On the Direct Extension of ADMM for Multi-block Separable Convex Programming and Beyond: From Variational Inequality Perspective

Bingsheng He[1] and **Xiaoming Yuan**[2]

Revised on April 6, 2014

Abstract. When the alternating direction method of multipliers (ADMM) is extended directly to a multi-block separable convex minimization model whose objective function is in form of more than two functions without coupled variables, it was recently shown that the convergence is not guaranteed. This fact urges to develop efficient algorithms that can preserve completely the numerical advantages of the direct extension of ADMM but with guaranteed convergence. One way to do so is to correct the output of the direct extension of ADMM slightly via a simple correction step (in the sense that it is completely free from step-size computing and its step size is bounded away from zero). In this paper, we propose a prototype algorithm in this framework. This prototype algorithm includes the augmented Lagrangian method (ALM), the original ADMM and the direct extension of ADMM as special cases. A unified and easily checkable condition to ensure the convergence of this prototype algorithm is given. The convergence failure of the direct extension of ADMM can be simply illustrated as that it does not satisfy this condition, while as a comparison, both ALM and ADMM do. The analysis is conducted in the variational inequality context. Based on this prototype algorithm, we propose a class of specific ADMM-based algorithms which are easy to implement. Theoretically, we show the contraction property, prove the global convergence and establish the worst-case convergence rate measured by the iteration complexity for the prototype algorithm. Numerically, we show by an image decomposition problem the efficiency of this class of algorithms. In particular, an important feature of this new class of algorithms is that they could easily outperform the direct extension of ADMM for the cases where the latter is indeed convergent, which is rare among existing methods of the same kind.

Keywords. Convex optimization, Alternating direction method of multipliers, Variational inequalities, Splitting methods, Contraction, Convergence rate.

1 Introduction

Let us start with the convex minimization model with linear constraints:

$$\min\{\theta(x) \mid Ax = b,\, x \in \mathcal{X}\}, \tag{1.1}$$

where $A \in \Re^{m\times n}$, $b \in \Re^m$, $\mathcal{X} \in \Re^n$ a closed convex set, $\theta : \Re^n \to \Re$ a convex but not necessarily smooth function. The solution set of (1.1) is denoted by $\mathcal{X}^*$ and it is assumed to be nonempty. Moreover, the set $\mathcal{X}$ is usually assumed to be simple.

[1]International Centre of Management Science and Engineering, and Department of Mathematics, Nanjing University, Nanjing, 210093, China. This author was supported by the NSFC Grant 91130007 and the MOEC fund 20110091110004. Email: hebma@nju.edu.cn.

[2]Department of Mathematics, Hong Kong Baptist University, Hong Kong, China. This author was supported by the General Research Fund from Hong Kong Research Grants Council: 203613. Email: xmyuan@hkbu.edu.hk.

The augmented Lagrangian method (ALM) in [18, 29] is a benchmark for solving (1.1). Its iterative scheme for (1.1) is

$$\left\{\begin{array}{rcl} x^{k+1} &=& \arg\min\{\theta(x)-(\lambda^k)^T(Ax-b)+\frac{\beta}{2}\|Ax-b\|^2 \,|\, x\in\mathcal{X}\} \\ \lambda^{k+1} &=& \lambda^k-\beta(Ax^{k+1}-b), \end{array}\right. \tag{1.2}$$

where $\lambda\in\Re^m$ is the Lagrange multiplier and $\beta>0$ is a penalty parameter. As analyzed in [31], the scheme (1.2) can be regarded as a dual ascent method over the dual variable λ, where the gradient of the objective function in the dual of (1.1) (i.e., $-(Ax^{k+1}-b)$) is updated recursively by solving a x-minimization problem over the primal variable x.

For many applications, the model (1.1) can be specified as a separable form

$$\min\{\theta_1(x_1)+\theta_2(x_2) \mid A_1x_1+A_2x_2=b,\, x_1\in\mathcal{X}_1, x_2\in\mathcal{X}_2\}, \tag{1.3}$$

where $A_1\in\Re^{m\times n_1}$, $A_2\in\Re^{m\times n_2}$, $b\in\Re^m$, $\mathcal{X}_i\subset\Re^{n_i}$ $(i=1,2)$ and $\theta_i:\Re^{n_i}\to\Re$ $(i=1,2)$. The model (1.3) corresponds to (1.1) with $\theta(x)=\theta_1(x_1)+\theta_2(x_2)$, $x=(x_1,x_2)$, $A=(A_1,A_2)$, $\mathcal{X}=\mathcal{X}_1\times\mathcal{X}_2$ and $n=n_1+n_2$. The matrix A_2 is usually assumed to be full column rank. A typical application of (1.3) is the widely-used l_2-l_1 model

$$\min\{\frac{1}{2}\|Kx-b\|^2+\mu\|x\|_1\},$$

where the least-square term $\frac{1}{2}\|Kx-b\|^2$ represents a data-fidelity term, the l_1-norm term $\|x\|_1$ is a regularization term for inducing sparse solutions, and $\mu>0$ is a trade-off parameter. Using an auxiliary variable y, this model can be reformulated as

$$\min\{\frac{1}{2}\|Kx-b\|^2+\mu\|y\|_1 \mid x-y=0\},$$

which is a special case of (1.3). For solving (1.3), the iterative scheme of ALM (1.2) is specified as

$$\left\{\begin{array}{rcl} (x_1^{k+1},x_2^{k+1}) &=& \arg\min\left\{\begin{array}{r|l} \theta_1(x_1)+\theta_2(x_2)-(\lambda^k)^T(A_1x_1+A_2x_2-b) & x_1\in\mathcal{X}_1 \\ +\frac{\beta}{2}\|A_1x_1+A_2x_2-b\|^2 & x_2\in\mathcal{X}_2 \end{array}\right\}, \\ \lambda^{k+1} &=& \lambda^k-\beta(A_1x^{k+1}+A_2x_2^{k+1}-b). \end{array}\right. \tag{1.4}$$

Although the (x_1,x_2)-subproblem in (1.4) provides an ideal input for updating the dual variable λ, its solvability critically depends on the properties of θ_1 and θ_2. In many cases such as some sparse and low-rank optimization models, we often encounter such a scenario where both θ_1 and θ_2 are simply, e.g., in the sense that the resolvent of $\partial\theta_i$ defined as

$$\big(I+\frac{1}{\beta}\partial\theta_i\big)^{-1}(x_i^0):=\arg\min\big\{\theta_i(x_i)+\frac{\beta}{2}\|x_i-x_i^0\|^2\big\}$$

has a closed-form expression for $\beta>0$, while the resolvent of $\partial\theta_1+\partial\theta_2$ does not. The mentioned l_2-l_1 model is such a case. Then, it is desired to decompose the (x_1,x_2)-subproblem in (1.4) to make use of the functions' properties more effectively. The well-known alternating direction method of multiplier (ADMM) proposed in [11] (see also [4, 9]) is such a method. More specifically, ignoring some constant terms in the sub-minimization problems, the iterative scheme of ADMM for (1.3) reads as

$$\left\{\begin{array}{l} x_1^{k+1}=\arg\min\big\{\theta_1(x_1)-(\lambda^k)^T(A_1x_1)+\frac{\beta}{2}\|A_1x_1+A_2x_2^k-b\|^2\,|\,x_1\in\mathcal{X}_1\big\}, \qquad (1.5a) \\ x_2^{k+1}=\arg\min\big\{\theta_2(x_2)-(\lambda^k)^T(A_2x_2)+\frac{\beta}{2}\|A_1x_1^{k+1}+A_2x_2-b\|^2\,|\,x_2\in\mathcal{X}_2\big\}, \qquad (1.5b) \\ \lambda^{k+1}=\lambda^k-\beta(A_1x_1^{k+1}+A_2x_2^{k+1}-b). \qquad (1.5c) \end{array}\right.$$

Note that the scheme (1.5) represents an inexact version of the ALM scheme (1.4) because the (x_1, x_2)-subproblem in (1.4) is decomposed into two smaller ones, resulting in some losses in accuracy. Meanwhile, because of the significant advantage in exploiting the properties of θ_1 and θ_2 individually, the decomposed subproblems in (1.5) are potentially much easier than the aggregated subproblem in (1.4), meaning some gains in solvability. In fact, the literature has demonstrated well that the gains in solvability outstrip the losses in accuracy, and it has inspired a recent "renaissance" of ADMM in many application domains such as image processing, statistical learning, computer vision, and so on. We refer to [3, 7, 10] for some review papers on ADMM.

Also inspired by numerous applications such as the robust principal component analysis model with noisy and incomplete data in [35], the latent variable Gaussian graphical model selection in [5] and the quadratic discriminant analysis model in [21], we are interested in considering a concrete case of (1.1) with a higher degree of separability:

$$\min\{\theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3) \mid A_1x_1 + A_2x_2 + A_3x_3 = b,\, x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, x_3 \in \mathcal{X}_3\}, \tag{1.6}$$

where $A_i \in \Re^{m\times n_i}$, $\mathcal{X}_i \subset \Re^{n_i}$, $\theta_i : \Re^{n_i} \to \Re$ for $i = 1, 2, 3$; and $b \in \Re^m$. This model corresponds to (1.1) with $\theta(x) = \theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3)$, $x = (x_1, x_2, x_3)$, $A = (A_1, A_2, A_3)$, $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$ and $n = n_1 + n_2 + n_3$. Throughout our discussion, the matrices A_2 and A_3 are assumed to be full column rank[3].

For the same reason as (1.5), in order to solve (1.6) efficiently, we prefer the following decomposed version of the ALM (1.2) instead of using it directly:

$$\begin{cases} x_1^{k+1} = \operatorname{argmin}\{\theta_1(x_1) - (\lambda^k)^T\big(A_1x_1\big) + \dfrac{\beta}{2}\big\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\big\|^2 \,|\, x_1 \in \mathcal{X}_1\}, & \text{(1.7a)}\\ x_2^{k+1} = \operatorname{argmin}\{\theta_2(x_2) - (\lambda^k)^T\big(A_2x_2\big) + \dfrac{\beta}{2}\big\|A_1x_1^{k+1} + A_2x_2 + A_3x_3^k - b\big\|^2 \,|\, x_2 \in \mathcal{X}_2\}, & \text{(1.7b)}\\ x_3^{k+1} = \operatorname{argmin}\{\theta_3(x_3) - (\lambda^k)^T\big(A_3x_3\big) + \dfrac{\beta}{2}\big\|A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3 - b\big\|^2 \,|\, x_3 \in \mathcal{X}_3\}, & \text{(1.7c)}\\ \lambda^{k+1} = \lambda^k - \beta(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - b). & \text{(1.7d)} \end{cases}$$

where $\lambda \in \Re^m$ and $\beta > 0$. The scheme (1.7) is a natural extension of the ADMM scheme (1.5) to the model (1.6), preserving completely the ability of exploiting the functions' properties individually and alleviating the ALM subproblems significantly. In fact, the efficiency of (1.7) has been demonstrated empirically in the literature, see e.g. [28, 35]. The convergence of the direct extension of ADMM (1.7), however, had been ambiguous for long until recently a counter example was given in [6] to show its failure[4]. The difficulty of proving the convergence of (1.7) had inspired some algorithms in [15, 16] whose common feature is to generate a new iterate via correcting the output of (1.7) by some correction steps. Numerically, these algorithms perform slightly slower than (but competitively with) the scheme (1.7), see e.g. [15, 26]; but their convergence can be proved rigorously. In this sense, the scheme (1.7) is substitutable by the methods in [15, 16].

Our motivation of this paper is the following. First, the scheme (1.7) is the most straightforward extension of the ALM (1.2) and ADMM (1.5) which are both convergent; while it is not necessarily convergent as demonstrated in [6]. Can we explain their difference of convergence proof in a unified,

[3]This assumption holds for most of the applications of (1.6) found in the literature, such as those in [5, 21, 35] whose coefficient matrices are actually all identity matrices. Even without this assumption, the convergence analysis to be presented can be modified slightly and be representable in terms of the sequence $\{A_2x_2^k, A_3x_3^k, \lambda^k\}$ instead of $\{x_2^k, x_3^k, \lambda^k\}$.

[4]We refer to [12, 16, 19] for the convergence of (1.7) under some additional assumptions.

comparable and clear way? Our first purpose is to give such an explanation. Second, is it possible to design an algorithm which can preserve the advantage of (1.7) and be even numerically faster than (1.7) while its convergence can be proved rigorously? This is a feature absent in current literature of existing methods of the same kind. Our second purpose is to propose such a class of algorithms. The framework of this class of algorithms is still constituted of the prediction step (1.7) and a correction step at each iteration. In order to outperform the direct extension of ADMM (1.7) numerically for the cases where (1.7) is indeed convergent, a basic requirement is that the correction step must be easy to implement with cheap computation (e.g., free of computing step-size). With the possibility of numerical acceleration, the provable global convergence and the estimate of the worst-case convergence rate measured by the iteration complexity, we regard these algorithms to be proposed as some theoretical and numerical improvement over (1.7) and also those in [15, 16].

Our analysis will be conducted in the context of variational inequality. This is a convenient theoretical tool making us carry out the analysis based on the optimality conditions of the models and enabling us to take advantage of some existing results in the literature. But we would emphasize that the analysis from variational inequality perspective is only for analysis purpose, and we do not require to solve any variational inequality to implement the algorithms to be proposed.

The rest of this paper is organized as follows. In Section 2, we propose a prototype algorithm for solving a variational inequality which serves as the unified model to capture the models (1.1), (1.3) and (1.6); and then a sufficient condition is given to ensure the convergence of this prototype algorithm. We then show that the ALM (1.2) and ADMM (1.5) both satisfy this condition, while the direct extension of ADMM (1.7) does not. In Sections 3 and 4, we prove the convergence of this prototype algorithm and establish its worst-case convergence rate, respectively. We specify the prototype algorithm as a class of algorithms for the model (1.6) in Section 5; and show their efficiency in solving some image processing applications in Section 6. Finally, some conclusions are made and some possible future works are discussed in Section 7.

2 A Prototype Algorithm

In this section, we propose a prototype algorithm for solving a variational inequality which can capture the models (1.1), (1.3) and (1.6) as special cases. The ALM (1.2), ADMM (1.5), direct extension of ADMM (1.7) are all special cases of this prototype algorithm. We thus can study them by a unified framework in the variational inequality context; and find out their difference in convergence proof. More specifically, we will give a unified and easily checkable condition to ensure the convergence of this prototype algorithm; and show that the direct extension of ADMM (1.7) does not satisfy this condition but both ALM and ADMM do.

We consider the following variational inequality in generic setting: Find $w^* \in \mathcal{W}$ such that

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

where $\mathcal{W}$ is a closed convex subset in $\Re^l$; $F : \Re^l \to \Re^l$ is a monotone mapping; $\Re^{l_0}$ ($l_0 \leq l$) is a subspace of $\Re^l$; $u \in \Re^{l_0}$ is the sub-vector of w in $\Re^{l_0}$ for any $w \in \mathcal{W}$; and $\theta : \Re^{l_0} \to \Re$ is a closed convex but not necessarily smooth function. This is called a mixed variational inequality in some literatures, e.g. [13, 30]. Our following analysis is valid for (2.1) in generic setting. But in this paper we are only interested in its special cases (1.1), (1.3) and (1.6). First of all, let us elaborate on how these models can be reformulated as special cases of (2.1).

