Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 4, pp. 645–667, October 2015

SOLVING MULTIPLE-BLOCK SEPARABLE CONVEX MINIMIZATION PROBLEMS USING TWO-BLOCK ALTERNATING DIRECTION METHOD OF MULTIPLIERS

Xiangfeng Wang*, Mingyi Hong, Shiqian Ma and Zhi-Quan Luo

Abstract: In this paper, we consider solving multiple-block separable convex minimization problems using alternating direction method of multipliers (ADMM). Motivated by the fact that the existing convergence theory for ADMM is mostly limited to the two-block case, we analyze in this paper, both theoretically and numerically, a new strategy that first transforms a multi-block problem into an equivalent two-block problem (either in the primal domain or in the dual domain) and then solves it using the standard two-block ADMM. In particular, we derive convergence results for this two-block ADMM approach to solve multi-block separable convex minimization problems, including an improved $\mathcal{O}(1/\epsilon)$ iteration complexity result. Moreover, we compare the numerical efficiency of this approach with the standard multi-block ADMM on several separable convex minimization problems which include basis pursuit, robust principal component analysis and latent variable Gaussian graphical model selection. The numerical results show that the multiple-block ADMM, although lacks theoretical convergence guarantees, typically outperforms two-block ADMMs.

Key words: *alternating direction method of multipliers, primal splitting, dual splitting*

Mathematics Subject Classification: *65K05, 65K15, 90C30, 90C33*

1 Introduction

In this paper, we consider the following convex optimization problem with m block variables and the objective being the sum of m ($m \geq 2$) separable convex functions:

$$\begin{aligned} \min_{\{x_1,x_2,\ldots,x_m\}} \ & \sum_{i=1}^{m} f_i(x_i), \\ \text{s.t.} \ & \sum_{i=1}^{m} A_i x_i = b, x_i \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}, i = 1,\ldots,m, \end{aligned} \tag{1.1}$$

where $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is a proper closed convex function (not necessarily smooth), $\mathcal{X}_i$ is a closed convex set and $A_i \in \mathbb{R}^{\ell \times n_i}$, $b \in \mathbb{R}^{\ell}$, and $\boldsymbol{x} = (x_1,\ldots,x_m) \in \prod_i \mathcal{X}_i$ is a partition of the decision variable.

The multi-block convex optimization problem (1.1) arises naturally in many practical applications. For example, consider the Robust Principal Component Analysis (RPCA)

*Xiangfeng Wang was supported by the Chinese Scholarship Council during his visit to University of Minnesota, and the Shanghai Funding 15YF1403400.

problem [3], whereby the goal is to recover the low-rank and sparse components of a given matrix $M \in \mathbb{R}^{\ell \times n}$ by solving the following nonsmooth convex optimization problem

$$\begin{aligned} \min_{L,S,Z} \ & \|L\|_* + \tau \|S\|_1, \\ \text{s.t.} \ & L + S + Z = M, \\ & Z \in \mathcal{H} := \{Z \in \mathbb{R}^{\ell \times n} \mid \|P_\Omega(Z)\|_F \leq \delta\}. \end{aligned} \tag{1.2}$$

In the above formulation, $L \in \mathbb{R}^{\ell \times n}$ and $S \in \mathbb{R}^{\ell \times n}$ are respectively the low-rank and the sparse components of the matrix M, while Z represents the observation noise. The notation $P_\Omega(\cdot)$ signifies the entry-wise projection operator for a given index set Ω:

$$P_\Omega(Z) = \begin{cases} Z_{ij}, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise,} \end{cases}$$

while $\|\cdot\|_*$, $\|\cdot\|_1$ and $\|\cdot\|_F$ denote respectively the matrix nuclear norm (i.e., the sum of the matrix singular eigenvalues), the L_1 and Frobenius norm of a matrix. Clearly problem (1.2) corresponds to the case of $m = 3$ in problem (1.1), with $x_1 = L$, $x_2 = S$, $x_3 = Z$ and

$$f_1(L) := \|L\|_*, \quad f_2(S) := \tau \|S\|_1, \quad f_3(Z) := \mathcal{I}_{\mathcal{H}}(Z),$$

where $\mathcal{I}_{\mathcal{H}}(\cdot)$ denotes the indicator function for the set $\mathcal{H}$.

Similarly, the so-called latent variable Gaussian graphical model selection (LVGGMS) problem [4], which is closely related to the inverse covariance matrix estimation problem, is also in the form of (1.1). In particular, suppose (X, Y) is a pair of $(p + r)$-dimensional joint multivariate Gaussian random variables, with covariance matrix denoted by $\Sigma_{(X,Y)} := [\Sigma_X, \Sigma_{XY}; \Sigma_{YX}, \Sigma_Y]$ and its inverse $\Theta_{(X,Y)} := [\Theta_X, \Theta_{XY}; \Theta_{YX}, \Theta_Y]$ respectively. The random variable $X := (X_1, X_2, \ldots, X_p)^T$ is observable while $Y := (Y_1, Y_2, \ldots, Y_r)^T$ is the latent (or hidden) random variable. In many applications, we typically have $r \ll p$. Moreover, the marginal distribution of the observed variables X usually follows a sparse graphical model and hence its concentration matrix Θ_X is sparse. Notice that the inverse of the covariance matrix for X can be expressed as

$$\Sigma_X^{-1} = \Theta_X - \Theta_{XY} \Theta_Y^{-1} \Theta_{YX}, \tag{1.3}$$

which is the difference between the sparse term Θ_X and the low-rank term $\Theta_{XY}\Theta_Y^{-1}\Theta_{YX}$ (since r is much less than p). Thus, the task of estimating the sparse marginal concentration matrix Θ_X can be accomplished by solving the following regularized maximum likelihood problem

$$\begin{aligned} \min_{S,L} \ & \langle S - L, \hat{\Sigma}_X \rangle - \operatorname{logdet}(S - L) + \alpha_1 \|S\|_1 + \alpha_2 \mathbf{Tr}(L), \\ \text{s.t.} \ & S - L \succ 0, \ L \succeq 0, \end{aligned} \tag{1.4}$$

where $\hat{\Sigma}_X \in \mathbb{R}^{p \times p}$ is the sample covariance matrix of $X \in \mathbb{R}^{n \times p}$ and $\mathbf{Tr}(L)$ denotes the trace of matrix $L \in \mathbb{R}^{p \times p}$, while $S \in \mathbb{R}^{p \times p}$. Evidently, we can rewrite (1.4) in the following equivalent form by introducing a new variable $R \in \mathbb{R}^{p \times p}$,

$$\begin{aligned} \min_{S,L} \ & \langle R, \hat{\Sigma}_X \rangle - \operatorname{logdet}(R) + \alpha_1 \|S\|_1 + \alpha_2 \mathbf{Tr}(L) + \mathcal{I}(L \succeq 0), \\ \text{s.t.} \ & R = S - L, \end{aligned} \tag{1.5}$$

where the constraint $R \succ 0$ is implicitly imposed by having the term $-\text{logdet}(R)$ in the objective function. It is easily seen that LVGGMS corresponds to the three block case $m = 3$ in problem (1.1) with $x = (R, S, L)$ and

$$f_1(R) := \langle R, \hat{\Sigma}_X \rangle - \text{logdet}(R),\ f_2(S) := \alpha_1 \|S\|_1,\ f_3(L) := \alpha_2 \mathbf{Tr}(L) + \mathcal{I}(L \succeq 0),$$

where the linear constraint is $R - S + L = 0$.

Problem (1.1) is a structured convex problem with a separable objective function and a single linear equality constraint. A popular algorithm for solving this class of problem is the so-called *Alternating Direction Method of Multipliers* (ADMM). To outline the basic steps of the ADMM, we first introduce the augmented Lagrangian function for problem (1.1)

$$L_\beta(x_1, \ldots, x_m, \lambda) = \sum_{i=1}^{m} f_i(x_i) - \langle \lambda, \sum_{i=1}^{m} A_i x_i - b \rangle + \frac{\beta}{2} \left\| \sum_{i=1}^{m} A_i x_i - b \right\|^2, \tag{1.6}$$

where β is the penalty parameter for the violation of the linear constraint and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. The ADMM method is a Gauss-Seidel iteration scheme in which the primal block variables $\{x_i\}$ and the Lagrangian multiplier λ for the equality constraint are updated sequentially. Specifically, for a fixed penalty coefficient $\beta > 0$, the ADMM for solving problem (1.1) can be described as follows:

Algorithm 1: Alternating Direction Method of Multipliers for (1.1)

Initialize $x_2^0, \ldots, x_m^0, \lambda^0$, and β.
For $k = 1, 2, \ldots,$ do
- Compute x_i^{k+1}, $\forall\ i = 1, \ldots, m$,

$$x_i^{k+1} = \arg\min_{x_i \in \mathcal{X}_i} f_i(x_i) + \tfrac{\beta}{2} \left\| \sum_{j=1}^{i-1} A_j x_j^{k+1} + A_i x_i + \sum_{j=i+1}^{m} A_j x_j^k - b - \tfrac{\lambda^k}{\beta} \right\|^2,$$

- Compute λ^{k+1},

$$\lambda^{k+1} = \lambda^k - \beta \left(\sum_{i=1}^{m} A_i x_i^{k+1} - b \right).$$

The history of ADMM dates back to 1970s in [20, 21] where the method was first developed for solving 2-block separable convex problems. In [19], it is shown that ADMM can be interpreted as a special case of an operator splitting method called *Douglas-Rachford Splitting Method* (DRSM) for finding a zero of the sum of two monotone operators $\mathcal{A}$ and $\mathcal{B}$ [13, 34]. Moreover, ADMM can also be related to Spingarn's method called *Partial Inverse* [43, 44]. Recently, ADMM has found its application in solving a wide range of large-scale problems from statistics, compressive sensing, image and video restoration, and machine learning, see e.g., [2, 5, 24, 31, 37, 38, 47–49] and the references therein.

