Comput Optim Appl (2018) 70:791–826
https://doi.org/10.1007/s10589-018-9994-1

A class of ADMM-based algorithms for three-block separable convex programming

Bingsheng He[1,2] · Xiaoming Yuan[3]

Received: 4 February 2017 / Published online: 6 March 2018

Abstract The alternating direction method of multipliers (ADMM) recently has found many applications in various domains whose models can be represented or reformulated as a separable convex minimization model with linear constraints and an objective function in sum of two functions without coupled variables. For more complicated applications that can only be represented by such a multi-block separable convex minimization model whose objective function is the sum of more than two functions without coupled variables, it was recently shown that the direct extension of ADMM is not necessarily convergent. On the other hand, despite the lack of convergence, the direct extension of ADMM is empirically efficient for many applications. Thus we are interested in such an algorithm that can be implemented as easily as the direct extension of ADMM, while with comparable or even better numerical performance and guaranteed convergence. In this paper, we suggest correcting the output of the direct extension of ADMM slightly by a simple correction step. The correction step is simple in the sense that it is completely free from step-size computing and its step size is bounded away from zero for any iterate. A prototype algorithm in this prediction-

This work was supported by the NSFC Grant 10471056, the NSFC/RGC Joint Research Scheme: N_PolyU504/14 and the General Research Fund from Hong Kong Research Grants Council: HKBU 12313516.

✉ Xiaoming Yuan
xmyuan@hku.hk

Bingsheng He
hebma@nju.edu.cn

[1] Department of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China

[2] Department of Mathematics, Nanjing University, Nanjing 210093, China

[3] Department of Mathematics, The University of Hong Kong, Hong Kong, China

correction framework is proposed; and a unified and easily checkable condition to ensure the convergence of this prototype algorithm is given. Theoretically, we show the contraction property, prove the global convergence and establish the worst-case convergence rate measured by the iteration complexity for this prototype algorithm. The analysis is conducted in the variational inequality context. Then, based on this prototype algorithm, we propose a class of specific ADMM-based algorithms that can be used for three-block separable convex minimization models. Their numerical efficiency is verified by an image decomposition problem.

Keywords Convex programming · Alternating direction method of multipliers · Splitting methods · Contraction · Convergence rate

Mathematics Subject Classification 90C25 · 90C30 · 90C33

1 Introduction

Many applications such as sparse and/or low-rank optimization, compressive sensing, statistical learning, computer vision and large-scale distributed wireless network can be modeled or reformulated as a convex minimization model with linear constraints and a separable objective function in form of the sum of more than one function. In this paper, inspired by some particular 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 [22], we consider the special separable convex minimization model

$$\min\{\theta_1(x) + \theta_2(y) + \theta_3(z) \mid Ax + By + Cz = b,\ x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}\}, \quad (1.1)$$

where $\theta_i : \Re^{n_i} \to \Re$ are closed proper convex (not necessarily smooth) functions for $i = 1, 2, 3$; $\mathcal{X} \subseteq \Re^{n_1}$, $\mathcal{Y} \subseteq \Re^{n_2}$ and $\mathcal{Z} \subseteq \Re^{n_3}$ are closed convex sets; $A \in \Re^{m\times n_1}$, $B \in \Re^{m\times n_2}$ and $C \in \Re^{m\times n_3}$; and $b \in \Re^m$. Let $n_1 + n_2 + n_3 = n$. Throughout our discussion, the solution set of (1.1) is assumed to be nonempty; and the matrices B and C are assumed to be full column rank. In addition, the sets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are assumed to be simple in sense of that the projections onto them under the Euclidean distance can be easily computed.

Conceptually, the augmented Lagrangian method (ALM) in [20,30] is applicable to (1.1), resulting in the scheme

$$\begin{cases} \begin{pmatrix} x^{k+1} \\ y^{k+1} \\ z^{k+1} \end{pmatrix} = \arg\min\left\{ \begin{array}{l} \theta_1(x)+\theta_2(y) + \theta_3(z) - (\lambda^k)^T(Ax + By + Cz - b) \\ + \frac{\beta}{2}\|Ax + By + Cz - b\|^2 \end{array} \middle| \, x\in\mathcal{X}, y\in\mathcal{Y}, z\in\mathcal{Z} \right\}, \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} + Cz^{k+1} - b), \end{cases} \quad (1.2)$$

where $\lambda \in \Re^m$ is the Lagrange multiplier associated with the linear constraints in (1.1), $\beta > 0$ is a penalty parameter and $\|\cdot\|$ represents the l_2-norm throughout.

For the direct application of ALM (1.2), each of the resulting subproblems requires minimizing an objective function consisting of all θ_i's and a quadratic term in which all the variables x, y and z are coupled. Usually, the (x, y, z)-subproblem in (1.2) is not easy to solve. We are interested in the particular scenario where each function θ_i in the objective has some particular structures and properties; and it is beneficial to explore these structures/properties in algorithmic design. For example, for some applications of the abstract model (1.1) arising in sparse- and/or low-rank-concerned areas, θ_i's are simple in the sense that the following minimization problems may have closed-form solutions for any $\beta > 0$ and $x^0 \in \Re^{n_1}$, $y^0 \in \Re^{n_2}$ and $z^0 \in \Re^{n_3}$:

$$\begin{aligned} &\min\big\{\theta_1(x) + \frac{\beta}{2}\|x - x^0\|^2\big\}, \quad \min\big\{\theta_2(y) + \frac{\beta}{2}\|y - y^0\|^2\big\} \quad \text{and} \\ &\min\big\{\theta_3(z) + \frac{\beta}{2}\|z - z^0\|^2\big\}. \end{aligned} \tag{1.3}$$

We are thus interested in such an algorithm whose subproblems are at most as difficult as

$$\begin{aligned} &\min_{x\in\mathcal{X}}\big\{\theta_1(x) + \frac{\beta}{2}\|Ax - p_1^0\|^2\big\}, \quad \min_{y\in\mathcal{Y}}\big\{\theta_2(y) + \frac{\beta}{2}\|By - p_2^0\|^2\big\} \quad \text{and} \\ &\min_{z\in\mathcal{Z}}\big\{\theta_3(z) + \frac{\beta}{2}\|Cz - p_3^0\|^2\big\} \end{aligned} \tag{1.4}$$

with $p_i^0 \in \Re^m$. Note that when the minimization problems in (1.3) have closed-form solutions, solving (1.4) could be generally easy, especially for the case that $\mathcal{X} = \Re^{n_1}$, $\mathcal{Y} = \Re^{n_2}$ and $\mathcal{Z} = \Re^{n_3}$. For instance, if $\mathcal{X} = \Re^{n_1}$, the problem (1.4) can be iteratively solved by linearizing the quadratic term in (1.4) because the linearized subproblem reduces to the problem (1.3). This is indeed an implementation of the forward-backward splitting method which was originated in [28]. To expose our main idea of algorithmic design with easier notation, we just focus on the discussion of designing an algorithm with subproblems in form of (1.4) and do not discuss its linearized counterparts whose subproblems are in form of (1.3). Thus, for the (x, y, λ)-subproblem in (1.2), we naturally want to decompose it into three smaller and easier ones in the Gauss–Seidel manner. That is, we may consider the following scheme instead of (1.2):

$$\left\{\begin{array}{ll} x^{k+1} = \text{argmin}\{\theta_1(x) - (\lambda^k)^T\big(Ax\big) + \frac{\beta}{2}\big\|Ax + By^k + Cz^k - b\big\|^2 \mid x \in \mathcal{X}\}, & (1.5\text{a}) \\ y^{k+1} = \text{argmin}\{\theta_2(y) - (\lambda^k)^T\big(By\big) + \frac{\beta}{2}\big\|Ax^{k+1} + By + Cz^k - b\big\|^2 \mid y \in \mathcal{Y}\}, & (1.5\text{b}) \\ z^{k+1} = \text{argmin}\{\theta_3(z) - (\lambda^k)^T\big(Cz\big) + \frac{\beta}{2}\big\|Ax^{k+1} + By^{k+1} + Cz - b\big\|^2 \mid z \in \mathcal{Z}\}, & (1.5\text{c}) \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} + Cz^{k+1} - b). & (1.5\text{d}) \end{array}\right.$$

Compared with the direct application of ALM (1.2), obviously the scheme (1.5) is more implementable because its subproblems are significantly easier under our mentioned simplicity assumption of θ_i's. The scheme (1.5) is indeed a direct extension of the alternating direction method of multiplier (ADMM), which was originally proposed in [12] (see also [4,10]) and recently has found many data-driven applications including those mentioned. We refer to [3,7,11] for some review papers on ADMM.

In fact, the numerical efficiency of (1.5) has been demonstrated empirically in the literature, see e.g. [29,35]. Without any further assumption, however, it was shown in [6] that the scheme (1.5) is not necessarily convergent. In [15,17], we have proposed some algorithms whose common feature is generating a new iterate by correcting the output of (1.5) with some correction steps, instead of using the output of (1.5) directly. With rigorously proved convergence, these algorithms still numerically perform less efficiently than the direct extension of ADMM (1.5) mainly because their correction steps need to determine step sizes iteratively with nonnegligible computation. The purpose of this paper is to find algorithms that could be implemented as easily as (1.5), while with comparable or even faster numerical performance and theoretically provable convergence. More specifically, because of the obvious advantage of (1.5) in exploiting the functions' structures individually and its empirical efficiency, we want to completely preserve the step of (1.5) but correct its output appropriately by a simple correction to generate a new iterate. That is, we still follow the prediction-correction algorithmic framework as those in [15,17], using the output of (1.5) as a predictor; but the correction step should be completely free from step-size computing and its step size is bounded away from zero for all iterates.

We will propose a prototype algorithm, based on which a class of specific ADMM-based algorithms for the model (1.1) can be easily obtained. A unified and easily checkable condition to ensure the convergence of this prototype algorithm will also be given. Indeed, the possible divergence of the direct extension of ADMM (1.5) can be simply illustrated as that it does not satisfy this condition. 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. The analysis is conducted in the variational inequality context. Numerically, we show the efficiency of this class of ADMM-based algorithms by an image decomposition problem. 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 some theoretical and numerical improvement over the direct extension of ADMM (1.5).

The rest of this paper is organized as follows. We recall some known results in Sect. 2 for further analysis. In Sect. 3, we propose a prototype algorithm in the context of variational inequality reformulation of (1.1); and then a sufficient condition is given to ensure the convergence of this prototype algorithm. We then show that the direct extension of ADMM (1.5) does not satisfy this condition. In Sects. 4 and 5, 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 ADMM-based specific algorithms for the model (1.1) in Sect. 6, and show their efficiency in solving some image processing applications in Sect. 7. Finally, some conclusions are made and some possible topics for future research are discussed in Sect. 8.

2 Preliminaries

In this section, we recall some known results and definitions that will be used later.

