lagrangian. *Optimization Methods and Software*, 2014. Published online: 06 November 2014.
- [21] Huahua Wang, Arindam Banerjee, and Zhi-Quan Luo. Parallel direction method of multipliers. Technical report, Department of Computer Science, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455, September 2014.

- [22] Junfeng Yang and Yin Zhang. Alternating direction algorithms for ℓ_1 -problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.
- [23] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. Technical report, Department of Mathematics, Hong Kong Baptist University, Hong Kong, China, November 2009.
- [24] Victor M Zavala. Stochastic optimal control model for natural gas network operations. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2013.