The ADMM convergence has long been established in the literature for the case of two block variables (i.e. $m = 2$ in problem (1.1)). References [20, 21] show that the algorithm converges globally when each subproblem is solved exactly. Such convergence results have also been obtained in the context of DRSM. In [16], the authors show that DRSM is a special case of the so-called *Proximal Point Algorithm* (PPA), for which the global convergence and the rate of convergence have been established (see [40]). Accordingly, under certain regularity conditions, the global convergence and the rate of convergence of DRSM (and hence ADMM) follow directly. On the other hand, for large scale problems such as those arising from compressive sensing, the global optimal solution for subproblems related to

certain block variables may not be easily computable [48]. In these cases the classical ADMM needs to be modified accordingly so that those difficult subproblems are solved *inexactly*. In [14,16,28,46,48,50], the authors show that by performing a simple proximal gradient step for each subproblem, global convergence results similar to those for the classical ADMM can also be obtained. Beyond global convergence, there are a few results characterizing the iteration complexity and the convergence rate of the ADMM. The authors of [32] have shown that to obtain an ϵ-optimal solution, the worst-case iteration complexity of both exact and inexact ADMM is $\mathcal{O}(1/\epsilon)$, where the ϵ optimality is defined using both the constraint and objective violation. In [40], Rockafellar has shown that if the inverse of the considered operator is Lipschitz continuous at origin point, the PPA converges linearly when the resolvent operator is solved either exactly or inexactly. Therefore the linear convergence of DRSM and ADMM follow directly under some assumptions on $\mathcal{A}$ and $\mathcal{B}$ in DRSM or $\{f_i\}$ and $\{A_i\}$ in ADMM. In [34], Lions and Mercier have proved that when operator $\mathcal{B}$ is both coercive and Lipschitz, then DRSM converges linearly. Further in [17], Eckstein and Bertsekas have shown the linear convergence rate of ADMM for linear programming. More recently the authors of [26] prove the local linear convergence of ADMM for quadratic programs without any further condition, and the authors of [12] show that for both exact version and inexact version involving a proximal term, ADMM converges linearly if the objective function is strongly convex and its gradient is Lipschitz continuous in at least one block variable, and that the matrices $\{A_i\}$ satisfy certain rank assumptions.

However, when the number of block variables is greater than two, the convergence of ADMM has not been well understood. Recently, the authors of [**?**] show the convergence of ADMM for multi-block separable convex minimization problems with the assumption of strongly convex. Further the authors of [33] prove the global (linear) convergence of the ADMM for the multiple-block problem (1.1), under the following assumptions: *a)* for each i, A_i is full column rank, *b)* β is sufficiently small and *c)* certain error bounds hold for the problem (1.1). The full column rank condition a) can be dropped if each subproblem is solved inexactly. However when conditions *b)* and *c)* are not satisfied, even the global convergence of the algorithm is still open in the literature. As a result, a number of variants of the classical ADMM have been proposed; see [27, 29, 30, 35]. For example, in [29, 30], by adding an additional Gaussian back substitution correction step in each iteration after all the block variables are updated, the authors establish the global convergence, iteration complexity and linear convergence rate (under some technical assumption on the iterates) of the modified algorithm. However such correction step is not always computationally efficient, especially when A_i's are not efficiently invertible. In [35], an alternating proximal gradient method is proposed, in which the proximal version of the ADMM is applied to problem (1.1), after grouping the m block variables into two blocks. However, the way that the block variables should be grouped is highly problem dependent. There is no general criterion guiding how such step should be done. Also recently, some multiple block splitting algorithms have been proposed for solving some models similar to (1.1) in [18, 23].

In this work, we systematically study ADMM algorithms for solving the multi-block problem (1.1). We first propose two novel algorithms that apply the two-block ADMM to certain reformulation of the original multi-block problem. We show in detail how these algorithms are derived and analyze their convergence properties. We then report numerical results comparing the original multi-block ADMM with the proposed approaches on problems with multiple block structures such as the basis pursuit problem, robust PCA and latent variable Gaussian graphical model selection. Our numerical experiments show that the multi-block ADMM performs much better than the two-block ADMM algorithms as well as many other existing algorithms.

Notation: Let $\|x\|_1 = \sum_{i=1}^n |x_i|$ and $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ denote the usual vector ℓ_1-norm and ℓ_2-norm respectively, where $x \in \mathbb{R}^n$ is a real vector. Let $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ denote the $n \times n$ identity matrix. For a matrix X, let $\rho(X)$ denote its spectral radius. Throughout the paper we assume that the following *proximity operator* is easy to compute:

$$\mathrm{Prox}_g^\tau(c) := \arg\min_{x\in\mathcal{X}} \ \tau g(x) + \frac{1}{2}\|x-c\|^2, \tag{1.7}$$

where $\tau > 0$ and c are given. For instance when $g(x) = \|x\|_1$,

$$\left(\mathrm{Prox}_{\|\cdot\|_1}^\tau(c)\right)_i = (\mathbf{shrinkage}(c,\tau))_i := \mathrm{sgn}(c_i)\cdot\max(|c_i|-\tau, 0). \tag{1.8}$$

We refer the readers to [10,35] for other easily computable proximity operators. The *conjugate function* of $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$f^*(y) := \sup_{x\in\mathbf{dom} f} (y^T x - f(x)). \tag{1.9}$$

Let $\mathcal{T}$ be a set-valued operator, c is any positive scalar, and denote the following operator

$$\mathcal{J}_{c\mathcal{T}} := (\mathcal{I} + c\mathcal{T})^{-1}, \tag{1.10}$$

as the *resolvent* of $\mathcal{T}$.

The rest of this paper is organized as follows: the primal and dual splitting ADMM algorithms are presented and analyzed in Section 2.1 and Section 2.2, respectively. In Section 3, numerical results for both synthetic and real problems are reported. Finally, some concluding remarks are given in Section **??**.

2 The Proposed Algorithms

Various modifications of the ADMM algorithm have been proposed in the literature to deal with the multi-block problem (1.1). Instead of proposing yet another ADMM variant, we propose to transform any multi-block problem into an equivalent two-block problem to which the classical two-block ADMM can be readily applied. The main idea is to appropriately introduce some auxiliary variables so that the original variables are completely decoupled from each other. In this section, this technique will be explained in details for both the primal and dual versions of problem (1.1).

2.1 Primal Splitting ADMM

Introducing a set of auxiliary variables $\{y_i\}_{i=1}^m$ with $y_i = A_i x_i - \frac{b}{m} \in \mathbb{R}^\ell$ for all i, problem (1.1) can be expressed in the following equivalent form

$$\begin{aligned} \min_{\{x_i\},\{y_i\}} \ & \sum_{i=1}^m f_i(x_i), \\ \text{s.t.} \ & A_i x_i - \frac{b}{m} = y_i, \ x_i \in \mathcal{X}_i, \ i = 1,\ldots,m, \\ & \sum_{i=1}^m y_i = 0, \ y_i \in \mathbb{R}^\ell. \end{aligned} \tag{2.1}$$

Problems (1.1) and (2.1) are equivalent in the sense that they share the same primal optimal solution set for the variables $\{x_i\}_{i=1}^m$, and achieve the same global optimal objective value. To apply the ADMM algorithm to the above reformulated problem, we write its (partial) augmented Lagrangian function, by keeping the constraint $\sum_{i=1}^m y_i = 0$ and penalizing the rest of the constraints:

$$L_\beta(\{x_i\},\{y_i\},\{\lambda_i\}) = \sum_{i=1}^m f_i(x_i) - \sum_{i=1}^m \langle \lambda_i, A_i x_i - \frac{b}{m} - y_i\rangle + \frac{\beta}{2}\sum_{i=1}^m \left\| A_i x_i - \frac{b}{m} - y_i \right\|^2, \quad (2.2)$$

where $\lambda_i \in \mathbb{R}^\ell$ is the Lagrangian multiplier and β is a penalty parameter. Denote $\boldsymbol{\lambda} := (\lambda_1^T, \ldots, \lambda_m^T)^T (\in \Lambda := \mathbb{R}^\ell \times \cdots \times \mathbb{R}^\ell)$. Obviously, in (2.2), all the x_i's are separable with each other, so are all te y_i's. It is then natural to take $\boldsymbol{x} := (x_1^T, \ldots, x_m^T)^T (\in \mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_m)$ and $\boldsymbol{y} := (y_1^T \ldots, y_m^T)^T (\in \mathcal{Y} := \{y \in \mathbb{R}^\ell \times \cdots \times \mathbb{R}^\ell \mid \sum_{i=1}^m y_i = 0\})$ as two block variables, and use the classical two-block ADMM to solve (2.1). The primal splitting ADMM is stated in the following table:

Algorithm 2: Primal Splitting ADMM for (2.1)

Initialize $\{x_1^0, \ldots, x_m^0\}, \{\lambda_1^0, \ldots, \lambda_m^0\}$, and β.
For $k = 0, 1, 2, \ldots,$ do

- Compute $\{y_1^{k+1}, \ldots, y_m^{k+1}\}$,
$$\boldsymbol{y}^{k+1} = \arg\min_{\boldsymbol{y}} \ \frac{\beta}{2}\sum_{i=1}^m \left\| A_i x_i^k - \frac{b}{m} - y_i - \frac{\lambda_i^k}{\beta}\right\|^2, \ \text{s.t.} \ \sum_{i=1}^m y_i = 0,$$
- Compute $x_i^{k+1}, \forall i = 1, \ldots, m$,
$$x_i^{k+1} = \arg\min_{x_i \in \mathcal{X}_i} \ f_i(x_i) + \frac{\beta}{2}\left\| A_i x_i - \frac{b}{m} - y_i^{k+1} - \frac{\lambda_i^k}{\beta}\right\|^2,$$
- Compute $\lambda_i^{k+1}, \forall i = 1, \ldots, m$,
$$\lambda_i^{k+1} = \lambda_i^k - \beta\left(A_i x_i^{k+1} - \frac{b}{m} - y_i^{k+1}\right).$$

We note that the subproblem for $\boldsymbol{y}$ is a projection onto the hyperplane $\sum_i y_i = 0$. As such, it admits the following closed-form solution

$$y_i^{k+1} = -\frac{1}{m}\left\{\sum_{i=1}^m A_i x_i^k - \frac{b}{m} - \frac{\lambda_i^k}{\beta}\right\} + \left(A_i x_i^k - \frac{b}{m} - \frac{\lambda_i^k}{\beta}\right), \ i = 1, \ldots, m. \quad (2.3)$$

Further, it is easy to see that the subproblem for x_i can be solved efficiently by using (1.7), provided that A_i is an identity matrix (or any constant multiple of it). Else, x_i can be updated by simply using a proximal gradient step:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i \in \mathcal{X}_i} f_i(x_i) + \left\langle \beta A_i^T(A_i x_i^k - \frac{b}{m} - y_i^{k+1} - \frac{\lambda_i^k}{\beta}), x_i - x_i^k\right\rangle + \frac{\tau_i}{2}\left\|x_i - x_i^k\right\|^2, \\ &= \arg\min_{x_i \in \mathcal{X}_i} f_i(x_i) + \frac{\tau_i}{2}\left\| x_i - x_i^k + \frac{\beta A_i^T(A_i x_i^k - \frac{b}{m} - y_i^{k+1} - \frac{\lambda_i^k}{\beta})}{\tau_i}\right\|^2, \\ &= \mathrm{Prox}_{f_i}^{\frac{1}{\tau_i}}\left(x_i^k - \frac{\beta A_i^T(A_i x_i^k - \frac{b}{m} - y_i^{k+1} - \frac{\lambda_i^k}{\beta})}{\tau_i}\right), \end{aligned} \quad (2.4)$$

where τ_i denotes the penalty parameter for the distance between x_i^{k+1} and x_i^k.

