A Modified Proximal Symmetric ADMM for Multi-Block Separable Convex Optimization with Linear Constraints

Yuan Shen[†] [1] **Yannian Zuo** [†2] **Xiayang Zhang** [‡3]

School of Applied Mathematics, Nanjing University of Finance & Economics, Nanjing, 210023, P.R.China.[†]

Department of Mathematics and Physics, Nanjing Institute of Technology, Nanjing, 211167, P.R.China. [‡]

Abstract: We consider the linearly constrained separable convex optimization problem whose objective function is separable w.r.t. m blocks of variables. A bunch of methods have been proposed and well studied. Specifically, a modified strictly contractive Peaceman-Rachford splitting method (SC-PRCM) has been well studied in the literature for the special case of $m = 3$. Based on the modified SC-PRCM, we present a modified proximal symmetric ADMM (MPS-ADMM) to solve the multi-block problems. In MPS-ADMM, all subproblems but the first one are attached with a simple proximal term, and the multipliers are updated twice. In addition, at the end of each iteration, the output is corrected via a simple correction step. Without stringent assumptions, we establish the global convergence result for the new algorithms. Preliminary numerical results show that our proposed methods are effective for solving the linearly constrained quadratic programming and the robust principal component analysis problems.

Keywords: separable convex optimization, multiple blocks, symmetric alternating direction method of multipliers, proximal point algorithm

1 Introduction

We consider the linearly constrained sparable convex optimization problem whose objective function is expressed as the sum of m individual functions:

$$\min\{\sum_{i=1}^{m}\theta_i(x_i)|\sum_{i=1}^{m}A_ix_i=b;x_i\in\mathcal{X}_i,i=1,2,...,m\},\tag{1.1}$$

where θ_i: $\mathcal{R}^{n_i}\to\mathcal{R}$ $(i=1,2,...,m)$ are closed proper convex functions (not necessarily smooth); $A_i\in\mathcal{R}^{l\times n_i}$ $(i=1,2,...,m)$; $\mathcal{X}_i\subseteq\mathcal{R}^{n_i}$ $(i=1,2,...,m)$ are nonempty closed convex sets; $b\in\mathcal{R}^l$ and $\sum_{i=1}^{m}n_i=n$. Throughout this paper, the solution set of (1.1) is assumed to be nonempty and A_i $(i=1,2,...,m)$ are assumed to have full column-rank. This model has numerous applications in many fields, such as the latent variable Gaussian graphical model selection [4], the quadratic discriminant analysis model [24] and the robust principal component analysis model with noisy and incomplete data [5, 29], and so on.

In the literature, the operator splitting methods for the special case of (1.1) with $m = 2$ have been well studied, and one of the most popular methods could be the alternating direction method of multipliers (ADMM) proposed originally in Glowinski & Marrocco (1975) [8] and Gabay & Mericier (1976) [7]. More specifically, for solving the special case of (1.1) with $m = 2$:

$$\min\{\theta_1(x_1)+\theta_2(x_2)|A_1x_1+A_2x_2=b,x_1\in\mathcal{X}_1,x_2\in\mathcal{X}_2\},\tag{1.2}$$

the iterative scheme of ADMM is

$$\begin{cases}x_1^{k+1}=\arg\min\{\mathcal{L}_\beta(x_1,x_2^k,\lambda^k)|x_1\in\mathcal{X}_1\},\\ x_2^{k+1}=\arg\min\{\mathcal{L}_\beta(x_1^{k+1},x_2,\lambda^k)|x_2\in\mathcal{X}_2\},\\ \lambda^{k+1}=\lambda^k-\beta(A_1x_1^{k+1}+A_2x_2^{k+1}-b),\end{cases}\tag{1.3}$$

[1] Email: ocsiban@126.com. This research was supported by the National Social Science Foundation of China [grant numbers 19AZD018, 19BGL205, 17BTQ063] and by the Social Science Foundation of Jiangsu province [grant numbers 18GLA002, 17ZTB011].

[2]Email: zynlcg@163.com.

[3](Corresponding author)Email: 369318324@qq.com

where the augmented Lagrangian function $\mathcal{L}_\beta(x_1, x_2, \lambda)$ is defined as follows:

$$\mathcal{L}_\beta(x_1, x_2, \lambda) = \theta_1(x_1) + \theta_2(x_2) - < \lambda, A_1x_1 + A_2x_2 - b > + \frac{\beta}{2}\|A_1x_1 + A_2x_2 - b\|^2$$

with λ being the Lagrange multiplier and $\beta > 0$ being a penalty parameter. In the standard ADMM scheme (1.3), the subproblem is decomposed into two subproblems with smaller size, hence the computation cost at each iteration is lower. In addition, the resulting subproblems of ADMM might be simple enough to have closed-form solutions in some practical applications.

Applying the Peaceman-Rachford splitting method [22, 26] to an equivalent model of (1.2), we obtain the iterative scheme of the symmetric ADMM (S-ADMM):

$$\begin{cases} x_1^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1, x_2^k, \lambda^k) | x_1 \in \mathcal{X}_1\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^k - b), \\ x_2^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2, \lambda^{k+\frac{1}{2}}) | x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases} \tag{1.4}$$

The scheme (1.4) differs from (1.3) only in that the Lagrangian multiplier is updated twice at each iteration. Although the scheme (1.4) indeed works well in some applications [9], however, it is elaborated in [3] that the scheme (1.4) is not necessarily convergent. To deal with this issue, He et al.[11] proposed a modified version of (1.4), yielding the following procedure:

$$\begin{cases} x_1^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1, x_2^k, \lambda^k) | x_1 \in \mathcal{X}_1\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(A_1x_1^{k+1} + A_2x_2^k - b), \\ x_2^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2, \lambda^{k+\frac{1}{2}}) | x_2 \in \mathcal{X}_2\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \tau\beta(A_1x_1^{k+1} + A_2x_2^{k+1} - b). \end{cases} \tag{1.5}$$

Note that the parameter $\tau \in (0, 1)$ in (1.5) serves as a factor enforcing the strict contractiveness of the iterative sequence and thus ensuring the convergence. It has been shown in [11] that this algorithm usually outperforms ADMM for a wide range of applications. Later, for seeking relaxed parameter condition, He et al. [10] proposed a new S-ADMM, which allows the parameter τ in (1.5) to be different on two dual updates. Its parameter domain is larger than that of the scheme (1.5), however, its applications scope is still restricted to the two-block problem.

Based on [7] and [11], Jiang et al. proposed a modified strictly contractive Peaceman-Rachford splitting method (SC-PRSM)[21] for the case of (1.1) with $m = 3$. It inherits the advantages of the scheme (1.5) while its convergence is still guaranteed under mild assumptions. The iterative scheme reads as follows:

$$\begin{cases} x_1^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1, x_2^k, x_3^k, \lambda^k) | x_1 \in \mathcal{X}_1\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(A_1x_1^{k+1} + A_2x_2^k + A_3x_3^k - b) \\ x_2^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2, x_3^k, \lambda^{k+\frac{1}{2}}) | x_2 \in \mathcal{X}_2\}, \\ x_3^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2^k, x_3, \lambda^{k+\frac{1}{2}}) | x_3 \in \mathcal{X}_3\}, \\ \lambda^{k+\frac{3}{4}} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^k + A_3x_3^k - b), \\ v^{k+1} = v^k - \gamma M(v^k - \tilde{v}^k), \end{cases} \tag{1.6}$$

where

$$M = \begin{bmatrix} I & & \\ & I & \\ -\tau\beta A_2 & -\tau\beta A_3 & 2\tau I \end{bmatrix}$$

with $\tau \in (0, 1)$. Actually, in the modified SC-PRCM [21], the Lagrangian multiplier is only updated once at each iteration and $\lambda^{k+\frac{1}{2}}$ can be regarded as an intermediate variable. The iterative sequence is proved to be strictly contractive with respect to the solution set of (1.1) with $m = 3$. Furthermore, the strict contractiveness property ensures its convergence and enables the analysis of its worst-case convergence rate in terms of iteration complexity. However, the modified SC-PRCM can be further improved.

A natural idea for handling the multi-block case $m \geq 3$ is to extend the iterative scheme (1.3) straightforwardly by adopting the Gauss-Seidel [14, 15, 18] or the Jacobian scheme [12, 18, 13], and the resulting extended ADMM empirically works well. However, a counterexample was demonstrated by Chen et al. [2], showing that without further assumptions, we can not derive the convergence of the multi-block ADMM.

One approach to ensure the convergence of the multi-block ADMM is to correct its output by some correction step, while another approach is to modify the subproblems of ADMM. For example, Bai et al. proposed a generalized symmetric ADMM (GS-ADMM)[1] which is a direct extension of S-ADMM for the multi-block case in which a special proximal term is added to each subproblem. As a symmetric ADMM, the multipliers are updated twice with suitable and different stepsizes at each iteration, and its dual step rule is relaxed which enables faster convergence. Numerical experimental results show that it outperforms several efficient ADMM-based algorithms. Shen et al. proposed a patrial PPa S-ADMM algorithm (P3SADMM) [27], which provides a new stepsize domain. Based on the partial PPa block-wise ADMM (P3BADMM)[28], a dual update is added between the two groups of subproblems just like GS-ADMM, so it can be regarded as a combination of GS-ADMM and P3BADMM. More importantly, it does not impose restriction on q, i.e., it allows arbitrary number of subproblems to be intact in the second group of subproblems, which gives more freedom in determining the grouping strategy. He et al. proposed an alternating direction method of multipliers with Gaussian back substitution (ADMM-G)[15] which is a combination of the extension of ADMM with a Gaussian back substitution procedure. In ADMM-G, each iteration consists of a ADMM procedure (forward procedure) and a Gaussian back substitution procedure (backward procedure). Its numerical efficiency is justified by some application problems. Song et al. proposed a twisted version of the proximal ADMM (TADMM) [30] with a simple relaxation step, which combines the correction and the modification strategies with ADMM. In TADMM, a simple proximal term is imposed to all the subproblems but the first one. Numerically, the efficiency of TADMM is verified on two types of problems [30]. Han et al. proposed an augmented Lagrangian based parallel splitting (ALBPS) method in which all the subproblems may be calculated simultaneously [18]. Its correction step is just a simple extension on all the working variables; thus, its extra computational cost is low.

