A Unified Alternating Direction Method of Multipliers by Majorization Minimization

Canyi Lu, *Student Member, IEEE*, Jiashi Feng, Shuicheng Yan, *Fellow, IEEE*, and Zhouchen Lin, *Senior Member, IEEE*

Abstract—Accompanied with the rising popularity of compressed sensing, the Alternating Direction Method of Multipliers (ADMM) has become the most widely used solver for linearly constrained convex problems with separable objectives. In this work, we observe that many existing ADMMs update the primal variable by minimizing different majorant functions with their convergence proofs given case by case. Inspired by the principle of majorization minimization, we respectively present the unified frameworks of Gauss-Seidel ADMMs and Jacobian ADMMs, which use different historical information for the current updating. Our frameworks generalize previous ADMMs to solve the problems with non-separable objectives. We also show that ADMMs converge faster when the used majorant function is tighter. We then propose the Mixed Gauss-Seidel and Jacobian ADMM (M-ADMM) which alleviates the slow convergence issue of Jacobian ADMMs by absorbing merits of the Gauss-Seidel ADMMs. M-ADMM can be further improved by backtracking and wise variable partition. We also propose to solve the multi-blocks problems by Proximal Gauss-Seidel ADMM which is of the Gauss-Seidel type. It convegences for non-strongly convex objective. Experiments on both synthesized and real-world data demonstrate the superiority of our new ADMMs. Finally, we release a toolbox that implements efficient ADMMs for many problems in compressed sensing.

Index Terms—Unified frameworks of ADMM, mixed ADMM, majorization minimization, convex optimization

1 Introduction

This work aims to solve the following convex problem

$$\min_{\mathbf{x}} f(\mathbf{x}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n), \ \text{s.t.} \ \mathbf{A}\mathbf{x} = \sum_{i=1}^{n} \mathbf{A}_i\mathbf{x}_i = \mathbf{b}, \quad (1)$$

where $f : \mathbb{R}^{p_1 \times \cdots \times p_n} \to \mathbb{R}$ is convex and n (≥ 2) denotes the block number of variables. We denote $\mathbf{x} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ with $\mathbf{x}_i \in \mathbb{R}^{p_i}$, and $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_n]$ with $\mathbf{A}_i \in \mathbb{R}^{d \times p_i}$. Problem (1) has drawn increasing attention recently for the emerging applications of compressive sensing in computer vision and signal processing, e.g., sparsity based face recognition [41], saliency detection [38], motion segmentation [11], [26], [30], image denoising [23], video denoising [19], texture repairing [21] and many others [5], [18], [29], [42], [43].

To solve (1), the popular Augmented Lagrangian Method (ALM) [16] updates the primal variable $\mathbf{x}$ by

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^k, \beta^{(k)}) = \arg\min_{\mathbf{x}} f(\mathbf{x}) + r^k(\mathbf{x}), \quad (2)$$

where $\mathcal{L}$ is the augmented Lagrangian function defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \beta) = f(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathbf{A}\mathbf{x} - \mathbf{b} \rangle + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2,$$

and

$$r^k(\mathbf{x}) = \frac{\beta^{(k)}}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2. \quad (3)$$

Then the dual variable $\boldsymbol{\lambda}$ is updated to minimize $-\mathcal{L}$ by gradient descent with the step size $\beta^{(k)}$, i.e.,

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}). \quad (4)$$

However, (2) may not be easily solvable, since r^k is non-separable. The Alternating Direction Method of Multipliers (ADMM) [12] instead solves (2) inexactly by updating $\mathbf{x}_i$'s in an alternating way and thus the per-iteration cost can be much lower. Many variants of ADMM have been proposed by using different properties of f and $\mathbf{A}$. We will review the most related works in Section 1.1, and claim our contributions in Section 1.2.

Notations. The ℓ_2-norm of a vector and Frobenius norm of a matrix are denoted as $\|\cdot\|$. The spectral norm and the smallest singular value of a matrix $\mathbf{A}$ are denoted as $\|\mathbf{A}\|_2$ and $\sigma_{\min}(\mathbf{A})$, respectively. The identity matrix is denoted as $\mathbf{I}$ without specifying its size. The all-one vector is denoted as $\mathbf{1}$. We denote $\mathbb{S}$ and $\mathbb{S}_+$ as the set of symmetry and positive semidefinite matrices respectively and define $\langle \boldsymbol{a}, \boldsymbol{a} \rangle_{\mathbf{A}} = \|\boldsymbol{a}\|_{\mathbf{A}}^2 = \boldsymbol{a}^\top \mathbf{A}\boldsymbol{a}$ for $\mathbf{A} \in \mathbb{S}$. If $\mathbf{A} - \mathbf{B}$ is positive semi-definite, then we denote $\mathbf{A} \succeq \mathbf{B}$. The block diagonal matrix $\mathrm{Diag}\{\mathbf{A}_i, i = 1, \ldots, n\}$ has $\mathbf{A}_i$ as its ith block on the diagonal. A function $f : \mathbb{R}^p \to \mathbb{R}$ is said to be L-smooth (or ∇f is Lipschitz continuous), if

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p. \quad (5)$$

- *C. Lu, J. Feng, and S. Yan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077. E-mail: canyilu@gmail.com, {elefjia, eleyans}@nus.edu.sg.*
- *Z. Lin is with the Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, Beijing Shi 100000, China, and the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Minhang Qu 200240, China. E-mail: zlin@pku.edu.cn.*

Manuscript received 7 July 2016; revised 25 Jan. 2017; accepted 15 Mar. 2017. Date of publication 28 Mar. 2017; date of current version 13 Feb. 2018. Recommended for acceptance by K. Weinberger.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2017.2689021

1.1 Review of ADMMs

Most of ADMMs are only able to solve (1) with separable f; i.e., there exist f_i's such that $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$. They can be categorized into Gauss-Seidel ADMMs and Jacobian ADMMs. The Gauss-Seidel ADMMs update $\mathbf{x}_i$'s in a sequential way, i.e., update $\mathbf{x}_i^{k+1}$ by fixing others as their latest versions, while the Jacobian ADMMs update $\mathbf{x}_i$'s in a parallel way, i.e., update each $\mathbf{x}_i^{k+1}$ by fixing $\mathbf{x}_j = \mathbf{x}_j^k$, for all $j \neq i$. We review these two types of ADMMs respectively. The difference between ADMMs lies in the updating of $\mathbf{x}_i$'s, while $\boldsymbol{\lambda}$ is updated in the same way by (4).

Gauss-Seidel ADMMs solve (1) with $n = 2$ blocks. The standard ADMM [2] solves (2) inexactly by updating $\mathbf{x}_1$ and $\mathbf{x}_2$ in a sequential way, i.e.,

$$\begin{aligned}\mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} \mathcal{L}([\mathbf{x}_1; \mathbf{x}_2^k], \boldsymbol{\lambda}^k, \beta^{(k)}) \\ &= \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) + r_1^k(\mathbf{x}_1),\end{aligned} \tag{6}$$

$$\begin{aligned}\mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} \mathcal{L}([\mathbf{x}_1^{k+1}; \mathbf{x}_2], \boldsymbol{\lambda}^k, \beta^{(k)}) \\ &= \arg\min_{\mathbf{x}_2} f_2(\mathbf{x}_2) + r_2^k(\mathbf{x}_2),\end{aligned} \tag{7}$$

where

$$r_1^k(\mathbf{x}_1) = \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2^k - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2, \tag{8}$$

$$r_2^k(\mathbf{x}_2) = \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_1\mathbf{x}_1^{k+1} + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2. \tag{9}$$

By using different properties of f_1 and $\mathbf{A}_1$, $\mathbf{x}_1$ (the same discussion is also applicable to $\mathbf{x}_2$) can be updated more efficiently than solving (6). If f_1 is L_1-smooth, then $\mathbf{x}_1$ can be updated by

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} \hat{f}_1(\mathbf{x}_1) + r_1^k(\mathbf{x}_1), \tag{10}$$

where $\hat{f}_1(\mathbf{x}_1) = f(\mathbf{x}_1^k) + \langle \nabla f_1(\mathbf{x}_1^k), \mathbf{x}_1 - \mathbf{x}_1^k\rangle + \frac{L_1}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k\|^2$. The motivation is that $\hat{f}_1$ is a majorant (upper bound) function of f_1, i.e., $\hat{f}_1 \geq f_1$ [1]. If $f_1 = g_1 + h_1$, where g_1 is convex and h_1 is convex and L_1-smooth, then $\mathbf{x}_1$ can be updated by (10) with $\hat{f}_1(\mathbf{x}_1) = g(\mathbf{x}_1) + h(\mathbf{x}_1^k) + \langle \nabla h_1(\mathbf{x}_1^k), \mathbf{x}_1 - \mathbf{x}_1^k\rangle + \frac{L_1}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k\|^2$. In this case, $\hat{f}_1 \geq f_1$. We name the method using (10) as Proximal ADMM (P-ADMM) for these two cases. Similar techniques have been used in [1], [36].

If the columns of $\mathbf{A}_1$ are not orthogonal, solving (6) is usually very expensive especially when f_1 is nonsmooth. Then Linearized ADMM (L-ADMM) [6], [24] instead updates $\mathbf{x}_1$ by

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) + \hat{r}_1^k(\mathbf{x}_1), \tag{11}$$

where $\hat{r}_1^k(\mathbf{x}_1) = r_1^k(\mathbf{x}_1^k) + \langle \nabla r_1^k(\mathbf{x}_1^k), \mathbf{x}_1 - \mathbf{x}_1^k\rangle + \frac{\eta_1}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k\|^2$ with $\eta_1 > \|\mathbf{A}_1\|_2^2$. Note that $\hat{r}_1^k \geq r_1^k$ since r_1^k is $\|\mathbf{A}_1\|_2^2$-smooth. For some nonsmooth f_1, e.g., the ℓ_1-norm, (11) can be solved efficiently with a closed form solution.

If f_1 is a sum of a nonsmooth function and an L_1-smooth function, then we can simultaneously use the majorant function $\hat{f}_1$ of f_1 as P-ADMM and $\hat{r}_1^k$ of r_1^k as L-ADMM. Thus $\hat{f}_1 + \hat{r}_1^k \geq f_1 + r_1^k$. This motivates the Proximal Linearized ADMM (PL-ADMM) which updates $\mathbf{x}_1$ by

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} \hat{f}_1(\mathbf{x}_1) + \hat{r}_1^k(\mathbf{x}_1). \tag{12}$$

For (1) with $n > 2$ blocks of variables, the naive extension of Gauss-Seidel ADMMs may diverge [3]. To address this issue, several Jacobian ADMMs have been proposed by using different properties of f_i and $\mathbf{A}_i$. The Linearized ADMM with Parallel Splitting (L-ADMM-PS) [28] solves (2) inexactly by linearizing r^k in (3) at $\mathbf{x}_i^k$'s and updates $\mathbf{x}_i$'s in parallel by

$$\begin{aligned}\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i}\ & f_i(\mathbf{x}_i) + \left\langle \mathbf{A}_i^\top(\beta^{(k)}(\mathbf{A}\mathbf{x}^k - \mathbf{b}) + \boldsymbol{\lambda}^k), \mathbf{x}_i\right\rangle \\ &+ \frac{\beta^{(k)}\eta_i}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2,\end{aligned} \tag{13}$$

where $\eta_i > n\|\mathbf{A}_i\|_2^2$. Proximal Jacobian ADMM (Prox-JADMM), a more general method in [10], updates $\mathbf{x}_i$'s in parallel by

$$\begin{aligned}\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} & f_i(\mathbf{x}_i) + \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_i\mathbf{x}_i + \sum_{j\neq i}\mathbf{A}_j\mathbf{x}_j^k - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2 \\ &+ \frac{\beta^{(k)}}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{\mathbf{G}_i}^2,\end{aligned} \tag{14}$$

where $\mathbf{G}_i \succ (n-1)\mathbf{A}_i^\top\mathbf{A}_i$. Actually (13) is a special case of (14) when $\mathbf{G}_i = \eta_i\mathbf{I} - \mathbf{A}_i^\top\mathbf{A}_i$ with $\eta_i > n\|\mathbf{A}_i\|_2^2$. If $f_i = g_i + h_i$, where g_i is convex and h_i is convex and L_i-smooth, then the Proximal Linearized ADMM with Parallel Splitting (PL-ADMM-PS) [23] updates $\mathbf{x}_i$'s in parallel by

$$\begin{aligned}\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i}\ & \hat{f}_i(\mathbf{x}_i) + \left\langle \mathbf{A}_i^\top(\beta^{(k)}(\mathbf{A}\mathbf{x}^k - \mathbf{b}) + \boldsymbol{\lambda}^k), \mathbf{x}_i\right\rangle \\ &+ \frac{\beta^{(k)}\eta_i}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2,\end{aligned} \tag{15}$$

where $\hat{f}_i(\mathbf{x}_i) = g(\mathbf{x}_i) + h(\mathbf{x}_i^k) + \langle \nabla h_i(\mathbf{x}_i^k), \mathbf{x}_i - \mathbf{x}_i^k\rangle + \frac{L_i}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2$ and $\eta_i > n\|\mathbf{A}_i\|_2^2$. As we will show later, the updating rules (14) and (15) are equivalent to minimizing different majorant functions of $f(\mathbf{x}) + r^k(\mathbf{x})$ in (2).

For the convergence guarantee, all the above ADMMs own the convergence rate $O(1/K)$ [6], [14], [23], [28] ($o(1/K)$ in [10] for Prox-JADMM), where K is the number of iterations. There are also some other works which consider different special cases of our problem (1) and give different convergence rates of ADMMs. For example, the works [13], [31] propose fast ADMMs with better convergence rate. But their considered problems are quite specific and their convergence guarantees require several additional assumptions. For (1) with separable objective and $n > 2$, the works [17], [22] prove the convergence of the naive multi-blocks extension of ADMM under various assumptions, e.g., full column rank of $\mathbf{A}_i$, strong convexity or Lipschitz continuity of some f_i and some others which may be hard to be verified in practice. The work [40] reformulates the multi-blocks problem into a two-block one by variable splitting and solves it by ADMM. But it is verified to be slower than Prox-JADMM in [10] since there are many more number of variables.

1.2 Contributions

From the above discussions, we observe that different ADMMs can be regarded as variants of inexact ALM in the sense that the primal variable $\mathbf{x}^{k+1}$ in ADMMs is updated by solving (2) in ALM approximately. This actually slows the convergence, but the per-iteration cost is lower. So there is a trade-off between the exactness of the subproblem optimization and the convergence speed. In practice, we balance both to choose the proper solver. Generally, if f is not very simple, e.g., sum of several nonsmooth functions, ADMMs are much more efficient than ALM. ADMMs use two main

techniques for approximation and update $\mathbf{x}^{k+1}$ in an easier way than ALM: Alternating Minimization (AM) and Majorization Minimization (MM) [20]. AM, which updates one block each time when fixing others, makes the subproblems easier to solve. For example, the updating of $[\mathbf{x}_1^{k+1}; \mathbf{x}_2^{k+1}]$ in ADMM (6) and (7) is easier than the one in ALM (2). But the cost of the one block updating may be still high and it can be further reduced by using MM, which minimizes a majorant function instead of the original objective to find an approximate solution. For example, as reviewed in Section 1.1, different Gauss-Seidel ADMMs update $\mathbf{x}_1$ by minimizing different majorant functions of the objective in standard ADMM (6), while different Jacobian ADMMs update $\mathbf{x}_i$'s by minimizing different majorant functions of the objective in ALM (2). Actually, Gauss-Seidel ADMMs first use AM and then apply MM to update each block, while Jacobian ADMMs first use MM and then AM to update each block (though this is equivalent to updating all blocks simultaneously). Besides the primal variables, the dual variable $\boldsymbol{\lambda}^{k+1}$ updating in (4) is also equivalent to minimizing a majorant function of $-\mathcal{L}(\mathbf{x}^{k+1}, \boldsymbol{\lambda}, \beta^{(k)})$, i.e.,

$$\boldsymbol{\lambda}^{k+1} = \arg\min_{\boldsymbol{\lambda}} -\mathcal{L}\big(\mathbf{x}^{k+1}, \boldsymbol{\lambda}, \beta^{(k)}\big) + \frac{1}{2\beta^{(k)}} \big\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^k\big\|^2. \quad (16)$$

These observations suggest that MM provides a new insight to interpret ADMMs. The convergences of ADMMs which use different majorant functions are guaranteed, but they are proved case by case. It is not clear what is the role of MM in ADMMs. Another issue is that, in practice, one can develop many ADMMs for the same problem. But it is generally difficult to see which one converges faster. The proved same rate $O(1/K)$ in the worst case fails to characterize the different speeds of ADMMs in practice. We lack practical principles and guidelines for designing efficient ADMMs. In this work, we raise several crucial questions:

1. What kind of majorant functions can be used in ADMMs?
2. Is is possible to give a unified convergence analysis of ADMMs which use different majorant functions?
3. What is the connection between the convergence speed of ADMMs and the used majorant functions?
4. How to choose the proper majorant functions for designing efficient ADMMs?
5. For (1) with $n > 2$, does there any Gauss-Seidel type ADMM converge without the strongly convex objective assumption?