Next we discuss the convergence of the above primal splitting ADMM algorithm.

Theorem 2.1. *For any $\beta > 0$, suppose the subproblem for x_i is either exactly solved, or is solved inexactly using* (2.4) *with $\tau_i > \beta \cdot \rho(A_i^T A_i)$. Let $(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k)$ be any sequence generated by Algorithm 2. Then starting with any initial point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{\lambda}^0) \in \mathcal{X} \times \mathcal{Y} \times \Lambda$, we have*

1. *The sequence $\{\boldsymbol{\lambda}^k\}$ converges to $\boldsymbol{\lambda}^*$, where $\boldsymbol{\lambda}^*$ is the dual optimal solution for problem* (2.1)*;*
2. *The sequence $\{\sum_{i=1}^m f_i(x_i^k)\}$ converges to p^*, where p^* is the primal optimal value for problem* (2.1)*;*
3. *The residual sequence $\{A_i x_i^k - y_i^k - \frac{b}{m}\}$ converges to 0 for each $i = 1, \ldots, m$;*
4. *If the subproblem for x_i is exactly solved for each i, then the sequence $\{A_i x_i^k\}$ and $\{y_i^k\}$ converge. Moreover, if A_i has full column rank, then $\{x_i^k\}$ converges. When the subproblem for x_i is solved inexactly using* (2.4) *with $\tau_i > \beta \cdot \rho(A_i^T A_i)$, the sequence $\{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}$ converges to an optimal solution to problem* (2.1).

Proof. When the subproblems are solved exactly, we are actually using the classical two-block ADMM to solve the equivalent formulation (2.1). As a result, the first three conclusions as well as the convergence of $\{A_i x_i^k\}$ and $\{y_i^k\}$ follow directly from the classical analysis of the two-block ADMM (see, e.g., [2, Section 3.2]). The convergence of $\{x_i^k\}$ is a straightforward consequence of the convergence of $\{A_i x_i^k\}$ and the assumption that A_i's are all full column rank. When the subproblems are solved inexactly via (2.4), because $\tau_i > \beta \cdot \rho(A_i^T A_i)$, all the conclusions are implied by the result in [28, Theorem 1]. □

Besides global convergence, we can elaborate on other convergence properties of **Algorithm 2**. First, for both the exact case and the inexact proximal case with $\tau_i > \beta \cdot \rho(A_i^T A_i)$, we can obtain the following iteration complexity result by adopting the variational inequality framework developed in [32, Theorem 4.1]. To illustrate, let $\boldsymbol{w} := (\boldsymbol{y}^T, \boldsymbol{x}^T, \boldsymbol{\lambda}^T)^T \in \mathcal{Y} \times \mathcal{X} \times \Lambda$, and let $\{\boldsymbol{w}^k\}$ denote the sequence generated by **Algorithm 2**. Further we define

$$\theta(\boldsymbol{x}) := \sum_{i=1}^m f_i(x_i), \quad F_1(\boldsymbol{w}) := \begin{pmatrix} \boldsymbol{\lambda} \\ -\boldsymbol{A}^T \boldsymbol{\lambda} \\ A_1 x_1 - y_1 - \frac{b}{m} \\ \vdots \\ A_m x_m - y_m - \frac{b}{m} \end{pmatrix},$$

where $\boldsymbol{A} := \operatorname{diag}\{A_1, \ldots, A_m\}$. By [32, Theorem 4.1], we have that at any given iteration $K > 0$, the solution $\tilde{\boldsymbol{w}}^K = \frac{1}{K+1} \sum_{k=0}^K \boldsymbol{w}^k$ is an $\epsilon-$optimal solution for problem (1.1). That is, we have

$$\theta(\tilde{\boldsymbol{x}}^K) - \theta(\boldsymbol{x}) + (\tilde{\boldsymbol{w}}^K - \boldsymbol{w})^T F_1(\boldsymbol{w}) \leq \frac{C_p^1}{2(K+1)}, \quad \forall\, \boldsymbol{w} \in \mathcal{Y} \times \mathcal{X} \times \Lambda, \tag{2.5}$$

where $C_p^1 = \max_{\boldsymbol{w} \in \mathcal{W}} \|\boldsymbol{w} - \boldsymbol{w}^0\|_H^2$ and H is a positive semi-definite matrix which is associated with $\{A_i\}$, β and $\{\tau_i\}$. Note that at optimality, the left hand side of (2.5) is no greater than zero, therefore the above inequality is indeed a possible measure of the optimality gap, although it is implicit. In the following, we show explicitly that the objective values decrease at the rate $\mathcal{O}(1/K)$.

Theorem 2.2. *Let* $\{\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k\}$ *be the sequence generated by* **Algorithm 2**, *and* $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{\lambda}^*)$ *be any optimal solution, then we have*

$$\theta(\tilde{\boldsymbol{x}}^K) - \theta(\boldsymbol{x}^*) \leq \frac{C_p^2}{2K},$$

where $\tilde{\boldsymbol{x}}^K = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}^k$ *and* $C_p^2 = \sum_{i=1}^{m}\left\{\frac{\|\lambda_i^0\|^2}{\beta} + \|x_i^* - x_i^0\|^2 + \beta\|y_i^* - A_i x_i^0 + \frac{b}{m}\|^2\right\}$.

Proof. Please see **Appendix**. □

It is worth noting that the complexity results presented in Theorem 2.2 and the one presented in (2.5) do not imply each other. Moreover, in the proof of Theorem 2.2, we have exploited certain structure of **Algorithm 2**, therefore this result does not carry over to the general ADMM algorithm.

Next we show that the linear rate of convergence for **Algorithm 2** can also be established using existing results for the *Douglas-Rachford Splitting Method* (DRSM). To this end, we first derive the relationship between **Algorithm 2** and the DRSM. Recall that DRSM solves the following problem

$$\text{Find} \quad u, \quad \text{s.t.} \quad 0 \in \mathcal{A}(u) + \mathcal{B}(u), \tag{2.6}$$

by generating two sequences $\{u^k\}$ and $\{v^k\}$ according to:

Douglas-Rachford Splitting Method

Initialize τ, and $v^0, u^0 = J_{\tau\mathcal{B}}(v^0)$.
For $k = 1, 2, \ldots,$ do
- $v^{k+1} = J_{\tau\mathcal{A}}(2J_{\tau\mathcal{B}} - \mathcal{I})(v^k) + (\mathcal{I} - J_{\tau\mathcal{B}})(v^k)$,
- $u^{k+1} = J_{\tau\mathcal{B}}(v^{k+1})$.

To see the exact form of the operators $\mathcal{A}$ and $\mathcal{B}$ for **Algorithm 2**, let us consider the dual formulation of (2.1), stated below

$$\begin{aligned}
&\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{Y}} \sum_{i=1}^{m} f_i(x_i) - \sum_{i=1}^{m}\langle\lambda_i, A_i x_i - \frac{b}{m} - y_i\rangle \\
&= \min_{\boldsymbol{\lambda}}\left\{\max_{\boldsymbol{x}} \sum_{i=1}^{m}\langle\lambda_i, A_i x_i\rangle - f_i(x_i) + \max_{\boldsymbol{y}\in\mathcal{Y}}\sum_{i=1}^{m}\langle\lambda_i, -\frac{b}{m} - y_i\rangle\right\} \\
&= \min_{\boldsymbol{\lambda}}\left\{\mathcal{I}(\boldsymbol{\lambda} : \lambda_1{=}\lambda_2{=}\ldots{=}\lambda_m) + \sum_{i=1}^{m}\langle\lambda_i, -\frac{b}{m}\rangle + \sum_{i=1}^{m} f_i^*(A_i^T\lambda_i)\right\},
\end{aligned} \tag{2.7}$$

where $\mathcal{I}(\cdot)$ denotes the indicator function. By setting

$$\mathcal{A} := \partial(\mathcal{I}(\boldsymbol{\lambda} : \lambda_1{=}\lambda_2{=}\ldots{=}\lambda_m)) - (I, \ldots, I)^T\frac{b}{m} \tag{2.8}$$

$$\mathcal{B} := \sum_{i=1}^{m}\partial(f_i^* \circ A_i^T), \tag{2.9}$$

we can rewrite the dual form of (2.1) (i.e., eq. (2.7)) equivalently as finding a $\boldsymbol{\lambda}^*$ that satisfies

$$0 \in \mathcal{A}(\boldsymbol{\lambda}^*) + \mathcal{B}(\boldsymbol{\lambda}^*). \tag{2.10}$$

Applying DRSM to solve (2.10), we obtain the $(k+1)$th iterate as follows

$$\begin{aligned}\boldsymbol{v}^{k+1}=&J_{\beta\mathcal{A}}(2J_{\beta\mathcal{B}}-\mathcal{I})(\boldsymbol{v}^k)+(\mathcal{I}-J_{\beta\mathcal{B}})(\boldsymbol{v}^k)\\ =&\arg\min_{\boldsymbol{v}}\left\{\mathcal{I}(\boldsymbol{v}:v_i=v_j)-\sum_{i=1}^m\langle v_i,\frac{b}{m}\rangle+\frac{1}{2\beta}\|\boldsymbol{v}-(2\boldsymbol{u}^k-\boldsymbol{v}^k)\|^2\right\}+(\boldsymbol{v}^k-\boldsymbol{u}^k),\end{aligned}\tag{2.11}$$

$$\boldsymbol{u}^{k+1}=J_{\beta\mathcal{B}}(\boldsymbol{v}^{k+1})=\arg\min_{\boldsymbol{u}}\ \sum_{i=1}^m f_i^*(A_i^Tu_i)+\frac{1}{2\beta}\|\boldsymbol{u}-\boldsymbol{v}^{k+1}\|^2.\tag{2.12}$$