- The model (1.1). The optimality condition of (1.1) can be written as finding $(x^*, \lambda^*) \in \mathcal{X} \times \Re^m$

such that

$$\theta(x) - \theta(x^*) + \begin{pmatrix} x - x^* \\ \lambda - \lambda^* \end{pmatrix}^T \begin{pmatrix} -A^T\lambda^* \\ Ax^* - b \end{pmatrix} \geq 0, \quad \forall\, (x, \lambda) \in \mathcal{X} \times \Re^m.$$

This corresponds to (2.1) with $l = m + n$, $l_0 = n$,

$$u = x, \quad w = \begin{pmatrix} x \\ \lambda \end{pmatrix}, \quad F(w) = \begin{pmatrix} -A^T\lambda \\ Ax - b \end{pmatrix} \quad \text{and} \quad \mathcal{W} = \mathcal{X} \times \Re^m. \tag{2.2}$$

- The model (1.3). Similarly, by its optimality condition, we can see that solving (1.3) is a special case of (2.1) where $l = m + n_1 + n_2$, $l_0 = n_1 + n_2$,

$$u = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad w = \begin{pmatrix} x_1 \\ x_2 \\ \lambda \end{pmatrix}, \qquad \theta(u) = \theta_1(x_1) + \theta_2(x_2), \tag{2.3a}$$

$$F(w) = \begin{pmatrix} -A_1^T\lambda \\ -A_2^T\lambda \\ A_1x_1 + A_2x_2 - b \end{pmatrix} \qquad \text{and} \qquad \mathcal{W} := \mathcal{X}_1 \times \mathcal{X}_2 \times \Re^m. \tag{2.3b}$$

- The model (1.6). With similar reasoning, we know that solving (1.6) is equivalent to (2.1) with $l = m + n_1 + n_2 + n_3$, $l_0 = n_1 + n_2 + n_3$ and

$$u = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \qquad w = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \lambda \end{pmatrix}, \qquad \theta(u) = \theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3), \tag{2.4a}$$

$$F(w) = \begin{pmatrix} -A_1^T\lambda \\ -A_2^T\lambda \\ -A_3^T\lambda \\ A_1x_1 + A_2x_2 + A_3x_3 - b \end{pmatrix} \qquad \text{and} \qquad \mathcal{W} := \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3 \times \Re^m. \tag{2.4b}$$

Obviously, the mapping $F(w)$ defined respectively in (2.2), (2.3b) and (2.4b) is affine with a skew-symmetric matrix; it is thus monotone. We denote by $\mathcal{W}^*$ the solution set of VI$(\mathcal{W}, F, \theta)$, and it is nonempty when the models (1.1), (1.3) and (1.6) are considered as we have assumed that the solution set of (1.1) is nonempty.

For the convenience of analysis, we make the following definition.

Definition 2.1. For an iterative algorithm solving VI$(\mathcal{W}, F, \theta)$, if some coordinates of w are not involved in the beginning of each iteration, then these coordinates are called intermediate variables and those required by the iteration are called essential variables (denoted by v).

For the ALM (1.2), we see that only the sequence $\{\lambda^k\}$ is required to execute the scheme and $\{x^k\}$ is not. We thus call the variable x an intermediate variable, meaning it being not involved in the iteration. Similarly, for the ADMM (1.5), x_1 is an intermediate variable and $v = (x_2, \lambda)$ are essential variables; for the direct extension of ADMM (1.7), x_1 is an intermediate variable and $v = (x_2, x_3, \lambda)$

are essential variables. Accordingly, the intention of the notations v^k, $\tilde{v}^k$, v^*, $\mathcal{V}$ and $\mathcal{V}^*$ should be clear from the context. When the ALM (1.2) is considered, we have

$$v = \lambda,\ \ \mathcal{V} = \Re^m,\ \ v^k = \lambda^k,\ \ \tilde{v}^k = \tilde{\lambda}^k,\ \ \forall k \in \mathcal{N};\ \ v^* = \lambda^*,\ \ \mathcal{V}^* = \{\lambda^* \,|\, (x^*, \lambda^*) \in \mathcal{W}^*\}.$$

When the ADMM (1.5) is considered, we have

$$v = (x_2, \lambda),\ \ \mathcal{V} = \mathcal{X}_2 \times \Re^m,\ \ v^k = (x_2^k, \lambda^k),\ \ \tilde{v}^k = (\tilde{x}_2^k, \tilde{\lambda}^k),\ \forall k \in \mathcal{N};$$

$$v^* = (x_2^*, \lambda^*),\ \ \mathcal{V}^* = \{(x_2^*, \lambda^*) \,|\, (x_1^*, x_2^*, \lambda^*) \in \mathcal{W}^*\}.$$

When the direct extension of ADMM (1.7) is considered, we have

$$v = (x_2, x_3, \lambda),\ \ \mathcal{V} = \mathcal{X}_2 \times \mathcal{X}_3 \times \Re^m,\ \ v^k = (x_2^k, x_3^k, \lambda^k),\ \ \tilde{v}^k = (\tilde{x}_2^k, \tilde{x}_3^k, \tilde{\lambda}^k),\ \ \forall k \in \mathcal{N};$$

$$v^* = (x_2^*, x_3^*, \lambda^*),\ \ \mathcal{V}^* = \{(x_2^*, x_3^*, \lambda^*) \,|\, (x_1^*, x_2^*, x_3^*, \lambda^*) \in \mathcal{W}^*\}.$$

2.1 A Prototype Algorithm for (2.1)

Now, we propose a prototype algorithm for VI$(\mathcal{W}, F, \theta)$ and give a condition ensuring its convergence. Recall we denote by v the essential variables of an algorithm; meaning its iteration essentially generating v^{k+1} with the given v^k. Then, a prototype algorithm for solving (2.1) can be summarized as follows.

A Prototype Algorithm for (2.1)

[Step 1.] With given v^k, find a vector $\tilde{w}^k \in \mathcal{W}$ and a matrix Q satisfying

$$\tilde{w}^k \in \Omega, \quad \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (v - \tilde{v}^k)^T Q(v^k - \tilde{v}^k), \quad \forall\, w \in \mathcal{W}, \tag{2.5a}$$

where the matrix Q has the property: $Q^T + Q$ is positive definite.

[Step 2.] Determine a nonsingular matrix M and a positive scalar α; and generate the new iterate v^{k+1} by

$$v^{k+1} = v^k - \alpha M(v^k - \tilde{v}^k). \tag{2.5b}$$

Then, the convergence of the prototype algorithm (2.5) can be guaranteed if the following condition is fulfilled.

Convergence Condition

For the matrices Q and M, and the step size α determined in (2.5), we define two matrices H and G respectively as

$$H = QM^{-1} \tag{2.6a}$$

and

$$G = Q^T + Q - \alpha M^T H M. \tag{2.6b}$$

Then, the prototype algorithm (2.5) is convergent if "H is symmetric and positive definite" and "G is positive semi-definite".

Remark 2.2. If $v^k = \tilde{v}^k$, then $\tilde{w}^k$ is a solution point of (2.1) because of (2.5a) and the definition of VI$(\mathcal{W}, F, \theta)$. We thus can use $\|v^k - \tilde{v}^k\| < \epsilon$ as a stopping criterion to implement a specific algorithm of the prototype (2.5), where ϵ is a given tolerance. For example, this explains the primal-dual stopping criterion used in [3] for the ADMM scheme (1.5), which will be shown as a special case of (2.5) in Section 2.3. On the other hand, if $v^k \neq \tilde{v}^k$, then setting $w = w^*$ in (2.5a) and using

$$\theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T F(\tilde{w}^k) = \theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T F(w^*) \geq 0,$$

we get

$$(\tilde{v}^k - v^*)^T Q(v^k - \tilde{v}^k) \geq 0, \quad \forall v^* \in \mathcal{V}^*.$$

By using $Q = HM$ and the above inequality, we obtain

$$(v^k - v^*)^T HM(v^k - \tilde{v}^k) \geq (v^k - \tilde{v}^k)^T \Big[\frac{Q^T + Q}{2}\Big](v^k - \tilde{v}^k), \quad \forall\, v^* \in \mathcal{V}^*. \tag{2.7}$$

In some sense, the inequality (2.7) implies that the direction $-M(v^k - \tilde{v}^k)$ is beneficial for reducing the proximity to the solution set $\mathcal{V}^*$ in H-norm. Hence, the correction step (2.5b) can also be explained as a contraction step which moves along the direction $-M(v^k - \tilde{v}^k)$ starting from v^k towards $\mathcal{V}^*$.

The convergence of the prototype algorithm (2.5) under the Convergence Condition will be proved rigorously in Section 3. Now, in order to know whether a specific algorithm arising from the prototype (2.5) is convergent or not, one way is to just check if the matrix H defined in (2.6a) is symmetric and positive definite, and if the matrix G defined in (2.6b) is positive semi-definite. Below we start to show that the ALM (1.2), ADMM (1.5) and the direct extension of ADMM (1.7) are all special cases of the prototype algorithm (2.5). But the convergence condition is satisfied by both (1.2) and (1.5), while not by (1.7). The convergence failure of the direct extension of ADMM (1.7) is thus illustrated.

2.2 ALM Satisfies the Convergence Condition

In this subsection, we show that the ALM scheme (1.2) is a special case of the prototype algorithm (2.5) and the Convergence Condition is satisfied. Recall the model (1.1) can be explained as the VI (2.1) with the specification given in (2.2).

In order to cast the ALM scheme (1.2) into a special case of (2.5), let us first define the vector $\tilde{w}^k = (\tilde{x}^k, \tilde{\lambda}^k)$ by

$$\tilde{x}^k = x^{k+1} \qquad \text{and} \quad \tilde{\lambda}^k = \lambda^k - \beta(Ax^{k+1} - b),$$

for given λ^k and the x^{k+1} generated by (1.2). Obviously, by the optimality condition, the iterate generated by (1.2) satisfies the following VI:

$$\theta(x) - \theta(\tilde{x}^k) + \begin{pmatrix} x - \tilde{x}^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \left\{ \begin{pmatrix} -A^T\tilde{\lambda}^k \\ A\tilde{x}^k - b \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \end{pmatrix} \right\} \geq 0, \quad \forall\, w \in \mathcal{W},$$

which is a special case of (2.5a) with $Q = \frac{1}{\beta} I_{m\times m}$. Recall the essential variable is λ for ALM. Moreover, we have

$$\lambda^{k+1} = \tilde{\lambda}^k = \lambda^k - (\lambda^k - \tilde{\lambda}^k),$$

which corresponds to (2.5b) with

$$M = I_{m\times m} \qquad \text{and} \qquad \alpha = 1.$$

Now, we show that the Convergence Condition is satisfied by the ALM scheme (1.2). For the matrices $Q = \frac{1}{\beta} I_{m\times m}$ and $M = I_{m\times m}$, and $\alpha = 1$ specified above, we have

$$H = QM^{-1} = \frac{1}{\beta} I_{m\times m},$$

and

$$G = Q^T + Q - \alpha M^T H M = Q^T + Q - M^T Q = \frac{1}{\beta} I_{m\times m}.$$

So, H and G are both symmetric and positive definite; and the Convergence Condition is satisfied. The convergence of ALM (1.2) is thus guaranteed.

2.3 ADMM Satisfies the Convergence Condition

This subsection shows that the ADMM scheme (1.5) is also a special case of the prototype algorithm (2.5) and the Convergence Condition is satisfied. Recall the model (1.3) can be explained as the VI (2.1) with the specification given in (2.8).

In order to cast the ADMM scheme (1.5) into a special case of (2.5), let us first define the vector $\tilde{w}^k = (\tilde{x}_1^k, \tilde{x}_2^k, \tilde{\lambda}^k)$ by

$$\tilde{x}_1^k = x_1^{k+1}, \qquad \tilde{x}_2^k = x_2^{k+1} \qquad \text{and} \qquad \tilde{\lambda}^k = \lambda^k - \beta(A_1 x_1^{k+1} + A_2 x_2^k - b), \tag{2.8}$$

for the given (x_2^k, λ^k) and (x_1^{k+1}, x_2^{k+1}) generated by the ADMM (1.5). Then, the iterate generated by the scheme (1.5) satisfies the following VI:

$$\theta(u) - \theta(\tilde{u}^k) + \begin{pmatrix} x_1 - \tilde{x}_1^k \\ x_2 - \tilde{x}_2^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \left\{ \begin{pmatrix} -A_1^T\tilde{\lambda}^k \\ -A_2^T\tilde{\lambda}^k \\ A_1\tilde{x}_1^k + A_2\tilde{x}_2^k - b \end{pmatrix} + \begin{pmatrix} 0 \\ \beta A_2^T A_2(\tilde{x}_2^k - x_2^k) \\ -A_2(\tilde{x}_2^k - x_2^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \end{pmatrix} \right\} \geq 0, \ \forall w \in \mathcal{W}.$$

which is a special case of (2.5a) with

$$Q = \begin{pmatrix} \beta A_2^T A_2 & 0 \\ -A_2 & \frac{1}{\beta} I \end{pmatrix}. \tag{2.9}$$

For the matrix Q defined in (2.9), we have

$$Q^T + Q = \begin{pmatrix} 2\beta A_2^T A_2 & -A_2^T \\ -A_2 & \frac{2}{\beta} I \end{pmatrix}.$$

which is positive definite under the assumption that A_2 be full column rank.

Recall the essential variable of the ADMM scheme (1.5) is (x_2, λ). Moreover, using the definition of $\tilde{w}^k$, the λ^{k+1} updated by (1.5c) can be represented as

$$\begin{aligned} \lambda^{k+1} &= \lambda^k - \beta(A_1\tilde{x}_1^k + A_2\tilde{x}_2^k - b) \\ &= \lambda^k - \big[(\lambda^k - \tilde{\lambda}^k) - \beta A_2(x_2^k - \tilde{x}_2^k)\big]. \end{aligned}$$

Therefore, the ADMM scheme (1.5) can be written as

$$\begin{pmatrix} x_2^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x_2^k \\ \lambda^k \end{pmatrix} - \begin{pmatrix} I & 0 \\ -\beta A_2 & I \end{pmatrix} \begin{pmatrix} x_2^k - \tilde{x}_2^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}.$$

which corresponds to the step (2.5b) with

$$M = \begin{pmatrix} I & 0 \\ -\beta A_2 & I \end{pmatrix} \qquad \text{and} \qquad \alpha = 1. \tag{2.10}$$

Now we check that the Convergence Condition is satisfied by the ADMM scheme (1.5). Indeed, for the matrix M in (2.10), we have

$$M^{-1} = \begin{pmatrix} I & 0 \\ \beta A_2 & I \end{pmatrix}.$$

Thus, by using (2.9) and (2.10), we obtain

$$H = QM^{-1} = \begin{pmatrix} \beta A_2^T A_2 & 0 \\ -A_2 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & 0 \\ \beta A_2 & I \end{pmatrix} = \begin{pmatrix} \beta A_2^T A_2 & 0 \\ 0 & \frac{1}{\beta} I \end{pmatrix},$$

and consequently

$$\begin{aligned} G &= Q^T + Q - \alpha M^T H M = Q^T + Q - Q^T M \\ &= \begin{pmatrix} 2\beta A_2^T A_2 & -A_2^T \\ -A_2 & \frac{2}{\beta} I \end{pmatrix} - \begin{pmatrix} \beta A_2^T A_2 & -A_2^T \\ 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & 0 \\ -\beta A_2 & I \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ 0 & \frac{1}{\beta} I \end{pmatrix}. \end{aligned}$$

Therefore, H is symmetric and positive definite under the assumption that A_2 be full column rank; and G is positive semi-definite. The Convergence Condition is satisfied; and thus the convergence of the ADMM scheme (1.5) is guaranteed.