Our purpose is to develop a modified proximal symmetric ADMM (MPS-ADMM) which is a generalization of the modified SC-PRSM. Compared with the rule proposed in [21], all subproblems but the first one are attached with a simple proximal term in MPS-ADMM, thus introducing more variables. Specially, by properly setting the parameters, the performance of MPS-ADMM is expected to be better than its ancestor the modified SC-PRSM. In addition, we propose two different correction formats to correct the output. We show that the proposed algorithms have inspiring numerical behavior compared with the existing ADMM-based algorithms via the numerical experiments on the linearly constrained quadratic programming and the robust principal component analysis problems.

The rest part of this paper is organized as follows. In Section 2, we define some notations and give some preliminaries for the subsequent analysis. In Section 3, we present the new algorithms and establish the convergence results. In Section 4, some numerical experiments are conducted to illustrate the performances of the proposed algorithms. Finally, we give some conclusions in Section 5.

2 Preliminaries

In this section, we summarize some basic definitions and related properties that will be used in later analysis.

2.1 Variational characterization

Let $\mathcal{W} := \mathcal{X}_1 \times \mathcal{X}_2 \times ... \times \mathcal{X}_m \times \mathcal{R}^l$. By deriving its optimality condition, it is easy to see that solving (1.1) is equivalent to finding $w^* := (x_1^*, x_2^*, ..., x_m^*, \lambda^*) \in \mathcal{W}$ such that

$$VI(\mathcal{W}, \mathcal{F}, \theta), \quad \theta(u) - \theta(u^*) + (w - w^*)^T \mathcal{F}(w^*) \geq 0, \quad \forall w \in \mathcal{W}, \tag{2.1}$$

where

$$u := \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}, \quad \theta(u) := \sum_{i=1}^{m} \theta_i(x_i), \quad w := \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ \lambda \end{pmatrix} \quad \text{and} \quad \mathcal{F}(w) := \begin{pmatrix} -A_1^T\lambda \\ -A_2^T\lambda \\ \vdots \\ -A_m^T\lambda \\ \sum_{i=1}^{m} A_i x_i - b \end{pmatrix}. \tag{2.2}$$

Note that u consists of all the primal variables in (1.1) and it is a sub-vector of w. In addition, the mapping $\mathcal{F}(w)$ defined in (2.2) satisfies:

$$(w' - w)^T(\mathcal{F}(w') - \mathcal{F}(w)) = 0 \quad \forall w', w \in \mathcal{R}^{n+l}. \tag{2.3}$$

Under the nonempty assumption on the solution set of (1.1), the solution set of $VI(\mathcal{W}, \mathcal{F}, \theta)$, which is denoted by $\mathcal{W}^*$, is also nonempty and convex [6].

2.2 Some notations

Let the matrices $M_i \in \mathcal{S}^n$ $(i = 2, ..., m)$, then we define some matrices for further analysis:

$$Q := \begin{bmatrix} \beta A_2^T A_2 + \sigma_2 \beta M_2 & & & & -\tau A_2^T \\ & \beta A_3^T A_3 + \sigma_3 \beta M_3 & & & -\tau A_3^T \\ & & \ddots & & \vdots \\ & & & \beta A_m^T A_m + \sigma_m \beta M_m & -\tau A_m^T \\ -A_2 & -A_3 & \cdots & -A_m & \frac{1}{\beta} I \end{bmatrix}, \tag{2.4}$$

$$G_1 := \begin{bmatrix} (1 - \frac{\tau}{2})\beta A_2^T A_2 + \sigma_2 \beta M_2 & \cdots & -\frac{\tau}{2}\beta A_2^T A_m & -\frac{1}{2} A_2^T \\ -\frac{\tau}{2}\beta A_3^T A_2 & \cdots & -\frac{\tau}{2}\beta A_3^T A_m & -\frac{1}{2} A_3^T \\ \vdots & \ddots & \vdots & \vdots \\ -\frac{\tau}{2}\beta A_m^T A_2 & \cdots & (1 - \frac{\tau}{2})\beta A_m^T A_m + \sigma_m \beta M_m & -\frac{1}{2} A_m^T \\ -\frac{1}{2} A_2 & \cdots & -\frac{1}{2} A_m & \frac{1}{2\tau\beta} I \end{bmatrix}, \tag{2.5}$$

$$N_1 := \begin{bmatrix} I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & I & \\ -\tau\beta A_2 & -\tau\beta A_3 & \cdots & -\tau\beta A_m & 2\tau I \end{bmatrix}, \tag{2.6}$$

$$G_2 := \begin{bmatrix} (2 - \tau)\beta A_2^T A_2 + \sigma_2 \beta M_2 & \cdots & (1 - \tau)\beta A_2^T A_m & -A_2^T \\ (1 - \tau)\beta A_3^T A_2 & \cdots & (1 - \tau)\beta A_3^T A_m & -A_3^T \\ \vdots & \ddots & \vdots & \vdots \\ (1 - \tau)\beta A_m^T A_2 & \cdots & (2 - \tau)\beta A_m^T A_m + \sigma_m \beta M_m & -A_m^T \\ -A_2 & \cdots & -A_m & \frac{1}{\beta} I \end{bmatrix}, \tag{2.7}$$

$$N_2 := \begin{bmatrix} I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & I & \\ (1 - \tau)\beta A_2 & (1 - \tau)\beta A_3 & \cdots & (1 - \tau)\beta A_m & I \end{bmatrix}. \tag{2.8}$$

We have two decompositions on the matrices Q and Q^T, respectively, as follows:

$$Q := G_1 N_1 \quad \text{and} \quad Q^T := G_2 N_2 \tag{2.9}$$

with G_1, N_1, G_2 and N_2 defined as (2.5), (2.6), (2.7) and (2.8).

In addition, we introduce the matrices $\overline{M}_i$ $(i = 2, ..., m)$ to define the matrices M_i $(i = 2, ..., m)$ as follows:

$$M_i := A_i^T \overline{M}_i A_i, \tag{2.10}$$

where the matrices $\overline{M}_i \in \mathcal{S}^l$ $(i = 2, ..., m)$ are arbitrary. Using (2.10), we can get M_i with given $\overline{M}_i$ $(i = 2, ...m)$. In addition, with given M_i, we can use the relation $\overline{M}_i = A_i(A_i^T A_i)^{-1} M_i (A_i^T A_i)^{-1} A_i^T$ and the full column-rank of A_i to get $\overline{M}_i$ $(i = 2, ..., m)$. Moreover, we use $\lambda_{\min}(\overline{M}_i)$ $(i = 2, ...,)$ to denote their minimal eigenvalues.

Finally, we summarize some facts regarding the matrices Q, G_1 and G_2 defined in (2.4), (2.5) and (2.7), respectively, in the following lemmas.

Lemma 2.1 *Let $\tau \in (0, \frac{1+\sigma_i \lambda_{\min}(\overline{M}_i)}{m-1})$ and $\sigma_i \geq 0$ $(i = 2, ..., m)$, then the matrix G_1 defined in (2.5) is positive definite.*

Proof Note that the matrix G_1 can be decomposed as:

$$G_1 := D^T \tilde{G}_1 D, \tag{2.11}$$

where

$$D := \begin{bmatrix} \beta^{\frac{1}{2}} A_2 & & & & \\ & \beta^{\frac{1}{2}} A_3 & & & \\ & & \ddots & & \\ & & & \beta^{\frac{1}{2}} A_m & \\ & & & & \beta^{-\frac{1}{2}} I \end{bmatrix} \tag{2.12}$$

and

$$\tilde{G}_1 := \begin{bmatrix} (1-\frac{\tau}{2})I + \sigma_2 \overline{M}_2 & \cdots & -\frac{\tau}{2} I & -\frac{1}{2} I \\ -\frac{\tau}{2} I & \cdots & -\frac{\tau}{2} I & -\frac{1}{2} I \\ \vdots & \ddots & \vdots & \vdots \\ -\frac{\tau}{2} I & \cdots & (1-\frac{\tau}{2})I + \sigma_m \overline{M}_m & -\frac{1}{2} I \\ -\frac{1}{2} I & \cdots & -\frac{1}{2} I & \frac{1}{2\tau} I \end{bmatrix}. \tag{2.13}$$

According to the fact that

$$\begin{aligned} & \begin{bmatrix} I & & & \tau I \\ & \ddots & & \vdots \\ & & I & \tau I \\ & & & I \end{bmatrix} \tilde{G}_1 \begin{bmatrix} I & & & \tau I \\ & \ddots & & \vdots \\ & & I & \tau I \\ & & & I \end{bmatrix}^T \\ = & \begin{bmatrix} (1-\tau)I + \sigma_2 \overline{M}_2 & \cdots & -\tau I & 0 \\ -\tau I & \cdots & -\tau I & 0 \\ \vdots & \ddots & \vdots & \vdots \\ -\tau I & \cdots & (1-\tau)I + \sigma_m \overline{M}_m & 0 \\ 0 & \cdots & 0 & \frac{1}{2\tau} I \end{bmatrix} \\ := & \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}, \end{aligned}$$

where

$$H_1 := \begin{bmatrix} (1-\tau)I + \sigma_2 \overline{M}_2 & \cdots & -\tau I \\ -\tau I & \cdots & -\tau I \\ \vdots & \ddots & \vdots \\ -\tau I & \cdots & (1-\tau)I + \sigma_m \overline{M}_m \end{bmatrix} \tag{2.14}$$

and

$$H_2 := \left[\frac{1}{2\tau} I\right]. \tag{2.15}$$

Obviously, when $\tau > 0$, the submatrix H_2 is positive definite. Then, we only need to show the positive definiteness of H_1.

