Journal of Computational Mathematics
Vol.34, No.6, 2016, 723–738.
http://www.global-sci.org/jcm
doi:10.4208/jcm.1612-m2016-0536

A PRIMAL-DUAL FIXED POINT ALGORITHM FOR MULTI-BLOCK CONVEX MINIMIZATION[*]

Peijun Chen
School of Mathematical Sciences, MOE-LSC and School of Biomedical Engineering, Shanghai Jiao Tong University, China
Department of Mathematics, Taiyuan University of Science and Technology, China
Email: chenpeijun@sjtu.edu.cn
Jianguo Huang
School of Mathematical Sciences, and MOE-LSC, Shanghai Jiao Tong University, China
Email: jghuang@sjtu.edu.cn
Xiaoqun Zhang
Institute of Natural Sciences, School of Mathematical Sciences, and MOE-LSC Shanghai Jiao Tong University, China
Email: xqzhang@sjtu.edu.cn

Abstract

We have proposed a primal-dual fixed point algorithm (PDFP) for solving minimization of the sum of three convex separable functions, which involves a smooth function with Lipschitz continuous gradient, a linear composite nonsmooth function, and a nonsmooth function. Compared with similar works, the parameters in PDFP are easier to choose and are allowed in a relatively larger range. We will extend PDFP to solve two kinds of separable multi-block minimization problems, arising in signal processing and imaging science. This work shows the flexibility of applying PDFP algorithm to multi-block problems and illustrates how practical and fully splitting schemes can be derived, especially for parallel implementation of large scale problems. The connections and comparisons to the alternating direction method of multiplier (ADMM) are also present. We demonstrate how different algorithms can be obtained by splitting the problems in different ways through the classic example of sparsity regularized least square model with constraint. In particular, for a class of linearly constrained problems, which are of great interest in the context of multi-block ADMM, can be also solved by PDFP with a guarantee of convergence. Finally, some experiments are provided to illustrate the performance of several schemes derived by the PDFP algorithm.

Mathematics subject classification: 65K05, 46N10, 90C06, 90C25
Key words: Primal-dual fixed point algorithm, Multi-block optimization problems.

1. Introduction

In this paper, we are concerned with extending the primal-dual fixed point (PDFP) algorithm proposed in [5] for solving two kinds of general multi-block problems (1.1) and (1.2) with fully splitting schemes. Let $\Gamma_0(\mathbb{R}^n)$ denote the collection of all proper lower semicontinuous convex functions from $\mathbb{R}^n$ to $(-\infty,+\infty]$. The first kind of problems are formulated as

$$\min_{x\in\mathbb{R}^n} f_1(x)+\sum_{i=1}^{N}\theta_i(B_ix+b_i)+f_3(x), \tag{1.1}$$

[*] Received February 1, 2016 / Revised version received October 30, 2016 / Accepted December 8, 2016 / Published online December 16, 2016 /

where $\theta_i \in \Gamma_0(\mathbb{R}^{m_i})$, $B_i : \mathbb{R}^n \to \mathbb{R}^{m_i}$ a linear transform, $b_i \in \mathbb{R}^{m_i}$, $i = 1, \cdots, N$. $f_1, f_3 \in \Gamma_0(\mathbb{R}^n)$ and f_1 is differentiable on $\mathbb{R}^n$ with $1/\beta$-Lipschitz continuous gradient for some $\beta \in (0, +\infty]$. If $f_1 = 0$, we can take $\beta = +\infty$. Many problems in image processing and signal recovery with multi-regularization terms can be formulated in the form of (1.1).

The second kind of problems under discussion are optimization problems with constraints, given as follows.

$$\min_{x_1,\cdots,x_N} \sum_{i=1}^{N_1} \theta_i(B_i x_i + b_i) + \sum_{i=N_1+1}^{N} \theta_i(x_i) \tag{1.2a}$$

$$\text{st.} \sum_{i=1}^{N} A_i x_i = a, \tag{1.2b}$$

$$x_i \in C_i, \quad i = 1, \cdots, N.$$

Here, $\theta_i \in \Gamma_0(\mathbb{R}^{m_i})$, $B_i : \mathbb{R}^{n_i} \to \mathbb{R}^{m_i}$ a linear transform and $b_i \in \mathbb{R}^{m_i}$ for $i = 1, \cdots, N_1$. Moreover, for $i = N_1 + 1, \cdots, N$, $\theta_i \in \Gamma_0(\mathbb{R}^{n_i})$ is differentiable on $\mathbb{R}^{n_i}$ with $1/\beta_i$-Lipschitz continuous gradient for some $\beta_i \in (0, +\infty]$. For $i = 1, \cdots, N$, the constraint set $C_i \subset \mathbb{R}^{n_i}$ is nonempty, closed and convex, A_i is a $l \times n_i$ matrix, and $a \in \mathbb{R}^l$.

Many problems can be formulated in the form (1.2), for example elliptic optimal control problems [6]. In some applications, the problem (1.1) can be viewed as a decomposition on the observed data, while the problem (1.2) is a mixture of the variables and data decomposition. In particular, for some special cases, both problems (1.1) and (1.2) can be abstracted as

$$\min_{x_1,x_2,\cdots,x_N} \sum_{i=1}^{N} \theta_i(x_i) \tag{1.3a}$$

$$\text{st.} \sum_{i=1}^{N} A_i x_i = a, \tag{1.3b}$$

$$x_i \in C_i, \quad i = 1, \cdots, N,$$

by properly introducing auxiliary variables, or vice-visa, depending on the simplicity of the functions θ_i involved. In the literature, many existing works have been devoted to solving (1.3), for example, the variants of popular alternating direction method of multipliers (ADMM) [9,11, 12] for three or more block problems. It deserves to point out that Davis and Yin [10] proposed a very interesting "primal-only" splitting scheme for solving an inclusion problem involving three maximal monotone operators, which was also used for solving (1.3) by themselves.

