A Partial PPa S-ADMM for Multi-Block for Separable Convex Optimization with Linear Constraints

Yuan Shen[†] [1] **Yannian Zuo** [†2] **Aolin Yu**[‡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: The symmetric alternating direction method of multipliers (S-ADMM) is a classical effective method for solving two-block separable convex optimization. However, its convergence may not be guaranteed for multi-block case providing there is no additional assumptions. In this paper, we propose a partial PPa S-ADMM (referred as P3SADMM), which updates the Lagrange multiplier twice with suitable stepsizes and adds a special proximal term to the first subproblem, to solve the multi-block separable convex optimization. The P3SADMM partitions the data into two group variables so that one group consists of p block variables while the other has q block variables, where $p \geq 1$ and $q \geq 1$ are two integers. In addition, we add a special proximal terms to the subproblems in the first group, i.e., the residual subproblems are intact. At the end of each iteration, an extension step on all variables is performed with a fixed step size. Without any additional assumptions, we prove the global convergence of the P3SADMM. Finally, some numerical results show that our proposed method is effective and promising.

Keywords: Separable convex optimization, Multiple blocks, Symmetric ADMM, alternating direction method of multipliers, Proximal point algorithm.

1 Introduction

We consider the grouped multi-block separable convex optimization problem :

$$\begin{cases} \min \sum_{i=1}^{p} f_i(x_i) + \sum_{j=1}^{q} g_j(y_j) \\ s.t. \sum_{i=1}^{p} A_i x_i + \sum_{j=1}^{q} B_j y_j = c, \\ x_i \in \mathcal{X}, i = 1, ..., p, \\ y_j \in \mathcal{Y}, j = 1, ..., q, \end{cases} \tag{1.1}$$

where $f_i(x_i) : \mathcal{R}^{m_i} \to \mathcal{R}$, $g_j(y_j) : \mathcal{R}^{d_j} \to \mathcal{R}$ are closed convex function(may not be smooth); $A_i \in \mathcal{R}^{n \times m_i}$, $B_j \in \mathcal{R}^{n \times d_j}$ and $c \in \mathcal{R}^n$; $\mathcal{X}_i \in \mathcal{R}_i^m$ and $\mathcal{Y}_j \in \mathcal{R}_j^d$ are closed convex sets; $p \geq 1$ and $q \geq 1$ are two integers. In this paper, it is assumed that the solution set of problem (1.1) is nonempty. In addition, it mildly assumed that all the matrices A_i, $i = 1, ..., p$, and B_j, $j = 1, ..., q$, have full column rank. We also introduce the following notations $\mathcal{A} = (A_1, ..., A_p)$, $\mathcal{B} = (B_1, ..., B_q)$, $x = (x_1^T, ...x_p^T)^T$, $y = (y_1^T, ..., y_q^T)$, $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times ...\mathcal{X}_p$, $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times ...\mathcal{Y}_q$ and $\Omega = \mathcal{X} \times \mathcal{Y} \times \mathcal{R}^n$.

The problem (1.1) has been extensively studied due to its wide application in different fields, such as sparse inverse covariance estimation problem [3], compressive sensing problem in signal processing [10], low rank and sparse representations [9] in image processing. One classical method to solve (1.1) is the Augmented Lagrangian method (ALM) [8], which minimizes the following augmented Lagrangian function:

$$\mathcal{L}_\beta = L(x, y, \lambda) + \frac{\beta}{2} \|\mathcal{A}x + \mathcal{B}y - c\|^2,$$

[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]; and the National Natural Science Foundation of China [grant numbers 11401295, 11726618].

[2]Email: zynlcg@163.com.

[3](Corresponding author)Email: 1208433174@qq.com.

where $\beta > 0$ is a penalty parameter, and the Lagrangian function $L(x, y, \lambda)$ is defined by

$$L(x, y, \lambda) = \sum_{i=1}^{p} f_i(x_i) + \sum_{j=1}^{q} g_j(y_j) - < \lambda, \mathcal{A}x + \mathcal{B}y - c >,$$

with the Lagrangian multiplier $\lambda \in \mathcal{R}^n$.

Then, the standard ALM procedure for solving (1.1) can be described as follows:

$$\begin{cases} (x^{k+1}, y^{k+1}) = \arg\min\{\mathcal{L}_\beta(x, y, \lambda^k) | x \in \mathcal{X}, y \in \mathcal{Y}\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.2}$$

The ALM is an overall good method due to its simple algorithmic framework and satisfactory convergence speed, however, ALM does not make full use of the separable structure of the objective function of (1.1) and hence, could not take advantage of the special properties of the component objective function $f_i(x_i)$ and $g_j(y_j)$. Therefore, this method may not be suitable for the problem (1.1).

One effective approach to overcome such difficulty is the alternating direction method of multiplies (ADMM) [5, 4]. Convex programming (1.1) has been extensively studied in the literature, researchers developed many numerical methods to solve it during the past few decades, which are mainly based on the well-known Douglas-Rachford splitting method [11, 12] and the Peachmen-Rachford splitting method [12, 13]. Based on the Douglas-Rachford splitting method [14, 15], we get the alternating direction method of multipliers (ADMM), which generates the iterative sequence via the following recursions:

$$\begin{cases} x^{k+1} = \arg\min\{\mathcal{L}_\beta(x, y^k, \lambda^k) | x \in \mathcal{X}\}, \\ y^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y, \lambda^k) | y \in \mathcal{Y}\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.3}$$

Based on another classic operator splitting method, ie., Peaceman-Pachford operator splitting method [16], one can derive the following method:

$$\begin{cases} x^{k+1} = \arg\min\{\mathcal{L}_\beta(x, y^k, \lambda^k) | x \in \mathcal{X}\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^k - c), \\ y^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y, \lambda^{k+\frac{1}{2}}) | y \in \mathcal{Y}\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.4}$$

While the global convergence of the alternating method of multipliers (1.3) can be established under very mild conditions [17], the convergence of the Peaceman-Rachford-based method (1.4) can not be guaranteed without further conditions [18]. To ensure the convergence, He at al. [19] proposed a modification of (1.4) by introducing a parameter τ to the update scheme of the dual variable λ in (1.4), yielding the following procedure:

$$\begin{cases} x^{k+1} = \arg\min\{\mathcal{L}_\beta(x, y^k, \lambda^k) | x \in \mathcal{X}\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^k - c), \\ y^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y, \lambda^{k+\frac{1}{2}}) | y \in \mathcal{Y}\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \tau\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.5}$$

The parameter τ in (1.5) plays the essential role in forcing the strict contractiveness of the generated sequence. Under the condition that $\tau \in (0, 1)$, they proved the same sublinear convergence rate as that for ADMM [20].

For seeking larger stepsizes, He et al. [6] proposed the symmetric ADMM (S-ADMM) with different stepsizes τ and s in (1.5). The iterating scheme of S-ADMM method is described as follows:

$$\begin{cases} x^{k+1} = \arg\min\{\mathcal{L}_\beta(x, y^k, \lambda^k) | x \in \mathcal{X}\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^k - c) \\ y^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y, \lambda^{k+\frac{1}{2}}) | y \in \mathcal{Y}\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - s\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.6}$$

and the stepsize (τ, s) were restricted into the domain:

$$(\tau, s) \in \mathcal{K} = \{(\tau, s) | \tau + s > 0, \tau \in (-1, 1), s \in (0, \frac{\sqrt{5}+1}{2}), |\tau| < 1 + s - s^2\}, \tag{1.7}$$

As elaborated in [6], the S-ADMM updates the Lagrangian multipliers twice with suitable and different stepsizes at each iteration. The main contribution is that the scheme (1.6) largely extends the domain of the step sizes (τ, s). What's more, numerical experimental results show that S-ADMM on solving the widely used basis pursuit model and the total-variational image debarring model significantly outperforms the original ADMM in terms of the CPU time and iteration number. However, this method has a nonsymmetric convergence domain of the stepsizes (τ, s) and still focuses on the two-block problem.

More recently, He at al. [7] proposed a novel ADMM-based algorithm without correction. By adding a special proximal term to a part of the subproblems, the algorithm performs the following updating scheme:

$$\begin{cases} x^{k+1} = \arg\min\{\mathcal{L}_\beta(x, y_1^k, ..., y_q^k, \lambda^k) | x \in \mathcal{X}\}, \\ y_j^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y_1^k, ..., y_q^k, \lambda^k) + \frac{\sigma\beta}{2} \parallel B_j(y_j - y_j^k) \parallel^2 | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.8}$$

with $\sigma > q - 1$. The splitting method that by adding certain proximal terms, allowed subproblems in the first group to be computed firstly and then the second subproblem to be solved in parallel, i.e., in a Jacobian fashion. The algorithm is suitable for exploiting properties of these individual function separably, resulting in subproblems which could easily enough have closed-from solutions if individual function is simple. An improvement of the new method over some pre-existing splitting methods is that no correction step is required. Moreover, its numerical performance has been verified on several practical problems, such as some sparse low-rank models and image painting problems.