2.1 Variational inequality reformulation

It is obvious that the model (1.1) is equivalent to the variational inequality: Finding $w^* \in \Omega$ such that

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

where

$$u = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \qquad w = \begin{pmatrix} x \\ y \\ z \\ \lambda \end{pmatrix}, \qquad \theta(u) = \theta_1(x) + \theta_2(y) + \theta_3(z), \tag{2.2a}$$

$$F(w) = \begin{pmatrix} -A^T\lambda \\ -B^T\lambda \\ -C^T\lambda \\ Ax + By + Cz - b \end{pmatrix} \qquad \text{and} \qquad \Omega := \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \Re^m, \tag{2.2b}$$

with $u \in \Re^n$, $w \in \Re^{n+m}$ and $\Omega \subseteq \Re^{n+m}$. Obviously, the mapping $F(w)$ defined in (2.2b) is affine with a skew-symmetric matrix; it is thus monotone. We denote by Ω^* the solution set of $\mathrm{VI}(\Omega, F, \theta)$, and it is nonempty because of the nonemptyness of the solution set of (1.1).

Our theoretical analysis will be conducted in the context of variational inequality. This is a convenient theoretical tool making us present the analysis based on the optimality condition of the model (1.1) and enabling us to take advantage of some existing results in the literature. But we would emphasize that the variational inequality reformulation is only for analysis purpose; we do not need to solve any variational inequality to implement the algorithms to be proposed.

2.2 Definitions

For convenience, we give the following definition to differentiate the roles of different coordinates of a variable in the iteration of an algorithm, see some similar discussions in [3].

Definition 1 For an iterative algorithm solving $\mathrm{VI}(\Omega, F, \theta)$, if some coordinates of w are not involved in the iteration, then these coordinates are called intermediate variables and those required by the iteration are called essential variables (denoted by v).

Therefore, for the direct extension of ADMM (1.5), x is an intermediate variable and $v = (y, z, \lambda)$ are essential variables. Accordingly, the intention of the notation v^k,

$\tilde{v}^k$, v^*, $\mathcal{V}$ and $\mathcal{V}^*$ should be clear from the context. We thus have

$$v = (y, z, \lambda), \quad \mathcal{V} = \mathcal{Y} \times \mathcal{Z} \times \Re^m, \quad v^k = (y^k, z^k, \lambda^k), \quad \tilde{v}^k = (\tilde{y}^k, \tilde{z}^k, \tilde{\lambda}^k), \quad \forall k \in \mathcal{N};$$
$$v^* = (y^*, z^*, \lambda^*), \quad \mathcal{V}^* = \{(y^*, z^*, \lambda^*) \mid (x^*, y^*, z^*, \lambda^*) \in \Omega^*\}.$$

3 A prototype algorithm

In this section, we propose a prototype algorithm for VI(Ω, F, θ) and give a condition ensuring its convergence. With this convergence-guaranteeing condition, we also give an explanation for why the direct extension of ADMM (1.5) is not necessarily convergent.

3.1 Algorithm

Recall that 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 $\tilde{w}^k \in \Omega$ and a matrix Q satisfying

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (v - \tilde{v}^k)^T Q(v^k - \tilde{v}^k), \quad \forall\, w \in \Omega, \tag{3.1a}$$

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

$$v^{k+1} = v^k - \alpha M(v^k - \tilde{v}^k). \tag{3.1b}$$

Then, the convergence of the prototype algorithm (3.1) can be guaranteed if the following condition is fulfilled.

Convergence Condition
For the matrices Q and M, and the step size α determined in (3.1), we define two matrices H and G respectively as

$$H = QM^{-1} \tag{3.2a}$$

and

$$G = Q^T + Q - \alpha M^T HM. \tag{3.2b}$$

Then, the prototype algorithm (3.1) is convergent if both H and G are positive definite.

Remark 1 If $v^k = \tilde{v}^k$, then $\tilde{w}^k$ is a solution point of (2.1) because of (3.1a) and the definition of VI(Ω, F, θ). We thus can use $\|v^k - \tilde{v}^k\| < \epsilon$ as a stopping criterion to implement a specific algorithm of the prototype (3.1), where ϵ is a given tolerance. On the other hand, if $v^k \neq \tilde{v}^k$, then setting $w = w^*$ in (3.1a) 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

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

Using $Q = HM$ and the above inequality, we obtain

$$(v^k - v^*)^T HM(v^k - \tilde{v}^k) \geq (v^k - \tilde{v}^k)^T \left[\frac{Q^T + Q}{2}\right](v^k - \tilde{v}^k), \quad \forall\, v^* \in \mathcal{V}^*. \tag{3.3}$$

In some sense, the inequality (3.3) 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 (3.1b) can also be explained as a contraction step which moves along the direction $-M(v^k - \tilde{v}^k)$ starting from v^k towards $\mathcal{V}^*$.

3.2 The direct extension of ADMM (1.5) does not satisfy the convergence condition

Now we show that the direct extension of ADMM (1.5) is also a special case of the prototype algorithm (3.1) but the Convergence Condition is not satisfied. Thus an explanation of the convergence failure is provided for (1.5). Recall that the model (1.1) can be explained as the VI (2.1) with the specification given in (2.2).

First, the optimality condition of the iteration (1.5) can be summarized as the following VIs:

$$\left\{\begin{array}{ll}
x^{k+1} \in \mathcal{X},\ \theta_1(x) - \theta_1(x^{k+1}) + (x - x^{k+1})^T & \\
\qquad \{-A^T[\lambda^k - \beta(Ax^{k+1} + By^k + Cz^k - b)]\} \geq 0, \quad \forall x \in \mathcal{X}, & \text{(3.4a)}\\
y^{k+1} \in \mathcal{Y},\ \theta_2(y) - \theta_2(y^{k+1}) + (y - y^{k+1})^T & \\
\qquad \{-B^T[\lambda^k - \beta(Ax^{k+1} + By^{k+1} + Cz^k - b)]\} \geq 0, \quad \forall y \in \mathcal{Y}, & \text{(3.4b)}\\
z^{k+1} \in \mathcal{Z},\ \theta_3(z) - \theta_3(z^{k+1}) + (z - z^{k+1})^T & \\
\qquad \{-C^T[\lambda^k - \beta(Ax^{k+1} + By^{k+1} + Cz^{k+1} - b)]\} \geq 0,\ \forall z \in \mathcal{Z}. & \text{(3.4c)}
\end{array}\right.$$

To see why the inequalities in (3.4) can be written as a special case of (3.1a), we define $\tilde{w}^k = (\tilde{x}^k, \tilde{y}^k, \tilde{z}^k, \tilde{\lambda}^k)$ with

$$\tilde{x}^k = x^{k+1}, \qquad \tilde{y}^k = y^{k+1}, \qquad \tilde{z}^k = z^{k+1}, \tag{3.5a}$$

and

$$\tilde{\lambda}^k = \lambda^k - \beta(Ax^{k+1} + By^k + Cz^k - b), \tag{3.5b}$$

for the given (y^k, z^k, λ^k) and the $(x^{k+1}, y^{k+1}, z^{k+1})$ generated by the direct extension of ADMM (1.5). Using (3.5), we can simplify the scheme (3.4) as $(\tilde{x}^k, \tilde{y}^k, \tilde{z}^k) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$,

$$\begin{cases} \theta_1(x) - \theta_1(\tilde{x}^k) + (x - \tilde{x}^k)^T\{-A^T\tilde{\lambda}^k\} \geq 0, \quad \forall x \in \mathcal{X}, & (3.6a) \\ \theta_2(y) - \theta_2(\tilde{y}^k) + (y - \tilde{y}^k)^T\{-B^T\tilde{\lambda}^k + \beta B^T B(\tilde{y}^k - y^k)]\} \geq 0, \quad \forall y \in \mathcal{Y}, & (3.6b) \\ \theta_3(z) - \theta_3(\tilde{z}^k) + (z - \tilde{z}^k)^T\{-C^T\tilde{\lambda}^k & \\ \quad +\beta C^T[B(\tilde{y}^k - y^k) + C(\tilde{z}^k - z^k)]\} \geq 0, \ \forall z \in \mathcal{Z}. & (3.6c) \end{cases}$$

We now show in the next lemma that the inequalities in (3.6) are indeed a specific implementation of the prototype algorithm (3.1). Recall that the essential variable of the scheme (1.5) is $v = (y, z, \lambda)$.

Lemma 1 *Let $u^{k+1} = (x^{k+1}, y^{k+1}, z^{k+1})$ be generated by the direct extension of ADMM (1.5) from the given $v^k = (y^k, z^k, \lambda^k)$; and $\tilde{w}^k$ be defined in (3.5). Then we have*

$$\begin{aligned} &\tilde{w}^k \in \Omega, \quad \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (v - \tilde{v}^k)^T Q(v^k - \tilde{v}^k), \\ &\forall\, w \in \Omega, \end{aligned} \tag{3.7}$$

where

$$Q = \begin{pmatrix} \beta B^T B & 0 & 0 \\ \beta C^T B & \beta C^T C & 0 \\ -B & -C & \frac{1}{\beta} I \end{pmatrix}. \tag{3.8}$$

Proof Using $\tilde{x}^k = x^{k+1}$, $\tilde{y}^k = y^{k+1}$ and $\tilde{z}^k = z^{k+1}$, and the Eq. (3.5b), we have

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

which can be rewritten as

$$\begin{aligned} &\tilde{\lambda}^k \in \Re^m, \quad (\lambda - \tilde{\lambda}^k)^T\{(A\tilde{x}^k + B\tilde{y}^k + C\tilde{z}^k - b) - B(\tilde{y}^k - y^k) - C(\tilde{z}^k - z^k) \\ &\quad + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k)\} \geq 0, \quad \forall \lambda \in \Re^m. \end{aligned}$$

Combining (3.6) with the last inequality together, we get

$$\tilde{w}^k \in \Omega, \quad \theta(u) - \theta(\tilde{u}^k) + \begin{pmatrix} x - \tilde{x}^k \\ y - \tilde{y}^k \\ z - \tilde{z}^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \left\{ \begin{pmatrix} -A^T\tilde{\lambda}^k \\ -B^T\tilde{\lambda}^k \\ -C^T\tilde{\lambda}^k \\ A\tilde{x}^k + B\tilde{y}^k + C\tilde{z}^k - b \end{pmatrix} \right.$$
$$\left. + \begin{pmatrix} 0 \\ \beta B^T B(\tilde{y}^k - y^k) \\ \beta C^T B(\tilde{y}^k - y^k) + \beta C^T C(\tilde{z}^k - z^k) \\ -B(\tilde{y}^k - y^k) - C(\tilde{z}^k - z^k) + \frac{1}{\beta}(\tilde{\lambda}^k - \lambda^k) \end{pmatrix} \right\} \geq 0, \qquad \forall w \in \Omega. \tag{3.9}$$

Based on the above inequality and using the notation $F(w)$ (see (2.2)) and matrix Q (see (3.8)), the assertion of this lemma is proved. □

Now, we show that the direct extension of ADMM (1.5) can be written as a special case of the prototype algorithm (3.1). With the definition Q in (3.8), we have

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

which means Q is positive definite whenever B and C are both full column rank. Moreover, using (3.5), the λ^{k+1} updated by (1.5d) can be represented as

$$\begin{aligned} \lambda^{k+1} &= \lambda^k - \beta(A\tilde{x}^k + B\tilde{y}^k + C\tilde{z}^k - b) \\ &= \lambda^k - \left[(\lambda^k - \tilde{\lambda}^k) - \beta B(y^k - \tilde{y}^k) - \beta C(z^k - \tilde{z}^k)\right]. \end{aligned} \tag{3.10}$$

Therefore, using (3.5), we can rewrite the direct extension of ADMM (1.5) as

$$\begin{pmatrix} y^{k+1} \\ z^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ z^k \\ \lambda^k \end{pmatrix} - \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\beta B & -\beta C & I \end{pmatrix} \begin{pmatrix} y^k - \tilde{y}^k \\ z^k - \tilde{z}^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix},$$

which corresponds to the step (3.1b) with

$$M = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\beta B & -\beta C & I \end{pmatrix} \quad \text{and} \quad \alpha = 1. \tag{3.11}$$

Indeed, the direct extension of ADMM (1.5) can be written in form of the prototype algorithm (3.1).