Now, let us recall the primal-dual fixed point algorithm PDFP in [5] for solving the following three-block problem

$$\min_{x \in \mathbb{R}^n} f_1(x) + f_2(Bx + b) + f_3(x). \tag{1.4}$$

In (1.4), $f_2 \in \Gamma_0(\mathbb{R}^m)$, $B : \mathbb{R}^n \to \mathbb{R}^m$ a linear transform, $b \in \mathbb{R}^m$, f_1 and f_3 are the same ones as given in (1.1). As usual, define the proximity operator prox_f of f by (cf. [7])

$$\text{prox}_f(x) = \arg\min_{y \in \mathbb{R}^n} f(y) + \frac{1}{2}\|x - y\|^2.$$

Then, our PDFP algorithm can be described as follows.

$$\text{(PDFP)}\quad\begin{cases} x^{k+1/2} = \operatorname{prox}_{\gamma f_3}(x^k - \gamma\nabla f_1(x^k) - \lambda B^T v^k), & (1.5a)\\ v^{k+1} = (I - \operatorname{prox}_{\frac{\gamma}{\lambda} f_2})(Bx^{k+1/2} + b + v^k), & (1.5b)\\ x^{k+1} = \operatorname{prox}_{\gamma f_3}(x^k - \gamma\nabla f_1(x^k) - \lambda B^T v^{k+1}), & (1.5c)\end{cases}$$

where $0 < \lambda < 1/\lambda_{\max}(BB^T)$ and $0 < \gamma < 2\beta$. Compared with similar work in [8,14], the parameters in PDFP are easier to choose and are allowed in a larger range, as they can be chosen independently according to the Lipschitz constant and the operator norm of BB^T. In this sense, our parameter rules are relatively more practical. In the numerical experiments, we can set λ to be close to $1/\lambda_{\max}(BB^T)$ and γ to be close to 2β for most of tests. Moreover, the results of $x^k - \gamma\nabla f_1(x^k)$ and $\lambda B^T v^{k+1}$ can be stored as two intermediate variables that can be reused in (1.5a) and (1.5c) during the iterations. Nevertheless, PDFP has an extra step (1.5a) and $\operatorname{prox}_{\gamma f_3}$ is computed twice. In practice, this step is often related to operations with low cost such as ℓ_1 shrinkage or straightforward projections. So the cost could be still ignorable in practice.

The purpose of this paper is intended to extend PDFP to solve the above two kinds of general multi-block problems (1.1) and (1.2) with fully splitting schemes. The key trick of our treatment is the use of PDFP combined with feasible reformulation of the multi-block problems in the form (1.4), so that we can derive many variants of iterative schemes with different structures. One obvious advantage of the extended schemes is their simplicity and the convenience for parallel implementation. Some of the algorithms derived in this paper already exist in the literature and some of them are new and effective. The new schemes are compared with the ADMM and we will show the connection and the difference later on. We mention in passing that similar techniques are also adopted in [4,8,13,16]. Compared with the schemes developed in [8,13,16], if a scheme is established based on PDFP with f_1 nonzero in (1.4), it is more convenient for us to choose parameters in applications, as shown in [5]. However, if a scheme is constructed based on PDFP by viewing f_1 equal to 0, our PDFP requires to compute the action of the operator $\operatorname{prox}_{\gamma f_3}$ twice, compared with the algorithms in [1,8,13,14]. If the calculation of $\operatorname{prox}_{\gamma f_3}$ is time-consuming, this will lead to additional computational cost.

The rest of the paper is organized as follows. In Section 2, we will show how PDFP can be extended to solve (1.1), present the connections and differences with ADMM, and derive different algorithms by using the constrained and sparse regularized image restoration model as an illustrative example. In Section 3, PDFP is extended to solve (1.2), and we also show the comparison with ADMM. In Section 4, the numerical performance and efficiency of the variants of PDFP are demonstrated through constrained total variation computerized tomography (CT) reconstruction and solving quadratic programming model.

2. PDFP for the Muti-block Problem (1.1)

2.1. Algorithm and its deduction

In this subsection, we formulate (1.1) as a special case of (1.4). Then the PDFP algorithm can be applied and formulated in parallel form due to the separability of f_2 on its variables. Similar technique has also been used in [4,8,13,16] and we present the details here for completeness.

Rewrite the second term in (1.1) as

$$f_2(Bx+b) = \sum_{i=1}^{N} \theta_i(B_i x + b_i)$$

with the symbols

$$f_2(y) = \sum_{i=1}^{N} \theta_i(y_i), y = Bx+b,$$

$$B = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}.$$

Thus, the problem (1.1) can be recast in the form of (1.4) and be resolved with PDFP. Since f_2 is separable in terms of its variables, the scheme (1.5) can be further expressed as

$$\begin{cases} x^{k+1/2} = \operatorname{prox}_{\gamma f_3}\Big(x^k - \gamma\nabla f_1(x^k) - \lambda \sum_{j=1}^{N} B_j^T v_j^k\Big), & (2.1a) \\ v_i^{k+1} = (I - \operatorname{prox}_{\frac{\gamma}{\lambda}\theta_i})(B_i x^{k+1/2} + b_i + v_i^k), \quad i = 1, \cdots, N, & (2.1b) \\ x^{k+1} = \operatorname{prox}_{\gamma f_3}\Big(x^k - \gamma\nabla f_1(x^k) - \lambda \sum_{j=1}^{N} B_j^T v_j^{k+1}\Big). & (2.1c) \end{cases}$$

The convergence condition of PDFP in [5] implies that the above algorithm is convergent whenever $0 < \lambda < 1/\sum_{i=1}^{N} \lambda_{\max}(B_i B_i^T)$ and $0 < \gamma < 2\beta$. The scheme (2.1) is naturally in a parallel form, which may be useful for large scale problems. Also for some special cases, such as $f_1 = 0$, $f_3 = \chi_C$, one may even get simpler forms (see [5] for details).

2.2. Comparison to ADMM

There are many works on ADMM methods [9,11,12]. We will show the difference between PDFP and ADMM for solving (1.1). Since our method for solving (1.1) is based on the PDFP (1.5) for solving (1.4). We first show how the ADMM resolves the same problem. In fact, we should first reformulate the problem in the form (1.3) by introducing auxiliary variables. Then, we can use the ADMM to drive the scheme for solving (1.4). However, our PDFP is developed based on a fixed point formulation the solution of (1.4) must satisfy. So the ideas of constructing the two methods are quite different.