In this work, we show many interesting findings about ADMMs through the lens of MM. We aim to address the above questions and in particular we make the following contributions. First, for a multivariable function f, we propose the majorant first-order surrogate function $\hat{f}$, which requires three conditions to be satisfied: majorization, proximity and separability. The first two guarantee that $\hat{f}$ is a reasonable approximation of f, while the last one makes the minimizing of $\hat{f}$ easy. Note that the objective f in (1) can be non-separable since we only need to minimize $\hat{f}$. Second, we present the unified frameworks of Gauss-Seidel ADMMs and Jacobian ADMMs based on our majorant first-order surrogate and give the unified convergence guarantee. They not only draw connections with existing ADMMs, but also extend them to solve new problems with non-separable objective. Third, we show that the bound which measures the convergence speed of ADMMs depends on the tightness of the used majorant function. The tighter, the faster. This explains our previous intuitive observation that ADMMs converge faster when (2) in ALM is solved more accurately. Fourth, we develop several useful techniques to tighten the majorant surrogates and thus improve the efficiency of ADMMs. Consider (1) with $n > 2$, we propose the Mixed Gauss-Seidel and Jacobian ADMM (M-ADMM) algorithm. It divides n blocks of variables into two super blocks, and then updates them in a sequential way as Gauss-Seidel ADMMs, while the variables in each super block are updated in a parallel way as Jacobian ADMMs. M-ADMM takes the structure of $\mathbf{A}$, e.g., $\frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ that may be partially separable, into account to compute a tighter majorant surrogate, while previous Jacobian ADMMs fail to do so. In addition, we show how to partition n blocks of variables into two super blocks wisely, which is crucial in the efficient implementation of ADMMs. Fifth, we propose to solve problem (1) with $n > 2$ blocks by Proximal Gauss-Seidel ADMM (Prox-GSADMM) and prove that it converges without the strongly convex objective assumption. To the best of our knowledge, this is the first convergent Gauss-Seidel type ADMM for such a problem. The last contribution is the developed toolbox which implements efficient ADMMs for many popular problems in compressed sensing. See https://github.com/canyilu/LibADMM. Though there are already many toolboxes in compressed sensing, the solved problems are more or less limited due to the applicability of the used solvers, e.g., SPAMS [34] and SLEP [27] focus more on sparse models and non-constrained problems. We instead focus on the constrained problem (1), which is much more general. See a list of problems in our toolbox in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2017.2689021.

2 Majorant First-Order Surrogate of a Multivariable Function

In this section, we propose the majorant first-order surrogate of the multivariable functions which enjoy some "good" properties.

Definition 1 (Lipschitz Continuity). *Let $f : \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_n} \to \mathbb{R}$ be differentiable. Then ∇f is called Lipschitz continuous if there exist $\mathbf{L}_i \succeq \mathbf{0}, i = 1, \ldots, n$, such that*

$$|f(\mathbf{x}) - f(\mathbf{y}) - \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle| \le \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{y}_i\|_{\mathbf{L}_i}^2, \quad (17)$$

for any $\mathbf{x} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ and $\mathbf{y} = [\mathbf{y}_1; \ldots; \mathbf{y}_n]$ with $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{p_i}$. In this case, we say that f is $\{\mathbf{L}_i\}_{i=1}^n$-smooth.

The Lipschitz continuity of the multivariable function is crucial in this work. It is different from the single variable case defined in (5). For $n = 1$, (17) holds if (5) holds (Lemma 1.2.3 in [35]), but not vice versa. This motivates the above definition.

Definition 2 (Strong Convexity). *Let $f : \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_n} \to \mathbb{R}$ and $g(\mathbf{x}) = f(\mathbf{x}) - \frac{1}{2}\sum_{i=1}^n \|\mathbf{x}_i - \mathbf{y}_i\|_{\mathbf{P}_i}^2$ be convex for any $\mathbf{y}_i$. If $\mathbf{P}_i \succeq 0$, we say that f is $\{\mathbf{P}_i\}_{i=1}^n$-convex. If $\mathbf{P}_i \succ 0$, we say that f is $\{\mathbf{P}_i\}_{i=1}^n$-strongly convex.*

Definition 3 (Majorant First-Order Surrogate). *A function $\hat{f} : \mathbb{R}^{p_1} \times \cdots \times \mathbb{R}^{p_n} \to \mathbb{R}$ is a majorant first-order*

surrogate of $f:\mathbb{R}^{p_1}\times\cdots\times\mathbb{R}^{p_n}\to\mathbb{R}$ *near* $\boldsymbol{\kappa}=[\boldsymbol{\kappa}_1;\ldots;\boldsymbol{\kappa}_n]$ *with* $\boldsymbol{\kappa}_i\in\mathbb{R}^{p_i}$ *when the following conditions are satisfied:*

- Majorization: $\hat{f}$ *is a majorant function of* f*, i.e.,* $\hat{f}(\mathbf{x})\geq f(\mathbf{x})$ *for any* $\mathbf{x}$.
- Proximity: *there exists* $\mathbf{L}_i\succeq\mathbf{0}$ *such that the approximation error* $h(\mathbf{x}):=\hat{f}(\mathbf{x})-f(\mathbf{x})$ *satisfies*

$$|h(\mathbf{x})|\leq\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2. \tag{18}$$

- Separability: $\hat{f}$ *is separable w.r.t.* $\mathbf{x}_i$*'s; i.e., there exist* $\hat{f}_i$*'s such that* $\hat{f}(\mathbf{x})=\sum_{i=1}^{n}\hat{f}_i(\mathbf{x}_i)$.

We denote by $\mathcal{S}_{\{\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$ *the set of such surrogates, and by* $\mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$ *the subset of* $\{\mathbf{P}_i\}_{i=1}^n$*-convex surrogates.*

In MM, one aims to find an approximate solution to $\min_{\mathbf{x}}f(\mathbf{x})$ by solving $\min_{\mathbf{x}}\hat{f}(\mathbf{x})$, which is easier. To this end, the above three conditions on $\hat{f}$ look reasonable. Majorization guarantees that $f(\mathbf{x})$ tends to be minimized when $\hat{f}(\mathbf{x})$ is minimized. Proximity means that $\hat{f}(\mathbf{x})$ cannot be too loose and this guarantees a controllable approximation to $f(\mathbf{x})$. The separability makes the optimization on $\hat{f}(\mathbf{x})$ easier than $f(\mathbf{x})$, which can be non-separable. This is important for multi-blocks optimization.

Note that $\mathbf{L}_i$ measures the difference $\hat{f}-f$, or the tightness of the majorant surrogate $\hat{f}$. If $\|\mathbf{L}_i\|_2$ is smaller, then the majorant surrogate is tighter. This plays an important role in this work.

Lemma 1. *If the approximation error* $h(\mathbf{x})=\hat{f}(\mathbf{x})-f(\mathbf{x})$ *satisfies the following* Smoothness *assumption, i.e.,*

$$h(\mathbf{x})\text{ is }\{\mathbf{L}_i\}_{i=1}^n\text{-smooth, }h(\boldsymbol{\kappa})=0\text{ and }\nabla h(\boldsymbol{\kappa})=0, \tag{19}$$

then the Proximity *assumption in (18) holds.*

Lemma 1 can be obtained by using (17) for h at $\boldsymbol{\kappa}$. Lemma 1 is useful to verify the Proximity assumption. Some widely used majorant first-order surrogates are (see Lemma 5 in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2017.2689021:

- *Proximal Surrogates.* Let f be a separable function. Define $\hat{f}(\mathbf{x})=f(\mathbf{x})+\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2$, where $\mathbf{L}_i\succeq 0$. Then, $\hat{f}\in\mathcal{S}_{\{2\mathbf{L}_i,\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. If f is $\{\mathbf{Q}_i\}_{i=1}^n$-strongly convex, $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{L}_i+\mathbf{Q}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$.
- *Lipschitz Gradient Surrogates.* Let f be $\{\mathbf{L}_i\}_{i=1}^n$-smooth. Define $\hat{f}(\mathbf{x})=f(\boldsymbol{\kappa})+\langle\nabla f(\boldsymbol{\kappa}),\mathbf{x}-\boldsymbol{\kappa}\rangle+\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2$. Then, $\hat{f}\in\mathcal{S}_{\{2\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. If f is convex, $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. If f is $\{\mathbf{Q}_i\}_{i=1}^n$-strongly convex, $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i-\mathbf{Q}_i,\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$.
- *Proximal Gradient Surrogates.* Let $f=f_1+f_2$, where f_1 is $\{\mathbf{L}_i\}_{i=1}^n$-smooth. Define $\hat{f}(\mathbf{x})=f_1(\boldsymbol{\kappa})+\langle\nabla f_1(\boldsymbol{\kappa}),\mathbf{x}-\boldsymbol{\kappa}\rangle+\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2+f_2(\mathbf{x})$. If f_1 and f_2 are convex, $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. Then, $\hat{f}\in\mathcal{S}_{\{2\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. If f_1 is $\{\mathbf{Q}_i\}_{i=1}^n$-strongly convex and f_2 is convex, $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i-\mathbf{Q}_i,\mathbf{L}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$.

If $\mathbf{L}_i=0$, the proximal surrogates reduce to $\hat{f}=f$. Some other examples of majorant functions, e.g., Jensen surrogates, can be found in [33]. Also, the positive linear combination of majorant surrogates is still a majorant surrogate. See more discussions in the Appendix, available online.

Lemma 2 (Key Property of the Majorant First-Order Surrogate). *Let* $\hat{f}\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\boldsymbol{\kappa})$. *Then, we have*

$$\begin{aligned}&f(\mathbf{x})+\langle\mathbf{u},\mathbf{y}-\mathbf{x}\rangle-f(\mathbf{y})\\&\quad\leq\frac{1}{2}\sum_{i=1}^{n}\left(\|\mathbf{y}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2-\|\mathbf{y}_i-\mathbf{x}_i\|_{\mathbf{P}_i}^2\right),\ \forall\mathbf{x},\mathbf{y},\end{aligned} \tag{20}$$

where $\mathbf{u}\in\partial\hat{f}(\mathbf{x})$ *is any subgradient of the convex function* $\hat{f}$. *If* f *is* $\{\mathbf{Q}\}_{i=1}^n$*-convex, we have*

$$\langle\mathbf{u}-\mathbf{v},\mathbf{x}-\mathbf{y}\rangle\geq\frac{1}{2}\sum_{i=1}^{n}\left(\|\mathbf{y}_i-\mathbf{x}_i\|_{\mathbf{P}_i+\mathbf{Q}_i}^2-\|\mathbf{y}_i-\boldsymbol{\kappa}_i\|_{\mathbf{L}_i}^2\right), \tag{21}$$

where $\mathbf{u}\in\partial\hat{f}(\mathbf{x})$ *and* $\mathbf{v}\in\partial f(\mathbf{y})$.

The majorant first-order surrogate given in Definition 3 is motivated by [33]. However, they have many key differences:

1. Our majorant surrogate is defined based on the multivariable function and it is much more general than the single variable case in [33]. The Lipschitz continuity of the multivariable function is different; the *Separability* of $\hat{f}$ is new.
2. For approximation error $h=\hat{f}-f$, we use the *Proximity* assumption in (18) which is less restrictive than the *Smoothness* assumption in (19). We only require the error h to be bounded, and it needs not necessarily be smooth.
3. The properties in Lemma 2 are new and they play a central role in our convergence analysis. Lemma 2.1 in [33] also introduces some properties of the majorant first-order surrogate. But their bounds are too loose and are not applicable to our proofs due to the constraint of (1) considered in this work.
4. The considered constrained problem in this work is different from the non-constrained problem in [33]. When proving Proposition 2.3 in [33], they use a key property $f(\mathbf{x}^{k+1})\leq f(\mathbf{x}^k)$, while this does not hold in ADMMs.

At the end of this section, we discuss some properties of $\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2$ which are important for in ADMMs.

Lemma 3. *Let* $r(\mathbf{x})=\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2$, *where* $\mathbf{x}=[\mathbf{x}_1;\ldots;\mathbf{x}_n]$, $\mathbf{A}=[\mathbf{A}_1,\ldots,\mathbf{A}_n]$ *and* $\mathbf{b}$ *are of compatible sizes. We have*

(1) $r(\mathbf{x})$ *is* $\{\mathbf{L}_i'\}_{i=1}^n$*-smooth. The choice of* $\mathbf{L}_i'$ *depends on the structure of* $\mathbf{A}$.

(2) $r(\mathbf{x})\leq\hat{r}(\mathbf{x})$, *where*

$$\begin{aligned}\hat{r}(\mathbf{x})=&\frac{1}{2}\sum_{i=1}^{n}\left\|\mathbf{A}_i\mathbf{x}_i+\sum_{j\neq i}\mathbf{A}_j\mathbf{y}_j-\mathbf{b}\right\|^2\\&+\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i-\mathbf{y}_i\|_{\mathbf{G}_i}^2+\frac{1-n}{2}\|\mathbf{A}\mathbf{y}-\mathbf{b}\|^2,\end{aligned} \tag{22}$$

for any $\mathbf{y}=[\mathbf{y}_1;\ldots;\mathbf{y}_n]$ *and* $\mathbf{G}_i\succeq\mathbf{L}_i'-\mathbf{A}_i^\top\mathbf{A}_i$.

(3) If $\mathbf{G}_i=\eta_i\mathbf{I}-\mathbf{A}_i^\top\mathbf{A}_i$ *with* $\eta_i\geq\|\mathbf{L}_i'\|_2$, *(22) reduces to*

$$\begin{aligned}\hat{r}(\mathbf{x})=&\sum_{i=1}^{n}\left\langle\mathbf{x}_i-\mathbf{y}_i,\mathbf{A}_i^\top(\mathbf{A}\mathbf{y}-\mathbf{b})\right\rangle\\&+\sum_{i=1}^{n}\frac{\eta_i}{2}\|\mathbf{x}_i-\mathbf{y}_i\|^2+\frac{1}{2}\|\mathbf{A}\mathbf{y}-\mathbf{b}\|^2.\end{aligned} \tag{23}$$

The choice of $\mathbf{L}_i'$ guarantees that $\hat{r}\geq r$. To make $\hat{r}$ tight, we expect that $\|\mathbf{L}_i'\|_2$ can be as small as possible. Generally, there

are two interesting choices: (1) $\mathbf{L}_i' = \|\mathbf{A}\|_2^2\mathbf{I}$; (2) $\mathbf{L}_i' = n\mathbf{A}_i^\top\mathbf{A}_i$. When considering $\|\mathbf{L}_i'\|_2$, it is not clear which choice of $\mathbf{L}_i'$ has smaller $\|\mathbf{L}_i'\|_2$. The second choice of $\mathbf{L}_i'$ is widely used in Jacobian ADMMs. It explains the choice of $\eta_i > \|\mathbf{L}_i'\|_2 = n\|\mathbf{A}_i\|_2^2$ in L-ADMM-PS (13). However, such a choice of $\mathbf{L}_i'$ may not be good since it does not make fully use of the structure of $\mathbf{A}$, and thus $\hat{r}$ may not be a tight surrogate of r. For example, let $\mathbf{A}_1 = [\mathbf{C}_1; \mathbf{0}]$, $\mathbf{A}_2 = [\mathbf{C}_2; \mathbf{0}]$, $\mathbf{A}_3 = [\mathbf{0}; \mathbf{C}_3]$, $\mathbf{A}_4 = [\mathbf{0}; \mathbf{C}_4]$, and $\mathbf{b} = [\mathbf{b}_1; \mathbf{b}_2]$ of compatible sizes. Then $r(\mathbf{x}) = \frac{1}{2}\|\sum_{i=1}^2 \mathbf{C}_i\mathbf{x}_i - \mathbf{b}_1\|^2 + \frac{1}{2}\|\sum_{i=3}^4 \mathbf{C}_i\mathbf{x}_i - \mathbf{b}_2\|^2$. We can choose $\mathbf{L}_i' = 2\mathbf{A}_i^\top\mathbf{A}_i$, which is much better than $4\mathbf{A}_i^\top\mathbf{A}_i$. Actually, the choice of $\mathbf{L}_i'$ depends on the separability of r. In practice, it is easy to compute $\mathbf{L}_i'$ when given $\mathbf{A}$. A good choice of $\mathbf{L}_i'$ gives a tight surrogate $\hat{r}$, and this may significantly improve the efficiency of Jacobian ADMMs (see Section 4).