The submatrix H_1 can be rewritten as:

$$H_1 := \tau \begin{bmatrix} \frac{1-\tau}{\tau} I + \frac{\sigma_2}{\tau}\overline{M}_2 & \cdots & -I \\ -I & \cdots & -I \\ \vdots & \ddots & \vdots \\ -I & \cdots & \frac{1-\tau}{\tau} I + \frac{\sigma_m}{\tau}\overline{M}_m \end{bmatrix}. \tag{2.16}$$

It is easy to verify that the matrix H_1 is positive definite if the following condition holds:

$$\frac{1-\tau}{\tau} + \frac{\sigma_i}{\tau}\lambda_{\min}(\overline{M}_i) > m-2. \tag{2.17}$$

or equivalently $\tau \in (0, \frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1})$ $(i = 2, ..., m)$. The assertion is proved. ■

Lemma 2.2 *Let $\tau \in (0, \frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1})$ and $\sigma_i \geq 0$ $(i = 2, ..., m)$, then the matrix G_2 defined in (2.7) is positive definite .*

The proof of this lemma is similar to that of Lemma 2.1, hence it is omitted.

Lemma 2.3 *Let the matrix Q be defined in (2.4), then the matrix defined as:*

$$\overline{Q} := Q + Q^T \tag{2.18}$$

is positive definite if $\tau \in (0, -1 + 2\sqrt{\frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1}})$ and $\sigma_i \geq 0$ $(i = 2, ..., m)$.

Proof Note that

$$\begin{aligned} \overline{Q} &:= Q + Q^T \\ &= \begin{bmatrix} 2\beta A_2^T A_2 + 2\sigma_2\beta M_2 & & & & -(\tau+1)A_2^T \\ & 2\beta A_3^T A_3 + 2\sigma_3\beta M_3 & & & -(\tau+1)A_3^T \\ & & \ddots & & \vdots \\ & & & 2\beta A_m^T A_m + 2\sigma_m\beta M_m & -(\tau+1)A_m^T \\ -(\tau+1)A_2 & -(\tau+1)A_3 & \cdots & -(\tau+1)A_m & \frac{2}{\beta} I \end{bmatrix} \\ &:= D^T \overline{Q}_0 D \end{aligned}$$

where D is defined in (2.12), $\overline{Q}_0$ is as follows:

$$\overline{Q}_0 := \begin{bmatrix} 2I + 2\sigma_2\overline{M}_2 & & & & -(\tau+1)I \\ & 2I + 2\sigma_3\overline{M}_3 & & & -(\tau+1)I \\ & & \ddots & & \vdots \\ & & & 2I + 2\sigma_m\overline{M}_m & -(\tau+1)I \\ -(\tau+1)I & -(\tau+1)I & \cdots & -(\tau+1)I & 2I \end{bmatrix}. \tag{2.19}$$

Furthermore, we have:

$$\begin{aligned} &\begin{bmatrix} I & & & \frac{\tau+1}{2}I \\ & \ddots & & \vdots \\ & & I & \frac{\tau+1}{2}I \\ & & & I \end{bmatrix} \overline{Q}_0 \begin{bmatrix} I & & & \frac{\tau+1}{2}I \\ & \ddots & & \vdots \\ & & I & \frac{\tau+1}{2}I \\ & & & I \end{bmatrix}^T \\ &:= \begin{bmatrix} \overline{Q}_{11} & 0 \\ 0 & \overline{Q}_{22} \end{bmatrix}, \end{aligned}$$

where

$$\overline{Q}_{11} := \begin{bmatrix} [2 - \frac{(\tau+1)^2}{2}]I + 2\sigma_2\overline{M}_2 & -\frac{(\tau+1)^2}{2}I & \cdots & -\frac{(\tau+1)^2}{2}I \\ -\frac{(\tau+1)^2}{2}I & [2 - \frac{(\tau+1)^2}{2}]I + 2\sigma_3\overline{M}_3 & \cdots & -\frac{(\tau+1)^2}{2}I \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{(\tau+1)^2}{2}I & -\frac{(\tau+1)^2}{2}I & \cdots & [2 - \frac{(\tau+1)^2}{2}]I + 2\sigma_3\overline{M}_3 \end{bmatrix} \tag{2.20}$$

and

$$\overline{Q}_{22} := \left[2I\right]. \tag{2.21}$$

It is easy to see that the matrix $\overline{Q}_{22}$ is positive definite. Following the same line with the proof of Lemma 2.2, we can derive that $\overline{Q}_{11}$ is positive definite when $\tau \in (0, -1 + 2\sqrt{\frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1}})$. The assertion is proved. ■

Remark 2.1 *Recall that the matrix Q defined in (2.4) is asymmetric and the condition $\tau \in (0, -1+2\sqrt{\frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1}})$ ensures the positive definiteness of $Q^T + Q$. Furthermore, we have $G_1N_1 + N_1^TG_1 - \gamma_k N_1^TG_1N_1$ and $G_2N_2 + N_2^TG_2 - \gamma_k N_2^TG_2N_2$ ($Q := G_1N_1$ and $Q^T := G_2N_2$) are positive definite when $\gamma_k = \gamma \cdot \gamma_k^*$ with $\gamma_k^* = \frac{(v^k-\tilde{v}^k)^TQ(v^k-\tilde{v}^k)}{\|N_i(v^k-\tilde{v}^k)\|^2_{G_i}}$ $(i = 1, 2)$ being the optimal stepsize.*

Therefore, we definite $P_1 := G_1N_1 + N_1^TG_1 - \gamma_k N_1^TG_1N_1$ and $P_2 := G_2N_2 + N_2^TG_2 - \gamma_k N_2^TG_2N_2$ for further analysis.

3 New algorithm

In this section, the iterative scheme of the new algorithms are first described. Then we establish the convergence results for the proposed algorithms.

3.1 Algorithm

Our new algorithms are descried as follows:

Algorithm 1: MPS-ADMM for (1.1).
Step 0: Initialization $(x_1^0, ..., x_m^0, \lambda^0)$, k=0. Choose $\beta > 0$, $\gamma \in (0, 2)$ or $\gamma_k = \gamma \cdot \gamma_k^*$, $\sigma_i \geq 0$ and $\tau \in (0, \frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1})$.
Step 1: Find $x_1^{k+1} \in \mathcal{X}_1$ such that:

$$x_1^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1, x_2^k, ..., x_m^k, \lambda^k)|x_1 \in \mathcal{X}_1\}. \tag{3.1}$$

Step 2: Update the Lagrange multiplier with x_1^{k+1}:

$$\lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(A_1x_1^{k+1} + \sum_{i=2}^{m} A_ix_i^k - b). \tag{3.2}$$

Step 3: Compute $\tilde{v}^k := (\tilde{x}_2^k, ..., \tilde{x}_m^k, \tilde{\lambda}^k) \in \mathcal{X}_2 \times ... \times \mathcal{X}_m \times \mathcal{R}^l$:

$$\begin{cases} \tilde{x}_2^k = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2, x_3^k, ..., x_m^k, \lambda^{k+\frac{1}{2}}) + \frac{\sigma_2\beta}{2}\|x_2 - x_2^k\|^2_{M_2}|x_2 \in \mathcal{X}_2\}, \\ \vdots \\ \tilde{x}_m^k = \arg\min\{\mathcal{L}_\beta(x_1^{k+1}, x_2^k, ..., x_{m-1}^k, x_m, \lambda^{k+\frac{1}{2}}) + \frac{\sigma_m\beta}{2}\|x_m - x_m^k\|^2_{M_m}|x_m \in \mathcal{X}_m\}, \\ \tilde{\lambda}^k = \lambda^k - \beta(A_1x_1^{k+1} + \sum_{i=2}^m A_ix_i^k - b). \end{cases} \tag{3.3}$$

Step 4: Set $\tilde{w}^k := (x_1^{k+1}, \tilde{v}^k)$. If a termination criterion is met, stop, otherwise go to Step 5.
Step 5: Compute $v^{k+1} := (x_2^{k+1}, ..., x_m^{k+1}, \lambda^{k+1}) \in \mathcal{X}_2 \times ... \times \mathcal{X}_m \times \mathcal{R}^l$ by

$$v^{k+1} = v^k - \gamma_k N_1(v^k - \tilde{v}^k), \tag{3.4}$$

with N_1 defined as (2.6).
Step 6: Increment k and go to step 1.