To show the difference of the two methods more clearly, we compare their schemes for solving (1.1) with $f_3 = 0$. PDFP for solving (1.1) have been given in (2.1) based on three blocks algorithm (1.5). We can also use the similar technique to achieve the ADMM method in this case:

$$\begin{cases} x^{k+1} = \operatorname*{argmin}_{x\in\mathbb{R}^n} f_1(x) + \frac{\alpha}{2}\sum_{i=1}^{N} \|B_i x + (b_i - y_i^k + v_i^k)\|^2, & (2.2a) \\ y_i^{k+1} = \operatorname{prox}_{\frac{1}{\alpha}\theta_i}(B_i x^{k+1} + b_i + v_i^k), \quad i = 1, \cdots, N, & (2.2b) \\ v_i^{k+1} = v_i^k + \tau(B_i x^{k+1} + b_i - y_i^{k+1}), \quad i = 1, \cdots, N. & (2.2c) \end{cases}$$

As a matter of fact, the scheme (2.2) follows from an application of the two block ADMM for solving (1.1) with $f_3 = 0$ and the convergence condition for (2.2) is still $\alpha > 0$ and $\tau \in (0, (1+\sqrt{5})/2)$.

2.3. Application to constrained sparse regularization problems

In this subsection, we will consider how to get different algorithms by using the extension of PDFP (2.1) for a specific problem. The problem that we are interested is the well-known constrained sparse regularization model in inverse problems and imaging:

$$\min_{x\in C} \frac{1}{2}\|Ax - a\|^2 + \mu\|Dx\|_1, \tag{2.3}$$

where $\|Ax - a\|^2$ is the smooth data-fidelity term, $\mu\|Dx\|_1$ is the regularization term to ensure the solution is sparse under the transform D and μ is the regularization parameter. The problem (2.3) is equivalent to

$$\min_{x\in \mathbb{R}^n} \frac{1}{2}\|Ax - a\|^2 + \mu\|Dx\|_1 + \chi_C(x), \tag{2.4}$$

where

$$\chi_C(x) = \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C. \end{cases}$$

First, applying PDFP (1.5) to the problem (1.4) with the three blocks given by $f_1(x) = \frac{1}{2}\|Ax - a\|^2$, $f_2 = \mu\|\cdot\|_1$, $B = D$, $b = 0$, $f_3 = \chi_C$ and noting $\mathrm{prox}_{\gamma\chi_C} = \mathrm{proj}_C$, we obtain

$$\text{(Scheme 1)} \begin{cases} x^{k+1/2} = \mathrm{proj}_C\Big(x^k - \gamma A^T(Ax^k - a) - \lambda D^T v^k\Big), \\ v^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2} + v^k), \\ x^{k+1} = \mathrm{proj}_C\Big(x^k - \gamma A^T(Ax^k - a) - \lambda D^T v^{k+1}\Big), \end{cases} \tag{2.5}$$

where $0 < \lambda < 1/\lambda_{\max}(DD^T)$ and $0 < \gamma < 2/\lambda_{\max}(A^TA)$. This is the original algorithm proposed in [5], and the main computation is matrix-vector multiplication that is very suitable for parallel computation.

The second scheme can be obtained by setting $f_1(x) = \frac{1}{2}\|Ax - a\|^2$, $\theta_1 = \mu\|\cdot\|_1$, $B_1 = D$, $b_1 = 0$, $\theta_2 = \chi_C$, $B_2 = I$, $b_2 = 0$, $f_3 = 0$, leading to

$$\text{(Scheme 2)} \begin{cases} x^{k+1/2} = x^k - \gamma A^T(Ax^k - a) - \lambda D^T v_1^k - \lambda v_2^k, \\ v_1^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2} + v_1^k), \\ v_2^{k+1} = (I - \mathrm{proj}_C)(x^{k+1/2} + v_2^k), \\ x^{k+1} = x^k - \gamma A^T(Ax^k - a) - \lambda D^T v_1^{k+1} - \lambda v_2^{k+1}, \end{cases} \tag{2.6}$$

where $0 < \lambda \le 1/(\lambda_{\max}(DD^T) + 1)$ and $0 < \gamma < 2/\lambda_{\max}(A^TA)$. This scheme (2.6) is the form proposed in [4] by recasting the problem in two-block. We note that x^{k+1} may not be a feasible solution during the iteration. In addition, an auxiliary variable v_2 is introduced and the permitted ranges of the parameter λ is also a little tighter compared with Scheme 1.

In the following, we present some schemes to use different properties of the objective functions $\frac{1}{2}\|Ax - a\|^2$, which may involve the main computational cost in inverse problem applications. By setting $f_1(x) = 0$, $\theta_1 = \mu\|\cdot\|_1$, $B_1 = D$, $b_1 = 0$, $\theta_2 = \frac{1}{2}\|Ax - a\|^2$, $B_2 = I$, $b_2 = 0$,

$f_3 = \chi_C$, we can use (2.1) to solve (2.3) and obtain

$$\begin{cases} x^{k+1/2} = \mathrm{proj}_C(x^k - \lambda(D^T v_1^k + v_2^k)), \\ v_1^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2} + v_1^k), \\ v_2^{k+1} = (x^{k+1/2} + v_2^k) - \Big(I + \dfrac{\gamma}{\lambda}A^T A\Big)^{-1}\Big(\dfrac{\gamma}{\lambda}A^T a + x^{k+1/2} + v_2^k\Big), \\ x^{k+1} = \mathrm{proj}_C\Big(x^k - \lambda(D^T v_1^{k+1} + v_2^{k+1})\Big), \end{cases} \tag{2.7}$$

where $0 < \lambda < 1/(\lambda_{\max}(DD^T) + 1)$ and $0 < \gamma < +\infty$. This scheme can be practical when the inverse of the matrix $(I + \frac{\gamma}{\lambda}A^T A)$ is easy to obtain, for examples, for some diagonalizable matrix $A^T A$.