3 Unified Gauss-Seidel ADMMs

In this section, we consider solving (1) with $n = 2$ blocks by a unified framework of Gauss-Seidel ADMMs. In the $(k+1)$th iteration, we compute the majorant surrogate $\hat{f}^k$ of f near $\mathbf{x}^k$, i.e., $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^2}(f, \mathbf{x}^k)$ and $\hat{f}^k$ is separable, i.e., $\hat{f}^k(\mathbf{x}) = \hat{f}_1^k(\mathbf{x}_1) + \hat{f}_2^k(\mathbf{x}_2)$. For r_1^k and r_2^k in (8) and (9), we construct their proximal surrogates respectively as follows[1]

$$\hat{r}_1^k(\mathbf{x}_1) = r_1^k(\mathbf{x}_1) + \frac{\beta^{(k)}}{2}\|\mathbf{x}_1 - \mathbf{x}_1^k\|_{\mathbf{G}_1}^2, \tag{24}$$

$$\hat{r}_2^k(\mathbf{x}_2) = r_2^k(\mathbf{x}_2) + \frac{\beta^{(k)}}{2}\|\mathbf{x}_2 - \mathbf{x}_2^k\|_{\mathbf{G}_2}^2, \tag{25}$$

where $\mathbf{G}_1 \succeq \mathbf{0}$ and $\mathbf{G}_2 \succ \mathbf{0}$. Then we update $\mathbf{x}_1$ and $\mathbf{x}_2$ by

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} \hat{f}_1^k(\mathbf{x}_1) + \hat{r}_1^k(\mathbf{x}_1), \tag{26}$$

$$\mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2} \hat{f}_2^k(\mathbf{x}_2) + \hat{r}_2^k(\mathbf{x}_2). \tag{27}$$

Finally, $\boldsymbol{\lambda}$ is updated by (4). This leads to the unified framework of Gauss-Seidel ADMMs, as shown in Algorithm 1.

Note that in Algorithm 1, f is not necessarily separable. In this case, our algorithm and the convergence guarantee shown later are completely new. If f is already separable, then the objectives in (26) and (27) are majorant surrogates of the ones in (6) and (7), respectively. Many previous Gauss-Seidel ADMMs are special cases by using different majorant surrogates $\hat{f}_1$ and $\hat{r}_1^k$ (depending on $\mathbf{G}_1^k$) in Algorithm 1. See Table 1 for a summary.

Assume that there exists at least one KKT point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ of problem (1), i.e., $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ and $-\mathbf{A}^\top\boldsymbol{\lambda}^* \in \partial f(\mathbf{x}^*)$. Previous works prove that ADMMs converge to the KKT point at the rate $O(1/K)$ (K is the number of iterations) in different ways. The works [14], [36] give the same rate of ADMM, L-ADMM, and P-ADMM. But they require that both the primal and dual feasible sets should be bounded. The work [23] removes the above assumptions and shows that the convergence rates of L-ADMM-PS and PL-ADMM-PS are

$$f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \langle\mathbf{A}^\top\boldsymbol{\lambda}^*, \bar{\mathbf{x}}^K - \mathbf{x}^*\rangle + \frac{\alpha}{2}\|\mathbf{A}\bar{\mathbf{x}}^K - \boldsymbol{b}\|^2 \le O(1/K), \tag{28}$$

1. Note that the definitions of $\hat{r}_i^k$ in Sections 3, 4, and 5 are different.

TABLE 1
Previous Gauss-Seidel ADMMs Are Special Cases of Algorithm 1 with Different $\hat{f}_1$ and $\mathbf{G}_1$

	$\hat{f}_1^k(\mathbf{x}_1)$	$\mathbf{G}_1$
ADMM [12]	$f_1(\mathbf{x}_1)$	$\mathbf{0}$
P-ADMM [36]	Lipschitz Gradient Surrogate or Proximal Gradient Surrogate	$\mathbf{0}$
L-ADMM [6]	$f_1(\mathbf{x}_1)$	$\eta_1\mathbf{I} - \mathbf{A}_1^\top\mathbf{A}_1$
PL-ADMM	Lipschitz Gradient Surrogate or Proximal Gradient Surrogate	$\eta_1\mathbf{I} - \mathbf{A}_1^\top\mathbf{A}_1$

In this table, $\eta_1 > \|\mathbf{A}_1\|_2^2$.

where $\bar{\mathbf{x}}^K$ is a weighted sum of $\mathbf{x}^{k\prime}$s and $\alpha > 0$. Now we give the convergence bound of Algorithm 1 as (28).

Algorithm 1. A Unified Framework of Gauss-Seidel ADMMs

For $k = 0, 1, 2, \ldots$ **do**
1. Compute a majorant first-order surrogate $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^2}(f, \mathbf{x}^k)$ with $\hat{f}^k(\mathbf{x}) = \hat{f}_1^k(\mathbf{x}_1) + \hat{f}_2^k(\mathbf{x}_2)$.
2. Update $\mathbf{x}_1$ by solving (26).
3. Update $\mathbf{x}_2$ by solving (27).
4. Update $\boldsymbol{\lambda}$ by $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b})$.
5. Choose $\beta^{(k+1)}$ such that $\beta^{(k)} \le \beta^{(k+1)} \le \beta_{\max}$.

end

Theorem 1. *In Algorithm 1, assume that* $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^2}(f, \mathbf{x}^k)$ *with* $\mathbf{P}_i \succeq \mathbf{L}_i \succeq \mathbf{0}$, $i = 1, 2$, $\mathbf{G}_1 \succeq \mathbf{0}$ *in (24), and* $\mathbf{G}_2 \succ \mathbf{0}$ *in (25). For any* $K > 0$, *let* $\bar{\mathbf{x}}^K = \sum_{k=0}^K \gamma^{(k)}\mathbf{x}^{k+1}$ *with* $\gamma^{(k)} = (\beta^{(k)})^{-1}/\sum_{k=0}^K(\beta^{(k)})^{-1}$. *Then*

$$f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \langle\mathbf{A}^\top\boldsymbol{\lambda}^*, \bar{\mathbf{x}}^K - \mathbf{x}^*\rangle + \frac{\beta^{(0)}\alpha}{2}\|\mathbf{A}\bar{\mathbf{x}}^K - \mathbf{b}\|^2 \le \frac{\sum_{i=1}^2 \|\mathbf{x}_i^* - \mathbf{x}_i^0\|_{\mathbf{H}_i^0}^2 + \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^0\|_{\mathbf{H}_3^0}^2}{2\sum_{k=0}^K\left(\beta^{(k)}\right)^{-1}}, \tag{29}$$

where $\alpha = \min\{\frac{1}{2}, \frac{\sigma_{\min}^2(\mathbf{G}_2)}{2\|\mathbf{A}_2\|_2^2}\}$, $\mathbf{H}_1^0 = \frac{1}{\beta^{(0)}}\mathbf{L}_1 + \mathbf{G}_1$, $\mathbf{H}_2^0 = \frac{1}{\beta^{(0)}}\mathbf{L}_2 + \mathbf{A}_2^\top\mathbf{A}_2 + \mathbf{G}_2$, *and* $\mathbf{H}_3^0 = (1/\beta^{(0)})^2\mathbf{I}$.

Consider $\mathbf{H}_i^0$, $i = 1, 2$, at the RHS of (29), it can be seen that they depend on $\mathbf{L}_i$ and $\mathbf{G}_i$, which control the difference $\hat{f} - f$ and $\hat{r}_i^k - r_i^k$, respectively. This suggests a faster convergence when using tighter majorant surrogates, though the convergence rate of Gauss-Seidel ADMMs in Algorithm 1 is $O(1/K)$ when $\beta^{(k)}$'s are bounded.

Note that the assumption $\mathbf{G}_2 \succ \mathbf{0}$ guarantees that $\alpha > 0$. Such an assumption is also used in [14], [36] which prove the same convergence rate in different ways. It suggests that using $\mathbf{G}_2 \succ \mathbf{0}$ instead of $\mathbf{G}_2 = \mathbf{0}$ in the traditional ADMM can achieve the $O(1/K)$ convergence rate.

4 Unified Jacobian ADMMs

In this section, we consider solving (1) with $n > 2$ by a unified framework of Jacobian ADMMs. The motivation is to solve (2) inexactly by minimizing a majorant surrogate of $f(\mathbf{x}) + r^k(\mathbf{x})$. In the $(k+1)$th iteration, we first compute the majorant surrogate of f near $\mathbf{x}^k$, i.e., $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$, and $\hat{f}^k$ is separable, $\hat{f}^k(\mathbf{x}) = \sum_{i=1}^n \hat{f}_i^k(\mathbf{x}_i)$. Assume that $\frac{1}{2}\|\mathbf{A}\mathbf{x}\|^2$ is

TABLE 2
Previous Jacobian ADMMs Are Special Cases of Algorithm 2 with Different $\hat{f}_i$ and $\mathbf{G}_i$

	$\hat{f}_i^k(\mathbf{x}_i)$	$\mathbf{G}_i$
L-ADMM-PS [28]	$f_i(\mathbf{x}_i)$	$\eta_i\mathbf{I}-\mathbf{A}_i^\top\mathbf{A}_i$,
PL-ADMM-PS [23]	Proximal Gradient Surrogate	$\eta_i\mathbf{I}-\mathbf{A}_i^\top\mathbf{A}_i$
Prox-JADMM [10]	$f_i(\mathbf{x}_i)$	$\succ(n-1)\mathbf{A}_i^\top\mathbf{A}_i$

In this table, $\eta_i > n\|\mathbf{A}_i\|_2^2$.

$\{\mathbf{L}_i'\}_{i=1}^n$-smooth. For r^k in (2), we define its majorant surrogate $\hat{r}^k$ by using (22), i.e., $\hat{r}^k(\mathbf{x})=\sum_{i=1}^n \hat{r}_i^k(\mathbf{x}_i)$, where

$$\frac{\hat{r}_i^k(\mathbf{x}_i)}{\beta^{(k)}}=\frac{1}{2}\left\|\mathbf{A}_i\mathbf{x}_i+\sum_{j\neq i}\mathbf{A}_j\mathbf{x}_j^k-\mathbf{b}+\frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2 +\frac{1}{2}\|\mathbf{x}_i-\mathbf{x}_i^k\|_{\mathbf{G}_i}^2+c_i^k, \tag{30}$$

with $\mathbf{G}_i\succ\mathbf{L}_i'-\mathbf{A}_i^\top\mathbf{A}_i$ and c_i^k's are constants satisfying $\sum_{i=1}^n c_i^k=\frac{1-n}{2}\|\mathbf{A}\mathbf{x}^k-\mathbf{b}\|^2$. Thus $\hat{f}^k(\mathbf{x})+\hat{r}^k(\mathbf{x})$ is a majorant surrogate of $f(\mathbf{x})+r^k(\mathbf{x})$ in (2). Now we minimize $\hat{f}^k(\mathbf{x})+\hat{r}^k(\mathbf{x})$ instead to update $\mathbf{x}$, i.e.,

$$\mathbf{x}^{k+1}=\arg\min_{\mathbf{x}}\ \hat{f}^k(\mathbf{x})+\hat{r}^k(\mathbf{x}). \tag{31}$$

Note that both $\hat{f}$ and $\hat{r}^k$ are separable. Thus solving (31) is equivalent to updating each $\mathbf{x}_i$ in parallel, i.e.,

$$\mathbf{x}_i^{k+1}=\arg\min_{\mathbf{x}_i}\ \hat{f}_i^k(\mathbf{x}_i)+\hat{r}_i^k(\mathbf{x}_i). \tag{32}$$

Finally $\boldsymbol{\lambda}$ is updated by (4). This leads to the unified framework of Jacobian ADMMs, as shown in Algorithm 2.

If f is non-separable, then our algorithm and convergence guarantee shown later are completely new. If f is separable, several previous Jacobian ADMMs are special cases by using different majorant surrogates $\hat{f}_i$ and $\hat{r}_i^k$ (depending on $\mathbf{G}_i$) in Algorithm 2. See Table 2 for a summary.

Algorithm 2. A Unified Framework of Jacobian ADMMs

For $k=0,1,2,\ldots$ **do**
1. Compute a majorant first-order surrogate $\hat{f}^k\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\mathbf{x}^k)$ with $\hat{f}^k(\mathbf{x})=\sum_{i=1}^n\hat{f}_i^k(\mathbf{x}_i)$.
2. Update $\mathbf{x}_i$, $i=1,\ldots,n$, in parallel by solving (32).
3. Update $\boldsymbol{\lambda}$ by $\boldsymbol{\lambda}^{k+1}=\boldsymbol{\lambda}^k+\beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1}-\mathbf{b})$.
4. Choose $\beta^{(k+1)}$ such that $\beta^{(k)}\le\beta^{(k+1)}\le\beta_{\max}$.

end

Theorem 2. *In Algorithm 2, assume that* $\hat{f}^k\in\mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\mathbf{x}^k)$ *with* $\mathbf{P}_i\succeq\mathbf{L}_i\succeq\mathbf{0}$, $\frac{1}{2}\|\mathbf{A}\mathbf{x}\|^2$ *is* $\{\mathbf{L}_i'\}_{i=1}^n$*-smooth, and* $\mathbf{G}_i\succ\mathbf{L}_i'-\mathbf{A}_i^\top\mathbf{A}_i$ *in (30). For any* $K>0$, *let* $\bar{\mathbf{x}}^K=\sum_{k=0}^K\gamma^{(k)}\mathbf{x}^{k+1}$ *with* $\gamma^{(k)}=(\beta^{(k)})^{-1}/\sum_{k=0}^K(\beta^{(k)})^{-1}$. *Then*

$$f(\bar{\mathbf{x}}^K)-f(\mathbf{x}^*)+\langle\mathbf{A}^\top\boldsymbol{\lambda}^*,\bar{\mathbf{x}}^K-\mathbf{x}^*\rangle+\frac{\beta^{(0)}\alpha}{2}\|\mathbf{A}\bar{\mathbf{x}}^K-\mathbf{b}\|^2 \le\frac{\sum_{i=1}^n\|\mathbf{x}_i^*-\mathbf{x}_i^0\|_{\mathbf{H}_i^0}^2+\|\boldsymbol{\lambda}^*-\boldsymbol{\lambda}^0\|_{\mathbf{H}_{n+1}^0}^2}{2\sum_{k=0}^K\left(\beta^{(k)}\right)^{-1}}, \tag{33}$$

where $\alpha=\min\{\frac{1}{2},\frac{\sigma_{\min}^2(\mathrm{Diag}\{\mathbf{A}_i^\top\mathbf{A}_i+\mathbf{G}_i,i=1,\ldots,n\}-\mathbf{A}^\top\mathbf{A})}{2\|\mathbf{A}\|_2^2}\}$, $\mathbf{H}_i^0=\frac{1}{\beta^{(0)}}\mathbf{L}_i+\mathbf{A}_i^\top\mathbf{A}_i+\mathbf{G}_i$, $i=1,\ldots,n$, *and* $\mathbf{H}_{n+1}^0=\left(1/\beta^{(0)}\right)^2\mathbf{I}$.