Furthermore, He and Yuan [21] proposed an entitled block-wise ADMM(BADMM) which takes the following iterative scheme:

$$\begin{cases} x_i^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma_1\beta}{2} \parallel A_i(x_i - x_i^k) \parallel^2 | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ y_j^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y_1^k, ..., y_{j-1}^k, y_j, ..., y_q^k, \lambda^k) + \frac{\sigma_2\beta}{2} \parallel B_j(y_j - y_j^k) \parallel^2 | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.9}$$

where a proximal term similar to that in (1.8) was added to each subproblem to ensure the global convergence wit $\sigma_1 \in (p-1, +\infty)$, $\sigma_2 \in (q-1, +\infty)$. The numerical efficiency of BADMM was further improved by introducing a relaxation factor on the multiplier update. The new algorithm [22] restricts the stepsize $\tau = s \in (0, \frac{\sqrt{5}+1}{2})$, which takes the following iterative scheme:

$$\begin{cases} x_i^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma_1\beta}{2} \parallel A_i(x_i - x_i^k) \parallel^2 | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ y_j^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y_1^k, ..., y_{j-1}^k, y_j, ..., y_q^k, \lambda^k) + \frac{\sigma_2\beta}{2} \parallel B_j(y_j - y_j^k) \parallel^2 | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \lambda^{k+1} = \lambda^k - \tau\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.10}$$

with $\sigma_1 > p-1$,$\sigma_2 > q-1$. This method can be regarded as a refined version of the block-wise ADMM in [23] with a relaxation factor for iteratively updating its Lagrangian multiplier; or a block-wise extension of the original ADMM with a relaxation factor in [24]. The relaxation factor in (1.10) can generate step sizes larger than 1 and it usually accelerates the convergence of the original ADMM without additional computation.

Based on [6, 7, 25], Bai et al. [2] proposed a generalized symmetric ADMM (GS-ADMM) with wider convergence domain of the stepsizes to tackle the multi-block separable convex programming model (1.1). The algorithm framework can be described as follows:

$$\begin{cases} x_i^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_i, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma_1\beta}{2} \parallel A_i(x_i - x_i^k) \parallel^2 | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^k - c), \\ y_j^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y_1^k, ..., y_j, ..., y_q^k, \lambda^{k+\frac{1}{2}}) + \frac{\sigma_2\beta}{2} \parallel B_i(y_j - y_j^k) \parallel^2 | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \lambda^{k+1} = \lambda^{k+\frac{1}{2}} - s\beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.11}$$

In the above GS-ADMM, τ and s are two stepsizes parameters satisfying:

$$(\tau, s) \in \Gamma = \{(\tau, s) | \tau + s > 0, \tau \leq 1, -\tau^2 - s^2 - \tau s + \tau + s + 1 > 0\}, \tag{1.12}$$

and $\sigma_1 \in (p-1, +\infty), \sigma_2 \in (q-1, +\infty)$ are two proximal parameters for the regularization terms. This method partitions the date into two group variables. In GS-ADMM,the Gauss-Seidel fashion is taken for updating the two grouped variables, while the block variables within each group are updated in a Jacobi scheme, which would make the algorithm more be attractive and effective for solving big size problems. By adding proper proximal terms to the subproblems, they specify the domain of the stepsizes to guarantee the global convergence. The main contribution of [2] is that they provide a new convergence domain for the stepsizes of the dual variables, which is significantly larger than the convergence domains given in S-ADMM.

Based on (1.10), Shen et al. proposed a partial PPA block-wise ADMM (P3BAMM) [1] for solving multi-block linearly constrained separable convex optimization, that is:

$$\begin{cases} x_i^{k+1} = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma\beta}{2} \parallel A_i(x_i - x_i^k) \parallel^2 | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ y_j^{k+1} = \arg\min\{\mathcal{L}_\beta(x^{k+1}, y_1^k, ..., y_{j-1}^k, y_j, ..., y_q^k, \lambda^k) | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \lambda^{k+1} = \lambda^k - \beta(\mathcal{A}x^{k+1} + \mathcal{B}y^{k+1} - c), \end{cases} \tag{1.13}$$

and the primal variables were restricted to satisfy $q \leq 3$, $\sigma > q - 1$, $p \geq 1$. The algorithm adds proximal terms to the subproblems in the first group, i.e., the subproblems in the second group are intact. As the subproblems in the second group are unmodified, the resulting sequence might yield better quality as well as potentially faster convergence speed. However, the algorithm only allows at most three subproblems to be intact, which limits its applications for solving large-scale problems with multiple block variable.

Mainly motivated by the work of [1, 2], we put forward a partial PPA S-ADMM algorithm (P3SADMM) without correction,which provide a new convergence domain of stepsizes to solve the multi-block separable convex optimization. What's more, it enables arbitrary number of subproblems to be intact. Thus it potentially has even better numerical behavior than the existing ADMM-based algorithms.

The rest part of this paper is organized as follows. In section 2, some necessary notations are defined and some preliminaries are given for the theoretical analysis. Section 3 presents the new algorithm as well as its convergence properties. Then the preliminary experimental results are reported in the fourth section. Finally, we give some conclusions in Section 5.

2 Preliminaries

In this section, some useful notations will be defined. Then the variational inequalities are employed to characterize the optimality conditions of (1.1). Finally, we recall a useful result for later analysis.

Throughout this paper, let$\mathcal{R}, \mathcal{R}^n, \mathcal{R}^{m\times n}$ be the set of real numbers, the set of n dimensional real column vectors and the set $m \times n$ dimensional real matrices, respectively. The symbol $\parallel x \parallel_2$ denotes the Euclidean norm of $x \in \mathcal{R}^n$, which is defined by $\parallel x \parallel_2 = \sqrt{\langle x, x\rangle}$ with the standard inner product $\langle \cdot, \cdot \rangle$. Given a symmetric matrix G $\in \mathcal{R}^{m\times n}$, $\parallel x \parallel_G = \sqrt{\langle x, Gx\rangle}$ represents the weighted G-norm of x. We also use $\top$, I, and 0 to stand for the transpose, the identity matrix and the zero matrix with proper dimensions, respectively.

It is known that solving problem (1.1) is equivalent to finding a saddle point of (1.2). Let $(x^*, y^*, \lambda^*) \in \Omega$ be a saddle point of (1.2) where $x^* = (x_1^*, ..., x_p^*), y^* = (y_1^*, ..., y_q^*)$. Then, for any $(x, y, \lambda) \in \Omega$, we have

$$L(x^*, y^*, \lambda) \leq L(x^*, y^*, \lambda^*) \leq L(x, y, \lambda^*).$$

For the upcoming theoretical analysis, we need the following variational inequalities to characterize the fist-order optimality condition of (1.1): finding $(x_i^*, ..., x_p^*, y_1^*, ..., y_q^*, \lambda^*) \in \Omega$, such that

$$\begin{cases} x_i^* \in \mathcal{X}_i, \quad f_i(x_i) - f_i(x_i^*) + (x_i - x_i^*)^T(-A_i^T\lambda^*) \geq 0, \quad \forall x_i \in \mathcal{X}_i, i = 1, ..., p, \\ y_j^* \in \mathcal{Y}_j, \quad g_j(y_j) - g_j(y_j^*) + (y_j - y_j^*)^T(-B_j^T\lambda^*) \geq 0, \quad \forall y_j \in \mathcal{Y}_j, j = 1, ..., q, \\ \lambda^* \in \mathcal{R}^n, \quad (\lambda - \lambda^*)^T(\mathcal{A}x^* + \mathcal{B}y^* - c) \geq 0, \quad \forall \lambda \in \mathcal{R}^n, \end{cases} \tag{2.1}$$

which can be rewritten as a variational inequality (VI):

$$VI(\Omega, \mathcal{F}, \theta), \quad w^* \in \Omega, \quad \theta(u) - \theta(u^*) + (w - w^*)^T\mathcal{F}(w^*) \geq 0, \quad \forall w \in \Omega. \tag{2.2}$$

where

$$f(x) = \sum_{i=1}^{p} f_i(x_i), \quad g(y) = \sum_{j=1}^{q} g_j(y_j), \quad \theta(u) = f(u) + g(u). \tag{2.3}$$

$$u = \begin{pmatrix} x_1 \\ \vdots \\ x_p \\ y_1 \\ \vdots \\ y_q \end{pmatrix}, \quad w = \begin{pmatrix} x_1 \\ \vdots \\ x_p \\ y_1 \\ \vdots \\ y_q \\ \lambda \end{pmatrix}, \quad \mathcal{F}(w) = \begin{pmatrix} -A_1^T\lambda \\ \vdots \\ -A_p^T\lambda \\ -B_1^T\lambda \\ \vdots \\ -B_q^T\lambda \\ \mathcal{A}x + \mathcal{B}y - c \end{pmatrix}. \tag{2.4}$$

Since the affine mapping $\mathcal{F}$ is skew-symmetric, i.e., $(w - \tilde{w})^T\{\mathcal{F}(w) - \mathcal{F}(\tilde{w})\} = 0$, holds for any w,$\tilde{w} \in \Omega$, hence $\mathcal{F}$ is monotone. For simplicity, we denote (2.2) by $VI(\Omega, \mathcal{F}, \theta)$ and its solutions set by Ω^* throughout this paper.