Now we demonstrate that the Convergence Condition is not satisfied by the direct extension of ADMM (1.5). In fact, for the matrix M in (3.11), we have

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

Thus, with easy algebra, we know

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

which is not symmetric. In addition, we have

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

which is not positive definite even if B and C are both full column rank. Therefore, the Convergence Condition is not satisfied by the direct extension of ADMM (1.5). This may explain the possible failure in convergence for the direct extension of ADMM (1.5).

4 Convergence

In this section, we show the contraction property for the sequence generated by the prototype algorithm (3.1) and then prove its global convergence. Recall the iteration of (3.1) 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 use the notation $\|v\|_G^2 := v^T G v$ where G is a positive definite matrix.

Lemma 2 *For the sequence generated by the prototype algorithm* (3.1) *where the Convergence Condition is satisfied. We have*

$$\alpha\big\{\theta(u)-\theta(\tilde{u}^k)+(w-\tilde{w}^k)^T F(\tilde{w}^k)\big\} \ge \frac{1}{2}\big(\|v-v^{k+1}\|_H^2-\|v-v^k\|_H^2\big) + \frac{\alpha}{2}\|v^k-\tilde{v}^k\|_G^2,\ \forall w\in\Omega, \tag{4.1}$$

where H and G are defined in (3.2).

Proof Using $Q = HM$ (see (3.2a)) and the update form (3.1b), the right-hand side of (3.1a) can be written as

$$\tilde{w}^k\in\Omega,\ \ \alpha\big\{\theta(u)-\theta(\tilde{u}^k)+(w-\tilde{w}^k)^T F(\tilde{w}^k)\big\} \ge (v-\tilde{v}^k)^T H(v^k-v^{k+1}),\ \ \forall w\in\Omega. \tag{4.2}$$

Applying the identity

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

to the right-hand side of (4.2) with

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

we thus obtain

$$(v-\tilde{v}^k)^T H(v^k-v^{k+1}) = \frac{1}{2}\big(\|v-v^{k+1}\|_H^2-\|v-v^k\|_H^2\big) + \frac{1}{2}(\|v^k-\tilde{v}^k\|_H^2-\|v^{k+1}-\tilde{v}^k\|_H^2). \tag{4.3}$$

For the last term of (4.3), we have

$$\begin{aligned}
&\|v^k-\tilde{v}^k\|_H^2-\|v^{k+1}-\tilde{v}^k\|_H^2\\
&\quad=\|v^k-\tilde{v}^k\|_H^2-\|(v^k-\tilde{v}^k)-(v^k-v^{k+1})\|_H^2\\
&\quad\overset{(3.1b)}{=}\|v^k-\tilde{v}^k\|_H^2-\|(v^k-\tilde{v}^k)-\alpha M(v^k-\tilde{v}^k)\|_H^2\\
&\quad=2\alpha(v^k-\tilde{v}^k)^T HM(v^k-\tilde{v}^k)-\alpha^2(v^k-\tilde{v}^k)^T M^T HM(v^k-\tilde{v}^k)\\
&\quad\overset{(3.2a)}{=}\alpha(v^k-\tilde{v}^k)^T(Q^T+Q-\alpha M^T HM)(v^k-\tilde{v}^k)\\
&\quad\overset{(3.2b)}{=}\alpha\|v^k-\tilde{v}^k\|_G^2.
\end{aligned}\tag{4.4}$$

Substituting (4.3) and (4.4) in (4.2), the assertion of this theorem is proved. □

Based on Lemma 2, we can prove the contraction property for the sequence $\{v^k\}$ generated by the prototype algorithm (3.1).

Theorem 1 *For the sequence $\{v^k\}$ generated by the prototype algorithm (3.1) where the Convergence Condition is satisfied, we have*

$$\|v^{k+1} - v^*\|_H^2 \le \|v^k - v^*\|_H^2 - \alpha\|v^k - \tilde{v}^k\|_G^2, \quad \forall v^* \in \mathcal{V}^*. \tag{4.5}$$

Proof Setting $w = w^*$ in (4.1), we get

$$\begin{aligned}\|v^k - v^*\|_H^2 - \|v^{k+1} - v^*\|_H^2 \ge \alpha\|v^k - \tilde{v}^k\|_G^2 + 2\alpha\big\{\theta(\tilde{u}^k) - \theta(u^*)\\ +(\tilde{w}^k - w^*)^T F(\tilde{w}^k)\big\}.\end{aligned} \tag{4.6}$$

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

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

The assertion (4.5) follows from (4.6) and the above fact directly. □

Recall that the matrix G defined in (3.2b) is assumed to be positive definite. It follows from (4.5) that the sequence $\{v^k\}$ is contractive with respect to $\mathcal{V}^*$. The convergence of $\{v^k\}$ generated by the prototype algorithm (3.1) thus follows the standard framework of contraction methods, see e.g. [2]. In fact, in Section 5.1, its convergence is also implied. In the following, for completeness, we still include the detail for the case $G \succ 0$.

Theorem 2 *For the sequence $\{v^k\}$ generated by the prototype algorithm (3.1) where the Convergence Condition is satisfied with $G \succ 0$, it converges to some v^∞ which belongs to $\mathcal{V}^*$.*

Proof According to (4.5), it holds that $\{v^k\}$ is bounded and

$$\lim_{k\to\infty} \|v^k - \tilde{v}^k\|_G = 0. \tag{4.7}$$

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 (3.1a) that

$$\begin{aligned}&\tilde{w}^{k_j} \in \Omega, \quad \theta(u) - \theta(\tilde{u}^{k_j}) + (w - \tilde{w}^{k_j})^T F(\tilde{w}^{k_j}) \ge (v - v^{k_j})^T Q(v^{k_j} - \tilde{v}^{k_j}),\\ &\forall\, w \in \Omega.\end{aligned}$$

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

$$w^\infty \in \Omega, \quad \theta(u) - \theta(u^\infty) + (w - w^\infty)^T F(w^\infty) \ge 0, \quad \forall\, w \in \Omega.$$

The above variational inequality indicates that w^∞ is a solution point of VI(Ω, F, θ). Moreover, it follows from (4.7) and the fact $\lim_{j\to\infty} v^{k_j} = v^\infty$ that the subsequence $\{v^{k_j}\}$

also converges to v^∞. Finally, because of (4.5), we have

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

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

5 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 (3.1), where t is the iteration counter. Note that we follow the work [24,25] 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 that the accuracy to a solution point 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. Lemma 2 is again crucial for the analysis.

5.1 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 (3.1) under the Convergence Condition.

Working on the reformulation VI(Ω, F, θ) of the model (1.1), one advantage is that a characterization of its solution set proposed in [9] becomes useful for establishing the worst-case convergence rate in the ergodic sense for the prototype algorithm (3.1).

Theorem 3 *The solution set of VI(Ω, F, θ) is convex and it can be characterized as*

$$\Omega^* = \bigcap_{w\in\Omega} \left\{\tilde{w} \in \Omega : \theta(u) - \theta(\tilde{u}) + (w - \tilde{w})^T F(w) \ge 0\right\}. \tag{5.1}$$

Proof The proof is an incremental extension of Theorem 2.3.5 in [9], or see the the proof of Theorem 2.1 in [18]. □

Theorem 3 thus implies that $\tilde{w} \in \Omega$ is an approximate solution of VI(Ω, F, θ) with an accuracy of $\epsilon > 0$ if it satisfies

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

where

$$\mathcal{D}(\tilde{u}) = \{u \mid \|u - \tilde{u}\| \le 1\}.$$

Now, we show that after t iterations of the prototype algorithm (3.1), we can find $\tilde{w} \in \Omega$ such that

$$\theta(\tilde{u}) - \theta(u) + (\tilde{w} - w)^T F(w) \le \epsilon, \quad \forall w \in \Omega \cap \mathcal{D}(\tilde{u}), \tag{5.2}$$

with $\epsilon = O(1/t)$, a worst-case $O(1/t)$ convergence rate is thus established for the prototype algorithm (3.1).

Prior to the proof, it is easy to see 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 the matrix G defined in (3.2b) is positive definite. Substituting it in (4.1), we obtain

$$\begin{aligned}\tilde{w}^k \in \Omega,\ \ \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(w) + \frac{1}{2\alpha}\|v - v^k\|_H^2 \\ \geq \frac{1}{2\alpha}\|v - v^{k+1}\|_H^2,\ \forall w \in \Omega.\end{aligned} \tag{5.3}$$

Theorem 4 *For the sequence generated by the prototype algorithm* (3.1) *where the Convergence Condition is satisfied and any integer $t > 0$, we have*

$$\theta(\bar{u}_t) - \theta(u) + (\bar{w}_t - w)^T F(w) \leq \frac{1}{2(t+1)\alpha}\|v - v^0\|_H^2, \quad \forall w \in \Omega, \tag{5.4}$$

where

$$\bar{w}_t = \frac{1}{t+1}\sum_{k=0}^{t} \tilde{w}^k. \tag{5.5}$$

Proof First, because of (5.3), it holds that $\tilde{w}^k \in \Omega$ for all $k \geq 0$. Together with the convexity of Ω, (5.5) implies that $\bar{w}_t \in \Omega$. Summing the inequality (5.3) over $k = 0, 1, \ldots, t$, we obtain

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

Using the notation of $\bar{w}_t$, we can rewrite the last inequality as

$$\frac{1}{t+1}\sum_{k=0}^{t}\theta(\tilde{u}^k) - \theta(u) + (\bar{w}_t - w)^T F(w) \leq \frac{1}{2(t+1)\alpha}\|v - v^0\|_H^2, \quad \forall w \in \Omega. \tag{5.6}$$

Since $\theta(u)$ is convex and

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

we have that

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

Substituting it in (5.6), the assertion of this theorem follows directly. □

Therefore, because of (5.2), the conclusion (5.4) indicates that the prototype algorithm (3.1) is able to generate an approximate solution point (i.e., $\tilde{w}_t$) of VI(Ω, F, θ) 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 (3.1).

5.2 Convergence rate in a nonergodic sense

In this subsection, we show that if the matrix G defined in (3.2b) is positive definite, a worst-case $O(1/t)$ convergence rate in a nonergodic sense can also be established for the prototype algorithm (3.1). Note that a nonergodic convergence rate is generally stronger than an ergodic convergence rate. We first need to prove the following lemma.