2.4 The Direct Extension of ADMM (1.7) Does Not Satisfy the Convergence Condition

Now we show that the direct extension of ADMM (1.7) is also a special case of the prototype algorithm (2.5) but the Convergence Condition is not satisfied. Thus an explanation of the convergence failure is provided for (1.7). Recall the model (1.6) can be explained as the VI (2.1) with the specification given in (2.4).

First, the optimality condition of the iteration (1.7) can be summarized as the following VIs:

$$\begin{cases} x_1^{k+1} \in \mathcal{X}_1, & \theta_1(x_1) - \theta_1(x_1^{k+1}) + (x_1 - x_1^{k+1})^T \\ & \{-A_1^T[\lambda^k - \beta(A_1 x_1^{k+1} + A_2 x_2^k + A_3 x_3^k - b)]\} \geq 0, \quad \forall x_1 \in \mathcal{X}_1, \end{cases} \tag{2.11a}$$

$$\begin{cases} x_2^{k+1} \in \mathcal{X}_2, & \theta_2(x_2) - \theta_2(x_2^{k+1}) + (x_2 - x_2^{k+1})^T \\ & \{-A_2^T[\lambda^k - \beta(A_1 x_1^{k+1} + A_2 x_2^{k+1} + A_3 x_3^k - b)]\} \geq 0, \quad \forall x_2 \in \mathcal{X}_2, \end{cases} \tag{2.11b}$$

$$\begin{cases} x_3^{k+1} \in \mathcal{X}_3, & \theta_3(x_3) - \theta_3(x_3^{k+1}) + (x_3 - x_3^{k+1})^T \\ & \{-A_3^T[\lambda^k - \beta(A_1 x_1^{k+1} + A_2 x_2^{k+1} + A_3 x_3^{k+1} - b)]\} \geq 0, \ \forall x_3 \in \mathcal{X}_3. \end{cases} \tag{2.11c}$$

In order to see the scheme (2.11) can be written as a special case of (2.5a), we define $\tilde{w}^k = (\tilde{x}_1^k, \tilde{x}_2^k, \tilde{x}_3^k, \tilde{\lambda}^k)$ by

$$\tilde{x}_1^k = x_1^{k+1}, \qquad \tilde{x}_2^k = x_2^{k+1}, \qquad \tilde{x}_3^k = x_3^{k+1}, \tag{2.12a}$$

and

$$\tilde{\lambda}^k = \lambda^k - \beta(A_1 x_1^{k+1} + A_2 x_2^k + A_3 x_3^k - b), \tag{2.12b}$$

for the given $(x_2^k, x_3^k, \lambda^k)$ and the $(x_1^{k+1}, x_2^{k+1}, x_3^{k+1})$ generated by the direct extension of ADMM (1.7). By using (2.12), the scheme (2.11) can be simplified as $(\tilde{x}_1^k, \tilde{x}_2^k, \tilde{x}_3^k) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$,

$$\begin{cases} \theta_1(x_1) - \theta_1(\tilde{x}_1^k) + (x_1 - \tilde{x}_1^k)^T\{-A_1^T\tilde{\lambda}^k\} \geq 0, \quad \forall x_1 \in \mathcal{X}_1, & \text{(2.13a)} \\ \theta_2(x_2) - \theta_2(\tilde{x}_2^k) + (x_2 - \tilde{x}_2^k)^T\{-A_2^T\tilde{\lambda}^k + \beta A_2^T A_2(\tilde{x}_2^k - x_2^k)]\} \geq 0, \quad \forall x_2 \in \mathcal{X}_2, & \text{(2.13b)} \\ \theta_3(x_3) - \theta_3(\tilde{x}_3^k) + (x_3 - \tilde{x}_3^k)^T\{-A_3^T\tilde{\lambda}^k + \beta A_3^T[A_2(\tilde{x}_2^k - x_2^k) + A_3(\tilde{x}_3^k - x_3^k)]\} \geq 0, \ \forall x_3 \in \mathcal{X}_3. & \text{(2.13c)} \end{cases}$$

We now show in the next lemma that the (2.13) is a special case of the prototype algorithm (2.5a). Recall the essential variable of the scheme (1.7) is $v = (x_2, x_3, \lambda)$.

Lemma 2.3. *Let $u^{k+1} = (x_1^{k+1}, x_2^{k+1}, x_3^{k+1})$ be generated by the direct extension of ADMM (1.7) from the given $v^k = (x_2^k, x_3^k, \lambda^k)$; and $\tilde{w}^k$ be defined in (2.12). Then we have*

$$\tilde{w}^k \in \mathcal{W}, \quad \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (v - \tilde{v}^k)^T Q(v^k - \tilde{v}^k), \quad \forall\, w \in \mathcal{W}, \tag{2.14}$$

where

$$Q = \begin{pmatrix} \beta A_2^T A_2 & 0 & 0 \\ \beta A_3^T A_2 & \beta A_3^T A_3 & 0 \\ -A_2 & -A_3 & \frac{1}{\beta} I \end{pmatrix}. \tag{2.15}$$

Proof. Using $\tilde{x}_j^k = x_j^{k+1},\ j = 1, 2, 3$, and the equation (2.12b), we have

$$(A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3\tilde{x}_3^k - b) - A_2(\tilde{x}_2^k - x_2^k) - A_3(\tilde{x}_3^k - x_3^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) = 0,$$

which can be rewritten as

$$\tilde{\lambda}^k \in \Re^m, \ \ (\lambda - \tilde{\lambda}^k)^T\{(A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3\tilde{x}_3^k - b) - A_2(\tilde{x}_2^k - x_2^k) - A_3(\tilde{x}_3^k - x_3^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \ \ \forall \lambda \in \Re^m.$$

Combining (2.13) and the last inequality together, we get

$$\begin{aligned} \tilde{w}^k \in \Omega, \quad \theta(u) - \theta(\tilde{u}^k) + \begin{pmatrix} x_1 - \tilde{x}_1^k \\ x_2 - \tilde{x}_2^k \\ x_3 - \tilde{x}_3^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \Bigg\{ \begin{pmatrix} -A_1^T\tilde{\lambda}^k \\ -A_2^T\tilde{\lambda}^k \\ -A_3^T\tilde{\lambda}^k \\ A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3\tilde{x}_3^k - b \end{pmatrix} \\ + \begin{pmatrix} 0 \\ \beta A_2^T A_2(\tilde{x}_2^k - x_2^k) \\ \beta A_3^T A_2(\tilde{x}_2^k - x_2^k) + \beta A_3^T A_3(\tilde{x}_3^k - x_3^k) \\ -A_2(\tilde{x}_2^k - x_2^k) - A_3(\tilde{x}_3^k - x_3^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \end{pmatrix} \Bigg\} \geq 0, \qquad \forall\, w \in \mathcal{W}. \end{aligned} \tag{2.16}$$

Based on the above inequality and using the notation $F(w)$ (see (2.4)) and matrix Q (see (2.15)), the assertion of this lemma is proved. □

With the definition Q in (2.15), we have

$$\begin{aligned} Q^T + Q &= \begin{pmatrix} \beta A_2^T A_2 & \beta A_2^T A_3 & -A_2^T \\ \beta A_3^T A_2 & \beta A_3^T A_3 & -A_3^T \\ -A_2 & -A_3 & \frac{1}{\beta} I \end{pmatrix} + \begin{pmatrix} \beta A_2^T A_2 & 0 & 0 \\ 0 & \beta A_3^T A_3 & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \\ &\succeq \begin{pmatrix} \beta A_2^T A_2 & 0 & 0 \\ 0 & \beta A_3^T A_3 & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \end{aligned}$$

which means Q is positive definite whenever A_2 and A_3 are both full column rank.

Moreover, using (2.12), the λ^{k+1} updated by (1.7d) can be represented as

$$\begin{aligned} \lambda^{k+1} &= \lambda^k - \beta(A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3\tilde{x}_3^k - b) \\ &= \lambda^k - \big[(\lambda^k - \tilde{\lambda}^k) - \beta A_2(x_2^k - \tilde{x}_2^k) - \beta A_3(x_3^k - \tilde{x}_3^k)\big]. \end{aligned} \tag{2.17}$$

Therefore, by using (2.12), the direct extension of ADMM (1.7) can be written as

$$\begin{pmatrix} x_2^{k+1} \\ x_3^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x_2^k \\ x_3^k \\ \lambda^k \end{pmatrix} - \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\beta A_2 & -\beta A_3 & I \end{pmatrix} \begin{pmatrix} x_2^k - \tilde{x}_2^k \\ x_3^k - \tilde{x}_3^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}$$

which corresponds to the step (2.5b) with

$$M = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\beta A_2 & -\beta A_3 & I \end{pmatrix} \qquad \text{and} \qquad \alpha = 1. \tag{2.18}$$

Indeed, the direct extension of ADMM (1.7) can be written in form of the prototype algorithm (2.5).

Now we demonstrate that the Convergence Condition is not satisfied by the direct extension of ADMM (1.7). In fact, for the matrix M in (2.18), we have

$$M^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \beta A_2 & \beta A_3 & I \end{pmatrix}.$$

and thus

$$H = QM^{-1} = \begin{pmatrix} \beta A_2^T A_2 & 0 & 0 \\ \beta A_3^T A_2 & \beta A_3^T A_3 & 0 \\ -A_2 & -A_3 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \beta A_2 & \beta A_3 & I \end{pmatrix} = \begin{pmatrix} \beta A_2^T A_2 & 0 & 0 \\ \beta A_3^T A_2 & \beta A_3^T A_3 & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}$$

which is not symmetric. In addition, we have

$$\begin{aligned} G &= Q^T + Q - Q^T M \\ &= Q^T + Q - \begin{pmatrix} \beta A_2^T A_2 & \beta A_2^T A_3 & -A_2^T \\ 0 & \beta A_3^T A_3 & -A_3^T \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\beta A_2 & -\beta A_3 & I \end{pmatrix} \\ &= \begin{pmatrix} 2\beta A_2^T A_2 & \beta A_2^T A_3 & -A_2^T \\ \beta A_3^T A_2 & 2\beta A_3^T A_3 & -A_3^T \\ -A_2 & -A_3 & \frac{2}{\beta} I \end{pmatrix} - \begin{pmatrix} 2\beta A_2^T A_2 & 2\beta A_2^T A_3 & -A_2^T \\ \beta A_3^T A_2 & 2\beta A_3^T A_3 & -A_3^T \\ -A_2 & -A_3 & \frac{1}{\beta} I \end{pmatrix} \\ &= \begin{pmatrix} 0 & -\beta A_2^T A_3 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}, \end{aligned}$$

which is not positive semi-definite. Therefore, the Convergence Condition is not satisfied by the direct extension of ADMM (1.7). This may explain its difficulty in convergence proof.

2.5 Some Remarks

In our analysis, because of the purpose of analyzing splitting methods, we treat the models (1.3) and (1.6) as special cases of the generic model (1.1) where the objective function in (1.1) is specified as more than one function and the variable is partitioned accordingly. Alternatively, we can treat the model (1.6) as a generic model while the model (1.1) is a special case with one block of function and variable; and (1.3) is a special case with two blocks of function and variable. Accordingly, the matrix Q and M in the context of the prototype algorithm (2.5) for ALM (1.2) and ADMM (1.5) can be obtained straightforwardly from Q and M in (2.15) and (2.18) by removing the first two rows and columns (accounting for "no x_2 and x_3 in (1.2)") and removing the second row and column (accounting for "no x_3 in (1.3)"), respectively. Because of this very consistent relationship in the VI framework (2.5), we regard the specification of (2.5) to the model (1.6) as a "natural" extension of the ALM and ADMM for (1.1) and (1.3), respectively.

3 Convergence

In this section, we show the contraction property for the sequence generated by the prototype algorithm (2.5) and then prove its global convergence. Recall the iteration of (2.5) only updates the essential variable v. We thus show the contraction property only for the sequence of essential variable $\{v^k\}$.

We first prove a lemma. In the following, we slightly abuse the notation $\|v\|_G^2 := v^T G v$ even though G might only be positive semi-definite.

Lemma 3.1. *For the sequence generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied. We have*

$$\alpha\big\{\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k)\big\} \geq \frac{1}{2}\big(\|v - v^{k+1}\|_H^2 - \|v - v^k\|_H^2\big) + \frac{\alpha}{2}\|v^k - \tilde{v}^k\|_G^2, \ \forall w \in \mathcal{W}, \quad (3.1)$$

where H and G are defined in (2.6).

Proof. Using $Q = HM$ (see (2.6a)) and the update form (2.5b), the right-hand side of (2.5a) can be written as

$$\tilde{w}^k \in \mathcal{W},\ \ \alpha\big\{\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k)\big\} \geq (v - \tilde{v}^k)^T H(v^k - v^{k+1}),\ \ \forall w \in \mathcal{W}. \tag{3.2}$$

Applying the identity

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

to the right-hand side of (3.2) with

$$a = v, \quad b = \tilde{v}^k, \quad c = v^k, \quad \text{and} \quad d = v^{k+1},$$

we thus obtain

$$(v - \tilde{v}^k)^T H(v^k - v^{k+1}) = \frac{1}{2}\big(\|v - v^{k+1}\|_H^2 - \|v - v^k\|_H^2\big) + \frac{1}{2}(\|v^k - \tilde{v}^k\|_H^2 - \|v^{k+1} - \tilde{v}^k\|_H^2). \tag{3.3}$$

For the last term of (3.3), we have

$$\begin{aligned}
&\|v^k - \tilde{v}^k\|_H^2 - \|v^{k+1} - \tilde{v}^k\|_H^2 \\
&\quad = \quad \|v^k - \tilde{v}^k\|_H^2 - \|(v^k - \tilde{v}^k) - (v^k - v^{k+1})\|_H^2 \\
&\quad \overset{(2.5b)}{=} \quad \|v^k - \tilde{v}^k\|_H^2 - \|(v^k - \tilde{v}^k) - \alpha M(v^k - \tilde{v}^k)\|_H^2 \\
&\quad = \quad 2\alpha(v^k - \tilde{v}^k)^T HM(v^k - \tilde{v}^k) - \alpha^2 (v^k - \tilde{v}^k)^T M^T HM(v^k - \tilde{v}^k) \\
&\quad \overset{(2.6a)}{=} \quad \alpha(v^k - \tilde{v}^k)^T (Q^T + Q - \alpha M^T HM)(v^k - \tilde{v}^k) \\
&\quad \overset{(2.6b)}{=} \quad \alpha\|v^k - \tilde{v}^k\|_G^2.
\end{aligned} \tag{3.4}$$

Substituting (3.3) and (3.4) in (3.2), the assertion of this theorem is proved. □

Based on Lemma 3.1, we can prove the contraction property for the sequence $\{v^k\}$ generated by the prototype algorithm (2.5).