Algorithm 2: MPS-ADMM for (1.1).
Step 0-Step 4 are same as Algorithm 1.
Step 5: Compute $v^{k+1} := (x_2^{k+1}, ..., x_m^{k+1}, \lambda^{k+1}) \in \mathcal{X}_2 \times ... \times \mathcal{X}_m \times \mathcal{R}^l$ by

$$v^{k+1} = v^k - \gamma_k N_2(v^k - \tilde{v}^k), \tag{3.5}$$

with N_2 defined as (2.8).
Step 6: Increment k and go to step 1.

Remark 3.2 *In the new algorithms, the matrices M_i $(i = 2, ..., m)$ are symmetric but possibly indefinite. This is different from GS-ADMM [1] with requires proximal matrices to be positive definite.*

Remark 3.3 *Compared with the modified SC-PRSM[21], whose application range is restricted to the three-block problem, our algorithms can adapt to the multi-block ($m > 3$) case. In addition, the proximal terms are added to all the subproblems but the first one and thus more parameters are introduced in our algorithms, which enables our algorithms to have potentially better performance than the modified SC-PRSM. Furthermore, when $m = 3$, it is easy to see that the modified SC-PRSM is a special case of Algorithm 1 with $\sigma = 0$, which indicates that MPS-ADMM is a generalization of the modified SC-PRSM.*

Remark 3.4 *Compared with TADMM [30] and a partially parallel splitting method (refered to as PPSM in this paper) proposed by Hou et al. [20], the matrix Q in [30] and [20] is required to be symmetric positive definite, while this condition is not required in our algorithms. Moreover, a relaxation step is performed to speed up the convergence at the end of each iteration in TADMM, while the output of the subproblems is corrected via an additional correction step in our algorithms. Furthermore, it is easy to see that TADMM and PPSM are both special cases of our methods with $\tau = 1$, M_i being symmetric matrics and $\tau = 1$, $\overline{M}_i = I$ $(i = 2, ..., m)$, $\gamma = 1$, respectively. Finally, MPS-ADMM and TADMM are more general than PPSM.*

Remark 3.5 *The difference between HTY proposed by He et al. [16] and MPS-ADMM is that the Lagrangian multiplier is updated only once and there is no correction step and no relaxation step in HTY. In addition, we can obtain that this method is a special case of our methods with $\tau = 0$.*

3.2 Convergence analysis

In this subsection, we will show that the sequences generated by Algorithm 1 and 2 are strictly contractive with respect to the solution set of (1.1). First, for simplicity, we introduce the following notations:

$$\tilde{w}^k := \begin{pmatrix} \tilde{x}_1^k \\ \tilde{x}_2^k \\ \vdots \\ \tilde{x}_m^k \\ \tilde{\lambda}^k \end{pmatrix}, \quad \tilde{u}^k := \begin{pmatrix} \tilde{x}_1^k \\ \tilde{x}_2^k \\ \vdots \\ \tilde{x}_m^k \end{pmatrix}, \quad \tilde{v}^k := \begin{pmatrix} \tilde{x}_2^k \\ \vdots \\ \tilde{x}_m^k \\ \tilde{\lambda}^k \end{pmatrix}. \tag{3.6}$$

Lemma 3.1 *Let the sequence $\{w^k\}$ be generated by Algorithm 1 and 2, then we have*

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T \mathcal{F}(\tilde{w}^k) \geq (v - \tilde{v}^k)^T Q(v^k - \tilde{v}^k), \quad \forall w \in \mathcal{W}, \tag{3.7}$$

where Q is defined in (2.4).

Proof According to the optimality condition of the x_1-subproblem in (3.1), we have $\tilde{x}_1^k \in \mathcal{X}_1$ such that:

$$\theta_1(x_1) - \theta_1(\tilde{x}_1^k) + (x_1 - \tilde{x}_1^k)^T \{-A_1^T \lambda^k + \beta A_1^T (A_1 \tilde{x}_1^k + \sum_{i=2}^m A_i x_i^k - b)\} \geq 0, \quad x_1 \in \mathcal{X}_1.$$

Using (3.3), the above relation can be simplified into:

$$\theta_1(x_1) - \theta_1(\tilde{x}_1^k) + (x_1 - \tilde{x}_1^k)^T \{-A_1^T \tilde{\lambda}^k\} \geq 0, x_1 \in \mathcal{X}_1. \tag{3.8}$$

In addition, $\lambda^{k+\frac{1}{2}}$ can be rewritten as:

$$\begin{aligned}\lambda^{k+\frac{1}{2}} &= \lambda^k - \tau\beta(A_1\tilde{x}_1^k + \sum_{i=2}^{m} A_i x_i^k - b)\\ &= \lambda^k - \tau(\lambda^k - \tilde{\lambda}^k).\end{aligned}$$

Similarly, for $i = 2, ..., m$, using the above relation, the optimality conditions of (3.3) appear as:

$$\theta_i(x_i) - \theta_i(\tilde{x}_i^k) + (x_i - \tilde{x}_i^k)^T\{-A_i^T\tilde{\lambda}^k + (\beta A_i^T A_i + \sigma_i\beta M_i)(\tilde{x}_i^k - x_i^k) - \tau A_i^T(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \quad x_i \in \mathcal{X}_i. \quad (3.9)$$

Finally, the equation (3.3) can be written as:

$$(\sum_{i=2}^{m} A_i\tilde{x}_i^k - b) - \sum_{i=2}^{m} A_i(\tilde{x}_i^k - x_i^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) = 0,$$

or equivalently

$$(\lambda - \tilde{\lambda}^k)^T\{(\sum_{i=2}^{m} A_i\tilde{x}_i^k - b) - \sum_{i=2}^{m} A_i(\tilde{x}_i^k - x_i^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \quad \lambda \in R^n. \quad (3.10)$$

The assertion is arrived by combining (3.8), (3.9) and (3.10). ■

Recalling that the matrices A_i are all full column ranks, noting $\tau \in (0, \frac{1+\sigma_i\lambda_{\min}(\overline{M}_i)}{m-1})$ and $\sigma_i \geq 0$ $(i = 2, ..., m)$, then the matrices G_i $(i = 1, 2)$ as defined in (2.5) and (2.7), respectively, are positive definite. Invoking the definition of γ_k^* in Algorithm 1 and 2, the lower bound of $\{\gamma_k^*\}$ right after is derived as follows:

$$\gamma_k^* = \frac{(v^k - \tilde{v}^k)^T Q(v^k - \tilde{v}^k)}{\|N_i(v^k - \tilde{v}^k)\|_{G_i}^2} \geq c_0 > 0. \quad (3.11)$$

Theorem 3.1 *Let $\{w^k\}$ be generated by Algorithm 1 and 2, then we have:*

$$\|v^{k+1} - v^*\|_{G_1}^2 \leq \|v^k - v^*\|_{G_1}^2 - \gamma c_0\|v^k - \tilde{v}^k\|_{P_1}^2 \quad (3.12)$$

and

$$\|v^{k+1} - v^*\|_{G_2}^2 \leq \|v^k - v^*\|_{G_2}^2 - \gamma c_0\|v^k - \tilde{v}^k\|_{P_2}^2 \quad (3.13)$$

with $\gamma \in (0, 2)$ and c_0 coming from (3.11).

Proof It following the correction step in Algorithm 1 that:

$$\begin{aligned}\|v^{k+1} - v^*\|_{G_1}^2 &= \|v^k - v^* - \gamma_k N_1(v^k - \tilde{v}^k)\|_{G_1}^2\\ &= \|v^k - v^*\|_{G_1}^2 - 2\gamma_k(v^k - v^*)^T G_1 N_1(v^k - \tilde{v}^k) + \gamma_k^2(v^k - \tilde{v}^k)^T N_1^T G_1 N_1(v^k - \tilde{v}^k)\\ &\leq \|v^k - v^*\|_{G_1}^2 - 2\gamma_k(v^k - \tilde{v}^k)^T G_1 N_1(v^k - \tilde{v}^k) + \gamma_k^2(v^k - \tilde{v}^k)^T N_1^T G_1 N_1(v^k - \tilde{v}^k)\\ &= \|v^k - v^*\|_{G_1}^2 - \gamma_k[(v^k - \tilde{v}^k)^T(G_1 N_1 + N_1^T G_1 - \gamma_k N_1^T G_1 N_1)(v^k - \tilde{v}^k)]\\ &= \|v^k - v^*\|_{G_1}^2 - \gamma_k\|v^k - \tilde{v}^k\|_{P_1}^2\\ &\leq \|v^k - v^*\|_{G_1}^2 - \gamma c_0\|v^k - \tilde{v}^k\|_{P_1}^2.\end{aligned}$$

The assertion (3.12) is proved. In addition, the proof of (3.13) is similar to that of (3.12), which is omitted. ■

Theorem 3.2 *The sequence $\{w^k\}$ generated by Algorithm 1 and 2 converges to a solution of $VI(\mathcal{W}, \mathcal{F}, \theta)$.*

Proof It follows from (3.12) or (3.13) that $\{v^k\}$ is bounded and

$$\lim_{k\to+\infty}\|v^k - \tilde{v}^k\|_{P_1}^2 = 0 \quad \text{or} \quad \lim_{k\to+\infty}\|v^k - \tilde{v}^k\|_{P_2}^2 = 0. \quad (3.14)$$