Lemma 3 *For the sequence generated by the prototype algorithm* (3.1) *where the Convergence Condition is satisfied, we have*

$$\begin{aligned}&\alpha(v^k-\tilde{v}^k)^T M^T H M\{(v^k-\tilde{v}^k)-(v^{k+1}-\tilde{v}^{k+1})\}\\ &\quad \ge \frac{1}{2}\|(v^k-\tilde{v}^k)-(v^{k+1}-\tilde{v}^{k+1})\|^2_{(Q^T+Q)}.\end{aligned} \tag{5.7}$$

Proof First, setting $w=\tilde{w}^{k+1}$ in (3.1a), we have

$$\theta(\tilde{u}^{k+1})-\theta(\tilde{u}^k)+(\tilde{w}^{k+1}-\tilde{w}^k)^T F(\tilde{w}^k) \ge (\tilde{v}^{k+1}-\tilde{v}^k)^T Q(v^k-\tilde{v}^k). \tag{5.8}$$

Note that (3.1a) is also true for $k := k+1$. Thus, we also have

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

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

$$\theta(\tilde{u}^k)-\theta(\tilde{u}^{k+1})+(\tilde{w}^k-\tilde{w}^{k+1})^T F(\tilde{w}^{k+1}) \ge (\tilde{v}^k-\tilde{v}^{k+1})Q(v^{k+1}-\tilde{v}^{k+1}). \tag{5.9}$$

Adding (5.8) and (5.9) and using the monotonicity of F, we get

$$(\tilde{v}^k-\tilde{v}^{k+1})^T Q\{(v^k-\tilde{v}^k)-(v^{k+1}-\tilde{v}^{k+1})\} \ge 0. \tag{5.10}$$

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 both sides of (5.10), and using $v^T Qv = \frac{1}{2} v^T (Q^T + Q)v$, we get

$$\begin{aligned} &(v^k - v^{k+1})^T Q\{(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\} \\ &\quad\ge \frac{1}{2}\|(v^k - \tilde{v}^k) - (v^{k+1} - \tilde{v}^{k+1})\|^2_{(Q^T+Q)}. \end{aligned}$$

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 (5.7) and the lemma is proved. □

Now, we are ready to prove (5.11), the key inequality in this section.

Theorem 5 *For the sequence generated by the prototype algorithm* (3.1) *where the Convergence Condition is satisfied, we have*

$$\|M(v^{k+1} - \tilde{v}^{k+1})\|_H \le \|M(v^k - \tilde{v}^k)\|_H, \quad \forall k > 0. \tag{5.11}$$

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

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

we obtain

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

Inserting (5.7) into the first term of the right-hand side of the last equality, we obtain

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

where the last inequality is because of the positive definiteness of the matrix $(Q^T + Q) - \alpha M^T HM \succ 0$. The assertion (5.11) follows immediately. □

Note that it follows from $G \succ 0$ and Theorem 1 there is a constant $c_0 > 0$ such that

$$\|v^{k+1} - v^*\|^2_H \le \|v^k - v^*\|^2_H - c_0\|M(v^k - \tilde{v}^k)\|^2_H, \quad \forall v^* \in \mathcal{V}^*. \tag{5.12}$$

Now, with (5.12) and (5.11), we can establish the worst-case $O(1/t)$ convergence rate in a nonergodic sense for the prototype algorithm (3.1).

Theorem 6 *Let $\{v^k\}$ and $\{\tilde{w}^k\}$ be the sequences generated by the prototype algorithm (3.1) under the Convergence Condition. For any integer $t > 0$, we have*

$$\|M(v^t - \tilde{v}^t)\|_H^2 \le \frac{1}{(t+1)c_0}\|v^0 - v^*\|_H^2. \tag{5.13}$$

Proof First, it follows from (5.12) that

$$\sum_{k=0}^{\infty} c_0\|M(v^k - \tilde{v}^k)\|_H^2 \le \|v^0 - v^*\|_H^2, \quad \forall\, v^* \in \mathcal{V}^*. \tag{5.14}$$

According to Theorem 5, the sequence $\{\|M(v^k - \tilde{v}^k)\|_H^2\}$ is monotonically non-increasing. Therefore, we have

$$(t+1)\|M(v^t - \tilde{v}^t)\|_H^2 \le \sum_{k=0}^{t} \|M(v^k - \tilde{v}^k)\|_H^2. \tag{5.15}$$

The assertion (5.13) follows from (5.14) and (5.15) immediately. □

Notice that $\mathcal{V}^*$ is convex and closed (see Theorem 3). Let $d := \inf\{\|v^0 - v^*\|_H \mid v^* \in \mathcal{V}^*\}$. Then, for any given $\epsilon > 0$, Theorem 6 shows that it needs at most $\lfloor d^2/c_0\epsilon \rfloor$ iterations to ensure that $\|M(v^k - \tilde{v}^k)\|_H^2 \le \epsilon$. Recall that a solution point of $\mathrm{VI}(\Omega, F, \theta)$ is found if $\|M(v^k - \tilde{v}^k)\|_H^2 = 0$ (see (3.7) and due to $Q = HM$). A worst-case $O(1/t)$ convergence rate in a nonergodic sense is thus established for the prototype algorithm (3.1).

6 A class of ADMM-based algorithms for (1.1)

Now we present a class of specific ADMM-based algorithms for model (1.1) based on the prototype algorithm (3.1). Other algorithms can also be generated if the matrix M and the scalar α are chosen otherwise in (3.1).

As mentioned, because of the verified efficiency of the direct extension of ADMM (1.5) in the literature, we stick to correcting the output of (1.5) via some correction step in algorithmic design. Thus, the step (3.1a) is exactly the direct extension of ADMM (1.5), meaning the matrix Q in (3.1a) can be chosen as the matrix defined in (3.8). In this case, $\tilde{w}^k$ is defined by (3.5). For the step (3.1b), we suggest choosing

$$M := M(\tau) = \begin{pmatrix} I & -(1-\tau)(B^TB)^{-1}B^TC & 0 \\ \tau(C^TC)^{-1}C^TB & I & 0 \\ -\beta B & -\beta C & I \end{pmatrix} \text{ with } \tau \in [0, 1]. \tag{6.1}$$

Then, with different choices of τ, a class of specific algorithms for solving (1.1) 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 (3.8) and (6.1), respectively. Also, we have to specify how to choose the constant step size α for the step (3.1b). For convenience, we define the left-upper sub-matrix of Q by Q_0, i.e.,

$$Q = \begin{pmatrix} Q_0 & & 0 \\ -B & -C & \frac{1}{\beta} I \end{pmatrix} \quad \text{and thus} \quad Q_0 = \begin{pmatrix} \beta B^T B & 0 \\ \beta C^T B & \beta C^T C \end{pmatrix}. \tag{6.2}$$

Theorem 7 *Let the matrices Q and M be given by* (3.8) *and* (6.1)*, respectively. Let $H = QM^{-1}$ as defined in* (3.2a)*. Then, for $\tau \in [0, 1]$, we have*

$$H = \begin{pmatrix} H_0 & 0 \\ 0 & \frac{1}{\beta} I \end{pmatrix}, \tag{6.3}$$

where

$$H_0 = \left[\tau D_0^{-1} + (1-\tau) Q_0^{-T} D_0 Q_0^{-1}\right]^{-1}, \tag{6.4}$$

Q_0 is defined in (6.2) *and*

$$D_0 = \begin{pmatrix} \beta B^T B & 0 \\ 0 & \beta C^T C \end{pmatrix}. \tag{6.5}$$

In addition, the matrixes H_0 and H are both symmetric and positive definite.

Proof For the matrix M defined in (6.1), we define its left-upper sub-matrix by M_0, i.e.,

$$M = \begin{pmatrix} & M_0 & 0 \\ -\beta B & -\beta C & I \end{pmatrix},$$

where

$$M_0 = \begin{pmatrix} I & -(1-\tau)(B^T B)^{-1} B^T C \\ \tau (C^T C)^{-1} C^T B & I \end{pmatrix}. \tag{6.6}$$

Indeed,

$$HM = \begin{pmatrix} H_0 & 0 \\ 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} & M_0 & 0 \\ -\beta B & -\beta C & I \end{pmatrix} = \begin{pmatrix} & H_0 M_0 & 0 \\ -B & -C & \frac{1}{\beta} I \end{pmatrix}.$$

According to (6.2), in order to show $HM = Q$, we need only to verify that

$$H_0 M_0 = Q_0. \tag{6.7}$$

Note that (see (6.6))

$$M_0 = \tau \begin{pmatrix} I & 0 \\ (C^T C)^{-1} C^T B & I \end{pmatrix} + (1-\tau) \begin{pmatrix} I & -(B^T B)^{-1} B^T C \\ 0 & I \end{pmatrix}. \tag{6.8}$$

Observing the two parts in (6.8), the first part

$$\begin{pmatrix} I & 0 \\ (C^T C)^{-1} C^T B & I \end{pmatrix} = \begin{pmatrix} \beta B^T B & 0 \\ 0 & \beta C^T C \end{pmatrix}^{-1} \begin{pmatrix} \beta B^T B & 0 \\ \beta C^T B & \beta C^T C \end{pmatrix} = D_0^{-1} Q_0, \tag{6.9}$$

while the second part is

$$\begin{aligned} \begin{pmatrix} I & -(B^T B)^{-1} B^T C \\ 0 & I \end{pmatrix} &= \begin{pmatrix} I & (B^T B)^{-1} B^T C \\ 0 & I \end{pmatrix}^{-1} \\ &= \left[\begin{pmatrix} (\beta B^T B)^{-1} & 0 \\ 0 & (\beta C^T C)^{-1} \end{pmatrix} \begin{pmatrix} \beta B^T B & \beta B^T C \\ 0 & \beta C^T C \end{pmatrix} \right]^{-1} \\ &= (D_0^{-1} Q_0^T)^{-1} = Q_0^{-T} D_0. \end{aligned} \tag{6.10}$$

Substituting (6.9) and (6.10) into (6.8), we obtain

$$M_0 = \tau D_0^{-1} Q_0 + (1-\tau) Q_0^{-T} D_0 = \big(\tau D_0^{-1} + (1-\tau) Q_0^{-T} D_0 Q_0^{-1}\big) Q_0. \tag{6.11}$$

Indeed, using (6.4) and the last equation, we have $M_0 = H_0^{-1} Q_0$ and thus (6.7) is verified. Note that

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

which is a convex combination of the matrices D_0^{-1} and $Q_0^{-T} D_0 Q_0^{-1}$. Recall that the coefficient matrices B and C are assumed to be full column rank. Thus, both D_0^{-1} and $Q_0^{-T} D_0 Q_0^{-1}$ are positive definite; and the symmetry and positive definiteness of H_0 follows immediately. Consequently, it follows from the definition of H in (6.3) that H is positive definite. The proof is complete. □

Remark 2 Note that the coefficient matrices B and C are assumed to be full column rank throughout our discussion. This assumption holds for most of the applications of (1.1) found in the literature, such as those in [5,22,35] whose coefficient matrices are actually all identity matrices. Even without this assumption, the convergence analysis can be modified slightly and be representable in terms of the sequence $\{By^k, Cz^k, \lambda^k\}$ instead of $\{y^k, z^k, \lambda^k\}$, see, e.g., [8,16], for similar discussions.