When the inverse of the matrix $(I+\frac{\gamma}{\lambda}A^T A)$ is not easy to compute, we can rewrite $\frac{1}{2}\|Ax-a\|^2$ as $\frac{1}{2}\|\cdot\|^2 \circ (Ax - a)$ and set $f_1(x) = 0$, $\theta_1 = \mu\|\cdot\|_1$, $B_1 = D$, $b_1 = 0$, $\theta_2 = \frac{1}{2}\|\cdot\|^2$, $B_2 = A$, $b_2 = -a$, $f_3 = \chi_C$. Then we have

$$\text{(Scheme 3)} \begin{cases} x^{k+1/2} = \mathrm{proj}_C\Big(x^k - \lambda(D^T v_1^k + A^T v_2^k)\Big), \\ v_1^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2} + v_1^k), \\ v_2^{k+1} = \dfrac{\gamma}{\gamma+\lambda}(Ax^{k+1/2} - a + v_2^k), \\ x^{k+1} = \mathrm{proj}_C\Big(x^k - \lambda(D^T v_1^{k+1} + A^T v_2^{k+1})\Big), \end{cases} \tag{2.8}$$

where $0 < \lambda < 1/(\lambda_{\max}(DD^T) + \lambda_{\max}(A^T A))$ and $0 < \gamma < +\infty$.

If we partition A and a into N block rows, namely $A = (A_1^T, A_2^T, \cdots, A_N^T)^T$, $a = (a_1^T, a_2^T, \cdots, a_N^T)^T$, where A_j is a $m_j \times n$ matrix and $m_j << n$, $a_j \in \mathbb{R}^{m_j}$, then

$$\frac{1}{2}\|Ax - a\|^2 = \frac{1}{2}\sum_{j=1}^{N}\|A_j x - a_j\|^2.$$

Here A_i is different from the ones in (1.2)-(1.3), and they are only used in this subsection. It is very easy to see that the scheme (2.8) can be written in a parallel form as

$$\begin{cases} x^{k+1/2} = \mathrm{proj}_C\Big(x^k - \lambda(D^T v_1^k + \sum_{j=1}^{N} A_j^T v_{2j}^k)\Big), \\ v_1^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2} + v_1^k), \\ v_{2i}^{k+1} = \dfrac{\gamma}{\gamma+\lambda}(A_i x^{k+1/2} - a_i + v_{2i}^k), \quad i = 1, \cdots, N, \\ x^{k+1} = \mathrm{proj}_C\Big(x^k - \lambda(D^T v_1^{k+1} + \sum_{j=1}^{N} A_j^T v_{2j}^{k+1})\Big), \end{cases} \tag{2.9}$$

where $0 < \lambda < 1/(\lambda_{\max}(DD^T) + \sum_{j=1}^N \lambda_{\max}(A_j^T A_j))$ and $0 < \gamma < +\infty$. It is evident that all v_{2i}^{k+1} $(1 \le i \le N)$ in the third substep of (2.9) can be computed in parallel.

The above schemes except (2.7) are fully explicit and involves only matrix-vector multiplication. In the following, we derive a semi-implicit scheme, which only involves the inverse of small size matrix. By setting $f_1(x) = 0$, $\theta_1(x) = \mu\|x\|_1$, $B_1 = D$, $b_1 = 0$, $\theta_{i+1}(x) = \frac{1}{2}\|A_i x - a_i\|^2$,

$B_{i+1}=I$, $b_{i+1}=0$, $i=1,\cdots,N$, $f_3=\chi_C$, we obtain the following scheme by applying (2.1):

$$\text{(Scheme 4)}\quad \begin{cases} x^{k+1/2}=\mathrm{proj}_C(x^k-\lambda(D^Tv_1^k+\sum_{j=1}^N v_{2j}^k)),\\ v_1^{k+1}=(I-\mathrm{prox}_{\frac{\gamma}{\lambda}\mu\|\cdot\|_1})(Dx^{k+1/2}+v_1^k),\\ v_{2i}^{k+1}=(x^{k+1/2}+v_{2i}^k)-(I+\frac{\gamma}{\lambda}A_i^TA_i)^{-1}\Big(\frac{\gamma}{\lambda}A_i^Ta_i+x^{k+1/2}+v_{2i}^k\Big),\\ \qquad\qquad i=1,\cdots,N,\\ x^{k+1}=\mathrm{proj}_C\Bigg(x^k-\lambda\Big(D^Tv_1^{k+1}+\sum_{j=1}^N v_{2j}^{k+1}\Big)\Bigg), \end{cases} \tag{2.10}$$

where $0<\lambda<1/(\lambda_{\max}(DD^T)+N)$ and $0<\gamma<+\infty$. At first glance, the size of the inverse in the third equation in (2.10) is the same with the third ones in (2.7). However, thanks to the well known Sherman-Morrison-Woodbury formula, we know

$$\left(I+\frac{\gamma}{\lambda}A_i^TA_i\right)^{-1}=I-\frac{\gamma}{\lambda}A_i^T\left(I+\frac{\gamma}{\lambda}A_iA_i^T\right)^{-1}A_i. \tag{2.11}$$

So we only need to invert a smaller size matrix $I+\frac{\gamma}{\lambda}A_iA_i^T$ instead of $I+\frac{\gamma}{\lambda}A_i^TA_i$. By using (2.11), the third equation in (2.10) is equivalent to

$$v_{2i}^{k+1}=\frac{\gamma}{\lambda}A_i^T\left(I+\frac{\gamma}{\lambda}A_iA_i^T\right)^{-1}A_i\left(\frac{\gamma}{\lambda}A_i^Ta_i+x^{k+1/2}+v_{2i}^k\right)-\frac{\gamma}{\lambda}A_i^Ta_i. \tag{2.12}$$

It is worth to point out that the quantities v_1^{k+1} and v_{2i}^{k+1} $(1\le i\le N)$ can be evaluated in parallel.