Further, applying the *Fenchel-Rockafellar Duality* [1, Definition 15.19], we can write the dual problem for (2.11) and (2.12) (with dual variables $\boldsymbol{y}$ and $\boldsymbol{x}$) as

$$\begin{aligned}\boldsymbol{y}^{k+1}=&\arg\min_{\boldsymbol{y}\in\mathcal{Y}}\left\{\frac{\beta}{2}\sum_{i=1}^m\left\|-y_i-\frac{b}{m}-\frac{2u_i^k-v_i^k}{\beta}\right\|^2\right\},\ v_i^{k+1}=u_i^k-\beta\left(-y_i^{k+1}-\frac{b}{m}\right),\\ \boldsymbol{x}^{k+1}=&\arg\min_{\boldsymbol{x}}\left\{\sum_{i=1}^m f_i(x_i)+\frac{\beta}{2}\left\|Ax_i-\frac{v_i^{k+1}}{\beta}\right\|^2\right\},\ u_i^{k+1}=v_i^{k+1}-\beta Ax_i^{k+1}.\end{aligned}$$

Then obviously when substituting $u_i^k=v_i^k-\beta Ax_i^k$ into the subproblem about $\boldsymbol{y}$, we obtain

$$\boldsymbol{y}^{k+1}=\arg\min_{\boldsymbol{y}\in\mathcal{Y}}\left\{\frac{\beta}{2}\sum_{i=1}^m\left\|Ax_i^k-y_i-\frac{b}{m}-\frac{u_i^k}{\beta}\right\|^2\right\}.$$

Similarly, when substituting $v_i^{k+1}=u_i^k-\beta\left(-y_i^{k+1}-\frac{b}{m}\right)$ into the subproblem about $\boldsymbol{x}$, we can get the following equivalent problem for $\boldsymbol{x}$

$$\boldsymbol{x}^{k+1}=\arg\min_{\boldsymbol{x}}\left\{\sum_{i=1}^m f_i(x_i)+\frac{\beta}{2}\left\|Ax_i-y_i^{k+1}-\frac{b}{m}-\frac{u_i^k}{\beta}\right\|^2\right\}.$$

Combining $u_i^{k+1}=v_i^{k+1}-\beta Ax_i^{k+1}$ and $v_i^{k+1}=u_i^k-\beta\left(-y_i^{k+1}-\frac{b}{m}\right)$, we obtain the update of $\boldsymbol{u}$,

$$u_i^{k+1}=u_i^k-\beta\left(Ax_i^{k+1}-y_i^{k+1}-\frac{b}{m}\right).$$

The above analysis indicates that the sequence $\{\boldsymbol{u}^k\}$ is the same as the multiplier sequence $\{\boldsymbol{\lambda}^k\}$ in **Algorithm 2**. As a result, **Algorithm 2** (or in general the two-block ADMM) can be considered as a special case of DRSM.

The linear convergence of DRSM has been well studied in [34, Proposition 4] with an assumption that operator $\mathcal{B}$ is both strongly monotone and Lipschitz, which means there exists $\alpha>0$ and M such that

$$\|\mathcal{B}(x_1)-\mathcal{B}(x_2)\|\le M\|x_1-x_2\|,$$

$$\langle\mathcal{B}(x_1)-\mathcal{B}(x_2),x_1-x_2\rangle\ge\alpha\|x_1-x_2\|^2.$$

By using the results in [22, 39, 42], we can show that if f_i's are all strongly convex with Lipschitz continuous gradients, and when A_i's are all full row rank, then the operator $\mathcal{B}$ is strongly monotone and Lipschitz. As a result, the sequences $\{\boldsymbol{x}^k\}$, $\{\boldsymbol{y}^k\}$ and $\{\boldsymbol{\lambda}^k\}$ generated by **Algorithm 2** converge linearly. Similarly, if each subproblem cannot be solved exactly,

then the linear convergence of the inexact version of **Algorithm 2** (cf. (2.4), with $\tau_i > \beta \cdot \rho(A_i^T A_i)$) can be established by following [12, THEOREM 4]. Again we require that f_i's are all strongly convex with Lipschitz continuous gradients, and A_i's all have full row rank.

To this point, all the convergence results characterize the behavior of **Algorithm 2** for solving problem (2.1). As problem (1.1) is an equivalent reformulation of (2.1), we can readily conclude that the sequence $\{\boldsymbol{x}^k\}$ generated by **Algorithm 2** converges to the primal optimal solution of (1.1), if either each subproblem is exactly solved and A_i's are all full rank, or the subproblems are solved using the proximal step (2.4) with $\tau_i > \beta \cdot \rho(A_i^T A_i)$. Further, if $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ is linearly convergent to an optimal solution of problem (2.1), then $\boldsymbol{x}^k$ converges linearly to the primal optimal solution of (1.1).

2.2 Dual Splitting ADMM

We can also apply the splitting technique to the dual formulation (1.1) to derive a dual splitting ADMM algorithm. In particular, let us first write (1.1) in its saddle point form

$$\min_{x_i \in \mathcal{X}_i} \ \max_{\lambda} \ \sum_{i=1}^m f_i(x_i) - \langle \lambda, \sum_{i=1}^m A_i x_i - b \rangle. \tag{2.1}$$

By exchanging the order of max and min, and using the definition of the conjugate function of f_i, we can rewrite (2.1) equivalently as

$$\begin{aligned} &\max_{\lambda} \ \min_{x_i \in \mathcal{X}_i} \ \sum_{i=1}^m f_i(x_i) - \langle \lambda, \sum_{i=1}^m A_i x_i - b \rangle \\ \Leftrightarrow &\max_{\lambda} \ \min_{x_i \in \mathcal{X}_i} \left\{ \sum_{i=1}^m \big(f_i(x_i) - \langle A_i^T \lambda, x_i \rangle \big) \right\} + \langle \lambda, b \rangle \\ \Leftrightarrow &\max_{\lambda} \ -\sum_{i=1}^m f_i^*(A_i^T \lambda) + \langle \lambda, b \rangle \ \Leftrightarrow \ \min_{\lambda} \ \sum_{i=1}^m f_i^*(A_i^T \lambda) - \langle \lambda, b \rangle, \end{aligned} \tag{2.2}$$

where λ denotes the dual variable of (1.1). We then split the dual variable λ by introducing a set of auxiliary variables $\{\lambda_i\}_{i=1}^m$, and rewrite (2.2) as

$$\min_{\lambda, \ \lambda_i} \ \sum_{i=1}^m f_i^*(A_i^T \lambda_i) - \langle \lambda, b \rangle, \quad \text{s.t. } \lambda = \lambda_i, \ i = 1, \ldots, m. \tag{2.3}$$

It is obvious that each primal optimal solution $\{\lambda_i^*\}_{i=1}^m$, λ^* of (2.3) corresponds to a dual optimal solution of (1.1) ($\lambda_i^* = \lambda^*$). The augmented Lagrangian function for this dual problem can be expressed as follows:

$$L(\{\lambda_i\}, \lambda, \{t_i\}) = \sum_{i=1}^m f_i^*(A_i^T \lambda_i) - \langle \lambda, b \rangle - \sum_{i=1}^m \langle t_i, \lambda - \lambda_i \rangle + \frac{\beta}{2} \sum_{i=1}^m \|\lambda - \lambda_i\|^2, \tag{2.4}$$

where β is the penalty parameter for the constraints violation, and $t_i \in \mathbb{R}^\ell$ is the Lagrangian multiplier associated with the constraint $\lambda = \lambda_i$. Denote $\boldsymbol{t} := (t_1^T, \ldots, t_m^T)^T \in \mathbb{R}^{m\ell}$ and $\tilde{\boldsymbol{\lambda}} := (\lambda_1^T, \ldots, \lambda_m^T)^T \in \mathbb{R}^{m\ell}$. It is clear now that optimizing the augmented Lagrangian for each auxiliary variable λ_i is independent of all other auxiliary variables $\{\lambda_j\}_{j \neq i}$. Consequently by treating $\tilde{\boldsymbol{\lambda}}$ and λ as two block variables, we can again apply the two-block ADMM to solve (2.3). In the following we take a closer look at the structure of each subproblem.

The subproblem for $\tilde{\boldsymbol{\lambda}}$ is related to the conjugate function $f_i^*(\cdot)$. At the kth iteration, this subproblem can be explicitly expressed as the following m independent problems (one for each variable λ_i):

$$\lambda_i^{k+1} = \arg\min_{\lambda_i} \; f_i^*(A_i^T\lambda_i) + \frac{\beta}{2}\left\|\lambda_i - \lambda^{k+1} + \frac{t_i^k}{\beta}\right\|^2. \tag{2.5}$$

By the classical *Fenchel-Rockafellar* duality [1,39], the dual problem of (2.5) can be expressed as

$$x_i^{k+1} = \arg\min_{x_i\in\mathcal{X}_i} \; f_i(x_i) + \frac{1}{2\beta}\left\|A_ix_i - \beta(\lambda^{k+1} - \frac{t_i^k}{\beta})\right\|^2, \tag{2.6}$$

where $\{x_i\}$ is precisely the set of primal variables of (1.1). The relationship between λ_i^{k+1} and x_i^{k+1} is as follows

$$\lambda_i^{k+1} = \lambda^{k+1} - \frac{t_i^k}{\beta} - \frac{1}{\beta}A_ix_i^{k+1}. \tag{2.7}$$

The dual splitting ADMM is stated formally in the following table.

Algorithm 3: Dual Splitting ADMM for (2.3)

Initialize $\{\lambda_1^0,\ldots,\lambda_m^0\}$, $\{t_1^0,\ldots,t_m^0\}$, and β.
For $k = 0, 1, 2, \ldots,$ do

- Compute λ^{k+1},
$$\lambda^{k+1} = \arg\min_{\lambda} \; -\lambda^T b - \sum_{i=1}^m \lambda^T t_i^k + \tfrac{\beta}{2}\sum_{i=1}^m \|\lambda - \lambda_i^k\|^2,$$
- Compute λ_i^{k+1}, for all $i = 1,\ldots,m$,
$$x_i^{k+1} = \arg\min_{x_i\in\mathcal{X}_i} \; f_i(x_i) + \tfrac{1}{2\beta}\left\|A_ix_i - \beta(\lambda^{k+1} - \tfrac{t_i^k}{\beta})\right\|^2,$$
$$\lambda_i^{k+1} = \lambda^{k+1} - \tfrac{t_i^k}{\beta} - \tfrac{1}{\beta}A_ix_i^{k+1},$$
- Compute t_i^{k+1}, $\forall\ i = 1,\ldots,m$,
$$t_i^{k+1} = t_i^k - \beta(\lambda^{k+1} - \lambda_i^{k+1}).$$

Similar to the case of primal splitting, the subproblem of λ can be solved easily in closed-form

$$\lambda^{k+1} = \frac{1}{m\beta}\left(b + \sum_{i=1}^m (t_i^k + \beta\lambda_i^k)\right). \tag{2.8}$$

Furthermore, the subproblem for the block variable x_i can be solved efficiently if A_i is an identity matrix (or any of its constant multiples), because (2.6) can be efficiently solved by computing the proximity operator (1.7). Else, a proximal gradient step can be performed, i.e.,

$$\begin{aligned} x_i^{k+1} =& \arg\min_{x_i\in\mathcal{X}_i} \; f_i(x_i) + \left\langle \frac{1}{\beta}A_i^T(A_ix_i^k - \beta\lambda^{k+1} + t_i^k), x_i - x_i^k\right\rangle + \frac{\tau_i}{2}\|x_i - x_i^k\|^2, \\ =& \arg\min_{x_i\in\mathcal{X}_i} \; f_i(x_i) + \frac{\tau_i}{2}\left\|x_i - x_i^k + \frac{1}{\tau_i\beta}A_i^T(A_ix_i^k - \beta\lambda^{k+1} + t_i^k)\right\|^2, \\ =& \mathrm{Prox}_{f_i}^{\frac{1}{\tau_i}}\left(x_i^k - \frac{1}{\tau_i\beta}A_i^T(A_ix_i^k - \beta\lambda^{k+1} + t_i^k)\right), \end{aligned} \tag{2.9}$$

where τ_i denotes the penalty parameter for the distance between x_i^{k+1} and x_i^k.