Now, let us discuss how to specify the step size α and ensure the positive definiteness of the G in (3.2b). Since both $Q^T + Q$ and H are positive definite, we can define

$$\alpha(\tau) := \arg\max\{\alpha \mid Q^T + Q - \alpha M^T H M \succeq 0\}, \tag{6.12}$$

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

Lemma 4 *Let the matrices Q and M be given by* (3.8) *and* (6.1)*, respectively, and $H = QM^{-1}$. Then for any $\alpha > 0$, we have*

$$\begin{aligned} &Q^T + Q - \alpha M^T HM \\ &= \begin{pmatrix} 2(1-\alpha)\beta B^T B - \alpha\tau\beta B^T C(C^T C)^{-1} C B^T & (1-\alpha(1+\tau))\beta B^T C & -(1-\alpha)B^T \\ (1-\alpha(1+\tau))\beta C^T B & 2(1-\alpha)\beta C^T C & -(1-\alpha)C^T \\ -(1-\alpha)B & -(1-\alpha)C & \frac{2-\alpha}{\beta} I \end{pmatrix}. \end{aligned} \tag{6.13}$$

Proof It follows from the definition of Q in (3.8) that

$$Q^T + Q = \begin{pmatrix} 2\beta B^T B & \beta B^T C & -B^T \\ \beta C^T B & 2\beta C^T C & -C^T \\ -B & -C & \frac{2}{\beta} I \end{pmatrix}. \tag{6.14}$$

Since $Q = HM$, we have

$$\begin{aligned} M^T HM &= Q^T M \\ &= \begin{pmatrix} \beta B^T B & \beta B^T C & -B^T \\ 0 & \beta C^T C & -C^T \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} I & -(1-\tau)(B^T B)^{-1} B^T C & 0 \\ \tau (C^T C)^{-1} C^T B & I & 0 \\ -\beta B & -\beta C & I \end{pmatrix} \\ &= \begin{pmatrix} 2\beta B^T B + \tau\beta B^T C(C^T C)^{-1} C B^T & (1+\tau)\beta B^T C & -B^T \\ (1+\tau)\beta C^T B & 2\beta C^T C & -C^T \\ -B & -C & \frac{1}{\beta} I \end{pmatrix}. \end{aligned} \tag{6.15}$$

Using (6.14) and (6.15), we get (6.13) and the assertion is proved. □

Theorem 8 *Let the matrices Q and M be given by* (3.8) *and* (6.1)*, respectively, and $H = QM^{-1}$. Then*

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

$$G = Q^T + Q - \alpha M^T HM \begin{cases} \succeq 0, & \textit{for } \ \alpha = 1; \\ \succ 0, & \forall\, \alpha \in (0, 1). \end{cases} \tag{6.16}$$

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

$$G = Q^T + Q - \alpha M^T HM \succ 0, \ \ \textit{for } \ \alpha = \frac{1}{1+\tau}. \tag{6.17}$$

Proof If $\tau = 0$, it follows from (6.13) that

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

Since

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

the assertion (6.16) is proved. Now, we turn to prove the second part. Indeed, setting $\alpha = 1/(1+\tau)$ in (6.13), we obtain

$$\begin{aligned} G &= \begin{pmatrix} \frac{2\tau}{1+\tau}\beta B^T B - \frac{\tau}{1+\tau}\beta B^T C (C^T C)^{-1} C B^T & 0 & -\frac{\tau}{1+\tau} B^T \\ 0 & \frac{2\tau}{1+\tau}\beta C^T C & -\frac{\tau}{1+\tau} C^T \\ -\frac{\tau}{1+\tau} B & -\frac{\tau}{1+\tau} C & \frac{(1+2\tau)}{(1+\tau)\beta} I \end{pmatrix} \\ &= \frac{\beta}{1+\tau} \begin{pmatrix} B^T & 0 & 0 \\ 0 & C^T & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} 2\tau I - \tau C (C^T C)^{-1} C^T & 0 & -\tau I \\ 0 & 2\tau I & -\tau I \\ -\tau I & -\tau I & (1+2\tau) I \end{pmatrix} \begin{pmatrix} B & 0 & 0 \\ 0 & C & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}. \end{aligned} \tag{6.18}$$

Obviously, it holds that $I - C(C^T C)^{-1} C^T \succeq 0$. Thus, we have

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

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

Theorems 7 and 8 mean the Convergence Condition is satisfied. Hence, the prototype algorithm (3.1) with Q in (3.1a) and M in (6.1) is convergent.

The proof of Theorems 7 and 8 also gives us the guidance of how to choose the step size α for the step (3.1b). In fact, for any fixed $\tau \in (0, 1]$, according to (6.12) and (6.17), 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 $\tau \in (0, 1]$. In fact, for some special values of τ, by a careful manipulation we can get tighter lower bounds of $\alpha(\tau)$. Note that we have

$$\begin{aligned} &Q^T + Q - \alpha M^T HM \\ &\quad = \beta \begin{pmatrix} B^T & 0 & 0 \\ 0 & C^T & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix} \begin{pmatrix} (2(1-\alpha)-\alpha\tau)I & (1-\alpha(1+\tau))I & -(1-\alpha)I \\ (1-\alpha(1+\tau))I & 2(1-\alpha)I & -(1-\alpha)I \\ -(1-\alpha)I & -(1-\alpha)I & (1+2\tau)I \end{pmatrix} \begin{pmatrix} B & 0 & 0 \\ 0 & C & 0 \\ 0 & 0 & \frac{1}{\beta} I \end{pmatrix}. \end{aligned}$$

In order to ensure $Q^T + Q - \alpha M^T HM \succ 0$, 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.

Now we are ready to present a class of specific ADMM-based algorithms for (1.1) based on the prototype algorithm (3.1).

A Class of ADMM-based Algorithms for (1.1)

[Step 1.] With the given $v^k = (y^k, z^k, \lambda^k)$, generate a vector $\tilde{w}^k \in \Omega$ via

$$\begin{cases} \tilde{x}^k = \operatorname{argmin}\{\theta_1(x) - \lambda^T(Ax) + \frac{\beta}{2}\|Ax + By^k + Cz^k - b\|^2 \mid x \in \mathcal{X}\}, & (6.19\text{aa}) \\ \tilde{y}^k = \operatorname{argmin}\{\theta_2(y) - \lambda^T(By) + \frac{\beta}{2}\|A\tilde{x}^k + By + Cz^k - b\|^2 \mid y \in \mathcal{Y}\}, & (6.19\text{ab}) \\ \tilde{z}^k = \operatorname{argmin}\{\theta_3(z) - \lambda^T(Cz) + \frac{\beta}{2}\|A\tilde{x}^k + B\tilde{y}^k + Cz - b\|^2 \mid z \in \mathcal{Z}\}, & (6.19\text{ac}) \\ \tilde{\lambda}^k = \lambda^k - \beta(A\tilde{x}^k + B\tilde{y}^k + C\tilde{z}^k - b). & (6.19\text{ad}) \end{cases}$$

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

$$\begin{pmatrix} y^{k+1} \\ z^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ z^k \\ \lambda^k \end{pmatrix} - \alpha \begin{pmatrix} I & -(1-\tau)(B^TB)^{-1}B^TC & 0 \\ \tau(C^TC)^{-1}C^TB & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} y^k - \tilde{y}^k \\ z^k - \tilde{z}^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}, \tag{6.19b}$$

where $\tau \in [0, 1]$, $\alpha \in (0, \alpha(\tau))$ and $\alpha(\tau)$ is defined in (6.12).

Remark 3 As mentioned, for many applications of (1.1), the coefficient matrices are identity matrices. Indeed, when $B = C = I_{m\times m}$, the step (6.19b) can be further simplified as

$$\begin{pmatrix} y^{k+1} \\ z^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ z^k \\ \lambda^k \end{pmatrix} - \alpha \begin{pmatrix} I & -(1-\tau)I & 0 \\ \tau I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} y^k - \tilde{y}^k \\ z^k - \tilde{z}^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}.$$

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

$\tau = 0$	1/5	1/4	1/3	1/2	2/3
$\alpha(\tau) > 1 - \epsilon$	7/8	6/7	4/5	3/4	5/8

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

$$\begin{pmatrix} y^{k+1} \\ z^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} y^{k} \\ z^{k} \\ \lambda^{k} \end{pmatrix} - \frac{3}{4} \begin{pmatrix} I & -\frac{1}{2}I & 0 \\ \frac{1}{2}I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} y^k - \tilde{y}^k \\ z^k - \tilde{z}^k \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix},$$

which is simple with almost no additional computation.

Remark 4 Note that the step (6.19b) can be written as

$$\begin{pmatrix} By^{k+1} \\ Cz^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} By^{k} \\ Cz^{k} \\ \lambda^{k} \end{pmatrix} - \alpha \begin{pmatrix} I & -(1-\tau)I & 0 \\ \tau I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} B(y^k - \tilde{y}^k) \\ C(z^k - \tilde{z}^k) \\ \lambda^k - \tilde{\lambda}^k \end{pmatrix}.$$

Meanwhile, what the step (6.19a) really needs is not (y^k, z^k), but (By^k, Cz^k). Thus, for the general case where $B \neq I$ and $C \neq I$, in the step (6.19b) we can just directly compute and store the vectors By^{k+1} and Cz^{k+1}, rather than y^{k+1} and z^{k+1} until the iteration is terminated. So, the inverses $(B^TB)^{-1}B^TC$ and $(C^TC)^{-1}C^TB$ in (6.19b) are just for algebraically showing the correction step explicitly; and they can be completely avoided in computation empirically. More specifically, the computation load in the step (6.19b) consists of only two multiplications of matrix and vector ($B\tilde{y}^k$ and $C\tilde{z}^k$) and some additions in order of $O(m)$. Thus the correction step (6.19b) is extremely simple and its iteration-independent step size is bounded away from zero. In this sense, we say that this class of algorithm (6.19a)–(6.19b) can be implemented as easily as the direct extension of ADMM (1.5).

Remark 5 In addition to the convergence already proved rigorously, we will show later that these algorithms (6.19) could even numerically outperform the direct extension of ADMM (1.5). These features make them significantly different from some existing splitting versions of the ALM such as [15, 17]. The outstanding numerical performance of (6.19) could be explained intuitively as follows. Taking a look at the step (6.19b), we see that this correction step does not change the output $\tilde{\lambda}^k$ generated by the direct extension of ADMM (6.19a); it actually only makes a balance between the variables y and z to generate a new iterate.[1] Indeed, for the direct extension of ADMM (1.5), there are two primal variables y and z 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 z uses the newest y. This lack of symmetry

[1] For the ADMM scheme, the primal variables x and y play different roles —- x is an intermediate variable which is not involved in the iteration and the only primal variable required by the iteration is y.

in primal variables may also explain why it fails to guarantee the convergence of the direct extension of ADMM (1.5). The step (6.19b) makes a balance between the output of y and z generated by the direct extension of ADMM (6.19a), and it may be regarded as a compensation to the loss of symmetry over y and z in (1.5). This may explain the reason why sometimes the algorithms (6.19) outperform the direct extension of ADMM (1.5), because the effort of balancing y and z may make the whole sequence $\{y^k, z^k, \lambda^k\}$ converge to the solution set in a more balance way and thus on a faster speed.

Remark 6 Note that we just need to check the convergence condition (3.2) when implementing the proposed algorithm (6.19); and do not need to compute the matrices H and G.