3. PDFP for Constrained Muti-block Problem (1.2)

3.1. Algorithms and its deduction

In this subsection, we will show how to extend PDFP to solve (1.2). (1.2) can be also seen as a special case of (1.4) by using operator B and vector b, so we can solve it with PDFP. As a matter of fact, by using the separability of f_1, f_2 and f_3 about their variables, respectively, we can get the primal-dual fixed point algorithm (3.2) for solving (1.2).

As a special case of indicator function χ_C on convex set C, for $C=\{0\}$, we define

$$\chi_0(x)=\begin{cases}0, & x=0,\\ +\infty, & x\neq 0.\end{cases}$$

Then (1.2) is equivalent to

$$\min_{x_1,\cdots,x_N}\sum_{i=N_1+1}^N\theta_i(x_i)+\left(\sum_{i=1}^{N_1}\theta_i(B_ix_i+b_i)+\chi_0(\sum_{i=1}^N A_ix_i-a)\right)+\sum_{i=1}^N\chi_{C_i}(x_i). \tag{3.1}$$

Let

$$f_1(x)=f_1(x_1,\cdots,x_N)=\sum_{i=N_1+1}^N\theta_i(x_i),$$

$$f_3(x)=f_3(x_1,\cdots,x_N)=\sum_{i=1}^N\chi_{C_i}(x_i).$$

We also let

$$y_i = B_i x_i + b_i, \quad i = 1, \cdots, N_1, \quad y_{N_1+1} = \sum_{i=1}^{N} A_i x_i - a,$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_{N_1} \\ y_{N_1+1} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_{N_1} \\ \vdots \\ x_N \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_{N_1} \\ -a \end{pmatrix},$$

$$B = \begin{pmatrix} B_1 & & & & \\ & \ddots & & & \\ & & B_{N_1} & & \\ A_1 & \cdots & A_{N_1} & \cdots & A_N \end{pmatrix},$$

$$f_2(y) = f_2(y_1, y_2, \cdots, y_{N_1}, y_{N_1+1}) = \sum_{i=1}^{N_1} \theta_i(y_i) + \chi_0(y_{N_1+1}).$$

Then we have

$$y = Bx + b, \quad f_2(Bx + b) = \sum_{i=1}^{N_1} \theta_i(B_i x_i + b_i) + \chi_0\Big(\sum_{i=1}^{N} A_i x_i - a\Big),$$

and problem (1.2) can be viewed as a special case of problem (1.4). Hence, we can use PDFP for solving (1.2). Observing that f_2 is separable about its variables $y_1, y_2, \cdots, y_{N_1+1}$, f_1 and f_3 are separable about their variables $x_1, x_2, \cdots, x_N$, $\text{prox}_{\frac{\gamma}{\lambda}\chi_{C_i}} = \text{proj}_{C_i}$, $\text{prox}_{\frac{\gamma}{\lambda}\chi_0}(w) = \text{proj}_0(w) = 0$ for all $w \in \mathbb{R}^l$, we have by (1.5) that

$$\begin{cases} x_i^{k+1/2} = \text{proj}_{C_i}\Big(x_i^k - \lambda(B_i^T v_i^k + A_i^T v_{N_1+1}^k)\Big), & i = 1, \cdots, N_1, \quad (3.2a) \\ x_i^{k+1/2} = \text{proj}_{C_i}\Big(x_i^k - \gamma \nabla\theta_i(x_i^k) - \lambda A_i^T v_{N_1+1}^k\Big), & i = N_1+1, \cdots, N, \quad (3.2b) \\ v_i^{k+1} = (I - \text{prox}_{\frac{\gamma}{\lambda}\theta_i})\Big(B_i x_i^{k+1/2} + b_i + v_i^k\Big), & i = 1, \cdots, N_1, \quad (3.2c) \\ v_{N_1+1}^{k+1} = \sum_{j=1}^{N} A_j x_j^{k+1/2} - a + v_{N_1+1}^k, & (3.2d) \\ x_i^{k+1} = \text{proj}_{C_i}\Big(x_i^k - \lambda(B_i^T v_i^{k+1} + A_i^T v_{N_1+1}^{k+1})\Big), & i = 1, \cdots, N_1, \quad (3.2e) \\ x_i^{k+1} = \text{proj}_{C_i}\Big(x_i^k - \gamma\nabla\theta_i(x_i^k) - \lambda A_i^T v_{N_1+1}^{k+1}\Big), & i = N_1+1, \cdots, N. \quad (3.2f) \end{cases}$$

The convergence condition of PDFP in [5] implies that the above algorithm is convergent whenever $0 < \lambda < 1/(\sum_{i=1}^{N} \lambda_{\max}(A_i A_i^T) + \max\{\lambda_{\max}(B_i B_i^T), i = 1, 2, \cdots, N_1\})$ and $0 < \gamma < 2\min\{\beta_i, i = N_1+1, N_1+2, \cdots, N\}$. It is easy to see that (3.2a)-(3.2b), (3.2c)-(3.2d) and (3.2e)-(3.2f) can be implemented in parallel, respectively. Also for some special cases, such as $N_1 = 0$ and $N_1 = N$, one may even get simpler forms from (3.2). Let $N_1 = N$, $B_i = I$ and

$b_i = 0$ in (3.2), we then have

$$\begin{cases} x_i^{k+1/2} = \mathrm{proj}_{C_i}\Big(x_i^k - \lambda(v_i^k + A_i^T v_{N+1}^k)\Big), \quad i = 1, \cdots, N, & (3.3a) \\ v_i^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\theta_i})\Big(x_i^{k+1/2} + v_i^k\Big), \quad i = 1, \cdots, N, & (3.3b) \\ v_{N+1}^{k+1} = \sum_{j=1}^N A_j x_j^{k+1/2} - a + v_{N+1}^k, & (3.3c) \\ x_i^{k+1} = \mathrm{proj}_{C_i}\Big(x_i^k - \lambda(v_i^{k+1} + A_i^T v_{N+1}^{k+1})\Big), \quad i = 1, \cdots, N, & (3.3d) \end{cases}$$

for solving (1.3), where $0 < \lambda < 1/(\sum_{i=1}^N \lambda_{\max}(A_i A_i^T) + 1)$ and $0 < \gamma < +\infty$. The scheme of (3.2), including (3.3), can be implemented in parallel, and there is no requirement for the subproblem solving if the proximity operator of θ_i have the closed-form representation.