Again, the global convergence of **Algorithm 3** is a straightforward consequence of the standard convergence results for the two-block ADMM.

Theorem 2.3. *For any $\beta > 0$, suppose the subproblem for x_i is either exactly solved, or is solved using* (2.9) *with $\tau_i > \frac{\rho(A_i^T A_i)}{\beta}$ associated with λ_i. Let $(\tilde{\boldsymbol{\lambda}}^k, \lambda^k, \boldsymbol{t}^k)$ be any sequence generated by Algorithm 3. Then starting with any initial point $(\tilde{\boldsymbol{\lambda}}^0, \lambda^0, \boldsymbol{t}^0) \in \mathbb{R}^{m\ell} \times \mathbb{R}^{\ell} \times \mathbb{R}^{m\ell}$, we have*

1. *The sequence $\{\boldsymbol{t}^k\}$ converges to the dual optimal solution for problem* (2.3).
2. *The sequence $\left\{ \sum_{i=1}^m f_i^*(A_i^T \lambda_i^k) - \langle \lambda^k, b \rangle \right\}$ converges to the primal optimal value for problem* (2.3).
3. *The residual sequence $\{\lambda^k - \lambda_i^k\}$ converges to 0 for each $i = 1, \ldots, m$.*
4. *The sequence $\{\tilde{\boldsymbol{\lambda}}^k, \lambda^k\}$ converges to the primal optimal solution for problem* (2.3).
5. *For each $i = 1, \ldots, m$, if the subproblem about x_i is exactly solved, then the sequence $\{A_i x_i^k\}$ converges. If A_i has full column rank, then $\{x_i^k\}$ converges to x_i^*, for all $i = 1, \ldots, m$; the same is true if the subproblem about x_i is solved using* (2.9) *with $\tau_i > \frac{\rho(A_i^T A_i)}{\beta}$.*

Proof. When the subproblems are solved exactly, **Algorithm 3** corresponds to the classical two-block ADMM applied to solve the equivalent formulation (2.3). As a result, the first four conclusions follow directly from the classical analysis of the two-block ADMM (see, e.g., [2, Section 3.2]). In the last conclusion, the convergence of $\{A_i x_i^k\}$ follows from the convergence of $\{\lambda_i^k\}$, λ^t and $\{t_i^t\}$; see (2.7). When the subproblems are solved inexactly with $\tau_i > \frac{\rho(A_i^T A_i)}{\beta}$, there is no existing result which covers the convergence of the algorithm. We will provide a proof for this case in the **Appendix**. □

Let us discuss some additional convergence properties of **Algorithm 3**. First of all, it is possible to derive the iteration complexity for both the exact and the inexact versions of the dual splitting ADMM algorithm. For the exact version, its iteration complexity based on variational inequalities follows from the existing results [32]. The iteration complexity of the inexact dual splitting ADMM algorithm is not covered by any existing result. As a result, in **Appendix**, we provide a unified iteration complexity analysis for the dual splitting ADMM algorithm. Additionally, similar to Theorem 2.2, we have the following result that bounds the gap of objective value. The proof is similar to that of Theorem 2.2, thus we omit it for brevity.

Corollary 2.4. *Let $\{\boldsymbol{x}^k, \lambda^k\}$ be the sequence generated by* **Algorithm 3** *(using either the exact minimization or the inexact version with $\tau_i > \frac{\rho(A_i^T A_i)}{\beta}$) and $\boldsymbol{x}^*$ be any primal optimal solution. Define $\tilde{\boldsymbol{x}}^K := \frac{1}{K}\sum_{k=1}^K \boldsymbol{x}^k$, $\tilde{\lambda}^K := \frac{1}{K}\sum_{k=1}^K \lambda^k$. We have*

$$\theta(\tilde{\boldsymbol{x}}^K) - \theta(\boldsymbol{x}) + \begin{pmatrix} \tilde{\boldsymbol{x}}^K - \boldsymbol{x} \\ \tilde{\lambda}^K - \lambda \end{pmatrix}^T F_2(\boldsymbol{x}, \lambda) \le \frac{C_d^1}{2K}, \ \forall \begin{pmatrix} \boldsymbol{x} \\ \lambda \end{pmatrix} \in \mathcal{X} \times \mathbb{R}^{\ell}, \tag{2.10}$$

$$\theta(\tilde{\boldsymbol{x}}^K) - \theta(\boldsymbol{x}^*) \le \frac{C_d^2}{2K},$$

$$C_d^1 = \max_{\boldsymbol{x}\in\mathcal{X},\lambda\in\mathbb{R}^l} \left\{ \sum_{i=1}^m \|x_i - x_i^0\|_{G_i}^2 + \beta\|\lambda - \lambda_i^0\|^2 \right\},$$

$$C_d^2 = \left\{ \frac{\beta}{2} \sum_{i=1}^m \left\|\lambda_i^0\right\|^2 + \frac{1}{2\beta} \sum_{i=1}^m \left\|t_i^* - t_i^0\right\|^2 + \frac{1}{2} \sum_{i=1}^m \left\|x_i^* - x_i^0\right\|_{\tilde{P}_i}^2 \right\},$$

where $F_2(\boldsymbol{x},\lambda) = \begin{pmatrix} -A_1^T\lambda \\ \cdots \\ -A_m^T\lambda \\ \sum_{i=1}^m A_i x_i - b \end{pmatrix}$ *and* $G_i = \frac{1}{\beta}A_i^T A_i$, $\tilde{P}_i = 0$ *for the exact version and* $G_i = \tau_i I$, $\tilde{P}_i = \tau_i I - \frac{1}{\beta}A_i^T A_i$ *for the inexact version.*

Further, the linear rate of convergence for **Algorithm 3** can also be established using existing results. We apply DSRM with two operators $\mathcal{A} := \partial(\mathcal{I}(\sum_{i=1}^m t_i + b = 0))$ and $\mathcal{B} := \sum_{i=1}^m \partial(f_i \circ A_i^T)$. By following an analysis similar to that of **Algorithm 2** and using assumptions that f_i's are strongly convex and have Lipschitz continuous gradients, and that the matrices A_i's have full row rank, we can prove that $\{\tilde{\boldsymbol{\lambda}}^k\}$, $\{\lambda^k\}$, $\{\boldsymbol{t}^k\}$ and $\{\boldsymbol{x}^k\}$ converge linearly.

From the equivalence relationship between the problems (1.1) and (2.3), we can readily claim that the primal optimal solution λ^* of (2.3) is the dual optimal solution of (1.1). Recall that from the discussion following (2.6), $\{x_i\}$ is the set of primal variables for the original problem (1.1).

2.3 Discussions

Several existing methods for solving (1.1) are similar to the two algorithms (**Algorithm 2** and **Algorithm 3**) proposed in this paper. Specifically, [17] presents an generalized ADMM framework which used the same derivation idea as **Algorithm 2** for linear programming. In particular, Spingarn [43] applied a method called *partial inverse method* [44] to separable convex problems. This method can be directly applied to (1.1) as follows. Let us define two subspaces A and B as:

$$A := \left\{ (\boldsymbol{x},\boldsymbol{u}) \mid \sum_{i=1}^m u_i = 0 \right\}, \qquad B := \{(0,\boldsymbol{u}) \mid u_1 = \cdots = u_m\},$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $u_i \in \mathbb{R}^\ell$, $\boldsymbol{u} := (u_1^T, \ldots, u_m^T)^T \in \mathbb{R}^{m\ell}$. Define the function

$$F(\boldsymbol{x},\boldsymbol{u}) = \begin{cases} \sum\limits_{i=1}^m f_i(x_i), & \text{if } A_i x_i - \frac{b}{m} = u_i, x_i \in \mathcal{X}_i, i = 1,\ldots,m, \\ +\infty, & \text{otherwise.} \end{cases}$$

Then problem (1.1) is equivalent to the one that minimizes $F(\boldsymbol{x},\boldsymbol{u})$ over A. Define two operators

$$P_A(\boldsymbol{x},\boldsymbol{u}) = \left(\boldsymbol{x}, \boldsymbol{u} - (I,\ldots,I)^T \sum_{i=1}^m \frac{u_i}{m} \right), \quad P_B(\boldsymbol{x},\boldsymbol{u}) = \left(0, (I,\ldots,I)^T \sum_{i=1}^m \frac{u_i}{m} \right).$$

To solve (1.1), partial inverse method generates a sequence of iterates $\{(\boldsymbol{x}^k,\boldsymbol{y}^k)\}$ and $\{(0,\boldsymbol{\lambda}^k)\}$:

$$(\boldsymbol{x}^{k+1},\boldsymbol{y}^{k+1}) = P_A(\tilde{\boldsymbol{x}}^k,\tilde{\boldsymbol{y}}^k), \quad (0,\boldsymbol{\lambda}^{k+1}) = P_B(0,\tilde{\boldsymbol{\lambda}}^k),$$

where $\{(\tilde{\boldsymbol{x}}^k, \tilde{\boldsymbol{y}}^k), (0, \tilde{\boldsymbol{\lambda}}^k)\}$ satisfies

$$(\boldsymbol{x}_k, \boldsymbol{y}^k) + (0, \boldsymbol{\lambda}_k) = (\tilde{\boldsymbol{x}}^k, \tilde{\boldsymbol{y}}^k) + (0, \tilde{\boldsymbol{\lambda}}^k),$$
$$\frac{1}{c_k} P_A(0, \tilde{\boldsymbol{\lambda}}^k) + P_B(0, \tilde{\boldsymbol{\lambda}}^k) \in \partial F \left(P_A(\tilde{\boldsymbol{x}}^k, \tilde{\boldsymbol{y}}^k) + \frac{1}{c_k} P_B(\tilde{\boldsymbol{x}}^k, \tilde{\boldsymbol{y}}^k) \right).$$

with positive sequence $\{c_k\}$ bounded away from zero.