Theorem 2.1 *The solution set of the variational inequality $VI(\Omega, \mathcal{F}, \theta)$ can be expressed as*

$$\Omega^* = \bigcap_{w \in \Omega} \{\tilde{w} \in \Omega | \theta(u) - \theta(\tilde{u}) + (w - \tilde{w})^T\mathcal{F}(w) \geq 0\}.$$

Lemma 2.1 *For any matrices A and B defined in Section 1, we have the following conclusion:*

$$p \ \operatorname{diag}(A^{\mathrm{T}}A) \succeq A^TA \quad \text{and} \quad q \ \operatorname{diag}(B^{\mathrm{T}}B) \succeq B^TB. \tag{2.5}$$

where $\operatorname{diag}(A^{\mathrm{T}}A)$ *and* $\operatorname{diag}(B^{\mathrm{T}}B)$ *are defined by*

$$\operatorname{diag}(A^{\mathrm{T}}A) = \begin{bmatrix} A_1^TA_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_p^TA_p \end{bmatrix} \quad \text{and} \quad \operatorname{diag}(B^{\mathrm{T}}B) = \begin{bmatrix} B_1^TB_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & B_q^TB_q \end{bmatrix}. \tag{2.6}$$

respectively.

The assertion is trivial, thus the proof is omitted. Furthermore, from (2.6), we can obtain

$$\kappa \operatorname{diag}(A^{\mathrm{T}}A) - A^TA \succ 0 \quad \text{with} \quad \kappa > p \quad \text{and} \quad \kappa \operatorname{diag}(B^{\mathrm{T}}B) - B^TB \succ 0 \quad \text{with} \quad \kappa > q. \tag{2.7}$$

3 A partial PPa S-ADMM

In this section, we first describe the new algorithm, then its convergence results will be established in the framework of variational inequality.

3.1 Algorithm

The new algorithm is described as follows:

Algorithm :P3SADMM for (1.1).

Let the parameters $p \geq 1, \quad q \geq 1, \quad \alpha \in (0, 1 + \frac{1-\tau q-\sqrt{(1-\tau q)(\tau+s)[(1-\tau q)(\tau+s)-2(1-\tau q)+q(1-\tau)^2]+(1-\tau q)^2}}{(1-\tau q)(\tau+s)})$, $\sigma \in (p-1, +\infty), \quad (\tau, s) \in \mathcal{H} = \{(\tau, s)|\tau + s > 0, \tau \in (-1 - \frac{2}{\sqrt{q}}, -1 + \frac{2}{\sqrt{q}})\}$. With given $w^k = (x^k, y^k, \lambda^k)$, the new iterate w^{k+1} is generated via the following procedure:

$$\begin{cases} \bar{x}_i^k = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma\beta}{2} \parallel A_i(x_i - x_i^k \parallel^2) | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ \bar{\lambda}^{k+\frac{1}{2}} = \lambda^k - \tau\beta(\mathcal{A}\bar{x}^k + \mathcal{B}y^k - c), \\ \bar{y}_j^k = \arg\min\{\mathcal{L}_\beta(\bar{x}^k, y_1^k, ..., y_{j-1}^k, y_j, y_{j+1}^k, ..., y_q^k, \bar{\lambda}^{k+\frac{1}{2}}) | y_j \in \mathcal{Y}_j, j = 1, ..., q\}, \\ \bar{\lambda}^k = \bar{\lambda}^{k+\frac{1}{2}} - s\beta(\mathcal{A}\bar{x}^k + \mathcal{B}\bar{y}^k - c), \\ w^{k+1} = w^k - \alpha(w^k - \bar{w}^k). \end{cases} \tag{3.1}$$

Remark 3.1 *In P3SADMM method, the Lagrange multipliers update twice with suitable stepsizes,and the subproblems in the first group are added a special proximal term. In addition, an extension step is applied on all the variables at the end of each iteration in order to guarantee convergence. More importantly, τ changes with q, and the graph between τ and s is an infinitely extended strip region. When q tends to be infinity, the stepsizes region becomes narrower and narrower, finally becomes a ray with $\tau = -1$,$s = 1$. The stepsizes region for q up to 4 is shown in Fig.1.*

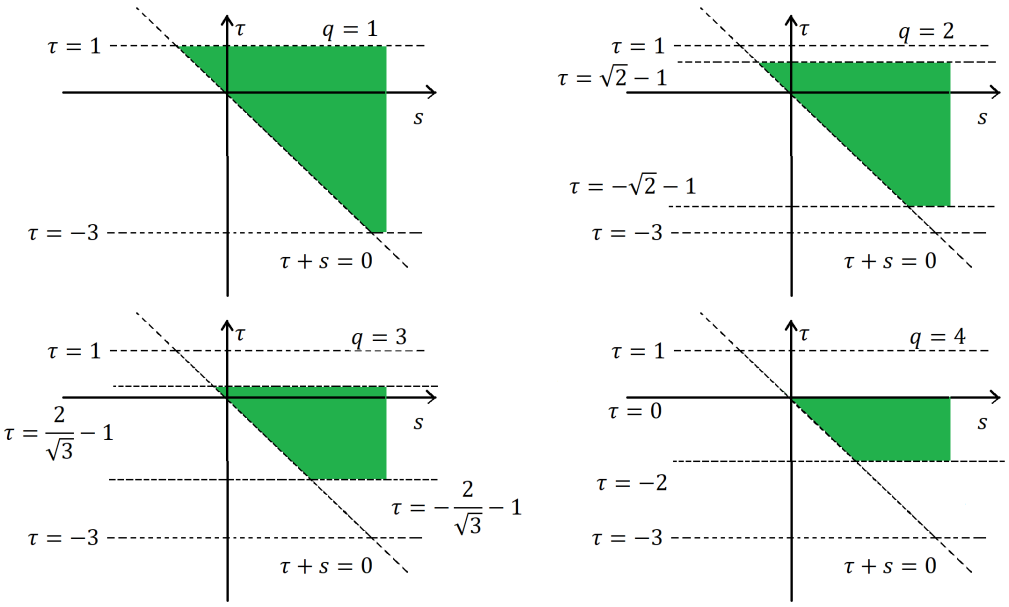

Figure 1: Stepsize region

Remark 3.2 *Note the extension step $w^{k+1} = w^k - \alpha(w^k - \bar{w}^k)$ is equivalent to*

$$w^{k+1} = (1 - \alpha)w^k + \alpha\bar{w}^k,$$

so it can also be regarded as a relaxed step with fixed factor α.

We do some simple comparisons between the proposed algorithm and some related algorithms:

• Compared with the partial PPA block-wise ADMM (P3BADMM) [1], we only need to update the Lagrange multiplier twice with suitable stepsizes. More importantly, the new algorithm enables arbitrary number of subproblems to be intact while the P3BADMM only allows at most three subproblems to be intact.

• Compared with GS-ADMM [2] in which the proximal terms are required in all subproblems, in our algorithm, we only need to add proximal terms to the subproblems in the first group. At the cost of this, there is an extension step on all variables at the end of each iteration. In addition, the convergence domain $\mathcal{H}$ in our algorithm is significantly larger than the domain Γ given in (1.12).

3.2 Convergence analysis

In this section, the convergence properties of the algorithm will be analyzed. Before analyzing the convenience, the following auxiliary variables are introduced:

$$\tilde{x}^k = \bar{x}^k, \quad \tilde{y}^k = \bar{y}^k, \tag{3.2}$$

$$\tilde{\lambda}^k = \lambda^k - \beta(\mathcal{A}\bar{x}^k + \mathcal{B}y^k - c), \tag{3.3}$$

$$\tilde{u}^k = \begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \end{pmatrix}, \quad \tilde{w}^k = \begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \\ \tilde{\lambda}^k \end{pmatrix}. \tag{3.4}$$

The scheme (3.1)can be written as a prediction-correction fashion:

Prediction :

$$\begin{cases} \tilde{x}_i^k = \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma\beta}{2} \parallel A_i(x_i - x_i^k \parallel^2) | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \\ \tilde{\lambda}^k = \lambda^k - \beta(\mathcal{A}\tilde{x}^k + \mathcal{B}y^k - c), \\ \lambda^{k+\frac{1}{2}} = \lambda^k - \tau(\lambda^k - \tilde{\lambda}^k), \\ \tilde{y}_j^k = \arg\min\{\mathcal{L}_\beta(\tilde{x}^k, y_1^k, ..., y_{j-1}^k, y_j, y_{j+1}^k, ..., y_q^k, \lambda^{k+\frac{1}{2}}) | y_j \in \mathcal{Y}_j, j = 1, ..., q\}. \end{cases} \tag{3.5}$$

Correction:

$$w^{k+1} = w^k - \alpha M(w^k - \tilde{w}^k),$$

where

$$M = \left[\begin{array}{c|ccc} I & & & \\ \hline & I & & \\ & & \ddots & \\ & & & I \\ & -s\beta B_1 & \cdots & -s\beta B_q \quad (\tau+s)I \end{array}\right]. \tag{3.6}$$