Theorem 3.2. *For the sequence $\{v^k\}$ generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied, we have*

$$\|v^{k+1} - v^*\|_H^2 \leq \|v^k - v^*\|_H^2 - \alpha\|v^k - \tilde{v}^k\|_G^2, \quad \forall v^* \in \mathcal{V}^*. \tag{3.5}$$

Proof. Setting $w = w^*$ in (3.1), we get

$$\|v^k - v^*\|_H^2 - \|v^{k+1} - v^*\|_H^2 \geq \alpha\|v^k - \tilde{v}^k\|_G^2 + 2\alpha\big\{\theta(\tilde{u}^k) - \theta(u^*) + (\tilde{w}^k - w^*)^T F(\tilde{w}^k)\big\}. \tag{3.6}$$

By using the optimality of w^* and the monotonicity of $F(w)$, we have

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

The assertion (3.5) follows from (3.6) and the above fact directly. □

Recall the matrix G defined in (2.6b) is assumed to be positive semi-definite. It follows from (3.5) that the sequence $\{v^k\}$ is contractive with respect to $\mathcal{V}^*$. The convergence of $\{v^k\}$ generated by the prototype algorithm (2.5) thus follows the standard framework of contraction methods, see e.g. [2]. In fact, in Section 4.1, its convergence is also implied. In the following, for completeness, we still include the detail for the case $G \succ 0$.

Theorem 3.3. *For the sequence $\{v^k\}$ generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied with $G \succ 0$, it converges to some v^∞ which belongs to $\mathcal{V}^*$.*

Proof. According to (3.5), it holds that $\{v^k\}$ is bounded and

$$\lim_{k\to\infty} \|v^k - \tilde{v}^k\|_G = 0. \tag{3.7}$$

So, $\{\tilde{v}^k\}$ is also bounded. Let v^∞ be a cluster point of $\{\tilde{v}^k\}$ and $\{\tilde{v}^{k_j}\}$ is a subsequence which converges to v^∞. Let $\{\tilde{w}^k\}$ and $\{\tilde{w}^{k_j}\}$ be the induced sequences by $\{\tilde{v}^k\}$ and $\{\tilde{v}^{k_j}\}$, respectively. It follows from (2.5a) that

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

Since the matrix Q is not singular, it follows from the continuity of $\theta(u)$ and $F(w)$ that

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

The above variational inequality indicates that w^∞ is a solution of $\mathrm{VI}(\Omega, F)$. By using (3.7) and $\lim_{j\to\infty} v^{k_j} = v^\infty$, the subsequence $\{v^{k_j}\}$ converges to v^∞. Due to (3.5), we have

$$\|v^{k+1} - v^\infty\|_H \leq \|v^k - v^\infty\|_H$$

and thus $\{v^k\}$ converges to v^∞. The proof is complete. □

4 Worst-case Convergence Rate

In this section, we estimate the worst-case $O(1/t)$ convergence rate measured by the iteration complexity for the prototype algorithm (2.5), where t is the iteration counter[5]. Lemma 3.1 is again crucial for the analysis.

4.1 Convergence Rate in the Ergodic Sense

We first show that a worst-case $O(1/t)$ convergence rate in the ergodic sense can be established for the prototype algorithm (2.5) under the Convergence Condition.

Working on the reformulation $\mathrm{VI}(\Omega, F, \theta)$ of the models (1.1), (1.3) and (1.6), one advantage is that a characterization of its solution set proposed in [8] becomes useful for establishing the worst-case convergence rate in the ergodic sense for the prototype algorithm (2.5).

Theorem 4.1. *The solution set of VI(Ω, F, θ) is convex and it can be characterized as*

$$\mathcal{W}^* = \bigcap_{w\in\mathcal{W}} \big\{\tilde{w} \in \mathcal{W} : \big(\theta(u) - \theta(\tilde{u})\big) + (w - \tilde{w})^T F(w) \geq 0\big\}. \tag{4.1}$$

Proof. The proof is an incremental extension of Theorem 2.3.5 in [8], or see the the proof of Theorem 2.1 in [17]. □

[5]We follow the work [23, 24] and many others to measure the worst-case convergence rate in term of the iteration complexity. That is, a worst-case $\mathcal{O}(1/t)$ convergence rate means the accuracy to a solution under certain criteria is of the order $\mathcal{O}(1/t)$ after t iterations of an iterative scheme; or equivalently, it requires at most $\mathcal{O}(1/\epsilon)$ iterations to achieve an approximate solution with an accuracy of ϵ.

Theorem 4.1 thus implies that $\tilde{w} \in \mathcal{W}$ is an approximate solution of $\mathrm{VI}(\mathcal{W}, F, \theta)$ with the accuracy $\epsilon > 0$ if it satisfies

$$\theta(u) - \theta(\tilde{u}) + F(w)^T(w - \tilde{w}) \geq -\epsilon, \ \ \forall w \in \mathcal{W} \cap \mathcal{D}(\tilde{u}),$$

where

$$\mathcal{D}(\tilde{u}) = \{u \,|\, \|u - \tilde{u}\| \leq 1\}.$$

In this paper, we will show that after t iterations of the proposed methods, we can find $\tilde{w} \in \mathcal{W}$ such that

$$\theta(\tilde{u}) - \theta(u) + (\tilde{w} - w)^T F(w) \leq \epsilon, \ \ \forall w \in \mathcal{W} \cap \mathcal{D}(\tilde{u}), \tag{4.2}$$

with $\epsilon = O(1/t)$, the convergence rate $O(1/t)$ of the proposed methods is thus proved.

Before the proof, we notice that the monotonicity of F implies that

$$(w - \tilde{w}^k)^T F(w) \geq (w - \tilde{w}^k)^T F(\tilde{w}^k).$$

Notice that under the condition that the matrix G defined in (2.6b) is positive semi-definite, by substituting it in (3.1), we obtain

$$\tilde{w}^k \in \Omega, \ \ \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(w) + \frac{1}{2\alpha}\|v - v^k\|_H^2 \geq \frac{1}{2\alpha}\|v - v^{k+1}\|_H^2, \ \forall w \in \mathcal{W}. \tag{4.3}$$

Theorem 4.2. *For the sequence generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied and any integer $t > 0$, we have*

$$\theta(\tilde{u}_t) - \theta(u) + (\tilde{w}_t - w)^T F(w) \leq \frac{1}{2(t+1)\alpha}\|v - v^0\|_H^2, \quad \forall w \in \mathcal{W}, \tag{4.4}$$

where

$$\tilde{w}_t = \frac{1}{t+1}\sum_{k=0}^{t} \tilde{w}^k. \tag{4.5}$$

Proof. First, because of (4.3), it holds that $\tilde{w}^k \in \Omega$ for all $k \geq 0$. Together with the convexity of $\mathcal{X}$ and $\mathcal{Y}$, (4.5) implies that $\tilde{w}_t \in \Omega$. Summing the inequality (4.3) over $k = 0, 1, \ldots, t$, we obtain

$$(t+1)\theta(u) - \sum_{k=0}^{t} \theta(\tilde{u}^k) + \Big((t+1)w - \sum_{k=0}^{t} \tilde{w}^k\Big)^T F(w) + \frac{1}{2\alpha}\|v - v^0\|_H^2 \geq 0, \quad \forall w \in \mathcal{W}.$$

Use the notation of $\tilde{w}_t$, it can be written as

$$\frac{1}{t+1}\sum_{k=0}^{t} \theta(\tilde{u}^k) - \theta(u) + (\tilde{w}_t - w)^T F(w) \leq \frac{1}{2(t+1)\alpha}\|v - v^0\|_H^2, \quad \forall w \in \mathcal{W}. \tag{4.6}$$

Since $\theta(u)$ is convex and

$$\tilde{u}_t = \frac{1}{t+1}\sum_{k=0}^{t} \tilde{u}^k,$$

we have that

$$\theta(\tilde{u}_t) \leq \frac{1}{t+1}\sum_{k=0}^{t} \theta(\tilde{u}^k).$$

Substituting it in (4.6), the assertion of this theorem follows directly. □

It follows from (4.2) that the conclusion (4.4) indicates that any algorithm fitting the prototype (2.5) is able to generate an approximate solution (i.e., $\tilde{w}_t$) of VI($\mathcal{W}, F, \theta$) with an accuracy of $O(1/t)$ after t iterations. That is, a worst-case $O(1/t)$ convergence rate in the ergodic sense is established for the prototype algorithm (2.5). Since we have shown that both the ALM (1.2) and ADMM (1.5) are special cases of the prototype (2.5), Theorem 4.2 particularly shows the worst-case $O(1/t)$ convergence rate in the ergodic sense for ALM and ADMM too. For instance, the result in [17] is a special case of Theorem 4.2.

4.2 Convergence Rate in a Nonergodic Sense

In this subsection, we show that if the matrix G defined in (2.6b) is positive definite, a worst-case $O(1/t)$ convergence rate in a nonergodic sense can also be established for the prototype algorithm (2.5). Note in general a nonergodic convergence rate is stronger than the ergodic convergence rate.

We first need to prove the following lemma.

Lemma 4.3. *For the sequence generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied, we have*

$$\alpha(v^k - \tilde{v}^k)^T M^T H M\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\} \geq \frac{1}{2}\|(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\|^2_{(Q^T+Q)}. \quad (4.7)$$

Proof. First, set $w = \tilde{w}^{k+1}$ in (2.5a), we have

$$\theta(\tilde{u}^{k+1}) - \theta(\tilde{u}^k) + (\tilde{w}^{k+1} - \tilde{w}^k)^T F(\tilde{w}^k) \geq (\tilde{v}^{k+1} - \tilde{v}^k)^T Q(v^k - \tilde{v}^k). \quad (4.8)$$

Note that (2.5a) is also true for $k := k + 1$ and thus we have

$$\theta(u) - \theta(\tilde{u}^{k+1}) + (w - \tilde{w}^{k+1})^T F(\tilde{v}^{k+1}) \geq (v - \tilde{v}^{k+1})^T Q(v^{k+1} - \tilde{v}^{k+1}), \quad \forall\, w \in \mathcal{W}.$$

Set $w = \tilde{w}^k$ in the above inequality, we obtain

$$\theta(\tilde{u}^k) - \theta(\tilde{u}^{k+1}) + (\tilde{w}^k - \tilde{w}^{k+1})^T F(\tilde{w}^{k+1}) \geq (\tilde{v}^k - \tilde{v}^{k+1}) Q(v^{k+1} - \tilde{v}^{k+1}). \quad (4.9)$$

Adding (4.8) and (4.9) and using the monotonicity of F, we get

$$(\tilde{v}^k - \tilde{v}^{k+1})^T Q\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\} \geq 0. \quad (4.10)$$

Adding the term

$$\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\}^T Q\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\}$$

to the both sides of (4.10), and using $v^T Q v = \frac{1}{2} v^T (Q^T + Q) v$, we get

$$(v^k - v^{k+1})^T Q\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\} \geq \frac{1}{2}\|(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\|^2_{(Q^T+Q)}.$$

Substituting $(v^k - v^{k+1}) = \alpha M(v^k - \tilde{v}^k)$ in the left-hand side of the last inequality and using $Q = HM$, we obtain (4.7) and the lemma is proved. □

Now, we are ready to prove (4.11), the key inequality in this section.

Theorem 4.4. *For the sequence generated by the prototype algorithm (2.5) where the Convergence Condition is satisfied, we have*

$$\|M(v^{k+1} - \tilde{v}^{k+1})\|_H \leq \|M(v^k - \tilde{v}^k)\|_H, \quad \forall\, k > 0. \quad (4.11)$$

Proof. Setting $a = M(v^k - \tilde{v}^k)$ and $b = M(v^{k+1} - \tilde{v}^{k+1})$ in the identity

$$\|a\|_H^2 - \|b\|_H^2 = 2a^T H(a-b) - \|a-b\|_H^2,$$

we obtain

$$\begin{aligned}
&\|M(v^k - \tilde{v}^k)\|_H^2 - \|M(v^{k+1} - \tilde{v}^{k+1})\|_H^2 \\
&\quad = \quad 2(v^k - \tilde{v}^k)^T M^T H M\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\} - \|M\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\}\|_H^2.
\end{aligned}$$

Inserting (4.7) into the first term of the right-hand side of the last equality, we obtain

$$\begin{aligned}
&\|M(v^k - \tilde{v}^k)\|_H^2 - \|M(v^{k+1} - \tilde{v}^{k+1})\|_H^2 \\
&\quad \geq \quad \frac{1}{\alpha}\|(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\|_{(Q^T+Q)}^2 - \|M\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\}\|_H^2 \\
&\quad = \quad \frac{1}{\alpha}\|(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\|_G^2 \geq 0,
\end{aligned}$$

where the last inequality is because of the positive definiteness of the matrix $(Q^T+Q) - \alpha M^T H M \succeq 0$. The assertion (4.11) follows immediately. □

Note that it follows from $G \succ 0$ and Theorem 3.2 there is a constant $c_0 > 0$ such that

$$\|v^{k+1} - v^*\|_H^2 \leq \|v^k - v^*\|_H^2 - c_0\|M(v^k - \tilde{v}^k)\|_H^2, \quad \forall v^* \in \mathcal{V}^*. \tag{4.12}$$

Now, with (4.12) and (4.11), we can establish the worst-case $O(1/t)$ convergence rate in a nonergodic sense for the prototype algorithm (2.5).

Theorem 4.5. *Let $\{v^k\}$ and $\{\tilde{w}^k\}$ be the sequences generated by the prototype algorithm (2.5) under the Convergence Condition. For any integer $t > 0$, we have*

$$\|M(v^t - \tilde{v}^t)\|_H^2 \leq \frac{1}{(t+1)c_0}\|v^0 - v^*\|_H^2. \tag{4.13}$$

Proof. First, it follows from (4.12) that

$$\sum_{k=0}^{\infty} c_0\|M(v^k - \tilde{v}^k)\|_H^2 \leq \|v^0 - v^*\|_H^2, \quad \forall\, v^* \in \mathcal{V}^*. \tag{4.14}$$

According to Theorem 4.4, the sequence $\{\|M(v^k - \tilde{v}^k)\|_H^2\}$ is monotonically non-increasing. Therefore, we have

$$(t+1)\|M(v^t - \tilde{v}^t)\|_H^2 \leq \sum_{k=0}^{t} \|M(v^k - \tilde{v}^k)\|_H^2. \tag{4.15}$$

The assertion (4.13) follows from (4.14) and (4.15) immediately. □

Notice that $\mathcal{V}^*$ is convex and closed (see Theorem 4.1). Let $d := \inf\{\|v^0 - v^*\|_H \,|\, v^* \in \mathcal{V}^*\}$. Then, for any given $\epsilon > 0$, Theorem 4.5 shows that the it needs at most $\lfloor d^2/c_0\epsilon \rfloor$ iterations to ensure that $\|M(v^k - \tilde{v}^k)\|_H^2 \leq \epsilon$. Recall that v^k is a solution of VI(Ω, F, θ) if $\|M(v^k - \tilde{v}^k)\|_H^2 = 0$ (see (2.14) and due to $Q = HM$). A worst-case $O(1/t)$ convergence rate in a nonergodic sense is thus established for the prototype algorithm (2.5).