The above bound implies an interesting connection between the convergence speed and the tightness of the majorant surrogates. For simplicity, let $\beta^{(k)}=\beta$. Then (33) reduces to

$$\frac{\sum_{i=1}^n\|\mathbf{x}_i^*-\mathbf{x}_i^0\|_{\beta\mathbf{H}_i^0}^2+\frac{1}{\beta}\|\boldsymbol{\lambda}^*-\boldsymbol{\lambda}^0\|^2}{2(K+1)} \le\frac{\frac{1}{2}\sum_{i=1}^n\|\mathbf{x}_i^*-\mathbf{x}_i^0\|_{\mathbf{L}_i+\beta\mathbf{L}_i'}^2+\frac{1}{2\beta}\|\boldsymbol{\lambda}^*-\boldsymbol{\lambda}^0\|^2}{(K+1)}, \tag{34}$$

where (34) uses $\mathbf{G}_i\succ\mathbf{L}_i'-\mathbf{A}_i^\top\mathbf{A}_i$. Now consider the two constant terms in the numerator of (34). The first term controls the tightness of the used majorant surrogate for the $\mathbf{x}$ updating, i.e., $|\hat{f}^0(\mathbf{x}^*)+\hat{r}^0(\mathbf{x}^*)-f(\mathbf{x}^*)-r(\mathbf{x}^*)|\le\frac{1}{2}\sum_{i=1}^n\|\mathbf{x}_i^*-\mathbf{x}_i^0\|_{\mathbf{L}_i+\beta\mathbf{L}_i'}^2$, which uses (18) with $\mathbf{x}=\mathbf{x}^*$ and $k=0$. The second term is actually the difference function $\frac{1}{2\beta}\left\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^k\right\|^2$ between $-\mathcal{L}(\mathbf{x}^{k+1},\boldsymbol{\lambda},\beta^{(k)})$ and its majorant surrogate in (16) when $\boldsymbol{\lambda}=\boldsymbol{\lambda}^*$ and $k=0$. So the convergence bound depends on the tightness of the used majorant surrogates for both the primal and dual variables updating. If $\hat{f}^k+\hat{r}^k$ is tighter (associated to the $\mathbf{x}$ updating) or β is larger (associated to the $\boldsymbol{\lambda}$ updating), the algorithm converges faster. In practice, ADMMs stop based on certain criteria induced by the KKT conditions. If β is relatively large, the algorithm seems to converge faster but the objective function value may be larger. How to choose the best β or $\beta^{(k)}$ is still an open issue. In this work, we focus the discussion on how to improve the tightness of the majorant surrogate for the primal variable updating.

Note that Algorithm 2 improves previous Jacobian ADMMs which use $\mathbf{L}_i'=n\mathbf{A}_i^\top\mathbf{A}_i$. Such a choice of $\mathbf{L}_i'$ does not fully use the structure of $\mathbf{A}$ (see the discussions after Lemma 3). In this work, we have a new choice $\mathbf{L}_i'=\|\mathbf{A}\|_2^2\mathbf{I}$. In practice, one may use the one which has smaller $\left\|\mathbf{L}_i'\right\|_2$. The reason behind is that $\mathbf{L}_i'$'s control the tightness of $\hat{r}^k(\mathbf{x})$. The tighter surrogate $\hat{r}^k(\mathbf{x})$ will make the algorithm converges faster. In Section 5, we discuss how to further improve the tightness of $\hat{r}^k(\mathbf{x})$ by introducing alternating minimization in Jacobian ADMMs and the backtracking technique.

5 Mixed Gauss-Seidel and Jacobian ADMM

Consider solving (1) with $n=2$ by Gauss-Seidel ADMMs and Jacobian ADMMs, the former one will converge faster. The reason is that Jacobian ADMMs require $\mathbf{G}_i\succ\mathbf{A}_i^\top\mathbf{A}_i$, while Gauss-Seidel ADMMs only require $\mathbf{G}_i\succ\mathbf{0}$. Thus the bound in (29) is expected to be tighter than the one in (33). The superiority of Gauss-Seidel ADMMs over Jacobian ADMMs is that the former first use alternating minimization to reduce the complexity of the problem (fewer variables) and then the used majorant surrogate can be tighter when using majorization minimization.

In this section, we consider (1) with $n>2$ blocks. We propose the Mixed Gauss-Seidel and Jacobian ADMM (M-ADMM), which introduces the alternating minimization before using majorization minimization. M-ADMM first divides these n blocks $\mathbf{x}=[\mathbf{x}_1;\ldots;\mathbf{x}_n]$ into two super blocks, i.e., $\mathbf{x}_{B_1}=[\mathbf{x}_i,i\in B_1]$ with n_1 blocks of variables, and $\mathbf{x}_{B_2}=[\mathbf{x}_i,i\in B_2]$ with n_2 blocks of variables, where B_1 and B_2 satisfy $B_1\cap B_2=\varnothing$ and $B_1\cup B_2=\{1,\ldots,n\}$. Then $\mathbf{x}_{B_1}$ and $\mathbf{x}_{B_2}$ are updated in a sequential way as Gauss-Seidel ADMMs, while $\mathbf{x}_i$'s in each super block are updated in

a parallel way as Jacobian ADMMs. As shown later, M-ADMM owns a tighter bound than (33), and thus it will be faster than Jacobian ADMMs. In the following, we first introduce M-ADMM, and then discuss the variable partition and backtracking technique which are crucial for the efficient implementation of M-ADMM in practice.

5.1 M-ADMM

Assume that we are given a partition of n blocks, denoted as $\{B_1, B_2\}$. We accordingly partition $\mathbf{A}$ into $\mathbf{A}_{B_1} = [\mathbf{A}_i, i \in B_1]$ and $\mathbf{A}_{B_2} = [\mathbf{A}_i, i \in B_2]$. Then (1) is equivalent to

$$\min_{\mathbf{x}_{B_1}, \mathbf{x}_{B_2}} f(\mathbf{x}), \quad \text{s.t.} \quad \mathbf{A}_{B_1}\mathbf{x}_{B_1} + \mathbf{A}_{B_2}\mathbf{x}_{B_2} = \mathbf{b}. \tag{35}$$

In the $(k+1)$th iteration, we first compute the majorant surrogate of f near $\mathbf{x}^k$, i.e., $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i, \mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$, and $\hat{f}^k$ is separable, $\hat{f}^k(\mathbf{x}) = \hat{f}^k_{B_1}(\mathbf{x}_{B_1}) + \hat{f}^k_{B_2}(\mathbf{x}_{B_2})$, where $\hat{f}^k_{B_i}(\mathbf{x}_{B_i}) = \sum_{j\in B_i} \hat{f}^k_j(\mathbf{x}_j)$, $i = 1, 2$. Then (35) can be solved by updating $\mathbf{x}_{B_1}$ and $\mathbf{x}_{B_2}$ as the traditional ADMM, i.e.,

$$\mathbf{x}^{k+1}_{B_1} = \arg\min_{\mathbf{x}_{B_1}} \hat{f}^k_{B_1}(\mathbf{x}_{B_1}) + r^k_{B_1}(\mathbf{x}_{B_1}), \tag{36}$$

$$\mathbf{x}^{k+1}_{B_2} = \arg\min_{\mathbf{x}_{B_2}} \hat{f}^k_{B_2}(\mathbf{x}_{B_2}) + r^k_{B_2}(\mathbf{x}_{B_2}), \tag{37}$$

where

$$r^k_{B_1}(\mathbf{x}_{B_1}) = \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_{B_1}\mathbf{x}_{B_1} + \mathbf{A}_{B_2}\mathbf{x}^k_{B_2} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2,$$

and

$$r^k_{B_2}(\mathbf{x}_{B_2}) = \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_{B_1}\mathbf{x}^{k+1}_{B_1} + \mathbf{A}_{B_2}\mathbf{x}_{B_2} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2.$$

However, the above problems are expensive to solve since they may not be separable w.r.t. $\mathbf{x}_i$'s in B_1 or B_2. Assume that $\frac{1}{2}\|\mathbf{A}_{B_1}\mathbf{x}_{B_1}\|^2$ is $\{\mathbf{L}'_i\}_{i\in B_1}$-smooth. By (22), we construct a majorant surrogate $\hat{r}^k_{B_1}$ of $r^k_{B_1}$ near $\mathbf{x}^k_{B_1}$, i.e., $\hat{r}^k_{B_1}(\mathbf{x}_{B_1}) = \sum_{i\in B_1}\hat{r}^k_i(\mathbf{x}_i)$, where

$$\begin{aligned}\frac{\hat{r}^k_i(\mathbf{x}_i)}{\beta^{(k)}} =& \frac{1}{2}\left\|\mathbf{A}_i\mathbf{x}_i + \sum_{\substack{j\in B_1\\ j\neq i}}\mathbf{A}_j\mathbf{x}^k_j + \mathbf{A}_{B_2}\mathbf{x}^k_{B_2} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2 \\ &+ \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}^k_i\|^2_{\mathbf{G}_i} + c^k_i,\ i \in B_1,\end{aligned} \tag{38}$$

with $\mathbf{G}_i \succeq \mathbf{L}'_i - \mathbf{A}^T_i\mathbf{A}_i$, $i \in B_1$, and c^k_i's satisfying $\sum_{i\in B_1} c^k_i = \frac{1-n_1}{2}\|\mathbf{A}\mathbf{x}^k - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\|^2$. Similarly, assume that $\frac{1}{2}\|\mathbf{A}_{B_2}\mathbf{x}_{B_2}\|^2$ is $\{\mathbf{L}'_i\}_{i\in B_2}$-smooth. Then a majorant surrogate $\hat{r}^k_{B_2}$ of $r^k_{B_2}$ near $\mathbf{x}^k_{B_2}$ is $\hat{r}^k_{B_2}(\mathbf{x}_{B_2}) = \sum_{i\in B_2}\hat{r}^k_i(\mathbf{x}_i)$, where

$$\begin{aligned}\frac{\hat{r}^k_i(\mathbf{x}_i)}{\beta^{(k)}} =& \frac{1}{2}\left\|\mathbf{A}_i\mathbf{x}_i + \sum_{\substack{j\in B_2\\ j\neq i}}\mathbf{A}_j\mathbf{x}^k_j + \mathbf{A}_{B_1}\mathbf{x}^{k+1}_{B_1} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2 \\ &+ \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}^k_i\|^2_{\mathbf{G}_i} + c^k_i,\ i \in B_2,\end{aligned} \tag{39}$$

with $\mathbf{G}_i \succ \mathbf{L}'_i - \mathbf{A}^\top_i\mathbf{A}_i$, $i \in B_2$, and c^k_i's satisfying $\sum_{i\in B_2} c^k_i = \frac{1-n_2}{2}\|\mathbf{A}_{B_1}\mathbf{x}^{k+1}_{B_1} + \mathbf{A}_{B_2}\mathbf{x}^k_{B_2} - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\|^2$. By replacing $r^k_{B_1}(\mathbf{x}_{B_1})$ and $r^k_{B_2}(\mathbf{x}_{B_2})$ with their majorant surrogates $\hat{r}^k_{B_1}(\mathbf{x}_{B_1})$ and $\hat{r}^k_{B_2}(\mathbf{x}_{B_2})$ in (36) and (37) respectively, we update $\mathbf{x}_{B_1}$ and $\mathbf{x}_{B_2}$ by

$$\mathbf{x}^{k+1}_{B_1} = \arg\min_{\mathbf{x}_{B_1}} \hat{f}^k_{B_1}(\mathbf{x}_{B_1}) + \hat{r}^k_{B_1}(\mathbf{x}_{B_1}),$$

$$\mathbf{x}^{k+1}_{B_2} = \arg\min_{\mathbf{x}_{B_2}} \hat{f}^k_{B_2}(\mathbf{x}_{B_2}) + \hat{r}^k_{B_2}(\mathbf{x}_{B_2}).$$

Note that the above two problems are separable for each $\mathbf{x}_i$ in B_1 and B_2. They are respectively equivalent to

$$\mathbf{x}^{k+1}_i = \arg\min_{\mathbf{x}_i} \hat{f}^k_i(\mathbf{x}_i) + \hat{r}^k_i(\mathbf{x}_i),\ i \in B_1, \tag{40}$$

$$\mathbf{x}^{k+1}_i = \arg\min_{\mathbf{x}_i} \hat{f}^k_i(\mathbf{x}_i) + \hat{r}^k_i(\mathbf{x}_i),\ i \in B_2. \tag{41}$$

Finally $\boldsymbol{\lambda}$ is updated by (4). This leads to the Mixed Gauss-Seidel and Jacobian ADMM (M-ADMM), as shown in Algorithm 3. Now we give its convergence bound as (28).

Algorithm 3. Mixed Gauss-Seidel and Jacobian ADMM

For $k = 0, 1, 2, \ldots$ **do**
1. Compute a majorant first-order surrogate $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i, \mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$ with $\hat{f}^k(\mathbf{x}) = \sum_{i=1}^n \hat{f}^k_i(\mathbf{x}_i)$.
2. For all $i \in B_1$, update $\mathbf{x}_i$'s in parallel by solving (40).
3. For all $i \in B_2$, update $\mathbf{x}_i$'s in parallel by solving (41).
4. Update $\boldsymbol{\lambda}$ by $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b})$.
5. Choose $\beta^{(k+1)}$ such that $\beta^{(k)} \le \beta^{(k+1)} \le \beta_{\max}$.

end

Theorem 3. *In Algorithm 3, assume that* $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i, \mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$ *with* $\mathbf{P}_i \succeq \mathbf{L}_i \succeq \mathbf{0}$, $\frac{1}{2}\|\mathbf{A}_{B_1}\mathbf{x}_{B_1}\|^2$ *is* $\{\mathbf{L}'_i\}_{i\in B_1}$*-smooth,* $\frac{1}{2}\|\mathbf{A}_{B_2}\mathbf{x}_{B_2}\|^2$ *is* $\{\mathbf{L}'_i\}_{i\in B_2}$*-smooth,* $\mathbf{G}_i \succeq \mathbf{L}'_i - \mathbf{A}^\top_i\mathbf{A}_i$, $i \in B_1$ *in (38) and* $\mathbf{G}_i \succ \mathbf{L}'_i - \mathbf{A}^\top_i\mathbf{A}_i$, $i \in B_2$ *in (39). For any* $K > 0$, *let* $\bar{\mathbf{x}}^K = \sum_{k=0}^K \gamma^{(k)}\mathbf{x}^{k+1}$ *with* $\gamma^{(k)} = (\beta^{(k)})^{-1}/\sum_{k=0}^K (\beta^{(k)})^{-1}$. *Then*

$$\begin{aligned}&f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \langle \mathbf{A}^\top\boldsymbol{\lambda}^*, \bar{\mathbf{x}}^K - \mathbf{x}^*\rangle + \frac{\beta^{(0)}\alpha}{2}\|\mathbf{A}\bar{\mathbf{x}}^K - \mathbf{b}\|^2 \\ &\le \frac{\sum_{j=1}^2 \|\mathbf{x}^*_{B_j} - \mathbf{x}^0_{B_j}\|^2_{\mathbf{H}^0_j} + \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^0\|^2_{\mathbf{H}^0_3}}{2\sum_{k=0}^K (\beta^{(k)})^{-1}},\end{aligned} \tag{42}$$

where $\alpha = \min\left\{\frac{1}{2}, \frac{\sigma^2_{\min}\left(\mathrm{Diag}\{\mathbf{A}^\top_i\mathbf{A}_i + \mathbf{G}_i, i\in B_2\} - \mathbf{A}^\top_{B_2}\mathbf{A}_{B_2}\right)}{2\|\mathbf{A}_{B_2}\|_2^2}\right\}$, $\mathbf{H}^0_1 = \mathrm{Diag}\{\frac{1}{\beta^{(0)}}\mathbf{L}_i + \mathbf{A}^\top_i\mathbf{A}_i + \mathbf{G}_i, i \in B_1\} - \mathbf{A}^\top_{B_1}\mathbf{A}_{B_1}$, $\mathbf{H}^0_2 = \mathrm{Diag}\{\frac{1}{\beta^{(0)}}\mathbf{L}_i + \mathbf{A}^\top_i\mathbf{A}_i + \mathbf{G}_i, i \in B_2\}$, and $\mathbf{H}^0_3 = (1/\beta^{(0)})^2\mathbf{I}$.