Lemma 3.1 *For the auxiliary variable $\tilde{w}^k$ defined in (3.4), we have*

$$\tilde{w}^k \in \Omega, \quad h(u) - h(\tilde{u}^k) + (w - \tilde{w}^k)^T \mathcal{F}(\tilde{w}^k) \geq (w - \tilde{w}^k)^T Q(\tilde{w}^k - w^k), \quad w \in \Omega. \tag{3.7}$$

where

$$Q = \begin{bmatrix} H_x & 0 \\ 0 & \tilde{Q} \end{bmatrix}, \tag{3.8}$$

with

$$H_x = \beta \begin{bmatrix} \sigma A_1^T A_1 & -A_1^T A_2 & \cdots & -A_1^T A_p \\ -A_2^T A_1 & \sigma A_2^T A_2 & \cdots & -A_2^T A_p \\ \vdots & \vdots & \ddots & \vdots \\ -A_p^T A_1 & -A_p^T A_2 & \cdots & \sigma A_p^T A_p \end{bmatrix}, \tag{3.9}$$

$$\tilde{Q} = \begin{bmatrix} \beta B_1^T B_1 & 0 & \cdots & 0 & -\tau B_1^T \\ 0 & \beta B_2^T B_2 & \cdots & 0 & -\tau B_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta B_q^T B_q & -\tau B_q^T \\ -B_1 & -B_2 & \cdots & -B_q & \frac{1}{\beta} I \end{bmatrix}. \tag{3.10}$$

Proof Note that the x_i-subproblem in (3.5) can be written as

$$\begin{aligned} \tilde{x}_i^k &= \arg\min\{\mathcal{L}_\beta(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_p^k, y^k, \lambda^k) + \frac{\sigma\beta}{2} \parallel A_i(x_i - x_i^k \parallel^2) | x_i \in \mathcal{X}_i, i = 1, ..., p\} \\ &= \arg\min\{f_i(x_i) - A_i^T \lambda^k x_i + \frac{\beta}{2} \parallel A_i(x_i - x_i^k) + \mathcal{A}x^k + \mathcal{B}y^k - C \parallel^2 \\ &+ \frac{\sigma\beta}{2} \parallel A_i(x_i - x_i^k) \parallel^2 | x_i \in \mathcal{X}_i, i = 1, ..., p\}, \end{aligned}$$

where some constant terms in the objective function are ignored. Its optimality condition can be written as:

$$\tilde{x}_i^k \in \mathcal{X}_i, f(x_i) - f(\tilde{x}_i^k) + (x_i - \tilde{x}_i^k)^T \{-A_i^T \lambda^k - \beta A_i^T \mathcal{A}(\tilde{x}_i^k - x_i^k) + (\sigma + 1)\beta A_i^T A_i(\tilde{x}_i^k - x_i^k)\} \geq 0, \forall x_i \in \mathcal{X}_i. \tag{3.11}$$

Plugging $\lambda^k = \tilde{\lambda}^k + \beta(\mathcal{A}\tilde{x}^k + \mathcal{B}y^k - C)$(see (3.3)) into (3.11),we obtain:

$$\tilde{x}_i^k \in \mathcal{X}_i, f(x_i) - f(\tilde{x}_i^k) + (x_i - \tilde{x}_i^k)^T \{-A_i^T \tilde{\lambda}^k - \beta A_i^T \mathcal{A}(\tilde{x}_i^k - x_i^k) + (\sigma + 1)\beta A_i^T A_i(\tilde{x}_i^k - x_i^k)\} \geq 0, \forall x_i \in \mathcal{X}_i. \tag{3.12}$$

Similarly, the $\tilde{y}_j$ satisfies

$$\tilde{y}_j^k \in \mathcal{Y}_j, g(y_j) - g(\tilde{y}_j^k) + (y_j - \tilde{y}_j^k)^T \{-B_j^T \lambda^{k+\frac{1}{2}} + \beta B_j^T [B_j(\tilde{y}_j^k - y_j^k) + \mathcal{A}\tilde{x}^k + \mathcal{B}y^k - C]\} \geq 0, \forall y_j \in \mathcal{Y}_j,$$

or equivalently

$$\tilde{y}_j^k \in \mathcal{Y}_j, g(y_j) - g(\tilde{y}_j^k) + (y_j - \tilde{y}_j^k)^T \{-B_j^T \tilde{\lambda}^k + \beta B_j^T B_j(\tilde{y}_j^k - y_j^k) - \tau B_j^T(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \forall y_j \in \mathcal{Y}_j. \tag{3.13}$$

Finally, the equation (3.3) can be rewritten as

$$(A\tilde{x}^k + B\tilde{y}^k - C) - B(\tilde{y}^k - y^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) = 0.$$

which is equivalent to

$$(\lambda - \tilde{\lambda}^k)^T \{(A\tilde{x}^k + B\tilde{y}^k - C) - B(\tilde{y}^k - y^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \forall \lambda \in \mathcal{R}^n. \tag{3.14}$$

The assertion is arrived by combining (3.12), (3.13) and (3.14). □

Lemma 3.2 *Let $\sigma \in (p - 1, +\infty)$, $\tau + s > 0$ and $\tau < \frac{1}{q}$, the matrices Q and M are defined in (3.8) and (3.6)respectively. We further define*

$$H = QM^{-1}. \tag{3.15}$$

then H is positive definite.

Proof For the matrix H defined in (3.15), we have

$$H = \begin{bmatrix} H_x & 0 \\ 0 & \tilde{H} \end{bmatrix}, \tag{3.16}$$

with H_x is defined in (3.9) and

$$\tilde{H} = \left[\begin{array}{cccc|c} (1 - \frac{\tau s}{\tau+s})\beta B_1^T B_1 & -\frac{\tau s}{\tau+s}\beta B_1^T B_2 & \cdots & -\frac{\tau s}{\tau+s}\beta B_1^T B_q & -\frac{\tau}{\tau+s} B_1^T \\ -\frac{\tau s}{\tau+s}\beta B_2^T B_1 & (1 - \frac{\tau s}{\tau+s})\beta B_2^T B_2 & \cdots & -\frac{\tau s}{\tau+s}\beta B_2^T B_q & -\frac{\tau}{\tau+s} B_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{\tau s}{\tau+s}\beta B_q^T B_1 & -\frac{\tau s}{\tau+s}\beta B_q^T B_2 & \cdots & (1 - \frac{\tau s}{\tau+s})\beta B_q^T B_q & -\frac{\tau}{\tau+s} B_q^T \\ \hline -\frac{\tau}{\tau+s} B_1 & -\frac{\tau}{\tau+s} B_2 & \cdots - \frac{\tau}{\tau+s} B_1 & -\frac{\tau}{\tau+s} B_q & \frac{1}{\beta(\tau+s)} I \end{array}\right], \tag{3.17}$$

We only need to show that the blocks H_x and $\tilde{H}$ are positive definite. Note that

$$H_x = \beta \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_p \end{bmatrix}^T H_{x,0} \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_p \end{bmatrix}, \tag{3.18}$$

where

$$H_{x,0} = \begin{bmatrix} \sigma I & -I & \cdots & -I \\ -I & \sigma I & \cdots & -I \\ \vdots & \vdots & \ddots & \vdots \\ -I & -I & \cdots & \sigma I \end{bmatrix}_{p\times p}. \tag{3.19}$$

If $\sigma \in (p-1, +\infty)$, $H_{x,0}$ is positive definite. Then, it follows from (3.18) that H_x is positive definite if $\sigma \in (p-1, +\infty)$ and all $A_i, i = 1, ..., p$ have full column rank.

Now, note that the matrix $\tilde{H}$ can be decomposed as

$$\tilde{H} = \tilde{D}^T \tilde{H}_0 \tilde{D}, \tag{3.20}$$

where

$$\tilde{D} = \begin{bmatrix} \beta^{\frac{1}{2}} B_1 & & & & \\ & \beta^{\frac{1}{2}} B_2 & & & \\ & & \ddots & & \\ & & & \beta^{\frac{1}{2}} B_q & \\ & & & & \beta^{-\frac{1}{2}} I \end{bmatrix}, \tag{3.21}$$

and

$$\tilde{H}_0 = \begin{bmatrix} (1-\frac{\tau s}{\tau+s})I & -\frac{\tau s}{\tau+s}I & \cdots & -\frac{\tau s}{\tau+s}I & -\frac{\tau}{\tau+s}I \\ -\frac{\tau s}{\tau+s}I & (1-\frac{\tau s}{\tau+s})I & \cdots & -\frac{\tau s}{\tau+s}I & -\frac{\tau}{\tau+s}I \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{\tau s}{\tau+s}I & -\frac{\tau s}{\tau+s}I & \cdots & (1-\frac{\tau s}{\tau+s})I & -\frac{\tau}{\tau+s}I \\ -\frac{\tau}{\tau+s}I & -\frac{\tau}{\tau+s}I & \cdots & -\frac{\tau}{\tau+s}I & \frac{1}{\tau+s}I \end{bmatrix}, \tag{3.22}$$