Therefore, $\{\tilde{v}^k\}$ is also bounded. Let v^∞ be a cluster point of $\{\tilde{v}^k\}$ and $\{\tilde{v}^{k_j}\}$ be a subsequence which converges to v^∞. It follows from (3.14) that $\lim_{j\to+\infty} v^{k_j} = v^\infty$. Since

$$\tilde{\lambda}^k = \lambda^k - \beta(A_1\tilde{x}_1^k + \sum_{i=2}^m A_i x_i^k - b), \tag{3.15}$$

we have

$$A_1\tilde{x}_1^{k_j} = \frac{1}{\beta}(\lambda^{k_j} - \tilde{\lambda}^{k_j}) - (\sum_{i=2}^m A_i x_i^{k_j} - b), \tag{3.16}$$

which implies that $\{A_1\tilde{x}_1^{k_j}\}$ is convergent because the limit of $\{\sum_{i=2}^m A_i x_i^{k_j} - b\}$ exists and $\lim_{j\to+\infty}(\lambda^{k_j} - \tilde{\lambda}^{k_j}) = 0$. Let $\mathcal{N}$ be its limit. We can get $A_1^TA_1\tilde{x}_1^{k_j} \to A_1^T\mathcal{N}$ $(j\to\infty)$ and $\tilde{x}_1^{k_j} \to (A_1^TA_1)^{-1}A_1^T\mathcal{N}$. Thus, we have $\lim_{j\to+\infty}\tilde{x}_1^{k_j} = x_1^\infty := (A_1^TA_1)^{-1}A_1^T\mathcal{N}$. It follows from (3.7) and (3.15) that

$$\begin{cases}
\theta_1(x_1^{k_j}) - \theta_1(\tilde{x}_1^{k_j}) + (x_1^{k_j} - \tilde{x}_1^{k_j})^T\{-A_1^T\tilde{\lambda}^{k_j}\} \geq 0,\\
\theta_2(x_2^{k_j}) - \theta_2(\tilde{x}_2^{k_j}) + (x_2^{k_j} - \tilde{x}_2^{k_j})^T\{-A_2^T\tilde{\lambda}^{k_j} + (\beta A_2^TA_2 + \sigma_2\beta M_2)(\tilde{x}_2^{k_j} - x_2^{k_j}) - \tau A_2^T(\tilde{\lambda}^{k_j} - \lambda^{k_j})\} \geq 0,\\
\vdots\\
\theta_m(x_m^{k_j}) - \theta_m(\tilde{x}_m^{k_j}) + (x_m^{k_j} - \tilde{x}_m^{k_j})^T\{-A_m^T\tilde{\lambda}^{k_j} + (\beta A_m^TA_m + \sigma_m\beta M_m)(\tilde{x}_m^{k_j} - x_m^{k_j}) - \tau A_m^T(\tilde{\lambda}^{k_j} - \lambda^{k_j})\} \geq 0,\\
(A_1\tilde{x}_1^{k_j} + \sum_{i=2}^m A_i x_i^{k_j} - b) + \frac{1}{\beta}(\tilde{\lambda}^{k_j} - \lambda^{k_j}) = 0.
\end{cases} \tag{3.17}$$

Taking the limit of (3.17) and using (3.14), we obtain

$$\begin{cases}
\theta_1(x_1^\infty) - \theta_1(\tilde{x}_1^\infty) + (x_1^\infty - \tilde{x}_1^\infty)^T\{-A_1^T\tilde{\lambda}^\infty\} \geq 0,\\
\theta_2(x_2^\infty) - \theta_2(\tilde{x}_2^\infty) + (x_2^\infty - \tilde{x}_2^\infty)^T\{-A_2^T\tilde{\lambda}^\infty\} \geq 0,\\
\vdots\\
\theta_m(x_m^\infty) - \theta_m(\tilde{x}_m^\infty) + (x_m^\infty - \tilde{x}_m^\infty)^T\{-A_m^T\tilde{\lambda}^\infty\} \geq 0,\\
A_1x_1^\infty + \sum_{i=2}^m A_i x_i^\infty - b = 0.
\end{cases} \tag{3.18}$$

Thus $(u^\infty, \lambda^\infty)$ is a solution point of $VI(\mathcal{W}, \mathcal{F}, \theta)$. Invoking (3.12) and (3.13) together with $\lim_{j\to+\infty} v^{k_j} = v^\infty$, we obtain the original sequence $\{v^k\}$ also converges to v^∞. Noting that $\tilde{x}_1^{k_j} \to x_1^\infty$ $(j\to+\infty)$ and the limit of $A_1\tilde{x}_1^k = \frac{1}{\beta}(\lambda^k - \tilde{\lambda}^k) - (\sum_{i=2}^m A_i x_i^k - b)$ exists, and using $\tilde{x}_1^{k_j} \to (A_1^TA_1)^{-1}A_1^T\mathcal{N}$, we can obtain that the limit of $\tilde{x}_1^k$ exists. That implies $\lim_{k\to+\infty} x_1^k = x_1^\infty$. In addition, the limit of $\{\tilde{x}_1^k\}$ exists because $\tilde{x}_1^k = (A_1^TA_1)^{-1}A_1^T[\frac{1}{\beta}(\lambda^k - \tilde{\lambda}^k) - (\sum_{i=2}^m A_i x_i^k - b)]$ and the limit of $\{\lambda^k - \tilde{\lambda}^k\}$ and $\{\sum_{i=2}^m A_i x_i^k - b\}$ exist. Based on the analysis, we have $\tilde{x}_1^{k_j} \to x_1^\infty$. Therefore, $\tilde{x}_1^k \to x_1^\infty$. Invoking $\tilde{x}_1^k = x_1^{k+1}$, we obtain $x_1^k \to x_1^\infty$. Owing to the above analysis, we have $w^k \to w^\infty = \begin{pmatrix} x_1^\infty \\ v^\infty \end{pmatrix}$. The proof is completed. ■

4 Numerical Experiments

In this section, we illustrate the efficiency of the proposed algorithms by applying it to solve the the linearly constrained quadratic programming (LCQP) problem and the robust principal component analysis (RPCA) problem. We code the proposed algorithms by MATLAB R2015a. All the experiments were implemented on a desktop computer with Intel Core i7 CPU at 3.6GHz with 8 GB memory.

4.1 The LCQP model

We first consider the following linearly constrained quadratic programming (LCQP):

$$\begin{aligned}
&\min_x f_1(x_1) + ... + f_p(x_p)\\
&\text{s.t.} \quad A_1x_1 + ... + A_px_p = c,
\end{aligned} \tag{4.1}$$

where $f_i(x_i) = \frac{1}{2}x_i^TH_ix_i + x_i^Tq_i$ $(i = 1, ..., p) \in \mathcal{R}^{m_i}$ with $H_i \in \mathcal{R}^{m_i\times m_i}$, $q \in \mathcal{R}^{m_i}$, $A_i \in \mathcal{R}^{n\times m_i}$ and $c \in \mathcal{R}^n$.

For ease of notation, we denote:

$$f(x) = \sum_{i=1}^{p} f_i(x_i), \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}, \quad A = \begin{pmatrix} A_1 & \cdots & A_p \end{pmatrix}. \tag{4.2}$$

Note that all the splitting subproblems for solving (4.1) are unconstrained quadratic programming, which is equivalent to linear equations. Since the Hessian (coefficient matrix) in all the subproblems are fixed, it is possible to speed up the algorithm by doing Cholesky decomposition on these Hessian at the beginning of the algorithm. Then, the computation of each subproblem can be decomposed into two simpler subproblems whose coefficient matrices are lower and upper triangle matrices, respectively.

To show the efficiency of the proposed algorithms, the numerical experiments of solving the problem (4.1) are carried out by comparing the new algorithms with two efficient ADMM-based algorithms: the Block-wise ADMM with Relaxation factor (BADMMR) [17] , the Generalized Symmetric ADMM (GS-ADMM)[1].

First, the framework of our algorithms determine that the grouping strategy can only be $1 \sim (p-1)$ for our algorithms. However, it is necessary to regroup the variables of (4.1) into two groups to implement BADMMR and GSADMM. Indeed, the scheme of these two algorithms is a mixture of Jacobi and Gauss-Seidel, and there is the freedom to select the grouping strategy. For example, when $p = 3$, it is possible to regroup the 3 blocks of variables by either $1 \sim 2$ or $2 \sim 1$ strategy, which can result in two iterative schemes for these two algorithms, respectively. In addition, for Algorithm 2, the optimal step size γ_k^* needs to be calculated, and the test about Algorithm 2 should be performed under the optimal step size γ_k^*. Based on the analysis, (m_i, n_i) can only be $m_i = n_i$ in Algorithm 2. Furthermore, all the working variables are initialized with zero vectors for all the test algorithms and the algorithms are stopped when $\mathrm{KKT}(k) < \mathrm{Tol}$ which is defined as follows:

$$KKT(k) := \max(KKT_1(k), ..., KKT_p(k), KKT_\lambda(k)), \tag{4.3}$$

where

$$KKT_i = \|H_i x_i^k + q_i - A_i^T \lambda^k\|, \quad i = 1, ..., p, \quad KKT_\lambda(k) = \|A_1 x_1^k + ... + A_p x_p^k - c\|. \tag{4.4}$$

Unless otherwise specified, the default setting of maxit and Tol are 1000 and 10^{-14}, respectively.