M-ADMM in Algorithm 3 further unifies Gauss-Seidel ADMMs in Algorithm 1 and Jacobian ADMMs in Algorithm 2. If $n = 2$, let $B_1 = \{1\}$ and $B_2 = \{2\}$. Then M-ADMM degenerates into the Gauss-Seidel ADMMs, and (42) reduces to (29). If $n > 2$, let $B_1 = \varnothing$ and $B_2 = \{1, \ldots, n\}$. Then M-ADMM degenerates into the Jacobian ADMMs, and (42) reduces to (33). More importantly, for the case of $n > 2$ and other choices of B_1 and B_2, M-ADMM will be faster than Jacobian ADMMs, since the right hand side of (42) may be much smaller than the one of (33). This is due to their different choices of $\mathbf{G}_i$. If we choose $\mathbf{L}'_i = n\mathbf{A}^\top_i\mathbf{A}_i$, Jacobian ADMMs require $\mathbf{G}_i \succ (n-1)\mathbf{A}^\top_i\mathbf{A}_i$ for all $i = 1, \ldots, n$, while M-ADMM only requires $\mathbf{G}_i \succ (n_1 - 1)\mathbf{A}^\top_i\mathbf{A}_i$ for $i \in B_1$ and $\mathbf{G}_i \succ (n_2 - 1)\mathbf{A}^\top_i\mathbf{A}_i$ for $i \in B_2$. Note that $n = n_1 + n_2$. The improvement benefits from the sequential updating rules of $\mathbf{x}_{B_1}$ and $\mathbf{x}_{B_2}$ by using tighter majorant surrogates in M-ADMM. Indeed, M-ADMM only needs to majorize $r^k_{B_1}(\mathbf{x}_{B_1})$ in (36) and $r^k_{B_2}(\mathbf{x}_{B_2})$ in (37) for

$\mathbf{x}_{B_1}$ and $\mathbf{x}_{B_2}$ respectively, while Jacobian ADMMs need to majorize $r^k(\mathbf{x})$ in (3) for all $\mathbf{x}_i$'s simultaneously.

Note that the block-wise ADMM in [15] and Hybrid ADMM (H-ADMM) in [9] share a similar mixed Gauss-Seidel and Jacobian updating scheme as our M-ADMM. The block-wise ADMM is a special case of our M-ADMM by taking $\mathbf{G}_i = \tau_i \mathbf{A}_i^\top \mathbf{A}_i$, where $\tau_i > n_1 - 1$, $i \in B_1$ and $\tau_i > n_2 - 1$, $i \in B_2$. The limitation of such a choice of $\mathbf{G}_i$ is that its subproblems may not be easy to solve especially when f_i is non-smooth. Also, their convergence analysis requires more restrictive assumption, i.e., each $\mathbf{A}_i$ has full column rank. H-ADMM is also a special case of our M-ADMM by taking $\mathbf{G}_i \succ \|\mathbf{A}_{B_1}\|_2^2\mathbf{I}$, $i \in B_1$ and $\mathbf{G}_i \succ \|\mathbf{A}_{B_2}\|_2^2\mathbf{I}$, $i \in B_2$. This is not as tight as ours since we can choose $\mathbf{G}_i \succ \|\mathbf{A}_{B_1}\|_2^2\mathbf{I} - \mathbf{A}_i^\top \mathbf{A}_i$, $i \in B_1$ and $\mathbf{G}_i \succ \|\mathbf{A}_{B_2}\|_2^2\mathbf{I} - \mathbf{A}_i^\top \mathbf{A}_i$, $i \in B_2$. Generally, M-ADMM is more general when considering the choices of $\hat{f}$, $\mathbf{G}^k$ and $\beta^{(k)}$. The backtracking technique and wise variable partition introduced below will further improve the convergence speed of M-ADMM. More importantly, we conclude that M-ADMM is generally faster than Jacobian ADMMs due to the used tighter majorant surrogates while block-wise ADMM and H-ADMM have no such a result and support in theory.

5.2 M-ADMM with Backtracking

We have given the convergence guarantee of M-ADMM when fixing $\mathbf{G}_i$. In practice, we can estimate it by the backtracking technique which will lead to tighter majorant surrogate. The effectiveness has been verified in first-order optimization [1]. Now, we introduce the backtracking technique into M-ADMM.

Algorithm 4. M-ADMM with Backtracking

Initialization: $k = 0$, $\mathbf{x}_i^k$, $\mathbf{G}_i^k \succ \mathbf{0}$, $\boldsymbol{\lambda}^k$, $\beta^{(k)} > 0$, $\tau > 0$, $\mu > 1$.
For $k = 0, 1, 2, \ldots$ **do**
1. Compute a majorant first-order surrogate $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i, \mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$ with $\hat{f}^k(\mathbf{x}) = \sum_{i=1}^n \hat{f}_i^k(\mathbf{x}_i)$.
2. Set $\mathbf{G}_i = \mathbf{G}_i^k$ and compute $\mathbf{x}_i^{k+1}$ by (40)-(41).
3. If (43) does not hold, set $\mathbf{G}_i^k = \mu \mathbf{G}_i^k$, $i \in B_1$. Go to 2).
   If (45) does not hold, set $\mathbf{G}_i^k = \mu \mathbf{G}_i^k$, $i \in B_2$. Go to 2).
4. Update $\boldsymbol{\lambda}$ by $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b})$.
5. Choose $\beta^{(k+1)}$ such that $\beta^{(k)} \le \beta^{(k+1)} \le \beta_{\max}$.
6. Set $\mathbf{G}_i^{k+1} = \mathbf{G}_i^k$, $i = 1, \ldots, n$.

end

To guarantee the convergence, $\mathbf{G}_i$ can be replaced by $\mathbf{G}_i^k$ such that $r_{B_1}^k(\mathbf{x}_{B_1}^{k+1}) \le \hat{r}_{B_1}^k(\mathbf{x}_{B_1}^{k+1})$ and $r_{B_2}^k(\mathbf{x}_{B_2}^{k+1}) \le \hat{r}_{B_2}^k(\mathbf{x}_{B_2}^{k+1})$. They are guaranteed when

$$\left\|\mathbf{A}_{B_1}(\mathbf{x}_{B_1}^{k+1} - \mathbf{x}_{B_1}^k)\right\|^2 \le \sum_{i \in B_1} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|_{\mathbf{G}_i^k + \mathbf{A}_i^\top \mathbf{A}_i}^2, \quad (43)$$

$$\left\|\mathbf{A}_{B_2}(\mathbf{x}_{B_2}^{k+1} - \mathbf{x}_{B_2}^k)\right\|^2 \le \sum_{i \in B_2} \left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|_{\mathbf{G}_i^k + \mathbf{A}_i^\top \mathbf{A}_i}^2. \quad (44)$$

To achieve the $O(1/K)$ convergence rate, we replace (44) as

$$\tau\left\|\mathbf{x}_{B_2}^{k+1} - \mathbf{x}_{B_2}^k\right\|^2 \le \left\|\mathbf{x}_{B_2}^{k+1} - \mathbf{x}_{B_2}^k\right\|_{\mathbf{K}_2^k - \mathbf{A}_{B_2}^\top \mathbf{A}_{B_2}}^2, \quad (45)$$

for some small constant $\tau > 0$ and $\mathbf{K}_2^k = \mathrm{Diag}\{\mathbf{A}_i^\top \mathbf{A}_i + \mathbf{G}_i^k, i \in B_2\}$. In this case, we may be able to find $\mathbf{G}_i^k$ with relatively smaller $\|\mathbf{G}_i^k\|_2$, and thus $\hat{r}_{B_1}^k(\mathbf{x}_{B_1}^{k+1})$ and $\hat{r}_{B_2}^k(\mathbf{x}_{B_2}^{k+1})$ are tighter upper bounds of $r_{B_1}^k(\mathbf{x}_{B_1}^{k+1})$ and $r_{B_2}^k(\mathbf{x}_{B_2}^{k+1})$, respectively. This leads to a better approximate solution and improves the efficiency. We summarize M-ADMM with backtracking in Algorithm 4. Note that Step 3) will only be performed for finitely many times. Similarly, the convergence guarantee is given as follows.

Theorem 4. *In Algorithm 4, assume that* $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i, \mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$ *with* $\mathbf{P}_i \succeq \mathbf{L}_i \succeq \mathbf{0}$. *Then (42) holds with* $\mathbf{H}_1^0 = \mathrm{Diag}\{\frac{1}{\beta^{(0)}}\mathbf{L}_i + \mathbf{A}_i^\top \mathbf{A}_i + \mathbf{G}_i^0, i \in B_1\} - \mathbf{A}_{B_1}^\top \mathbf{A}_{B_1}$, $\mathbf{H}_2^0 = \mathrm{Diag}\{\frac{1}{\beta^{(0)}}\mathbf{L}_i + \mathbf{A}_i^\top \mathbf{A}_i + \mathbf{G}_i^0, i \in B_2\}$, $\mathbf{H}_3^0 = \left(1/\beta^{(0)}\right)^2 \mathbf{I}$, *and* $\alpha = \min\{\frac{1}{2}, \frac{\tau}{2\|\mathbf{A}_{B_2}\|_2^2}\}$.

Note that Algorithm 4 reduces to Algorithm 3 by choosing $\mathbf{G}_i$'s in Theorem 3. Theorem 3 is a special case of Theorem 4 by setting $\tau = \sigma_{\min}^2(\mathrm{Diag}\{\mathbf{A}_i^\top \mathbf{A}_i + \mathbf{G}_i, i \in B_2\} - \mathbf{A}_{B_2}^\top \mathbf{A}_{B_2})$. So we only give the proof of Theorem 4 in Appendix, available online. It is worth mentioning that, when using backtracking, $\hat{r}_{B_j}^k$ is not a majorant first-order surrogate of $r_{B_j}^k$, since the majorization condition may not hold. Actually, $\hat{r}_{B_j}^k$ only majorizes $r_{B_j}^k$ locally at $\mathbf{x}_{B_j}^{k+1}$. But this is sufficient for the convergence proof, since the formulations of $r_{B_j}^k$ and $\hat{r}_{B_j}^k$ are known and we are able to use their specific properties instead of (20) in the proofs.

5.3 Variable Partition

For (1) with $n > 2$, M-ADMM requires partitioning variables into 2 super blocks B_1 and B_2. Different variable partitions lead to different choices of $\mathbf{L}_i'$ which controls the tightness of the majorant surrogates, and thus the convergence behaviors of M-ADMM are different. Looking for an intelligent way of variable partition may significantly improve the efficiency of M-ADMM. We discuss how to partition variables in three cases by considering the property of $\mathbf{A}_i$'s in (1). The principle is to find a partition such that the constructed surrogate $\hat{r}_{B_1}^k$ for $r_{B_1}^k$ in (36) and $\hat{r}_{B_2}^k$ for $r_{B_2}^k$ in (37) can be as tight as possible.

Case I (Complicated Case). $\mathbf{A}_i^\top \mathbf{A}_l \neq 0$ for any $i \neq l$. This case is complicated since $r_{B_j}^k$, $j = 1, 2$ in (36) and (37) are non-separable for any partition. Then the separable surrogates $\hat{r}_{B_j}^k$'s may be loose when considering the choices of $\mathbf{G}_i$ in (38) and (39). As suggested by Theorem 3, to tighten the bound of (42), a reasonable partition is to make $L_{B_1} + L_{B_2}$, where $L_{B_1} = (n_1 - 1)\sum_{i \in B_1} \|\mathbf{A}_i\|_2^2 - \|\mathbf{A}_{B_1}\|_2^2$ and $L_{B_2} = (n_2 - 1)\sum_{i \in B_2} \|\mathbf{A}_i\|_2^2$, as small as possible.[2] We have a heuristic approach to this end:

Step 1: Sort $\|\mathbf{A}_i\|_2^2$'s in a descending order.

Step 2: Group the largest n_1 elements as the first block and the rest as the second block.

Step 3: The best value of n_1 is the one which minimizes $L_{B_1} + L_{B_2}$ by a one-shot searching from 1 to n.

Case II (Simple Case). There exists a partition such that

$$\mathbf{A}_i^\top \mathbf{A}_l = 0,\ i \neq l, \text{ for any } i, l \in B_1 \text{ and } i, l \in B_2. \quad (46)$$

This case is simple since the above partition makes $r_{B_j}^k$, $j = 1, 2$ in (36) and (37) separable. Then $\hat{r}_{B_j}^k$'s tend to be tight since we can compute each $\hat{r}_i^k$ independently and use $\mathbf{G}_i \succeq 0$, $i \in B_1$ in (38) and $\mathbf{G}_i \succ 0$, $i \in B_2$ in (39). Even, the

2. If n_1 is not very small, $\|\mathbf{A}_{B_1}\|_2^2$ is usually much smaller than $(n_1 - 1)\sum_{i \in B_1} \|\mathbf{A}_i\|_2^2$. We can use $L_{B_1} = (n_1 - 1)\sum_{i \in B_1} \|\mathbf{A}_i\|_2^2$ in this case.

per-iteration cost is cheap when using $\hat{r}_i^k = r_i^k$ for many problems in practice. In this case, (35) can be solved by (36) and (37), which is similar to the standard ADMM. For example, the Low-Rank Representation model in [25] satisfies (46),

$$\min_{\mathbf{Z},\mathbf{J},\mathbf{E}} \|\mathbf{J}\|_* + \lambda\|\mathbf{E}\|_{2,1}, \text{ s.t. } \mathbf{X} = \mathbf{AZ} + \mathbf{E}, \mathbf{Z} = \mathbf{J}, \tag{47}$$

where $\lambda > 0$. The augmented Lagrangian function is

$$\mathcal{L}(\mathbf{Z},\mathbf{J},\mathbf{E},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2) = \|\mathbf{J}\|_* + \lambda\|\mathbf{E}\|_{2,1} + \langle\boldsymbol{\lambda}_1, \mathbf{X} - \mathbf{AZ} - \mathbf{E}\rangle + \langle\boldsymbol{\lambda}_2, \mathbf{Z} - \mathbf{J}\rangle + \frac{\beta}{2}\left(\|\mathbf{X} - \mathbf{AZ} - \mathbf{E}\|^2 + \|\mathbf{Z} - \mathbf{J}\|^2\right).$$

Based on the partition $\{\mathbf{J},\mathbf{E}\}$ and $\{\mathbf{Z}\}$, they can be updated by

$$\begin{cases} \{\mathbf{J}^{k+1},\mathbf{E}^{k+1}\} &= \arg\min_{\mathbf{J},\mathbf{E}} \mathcal{L}(\mathbf{Z}^k,\mathbf{J},\mathbf{E},\boldsymbol{\lambda}_1^k,\boldsymbol{\lambda}_2^k), \\ \mathbf{Z}^{k+1} &= \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{Z},\mathbf{J}^{k+1},\mathbf{E}^{k+1},\boldsymbol{\lambda}_1^k,\boldsymbol{\lambda}_2^k). \end{cases}$$

This is the standard ADMM and its convergence is guaranteed. Note that $\mathcal{L}(\mathbf{Z}^k,\mathbf{J},\mathbf{E},\boldsymbol{\lambda}_1^k,\boldsymbol{\lambda}_2^k)$ is separable w.r.t. $\mathbf{J}$ and $\mathbf{E}$ and thus $\mathbf{J}^{k+1}$ and $\mathbf{E}^{k+1}$ can be computed independently. The updates of the three blocks are similar to the naive multi-block extension of ADMM used in [25], but in different updating orders. Our simple modification fixes the convergence issue of the naive multi-block extension of ADMM in [25] for (47).