According to the fact that

$$\begin{aligned} &\begin{bmatrix} I & & \tau I \\ & \ddots & \\ & & \tau I \\ & & I \end{bmatrix} \tilde{H}_0 \begin{bmatrix} I & & \tau I \\ & \ddots & \\ & & \tau I \\ & & I \end{bmatrix}^T \\ &= \left[\begin{array}{cccc|c} (1-\tau)I & -\tau I & \cdots & -\tau I & 0 \\ -\tau I & (1-\tau)I & \cdots & -\tau I & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\tau I & -\tau I & \cdots & (1-\tau)I & 0 \\ \hline 0 & 0 & \cdots & 0 & \frac{1}{\tau+s}I \end{array}\right] \\ &= \begin{bmatrix} \mathrm{diag}(\mathrm{EE^T}) - \tau \mathrm{EE^T} & 0 \\ 0 & \frac{1}{\tau+s}I \end{bmatrix}. \end{aligned}$$

where

$$E = \begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix}.$$

we have $\tilde{H}_0$ is positive definite if and only if $\tau < \frac{1}{q}$ and $\tau + s > 0$. Then, it follows from (3.20) that$\tilde{H}$ is positive definite, if $\tau + s > 0, \tau < \frac{1}{q}$ and all the matrix $B_j, j = 1, ..., q$, have full column rank.
The assertion is proved. □

Lemma 3.3 *Let*

$$\sigma \in (p-1, +\infty), \alpha \in (0, 1 + \frac{1 - \tau q - \sqrt{(1-\tau q)(\tau+s)[(1-\tau q)(\tau+s) - 2(1-\tau q) + q(1-\tau)^2] + (1-\tau q)^2}}{(\tau+s)(1-\tau q)}),$$

$$\tau + s > 0, \tau \in (-1 - \frac{2}{\sqrt{q}}, -1 + \frac{2}{\sqrt{q}}). \tag{3.23}$$

and the matrices Q and M are defined in (3.8) and (3.6) respectively. We further define

$$G = Q^T + Q - \alpha M^T H M.$$

then the matrix G is positive definite.

Proof First, we have

$$\begin{aligned}
G &= Q^T + Q - \alpha M^T H M \\
&= Q^T + Q - \alpha Q^T M \\
&= \left[\begin{array}{c|ccccc}
(2-\alpha)H_x & & & & & \\
\hline
 & (2-\alpha-\alpha s)\beta B_1^T B_1 & \cdots & -\alpha s\beta B_1^T B_q & [\alpha(\tau+s)-(\tau+1)]B_1^T \\
 & -\alpha s\beta B_2^T B_1 & \cdots & -\alpha s\beta B_2^T B_q & [\alpha(\tau+s)-(\tau+1)]B_2^T \\
 & & \ddots & & \\
 & -\alpha s\beta B_q^T B_1 & \cdots & (2-\alpha-\alpha s)\beta B_q^T B_q & [\alpha(\tau+s)-(\tau+1)]B_q^T \\
 & [\alpha(\tau+s)-(\tau+1)]B_1 & \cdots & [\alpha(\tau+s)-(\tau+1)]B_q & \frac{2-\alpha(\tau+s)}{\beta} I
\end{array}\right] \\
&= \begin{bmatrix} G_{11} & 0 \\ 0 & G_{22} \end{bmatrix},
\end{aligned}$$

where

$$G_{11} = (2-\alpha)H_x,$$

$$G_{22} = \begin{bmatrix}
(2-\alpha-\alpha s)\beta B_1^T B_1 & \cdots & -\alpha s\beta B_1^T B_q & [\alpha(\tau+s)-(\tau+1)]B_1^T \\
-\alpha s\beta B_2^T B_1 & \cdots & -\alpha s\beta B_2^T B_q & [\alpha(\tau+s)-(\tau+1)]B_2^T \\
 & \ddots & & \\
-\alpha s\beta B_q^T B_1 & \cdots & (2-\alpha-\alpha s)\beta B_q^T B_q & [\alpha(\tau+s)-(\tau+1)]B_q^T \\
[\alpha(\tau+s)-(\tau+1)]B_1 & \cdots & [\alpha(\tau+s)-(\tau+1)]B_q & \frac{2-\alpha(\tau+s)}{\beta} I
\end{bmatrix},$$

Recalling (3.18), and noticing $\alpha \in (0, 2)$, the submatrix G_{11} is positive definite, we then only need to show the positive definiteness of G_{22}.

Noticing that the matrix G_{22} can be decomposed as

$$G_{22} = \tilde{D}^T \tilde{G}_0 \tilde{D}, \tag{3.24}$$

where $\tilde{D}$ is defined in (3.21) and

$$\tilde{G}_0 = \begin{bmatrix}
(2-\alpha s-\alpha)I & -\alpha s I & \cdots & -\alpha s I & [\alpha(\tau+s)-(\tau+1)]I \\
-\alpha s I & (2-\alpha s-\alpha)I & \cdots & -\alpha s I & [\alpha(\tau+s)-(\tau+1)]I \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
-\alpha s I & -\alpha s I & \cdots & (2-\alpha s-\alpha)I & [\alpha(\tau+s)-(\tau+1)]I \\
[\alpha(\tau+s)-(\tau+1)]I & [\alpha(\tau+s)-(\tau+1)]I & \cdots & [\alpha(\tau+s)-(\tau+1)]I & [2-\alpha(\tau+s)]I
\end{bmatrix}, \tag{3.25}$$

and using the fact that

$$\begin{aligned}
&\begin{bmatrix} I & & I \\ & \ddots & I \\ & & I \end{bmatrix} \tilde{G}_0 \begin{bmatrix} I & & I \\ & \ddots & I \\ & & I \end{bmatrix}^T \\
&= \begin{bmatrix} (1-\tau)(2-\alpha)I & -\tau(2-\alpha)I & \cdots & -\tau(2-\alpha)I & (1-\tau)I \\ -\tau(2-\alpha)I & (1-\tau)(2-\alpha)I & \cdots & -\tau(2-\alpha)I & (1-\tau)I \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\tau(2-\alpha)I & -\tau(2-\alpha)I & \cdots & (1-\tau)(2-\alpha)I & (1-\tau)I \\ (1-\tau)I & (1-\tau)I & \cdots & (1-\tau)I & [2-\alpha(\tau+s)]I \end{bmatrix} \\
&= \bar{G}.
\end{aligned}$$

the remaining task is to show:

As q is a positive integer and $(\tau, s) \in \mathcal{H}$, $\alpha \in (0, 2)$, it is easy to verify that

$$\begin{aligned}
\bar{G} \succ 0 &\Leftrightarrow |\bar{G}| = (2-\alpha)^{q-1}\{\alpha^2(\tau+s)(1-\tau q) - 2\alpha(1-\tau q)(\tau+s+1) + 4(1-\tau q) - q(1-\tau)^2\} > 0 \\
&\Leftrightarrow \alpha^2(\tau+s)(1-\tau q) - 2\alpha(1-\tau q)(\tau+s+1) + 4(1-\tau q) - q(1-\tau)^2 > 0 \\
&\Leftrightarrow \begin{cases} \Delta = 4(1-\tau q)^2(\tau+s+1)^2 - 4(\tau+s)(1-\tau q)[4(1-\tau q) - q(1-\tau)^2] < 0 & (1) \\ or \quad \alpha > 1 + \frac{1-\tau q+\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} & (2) \\ or \quad \alpha > 1 + \frac{1-\tau q-\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} & (3) \end{cases}
\end{aligned}$$

Obviously (1) doesn't work. Then, analyze the remaining two cases.

If (2) works, we can get that:

$$\begin{aligned}
\alpha > 1 + \frac{1-\tau q+\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} &\Leftrightarrow 2 > 1 + \frac{1-\tau q+\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} \\
&\Leftrightarrow 1 > \frac{1-\tau q+\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} \\
&\Leftrightarrow 4(1-\tau q)^2(\tau+s+1)^2 > \Delta \\
&\Leftrightarrow 0 > q(\tau+s)(1-\tau)^2
\end{aligned}$$

Noticing $q \geq 1$, $\tau + s > 0$, (2) is untenable.