The setting of parameter β in all the test algorithms is critical to their performances, but its optimal setting can be different with different problem settings, hence its setting is empirically chosen to maximize their performances in our experiments. We set $\overline{M}_i = I$ in our algorithms. In Algorithm 1 (MPSADMM), for the 3 blocks of variables, we set $\tau = 0.3$, $\gamma = 1.9$, $\sigma_2 = 1.01$ and $\sigma_3 = 1.01$ when $(m_i, n_i) = (100, 40)$, $(100, 100)$, and $\beta = 2.1, 0.2$, respectively; for the 4 blocks of variables, we set $\beta = 0.6$, $\tau = 0.4$, $\gamma = 1.25$, $\sigma_2 = 1.01$, $\sigma_3 = 1.01$ and $\sigma_4 = 1.01$ when $(m_i, n_i) = (100, 30)$. In Algorithm 2 (MPSADMM2), for the 3 blocks of variables, we set $\tau = 0.5$, $\beta = 0.6$, $\gamma_k^* = 0.8\frac{(v^k - \tilde{v}^k)^T Q(v^k - \tilde{v}^k)}{\|N_2(v^k - \tilde{v}^k)\|_{G_2}^2}$, $\sigma_2 = 1.01$ and $\sigma_3 = 1.01$ when $(m_i, n_i) = (100, 100)$; for the 4 blocks of variables, we set $\beta = 0.12$, $\tau = 0.4$, $\gamma_k^* = 0.6\frac{(v^k - \tilde{v}^k)^T Q(v^k - \tilde{v}^k)}{\|N_2(v^k - \tilde{v}^k)\|_{G_2}^2}$, $\sigma_2 = 1.01$, $\sigma_3 = 1.01$ and $\sigma_4 = 1.01$ when $(m_i, n_i) = (100, 100)$. For BADMMR, we set the λ update to be 1.6 (close to $\frac{\sqrt{5}+1}{2}$) as suggested in [17]. In the iterating scheme of GSADMM which updates multipliers twice with two extension factors, we set them to be 0.9 and 1.09 as suggested in [1].

First, we would like to investigate the speed performance of MPSADMM and MPSADMM2. The average results obtained with different problem settings are plotted when $(m_i, n_i) = (100, 100)$. We observe from Figure 1 that: for 3 blocks of variables, MPSADMM2 outperforms the other three algorithms and MPSADMM performs better than GSADMM and BADMMR; for 4 blocks of variables, MPSADMM2 converges very slowly while MPSADMM performs better than MPSADMM2, GSADMM and BADMMR, when the grouping strategy is $2 \sim 2$; for 4 blocks of variables, MPSADMM and MPSADMM2 outperform the other two algorithms, especially the performance gap of MPSADMM is obvious when the grouping strategies are $1 \sim 3$ and $2 \sim 1$. In addition, for MPSADMM, there are more choices for (m_i, n_i), and the performance stability of MPSADMM are better than MPSADMM. Therefore, we only compare MPSADMM with GSADMM and BADMMR in the next experiments.

Next, under the case of 3 block variables, we fix other parameters but (m_i, n_i), and the iteration process of KKT violations with several different problem settings are plotted in Figure 2. We observe from Figure 2 that the convergence speed of MPSADMM always outperforms the other algorithms, especially when the value of

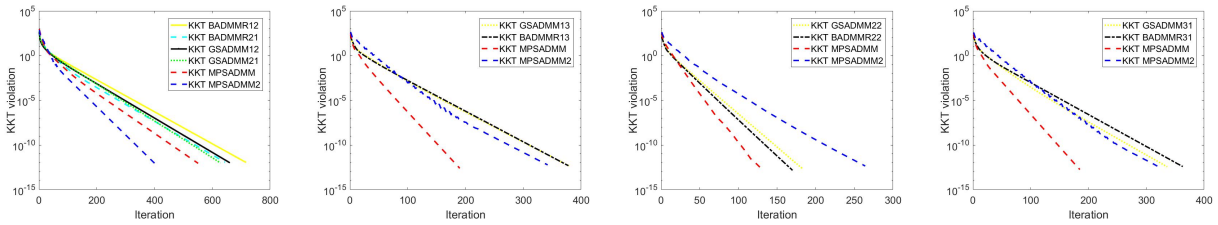

Figure 1: The first case is 3-block, and the rest is 4-block.

m/n is larger, the performance gap between MPSADMM and other algorithms is larger which implies that we can get faster convergence speed than other two algorithms.

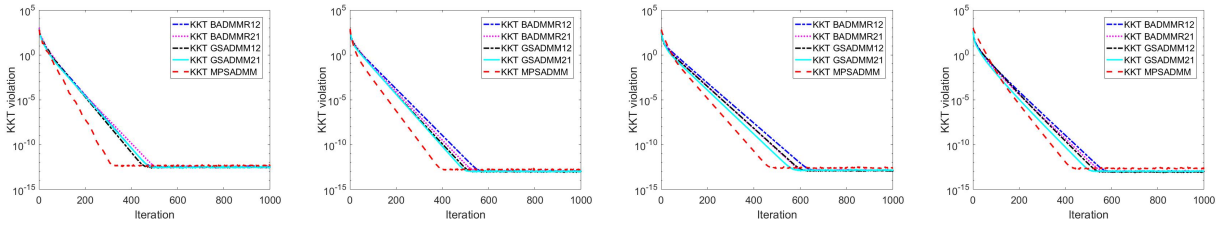

Figure 2: From left to right are the results with (m, n_i)=(100,40), (100,60), (100,80) and (100,100), respectively.

Third, in order to investigate the speed performance of MPSADMM with more blocks of variables, we report the numerical results with 4 and 5 blocks of variables with $(m, n) = (100, 30)$ and $(m, n) = (100, 100)$ in Figure 3 and 4, respectively. It is observed that the convergence of MPSADMM is always faster than that of the other algorithms from Figure 3 and 4.

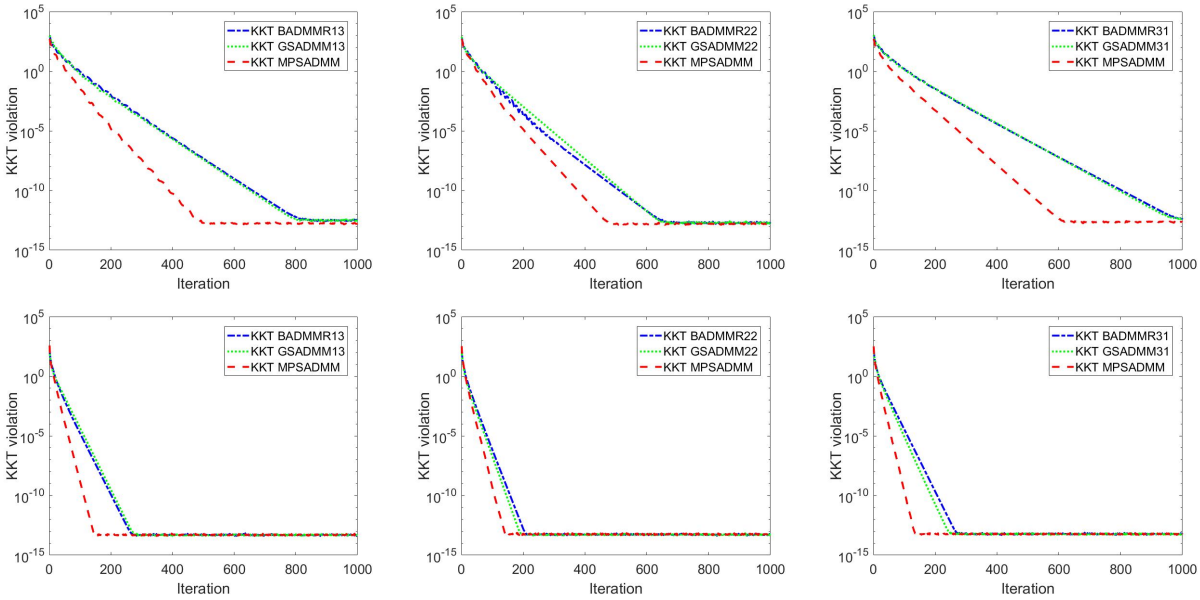

Figure 3: For 4 block variables, from above to below are the results with (m, n_i)=(100,30) and (100,100), respectively.

Finally, since the Hessian in all the subproblems can effect the convergence speed of the test algorithms, we tested this experiment with different condition number of H ($cond(H)$) under 3 blocks of variables. The results are plotted in Figure 5. For each case, we set $(m, n) = (100, 40)$, $\tau = 0.3$, $\beta = 2.1$ and $\gamma = 1.9$ while we let $cond(H)$ vary between 10^2 and 10^{10}. In Figure 5, we observe that MPSADMM is always better than

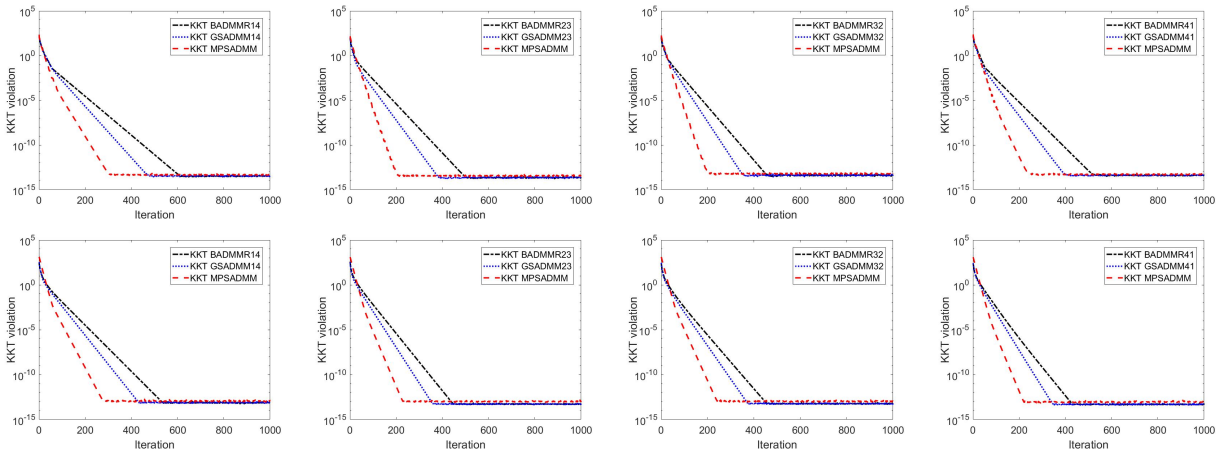

Figure 4: For 5 block variables, from above to below are the results with (m, n_i)=(100,30) and (100,100), respectively.

the other two algorithms, especially under ill-conditioned. For example, when $cond(H) = 10^{10}$, MPS-ADMM is about 149% faster than GSADMM12.