From [43, Algorithm 2], it is known that the partial inverse method is the same as ADMM for minimizing $F(\boldsymbol{x}, \boldsymbol{y})$ over A. That is, the variables $(\boldsymbol{x}, \boldsymbol{y})$ in the Partial Inverse are the same as those in **Algorithm 2** when the subproblems about x_i are solved exactly. However, the subproblem about $\boldsymbol{\lambda}$ in the partial inverse method additionally requires every component of $\boldsymbol{\lambda}$ to be equal, which is different from that in **Algorithm 2**. Notice that the results in [43] do not apply to the case when the subproblems for x_i are solved inexactly.

Algorithm 3 is related to the proximal decomposition method proposed in [9]. The latter solves

$$\min_{x \in \mathcal{X}} \quad \sum_{i=1}^{m} f_i(x) \tag{2.11}$$

by applying the two block ADMM to the following reformulation

$$\min_{\{x_1, x_2, \ldots, x_m\}} \quad \sum_{i=1}^{m} f_i(x_i) \quad \text{s.t.} \quad x_i = y,\ x_i \in \mathcal{X},\ i = 1, \ldots, m, \tag{2.12}$$

where $\mathcal{X}$ is the common closed convex constrained set for all $\{x_i\}_{i=1}^m$. In this decomposition, a single variable x is split into m copies $\{x_i\}_{i=1}^m$, and the consistency among these copies are enforced using the linking variable y. This decomposition technique is also used in **Algorithm 3**, but for solving the dual reformulation of (1.1).

Algorithm 2 is closely related to the *distributed sharing* algorithm presented in [2, Chapter 7]. Consider the following sharing problem, in which m agents jointly solve the following problem

$$\min_{\{x_1, x_2, \ldots, x_m\}} \quad \sum_{i=1}^{m} f_i(x_i) + g\left(\sum_{i=1}^{m} A_i x_i - b \right), \quad \text{s.t.} \quad x_i \in \mathcal{X}_i,\ i = 1, \ldots, m, \tag{2.13}$$

where $f_i(\cdot)$ is the cost related to agent i; $g(\cdot)$ is the cost shared among all the agents. The distributed sharing algorithm introduces a set of extra variables $y_i = A_i x_i - \frac{b}{m},\ \forall\ i$, and applies the two-block ADMM to the following reformulation

$$\min_{\{x_1, x_2, \ldots, x_m\}} \quad \sum_{i=1}^{m} f_i(x_i) + g\left(\sum_{i=1}^{m} y_i \right), \quad \text{s.t.} \quad A_i x_i - \frac{b}{m} = y_i,\ x_i \in \mathcal{X}_i,\ i = 1, \ldots, m, \tag{2.14}$$

To see the relationship between **Algorithm 2** and the distributed sharing algorithm, we note that problem (1.1) is a special case of problem (2.13), with $g(\cdot)$ being the indicator function. Hence **Algorithm 2** with the subproblems being solved exactly can be viewed as a special case of the distributed sharing algorithm.

3 Numerical Experiments

In this section, we test **Algorithms 1, 2 and 3** on three problems: Basis Pursuit, Latent Variable Gaussian Graphical Model Selection and Robust Principal Component Analysis,

and compare their performance with the *Alternating Direction Method* (ADM) [48], *Proximal Gradient based ADM* (PGADM) [37], *ADMM with Gaussian Back Substitution* (ADMGBS) [29], and *Variant Alternating Splitting Augmented Lagrangian Mehtod* (VASALM) [45]. Our codes were written in MATLAB 7.14(R2012a) and all experiments were conducted on a laptop with Intel Core 2 Duo@2.40GHz CPU and 4GB of memory.

3.1 Basis Pursuit

Consider the following basis pursuit (BP) problem [8]

$$\min \ \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad A\boldsymbol{x} = b, \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^p$, and $A \in \mathbb{R}^{n\times p}$, $b \in \mathbb{R}^n$. This model has applications in compressed sensing where a sparse signal x needs to be recovered using a small number of observations b (i.e., $n \ll p$) [8].

By letting $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, the BP problem can be viewed as a special case of problem (1.1) with m block variables. If we set $m = p$ (i.e., each component x_i is viewed as a single block variable), then **Algorithm 1** can be used, with each of its primal iteration given by

$$x_i^{k+1} = \arg\min_{x_i} \ |x_i| + \frac{\beta}{2}\left\|\sum_{j=1}^{i-1} a_j x_j^{k+1} + a_i x_i + \sum_{j=i+1}^{p} a_j x_j^k - b - \frac{\lambda^k}{\beta}\right\|^2$$

$$= \textbf{shrinkage}\left(\frac{1}{\sqrt{a_i^T a_i}}\left[b + \frac{\lambda^k}{\beta} - \sum_{j=1}^{i-1} a_j x_j^{k+1} - \sum_{j=i+1}^{p} a_j x_j^k\right], \frac{1}{\beta(a_i^T a_i)}\right). \tag{3.2}$$

Alternatively, when the number of blocks m is chosen as $m < p$, the primal subproblem in **Algorithm 1** cannot be solved exactly. In this case, the inexact version of **Algorithm 2** and **Algorithm 3** can be used, where the primal subproblems (2.4) and (2.9) are respectively given by

$$(2.4) \Leftrightarrow \boldsymbol{x}_i^{k+1} = \textbf{shrinkage}\left(\boldsymbol{x}_i^k - \frac{\beta A_i^T(A_i\boldsymbol{x}_i^k - \frac{b}{m} - \boldsymbol{y}_i^{k+1} - \frac{\lambda_i^k}{\beta})}{\tau_i}, \frac{1}{\tau_i}\right),$$

$$(2.9) \Leftrightarrow \boldsymbol{x}_i^{k+1} = \textbf{shrinkage}\left(\boldsymbol{x}_i^k - \frac{1}{\tau_i\beta}A_i^T(A_i\boldsymbol{x}_i^k - \beta\boldsymbol{\lambda}^{k+1} + \boldsymbol{t}_i^k), \frac{1}{\tau_i}\right).$$

In the following, we compare **Algorithm 1** (with $m = p$), and the inexact versions of **Algorithm 2**, **Algorithm 3** (with $m = 2, 5, 10, 20, 50, 100, 200$) with the ADM algorithm [48], which has been shown to be effective for solving BP. **Algorithms 1–3** are denoted as MULTADMM, PSADMM and DSADMM, respectively.

In our experiment, the matrix A is randomly generated using standard Gaussian distribution per element; the true solution x^* is also generated using standard Gaussian distribution, with 6% sparsity level, i.e., 94% of the components are zero; the "observation" vector b is computed by $b = Ax^*$. The penalty coefficient β is set to be $\frac{400}{\|b\|_1}$, $\frac{400}{\|b\|_1}$, 10 and $\frac{400}{\|b\|_1}$ for MULTADMM, PSADMM, DSADMM and ADM respectively. Note that the penalty coefficient for the DSADMM is chosen differently because it is the ADMM applied to the dual of (1.1), while the rest of the algorithms applies directly to the primal version of (1.1). To ensure convergence, the proximal parameters are set to be $\tau_i = 1.01\beta \times \rho(A_i^T A_i)$ for PSADMM,

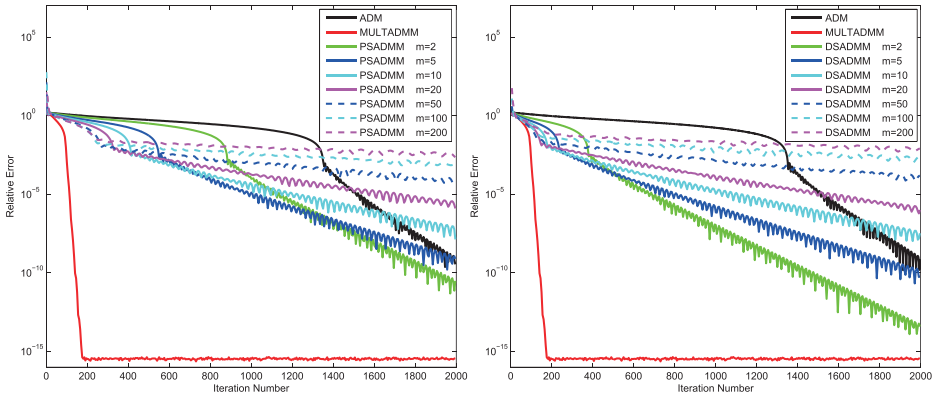

Figure 1: Numerical performance of PSADMM and DSADMM for $n = 300$ and $p = 1000$

$\tau_i = 1.01 \times \frac{\rho(A_i^T A_i)}{\beta}$ for DSADMM, and $\tau = 1.01\beta \times \rho(A^T A)$ for ADM, respectively. Figure 1 shows the convergence progress of the different algorithms. The curves in the figure represent the relative error for different algorithms along the iterations averaged over 100 runs. For a given iterate x^k, the relative error is defined as

$$\text{Relative Error} := \frac{\|x^k - x^*\|_2}{\|x^*\|_2}. \tag{3.3}$$

The left part of Figure 1 shows the performance of MULTADMM, PSADMM and ADM. We observe that PSADMM converges faster than ADM when the number of blocks is relatively small. When the number of blocks increases, PSADMM converges fast at the beginning, but becomes slower after about 200 iterations. This is because larger number of blocks results in smaller proximal parameter τ_i, hence larger penalty coefficients can be taken. On the other hand, it becomes increasingly difficult to simultaneously satisfy all the constraints. (Note that the number of constraint is the same as the number of block variables). We also observe that the MULTADMM performs much better than all other methods, in terms of both the convergence speed and the solution accuracy. The main reason for its superior performance is the exact solvability of the primal subproblems. Similar observations can be obtained from the right part of Figure 1, where the performance of MULTADMM, ADM and DSADMM are compared.

In Table 1 and Table 2, we report the performance of different algorithms for cases with $n = 300$, $p = 1000$ and $n = 600$, $p = 2000$, respectively. In these tables, 'Iter' denotes the iteration number with 2000 as the default maximum iteration number; 'Obj' denotes the objective value; 'Time' denotes the CPU time used; '$\sim$' indicates that the algorithm did not converge in 2000 iterations. Again we observe that to obtain the same accuracy, MULTADMM requires significantly fewer iterations and less CPU time than all other methods. In the meantime, PSADMM and DSADMM perform better than ADM when the number of blocks is not too large.