For solving (1.3), we can also get many others algorithms, by viewing parts of θ_i as f_1, parts of θ_i as $f_2 \circ B$ and parts of θ_i as f_3. Here we just give an example to show the idea. Let

$$\begin{aligned} & f_1(x) = 0, \quad f_2(y) = \chi_0(y), \\ & B = (A_1, A_2, \cdots, A_N), \quad b = -a, \quad y = Bx + b, \\ & f_3(x) = f_3(x_1, x_2, \cdots, x_N) = \sum_{i=1}^N (\theta_i(x_i) + \chi_{C_i}(x_i)). \end{aligned}$$

Due to the separability of f_3, PDFP (1.5) can be further expressed as

$$\begin{cases} x_i^{k+1/2} = \underset{x_i \in C_i}{\mathrm{argmin}}\ \theta_i(x_i) + \frac{1}{2\gamma}\|x_i - (x_i^k - \lambda A_i^T v^k)\|^2, \quad i = 1, \cdots, N, & (3.4a) \\ v^{k+1} = v^k + \Big(\sum_{j=1}^N A_j x_j^{k+1/2} - a\Big), & (3.4b) \\ x_i^{k+1} = \underset{x_i \in C_i}{\mathrm{argmin}}\ \theta_i(x_i) + \frac{1}{2\gamma}\|x_i - (x_i^k - \lambda A_i^T v^{k+1})\|^2, \quad i = 1, \cdots, N, & (3.4c) \end{cases}$$

where $0 < \lambda < 1/\sum_{i=1}^N \lambda_{\max}(A_i A_i^T)$ and $0 < \gamma < +\infty$. We can write the explicit solution of (3.4a) and (3.4c) for some special θ_i and C_i, for example $\theta_i = \|\cdot\|_1$ and C_i are rectangular domains. If $C_i = \mathbb{R}^{n_i}$, for the schemes (3.4a) and (3.4c), we just need to work out the proximity operator of θ_i. So the scheme is parallel and easy to implement for solving (1.3), which is the basic problem considered in the context of ADMM.

We can write the problem (1.2) (or problem (1.1)) in the form (1.4) with many other ways, and then derive new schemes to solve it in terms of PDFP (1.5). Since the discussion is routine, we omit the details. What we have to emphasize is that our method for constructing algorithms for solving problem (1.2) or (1.1) is very flexible.

3.2. Comparison to ADMM-like algorithms

In this subsection, we show the difference between the classic ADMM and PDFP for (1.3) by solving the following typical problem:

$$\begin{aligned} & \min \theta_1(x_1) + \theta_2(x_2) + \theta_3(x_3) \\ & \text{st. } A_1 x_1 + A_2 x_2 + A_3 x_3 = a, \\ & \quad x_1 \in C_1,\ x_2 \in C_2,\ x_3 \in C_3, \end{aligned} \tag{3.5}$$

where $x_i \in \mathbb{R}^{n_i}, i = 1, 2, 3$. For the ADMM method, the above problem is first transformed to solve the following min-max problem:

$$\begin{aligned}&\min_{x_1\in C_1, x_2\in C_2, x_3\in C_3} \max_{w} \quad \mathcal{L}_\alpha(x_1, x_2, x_3, w)\\ &= \sum_{i=1}^{3} \theta_i(x_i) - \langle w, \sum_{i=1}^{3} A_i x_i - a\rangle + \frac{\alpha}{2}\|\sum_{i=1}^{3} A_i x_i - a\|^2.\end{aligned} \tag{3.6}$$

Let $v = w/\alpha$. We then use the alternating direction method to solve problem (3.6), leading to the following algorithm

$$\begin{cases} x_1^{k+1} = \operatorname*{argmin}_{x_1\in C_1} \theta_1(x_1) + \frac{\alpha}{2}\|A_1x_1 + (A_2x_2^k + A_3x_3^k - v^k - a)\|^2, & (3.7a)\\ x_2^{k+1} = \operatorname*{argmin}_{x_2\in C_2} \theta_2(x_2) + \frac{\alpha}{2}\|A_2x_2 + (A_1x_1^{k+1} + A_3x_3^k - v^k - a)\|^2, & (3.7b)\\ x_3^{k+1} = \operatorname*{argmin}_{x_3\in C_3} \theta_3(x_3) + \frac{\alpha}{2}\|A_3x_3 + (A_1x_1^{k+1} + A_2x_2^{k+1} - v^k - a)\|^2, & (3.7c)\\ v^{k+1} = v^k - \tau(A_1x_1^{k+1} + A_2x_2^{k+1} + A_3x_3^{k+1} - a). & (3.7d)\end{cases}$$

In general, (3.7a)-(3.7c) need to solve subprograms whenever $A_i \neq I$ and the scheme is not a parallel algorithm. In addition, if one of (3.7a)-(3.7b) is not easy to solve due to the constraints C_i, we must introduce new auxiliary variables to get the solution. Though the treatment is routine, the solution process will become rather complicated. More importantly, as showed in [2], the scheme (3.7) is not necessarily convergent if there is no further assumption on (3.5). Recently it is popular to propose some variants of ADMM to overcome this disadvantage, for example, some prediction-correction methods were proposed in [11], and the Jacobian decomposition of augmented Lagrangian method (ALM) with proximal terms was introduced in [12]. A very interesting "primal-only" splitting scheme are also be used for solving similar problems in [10].