7 Numerical results

In this section, we apply the proposed algorithms (6.19) to test a concrete application of the model (1.1) 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 2.30 GHZ and 8G memory.

7.1 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 μ and ν, i.e., $f = \mu + \nu$, where μ represents the cartoon part containing the structural component and sketchy approximations of f and ν is the texture part containing the oscillations and repeated patterns of f. See more details in, e.g., [1,23,26,27,34,36]. Analytically, the cartoon part μ can be described as a piecewise smooth function and the texture part ν is typically an oscillating function. In addition to the data fidelity term $\|(\mu + \nu) - f\|_2^2$, 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 μ, it is standard to characterize it by the total variation semi-norm in [32]. To capture the feature of the texture part ν, in [33] it was suggest to first partition the texture part ν 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 $\omega_i \in R^{r^2}$ $(i = 1, 2, \cdots, s)$. Here, $s = \lceil n/r^2 \rceil$ with $\lceil \cdot \rceil$ rounding a scalar as the nearest integer towards infinity. By further realigning all ω_i's together, an r^2-by-s matrix, denoted by V, is attained. The patch mapping

$\mathcal{P} : \mathcal{R}^n \to \mathcal{R}^{r^2 \times s}$, which describes the preceding process on rearranging the texture part ν of an image as a matrix V, is defined as

$$V = \mathcal{P}\nu = [\omega_1, \omega_2, \cdots, \omega_s] \, .$$

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

$$\min_{\mu \in \mathcal{R}^n, \nu \in \mathcal{R}^n} \tau_1 \| |\nabla \mu| \|_1 + \tau_2 \|\mathcal{P}\nu\|_* + \frac{\tau_3}{2} \|K(\mu + \nu) - f\|_2^2. \tag{7.1}$$

In (7.1), $\| |\nabla \cdot | \|_1$ denotes the total variation semi-norm in [32] in order to induce the cartoon part μ; $\| \cdot \|_*$ 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 $\mathcal{P}\nu$ where ν 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.

7.2 Reformulation of (7.1)

To solve the patched low-rank image decomposition model (7.1), we first reformulate it as a special case of the model (1.1) and then delineate the subproblems when the proposed algorithms (6.19) are applied.

Introducing the auxiliary variables $z_1 \in \mathcal{R}^n \times \mathcal{R}^n$, $z_2 \in \mathcal{R}^{r^2 \times s}$ and $z_3 \in \mathcal{R}^n$, the model (7.1) can be rewritten as

$$\begin{array}{ll} \min & \tau_1 \| |z_1| \|_1 + \tau_2 \|z_2\|_* + \frac{\tau_3}{2} \|K z_3 - f\|_2^2 \\ \text{s.t.} & z_1 = \nabla \mu \\ & z_2 = \mathcal{P}\nu \\ & z_3 = \mu + \nu. \end{array} \tag{7.2}$$

Thus, (7.2) is a special case of the abstract model (1.1) with the following specifications:

- $x = \mu$, $y = \nu$, $z = (z_1, z_2, z_3)$, and $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ are all full Euclidean spaces.
- $\theta_1(x) = 0$, $\theta_2(y) = 0$ and $\theta_3(z) = \tau_1 \| |z_1| \|_1 + \tau_2 \|z_2\|_* + \frac{\tau_3}{2} \|K z_3 - f\|_2^2$.

– The coefficient matrices and the vector b are given by:

$$A = \begin{pmatrix} \nabla \\ 0 \\ I \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \mathcal{P} \\ I \end{pmatrix}, \quad C = \begin{pmatrix} -I & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{pmatrix} \quad \text{and} \quad b := 0.$$

Now, let us analyze the main subproblems when the proposed algorithms (6.19) are applied to solve the reformulation (7.2). Since the algorithms in (6.19) 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 $\tilde{x}$-subproblem in (6.19), i.e., the $\tilde{\mu}$-subproblem for (7.2), is

$$\begin{aligned} \tilde{\mu}^k &= \arg\min_{\mu} \left\{ \frac{\beta_1}{2} \left\| \nabla\mu - z_1^k - \frac{\lambda_1^k}{\beta_1} \right\|_2^2 + \frac{\beta_3}{2} \left\| \mu + \nu^k - z_3^k - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \right\} \\ &\Leftrightarrow (\beta_1 \nabla^T \nabla + \beta_3 I)\mu = \nabla^T(\beta_1 z_1^k + \lambda_1^k) + \beta_3(z_3^k - \nu^k) + \lambda_3^k, \end{aligned}$$

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 $\tilde{y}$-subproblem in (6.19), i.e., the $\tilde{\nu}$-subproblem, reads as

$$\begin{aligned} \tilde{\nu}^k &= \arg\min_{\nu} \left\{ \frac{\beta_2}{2} \left\| \mathcal{P}\nu - z_2^k - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 + \frac{\beta_3}{2} \left\| \mu^{k+1} + \nu - z_3^k - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \right\} \\ &= \frac{1}{\beta_2 + \beta_3} \left[\mathcal{P}^{-1}(\beta_2 z_2^k - \lambda_2^k) + \beta_3(z_3^k - \mu^k) + \lambda_3^k \right]. \end{aligned}$$

See more details in [33].

– The $\tilde{z}$-subproblem in (6.19), i.e., the $(\tilde{z_1}, \tilde{z_2}, \tilde{z_3})$-subproblem, is

$$\begin{aligned} (\tilde{z_1}^k, \tilde{z_2}^k, \tilde{z_3}^k) = \arg\min_{x,y,z} \Bigg\{ & \tau_1 \||z_1|\|_1 + \tau_2 \|z_2\|_* + \frac{\tau_3}{2} \|K z_3 - f\|_2^2 \\ & + \frac{\beta_1}{2} \left\| \nabla \tilde{\mu}^k - z_1 - \frac{\lambda_1^k}{\beta_1} \right\|_2^2 \\ & + \frac{\beta_2}{2} \left\| \mathcal{P}\tilde{\nu}^k - z_2 - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 + \frac{\beta_3}{2} \left\| \tilde{\mu}^k + \tilde{\nu}^k - z_3 - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \Bigg\}, \end{aligned}$$

and all the variables $\tilde{z_1}$, $\tilde{z_2}$ and $\tilde{z_3}$ can be solved simultaneously as follows.

– The $\tilde{z_1}$-subproblem can be solved explicitly by the soft-thresholding operator

$$\begin{aligned}\tilde{z_1}^k &= \arg\min_{z_1} \left\{ \tau_1 |||z_1|||_1 + \frac{\beta_1}{2} \left\| \nabla \tilde{\mu}^k - z_1 - \frac{\lambda_1^k}{\beta_1} \right\|_2^2 \right\} \\ &= \text{shrink}_{\frac{\tau_1}{\beta_1}} \left(\nabla \tilde{\mu}^k - \frac{\lambda_1^k}{\beta_1} \right),\end{aligned}$$

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

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

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

– The $\tilde{z_2}$-subproblem can be solved explicitly by the singular value decomposition (SVD)

$$\tilde{z_2}^k = \arg\min_{z_2} \left\{ \tau_2 \|z_2\|_* + \frac{\beta_2}{2} \left\| \mathcal{P}\tilde{\nu}^k - z_2 - \frac{\lambda_2^k}{\beta_2} \right\|_2^2 \right\} = \mathcal{D}_{\frac{\tau_2}{\beta_2}} \left(\mathcal{P}\tilde{\nu}^k - \frac{\lambda_2^k}{\beta_2} \right),$$

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

$$\mathcal{D}_c(W) := U \hat{\Sigma} V^T, \quad \forall\, W \in \mathcal{R}^{r^2 \times s}, \tag{7.3}$$

where $U \Sigma V^T$ is the SVD of W, and $\hat{\Sigma}_{ij} = \max\{\Sigma_{ij} - c, 0\}$ for all $1 \le i \le r^2$ and $1 \le j \le s$.

– The $\tilde{z_3}$-subproblem is solved involving in the linear operator K

$$\begin{aligned}\tilde{z_3}^k \;&=\; \arg\min_{z_3} \left\{ \frac{\tau_3}{2} \|K z_3 - f\|_2^2 + \frac{\beta_3}{2} \left\| \tilde{\mu}^k + \tilde{\nu}^k - z_3 - \frac{\lambda_3^k}{\beta_3} \right\|_2^2 \right\} \\ &\Leftrightarrow (\tau_3 K^T K + \beta_3 I) z_3 = \tau_3 K^T f + \beta_3(\tilde{\mu}^k + \tilde{\nu}^k) - \lambda_3^k.\end{aligned} \tag{7.4}$$

If K is the identity or diagonal matrix, then $\tilde{z_3}$ in (7.4) can be attained directly. If K is a convolutional matrix, then $\tilde{z}^k$ can be solved efficiently by using FFT or DCT.

Therefore, when the proposed algorithms (6.19) are applied to solve the image decomposition model (7.1), 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.

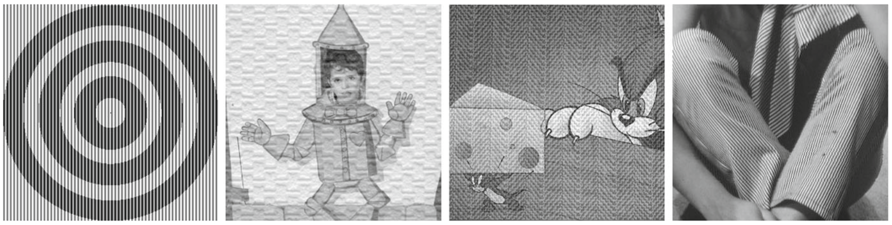

Fig. 1 Test images. From left to right: 256 × 256 Circles.png, 256 × 256 Boy.png, 512 × 512 Tomjerry.png and 256 × 256 Barbara.png

7.3 Numerical results

Now, we test different cases of (7.1) 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 (6.19) for a given case of the model (1.1); especially to see the difference of various choices of the step size α for different values of τ. As mentioned, the direct extension of ADMM (1.5) ("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 fastest method among existing splitting methods originated from the ALM in the literature. 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 (7.1), which is not the theme of this paper. Instead, we simply follow the suggestion in [33, Theorems 3.3-3.6] 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 (7.3) is executed by an efficient MATLAB Mex interface for computing SVD via a divide-and-conquer routine (dgesdd) implemented in LAPACK. (3) When the value of τ is determined in (6.19b), the step size α in this correction step can be chosen as any value in the interval $(0, \alpha(\tau))$. Usually we need to avoid smaller step size whenever possible. Thus we prefer larger values of α close to $\alpha(\tau)$. For example, we can choose larger values of α according to the methodology in Table 1. (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 μ and texture ν obtained via an image decomposition model:

$$\mathrm{Corr}(\mu, \nu) := \mathrm{cov}(\mu, \nu) \Big/ \sqrt{\mathrm{var}(\mu) \cdot \mathrm{var}(\nu)}, \tag{7.5}$$