In computer vision and signal processing, there are a lot of multi-blocks problems, or their equivalent ones by introducing auxiliary variables, with the property (46) and thus can be solved more efficiently by the Gauss-Seidel ADMMs than Jacobian ADMMs, e.g., sparse subspace clustering model (70) in [11], nonnegative matrix completion problem (143) in [23], multi-task low-rank affinity pursuit model (4) in [5], sparse spectral clustering model (6) in [32], nonnegative low-rank and sparse graph model (5) in [43], simultaneously structured models (3.3) in [37], convex program (8) in [4] for graph clustering, robust multi-view spectral clustering model (3) in [42] and consolidated tensor recovery model (2.6) in [18]. However, some of previous works do not use the property (46) to implement the efficient ADMMs, and this is the reason why we release the toolbox.

Case III (Other Cases). Neither assumptions in Cases I and II holds. It is generally difficult to find the best partition in this case. But one can combine the ideas in both Cases I and II. For example, there exists one or more subgroups B_S, such that $\mathbf{A}_i^\top\mathbf{A}_l = 0$, $i \neq l$, for any $i,l \in B_S$. We can put the whole subgroup in one super block, i.e., $B_S \subset B_1$.

In practice, one usually needs to reformulate the original problem as an equivalent one by introducing auxiliary variables such that the subproblem in ADMMs can be simple. When designing efficient ADMMs, the problem reformulation and the above variable partition strategies should be considered simultaneously. Some more examples can be found in our released toolbox.

6 Proximal Gauss-Seidel ADMM for Multi-Blocks Problems

Consider problem (1) with $n > 2$ blocks, existing Jacobian type ADMMs and our M-ADMM converges only based on the convex objective function assumption. Some recent works [7], [9] propose Gauss-Seidel type ADMMs but their convergence guarantees require much stronger assumption, e.g., strongly convex objective function or the stepsize should be small enough. An interesting open problem is, for (1) with $n > 2$ blocks, does there exist a Gauss-Seidel type ADMM converges without the strongly convex objective assumption? In this section, we propose to solve (1) with $n > 2$ by the Proximal Gauss-Seidel ADMM (Prox-GSADMM), which is a Gauss-Seidel type ADMM. Its convergence requires the objective f or its majorant surrogate $\hat{f}$ to be strongly convex.

In the $(k+1)$th iteration, we first compute the majorant surrogate of f near $\mathbf{x}^k$, i.e., $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\mathbf{x}^k)$, and $\hat{f}^k$ is separable, $\hat{f}^k(\mathbf{x}) = \sum_{i=1}^n \hat{f}_i^k(\mathbf{x}_i)$. Then we update $\mathbf{x}_i$, $i = 1,\ldots,n$, in a Gauss-Seidel fashion, i.e.,

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \hat{f}_i^k(\mathbf{x}_i) + \frac{\beta^{(k)}}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|_{\mathbf{G}_i^k}^2 + \frac{\beta^{(k)}}{2}\left\|\mathbf{A}_i\mathbf{x}_i + \sum_{j=1}^{i-1}\mathbf{A}_j\mathbf{x}_j^{k+1} + \sum_{j=i+1}^{n}\mathbf{A}_j\mathbf{x}_j^k - \mathbf{b} + \frac{\boldsymbol{\lambda}^k}{\beta^{(k)}}\right\|^2, \tag{48}$$

where the choice of $\mathbf{G}_i^k$ is given below. The updates of $\boldsymbol{\lambda}^{k+1}$ and $\beta^{(k+1)}$ are the same as previous frameworks. We summarize the whole procedure of Prox-GSADMM in Algorithm 5. Now, we give the convergence result of Prox-GSADMM. We define $\mathbf{G} = \mathrm{Diag}\{\mathbf{G}_i, i = 1,\ldots,n\}$, $\mathbf{L} = \mathrm{Diag}\{\mathbf{L}_i, i = 1,\ldots,n\}$, $\mathbf{P} = \mathrm{Diag}\{\mathbf{P}_i, i = 1,\ldots,n\}$, $\mathbf{Q} = \mathrm{Diag}\{\mathbf{Q}_i, i = 1,\ldots,n\}$, and

$$\hat{\mathbf{A}} = \begin{bmatrix} 0 & \mathbf{A}_1^\top\mathbf{A}_2 & \cdots & \mathbf{A}_1^\top\mathbf{A}_n \\ & 0 & \ddots & \vdots \\ & & \ddots & \mathbf{A}_{n-1}^\top\mathbf{A}_n \\ & & & 0 \end{bmatrix}.$$

Algorithm 5. Proximal Gauss-Seidel ADMM for (1) with $n > 2$

For $k = 0,1,2,\ldots$ **do**

1. Compute a majorant first-order surrogate $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\mathbf{x}^k)$ with $\hat{f}^k(\mathbf{x}) = \sum_{i=1}^n \hat{f}_i^k(\mathbf{x}_i)$.
2. Update $\mathbf{x}_i$, $i = 1,\ldots,n$, by (48) in a Gauss-Seidel fashion.
3. Update $\boldsymbol{\lambda}$ by $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b})$.
4. Choose $\beta^{(k+1)}$ such that $\beta^{(k)} \leq \beta^{(k+1)} \leq \beta_{\max}$.

end

Theorem 5. *In Algorithm 5, assume that f is $\{\mathbf{Q}_i\}_{i=1}^n$-convex, $\mathbf{Q}_i \succeq 0$, $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f,\mathbf{x}^k)$, $\mathbf{P}_i \succeq \mathbf{L}_i \succeq 0$, and $\mathbf{P}_i + \mathbf{Q}_i \succ 0$. Let $\mathbf{G}^k \succ \frac{1}{\beta^{(k)}}\mathbf{L} + \beta^{(k)}\hat{\mathbf{A}}^\top(\mathbf{P}+\mathbf{Q})^{-1}\hat{\mathbf{A}}$. Then the sequence $\{\mathbf{x}^k\}$ converges to some $\mathbf{x}^L$ that is a solution to problem (1).*

The above theorem shows that when $\mathbf{P}_i + \mathbf{Q}_i \succ 0$, with a proper choice of $\mathbf{G}_i^k$, Prox-GSADMM converges to the optimal solution. The assumption $\mathbf{P}_i + \mathbf{Q}_i \succ 0$ implies that, f and $\hat{f}$, at least one of them should be strongly convex. In the case $\mathbf{Q}_i \succ 0$, i.e., f is strongly convex, our Prox-GSADMM can be regarded as a generalization of the Algorithm 5 in [9]. A more important case is $\mathbf{Q}_i = 0$ and $\mathbf{P}_i \succ 0$, i.e., f is convex and $\hat{f}$ is strongly convex, our Prox-GSADMM and convergence result are completely new. For convex f, Section 2 gives some examples of $\hat{f}$ which are $\{\mathbf{P}_i\}_{i=1}^n$-strongly convex. Among them, the proximal surrogate is the most general choice for constructing $\{\mathbf{P}_i\}_{i=1}^n$-strongly convex $\hat{f}$ with regardless of the structure of f. For example, let $f(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x}_i)$ be convex. We can simply use

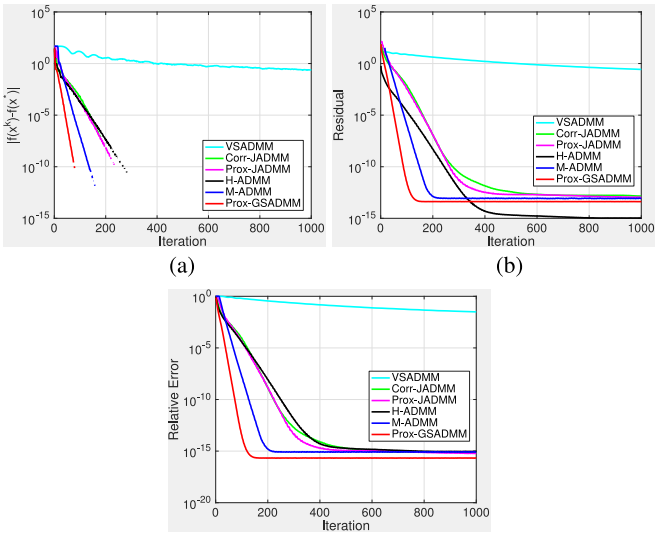


Fig. 1. Plots of (a) $|f(\mathbf{x}^k) - f(\mathbf{x}^*)|$; (b) residual $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|$; (c) relative error $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^*\|$ on the basis pursuit problem.

$$\hat{f}^k(\mathbf{x}) = \sum_{i=1}^{n}\left(f_i(\mathbf{x}_i) + \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{\mathbf{L}_i}^2\right), \tag{49}$$

where $\mathbf{L}_i \succ 0$. Then $\hat{f}^k \in \mathcal{S}_{\{\mathbf{L}_i,\mathbf{P}_i\}_{i=1}^n}(f, \mathbf{x}^k)$, where $\mathbf{P}_i = \mathbf{L}_i$. One may use different $\mathbf{L}_i$ in different iterations. The above choice of $\hat{f}$ guarantees that $\mathbf{P}_i + \mathbf{Q}_i \succ 0$ when f is convex (but non-strongly convex). Thus our Prox-GSADMM is very general and practical.

Note that the convergence result in Theorem 5 and its proof are different from M-ADMM. The convergence rate of Prox-GSADMM is currently not clear, though it is expected to be not slower than M-ADMM since the latest version of $\mathbf{x}$ is always used for the current updating. Also, the choice of $\mathbf{G}^k$ lacks of an intuitive insight from the perspective of majorization minimization and it may not be tight. We leave these for interesting future works.

7 Experiments

In this section, we conduct several experiments to show the effectiveness of our new ADMMs. All the algorithms are implemented by Matlab and are tested on a PC with 8 GB of RAM and Intel Core 2 Quad CPU Q9550. The details of the compared solvers can be found in the Appendix, available online.

7.1 Solving Toy Problems with Multi-Blocks

Besides the unified analysis of several variants of ADMMs, we have two new methods, M-ADMM and Prox-GSADMM for multi-block problems. In this section, we conduct two experiments to demonstrate the effectiveness of our new methods. The first experiment is to verify the improvement of our M-ADMM and Prox-GSADMM over existing methods, e.g., Prox-JADMM [10], on the basis pursuit problem. The second experiment is to verify the effectiveness of our proposed variable partition strategy for M-ADMM in Section 5.3.

7.1.1 ℓ_1-Minimization Problem

We conduct an experiment to show the effectiveness of our M-ADMM and Prox-GSADMM for multi-blocks problems. We test on the same ℓ_1-minimization problem as that in [10] for finding sparse solutions of an underdetermined linear system

$$\min_{\{\mathbf{x}_i\}} f(\mathbf{x}) = \sum_{i=1}^{n}\|\mathbf{x}_i\|_1, \text{ s.t. } \mathbf{b} = \sum_{i=1}^{n}\mathbf{A}_i\mathbf{x}_i, \tag{50}$$

where there has n blocks and the data is partitioned accordingly $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_n]$. This problem is known as the basis pursuit and has many applications in computer vision and signal processing. We generate the data as follows. The sparse representation vector $\mathbf{x}^*$ is randomly generated with $k = 60$ nonzeros drawn from the $N(0,1)$ distribution. We generate $\mathbf{A} \in \mathbb{R}^{d\times m}$, where $d = 300$ and $m = 1{,}000$, with its elements independently sampled from an $N(0,1)$ distribution. Then $\mathbf{A}$ is partitioned evenly into $n = 100$ blocks. The response $\mathbf{y}$ is computed by $\mathbf{y} = \mathbf{A}\mathbf{x}^*$.

M-ADMM solves (50) by the following rules

$$\begin{cases} \mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i}\|\mathbf{x}_i\|_1 + \frac{\beta^{(k)}\eta_i^{(k)}}{2}\left\|\mathbf{x}_i - \mathbf{u}_i^k\right\|^2, \ i \in B_1, \\ \mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i}\|\mathbf{x}_i\|_1 + \frac{\beta^{(k)}\eta_i^{(k)}}{2}\left\|\mathbf{x}_i - \mathbf{v}_i^k\right\|^2, \ i \in B_2, \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}), \end{cases}$$

where $\mathbf{u}_i^k = \mathbf{x}_i^k - \frac{\mathbf{A}_i^T(\boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^k - \mathbf{b}))}{\beta^{(k)}\eta_i^{(k)}}$,

$$\mathbf{v}_i^k = \mathbf{x}_i^k - \frac{\mathbf{A}_i^T(\boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}_{B_1}\mathbf{x}_{B_1}^{k+1} + \mathbf{A}_{B_2}\mathbf{x}_{B_2}^k - \mathbf{b}))}{\beta^{(k)}\eta_i^{(k)}}.$$

In this experiment, we simply choose $B_1 = \{1, 2, \ldots, 50\}$ and $B_2 = \{51, 52, \ldots, 100\}$. We initialize $\mathbf{x}_i$ and $\boldsymbol{\lambda}$ as zeros. We set $\beta^{(0)} = 10/\|\mathbf{b}\|_1$ as suggested in [10] and update $\beta^{(k+1)} = \min(\rho\beta^{(k)}, 10^6)$ with $\rho = 1.1$. The backtracking technique in Algorithm 4 is used to estimate $\eta^{(k)}$ adaptively. We set $\mu = 1.3$, $\eta_i^{(0)} = \rho_\eta n_1\|\mathbf{A}_i\|_2^2$, $i \in B_1$ and $\eta_i^{(0)} = \rho_\eta n_2\|\mathbf{A}_i\|_2^2$, $i \in B_2$, where $\rho_\eta = 5 \times 10^{-3}$. For Prox-GSADMM, we use (49) to construct $\hat{f}$ with $\mathbf{L}_i = 0.3\mathbf{I}$. We set $\mathbf{G}_i^k = \eta_i\mathbf{I} - \mathbf{A}_i^\top\mathbf{A}_i$, where $\eta_i = \rho_\eta\|\mathrm{Diag}(\mathbf{A}) + \frac{\beta^{(0)}}{p_i}\hat{\mathbf{A}}^\top\hat{\mathbf{A}}\|_2$ and $\rho_\eta = 0.01$. The settings of $\beta^{(0)}$ and $\beta^{(k+1)}$ are the same as those in M-ADMM. We also compare our methods with other four ADMM variants: Variable Splitting ADMM (VSADMM) [40], Jacobian ADMM with correction steps (Corr-JADMM) [8], Prox-JADMM [10] and H-ADMM [9]. The detailed updating rules and settings of the first three methods can be found in [10]. For H-ADMM, it has the same updating rule as our M-ADMM on problem (50). The key difference is that our M-ADMM uses the backtracking technique to estimate $\eta_i^{(k)}$ and the stepsize $\beta^{(k)}$ can be increased, while H-ADMM fix both (we use $\tau_i = 0.04\frac{\rho^2}{2}\|\mathbf{A}\|_2^4$ which is slightly better than the choice in [9]). In each iteration, all the compared methods require computing the proximal mapping of the ℓ_1-norm which has a closed form solution. Thus, these methods have the same per-iteration complexity on problem (50).

It is known that, under certain incoherence conditions, the ground truth $\mathbf{x}^*$ can be exactly recovered by solving (50). So we consider the following three measures to evaluate the performance of different solvers: (1) $|f(\mathbf{x}^k) - f(\mathbf{x}^*)|$; (2) residual $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|$; (3) relative error $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^*\|$. We run all the compared methods for 1,000 iterations and record the above three measures. We run on 100 random trials and plot the averages in Fig. 1. Note that in Fig. 1a, many methods seem to stop within 400 iterations. This is because the iterations with $|f(\mathbf{x}^k) - f(\mathbf{x}^*)| = 0$ cannot be plotted in logarithmic scale. It can be seen that VSADMM is much slower than the others, possibly due to many more

introduced blocks of variables. Prox-JADMM, Corr-JADMM and H-ADMM significantly improves VSADMM and they have similar performance.[3] The reason that H-ADMM does not perform better than Prox-JADMM and Corr-JADMM is that H-ADMM uses a fixed hand-tuned parameter while the other two use an adaptive parameter tunning scheme. Our M-ADMM outperforms existing methods based on three evaluation measures. This demonstrates the effectiveness of the mixed updating rules, backtracking technique and the flexible choice of $\beta^{(k)}$ used in M-ADMM. Prox-GSADMM further improves M-ADMM and this shows the advantage of the Gauss-Seidel scheme over the Jacobian scheme.