Note that the 2-norm of a matrix (i.e., its largest singular value) cannot decrease if some of its columns are removed.

Table 1. Numerical comparison for Basis Pursuit with $n = 300$ and $p = 1000$

	tol =1e-3				tol =1e-5			
Method	Iter	Obj	Time	Error	Iter	Obj	Time	Error
ADM	1129	48.43203	51.95	8.95e-04	1293	48.41133	59.49	9.17e-06
MULTADMM	102	48.40617	5.19	7.03e-04	113	48.40617	5.27	8.96e-06
PSADMM($m = 2$)	744	48.42083	29.12	8.88e-04	979	48.41131	37.57	9.16e-06
PSADMM($m = 5$)	522	48.41351	15.93	9.65e-04	883	48.41135	25.41	9.22e-06
PSADMM($m = 10$)	488	48.41123	16.46	9.49e-04	1050	48.41128	33.57	9.49e-06
PSADMM($m = 20$)	594	48.40421	25.03	9.76e-04	1464	48.41129	58.93	9.73e-06
PSADMM($m = 50$)	986	48.40993	114.10	9.85e-04	2000	48.41146	238.27	$\sim$
PSADMM($m = 100$)	1769	48.41169	250.37	9.82e-04	2000	48.41437	270.62	$\sim$
PSADMM($m = 200$)	2000	48.40564	315.62	$\sim$	2000	48.40564	318.10	$\sim$
DSADMM($m = 2$)	386	48.41086	39.58	9.58e-04	623	48.41132	63.93	8.75e-06
DSADMM($m = 5$)	329	48.40631	18.11	9.68e-04	718	48.41135	40.18	9.48e-06
DSADMM($m = 10$)	412	48.41406	26.04	9.77e-04	972	48.41130	56.93	9.14e-06
DSADMM($m = 20$)	565	48.40797	42.50	9.59e-04	1475	48.41129	111.09	9.54e-06
DSADMM($m = 50$)	1133	48.41005	149.80	9.81e-04	2000	48.41025	261.77	$\sim$
DSADMM($m = 100$)	1941	48.40482	354.21	$\sim$	2000	48.41040	359.20	$\sim$
DSADMM($m = 200$)	2000	48.41438	369.55	$\sim$	2000	48.41438	373.04	$\sim$

Table 2. Numerical comparison for Basis Pursuit with $n = 600$ and $p = 2000$

	tol =1e-3				tol =1e-5			
Method	Iter	Obj	Time	Error	Iter	Obj	Time	Error
ADM	883	96.66843	180.72	8.56e-04	1064	96.61717	217.73	9.32e-06
MULTADMM	66	96.65517	15.2	8.31e-04	83	96.61731	20.15	9.57e-06
PSADMM($m = 2$)	599	96.61964	112.1	9.67e-04	845	96.61708	157.4	9.49e-06
PSADMM($m = 5$)	446	96.61319	53.6	9.75e-04	840	96.61709	100.8	9.70e-06
PSADMM($m = 10$)	445	96.61696	47.8	9.78e-04	1060	96.61711	114.1	9.85e-06
PSADMM($m = 20$)	563	96.61880	65.7	9.87e-04	1584	96.61710	183.6	9.77e-06
PSADMM($m = 50$)	1068	96.61684	143.9	9.84e-04	2000	96.61755	267.5	$\sim$
PSADMM($m = 100$)	1916	96.61227	381.7	9.98e-04	2000	96.60569	394.9	$\sim$
PSADMM($m = 200$)	2000	96.58216	612.7	$\sim$	2000	96.58216	606.9	$\sim$
DSADMM($m = 2$)	757	96.65609	337.11	9.21e-04	1000	96.61711	443.90	9.41e-06
DSADMM($m = 5$)	529	96.62273	198.85	9.78e-04	924	96.61712	346.16	9.58e-06
DSADMM($m = 10$)	502	96.61274	159.71	9.78e-04	1109	96.61718	351.72	9.69e-06
DSADMM($m = 20$)	587	96.61563	107.28	9.75e-04	1572	96.61720	286.33	9.74e-06
DSADMM($m = 50$)	996	96.61757	228.22	9.87e-04	2000	96.61704	457.79	$\sim$
DSADMM($m = 100$)	1815	96.61811	619.64	9.96e-04	2000	96.61395	678.21	$\sim$
DSADMM($m = 200$)	2000	96.59811	1001.6	$\sim$	2000	96.59811	993.54	$\sim$

3.2 Latent Variable Gaussian Graphical Model Selection

The problem of latent variable Gaussian graphical model selection has been briefly introduced in Section 1. Recall that one of its equivalent reformulation is given by

$$\begin{aligned} \min_{S,L} \ & \langle R, \hat{\Sigma}_X \rangle - \text{logdet}(R) + \alpha_1 \|S\|_1 + \alpha_2 \mathbf{Tr}(L) + \mathcal{I}(L \succeq 0), \\ \text{s.t.} \ & R - S + L = 0. \end{aligned} \tag{3.4}$$

This model can be viewed as a combination of dimensionality reduction (to identify latent variables) and graphical modeling (to capture remaining statistical structure that is not attributable to the latent variables). It consistently estimates both the number of hidden components and the conditional graphical model structure among the observed variables. In the following, we show that to solve (3.4), the primal subproblems for **Algorithm 1**, **Algorithm 2** and **Algorithm 3** can be solved exactly and efficiently. To this end, we use two lemmas which can be found in [11, 36] to define tow operators as the same time $\mathcal{S}_\mu(\cdot)$ and $\mathcal{D}_\mu(\cdot)$ as in [36, Definition 3 and Definition 4]. Now we are ready to present the steps of different algorithms for solving (1.5), by using the previous two lemmas.

Algorithm 1: At the k-th iteration, the update rule is given by:

$$R^{k+1} = U \text{diag}(\gamma) U^T, S^{k+1} = \mathcal{S}_{\frac{\alpha_1}{\beta}}\left(R^{k+1} + L^k - \frac{\lambda^k}{\beta}\right), L^{k+1} = \mathcal{D}_{\frac{\alpha_2}{\beta}}\left(\frac{\lambda^k}{\beta} - R^{k+1} + S^{k+1}\right),$$

where $U\mathrm{diag}(\sigma)U^T$ is the eigenvalue decomposition of matrix $\frac{1}{\beta}\hat{\Sigma}_X-\frac{1}{\beta}\lambda^k-S^k+L^k$ and $\gamma_i=\left(-\sigma_i+\sqrt{\sigma_i^2+\frac{4}{\beta}}\right)/2,\ \forall\, i=1,\ldots,p.$

Algorithm 2: At the k-th iteration, the update rule is given by:

$$R^{k+1}=U\mathrm{diag}(\gamma)U^T,\ S^{k+1}=\mathcal{S}_{\frac{\alpha_1}{\beta}}\left(-y_2^{k+1}-\frac{\lambda_2^k}{\beta}\right),\ L^{k+1}=\mathcal{D}_{\frac{\alpha_2}{\beta}}\left(y_3^{k+1}+\frac{\lambda_3^k}{\beta}\right),$$

where $U\mathrm{diag}(\sigma)U^T$ is the eigenvalue decomposition of another matrix $\frac{1}{\beta}\hat{\Sigma}_X-(y_1^{k+1}+\frac{1}{\beta}\lambda_1^k)$ and $\gamma_i=\left(-\sigma_i+\sqrt{\sigma_i^2+\frac{4}{\beta}}\right)/2,\ \forall\, i=1,\ldots,p.$

Algorithm 3: At the k-th iteration, the update rule is given by:

$$R^{k+1}=U\mathrm{diag}(\gamma)U^T,\ S^{k+1}=\mathcal{S}_{\frac{\alpha_1}{\beta}}\left(-\beta\lambda^{k+1}+t_2^k\right),\ L^{k+1}=\mathcal{D}_{\frac{\alpha_2}{\beta}}\left(\beta\lambda^{k+1}-t_3^k\right),$$

where $U\mathrm{diag}(\sigma)U^T$ is the eigenvalue decomposition of another new matrix $\frac{1}{\beta}\hat{\Sigma}_X-(\beta\lambda^{k+1}-t_1^k)$ and $\gamma_i=\left(-\sigma_i+\sqrt{\sigma_i^2+\frac{4}{\beta}}\right)/2,\ \forall\, i=1,\ldots,p.$

In the following, all three methods are compared with PGADM [37], which is used to solve the same problem. The stopping criterion is set to be

$$\text{Relative Error:}=\max\left\{\frac{\|R^{k+1}-R^k\|_F}{\|R^k\|_F},\frac{\|S^{k+1}-S^k\|_F}{\|S^k\|_F},\frac{\|L^{k+1}-L^k\|_F}{\|L^k\|_F},\right.$$
$$\left.\frac{\|R^{k+1}-S^{k+1}+L^{k+1}\|_F}{\max\{1,\|R^k\|_F,\|S^k\|_F,\|L^k\|_F\}}\right\}\le\epsilon,$$

where ϵ is some given error tolerance. All the variables are initialized as zero matrices, and the error tolerance ϵ is set to be 10^{-5} in all the experiments. The comparison results are presented in Table 3, in which 'Iter', 'Obj' and 'Time' denote respectively the iteration number, the objective function value and the CPU time.

3.2.1 Synthetic Dataset

We first test the algorithms on synthetic dataset. The sample covariance matrix $\hat{\Sigma}_X$ is generated using the same procedure as in [37]. Let p and r denote the given dimension of the observed and the latent variables, respectively. We first randomly create a sparse matrix $U\in\mathbb{R}^{(p+r)\times(p+r)}$ with 90% of the entries being zeros, while the nonzero entries were set to be -1 or 1 with equal probability. Then the true covariance matrix is computed by $\Sigma_{X,Y}=(U\cdot U^T)^{-1}$, and the true concentration matrix is given by $\Theta_{X,Y}=U\cdot U^T$. According to (1.3), the sparse part of Σ_X^{-1} is given by

$$\Theta_X=\Theta_{X,Y}(1:p,1:p),$$

while its low rank part is computed as $\Theta_{XY}\Theta_Y^{-1}\Theta_{YX}=\Theta_{X,Y}(1:p,p+1:p+r)\cdot\Theta_{X,Y}(p+1:p+r,p+1:p+r)^{-1}\cdot\Theta_{X,Y}(p+1:p+r,1:p)$.