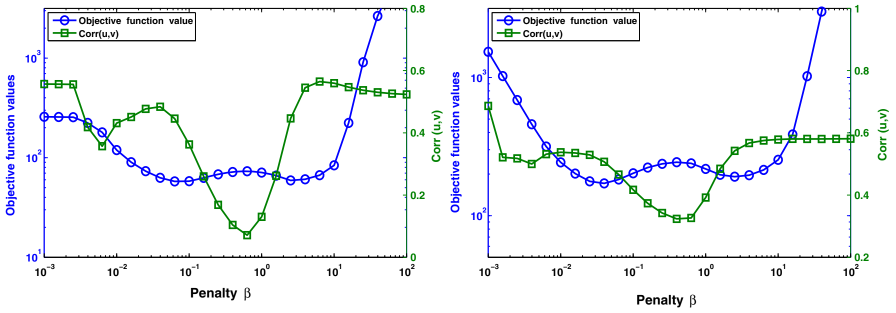

Fig. 2 The implementation of (6.19) with $(\tau, \alpha) = (1/5, 7/8)$ to (7.1) with $K = I$ for Barbara, with different β. Evolution of the objective function values and Corr(μ, ν) w.r.t. different β. Left: for Barbara. Right: for Tomjerry

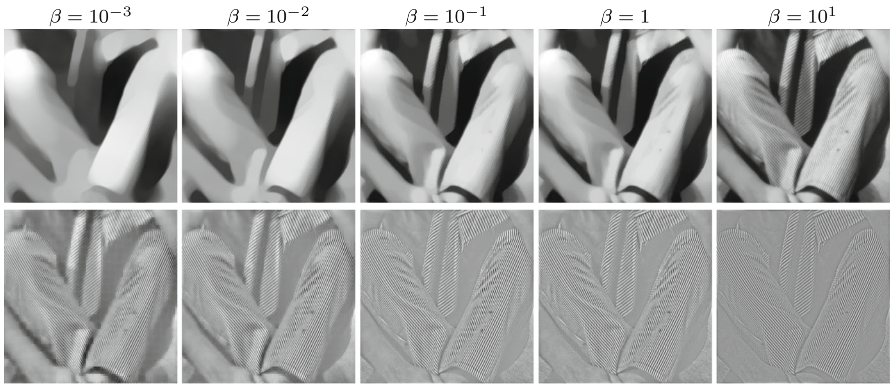

Fig. 3 The implementation of (6.19) with $(\tau, \alpha) = (1/5, 7/8)$ to (7.1) with $K = I$ for Barbaba, with different β. Decomposed cartoons (top row) and textures (bottom row)

where var$(\cdot)$ and 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 (6.19) with $(\tau, \alpha) = (1/5, 7/8)$ to the special case of the model (7.1) with $K = I$. We take the Barbara and Tomjerry images in Fig. 1 as the tested images. A total number of 25 cases of β in the interval $[10^{-3}, 10^{2}]$ are tested on the points 10^{-l} where l varies from -3 to 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 (7.1) and the magnitude of the correlation Corr(μ, ν). The plots are displayed in Fig. 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 Fig. 3, the decomposed cartoon and texture parts are displayed for some values of β in different levels of magnitude.

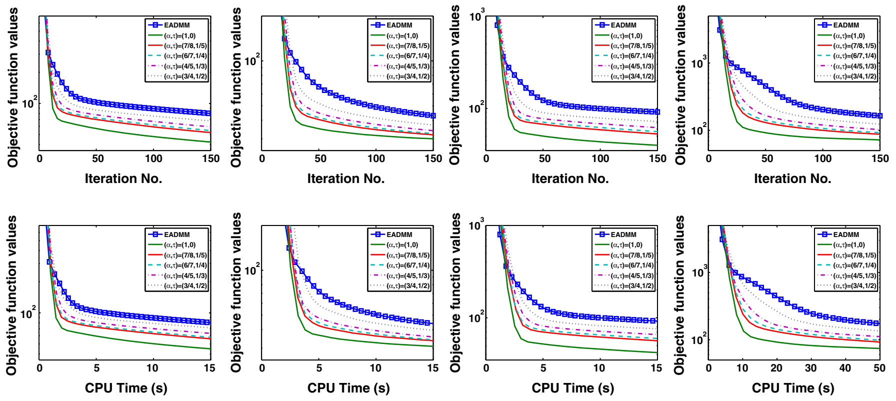

Fig. 4 The implementation of (6.19) with different (τ, α) to (7.1) with $K = I$, $\beta = 1$. Evolutions of the objective function values w.r.t. the number of iterations (upper row) and computing time in seconds (lower row). From left column to right column: for Circles, Boy, Barbara and Tomjerry images

7.3.1 The case of $K = I$

For the case $K = I$, the model (7.1) is for decomposing a clean image without any degradation. The parameters (τ_1, τ_2, τ_3) in (7.1) 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 (6.19) are taken as zeros.

In Fig. 4, we plot the evolutions of the objective function values for the model (7.1) with respect to the number of iterations and computing time in seconds, for several cases of the algorithms in (6.19) 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 (6.19b). The decomposed cartoon and texture parts by implementing the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations are displayed in Fig. 5.

7.3.2 The case of $K = S$

We now test the case of model (7.1) 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 Fig. 6.

For this case, the parameters (τ_1, τ_2, τ_3) in (7.2) 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 (6.19) are taken as zeros.

In addition to the objective function value, we 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

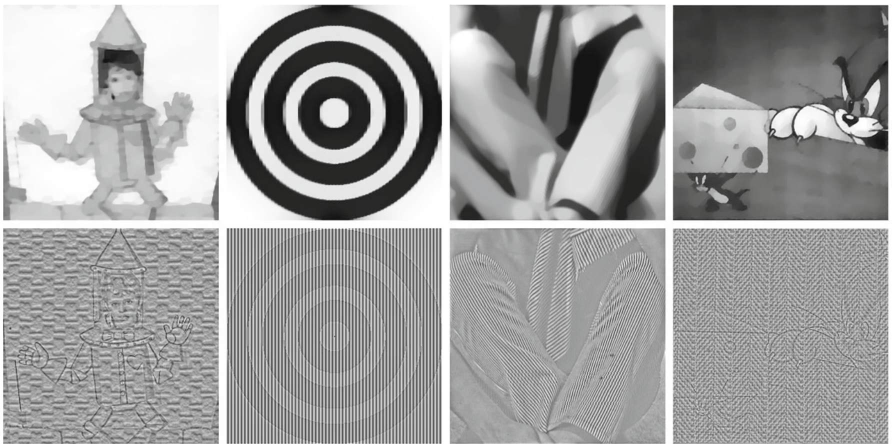
Fig. 5 Decomposed cartoon (top row) and texture (bottom row) parts by implementing (6.19) with $(\tau, \alpha) = (1/5, 7/8)$ to (7.1) with $K = I$ for 150 iterations

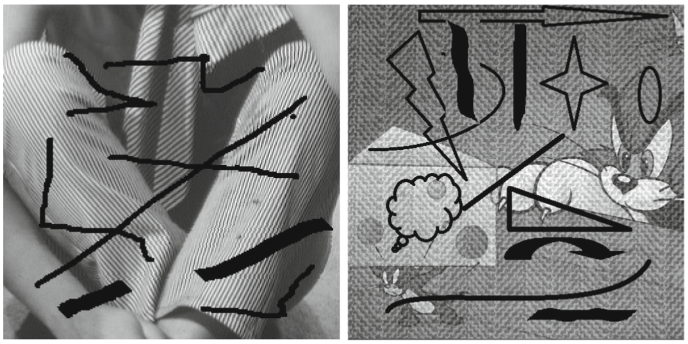
Fig. 6 The case of (7.1) with $K = S$. Left: corrupted Barbara image; Right: corrupted Tomjerry image

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

where $\bar{f}$ is the approximation of the ground truth f^*. For the corrupted images listed in Fig. 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 Fig. 7, for several cases of the algorithms in (6.19) 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 Fig. 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 Fig. 8.

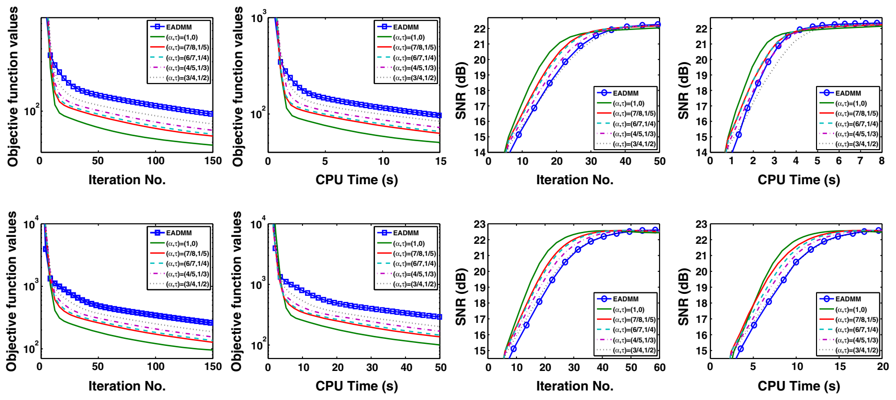

Fig. 7 The implementation of (6.19) with different (τ, α) to (7.1) with $K = S$, $\beta = 1$. Evolutions of the objective function value and SNR value w.r.t. the number of iterations and computing time in seconds. Top row: for Barbara. Bottom row: for Tomjerry

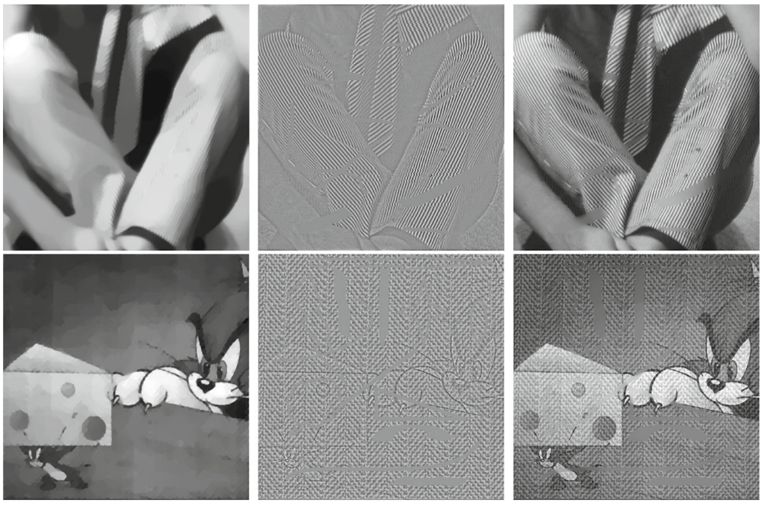

Fig. 8 The decomposed cartoon (left column) and texture (middle column) parts by the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations; and the superposed (right column) images. Top row: for Barbara. Bottom row: for Tomjerry

7.3.3 The case of $K = B$

Finally, we test the case of the model (7.1) with $K = B$. 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 Fig. 9. For the corrupted images listed in Fig. 9, the SNR values are 13.18dB for the corrupted Barbara image and 24.18dB for the corrupted Boy image, respectively.