Finally, for the third case, we have

$$\begin{aligned}
\alpha > 1 + \frac{1-\tau q-\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} &\Leftrightarrow 1 + \frac{1-\tau q-\sqrt{\frac{\Delta}{4}}}{(\tau+s)(1-\tau q)} > 0 \\
&\Leftrightarrow 4(1-\tau q)^2(\tau+s+1)^2 > \Delta \\
&\Leftrightarrow 4(1-\tau q) - q(1-\tau)^2 > 0 \\
&\Leftrightarrow \tau^2 q + 2\tau q + q - 4 < 0 \\
&\Leftrightarrow (\tau+1)^2 < \frac{4}{q} \\
&\Leftrightarrow -\frac{2}{\sqrt{q}} - 1 < \tau < \frac{2}{\sqrt{q}} - 1
\end{aligned}$$

Therefor, the assertion is proved. □

Lemma 3.4 *Let $\{w^k\}$ be generated by algorithm (3.1). Then, we have*

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k) \geq \frac{1}{2\alpha}(\| w - w^{k+1} \|_H^2 - \| w - w^k \|_H^2) + \frac{1}{2} \| w - \tilde{w}^k \|_G^2, \forall w \in \Omega. \tag{3.26}$$

Proof According to $Q = HM$ and the relation (3.6) ,(3.7)can be written as

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T \mathcal{F}(\tilde{w}^k) \geq \frac{1}{\alpha}(w - \tilde{w}^k)^T H(w^k - w^{k+1}), \forall w \in \Omega. \tag{3.27}$$

Applying the identity

$$(a-b)^T H(c-d) = \frac{1}{2}(\| a-d \|_H^2 - \| a-c \|_H^2) + \frac{1}{2}(\| c-b \|_H^2 - \| d-b \|_H^2) \tag{3.28}$$

to the right-hand sight of (3.27) with

$$a = w, b = \tilde{w}^k, c = w^k, d = w^{k+1}$$

we get

$$(w - \tilde{w}^k)^T H(w^k - w^{k+1}) = \frac{1}{2}(\| w - w^{k+1} \|_H^2 - \| w - w^k \|_H^2) + \frac{1}{2}(\| w^k - \tilde{w}^k \|_H^2 - \| w^{k+1} - \tilde{w}^k \|_H^2) \tag{3.29}$$

For the terms in the last round bracket of the right hand side of (3.28), we have

$$\begin{aligned}
&\| w^k - \tilde{w}^k \|_H^2 - \| w^{k+1} - \tilde{w}^k \|_H^2 \\
=&\| w^k - \tilde{w}^k \|_H^2 - \| (w^k - \tilde{w}^k) - (w^k - w^{k+1}) \|_H^2 \\
=&\| w^k - \tilde{w}^k \|_H^2 - \| (w^k - \tilde{w}^k) - \alpha M(w^k - \tilde{w}^k) \|_H^2 \\
=& 2\alpha(w^k - \tilde{w}^k)^T HM(w^k - \tilde{w}^k) - \alpha^2 (w^k - \tilde{w}^k)^T M^T HM(w^k - \tilde{w}^k) \\
=& \alpha(w^k - \tilde{w}^k)^T (Q^T + Q - \alpha M^T HM)(w^k - \tilde{w}^k) \\
=& \alpha \| w^k - \tilde{w}^k \|_G^2 .
\end{aligned}$$

The lemma is proved. □

Theorem 3.1 *For the sequence $\{w^k\}$ generated by algorithm (3.1), the following inequality holds*

$$\| w^k - w^* \|_H^2 - \| w^{k+1} - w^* \|_H^2 \geq \alpha \| w^k - \tilde{w}^k \|_G^2 . \tag{3.30}$$

Proof Setting $w = w^*$ in (3.25), we get

$$\| w^k - w^* \|_H^2 - \| w^{k+1} - w^* \|_H^2 \geq \alpha \| w^k - \tilde{w}^k \|_G^2 + 2\alpha\{\theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T \mathcal{F}(\tilde{w}^k)\}. \tag{3.31}$$

Invoking the monotonicity of $\mathcal{F}$, we obtain

$$\theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T \mathcal{F}(\tilde{w}^k) \geq \theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T \mathcal{F}(w^*) \geq 0. \tag{3.32}$$

Obviously,the assertion (3.29) is arrived. □

Theorem 3.2 *Let $\{w^k\}$ be the sequence generated by the algorithm (3.1), then it converges to a solution of* (1.1).

Proof First according to (3.29), we have that the sequence $\{w^k\}$ is bounded and

$$\lim_{k \to \infty} \| w^k - \tilde{w}^k \|_G^2 = 0. \tag{3.33}$$

Therefore the sequence $\{w^k\}$ has at least one cluster point.

We then turn to show that every cluster point of $\{w^k\}$ is a solution of $VI(\Omega, \mathcal{F}, h)$. Let w^∞ be an arbitrary cluster point of the sequence $\{w^k\}$ and $\{w^{k_j}\}$ be the subsequence such that $w^{k_j} \to w^\infty$ as $j \to \infty$.

It follows from the assertion of Lemma 3.1 that

$$h(u) - h(\tilde{u}^{k_j}) + (w - \tilde{w}^{k_j})^T \mathcal{F}(\tilde{w}^{k_j}) \geq (w - \tilde{w}^{k_j})^T Q(w^{k_j} - \tilde{w}^{k_j}), \tag{3.34}$$

by taking $k = k_j$.

Plugging $k = k_j$ into (3.23), we obtain we obtain $w^{k_j} - \tilde{w}^{k_j} \to 0$.

Taking the limit over j in (3.34), noticing the continuity of $\mathcal{F}(\cdot)$, we can conclude that

$$\theta(u) - \theta(u^\infty) + (w - w^\infty)^T \mathcal{F}(w^\infty) \geq 0, \forall w \in \Omega. \tag{3.35}$$

thus w^∞ turns out to be a solution of $VI(\Omega, \mathcal{F}, h)$. Recalling the strictly contractive property of the original sequence $\{w^k\}$ (3.29) , then $\{w^k\}$ converges to w^∞.

The assertion is then proved. □

4 Numerical Experiments

In this section, the numerical performance of the proposed algorithm will be investigated for solving two types of problems. All the algorithms are coded in MATLAB R2015a and carried out on a desktop computer with Inter Core i7 CPU at 3.6GHz with 8 GB memory.

4.1 Numerical results on LCQP problem

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

$$\min_x f_1(x_1) + ... + f_m(x_m)$$

$$s.t. \quad A_1x_1 + ... + A_mx_m = c,$$

where $f_i(x_i) = \frac{1}{2}x_i^T H_i x_i + x_i^T q_i (i = 1, ..., m) \in \mathcal{R}^{m_i} \to \mathcal{R}$ 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

$$f(x) = \sum_{i=1}^{m} f_i(x_i), x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}, w = \begin{pmatrix} x \\ \lambda \end{pmatrix}, A = \begin{pmatrix} A_1 & \cdots & A_m \end{pmatrix}. \tag{4.1}$$

To show the efficiency of the proposed algorithm, the numerical experiments of solving the problem (4.1) are carried out by comparing the new algorithm with the other three efficient methods: the Block-wise ADMM with Relaxation factor [22] (denoted by BADMMR); the Generalized Symmetric ADMM [2] (denoted by GSADMM); the partial PPA block-wise ADMM [1] (denoted by P3BADMM).

As all the test algorithms are basically block-wise ADMM,in order to implement these algorithms,we first need to regroup the variables of (4.1). For our algorithm, when $m = 3$,we can regroup the variables by either $1 \sim 2$ or $2 \sim 1$ strategy; when $m \geq 4$, the grouping strategy can be $(m-3) \sim 3, (m-2) \sim 2$ or $(m-1) \sim 1$. For the other three test algorithms,there is no restriction on the grouping strategy,i.e., the grouping strategy can be $1 \sim (m-1), ..., (m-1) \sim 1$.

In order to save space,we only report the numerical results with $m = 3$. In this case,by two different ways of grouping the variables of (4.1), the proposed algorithm can lead to the following three schemes:

$$\begin{cases} \bar{x}_1 = \arg\min\{\frac{1}{2}x_1^T H_1 x_1 + x_1^T q_1 + < \lambda^k, A_1x_1 > + \frac{\beta}{2} \parallel A_1x_1 + A_2x_2^k + A_3x_3^k - c \parallel_F^2 + \frac{\sigma\beta}{2} \parallel A_1(x_1 - x_1^k) \parallel_F^2\}, \\ \bar{x}_2 = \arg\min\{\frac{1}{2}x_2^T H_2 x_2 + x_2^T q_2 + < \lambda^k, A_2x_2 > + \frac{\beta}{2} \parallel A_1x_1^k + A_2x_2 + A_3x_3^k - c \parallel_F^2 + \frac{\sigma\beta}{2} \parallel A_2(x_2 - x_2^k) \parallel_F^2\}, \\ \bar{\lambda}^{k+\frac{1}{2}} = \lambda^k - \tau\beta(A_1\bar{x}_1^k + A_2\bar{x}_2^k + A_3x_3^k - c), \\ \bar{x}_3 = \arg\min\{\frac{1}{2}x_3^T H_3 x_3 + x_3^T q_3 + < \bar{\lambda}^{k+\frac{1}{2}}, A_3x_3 > + \frac{\beta}{2} \parallel A_1\bar{x}_1^k + A_2\bar{x}_2^k + A_3x_3 - c \parallel_F^2\}, \\ \bar{\lambda}^k = \lambda^k - s\beta(A\bar{x} - c), \\ w^{k+1} = w^k - \alpha(w^k - \bar{w}^k), \quad \sigma > 1, \quad \tau + s > 0, \quad \tau \in (-3, 1), \quad \alpha \in (0, 1 + \frac{1-\sqrt{(\tau+s)(s-1)+1}}{\tau+s}) \end{cases} \tag{4.2}$$