5 A Class of Algorithms for (1.6)

Now we present a class of specific algorithms for the particular model (1.6) based on the prototype algorithm (2.5).

As we have mentioned, because of the verified efficiency of the direct extension of ADMM (1.7) in the literature, we stick to correcting the output of (1.7) via some correction step in algorithmic design. Thus, the step (2.5a) is exactly the direct extension of ADMM (1.7), meaning the matrix Q in (2.5a) being chosen as the matrix defined in (2.15). In this case, $\tilde{w}^k$ is defined by (2.12). For the step (2.5b), we suggest to choose

$$M := M(\tau) = \begin{pmatrix} I & -(1-\tau)(A_2^TA_2)^{-1}A_2^TA_3 & 0 \\ \tau(A_3^TA_3)^{-1}A_3^TA_2 & I & 0 \\ -\beta A_2 & -\beta A_3 & I \end{pmatrix} \text{ with } \tau \in [0,1]. \tag{5.1}$$

Then, with different choices of τ, a class of specific algorithms for solving (1.6) is proposed. We have to verify the Convergence Condition in order to ensure the convergence of the proposed algorithms with Q and M defined in (2.5a) and (5.1), respectively. Also, we have to specify how to choose the constant step size α for the step (2.5b). For convenience, we define the left-upper sub-matrix of Q by Q_0, *i. e.*,

$$Q = \begin{pmatrix} Q_0 & 0 \\ -A_2 \quad -A_3 & \frac{1}{\beta}I \end{pmatrix} \quad \text{and thus} \quad Q_0 = \begin{pmatrix} \beta A_2^TA_2 & 0 \\ \beta A_3^TA_2 & \beta A_3^TA_3 \end{pmatrix}. \tag{5.2}$$

Theorem 5.1. *Let the matrices Q and M be given by* (2.15) *and* (5.1)*, respectively. Let $H = QM^{-1}$ as defined in* (2.6a)*. Then, for $\tau \in [0,1]$, we have*

$$H = \begin{pmatrix} H_0 & 0 \\ 0 \quad 0 & \frac{1}{\beta}I \end{pmatrix},$$

where

$$H_0 = \left[\tau D_0^{-1} + (1-\tau)Q_0^{-T}D_0Q_0^{-1}\right]^{-1}, \tag{5.3}$$

Q_0 is defined in (5.2) *and*

$$D_0 = \begin{pmatrix} \beta A_2^TA_2 & 0 \\ 0 & \beta A_3^TA_3 \end{pmatrix}. \tag{5.4}$$

In addition, the matrix H is positive definite.

Proof. For the matrix M defined in (5.1), we define its left-upper sub-matrix by M_0, *i. e.*,

$$M = \begin{pmatrix} M_0 & 0 \\ -\beta A_2 \quad -\beta A_3 & I \end{pmatrix},$$

where

$$M_0 = \begin{pmatrix} I & -(1-\tau)(A_2^TA_2)^{-1}A_2^TA_3 \\ \tau(A_3^TA_3)^{-1}A_3^TA_2 & I \end{pmatrix}. \tag{5.5}$$

Indeed,

$$HM = \begin{pmatrix} H_0 & 0 \\ 0 \quad 0 & \frac{1}{\beta}I \end{pmatrix} \begin{pmatrix} M_0 & 0 \\ -\beta A_2 \quad -\beta A_3 & I \end{pmatrix} = \begin{pmatrix} H_0 M_0 & 0 \\ -A_2 \quad -A_3 & \frac{1}{\beta}I \end{pmatrix}.$$

According to (5.2), in order to show $HM = Q$, we need only to verify that

$$H_0 M_0 = Q_0. \tag{5.6}$$

Note that (see (5.5))

$$M_0 = \tau \begin{pmatrix} I & 0 \\ (A_3^T A_3)^{-1} A_3^T A_2 & I \end{pmatrix} + (1-\tau) \begin{pmatrix} I & -(A_2^T A_2)^{-1} A_2^T A_3 \\ 0 & I \end{pmatrix}. \tag{5.7}$$

Observing the two parts in (5.7), the first part

$$\begin{pmatrix} I & 0 \\ (A_3^T A_3)^{-1} A_3^T A_2 & I \end{pmatrix} = \begin{pmatrix} \beta A_2^T A_2 & 0 \\ 0 & \beta A_3^T A_3 \end{pmatrix}^{-1} \begin{pmatrix} \beta A_2^T A_2 & 0 \\ \beta A_3^T A_2 & \beta A_3^T A_3 \end{pmatrix} = D_0^{-1} Q_0, \tag{5.8}$$

while the second part is

$$\begin{aligned} \begin{pmatrix} I & -(A_2^T A_2)^{-1} A_2^T A_3 \\ 0 & I \end{pmatrix} &= \begin{bmatrix} I & (A_2^T A_2)^{-1} A_2^T A_3 \\ 0 & I \end{bmatrix}^{-1} \\ &= \left[\begin{pmatrix} (\beta A_2^T A_2)^{-1} & 0 \\ 0 & (\beta A_3^T A_3)^{-1} \end{pmatrix} \begin{pmatrix} \beta A_2^T A_2 & \beta A_2^T A_3 \\ 0 & \beta A_3^T A_3 \end{pmatrix} \right]^{-1} \\ &= (D_0^{-1} Q_0^T)^{-1} = Q_0^{-T} D_0. \end{aligned} \tag{5.9}$$

Substituting (5.8) and (5.9) into (5.7), we obtain

$$M_0 = \tau D_0^{-1} Q_0 + (1-\tau) Q_0^{-T} D_0 = \big(\tau D_0^{-1} + (1-\tau) Q_0^{-T} D_0 Q_0^{-1}\big) Q_0. \tag{5.10}$$

Indeed, by using (5.3) and the last equation, we have $M_0 = H_0^{-1} Q_0$ and thus (5.6) is verified. Since

$$H_0^{-1} = \tau D_0^{-1} + (1-\tau) Q_0^{-T} D_0 Q_0^{-1}$$

is a convex combination of the positive definite matrices D_0^{-1} and $Q_0^{-T} D_0 Q_0^{-1}$, the positivity of H_0 follows immediately. Consequently, H is positive definite. The proof is complete. □

Since $Q^T + Q$ and H are positive definite, we define

$$\alpha(\tau) := \arg\max\{\alpha \,|\, Q^T + Q - \alpha M^T H M \succeq 0\}, \tag{5.11}$$

and thus $\alpha(\tau) > 0$. For any $\alpha \in (0, \alpha(\tau))$, the matrix $Q^T + Q - \alpha M^T H M$ is positive definite. In order to estimate the magnitude of $\alpha(\tau)$, we need the expression in the following lemma.

Lemma 5.2. *Let the matrices Q and M be given by* (2.15) *and* (5.1)*, respectively, and $H = QM^{-1}$. Then for any $\alpha > 0$, we have*

$$\begin{aligned} &Q^T + Q - \alpha M^T H M \\ &= \begin{pmatrix} 2(1-\alpha)\beta A_2^T A_2 - \alpha\tau\beta A_2^T A_3 (A_3^T A_3)^{-1} A_3 A_2^T & (1-\alpha(1+\tau))\beta A_2^T A_3 & -(1-\alpha)A_2^T \\ (1-\alpha(1+\tau))\beta A_3^T A_2 & 2(1-\alpha)\beta A_3^T A_3 & -(1-\alpha)A_3^T \\ -(1-\alpha)A_2 & -(1-\alpha)A_3 & \frac{2-\alpha}{\beta} I \end{pmatrix}. \end{aligned} \tag{5.12}$$

Proof. Note that (see (2.15))

$$Q^T + Q = \begin{pmatrix} 2\beta A_2^T A_2 & \beta A_2^T A_3 & -A_2^T \\ \beta A_3^T A_2 & 2\beta A_3^T A_3 & -A_3^T \\ -A_2 & -A_3 & \frac{2}{\beta} I \end{pmatrix}. \tag{5.13}$$

Since $Q = HM$, thus

$$\begin{aligned} M^T H M &= Q^T M \\ &= \begin{pmatrix} \beta A_2^T A_2 & \beta A_2^T A_3 & -A_2^T \\ 0 & \beta A_3^T A_3 & -A_3^T \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & -(1-\tau)(A_2^T A_2)^{-1} A_2^T A_3 & 0 \\ \tau (A_3^T A_3)^{-1} A_3^T A_2 & I & 0 \\ -\beta A_2 & -\beta A_3 & I \end{pmatrix} \\ &= \begin{pmatrix} 2\beta A_2^T A_2 + \tau \beta A_2^T A_3 (A_3^T A_3)^{-1} A_3 A_2^T & (1+\tau)\beta A_2^T A_3 & -A_2^T \\ (1+\tau)\beta A_3^T A_2 & 2\beta A_3^T A_3 & -A_3^T \\ -A_2 & -A_3 & \frac{1}{\beta} I \end{pmatrix}. \end{aligned} \tag{5.14}$$

By using (5.13) and (5.14), we get (5.12) and the assertion is proved. □

Theorem 5.3. *Let the matrices Q and M be given by* (2.15) *and* (5.1)*, respectively, and $H = QM^{-1}$. Then*

1. *if $\tau = 0$, then*

$$G = Q^T + Q - \alpha M^T H M \begin{cases} \succeq 0, & \text{for} \quad \alpha = 1; \\ \succ 0, & \forall\, \alpha \in (0,1). \end{cases} \tag{5.15}$$

2. *in the case $\tau \in (0,1]$,*

$$G = Q^T + Q - \alpha M^T H M \succ 0, \ \text{ for } \ \alpha = \frac{1}{1+\tau}. \tag{5.16}$$

Proof. If $\tau = 0$, it follows from (5.12) that

$$\begin{aligned} G &= Q^T + Q - \alpha M^T H M \\ &= \begin{pmatrix} 2(1-\alpha)\beta A_2^T A_2 - & (1-\alpha)\beta A_2^T A_3 & -(1-\alpha)A_2^T \\ (1-\alpha)\beta A_3^T A_2 & 2(1-\alpha)\beta A_3^T A_3 & -(1-\alpha)A_3^T \\ -(1-\alpha)A_2 & -(1-\alpha)A_3 & \frac{2-\alpha}{\beta} I \end{pmatrix} \\ &= \beta \begin{pmatrix} A_2^T & 0 & 0 \\ 0 & A_3^T & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} 2(1-\alpha)I & (1-\alpha)I & -(1-\alpha)I \\ (1-\alpha)I & 2(1-\alpha)I & -(1-\alpha)I \\ -(1-\alpha)I & -(1-\alpha)I & (2-\alpha)I \end{pmatrix} \begin{pmatrix} A_2 & 0 & 0 \\ 0 & A_3 & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}. \end{aligned}$$

Since

$$\begin{pmatrix} 2(1-\alpha) & (1-\alpha) & -(1-\alpha) \\ (1-\alpha) & 2(1-\alpha) & -(1-\alpha) \\ -(1-\alpha) & -(1-\alpha) & (2-\alpha) \end{pmatrix} \begin{cases} \succeq 0, & \text{for} \quad \alpha = 1; \\ \succ 0, & \forall\, \alpha \in (0,1), \end{cases}$$

the assertion (5.15) is proved. Now, we turn to prove the second part. Set $\alpha = 1/(1+\tau)$ in (5.12), we obtain

$$\begin{aligned} G &= \begin{pmatrix} \frac{2\tau}{1+\tau}\beta A_2^T A_2 - \frac{\tau}{1+\tau}\beta A_2^T A_3 (A_3^T A_3)^{-1} A_3 A_2^T & 0 & -\frac{\tau}{1+\tau}A_2^T \\ 0 & \frac{2\tau}{1+\tau}\beta A_3^T A_3 & -\frac{\tau}{1+\tau}A_3^T \\ -\frac{\tau}{1+\tau}A_2 & -\frac{\tau}{1+\tau}A_3 & \frac{(1+2\tau)}{(1+\tau)\beta}I \end{pmatrix} \\ &= \frac{\beta}{1+\tau}\begin{pmatrix} A_2^T & 0 & 0 \\ 0 & A_3^T & 0 \\ 0 & 0 & \frac{1}{\beta}I \end{pmatrix}\begin{pmatrix} 2\tau I - \tau A_3(A_3^T A_3)^{-1}A_3^T & 0 & -\tau I \\ 0 & 2\tau I & -\tau I \\ -\tau I & -\tau I & (1+2\tau)I \end{pmatrix}\begin{pmatrix} A_2 & 0 & 0 \\ 0 & A_3 & 0 \\ 0 & 0 & \frac{1}{\beta}I \end{pmatrix} \quad (5.17) \end{aligned}$$

Because $I - A_3(A_3^T A_3)^{-1}A_3^T \succeq 0$, it follows that

$$\begin{pmatrix} 2\tau I - \tau A_3(A_3^T A_3)^{-1}A_3^T & 0 & -\tau I \\ 0 & 2\tau I & -\tau I \\ -\tau I & -\tau I & (1+2\tau)I \end{pmatrix} \succeq \begin{pmatrix} \tau I & 0 & -\tau I \\ 0 & 2\tau I & -\tau I \\ -\tau I & -\tau I & (1+2\tau)I \end{pmatrix} \succ 0.$$

Thus, G is positive definite for all $\tau \in (0,1]$ and $\alpha = \frac{1}{1+\tau}$. The proof is complete. □

Theorems 5.1 and 5.3 mean the Convergence Condition is satisfied. Hence, the prototype algorithm (2.5) with Q in (2.5a) and M in (5.1) is convergent.

The proof of Theorems 5.1 and 5.3 also gives us the guidance of how to choose the step size α for the step (2.5b). In fact, for any fixed $\tau \in (0,1]$, according to (5.11) and (5.16), we have $\alpha(\tau) > \frac{1}{1+\tau}$, meaning the step size could be chosen easily and it being bounded below away for any given $\tau \in (01]$. In fact, for some special values of τ, by a careful manipulation we can get tighter lower bounds of $\alpha(\tau)$. Note we have

$$\begin{aligned} &Q^T + Q - \alpha M^T H M \\ &= \beta\begin{pmatrix} A_2^T & 0 & 0 \\ 0 & A_3^T & 0 \\ 0 & 0 & \frac{1}{\beta}I \end{pmatrix}\begin{pmatrix} (2(1-\alpha)-\alpha\tau)I & (1-\alpha(1+\tau))I & -(1-\alpha)I \\ (1-\alpha(1+\tau))I & 2(1-\alpha)I & -(1-\alpha)I \\ -(1-\alpha)I & -(1-\alpha)I & (1+2\tau)I \end{pmatrix}\begin{pmatrix} A_2 & 0 & 0 \\ 0 & A_3 & 0 \\ 0 & 0 & \frac{1}{\beta}I \end{pmatrix}. \end{aligned}$$

In order to ensure $Q^T + Q - \alpha M^T H M \succ 0$, we can choose α such that

$$\begin{pmatrix} 2(1-\alpha)-\alpha\tau & 1-\alpha(1+\tau) & -(1-\alpha) \\ 1-\alpha(1+\tau) & 2(1-\alpha) & -(1-\alpha) \\ -(1-\alpha) & -(1-\alpha) & (1+2\tau) \end{pmatrix} \succ 0.$$

In the following table, we list the lower bounds of $\alpha(\tau)$ for some values of τ.