7.1.2 Analysis of the Proposed Variable Partition Strategy

In this work, from the convergence bound, we observe that ADMMs generally converge faster if the used majorant function is tighter. Considering the problem with $n > 2$, the convergence behaviors of M-ADMM may be quite different when using different variable partitions. Now, we conduct an experiment to compare the convergence behaviors of M-ADMM with different variable partitions and demonstrate the effectiveness of the proposed partition method in Case I in Section 5.3. We consider the following group sparse optimization problem

$$\min_{\{\mathbf{x}_i\}} f(\mathbf{x}) = \sum_{i=1}^{n} \|\mathbf{x}_i\|, \text{ s.t. } \mathbf{b} = \sum_{i=1}^{n} \mathbf{A}_i \mathbf{x}_i, \tag{51}$$

where $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n]$ has n blocks and $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_n]$, in which $\mathbf{A}_i \in \mathbb{R}^{d \times m_i}$ has a compatible dimension with $\mathbf{x}_i \in \mathbb{R}^{m_i}$ and $\sum_{i=1}^{n} m_i = m$. Note that in practice, it is not necessary that the values of $\|\mathbf{A}_i\|_2^2$, $i = 1, \ldots, n$, are similar. They can be quite different. Considering to solve problem (51) by M-ADMM, the convergence speed may be quite different when the variable partition is different in this case. To see this, we generate the synthetic data as follows. We set $n = 100$, $d = 100$, $m_i = 10$ and the elements of $\mathbf{A}_i \in \mathbb{R}^{d \times m_i}$ are independently sampled from a $N(0, i)$ distribution. In this case, $\mathbf{A}_i$'s have the same size, but $\|\mathbf{A}_i\|_2^2$'s are quite different. We plot the sorted $\|\mathbf{A}_i\|_2^2$'s in descending order in Fig. 2a. We generate $\mathbf{x} \in \mathbb{R}^m$ with each element independently sampling from an $N(0, 1)$ distribution. Then we compute $\mathbf{b} = \mathbf{A}\mathbf{x}$.

By choosing $\mathbf{G}_i \succeq \eta_i \mathbf{I} - \mathbf{A}_i^\top \mathbf{A}_i$ in (38) and (39), M-ADMM solves (51) by the following rules

$$\begin{cases} \mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \|\mathbf{x}_i\| + \frac{\beta^{(k)}\eta_i}{2} \left\|\mathbf{x}_i - \mathbf{u}_i^k\right\|^2, \ i \in B_1, \\ \mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \|\mathbf{x}_i\| + \frac{\beta^{(k)}\eta_i}{2} \left\|\mathbf{x}_i - \mathbf{v}_i^k\right\|^2, \ i \in B_2, \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}), \end{cases}$$

where

$$\mathbf{u}_i^k = \mathbf{x}_i^k - \frac{\mathbf{A}_i^T(\boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}\mathbf{x}^k - \mathbf{b}))}{\beta^{(k)}\eta_i},$$

$$\mathbf{v}_i^k = \mathbf{x}_i^k - \frac{\mathbf{A}_i^T(\boldsymbol{\lambda}^k + \beta^{(k)}(\mathbf{A}_{B_1}\mathbf{x}_{B_1}^{k+1} + \mathbf{A}_{B_2}\mathbf{x}_{B_2}^{k} - \mathbf{b}))}{\beta^{(k)}\eta_i}.$$

3. The experiments in [9] show that H-ADMM outperforms Prox-JADMM. The reason is that they do not replicate the results of Prox-JADMM in [10]. See Remark 15 in [9].

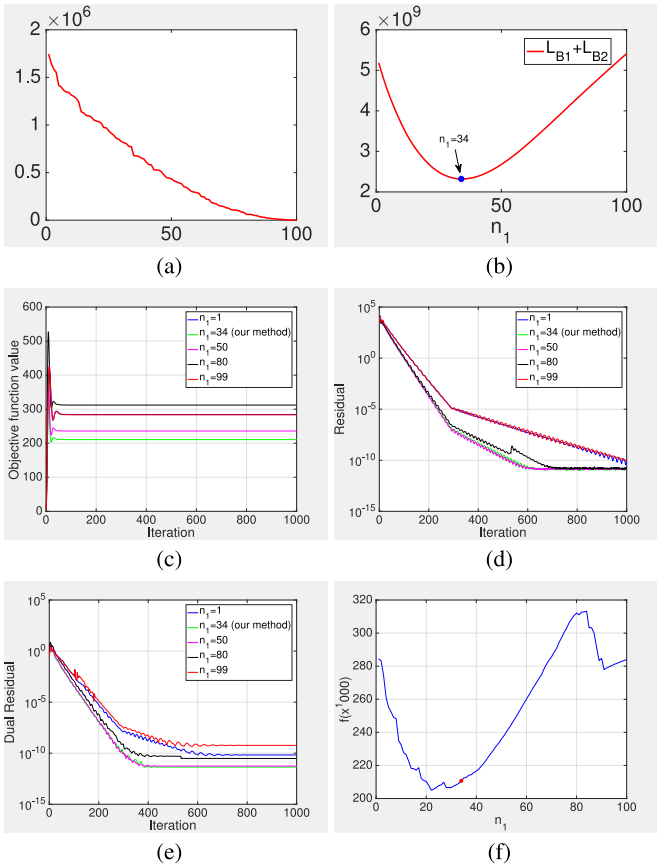


Fig. 2. Plots of (a) sorted $\{\|\mathbf{A}_i\|_2^2, i = 1, \ldots, n\}$ in descending order; (b) $L_{B_1} + L_{B_2}$ versus n_1; (c) $f(\mathbf{x}^k)$ versus k; (d) residual $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|$ versus k based on different variable partitions (corresponding to different n_1); (e) dual residual $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$ versus k, and (f) $f(\mathbf{x}^{1000})$ versus n_1.

In M-ADMM, $\mathbf{x}_i$ and $\boldsymbol{\lambda}$ are initialized as zeros. We set $\beta^{(0)} = 10^{-6}$ and update $\beta^{(k+1)} = \min(\rho\beta^{(k)}, 10^6)$ with $\rho = 1.1$. Let $\eta_i = n_1\|\mathbf{A}_i\|_2^2$ for $i \in B_1$, and $\eta_i = 1.01 n_2 \|\mathbf{A}_i\|_2^2$ for $i \in B_2$. M-ADMM requires dividing these n blocks of variables into two super blocks, i.e., $\mathbf{x}_{B_1}$ with n_1 blocks, and $\mathbf{x}_{B_2}$ with n_2 blocks. Our partition strategy in Case I in Section 5.3 finds n_1 by minimizing $L_{B_1} + L_{B_2}$, where $L_{B_1} = (n_1 - 1)\sum_{i \in B_1} \|\mathbf{A}_i\|_2^2 - \left\|\mathbf{A}_{B_1}\right\|_2^2$ and $L_{B_2} = (n_2 - 1)\sum_{i \in B_2} \|\mathbf{A}_i\|_2^2$. In this experiment, our method gives the best $n_1 = 34$. See the plot of $L_{B_1} + L_{B_2}$ versus n_1 in Fig. 2b. Note that one may have many other choices of $n_1 \in \{1, 2 \ldots, 100\}$. Figs. 2c, 2d, and 2e plot the objective function value $f(\mathbf{x}^k)$ versus iteration k ($\leq$ 1,000), the residual $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|$ versus k and dual residual $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$ versus k, based on different choices of $n_1 \in \{1, 34, 50, 80, 99\}$. Generally, it can be seen that our choice $n_1 = 34$ performs better than other choices of n_1 in the sense that the obtained objective function value is much smaller and the two residuals decrease to 0 faster. If n_1 is relatively small (e.g., $n_1 = 1$) or relatively large $n_1 = 99$, M-ADMM does not perform well since the two super blocks are more unbalanced. Furthermore, we consider all the choices of $n_1 = 1, 2, \ldots, 100$ (each choice of n_1 corresponds to a partition), and run M-ADMM for 1,000 iterations. We record the objective function value at $k = 1{,}000$ for each n_1 and plot $f(\mathbf{x}^{1000})$ versus n_1 in Fig. 2f. It can be seen that the trends of the lines in Figs. 2b and 2f are similar. This verifies our key observation that the M-ADMM converges

faster when the used majorant surrogate is tighter which is implied by a smaller value of $L_{B_1} + L_{B_2}$ (or the two super blocks are more balanced). Also, the obtained function value $f(\mathbf{x}^{1000})$ is quite different for different n_1 (different variable partition), and our choice $n_1 = 34$ is close to the best. This demonstrates the effectiveness of our proposed variable partition strategy.

7.2 Solving Non-Separable Objective Problem

To show that M-ADMM can solve the problem with non-separable objective, we consider the Latent Low-Rank Representation (LatLRR) problem [26] for affine subspace clustering

$$\min_{\mathbf{Z},\mathbf{L}} \|\mathbf{Z}\|_* + \|\mathbf{L}\|_* + \frac{\lambda}{2}\|\mathbf{XZ} + \mathbf{LX} - \mathbf{X}\|^2, \text{ s.t. } \mathbf{1}^\top\mathbf{Z} = \mathbf{1}^\top, \quad (52)$$

where $\lambda > 0$ and the constraint is due to the affine subspace structure of data $\mathbf{X}$ [11]. The objective of (52) is non-separable and can be rewritten as the following one with separable objective

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{L},\mathbf{E}} \|\mathbf{Z}\|_* + \|\mathbf{L}\|_* + \frac{\lambda}{2}\|\mathbf{E}\|^2, \\ &\text{s.t. } \mathbf{1}^\top\mathbf{Z} = \mathbf{1}^\top, \ \mathbf{XZ} + \mathbf{LX} - \mathbf{X} = \mathbf{E}. \end{aligned} \quad (53)$$

We compare the following three solvers which own the convergence guarantee to solve the latent LRR problem:

- L-ADMM-PS (3): use (13) for three blocks problem (53).
- M-ADMM (3): use M-ADMM for three blocks problem (53).
- M-ADMM (2): use M-ADMM for three blocks problem (52).

Note that $h(\mathbf{Z},\mathbf{L}) = \frac{1}{2}\|\mathbf{XZ} + \mathbf{LX} - \mathbf{X}\|^2$ in (52) is $\{2\|\mathbf{X}\|_2^2\mathbf{I}, 2\|\mathbf{X}\|_2^2\mathbf{I}\}$-smooth. M-ADMM (2) uses the Lipschitz gradient surrogate in (23) to make the subproblems separable. For M-ADMM (3), we partition the three variables into two super blocks: $\{\mathbf{Z}\}$ and $\{\mathbf{L},\mathbf{E}\}$, and update them in the Gauss-Seidel way. In contrast, L-ADMM-PS updates $\mathbf{Z}$, $\mathbf{L}$ and $\mathbf{E}$ in parallel.

We apply latent LRR for subspace clustering by using the learned $\mathbf{Z}$ based on both the synthesized and real data. For the synthesized data, we generate $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots]$ with its columns sampled from different subspaces. We construct $k = 5$ independent subspaces $\{\mathcal{S}_i\}_{i=1}^5 \subseteq \mathbb{R}^{200}$ whose bases $\{\mathbf{U}_i\}_{i=1}^5$ are computed by $\mathbf{U}_i = \mathbf{TU}_i$, $1 \le i \le 4$, where $\mathbf{T}$ is a random rotation and $\mathbf{U}_1 \in \mathbb{R}^{200\times 5}$ is a random orthogonal matrix. We sample 100 vectors from each subspace by $\mathbf{X}_i = \mathbf{U}_i\mathbf{Q} + 0.1$, $1 \le i \le 5$ with $\mathbf{Q} \in \mathbb{R}^{5\times 100}$ being an i.i.d. $N(0,1)$ matrix. Furthermore, 20 percent of data vectors are chosen to be corrupted, e.g., for a data vector $\mathbf{x}$ chosen to be corrupted, its observed vector is computed by adding Gaussian noise with zero mean and variance $0.2\|\mathbf{x}\|$. Given $\mathbf{X} \in \mathbb{R}^{200\times 500}$ by the above way, we can solve the latent LRR problem by the three solvers and obtain the solution $\mathbf{Z}^*$. Then the data vectors can be grouped into k groups based on the affinity matrix $(|\mathbf{Z}^*| + |(\mathbf{Z}^*)^\top|)/2$ by spectral clustering [26]. The clustering accuracy is used to evaluate the clustering performance [26]. We test on different choices of λ and compare the three solvers based on $f(\mathbf{x}^k)$ versus CPU time (in seconds), $\|\mathbf{Ax} - \mathbf{b}\|$ versus CPU time and clustering accuracy. The results are shown in Fig. 3 and we have the following observations:

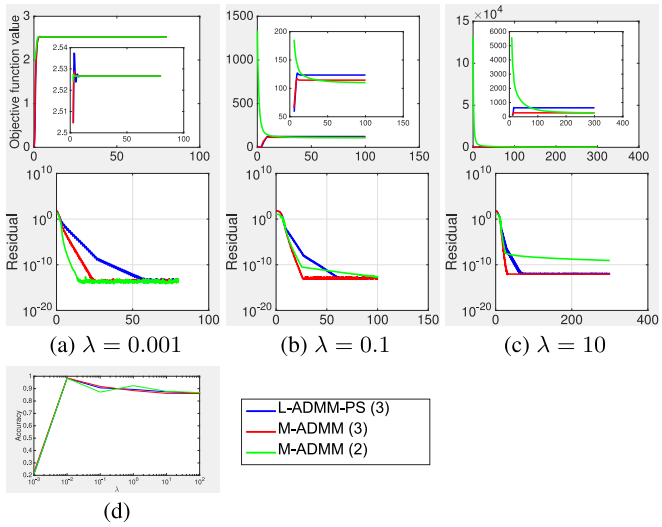


Fig. 3. Comparison of L-ADMM-PS (3), M-ADMM (3) and M-ADMM (2) on different choices of λ: (a) $\lambda = 0.001$; (b) $\lambda = 0.1$ and (c) $\lambda = 10$. Top row: plots of $f(\mathbf{x}^k)$ versus CPU time; middle row: Plots of $\|\mathbf{Ax}^k - \mathbf{b}\|$ versus CPU time. (d) Subspace clustering accuracy versus λ. In (a)-(c), for better visualization, we plot the objective value in a relatively smaller range in sub-figures.

- M-ADMM (3) always outperforms L-ADMM-PS (3) in the sense that the objective value is smaller when the algorithms converge and the residual decreases much faster. Both solve the same problem (53) with three blocks of variables. But M-ADMM (3) updates $\mathbf{Z}$ and $\{\mathbf{L},\mathbf{E}\}$ sequentially, and thus it is faster than L-ADMM-PS (3) which updates them in parallel. This is consistent with our analysis at the end of Section 5.1.
- When λ is relatively small, M-ADMM (2) converges faster than M-ADMM (3). When λ is relatively large, M-ADMM (2) leads to a smaller objective value, but it requires much more running time (many more iterations). Both solvers have their advantages and disadvantages. In this experiment, the block number n and the looseness of the surrogate are two crucial factors. M-ADMM (2) solves (52) with only two blocks, but it requires constructing the Lipschitz gradient surrogate by (23) for $h(\mathbf{Z},\mathbf{L}) = \frac{\lambda}{2}\|\mathbf{XZ} + \mathbf{LX} - \mathbf{X}\|^2$. This surrogate is looser when λ is lager. This is why M-ADMM (2) is slower when λ increases (the same phenomenon also appears in ISTA and FISTA [1]). On the other hand, M-ADMM (3) for 3 blocks problem (53) converges quickly regardless of the choice of λ. The issue of M-ADMM (3) is that the surrogate $\hat{r}_i^k(\mathbf{x}_i)$ in (38) and (39) also becomes looser when $\beta^{(k)}$ increases. So M-ADMM (3) may quickly get stuck and the final objective value is larger than M-ADMM (2). In practice, one has to balance the effects of both the block number n and the looseness of the surrogate, by considering the specific problems.