Now, we continue to show how to solve (3.5) in view of PDFP. By using indicator functions, (3.5) is equivalent to

$$\min \sum_{i=1}^{3} \theta_i(x_i) + \chi_0\Big(\sum_{i=1}^{3} A_i x_i - a\Big) + \sum_{i=1}^{3} \chi_{C_i}(x_i). \tag{3.8}$$

Then we can use PDFP to solve (3.8) in various forms. For example, by setting $N = 3$ in (3.4), we can get the following algorithm

$$\begin{cases} x_i^{k+1/2} = \operatorname*{argmin}_{x_i\in C_i} \theta_i(x_i) + \frac{1}{2\gamma}\|x_i - (x_i^k - \lambda A_i^T v^k)\|^2, \quad i = 1, 2, 3, & (3.9a)\\ v^{k+1} = v^k + (A_1x_1^{k+1/2} + A_2x_2^{k+1/2} + A_3x_3^{k+1/2} - a), & (3.9b)\\ x_i^{k+1} = \operatorname*{argmin}_{x_i\in C_i} \theta_i(x_i) + \frac{1}{2\gamma}\|x_i - (x_i^k - \lambda A_i^T v^{k+1})\|^2, \quad i = 1, 2, 3, & (3.9c)\end{cases}$$

where $0 < \lambda < 1/\sum_{i=1}^{3} \lambda_{\max}(A_iA_i^T)$ and $0 < \gamma < +\infty$. compared with the scheme of (3.7), the scheme of (3.9) is parallel and always convergent. Nevertheless, the computational cost increases with the addition of a symmetric step, which may double the work of each step. To avoid the disadvantage, we can also extend the scheme in [1, 8, 13, 14] with the same treatment given above.

When the subproblems in (3.9a) are not easy to solve due to the constraints C_i, we can also use (3.3) and get

$$\begin{cases} x_i^{k+1/2} = \mathrm{proj}_{C_i}\Big(x_i^k - \lambda(v_i^k + A_i^T v_4^k)\Big), & i=1,2,3, \quad (3.10a)\\ v_i^{k+1} = (I - \mathrm{prox}_{\frac{\gamma}{\lambda}\theta_i})\Big(x_i^{k+1/2} + v_i^k\Big), & i=1,2,3, \quad (3.10b)\\ v_4^{k+1} = v_4^k + \Big(A_1 x_1^{k+1/2} + A_2 x_2^{k+1/2} + A_3 x_3^{k+1/2} - a\Big), & (3.10c)\\ x_i^{k+1} = \mathrm{proj}_{C_i}\Big(x_i^k - \lambda(v_i^k + A_i^T v_4^{k+1})\Big), & i=1,2,3, \quad (3.10d)\end{cases}$$

where $0<\lambda<1/(\sum_{i=1}^3 \lambda_{\max}(A_iA_i^T)+1)$ and $0<\gamma<+\infty$.

If θ_i are both differentiable with $1/\beta_i$-Lipschitz continuous gradient, respectively. We can set $N_1=0$ and $N=3$ in (3.2), and then obtain the following scheme

$$\begin{cases} x_i^{k+1/2} = \mathrm{proj}_{C_i}\Big(x_i^k - \gamma\nabla\theta_i(x_i^k) - \lambda A_i^T v^k\Big), & i=1,2,3, \quad (3.11a)\\ v^{k+1} = v^k + \Big(A_1 x_1^{k+1/2} + A_2 x_2^{k+1/2} + A_3 x_3^{k+1/2} - a\Big), & (3.11b)\\ x_i^{k+1} = \mathrm{proj}_{C_i}\Big(x_i^k - \gamma\nabla\theta_i(x_i^k) - \lambda A_i^T v^{k+1}\Big), & i=1,2,3, \quad (3.11c)\end{cases}$$

where $0<\lambda<1/\sum_{i=1}^3 \lambda_{\max}(A_iA_i^T)$ and $0<\gamma<2\min\{\beta_1,\beta_2,\beta_3\}$.

4. Numerical Experiments

In this section, we will illustrate the application of PDFP for multi-block problems through two examples, related to (1.1) and (1.2), respectively. The first one is the total variation regularized computerized tomography (CT) reconstruction with constraints, and the second one is on some quadratic programming or linear equation examples given in [2] as the counter examples for the convergence of thee-block ADMM.

4.1. CT reconstruction

The standard CT reconstruction algorithm in clinical applications is the so-called Filtered Back Projection (FBP) algorithm. In the presence of noise, this problem becomes difficult since the inverse of Radon transform is unbounded and ill-posed. In the literature, the model is constructed based on TV regularization (2.3), i.e.

$$\min_{x\in C} \ \frac{1}{2}\|Ax-a\|^2 + \mu\|Dx\|_1.$$

Here A is the Radon transform matrix, a is the measured projections vector, and D is the discrete gradient operator. The size of A is generally huge and it is very difficult for us to efficiently solve a linear system with A as the coefficient matrix. $\|Dx\|_1$ is the usual ℓ_1 based regularization in order to promote sparsity under the transform D and $\mu>0$ is the regularization parameter. To be more precise, we use the isotropic total variation as the regularization term, and assume that the solution should belong to [0,255], in other words, the constraint set is defined as $C=\{x=(x_1,x_2,\cdots,x_n)^T\in\mathbb{R}^n | x_i\in[0,255], i=1,2,\cdots,n\}$. We have shown in [4] that it is useful to impose the above constraints in CT to improve the quality of reconstructed images.

In our numerical simulation, we still use the same example tested in [17], i.e., 50 uniformly oriented projections are simulated for a 128×128 Shepp-Logan phantom image and then white Gaussian noise of mean 0 and variance 1 is added to the data. For this example, we set $\mu = 0.05$ and work out $\lambda_{\max}(AA^T) = 1.5086$. It is well known in total variation application that $\lambda_{\max}(DD^T) = 8$. So we set $\gamma = 1.3$, $\lambda = 1/8$ ($0 < \gamma < 2/\lambda_{\max}(AA^T) = 1.3257$ and $0 < \lambda < 1/8$ in PDFP according to Theorem 3.1 in [5] in Scheme 1 (cf. (2.5)). Correspondingly we set $\gamma = 1.3$, $\lambda = 1/9$ in Scheme 2 (cf. (2.6)). Set $\gamma = 20$ and $\lambda = 1/(8 + 1.5086)$ in Scheme 3 (cf. (2.8)). Set $\gamma = 100$, $\lambda = 1/(8 + N)$, $N = 20$ in Scheme 4 (cf. (2.10) and (2.12)). Here we do not implement (2.7) since it needs to solve a large linear system, nor (2.9) as it is a parallel form of Scheme 3.