Table 1: Tighter lower bounds of $\alpha(\tau)$ for some values of τ

$\tau =$	1/5	1/4	1/3	1/2	2/3
$\alpha(\tau) >$	7/8	6/7	4/5	3/4	5/8

Now we are ready to present a class of specific algorithms for (1.6) based on the prototype algorithm (2.5).

A Class of Algorithms for (1.6)

[Step 1.] With the given $v^k = (x_2^k, x_3^k, \lambda^k)$, generate a vector $\tilde{w}^k \in \mathcal{W}$ via

$$\begin{cases} \tilde{x}_1^k = \operatorname{argmin}\{\theta_1(x_1) - \lambda^T\big(A_1x_1\big) + \dfrac{\beta}{2}\big\|A_1x_1 + A_2x_2^k + A_3x_3^k - b\big\|^2 \,|\, x_1 \in \mathcal{X}_1\}, & (5.18aa) \\ \tilde{x}_2^k = \operatorname{argmin}\{\theta_2(x_2) - \lambda^T\big(A_2x_2\big) + \dfrac{\beta}{2}\big\|A_1\tilde{x}_1^k + A_2x_2 + A_3x_3^k - b\big\|^2 \,|\, x_2 \in \mathcal{X}_2\}, & (5.18ab) \\ \tilde{x}_3^k = \operatorname{argmin}\{\theta_3(x_3) - \lambda^T\big(A_3x_3\big) + \dfrac{\beta}{2}\big\|A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3x_3 - b\big\|^2 \,|\, x_3 \in \mathcal{X}_3\}, & (5.18ac) \\ \tilde{\lambda}^k = \lambda^k - \beta(A_1\tilde{x}_1^k + A_2\tilde{x}_2^k + A_3\tilde{x}_3^k - b). & (5.18ad) \end{cases}$$

[Step 2.] Generate the new iterate $v^{k+1} = (x_2^{k+1}, x_3^{k+1}, \lambda^{k+1})$ by

$$\begin{pmatrix} x_2^{k+1} \\ x_3^{k+1} \\ \lambda^{k+1} \end{pmatrix} := \begin{pmatrix} x_2^k \\ x_3^k \\ \lambda^k \end{pmatrix} - \alpha \begin{pmatrix} I & -(1-\tau)(A_2^TA_2)^{-1}A_2^TA_3 & 0 \\ \tau(A_3^TA_3)^{-1}A_3^TA_2 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_2^k - \tilde{x}_2^k \\ x_3^k - \tilde{x}_3^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}, \quad (5.18b)$$

where $\tau \in [0,1]$, $\alpha \in (0, \alpha(\tau)]$ and $\alpha(\tau)$ is defined in (5.11).

Remark 5.4. As we have mentioned, for most of the applications of (1.6), the coefficient matrices are identity matrices. For such a case, since $A_2 = A_3 = I_{m\times m}$, the step (5.18b) can be further simplified to

$$\begin{pmatrix} x_2^{k+1} \\ x_3^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x_2^k \\ x_3^k \\ \lambda^k \end{pmatrix} - \alpha \begin{pmatrix} I & -(1-\tau)I & 0 \\ \tau I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_2^k - \tilde{x}_2^k \\ x_3^k - \tilde{x}_3^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}.$$

In particular, if we choose $\tau = \frac{1}{2}$, then according to Table 1, we can just choose the step size as $\frac{3}{4}$. The above step can be specified as

$$\begin{pmatrix} x_2^{k+1} \\ x_3^{k+1} \\ \lambda^{k+1} \end{pmatrix} := \begin{pmatrix} x_2^k \\ x_3^k \\ \lambda^k \end{pmatrix} - \frac{3}{4} \begin{pmatrix} I & -\frac{1}{2}I & 0 \\ \frac{1}{2}I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_2^k - \tilde{x}_2^k \\ x_3^k - \tilde{x}_3^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix},$$

which is really simple with almost no additional computation.

Remark 5.5. As we will show, these algorithms (5.18) could even numerically outperform the direct extension of ADMM (1.7) when it is indeed convergent, and their convergence can be proved rigorously. These features make them significantly different from the existing splitting versions of the ALM (1.2) such as [15, 16]. The outstanding numerical performance of (5.18) could be explained intuitively as follows. Taking a look at the step (5.18b), we see that this correction step does not change the output $\tilde{\lambda}^k$ generated by the direct extension of ADMM (5.18a); and only make a balance between the variables x_2 and x_3 to generate a new iterate. For the ADMM scheme (1.5), the primal variables x_1 and x_2 play different roles —- x_1 is an intermediate variable which is not involved in the iteration and the only primal variable required by the iteration is x_2. But for the direct extension of

ADMM (1.7), there are two primal variables x_2 and x_3 that are essentially required by the iteration. The scheme (1.5), however, treats these two essential primal variable differently —- they are updated consecutively and the update of x_3 uses the newest x_2. This lack of symmetry in primal variables may also explain why it fails to show the convergence of the direct extension of ADMM (1.7). The step (5.18b) makes a balance between the output of x_2 and x_3 generated by the direct extension of ADMM (5.18a), and it may be regarded as a compensation to the loss of symmetry over x_2 and x_3 in (1.7). This may explain the reason why sometimes the algorithms (5.18) outperform the direct extension of ADMM (1.7), because the effort of balancing x_2 and x_3 may make the whole sequence $\{x_2^k, x_3^k, \lambda^k\}$ converge to the solution set in a more balance way and thus on a faster speed.

Remark 5.6. Note that we just need to check the convergence condition (2.6) when implementing the proposed algorithm (5.18); and do not need to compute the matrices H and G.

6 Numerical Results

In this section, we apply the proposed algorithms (5.18) to test a concrete application of the model (1.6) arising in image processing; and report the numerical results. Our codes were written in MATLAB 7.9 and they were run on a Lenovo personal computer with Intel Core (TM) CPU 2.30GHZ and 8G memory.

6.1 An Image Decomposition Model

We focus on the image decomposition model proposed recently in [33]. First, we review some background of the image decomposition problem briefly. Let $f \in \mathcal{R}^n$ represent a digital image. Note that in our discussion a two-dimensional image is stacked as a one-column vector, e.g., in the lexicographic order, and the pixel values of an image are re-scaled into the interval [0,1]. The goal of image decomposition is to decompose an image into two meaningful components. Image decomposition is an important problem in image processing and it plays a significant role in many realms such as object recognition and biomedical engineering. A very useful task is to decompose the target image f into two components u and v, i.e., $f = u + v$, where u represents the cartoon part containing the structural component and sketchy approximations of f and v is the texture part containing the oscillations and repeated patterns of f. See more details in, e.g., [1, 22, 26, 27, 34, 36]. Analytically, the cartoon part u can be described as a piecewise smooth function and the texture part v is typically an oscillating function. In addition to the data fidelity term $\|(u+v) - f\|_2^2$, we usually have to use other terms to mathematically characterize these two parts in the objective function of an image decomposition model. For the cartoon part u, it is standard to characterize it by the total variation semi-norm in [32]. To capture the feature of the texture part v, in [33] it was suggest to first partition the texture part v of an n_1-by-n_2 (with $n = n_1 \cdot n_2$) image orderly into a series of r-by-r (with $r \ll n$) non-overlapping patches under some boundary condition; then stack each individual patch as a column vector, denoted by $w_i \in R^{r^2}$ $(i = 1, 2, \cdots, l)$. Here, $l = \lceil n/r^2 \rceil$ with $\lceil \cdot \rceil$ rounding a scalar as the nearest integer towards infinity. By further realigning all w_i's together, an r^2-by-l matrix, denoted by V, is attained. The patch mapping $\mathcal{P} : \mathcal{R}^n \to \mathcal{R}^{r^2 \times l}$, which describes the preceding process on rearranging the texture part of an image v as a matrix V, is defined as

$$V := \mathcal{P}v = [w_1, w_2, \cdots, w_l].$$

We refer to [33] for more details. Since the texture part of a target image possesses many repeated patterns, the matrix V, i.e., $\mathcal{P}v$, can be analytically perceived as a low-rank matrix. Accordingly,

the following low-rank based model for image decomposition was developed in [33]:

$$\min_{u\in\mathcal{R}^n, v\in\mathcal{R}^n} \tau_1 |||\nabla u|||_1 + \tau_2 \|\mathcal{P}v\|_* + \frac{\tau_3}{2}\|K(u+v)-f\|_2^2. \tag{6.1}$$

In (6.1), $|||\nabla\cdot|||_1$ denotes the total variation semi-norm in [32] in order to induce the cartoon part u; $\|\cdot\|_*$ is the nuclear norm which is defined as the sum of all singular values of a matrix in order to reflect the low-rank feature of the matrix $\mathcal{P}v$ where v is the texture part; K is a linear operator corresponding to certain corruption on the target image f: (i) $K = I$ with I being the identity matrix corresponds to the case where the image f is clean without noise, (ii) $K = S$ with S being a diagonal binary matrix corresponds to the case where some pixels of f are missing (we assume that the missing pixels admit zero values), and (iii) $K = B$ with B being a convolutional matrix associated with a spatially invariant point spread function corresponds to the case where f is corrupted by some blur; the parameters τ_i $(i = 1, 2, 3)$ are positive scalars to balance the three terms (piecewise constant feature of the cartoon part, low-rank feature of the patched texture part and the data-fidelity of the decomposition) in the objective function.

6.2 Reformulation of (6.1)

To solve the patched low-rank image decomposition model (6.1), we first reformulate it as a special case of the model (1.6) and then delineate the subproblems when the proposed algorithms (5.18) are applied.

Introducing the auxiliary variables $x \in \mathcal{R}^n \times \mathcal{R}^n$, $y \in \mathcal{R}^{r^2\times l}$ and $z \in \mathcal{R}^n$, the model (6.1) can be rewritten as

$$\begin{array}{ll} \min & \tau_1 |||x|||_1 + \tau_2\|y\|_* + \frac{\tau_3}{2}\|Kz-f\|_2^2 \\ \text{s.t.} & x = \nabla u \\ & y = \mathcal{P}v \\ & z = u+v. \end{array} \tag{6.2}$$

Thus, (6.2) is a special case of the abstract model (1.6) with the following specifications:

- $x_1 := u$, $x_2 := v$, $x_3 := (x, y, z)$, and all $\mathcal{X}_i$'s $(i = 1, 2, 3)$ are full Euclidean spaces;
- $\theta_1(x_1) := 0$, $\theta_2(x_2) := 0$ and $\theta_3(x_3) := \tau_1 |||x|||_1 + \tau_2\|y\|_* + \frac{\tau_3}{2}\|Kz-f\|_2^2$;
- The coefficients A_i $(i = 1, 2, 3)$ and the vector b are given by:

$$A_1 := \begin{pmatrix} \nabla \\ 0 \\ I \end{pmatrix}, \quad A_2 := \begin{pmatrix} 0 \\ \mathcal{P} \\ I \end{pmatrix}, \quad A_3 := \begin{pmatrix} -I & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{pmatrix} \quad \text{and} \quad b := 0.$$

Now, let us analyze the main subproblems when the proposed algorithms (5.18) are applied to solve the reformulation (6.2). Since the algorithms in (5.18) differs only in the correction steps and they share the same ADMM subproblems, we only focus on the detail of the resulting ADMM subproblems.

- The $\tilde{x}_1$-subproblem in (5.18), i.e., the $\tilde{u}$-subproblem for (6.2), is

$$\begin{array}{rcl} \tilde{u}^k & = & \arg\min\limits_u \left\{ \frac{\beta_1}{2}\left\|\nabla u - x^k - \frac{\lambda_1^k}{\beta_1}\right\|_2^2 + \frac{\beta_3}{2}\left\|u + v^k - z^k - \frac{\lambda_3^k}{\beta_3}\right\|_2^2 \right\} \\ & \Leftrightarrow & (\beta_1\nabla^T\nabla + \beta_3 I)u = \nabla^T(\beta_1 x^k + \lambda_1^k) + \beta_3(z^k - v^k) + \lambda_3^k, \end{array}$$

which has a closed-form solution. In fact, this system of equations can be solved efficiently by using fast Fourier transform (FFT) or discrete cosine transform (DCT) if the periodic or reflective boundary condition is employed for the derivative operator ∇ (see more details in, e.g., [14, Chapter 7]).

- The $\tilde{x}_2$-subproblem in (5.18), i.e., the $\tilde{v}$-subproblem, reads as

$$\begin{array}{rcl} \tilde{v}^k & = & \arg\min\limits_{v} \left\{ \frac{\beta_2}{2} \left\| \mathcal{P}v - y^k - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 + \frac{\beta_3}{2} \left\| u^{k+1} + v - z^k - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \right\} \\ & = & \frac{1}{\beta_2+\beta_3} \left[\mathcal{P}^{-1}(\beta_2 y^k - \lambda_2^k) + \beta_3(z^k - u^k) + \lambda_3^k \right]. \end{array}$$

  See more details in [33].

- The $\tilde{x}_3$-subproblem in (5.18), i.e., the $(\tilde{x}, \tilde{y}, \tilde{z})$-subproblem, is

$$\begin{array}{rcl} (\tilde{x}^k, \tilde{y}^k, \tilde{z}^k) & = & \arg\min\limits_{x,y,z} \left\{ \tau_1 |||x|||_1 + \tau_2 \|y\|_* + \frac{\tau_3}{2} \|Kz - f\|_2^2 + \frac{\beta_1}{2} \left\| \nabla\tilde{u}^k - x - \frac{\lambda_1^k}{\beta_1} \right\|_2^2 \right\} \\ & & + \frac{\beta_2}{2} \left\| \mathcal{P}\tilde{v}^k - y - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 + \frac{\beta_3}{2} \left\| \tilde{u}^k + \tilde{v}^k - z - \frac{\lambda_3^k}{\beta_3} \right\|_2^2, \end{array}$$

  and all the variables $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$ can be solved simultaneously as follows.

  - The $\tilde{x}$-subproblem can be solved explicitly by the soft-thresholding operator

$$\begin{array}{rcl} \tilde{x}^k & = & \arg\min\limits_{x} \left\{ \tau_1 |||x|||_1 + \frac{\beta_1}{2} \left\| \nabla\tilde{u}^k - x - \frac{\lambda_1^k}{\beta_1} \right\|_2^2 \right\} \\ & = & \mathrm{shrink}_{\frac{\tau_1}{\beta_1}} \left(\nabla\tilde{u}^k - \frac{\lambda_1^k}{\beta_1} \right), \end{array}$$

    where, for any $c > 0$, the mapping $\mathrm{shrink}_c(\cdot)$ is defined as

$$\mathrm{shrink}_c(g) := g - \min\{c, |g|\} \frac{g}{|g|}, \ \ \forall g \in \mathcal{R}^n \times \mathcal{R}^n,$$

    and $(\frac{g}{|g|})_i$ should be taken as 0 if $|g|_i = 0$.