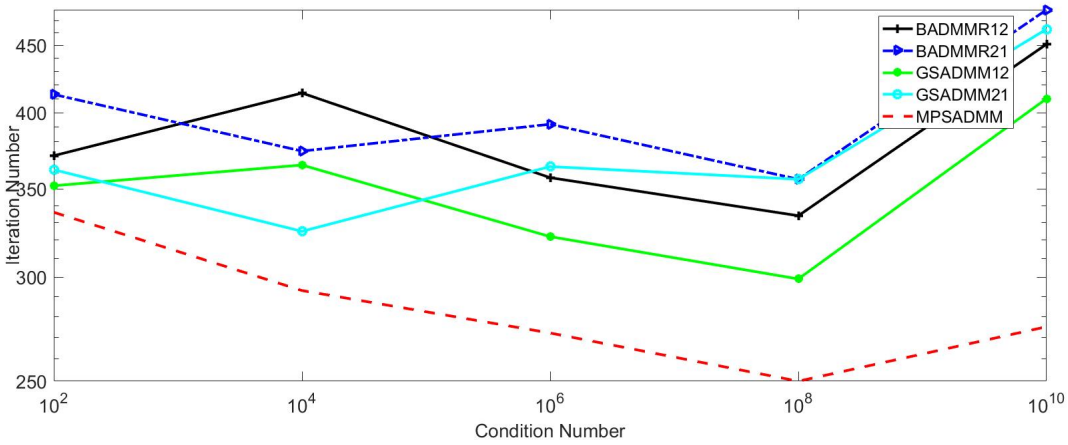

Figure 5: When $(m, n) = (100, 40)$, the iteration number with different condition number settings.

In summary, we may conclude that, when the value of m/n is larger and the Hessian is in ill-conditioned, the convergence speed of our algorithms are faster than that of some most efficient ADMM-based algorithms, which means that our algorithms are overall competitive algorithm for solving the LCQP problem.

4.2 The RPCA model

In [5], the following RPCA model is considered:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \mu\|E\|_1 \\ \text{s.t.} \quad & A + E = C, \end{aligned} \tag{4.5}$$

where $C \in \mathcal{R}^{l\times n}$ is a given matrix; $\|\cdot\|_*$ is the nuclear norm which is defined as the sum of all singular values, and it is to induce the low-rank feature in the component A; $\|\cdot\|_1$ denotes the sum of the absolute values of all entries, and it is to induce sparsity in the component E; and $\mu > 0$ is a constant balancing the low-rank and sparsity. It has been showed that, under certain mild assumptions, model (4.5) can recover the original solution accurately [5].

The observation C could be corrupted by additive Gaussian noise in many real applications; thus, the equality constraint in (4.5) may not be reasonable. To this end, the following relaxation model was suggested [29]:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \mu\|E\|_1 \\ \text{s.t.} \quad & A + E + Z = C, \quad \|Z\|_F \le \delta, \end{aligned} \tag{4.6}$$

where $\|\cdot\|_F$ represents the Frobenius norm and δ is a parameter related to the noise level.

We observe that model (4.6) is a particular case of the general multi-block model (1.1) with 3 blocks of variables, therefore, our algorithms are suitable for solving it. Moreover, in our experiments, we do the singular value decomposition (SVD) by the package of PROPACK in [23] to compute those singular values that are larger than a particular threshold and their corresponding singular vectors in A-subproblems (see details in [29]). The relative error of A and E ($err(A)$:=$\frac{\|A^k-A^*\|_F}{\|A^*\|_F}$ and $err(E)$:=$\frac{\|E^k-E^*\|_F}{\|E^*\|_F}$, respectively) are used to measure the solution accuracy.

The proposed algorithms (denoted by MPSADMM and MPSADMM2, respectively) are compared with some efficient algorithms to reveal the speed performance: Generalized Symmetric ADMM [1] in $1 \sim 2$ or $2 \sim 1$ fashion (denoted by GSADMM12 and GSADMM21, respectively); the splitting method [16] proposed by He, Tao and Yuan (denoted by ADMMHTY); the partial splitting augmented Lagrangian method [19] proposed by Han and et. al (denoted by PSAL).

Some different choices of rr (the ratios of sample entries of E^*) and k (rank of A^*) are given when (m, n)=(100,100) in Figure 6 and 7. From Figure 6 and 7, as a whole, we can see that the results remain almost unchanged while we let rr and k vary, and MPSADMM are always better than MPSADMM2. In detail, for example, in Figure 6, when $rr = 0.3$, we obverse that the convergence speed of MPSADMM is the best, but MPSADMM has slightly less accuracy than PSAL in $err(A)$; the convergence speed of MPSADMM is close to PSAL at the beginning stage of iteration progress, and the solution accuracy of MPSADMM is not bad in $err(E)$. In Figure 7, the details are the same as the analysis in Figure 6, when $rr = 0.3$. In fact, on the one hand, the ending stage of iteration progress does not improve the quality of the solution, so the convergence speed at the beginning stage is more important. On the other hand, we usually focus on the matrix A, so it makes more sense to study the convergence speed in terms of the performance of matrix A. Overall, compared with the other tested algorithms, the performance of MPSADMM is competitive.

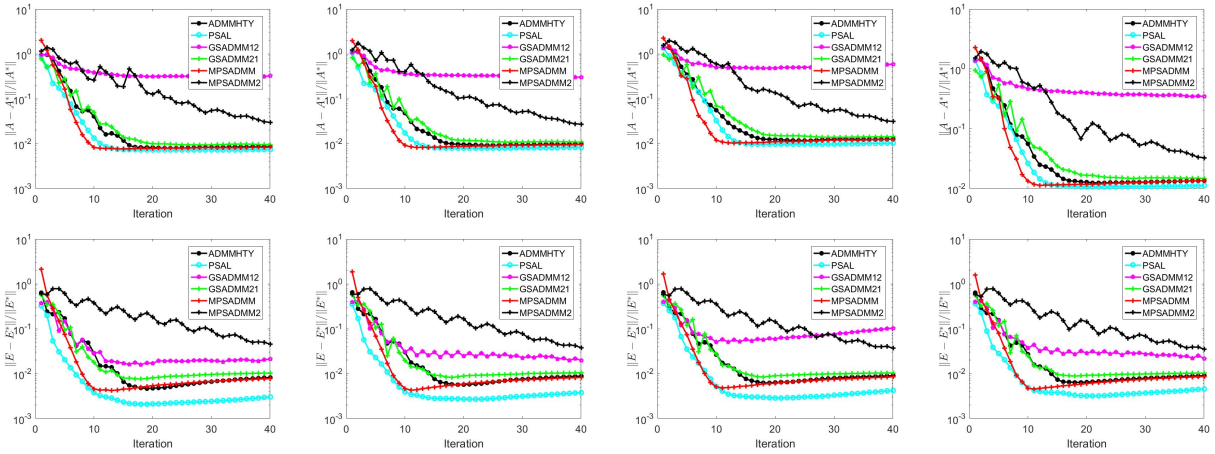

Figure 6: The above to below are the results with the ratios of sample entries of $err(A)$ and $err(E)$, respectively. From left to right are the results with rr=0.03, 0.04, 0.05, 0.06, respectively.