We then draw $N=5p$ independent and identically distributed vectors, $Y_1,\ldots,Y_N$, from the Gaussian distribution $\mathcal{N}(0,(\Theta_X-\Theta_{XY}\Theta_Y^{-1}\Theta_{YX})^{-1})$, and compute a sample covariance matrix of the observed variables according to $\hat{\Sigma}_X:=\frac{1}{N}\sum_{i=1}^N Y_iY_i^T$. The parameter β is set to be 0.1 for MULTADMM, 0.01 for PSADMM and DSADMM. For PGADM, following [37], β is set to be 0.1, and the parameter τ in PGADM is set to be 0.6.

Table 3 compares the algorithms with different choices of penalty parameters α_1 and α_2. We can find that the proposed methods PSADMM and DSADMM appear to perform better than the state-of-the-art method PGADM: similar objective function values are achieved using significantly less computational time and fewer iterations. Furthermore, MULTADMM is much faster than all the remaining three algorithms, although no theoretical results have been proved yet.

Table 3. Numerical comparison for LVGGMS

Penalty Parameter	Method	Iter	Obj	Time	Error
$\alpha_1 = 0.005$ $\alpha_2 = 0.05$	MULTADMM	50	-1.831463e+03	277.45	9.949149e-06
	PGADM	162	-1.835969e+03	871.57	9.902557e-06
	PSADMM	62	-1.836545e+03	346.44	8.954349e-06
	DSADMM	60	-1.836545e+03	340.86	8.849412e-06
$\alpha_1 = 0.01$ $\alpha_2 = 0.1$	MULTADMM	19	-1.739141e+03	97.41	7.423706e-06
	PGADM	124	-1.738876e+03	626.29	9.944299e-06
	PSADMM	47	-1.739141e+03	248.06	9.995064e-06
	DSADMM	48	-1.739141e+03	251.03	9.957426e-06
$\alpha_1 = 0.02$ $\alpha_2 = 0.2$	MULTADMM	19	-1.593774e+03	67.42	9.956842e-06
	PGADM	106	-1.593676e+03	534.68	9.671576e-06
	PSADMM	37	-1.593770e+03	195.42	8.702799e-06
	DSADMM	37	-1.593770e+03	193.00	8.719285e-06
$\alpha_1 = 0.04$ $\alpha_2 = 0.4$	MULTADMM	17	-1.356307e+03	89.87	7.067959e-06
	PGADM	84	-1.356285e+03	420.66	9.996129e-06
	PSADMM	35	-1.356306e+03	188.52	9.622188e-06
	DSADMM	35	-1.356306e+03	185.05	9.622188e-06

Acknowledgment

The authors are grateful to Professor Bingsheng He of Nanjing University, Professor Xiaoming Yuan of HongKong Baptist University and Dr. Tsung-Hui Chang of National Taiwan University of Science and Technology for valuable suggestions and discussions.

Appendix

Because of space limitation, the appendix document can be found on the authors website. Please download it if you are interested.

References

[1] H.H. Bauschke and P.L. Combettes, Fenchel-Rockafellar duality, in *Chapter 15 Convex Analysis and Monotone Operator Theory in Hulbert Spaces*, Springer, New York, 2011, pp. 207-222.

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

[3] E.J. Candès, X. Li, Y. Ma and J. Wright, Robust principal component analysis?, *Journal of the ACM* 48 (2011) 1–37.

[4] V. Chandrasekaran, P.A. Parrilo and A. S. Willsky, Latent variable graphical model selection via convex optimization, *The Annals of Statistics*, 40 (2012) 1935–1967.

[5] C.H. Chen, B.S. He and X. M. Yuan, Matrix completion via an alternating direction method, *IMA Journal of Numerical Analysis* 32 (2012) 227–245.

[6] G. Chen and M. Teboulle, A proximal-based decomposition method for convex minimization problems, *Mathematical Programming* 64 (1994) 81–101.

[7] S.S. Chen, D.L. Donoho and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review* 43 (2001) 129–159.

[8] S.S. Chen, D.L. Donoho and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing* 20 (1998) 33–61.

[9] P.L. Combettes and J.-C Pesquet, A proximal decomposition method for solving convex variational inverse problems, *Inverse Problems* 24 (2008) 065014.

[10] P.L. Combettes and J.-C Pesquet, Proximal splitting methods in signal processing, in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H.H. Bauschke et al., (eds), Springer, New York, 2011, pp. 185–212.

[11] I. Daubechies, M. Defrise and C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Communications in Pure and Applied Mathematics* 57 (2004) 1413–1457.

[12] W. Deng and W. T. Yin, On the global and linear convergence of the generalized alternating direction method of multipliers, *Journal of Scientific Computing* (2015) 1–28.

[13] J. Douglas and H.H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, *Transactions of the American Mathematical Society* 82 (1956) 421–439.

[14] J. Eckstein, Some saddle-function splitting methods for convex programming, *Optimization Methods and Software* 4 (1994) 75–83.

[15] J. Eckstein, Splitting methods for monotone operators with applications to parallel optimization, *PhD Thesis*, Massachusetts Institute of Technology, 1989.

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

[17] J. Eckstein and D. P. Bertsekas, An alternating direction method for linear programming, *MIT. Laboratory for Information, and Decision Systems*, 1990.

[18] J. Eckstein and B.F. Svaiter, General projective splitting Methods for sums of maximal monotone operators, *SIAM Journal on Control and Optimization* 48 (2009) 787–811.

[19] D. Gabay, Chapter IX applications of the method of multipliers to variational inequalities, *Studies in mathematics and its applications* 15 (1983) 299–331.

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

[21] R. Glowinski and A. Marrocco, Sur l'approximation par èlèments finis et la rèsolution par pènalisation-dualitè d'une classe de problèmes de Dirichlet non linèaires, RAIRO, R-2, 1975, pp.41–76.

[22] R. Goebel and R. T. Rockafellar, Local strong convexity and local Lipschitz continuity of the gradient of convex functions, *Journal of Convex Analysis* 15 (2008) 263–270.

[23] D. Goldfarb and S.Q. Ma, Fast multiple splitting algorithms for convex optimization, *SIAM Journal on Optimization* 22 (2012) 533-556.

[24] D. Goldfarb and S.Q. Ma and K. Scheinberg, Fast alternating linearization methods for minimizing the sum of two convex functions, *Mathematical Programming* 141(2013) 349–382.

[25] T. Goldstein, B. O'Donoghue, S. Setzer and R. Baraniuk, Fast alternating direction optimization methods, *SIAM Journal on Imaging Sciences* 7 (2014) 1588–1623.

[26] D.R. Han and X.M. Yuan, Local linear convergence of the alternating direction method of multipliers for quadratic programs, *SIAM Journal on Numerical Analysis* 51 (2013) 3446–3457.

[27] D.R. Han, X.M. Yuan and W.X. Zhang, An augmented-Lagrangian-based parallel splitting method for separable convex programming with applications to image processing, *Mathematics of Computation* 83 (2014) 2263–2291.

[28] B.S. He, L.Z. Liao, D.R. Han and H. Yang, A new inexact alternating directions method for monotone variational inequalities, *Mathematical Programming* 92 (2002) 103–118.

[29] B.S. He, M. Tao and X.M. Yuan, Alternating direction method with Gaussian back substitution for separable convex programming, *SIAM Journal on Optimization* 22 (2012) 313–340.

[30] B.S. He, M. Tao and X.M. Yuan, Convergence rate and iteration complexity on the alternating direction method of multipliers with a substitution procedure for separable convex programming, *Mathematics of Operations Research*, under revision, 2015.

[31] B.S. He, M.H. Xu and X. M. Yuan, Solving large-scale least squares semidefinite programming by alternating direction methods, *SIAM Journal on Matrix Analysis and Applications* 32 (2011) 136–152.

[32] B.S. He and X.M. Yuan, On the O(1/n) convergence rate of the Douglas-Rachford alternating direction method, *SIAM Journal on Numerical Analysis* 50 (2012) 700–709.

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

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

[35] S.Q. Ma, Alternating proximal gradient method for convex minimization, preprint, 2012.

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

[37] S.Q. Ma, L.Z. Xue and H. Zou, Alternating direction methods for latent variable gaussian graphical model selection, *Neural computation* 25 (2013) 2172–2198.

[38] M.K. Ng, P. Weiss and X. M. Yuan, Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods, *SIAM Journal on Scientific Computing* 32 (2010) 2710–2736.

[39] R. T. Rockafellar, Convex analysis, *Princeton University Press*, Princeton, 1996.

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

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

[42] R. T. Rockafellar and R. J-B. Wets, *Variational analysis: Groundlehren der mathematischen wissenschaften*, Springer Verlag, Berlin, 1998.

[43] J. E. Spingarn, Applications of the method of partial inverses to convex programming: decomposition, *Mathematical Programming* 32 (1985) 199-223.

[44] J. E. Spingarn, Partial inverse of a monotone operator, *Applied Mathematics and Optimization* 10 (1983) 247–265.

[45] M. Tao and X.M. Yuan, Recovering low-rank and sparse components of matrices from incomplete and noisy observations, *SIAM Journal on Optimization* 21(2011) 57–81.

[46] X.F. Wang and X.M. Yuan, The linearized alternating direction method of multipliers for Dantzig Selector, *SIAM Journal on Scientific Computing* 34 (2012) 2792–2811.

[47] L.Z. Xue, S.Q. Ma and H. Zou, Positive definite L_1 penalized estimation of large covariance matrices, *Journal of the American Statistical Association* 107 (2012) 1480–1491.

[48] J.F. Yang and Y. Zhang, Alternating direction algorithms for L_1-problems in compressive sensing, *SIAM Journal on Scientific Computing* 33 (2011) 250–278.

[49] X.M. Yuan, Alternating direction method for covariance selection models, *Journal of Scientific Computing* 51 (2012) 261–273.

[50] X.Q. Zhang, M. Burger and S. Osher, A unified primal-dual algorithm framework based on Bregman iteration, *Journal of Scientific Computing* 46 (2011) 20–46.

Manuscript received 12 February 2014
revised 6 April 2014
accepted for publication 6 April 2014

Xiangfeng Wang
Shanghai Key Laboratory of Trustworthy Computing
Software Engineering Institute, East China Normal University
Shanghai 200062, China
E-mail address: xfwang@sei.ecnu.edu.cn

Mingyi Hong
Department of Industrial and Manufacturing Systems Engineering
Iowa State University, Ames, 50011, USA
E-mail address: mingyi@iastate.edu

Shiqian Ma
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
E-mail address: sqma@se.cuhk.edu.hk

Zhi-Quan Luo
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, 55455, USA
E-mail address: luozq@umn.edu