  - The $\tilde{y}$-subproblem can be solved explicitly by the singular value decomposition (SVD)

$$\tilde{y}^k = \arg\min_{y} \left\{ \tau_2 \|y\|_* + \frac{\beta_2}{2} \left\| \mathcal{P}\tilde{v}^k - y - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 \right\} = \mathcal{D}_{\frac{\tau_2}{\beta_2}} \left(\mathcal{P}\tilde{v}^k - \frac{\lambda_2^k}{\beta_2} \right),$$

    Here, for any $c > 0$, the mapping $\mathcal{D}_c(\cdot)$ is defined as

$$\mathcal{D}_c(T) := U\hat{\Sigma}V^T, \ \ \forall T \in \mathcal{R}^{m\times n}, \tag{6.3}$$

    where $U\Sigma V^T$ is the SVD of T, and $\hat{\Sigma}_{ij} = \max\{\Sigma_{ij} - c, 0\}$ for all $1 \le i \le m$ and $1 \le j \le n$.

  - The $\tilde{z}$-subproblem is solved involving in the linear operator K

$$\begin{array}{rcl} \tilde{z}^k & = & \arg\min\limits_{z} \left\{ \frac{\tau_3}{2} \|Kz - f\|_2^2 + \frac{\beta_3}{2} \left\| \tilde{u}^k + \tilde{v}^k - z - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \right\} \\ & \Leftrightarrow & (\tau_3 K^T K + \beta_3) z = \tau_3 K^T f + \beta_3(\tilde{u}^k + \tilde{v}^k) - \lambda_3^k. \end{array} \tag{6.4}$$

    If K is the identity or diagonal matrix, then $\tilde{z}$ in (6.4) can be attained directly. If K is a convolutional matrix, then $\tilde{z}$ can be solved efficiently by using FFT or DCT.

Therefore, when the proposed algorithms (5.18) are applied to solve the image decomposition model (6.1), all the resulting subproblems are easy to solve. This fact contributes much to the efficiency of the new algorithms, as we shall see in the next subsection.

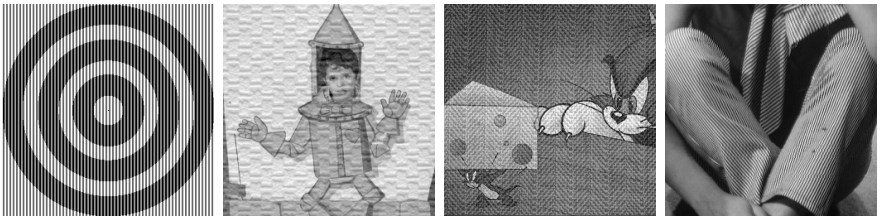

Figure 1: Test images. From left to right: 256×256 Circles.png, 256×256 Boy.png, 512×512 Tomjerry.png and 256×256 Barbara.png.

6.3 Numerical Results

Now, we test different cases of (6.1) where $K = I$ (the case where f is clean), $K = S$ (the case where some pixels of f are missing) and $K = B$ (the case where f is corrupted by blur). The images to be tested are displayed in Figure 1.

Note that our purpose is to numerically verify the efficiency of the algorithms in (5.18) for a given case of the model (1.6); especially to see the difference of various choices of the step size α for different values of τ. As we have mentioned, the direct extension of ADMM (1.7) ("EADMM" for abbreviation) is not necessarily convergent; but it does work very well for many applications if it is indeed convergent. In fact, it usually represents the fast scenario among the existing splitting methods originated from the ALM in the literature (to the best of our knowledge, some exceptional cases can only be found in [16]). We thus use EADMM as the benchmark for comparison in our experiments. Furthermore, we do not discuss how to determine the values of the parameters (τ_1, τ_2, τ_3) in the model (6.1), which is not the theme of this paper. Instead, we simply follow the suggestion in [33, Theorem3.3-3.6] to choose these parameters. That is, we adopt $\tau_1 \in [10^{-2}, 10^{-1}]$, $\tau_2 \in [10^{-3}, 10^{-2}]$ (their precise values will be specified later when a particular case is discussed) and $\tau_3 \equiv 1$. For the patch size, i.e., the scalar r of the mapping $\mathcal{P}$, it can be easily estimated by the number of spikes of the target image f in the Fourier domain (see [33] for details). Empirically, the integer r is chosen as 11 throughout all numerical simulations.

Some other set-ups of the experiments are as follows. (1) We adopt the periodic boundary condition for all the images to be tested and thus the FFT will be used for the resulting system of equations. (2) The SVD in (6.3) is executed by an efficient MATLAB Mex interface for computing SVD via a divide-and-conquer routine (dgesdd) implemented in LAPACK. (3). For a chosen τ, the step size α for the correction step in (5.18b) can be any value in the interval $(0, \alpha(\tau)]$. We only report the case where the upper bound is chosen, i.e., $\alpha = \alpha(\tau)$, because usually we need to avoid smaller step size whenever possible. (4). The penalty parameter β is chosen by the cross-validation technique. More specifically, as analyzed in [1], the quality of image decomposition can be measured by the magnitude of the correlation between the cartoon u and texture v obtained via an image decomposition model:

$$\mathrm{Corr}(u, v) := \mathrm{cov}(u, v) \Big/ \sqrt{\mathrm{var}(u) \cdot \mathrm{var}(v)} \tag{6.5}$$

where $\mathrm{var}(\cdot)$ and $\mathrm{cov}(\cdot, \cdot)$ are the variance and covariance of two given variables, respectively. Therefore, to determine an appropriate value of β, we focus on the implementation of the algorithm in (5.18) with $(\tau, \alpha) = (1/5, 7/8)$ to the special case of the model (6.1) with $K = I$. We take the Barbara and Tomjerry images in Figure 1 as the tested images. A total number of 50 cases of β in the interval $[10^{-3}, 10^2]$ are tested on the points 10^{-k} where k varies from 10^{-3} to 10^2 with an equal distance of 0.2. For each selected value of β, we run the algorithm by $1,000$ iterations and record the

objective function value for the model (6.1) and the magnitude of the correlation $\mathrm{Corr}(u, v)$. The plots are displayed in Figure 2. This figure indicates that values in the interval $(0.1, 1)$ can yield relatively lower objective function values and correlation values. As a result, we hereafter fix the penalty parameter $\beta_i \equiv 1$ throughout our experiments. In Figure 3, the decomposed cartoon and texture parts are displayed for some values of β in different levels of magnitude.

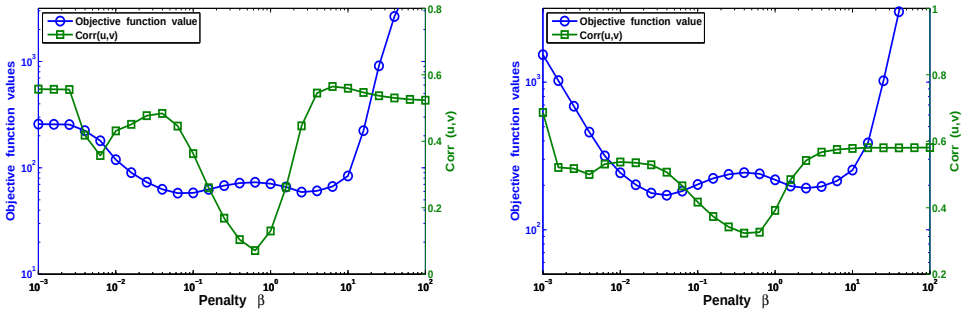

Figure 2: The implementation of (5.18) with $(\tau, \alpha) = (1/5, 7/8)$ to (6.1) with $K = I$ for Barbara, with different β. Evolution of the objective function values and $\mathrm{Corr}(u, v)$ w.r.t. different β. Left: for Barbara. Right: for Tomjerry.

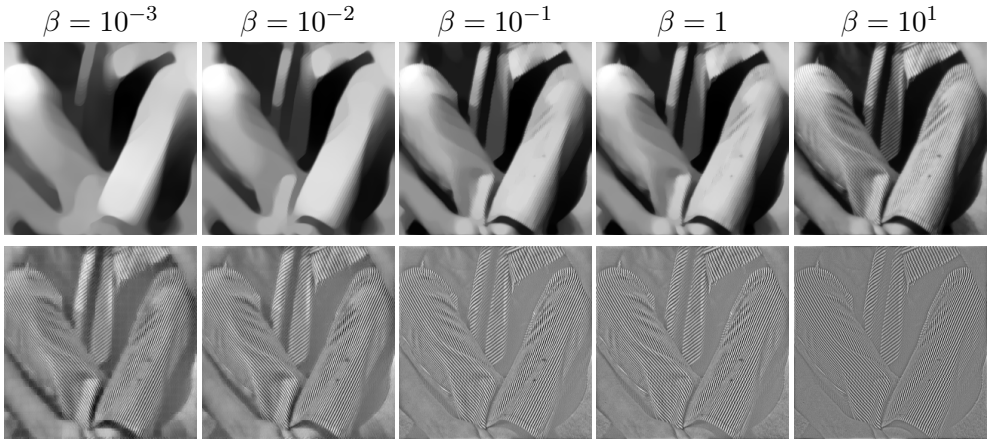

Figure 3: The implementation of (5.18) with $(\tau, \alpha) = (1/5, 7/8)$ to (6.1) with $K = I$ for Barbaba, with different β. Decomposed cartoons (Top row) and textures (Bottom row).

6.3.1 The Case of $K = I$

For the case $K = I$, the model (6.1) is for decomposing a clean image without any degradation. The parameters (τ_1, τ_2, τ_3) in (6.1) are fixed as $(\tau_1, \tau_2, \tau_3) = (0.01, 0.005, 1)$, as suggested in [33]. The initial iterates for implementing the algorithms in (5.18) are taken as zeros.

In Figure 4, we plot the evolutions of the objective function values for the model (6.1) with respect to the number of iterations and computing time in seconds, for several cases of the algorithms in (5.18) with different choices of τ. It is shown that that the proposed algorithms can easily outperform EADMM. These plots also show that a smaller value of τ seems more preferable because it can yield a larger step size $\alpha(\tau)$ for the correction steps in (5.18b). The decomposed cartoon and texture parts by implementing the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations are displayed in Figure 5.

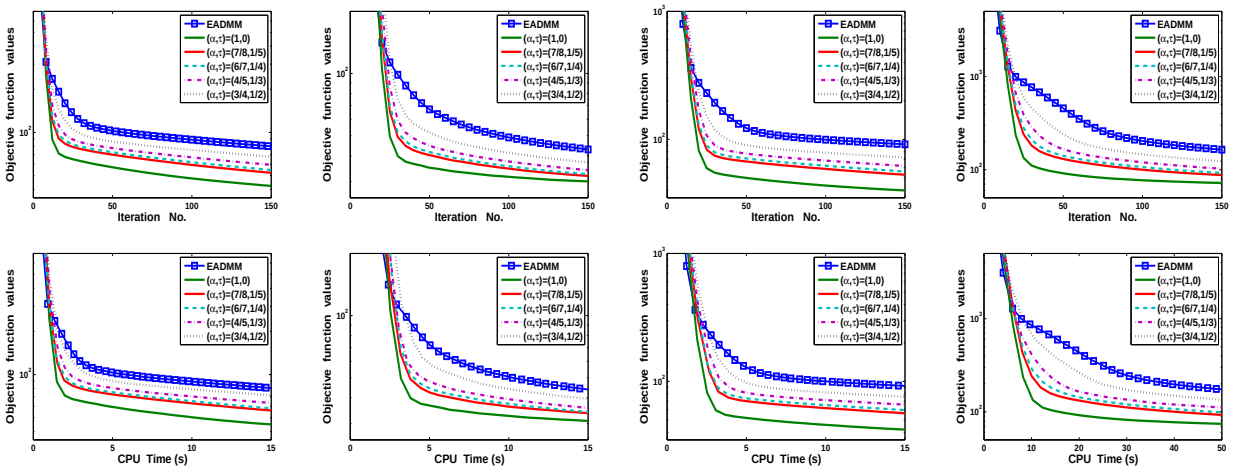

Figure 4: The implementation of (5.18) with different (τ, α) to (6.1) with $K = I$, $\beta = 1$. Evolutions of the objective function values w.r.t. the number of iterations (upper row) and computing time in seconds (lower row). From left column to right column: for Circles, Boy, Barbara and Tomjerry images.

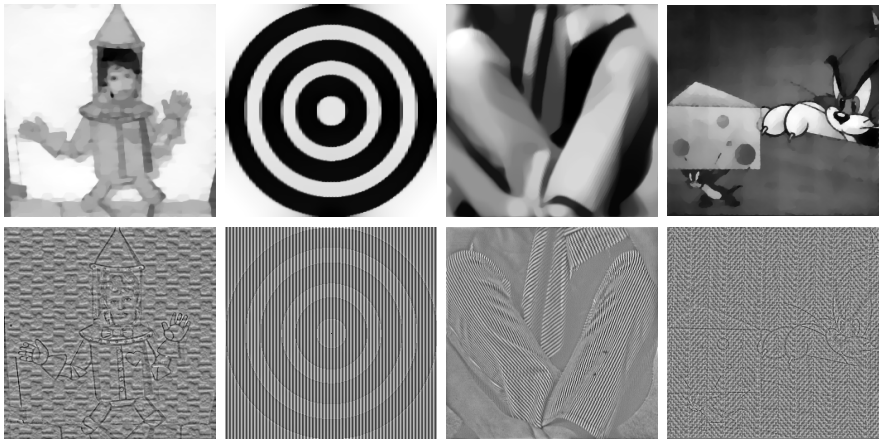

Figure 5: Decomposed cartoon (Top row) and texture (Bottom row) parts by implementing (5.18) with $(\tau, \alpha) = (1/5, 7/8)$ to (6.1) with $K = I$ for 150 iterations.

6.3.2 The Case of $K = S$

We now test the case of model (6.1) with $K = S$. That is, some pixels of the image under test are missing. For succinctness, we only test the Barbara image with 9.12% missing pixels and the Tomjerry image with 12.76% missing pixels. The degraded images with missing pixels are shown in Figure 6.

For this case, the parameters (τ_1, τ_2, τ_3) in (6.2) are chosen as $(0.08, 0.005, 1)$, as suggested in [33, Theorem 3.3-3.6]. All initial iterates for implementing the proposed algorithms in (5.18) are taken as zeros.

In addition to the objective function value, we also report the signal-to-noise ratio (SNR) value in unit of dB which is commonly used to measure the quality of the restored/reconstructed images. It is defined as

$$\mathrm{SNR} = 20 \log_{10} \frac{\|f^*\|}{\|\bar{f} - f^*\|},$$

where $\bar{f}$ is the approximation of the ground truth f^*. For the corrupted images listed in Figure 6, the SNR values are 10.88dB for Barbara image and 9.06dB for Tomjerry image.