We further apply latent LRR for motion segmentation and test on the Hopkins 155 dataset [39]. This dataset contains 156 sequences, each with 39∼550 vectors drawn from two or three motions (one motion corresponds to one subspace). Each sequence is a sole segmentation (clustering) task and thus there are 156 clustering tasks in total. We follow the experimental settings in [26] but without the complex post-processing. We set $\lambda = 500$ and compare the

TABLE 3
Comparison of L-ADMM-PS (3) and M-ADMM (3) and M-ADMM (2) for Latent LRR on the Hopkins 155 Dataset

Methods	L-ADMM-PS (3)	M-ADMM (3)	M-ADMM (2)
Accuracy (%)	90.9	**92.7**	87.1
CPU Time (s)	756.2	**738.5**	932.1

Fig. 4. Images used for nonnegative matrix completion.

performance by using M-ADMM (2), L-ADMM-PS (3) and M-ADMM (3). We stop the algorithms when

$$\left\|\mathbf{A}\mathbf{x}^k-\mathbf{b}\right\|/\|\mathbf{b}\|\le\epsilon,\ \text{and}\ \left\|\mathbf{x}^{k+1}-\mathbf{x}^k\right\|/\|\mathbf{b}\|\le\epsilon, \tag{54}$$

where $\epsilon=10^{-4}$. For each motion sequence, we record the clustering accuracy and the CPU time of solvers. Then the mean clustering accuracy and the total CPU time of all 156 sequences are reported in Table 3. It can be seen that, due to the same stopping criteria in (54), the CPU time of L-ADMM-PS (3) and that of M-ADMM (3) are similar. But the solution to latent LRR obtained by M-ADMM (3) achieves better clustering accuracy than L-ADMM-PS (3). The reason is that M-ADMM (3) obtains a better solution with much smaller objective value within similar running time (or similar number of iterations). In this experiment, M-ADMM (2) for (52) is inferior to the other two solvers since the used λ is relatively large and thus the used majorant surrogate is loose.

7.3 Solving Nonnegative Matrix Completion

In this section, we show how to use Gauss-Seidel ADMM to solve a class of problems ($n>2$) with the condition (46) being satisfied. We consider the following nonnegative noisy matrix completion problem [28]

$$\min_{\mathbf{X},\mathbf{E}}\ \|\mathbf{X}\|_*+\frac{\lambda}{2}\|\mathbf{E}\|^2,\ \text{s.t.}\ \mathcal{P}_\Omega(\mathbf{X})+\mathbf{E}=\mathbf{B},\ \mathbf{X}\ge\mathbf{0}, \tag{55}$$

where Ω is an index set and $\mathcal{P}_\Omega$ is a linear mapping that keeps the entries in Ω unchanged and those outside Ω zeros. The above problem can be reformulated as a three blocks problem by (94) in [28] and then solved by L-ADMM-PS. We instead reformulate (55) as

$$\begin{aligned}&\min_{\mathbf{X},\mathbf{E},\mathbf{Z}}\ \|\mathbf{X}\|_*+\frac{\lambda}{2}\|\mathbf{E}\|^2,\\ &\text{s.t.}\ \mathcal{P}_\Omega(\mathbf{Z})+\mathbf{E}=\mathbf{B},\ \mathbf{X}=\mathbf{Z},\ \mathbf{Z}\ge\mathbf{0}.\end{aligned} \tag{56}$$

Note that (46) holds for (56) with the partition $\{\mathbf{X},\mathbf{E}\}$ and $\{\mathbf{Z}\}$. Thus (56) can be solved using (36) and (37) with closed form solutions for each variable. We still refer to this method as M-ADMM in this experiment.

We consider the same image inpainting problem as in [28] which is to fill in the missing pixel values of a corrupted image. As the pixel values are nonnegative, the image inpainting problem can be solved by (55). The corrupted image is generated from the original image by sampling 60 percent of the pixels uniformly at random and adding Gaussian noise with mean zero and standard deviation 0.1. We use the same adaptive penalty to update $\beta^{(k)}$ as [28].

TABLE 4
Numerical Comparison on the Image Inpainting

	L-ADMM-PS			M-ADMM		
images	PSNR	CPU	# Iter.	PSNR	CPU	#Iter.
parrot	28.51	3.50	87	28.54	2.00	55
barbara	27.69	3.36	85	27.72	2.27	60
boat	28.91	3.54	85	28.93	2.21	58
cameraman	26.06	3.33	84	26.08	2.15	58
foreman	31.83	3.80	86	31.84	2.06	54
house	31.26	3.48	87	31.26	2.29	56
lena	27.65	3.55	85	27.68	2.33	62
monarch	25.29	3.47	85	25.33	2.52	63

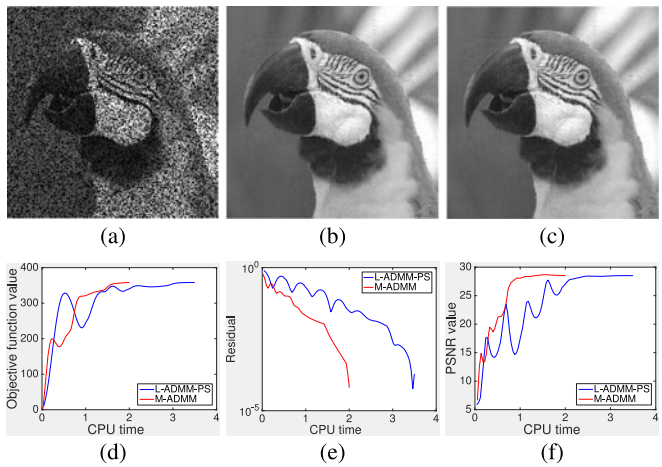


Fig. 5. Top row: The observed noisy image (left), recovered image by L-ADMM-PS (middle), and recovered image by M-ADMM (right). Bottom row: plots of $f(\mathbf{x}^k)$ versus CPU time (left), plots of $\left\|\mathbf{A}\mathbf{x}^k-\mathbf{b}\right\|$ versus CPU time (middle), and PSNR values versus CPU time (right).

We set $\lambda=10$, $\epsilon_1=10^{-3}$, $\epsilon_2=10^{-4}$ and $\beta^{(0)}=\min(d_1,d_2)\epsilon_2$, where $d_1\times d_2$ is the size of $\mathbf{X}$. We update $\beta^{(k+1)}=\max(10\beta^{(k)},10^6)$ when $\max_i(\beta^{(k)}\|\mathbf{x}_i^{k+1}-\mathbf{x}_i^k\|/\|\mathbf{b}\|)\le\epsilon_1$. The stopping criteria are $\max_i(\|\mathbf{x}_i^{k+1}-\mathbf{x}_i^k\|/\|\mathbf{b}\|)\le\epsilon_2$ and $\left\|\mathbf{A}\mathbf{x}^k-\mathbf{b}\right\|/\|\mathbf{b}\|\le\epsilon_1$. We test on eight images, all with size 256×256, in Fig. 4 and evaluate the recovery performance based on the PSNR value. The higher PSNR value indicates better recovery performance. The quantitative results are reported in Table 4 and Fig. 5 gives more results test on the parrot image. It can be seen that, with slightly better recovery performance, M-ADMM converges faster than L-ADMM-PS. The improvement benefits from the sequential updating of $\{\mathbf{X}\}$ and $\{\mathbf{Z},\mathbf{E}\}$ and avoids computing of the majorant surrogate as that in L-ADMM-PS.

8 Conclusion

This paper revisits ADMM, an old but reborn method for convex problems with linear constraint. Many previous ADMMs can be categorized into the Gauss-Seidel ADMMs and Jacobian ADMMs according to different updating orders of the primal variables. We observed that many previous ADMMs update the primal variables by minimizing different majorant functions. Then we proposed the majorant first-order surrogate functions and presented the unified frameworks with unified convergence analysis. They not only draw the connections with existing ADMMs, but also can be used to solve new problems with non-separable

objectives. The convergence bound show that the convergence speed depends on the tightness of the used majorant functions. We then analyzed how to improve the tightness to improve the efficiency. We improve Jacobian ADMMs by introducing the Mixed Gauss-Seidel and Jacobian ADMM and the backtracking technique. We also discussed how to perform variable partition for efficient implementations. Experiments on both synthesized and real-world data well demonstrated the effectiveness of our new ADMMs.

In the future, one may consider extending our unified analysis based on MM to develop new ADMMs or solve other problems, e.g., strongly convex or nonconvex problems, and other ADMMs, e.g., stochastic ADMMs.

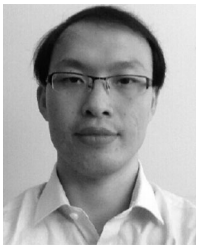
Acknowledgments

J. Feng is partially supported by National University of Singapore startup grant R-263-000-C08-133 and Ministry of Education of Singapore AcRF Tier One grant R-263-000-C21-112. Z. Lin is supported by National Basic Research Program of China (973 Program) (Grant no. 2015CB352502) and National Natural Science Foundation (NSF) of China (Grant nos. 61625301 and 61231002). Z. Lin is the corresponding author for this paper.

References

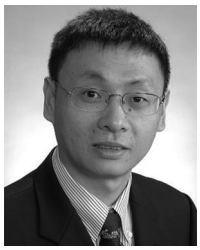
[1] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, pp. 183–202, 2009.
[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[3] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Math. Program.*, vol. 155, nos. 1-2, pp. 57–59, 2013.
[4] Y. Chen, S. Sanghavi, and H. Xu, "Improved graph clustering," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 6440–6455, Oct. 2014.
[5] B. Cheng, G. Liu, J. Wang, Z. Huang, and S. Yan, "Multi-task low-rank affinity pursuit for image segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2439–2446.
[6] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *J. Sci. Comput.*, vol. 66, no. 3, pp. 889–916, 2016.
[7] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *J. Optimization Theory Appl.*, vol. 155, no. 1, pp. 227–238, 2012.
[8] B. He, L. Hou, and X. Yuan, "On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming," *SIAM J. Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015.
[9] D. P. Robinson and R. E. Tappenden, "A flexible ADMM algorithm for big data applications," *J. Sci. Comput.*, pp. 1–33, 2015.
[10] W. Deng, M.-J. Lai, and W. Yin, "Parallel multi-block ADMM with $o(1/k)$ convergence," *J. Sci. Comput.*, pp. 1–25, 2014.
[11] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.
[12] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Comput. Math. Appl.*, vol. 2, no. 1, pp. 17–40, 1976.
[13] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.
[14] B. He and X. Yuan, "On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method," *SIAM J. Numerical Anal.*, vol. 50, no. 2, pp. 700–709, 2012.
[15] B. He and X. Yuan, "Block-wise alternating direction method of multipliers for multipleblock convex programming and beyond, 2015, available: http://www.optimization-online.org/DB_FILE/2014/09/4544.pdf
[16] M. R. Hestenes, "Multiplier and gradient methods," *J. Optimization Theory Appl.*, vol. 4, no. 5, pp. 303–320, 1969.
[17] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Math. Program.*, pp. 1–35, 2012.
[18] B. Huang, C. Mu, D. Goldfarb, and J. Wright, "Provable models for robust low-rank tensor completion," *Pacific J. Optimization*, vol. 11, no. 2, pp. 339–364, 2015.
[19] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1791–1798.
[20] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *J. Comput. Graph. Statist.*, vol. 9, no. 1, pp. 1–20, 2000.
[21] X. Liang, X. Ren, Z. Zhang, and Y. Ma, "Repairing sparse low-rank texture," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 482–495.
[22] T.Y. Lin, S.-Q. Ma, and S.-Z. Zhang, "On the sublinear convergence rate of multi-block ADMM," *J. Oper. Research Soc. China*, vol. 3, no. 3, pp. 251–274, 2015.
[23] Z. Lin, R. Liu, and H. Li, "Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning," *Mach. Learn.*, vol. 99, no. 2, pp. 287–325, 2015.
[24] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *Proc. Advances Neural Inf. Process. Syst.*, 2011, pp. 612–620.
[25] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.
[26] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1615–1622.
[27] J. Liu, S. Ji, and J. Ye, "SLEP: Sparse learning with efficient projections," 2009. [Online]. Available: http://www.public.asu.edu/~jye02/Software/SLEP
[28] R. Liu, Z. Lin, and Z. Su, "Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning," in *Proc. Asian Conf. Mach. Learn.*, 2013, pp. 116–132.
[29] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5249–5257.
[30] C. Lu, J. Feng, Z. Lin, and S. Yan, "Correlation adaptive subspace segmentation by trace Lasso," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1345–1352.
[31] C. Lu, H. Li, Z. Lin, and S. Yan, "Fast proximal linearized alternating direction method of multiplier with parallel splitting," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 739–745.
[32] C. Lu, S. Yan, and Z. Lin, "Convex sparse spectral clustering: Single-view to multi-view," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2833–2843, Jun. 2016.
[33] J. Mairal, "Optimization with first-order surrogate functions," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 783–791.
[34] J. Mairal, F. Bach, J. Ponce, G. Sapiro, R. Jenatton, and G. Obozinski, "SPAMS: Sparse modeling software," 2011. [Online]. Available: http://spams-devel.gforge.inria.fr/index.html
[35] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Berlin, Germany: Springer, 2004.
[36] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic alternating direction method of multipliers," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 80–88.
[37] S. Oymak, A. Jalali, M. Fazel, Y. C. Eldar, and B. Hassibi, "Simultaneously structured models with application to sparse and low-rank matrices," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2886–2908, May 2015.
[38] X. Shen and Y. Wu, "A unified approach to salient object detection via low rank matrix recovery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 853–860.
[39] R. Tron and R. Vidal, "A benchmark for the comparison of 3-D motion segmentation algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[40] X. Wang, M. Hong, S. Ma, and Z.-Q. Luo, "Solving multiple-block separable convex minimization problems using two-block alternating direction method of multipliers," *Pacific J. Optimization*, vol. 11, no. 4, pp. 645–667, 2015.

[41] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[42] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2149–2155.
[43] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, and N. Yu, "Non-negative low rank and sparse graph for semi-supervised learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2328–2335.



Canyi Lu is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, National University of Singapore. His current research interests include computer vision, machine learning, pattern recognition, and optimization. He was the winner of the Microsoft Research Asia Fellowship 2014. He is a student member of the IEEE.

Jiashi Feng received the BE degree from the University of Science and Technology, China, in 2007 and the PhD degree from National University of Singapore, in 2014. He is currently an assistant professor in the Department of Electrical and Computer Engineering, National University of Singapore. He was a postdoc researcher with the University of California from 2014 to 2015. His current research interests focus on machine learning and computer vision techniques for large-scale data analysis. Specifically, he has done work in object recognition, deep learning, machine learning, high-dimensional statistics, and big data analysis.



Shuicheng Yan is currently an associate professor in the Department of Electrical and Computer Engineering, National University of Singapore, and the founding lead of the Learning and Vision Research Group (http://www.lv-nus.org). His research areas include machine learning, computer vision and multimedia, and he has authored/co-authored hundreds of technical papers over a wide range of research topics, with Google Scholar citation $>$30,000 times and H-index 64. He is ISI Highly-cited Researcher, 2014 and IAPR Fellow 2014. He has been serving as an associate editor of the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Circuits and Systems for Video Technology,* and the *ACM Transactions on Intelligent Systems and Technology*. He received the Best Paper Awards from ACM MM'13 (Best Paper and Best Student Paper), ACM MM12 (Best Demo), PCM'11, ACM MM10, ICME10 and ICIMCS'09, the runner-up prize of ILSVRC'13, the winner prize of ILSVRC14 detection task, the winner prizes of the classification task in PASCAL VOC 2010-2012, the winner prize of the segmentation task in PASCAL VOC 2012, the honourable mention prize of the detection task in PASCAL VOC'10, 2010 TCSVT Best Associate Editor (BAE) Award, 2010 Young Faculty Research Award, 2011 Singapore Young Scientist Award, and 2012 NUS Young Researcher Award. He is a fellow of the IEEE.

Zhouchen Lin (M'00-SM'08) received the PhD degree in applied mathematics from Peking University, in 2000. He is currently a professor in the Key Laboratory of Machine Perception, School of Electronics Engineering and Computer Science, Peking University. He is also a chair professor of Northeast Normal University. His research areas include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He is an area chair of CVPR 2014/2016, ICCV 2015, and NIPS 2015, and a senior program committee member of AAAI 2016/2017 and IJCAI 2016. He is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *International Journal of Computer Vision*. He is a fellow of the IAPR and senior member of the IEEE.