$$\begin{cases} \bar{x}_1 = \arg\min\{\frac{1}{2}x_1^T H_1 x_1 + x_1^T q_1 + < \lambda^k, A_1x_1 > + \frac{\beta}{2} \parallel A_1x_1 + A_2x_2^k + A_3x_3^k + A_4x_4^k - c \parallel_F^2 + \frac{\sigma\beta}{2} \parallel A_1(x_1 - x_1^k) \parallel_F^2\}, \\ \bar{\lambda}^{k+\frac{1}{2}} = \lambda^k - \tau\beta(A_1\bar{x}_1^k + A_2x_2^k + A_3x_3^k - c), \\ \bar{x}_2 = \arg\min\{\frac{1}{2}x_2^T H_2 x_2 + x_2^T q_2 + < \lambda^{k+\frac{1}{2}}, A_2x_2 > + \frac{\beta}{2} \parallel A_1\bar{x}_1^k + A_2x_2 + A_3x_3^k - c \parallel_F^2\}, \\ \bar{x}_3 = \arg\min\{\frac{1}{2}x_3^T H_3 x_3 + x_3^T q_3 + < \bar{\lambda}^{k+\frac{1}{2}}, A_3x_3 > + \frac{\beta}{2} \parallel A_1\bar{x}_1^k + A_2x_2^k + A_3x_3 - c \parallel_F^2\}, \\ \bar{\lambda}^k = \lambda^k - s\beta(A\bar{x} - c), \\ w^{k+1} = w^k - \alpha(w^k - \bar{w}^k), \quad \sigma > 0, \quad \tau + s > 0, \quad \tau \in (-\sqrt{2} - 1, \sqrt{2} - 1), \\ \quad \alpha \in (0, 1 + \frac{1-2\tau-\sqrt{(1-2\tau)(\tau+s)[(1-2\tau)(\tau+s)-2(1-2\tau)+2(1-\tau)^2]-(1-2\tau)^2}}{(1-2\tau)(\tau+s)}) \end{cases} \tag{4.3}$$

Note that all the subproblems in (4.2) and (4.3) are unconstrained quadratic programming which can be efficiently solved. As the Hessian in each subproblem is fixed,we can boost the performance further by doing a Cholesky decomposition on these Hessian at the beginning,then the computation of subproblems at

each iteration can be decomposed into two simpler subproblems whose Hessian are either upper or lower triangle,hence they are much easier to solve.

The test problems are randomly generated such that all the entries in H, q, A and c are i.i.d.Gaussian. For all the test algorithms, the initial values of all working variables are set as zero vectors, and the stopping criterion is either a maximal number of iterations is attained(default setting: 2000) or the relative changes of all working variables are smaller than some prescribed tolerance (default setting: 10^{-10}):

$$\text{relchg(k)} = \max\{\| x_1^k - x_1^{k-1} \| / \| x_1^{k-1} \|, ..., \| x_m^k - x_m^{k-1} \| / \| x_m^{k-1} \|, \| \lambda^k - \lambda^{k-1} \| / \| \lambda^{k-1} \|\} < \text{tol.} \quad (4.4)$$

The setting of parameter β is critical to the performance of the test algorithms, hence its setting is manually tuned to optimize the performance in our experiments, and its optimal setting can be different with different problem settings. For all the test algorithms, the parameters are always chosen to be close to their upper or lower bound: for the scheme (4.3), we set $\beta = 0.2, \alpha = 0.57, \tau = -0.05$ and $s = 1.1$; for BADMMR which needs to apply an extension factor to the λ update, we set it to be 1.6 which is close to its upper bound $((\sqrt{5}+1)/2)$; for GSADMM which requires two extension factors on the two updates of multipliers respectively,we set them to be 0.9 and 1.09 as suggested in [2]; for P3BADMM which needs to apply an extension step with a fixed step size, we set $\alpha = 0.57$, $\sigma = 1.01$.

We compare the test algorithms from three aspects: number of iterations, computation time and accuracy. As we do not know the true solution of the underlying problem (4.1), instead, the KKT violation is adopted as a surrogate of the accuracy measurement. The KKT violation at $k-$th iteration is defined as follows:

$$KKT(k) := \max(KKT_1(k), ..., KKT_m(k), KKT_\lambda(k)), \quad (4.5)$$

where

$$KKT_i(k) = \| H_i x_i^k + q_i - A_i^T \lambda^k \|, \quad i = 1, ..., m, \quad KKT_\lambda(k) = \| A_1 x_1^k + ... + A_m x_m^k - c \| . \quad (4.6)$$

To investigate the convergence behavior of the test algorithms in a more precise way, the iteration progress of KKT violations with several different problem settings are plotted in Figure 2. For instance, when the grouping strategy is $1 \sim 2$, our algorithm is not as good as GSADMM or BADMMR, but it is better than P3BADMM; when the grouping strategy is $2 \sim 1$, our algorithm always outperforms the other algorithms (this is more obvious when $(n, m_i) = (100, 100)$).

4.2 Numerical results on RPCA problem

In this subsection, we consider the robust principal component analysis (RPCA) problem which aims at recovering a low-rank and a sparse matrices from their sum. This problem arises from various areas such as model selection or image processing, see [26, 27]. Specially, the following convex relaxation problem is usually adopted:

$$\begin{aligned} &\min_{A,E} \| A \|_* + \mu \| E \|_1 \\ &s.t. \quad A + E = C, \end{aligned} \quad (4.7)$$

where $C \in \mathcal{R}^{m \times n}$ is the data matrix. The nuclear norm $\| \cdot \|_*$ is a convex surrogate which is to induce the low-rank component of C while the elementwise ℓ_1 norm $\| \cdot \|_1$ can catch the sparse component of C. It has been verified that, under certain mild assumptions, the model (4.7) can recover the original solution accurately.

In real applications, the observation is usually corrupted by additive white Gaussian noise, then the solution of (4.7) may not be accurate, hence the following model was suggested[20]:

$$\begin{aligned} &\min_{A,E} \| A \|_* + \mu \| E \|_1 \\ &s.t. \quad A + E + Z = C, \quad \| Z \|_F \leq \delta, \end{aligned} \quad (4.8)$$

where the setting of parameter δ depends on the noise level and $\| \cdot \|_F$ is the Frobenius norm.

It is easy to observe that model (4.8) is a special case of the general multi-block model (1.1) with 3 block variables, hence we can use our proposed method to solve it with either $1 \sim 2$ or $2 \sim 1$ fashion (denoted by P3BADMM12 and P3BADMM21,respectively). To reveal the efficiency of the proposed method,some efficient

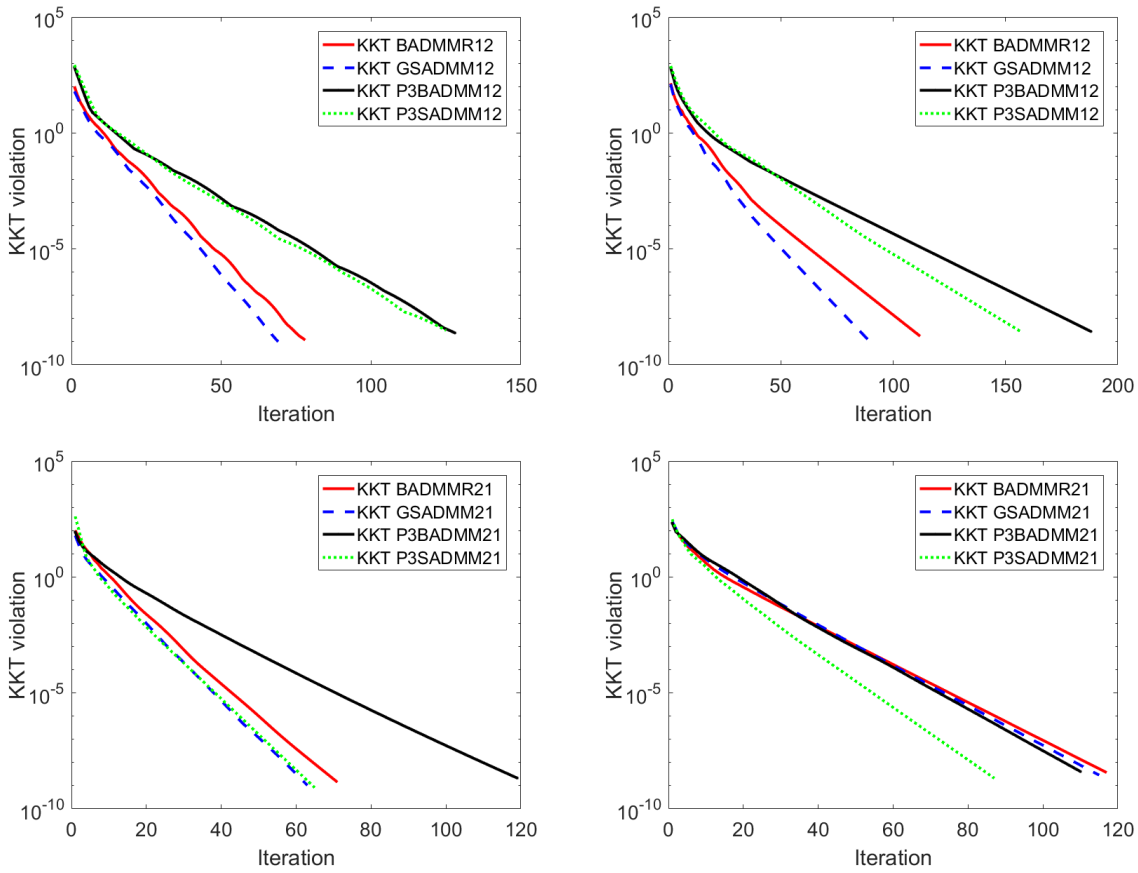

Figure 2: From left to right are the results wit $(n, m_i) = (100, 50), (100, 100)$, respectively. From above to below are the results with $1 \sim 2, 2 \sim 1$,respectively.