We plot the evolutions of the objective function value and the SNR value with respect to the number of iterations and computing time in seconds in Figure 7, for several cases of the algorithms in (5.18) with different choices of τ. The decomposed cartoon and texture parts by the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations are illustrated in Figure 8. To see the quality of decomposition more clearly, we display the images by superposing the obtained cartoon and texture components in the third column of Figure 8.

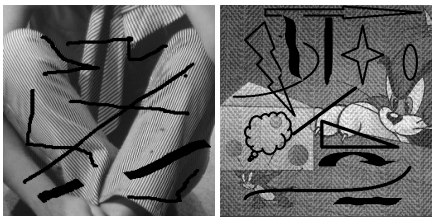

Figure 6: The case of (6.1) with $K = S$. Left: corrupted Barbara image; Right: corrupted Tomjerry image.

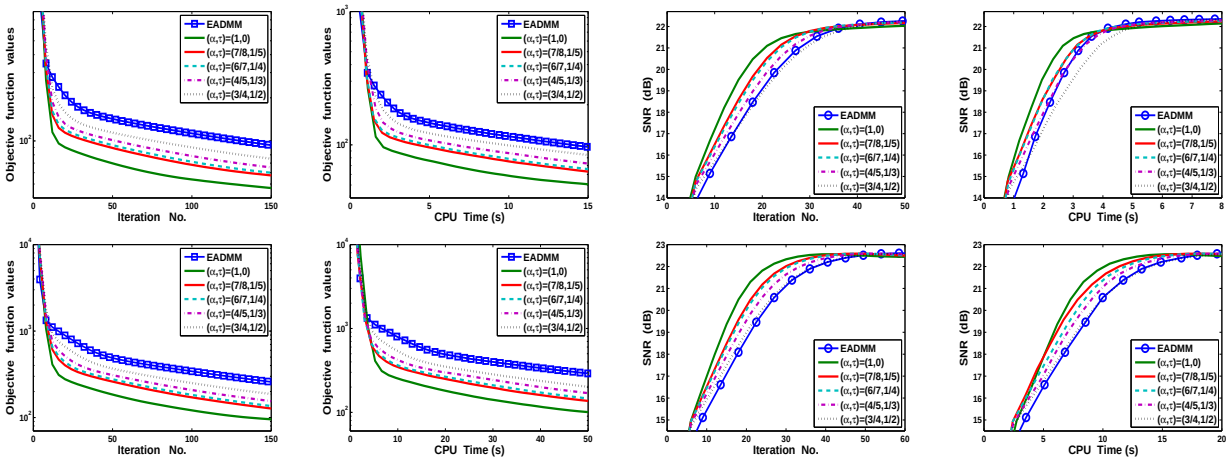

Figure 7: The implementation of (5.18) with different (τ, α) to (6.1) with $K = S$, $\beta = 1$. Evolutions of the objective function value and SNR value w.r.t. the number of iterations and computing time in seconds. Top row: for Barbara. Bottom row: for Tomjerry.

6.3.3 The Case of $K = H$

Finally, we test the case of the model (6.1) with $K = H$. That is, the image f to be decomposed is degraded by some blur. Again, for succinctness, we only test the Barbara and Boy images with out-of-focus blur as shown in Figure 9. For the corrupted images listed in Figure 9, the SNR values are 13.18dB for the corrupted Barbara image and 24.18dB for the corrupted Boy image, respectively.

For this case, the parameters (τ_1, τ_2, τ_3) in (6.2) are chosen as $(0.08, 0.005, 1)$, as suggested in [33, Theorem 3.3-3.6]. All initial iterates for implementing the algorithms in (5.18) are taken as zeros.

We plot the evolutions of the objective function value and SNR value with respect to the number of iterations and computing time in seconds in Figure 10, for several cases of the algorithms in (5.18) with different choices of τ. The decomposed cartoon and texture parts by the proposed algorithm with

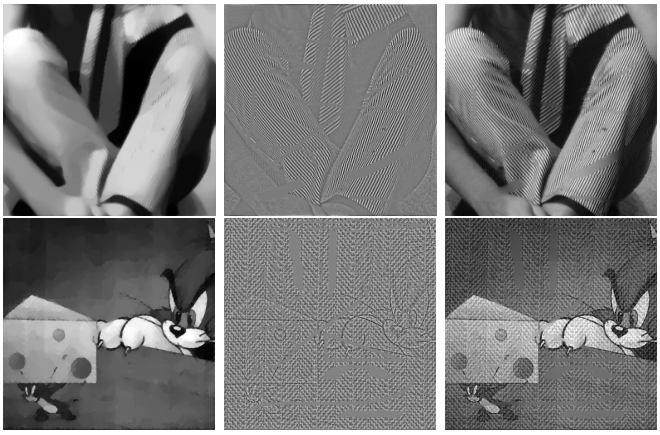

Figure 8: The decomposed cartoon (left column) and texture (middle column) parts by the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations; and the superposed (right column) images. Top row: for Barbara. Bottom row: for Tomjerry

$(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations are illustrated in Figure 11. To see the quality of decomposition more clearly, we display the images by superposing the obtained cartoon and texture components in the third column of Figure 11.

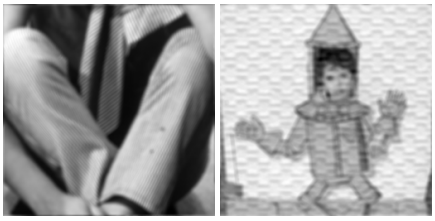

Figure 9: The case of (6.1) with $K = H$. Convolved Barbara (left) and Boy (right) images by out-of-focus blur with radius 3.

7 Conclusions

We focused on the convex minimization model with linear constraints and its objective function is the sum of three functions without coupled variables; and discussed splitting methods originated from the augmented Lagrangian method (ALM). The direct extension of the well-known alternating direction method of multipliers (ADMM), together with the ALM and the original ADMM, were discussed in the variational inequality context uniformly. It was demonstrated that the difference of these three methods in convergence proof can be explained as that the direct extension of ADMM does not satisfy an easily checkable condition which only requires to verify the symmetry and positive definiteness or semi-definiteness of two matrices. On the country, both ALM and ADMM satisfy this condition. This may be a meaningful explanation to the convergence failure of the direct extension of ADMM. We also observed the dissymmetry appearing in the direct extension of ADMM over the primal variables which are involved in its iteration, and thus proposed to refine its output by making a balance between these variables. A class of easily implementable algorithms were thus proposed with proved convergence, and we verified by numerical examples that these algorithms could be even faster than the direct extension of ADMM when it is indeed convergent. Because of its representativeness

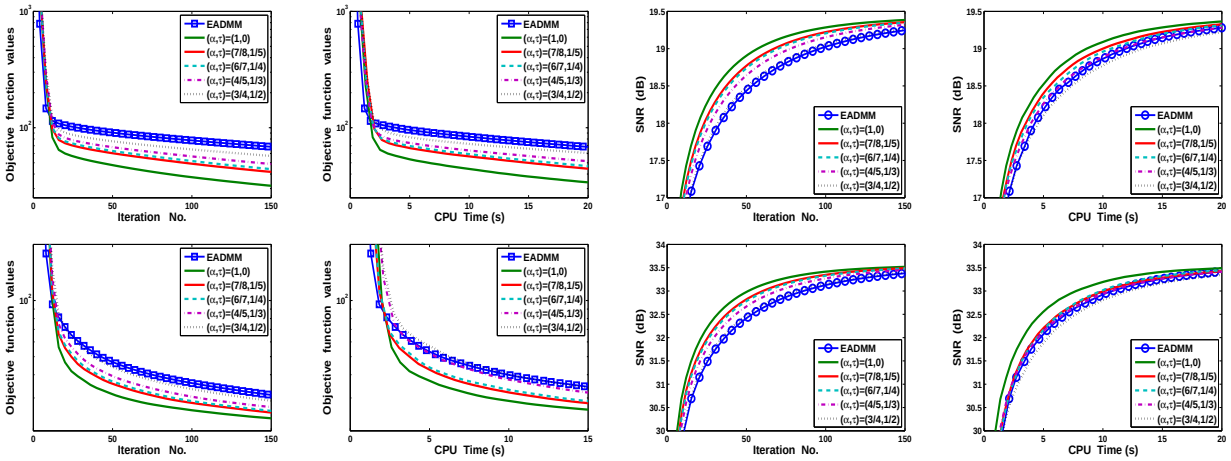

Figure 10: The implementation of (5.18) with different (τ, α) to (6.1) with $K = H$, $\beta = 1$. Evolutions of the objective function value and SNR value w.r.t. the number of iterations and computing time in seconds. Top row: for Barbara. Bottom row: for Boy.

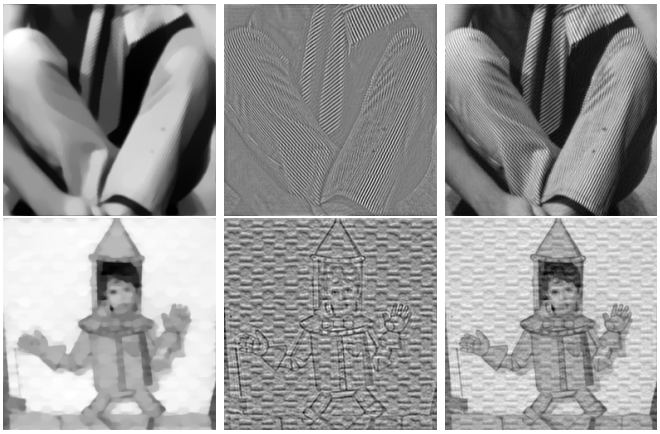

Figure 11: The decomposed cartoon (left column) and texture (middle column) parts by the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations; and the superposed (right column) images. Top row: for Barbara. Bottom row: for Boy.

and in order to exposure our idea clearly, in this paper we only focused on the convex minimization model where there are three functions in its objective. In the future, we will consider extending this work to the more general case where there are more than three functions in the objective. To some extent, the analysis will follow the analytic framework of what we have presented in this paper. But technically, the analysis for the general case is significantly more demanding and the conclusion may also be different. For example, the lower bound for the step size α in the prototype algorithm (2.5) is expected to be more conservative because the case under consideration is more general.

References

[1] J. F. Aujol, G. Gilboa, T. Chan, and S. Osher, *Structure-texture image decomposition—modeling, algorithms, and parameter selection*, Int. J. Comput. Vision, 67 (2006), pp. 111–136.

[2] E. Blum and W. Oettli, Mathematische Optimierung. Grundlagen und Verfahren. Ökonometrie und Unternehmensforschung, Springer-Verlag, Berlin-Heidelberg-New York, 1975.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 3 (2010), pp. 1-122.

[4] T. F. Chan and R. Glowinski, *Finite element approximation and iterative solution of a class of mildly non-linear elliptic equations*, technical report, Stanford University, 1978.

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

[6] C. H. Chen, B. S. He, Y. Y. Ye and X. M. Yuan, *The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent*, manuscript, 2013.

[7] J. Eckstein, *Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results*, manuscript, 2012.

[8] F. Facchinei and J. S. Pang, Finite-Dimensional Variational Inequalities and Complementarity Problems, Vol. I. Springer Series in Operations Research, Springer Verlag, New York, 2003.

[9] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Comput. Math. Appli., 2 (1976), pp. 17-40.

[10] R. Glowinski, *On alternating directon methods of multipliers: a historical perspective*, Springer Proceedings of a Conference Dedicated to J. Periaux, to appear.

[11] R. Glowinski and A. Marrocco, *Approximation par éléments finis d'ordre un et résolution par pénalisation-dualité d'une classe de problèmes non linéaires*, R.A.I.R.O., R2 (1975), pp. 41-76.

[12] D. R. Han and X. M. Yuan, *A note on the alternating direction method of multipliers*, J. Optim. Theory Appl., 155 (1) (2012), pp. 227-238.

[13] W. M. Han and B. D. Reddy, On the finite element method for mixed variational inequalities airising in elastoplasticity, SIAM J. Num. Anal., 32 (1995), pp. 1778-1807.

[14] P. Hansen, J. Nagy, and D. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, (2006).

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

[16] B. S. He, M. Tao and X. M. Yuan, *Convergence rate and iteration complexity on the alternating direction method of multipliers with a substitution procedure for separable convex programming*, Math. Oper. Res., under revision.

[17] B. S. He and X. M. Yuan, *On the O(1/n) convergence rate of Douglas-Rachford alternating direction method*, SIAM J. Num. Anal. 50 (2012), pp. 700-709.

[18] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appli., 4(1969), pp. 303-320.

[19] M. Hong and Z. Q. Luo, *On the linear convergence of the alternating direction method of multipliers*, manuscript, August 2012.

[20] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM J. Nmu. Anal., 16 (1979), pp. 964–979.

[21] G. J. McLachlan, Discriminant analysis and statistical pattern recognition, volume 544. Wiley-Interscience, 2004.

[22] Y. Meyer, *Oscillating patterns in image processing and nonlinear evolution equations*, University Lecture Series, AMS, (2002).

[23] A. S. Nemirovsky and D. B. Yudin, Problem Complexity and Method Efficiency in Optimization, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, New York, 1983.

[24] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence* $\mathcal{O}(1/k^2)$, Doklady AN SSSR, 269 (1983), pp. 543-547.

[25] M. K. Ng, P. Weiss, and X. M. Yuan, *Solving constrained total-variation problems via alternating direction methods*, SIAM J. Sci. Comput., 32 (2010), pp. 2710–2736.

[26] M. K. Ng, X. M. Yuan and W. X. Zhang, *A coupled variational image decomposition and restoration model for blurred cartoon-plus-texture images with missing pixels*, IEEE Tran. Imaging Proc., 22(2013), pp. 2233-2246.

[27] S. Osher, A. Sole, and L. Vese, *Image decomposition and restoration using total variation minimization and the* H^{-1} *norm*, Multiscale Model. Simul., 1 (2003), pp. 349–370.

[28] Y. G. Peng, A. Ganesh, J. Wright, W. L. Xu and Y. Ma, *Robust alignment by sparse and low-rank decomposition for linearly correlated images*, IEEE Tran. Pattern Anal. Mach. Intel. 34 (2012), pp. 2233-2246.

[29] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, In Optimization edited by R. Fletcher, pp. 283-298, Academic Press, New York, 1969.

[30] B. D. Reddy, Mixed variational inequalities arising in elastoplasticity, Nonlinear Anal., 19 (1992), pp. 1071-1089.

[31] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp.97-116.

[32] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D., 60 (1992), pp. 259–268.

[33] H. Schaeffer and S. Osher, *A low patch-rank interpretation of texture*, SIAM J. Imaging Sci., 6 (2013), pp. 226–262.

[34] J. Starck, M. Elad, and D. L. Donoho, *Image decomposition via the combination of sparse representations and a variational approach*, IEEE Trans. Image Process., 14 (2005), pp. 1570–1582.

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

[36] L. Vese and S. Osher, *Modeling textures with total variation minimization and oscillating patterns in image processing*, J. Sci. Comput., 19 (2003), pp. 553–572.