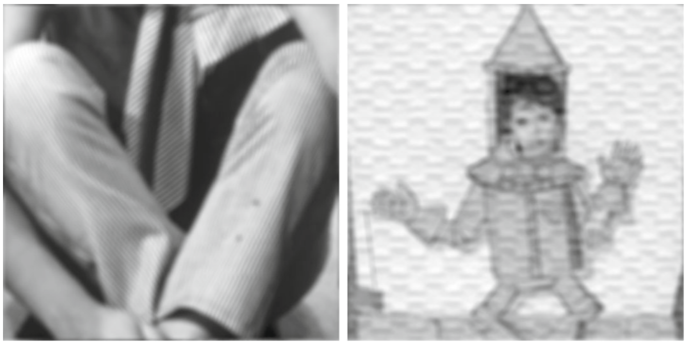

Fig. 9 The case of (7.1) with $K = B$. Convolved Barbara (left) and Boy (right) images by out-of-focus blur with radius 3

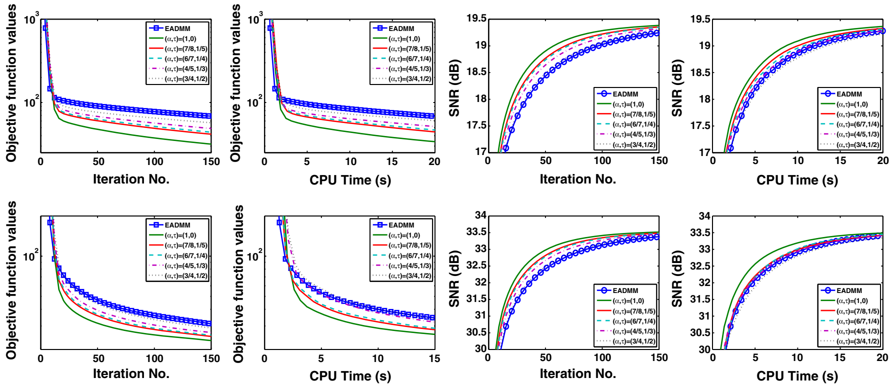

Fig. 10 The implementation of (6.19) with different (τ, α) to (7.1) with $K = B$, $\beta = 1$. Evolutions of the objective function value and SNR value w.r.t. the number of iterations and computing time in seconds. Top row: for Barbara. Bottom row: for Boy

For this case, the parameters (τ_1, τ_2, τ_3) in (7.2) 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 (6.19) 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 Fig. 10, for several cases of the algorithms in (6.19) 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 Fig. 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 Fig. 11.

8 Conclusions

We focus on a convex minimization model with linear constraints and an objective function in form of the sum of three functions without coupled variables; and discuss

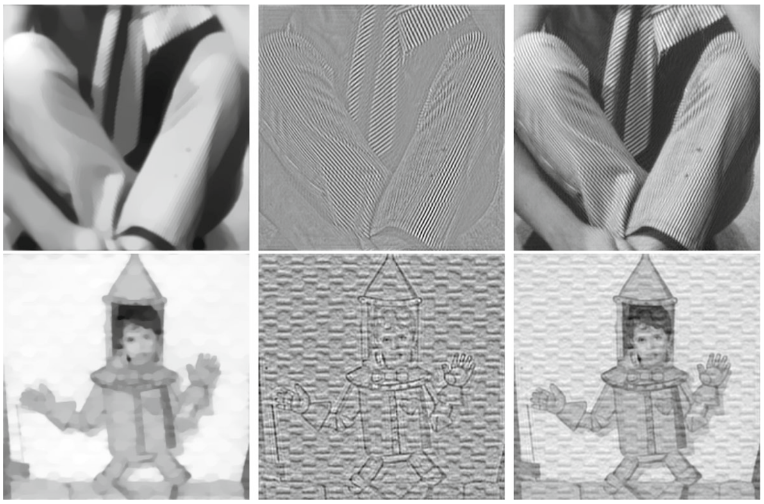

Fig. 11 The decomposed cartoon (left column) and texture (middle column) parts by the proposed algorithm with $(\tau, \alpha) = (1/5, 7/8)$ for 150 iterations; and the superposed (right column) images. Top row: for Barbara. Bottom row: for Boy

how to develop a class of splitting algorithms. We propose a new prototype algorithm in the prediction-correction framework, which uses the output of the direct extension of the alternating direction method of multipliers (ADMM) as a predictor and corrects it by a simple correction step. A unified and easily checkable condition to ensure the convergence of this prototype algorithm is also given, based on which the possible divergence of the direct extension of ADMM can be easily explained. A class of specific ADMM-based algorithms are easily developed based on this prototype algorithm. These new algorithms can be implemented as easily as the direct extension of ADMM; but they could be even faster numerically and they have proved convergence and estimated worst-case convergence rates measured by the iteration complexity. They expect to find various applications.

As mentioned in the introduction, we only discuss the case where the direct extension of ADMM (1.5) is completely preserved and thus the subproblems of the proposed new algorithms are in form of (1.4). With the purpose of further simplifying the subproblems as easy as (1.3), linearized versions of the proposed algorithms can be developed. Such a linearized version treats the subproblems more sophisticatedly, and it requires more meticulous analysis to prove its convergence and convergence rate. The convergence analysis techniques in some references, e.g., [8,19,37,38], are useful for considering linearized versions of the proposed algorithms. Also, to expose our main idea more clearly, we only focus on the convex minimization model where there are three functions in its objective; and it is interesting to 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 in this paper. But technically, the analysis for the general case should be more demanding and the conclusion should also be different. For example, the lower bound for the step size α in the prototype algorithm (3.1) is expected to be more conservative because of the more general setting.

Acknowledgements Fundig was provided by National Natural Science Foundation of China (Grant No. 10471056), NSFC/RGC Joint Research Scheme (Grant No. N_PolyU504/14) and Research Grants Council, University Grants Committee (Grant No. HKBU 12313516).

References

1. Aujol, J.F., Gilboa, G., Chan, T., Osher, S.: Structure-texture image decomposition–modeling, algorithms, and parameter selection. Int. J. Comput. Vis. **67**, 111–136 (2006)
2. Blum, E., Oettli, W.: Mathematische Optimierung. Grundlagen und Verfahren. Ökonometrie und Unternehmensforschung. Springer, Berlin (1975)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foun. Trends Mach. Learn. **3**, 1–122 (2010)
4. Chan, T. F., Glowinski, R.: Finite element approximation and iterative solution of a class of mildly non-linear elliptic equations. Technical Report, Stanford University (1978)
5. Chandrasekaran, V., Parrilo, P.A., Willsky, A.S.: Latent variable graphical model selection via convex optimization. Ann. Stat. **40**, 1935–1967 (2012)
6. Chen, C.H., He, B.S., Ye, Y.Y., Yuan, X.M.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Program. Ser. A **155**, 57–79 (2016)
7. Eckstein, J., Yao, W.: Augmented Lagrangian and alternating direction methods for convex optimization: a tutorial and some illustrative computational results. Pac. J. Optim. **11**(4), 619–644 (2015)
8. Esser, E., Zhang, X., Chan, T.F.: A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. SIAM J. Imaging Sci. **3**, 1015–1046 (2010)
9. Facchinei, F., Pang, J.S.: Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer Series in Operations Research, vol. I. Springer, New York (2003)
10. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. Comput. Math. Appl. **2**, 17–40 (1976)
11. Glowinski, R.: On alternating directon methods of multipliers: a historical perspective. In: Fitzgibbon, W., Kuznetsov, Y.A., Neittaanmaki, P., Pironneau, O. (eds.) Modeling, Simulation and Optimization for Science and Technology, pp. 59–82. Springer, Dordrecht (2014)
12. Glowinski, R., Marrocco, A.: Approximation par èlèments finis d'ordre un et résolution par pénalisation-dualité d'une classe de problèmes non linéaires, R.A.I.R.O., R2, 41–76 (1975)
13. Han, D.R., Yuan, X.M.: A note on the alternating direction method of multipliers. J. Optim. Theory Appl. **155**, 227–238 (2012)
14. Hansen, P., Nagy, J., O'Leary, D.: Deblurring Images: Matrices, Spectra, and Filtering. SIAM, Philadelphia (2006)
15. He, B.S., Tao, M., Yuan, X.M.: Alternating direction method with Gaussian back substitution for separable convex programming. SIAM J. Optim. **22**, 313–340 (2012)
16. He, B.S., Tao, M., Yuan, X.M.: A splitting method for separable convex programming. IMA J. Numer. Anal. **35**, 394–426 (2015)
17. He, B.S., Tao, M., Yuan, X.M.: Convergence rate and iteration complexity on the alternating direction method of multipliers with a substitution procedure for separable convex programming. Math. Oper. Res. **42**(3), 662–691 (2017)
18. He, B.S., Yuan, X.M.: On the O(1/n) convergence rate of Douglas-Rachford alternating direction method. SIAM J. Numer. Anal. **50**, 700–709 (2012)
19. He, B.S., Yuan, X.M.: Linearized alternating direction method with Gaussian back substitution for separable convex programming. Numer. Algebra Control Optim. **3**, 247–260 (2013)
20. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theory Appl. **4**, 303–320 (1969)
21. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**, 964–979 (1979)
22. McLachlan, G.J.: Discriminant Analysis and Statistical Pattern Recognition. Wiley, Hoboken (2004)
23. Meyer, Y.: Oscillating Patterns in Image Processing and Nonlinear Evolution Equations. University Lecture Series. AMS, Providence (2002)

24. Nemirovsky, A.S., Yudin, D.B.: Problem Complexity and Method Efficiency in Optimization. Wiley-Interscience Series in Discrete Mathematics. Wiley, New York (1983)
25. Nesterov, Y.E.: A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. Doklady AN SSSR **269**, 543–547 (1983)
26. Ng, M.K., Yuan, X.M., Zhang, W.X.: A coupled variational image decomposition and restoration model for blurred cartoon-plus-texture images with missing pixels. IEEE Trans. Imaging Proc. **22**, 2233–2246 (2013)
27. Osher, S., Sole, A., Vese, L.: Image decomposition and restoration using total variation minimization and the H^{-1} norm. Multiscale Model. Simul. **1**, 349–370 (2003)
28. Passty, G.B.: Ergodic convergence to a zero of the sum of monotone operators in Hilbert space. J. Math. Anal. Appl. **72**, 383–390 (1979)
29. Peng, Y.G., Ganesh, A., Wright, J., Xu, W.L., Ma, Y.: Robust alignment by sparse and low-rank decomposition for linearly correlated images. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 2233–2246 (2012)
30. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, New York (1969)
31. Rockafellar, R.T.: Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. **1**, 97–116 (1976)
32. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D **60**, 259–268 (1992)
33. Schaeffer, H., Osher, S.: A low patch-rank interpretation of texture. SIAM J. Imaging Sci. **6**, 226–262 (2013)
34. Starck, J., Elad, M., Donoho, D.L.: Image decomposition via the combination of sparse representations and a variational approach. IEEE Trans. Image Process. **14**, 1570–1582 (2005)
35. Tao, M., Yuan, X.M.: Recovering low-rank and sparse components of matrices from incomplete and noisy observations. SIAM J. Optim. **21**, 57–81 (2011)
36. Vese, L., Osher, S.: Modeling textures with total variation minimization and oscillating patterns in image processing. J. Sci. Comput. **19**, 553–572 (2003)
37. Wang, X.F., Yuan, X.M.: The linearized alternating direction method for Dantzig Selector. SIAM J. Sci. Comput. **34**, A2792–A2811 (2012)
38. Yang, J.F., Yuan, X.M.: Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization. Math. Comput. **82**, 301–329 (2013)