algorithms are included in our comparison: the Generalized Symmetric ADMM [2] in $1 \sim 2$ or $2 \sim 1$ fashion (denoted by GSADMM12 and GSADMM21,respectively); the splitting method [7] (denoted by ADMMHTY); the partial splitting augmented Lagrangian method (denoted by PSAL). The subproblems involved in these algorithms (including our algorithm) is solved explicitly: the A-subproblem is solved by partial singular value decomposition (SVD); the E-subproblem is solved by the soft shrinkage operation; and the Z-subproblem is solved by a projection onto a ball. The main cost at each iteration lies in the partial SVD. Although there have been a bunch of efficient solvers for solving partial SVD (e.g.,LMSVD,SLRP,...), however, as reducing the per-iteration cost is not our main concern, we still executed the partial SVD by implementing the package of PROPACK for all test algorithms.

We generate A by $A = UV^T$ where $U \in \mathcal{R}^{m\times k}$ are randomly generated with i.i.d. Gaussian entries. The sparse matrix E is randomly generated whose non-zero entries are i.i.d. uniformly distributed in the interval [-50,50], and SR denotes the ratio of non-zero entries (i.e.,$\| E \|_0 /mn$). The entries of the noise matrix Z are i.i.d. Gaussian with distribution $N(0,1)$, then it is scaled such that $\| Z \|_F = \eta \| A + E \|_F$ where η is the noise level parameter.

We compare the test algorithms from the aspects of computational cost and solution accuracy. To measure the solution accuracy, we use the relative error of A and E with the accurate solutions defined as follows:

$$err(A) := \frac{\| A^k - A^* \|_F}{\| A^* \|_F} \quad and \quad err(E) := \frac{\| E^k - E^* \|_F}{\| E^* \|_F} \tag{4.9}$$

All variables in the test algorithms were initialized with random matrices whose entries are i.i.d. Gaussian with distribution $N(0,1)$. We let all algorithms run for fixed 100 iterations,and record the iteration progress of relative error of the recovered matrices. The settings of algorithm parameters follow that in the previous subsection,and some other necessary parameters are set as follows: $\mu = 0.001, \eta = 0.01, \delta = 0.001$.

The results with two different problem settings are reported in Figure 3. We observed from Figure 3 that, the convergence speed of P3SADMM12 is not the best, but it is able to attain the smallest relative error which means that its output has the best accuracy.

According to Figure 2 and Figure 3, we find that P3SADMM is attractive and promising in the sense that it is able to deal with the multi-block separable convex optimization problem with best accuracy. In terms of the convergence speed, P3SADMM is not best in some conditions.

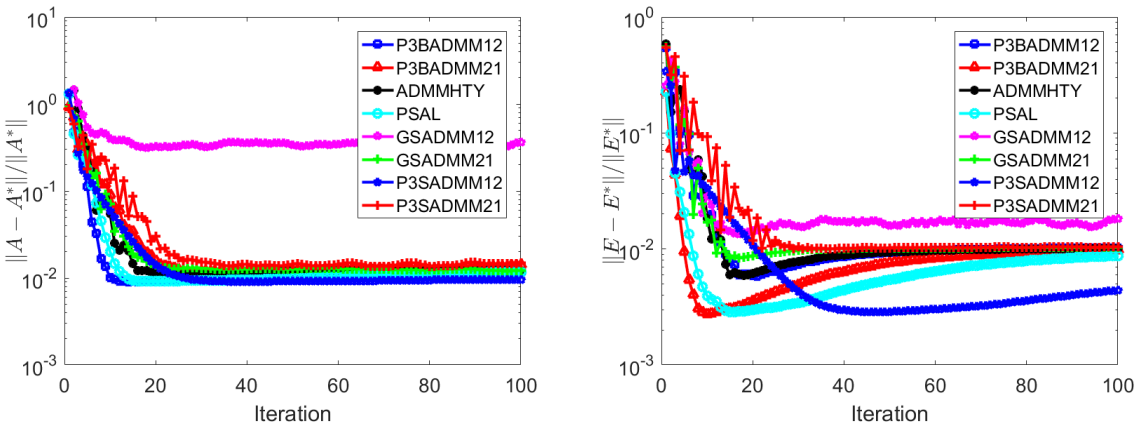

Figure 3: Iteration progress of relative error $(m,n) = (100,100)$.

5 Conclusions

In this paper, based on the S-ADMM proposed by He and et.al and the partial PPa block-wise ADMM proposed by Shen and et.al, a partial PPa S-ADMM for multi-block separable convex optimization is proposed. The novel algorithm could solve the general model (1.1) by taking advantages of the multi-block structure. Based on P3BADMM, the P3SADMM updates the Lagrange multipliers twice with suitable stepsizes,and extends

parameters range. In addition, some numerical results are given, which illustrate that the new method often performs better with proper parameters and could be very promising.

Note that this paper only discusses the partial PPA S-ADMM for multi-block separable convex optimization. In the future, we shall study the PPA S-ADMM with linearization for the multi-block case.

References

[1] Yuan Shen, Xingying Zhang , Xiayang Zhang (2020): A partial PPA block- wise ADMM for multi-block linearly constrained separable convex optimization, Optimization, DOI: 10.1080/02331934.2020.1728756

[2] Bai J , Li J , Xu F , et al. Generalized symmetric ADMM for separable convex optimization[J]. Computational Optimization and Applications, 2017.

[3] J.Bien and R.J.Tibshirani. Sparse estimation of a covariance matrix. Biometrika,98(4):807-820,2011

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

[5] R.Glowinski and A.Marrocco. Sur lapproximation par elements finis dordre un,et la resolution par penalisation-dualite dune classe de problemes de dirichlet nonlineaires. Rev. Francaise dAut. Inf. Rech. Oper., R-2, pages 41-76,1975.

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

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

[8] M.R. Hestenes. Multiplier and gradient methods. J. Optim. Theory Appl., 4:303-320,1969.

[9] Z.S. Liu, J.C. Li, G. Li, J.C. Bai, and X.N. Liu. A new model for sparse and low-rank matrix decomposition. J. Appl. Anal. Comput., 7(2):600-616, 2017.

[10] J.F. Yang and Y. Zhang. Alternating direction algorithms for l-problems in compressive sensing. SIAM J. Sci. Comput., 33(1):250-278, 2011.

[11] Douglas, J, Rachford, HH: On the numerical solution of the heat conduction problem in 2 and 3 space variables. Trans. Am. Math. Soc. 82(82), 421-439 (1956)

[12] Lions, PL, Mercier, B: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. 16(16), 964-979 (1979)

[13] Peaceman, DH, Rachford, HH: The numerical solution of parabolic elliptic differential equations. J. Soc. Ind. Appl. Math. 3(1), 28-41 (1955)

[14] Gabay, D: Applications of the method of multipliers to variational inequalities. In: Fortin, M, Glowinski, R (eds.) Augmented Lagrange Methods: Applications to the Solution of Boundary-Valued Problems, pp. 299-331. North-Holland, Amsterdam (1983)

[15] Glowinski, R, Tallec, PL: Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics. SIAM Studies in Applied Mathematics. SIAM, Philadelphia (1989)

[16] D. W. Peaceman and H. H. Rachford, Jr, The numerical solution of parabolic and elliptic differential equations, Journal of the Society for Industrial and Applied Mathematics, 3 (1955), pp. 28C41.

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

[18] E. Corman and X. Yuan, A generalized proximal point algorithm and its convergence rate, SIAM Journal on Optimization, 24 (2014), pp. 1614-1638.

[19] B. He, H. Liu, Z. Wang, and X. Yuan, A strictly contractive PeacemanCRachford splitting method for convex programming, SIAM Journal on Optimization, 24 (2014), pp. 1011-1040.

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

[21] B.S. He and X.M. Yuan. Block-wise alternating direction method of multipliers for multiple-block convex programming and beyond. SMAI J. Comput. Math., 1:145-174, 2015.

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

[23] B. S. He and X. M. Yuan, Block-wise alternating direction method of multipliers for multiple-block convex programming and beyond, Manucript, August, 2014.

[24] M. Fortin and R. Glowinski, Augmented Lagrangian methods, Studies in Mathematics and its Applications, North-Holland Publishing Co., Amsterdam, 1983.

[25] Gu, Y., Jiang, B., Han, D.: A semi-proximal-based strictly contractive Peaceman-Rachford splitting method.

[26] E. J. Candés, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? J. ACM, 58(3):Article No. 11, 2009.

[27] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In Proceedings of Neural Information Processing Systems (NIPS), December 2009.