Next, we fix $rr = 0.05$, $k = 5$ and m but n, and the iteration progress of relative error with several different problem settings are plotted in Figure 8. From Figure 8, we observe that, when the value of m/n is larger, the performance gap between MPSADMM and other tested algorithms is larger which implies that MPSADMM

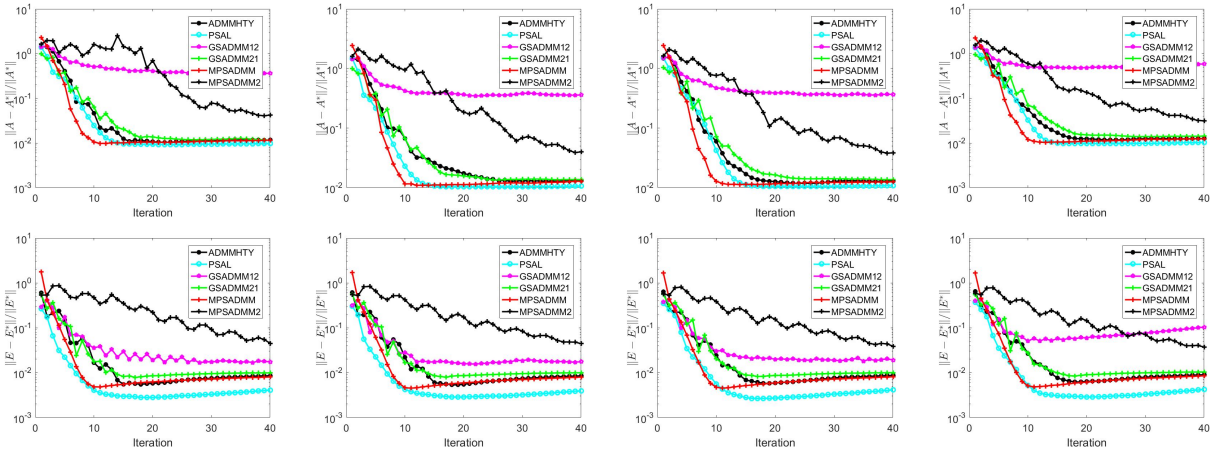

Figure 7: The above to below are the results with the ratios of sample entries of $err(A)$ and $err(E)$, respectively. From left to right are the results with k=2, 3, 4, 5, respectively.

is always the best one among all the test algorithms from the aspects of both convergence speed and solution accuracy.

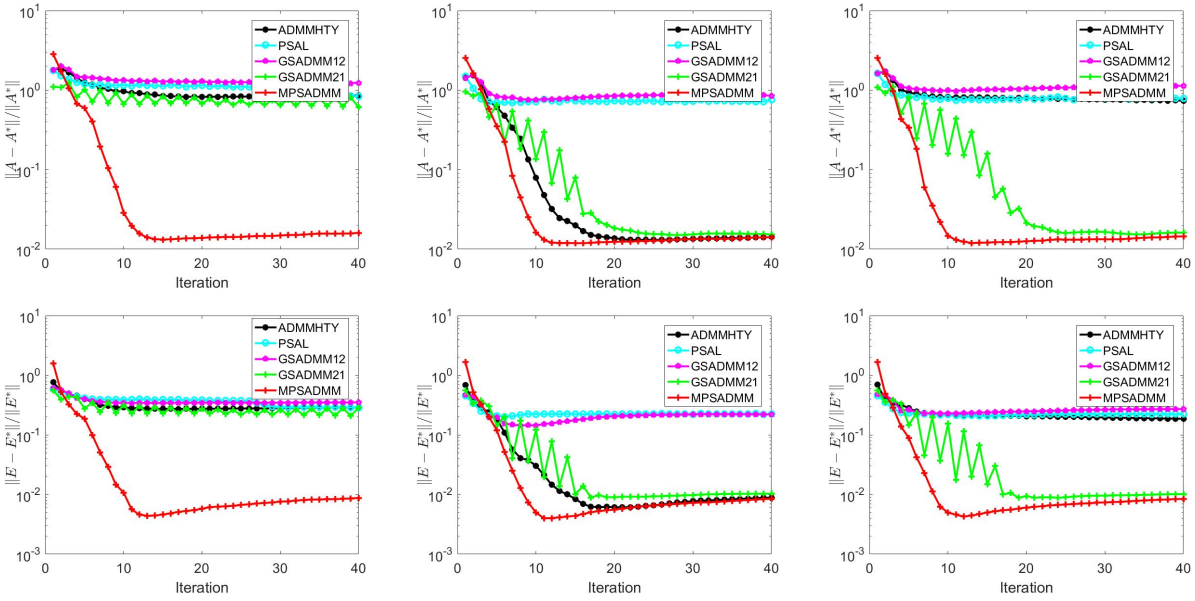

Figure 8: From above to below are the results with $err(A)$ and $err(E)$, respectively. From left to right are the results with (m, n_i) =(100,30), (100,50), (100,60), respectively.

In summary, the above experimental results indicate that the proposed algorithm is effective for solving RPCA problem, and its performance is competitive compared with some most efficient algorithms.

5 Conclusions

In this paper, motivated by the modified SC-PRSM proposed by Jiang et al., two new algorithms are proposed to solve the multi-block linearly constrained optimization, which are generalizations of the modified SC-PRSM. In addition, several popular ADMM-based algorithms can be seen as special cases of our algorithms. Preliminary numerical results on two types of problems are given, which illustrate that the performance of the proposed algorithm is competitive for solving multi-block linearly constrained optimization compared with

some popular ADMM-based algorithms.

References

[1] Bai, J., Li, J., Xu, F., and Zhang, H.: Generalized symmetric admm for separable convex optimization. Comput. Optim. Appl., 70:129-170, 2018.

[2] Chen, C., He, B., Ye, Y., and Yuan, X.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Program., 155(1-2):57-79, 2016.

[3] Corman, E., and Yuan, X.: A generalized proximal point algorithm and its convergence rate. SIAM J. Optim., 24:1614-1638, 2014.

[4] Chandrasekaran, V., Parrilo, P., and Willsky, A.: Latent variable graphical model selection via convex optimization. Ann. Statist. 40:1935-1967, 2012.

[5] Candès, E., Li, X., and Wright, J.: Robust principal component analysis? J. ACM. 58 (1):1-37, 2011.

[6] Facchinei, F., and Pang, J.: Finite Dimensional Variational Inequalities and Complementarity problems. Spri. Oper. Vol. I. Berlin: Springer, 2003.

[7] Gabay, D., and Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput. Math. Appl., 2:17-40, 1976.

[8] Glowinski, R., Oden, J., and Tallec, P.: Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics. [M]// Augmented Lagrangian and operator-splitting methods in nonlinear mechanics. SIAM, 1989.

[9] Glowinski, R., Kärkkäinen, T., and Majava, K.: On the convergence of operator-splitting methods. SIAM. Vol. Appl. 2003.

[10] He, B., Ma, F., and Yuan, X.: Convergence study on the symmetric version of ADMM with larger step sizes. SIAM J. Imaging Sci. 9:1467-1501, 2016.

[11] He, B., Liu, H., Wang, Z., and Yuan, X.: A strictly contractive Peaceman-Rachford splitting method for convex programming. SIAM J. Optim., 24:1011-1040, 2014.

[12] He, B.: Parallel splitting augmented lagrangian methods for monotone structured variational inequalities. Comput. Optim. Appl., 42:195-212, 2009.

[13] He, B., Hou, L., and Yuan, X.: On full jacobian decomposition of the augmented lagrangian method for separable convex programming. SIAM J. Optim., 25(4):2274-2312, 2015.

[14] He, B., Tao, M., Xu, M., and Yuan, X.: An alternating direction-based contraction method for linearly constrained separable convex programming problems. Optimization, 62(4):573-596, 2013.

[15] He, B., Tao, M., and Yuan, X.: Alternating direction method with gaussian back substitution for separable convex programming. SIAM J. Optim., 22(2):313-340, 2012.

[16] He, B., Tao, M., and Yuan, X.: A splitting method for separable convex programming. IMA J. Numer. Anal., 35(1):394-426, 2015.

[17] He, B., Xu, M., and Yuan, X.: Block-wise admm with a relaxation factor for multiple-block convex programming. J. Oper. Res. Soc. China, 6(4):485-506, 2018.

[18] Han, D., Yuan, X., and Zhang, W.: An augmented lagrangian based parallel splitting method for separable convex minimization with applications to image processing. Math. Comput., 83:2263-2291, 2014.

[19] Han, D., Kong, W., and Zhang, W.: A partial splitting augmented lagrangian method for low patch-rank image decomposition. J. Math. Imaging Vis., 51:145-160, 2015.

[20] Hou, L., He, H., and Yang, J.: A partially parallel splitting method for multiple-block separable convex programming with applications fto robust pca. Comput. Optim. Appl., 63(1):273-303, 2016.

[21] Jiang, S., and Li, M.: A modified strictly contractive peaceman-rachford splitting method for multi-block separable convex programming. J. Ind. Manag. Optim. Vol. 14: 397-412, 2018.

[22] Lions, P., and Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal., 16: 964-979, 2976.

[23] Larsen, R.: Lancaos bidiagonalization with partial reorthogonalization. Department of Computer Science, Aarhus University. Available at http://soi.stanford.edu/rmunk/PROPACK. Technical report, DAIMI PB-357, 1998.

[24] McLachlan, G.: Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience, 2004.

[25] Ma, F., and Ni, M.: A class of customized proximal point algorithms for linearly constrained convex optimization. Comp. Appl. Math. 2016.

[26] Peaceman, D., and Rachford, H.: The numerical solution of parabolic elliptic differential equations. SIAM J. Math. Appl., 3:28-41, 1955.

[27] Shen, Y., Zuo, Y., and Yu, A.: A Partial PPa S-ADMM for Multi-Block for Separable Convex Optimization with Linear Constraints. Optimization, http://www.optimization-line.org/DB_HTML/2020/03/7707.html.

[28] Shen, Y., Zhang, X., and Zhang, X.: A partial PPA block-wise ADMM for multi-block linearly constrained separable convex optimization. Optimization, DOI:10.1080/02331934.2020.1728756, 2020.

[29] Tao, M., and Yuan, X.: Recovering low-rank and sparse components of matrices from incomplete and noisy observations. SIAM J. Optim. 21:57-81, 2011.

[30] Wang, J., ang Song, W.: An algorithm twisted from generalized admm for multi-block separable convex minimization models. J. Comput. Appl. Math., 309:342-358, 2017.