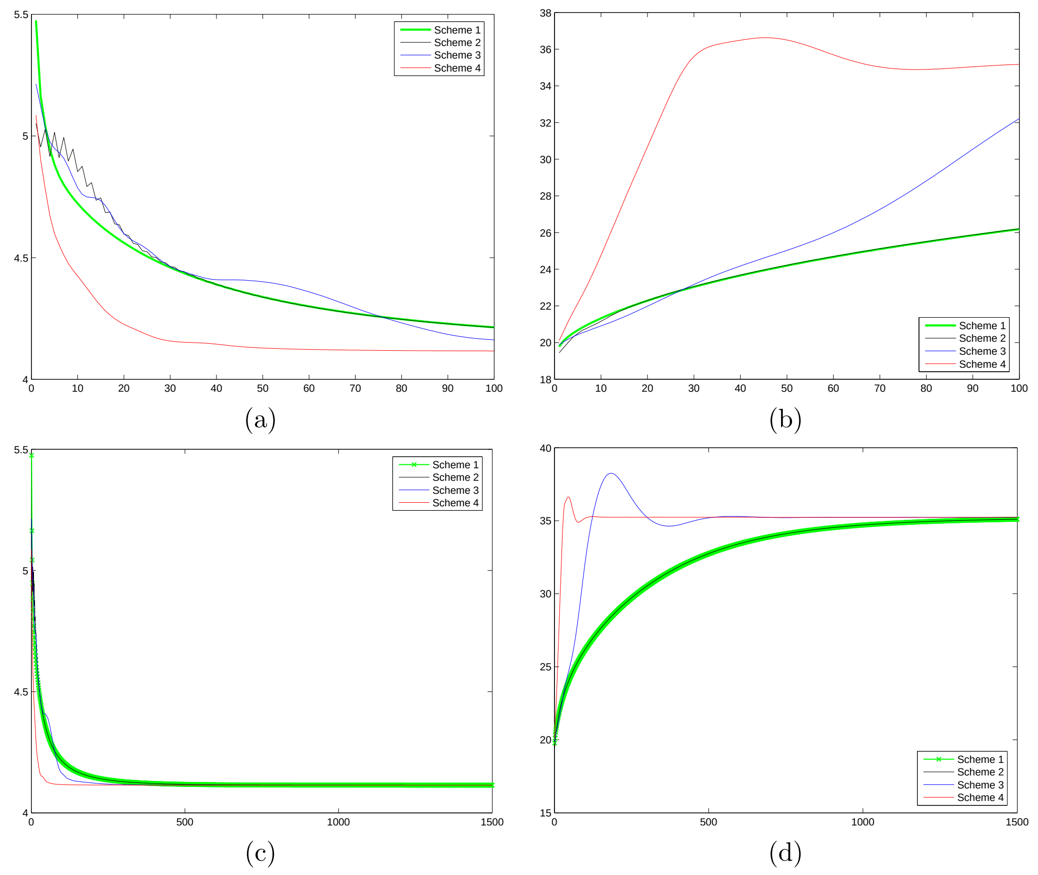
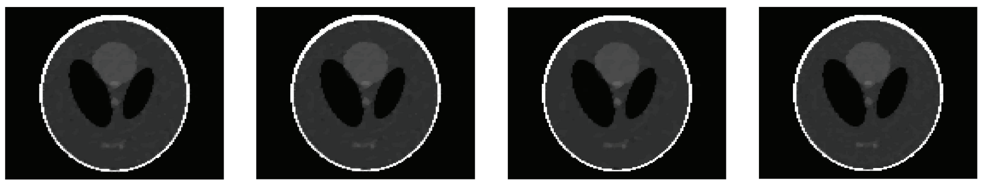


Fig. 4.1. Evolution of four different PDFP algorithms in CT reconstruction. From left to right: (a) $\log_{10}$(energy) *versus* 100 iterations; (b) PSNR *versus* 100 iterations; (c) $\log_{10}$(energy) *versus* 1500 iterations; (d) PSNR *versus* 1500 iterations.

From Figure 4.1, we can see that Scheme 1 and Scheme 2 have similar performance except for the first steps. The final results of the four schemes are very close, and Scheme 3 and Scheme 4 can produce higher PSNR results than those from Scheme 1 and Scheme 2. The other point to be emphasized is that, compared with Scheme 1 and Scheme 2, Scheme 3 and Scheme 4 can get relatively better results with higher PSNR while use far less iteration steps, as shown in Figure 4.2. In deriving Scheme 3 and Scheme 4, we view $\frac{1}{2}\|Ax - a\|^2$ as a part of $f_2 \circ B$ in (1.4), and thus f_1 is taken to be *zero*, which leads to faster convergent algorithms. Compared with



	Scheme 1	Scheme 2	Scheme 3	Scheme 4
itn	1500	1500	183	45
time	35.33	34.76	4.29	23.53
PSNR	35.0997	35.1003	38.2467	36.6266

Fig. 4.2. The best recovery results for CT in 1500 iterations.

the numerical results from Scheme 3 and Scheme 4, we may find Scheme 4 requires much less iteration numbers to obtain a solution of problem (2.3), since the preconditioning technique is used implicitly. But it requires more computational cost due to linear systems solving in sub-steps. Here, we simply use MATLAB command "\" for solving these systems. If they can be solved more efficiently based on the structures of coefficient matrices, the efficiency of Scheme 4 could be greatly improved. A clear advantage of Scheme 3 is that it mainly involves matrix-vector multiplication operations, and it can also lead to a even better solution quickly as shown in this example.

4.2. Application to non convergent examples for the direct extension of ADMM

As showed in [2], the direct extension scheme (3.7) is not necessarily convergent if there is no further assumption on (3.5). Some non-convergent examples of ADMM are given in [2]. Now, there have developed many algorithms overcoming the non-convergence of the classic ADMM (cf. [9–12]). Since our PDFP is convergent for a general problem (1.2) according to the theory in [5], here we are interested in studying the numerical performance of this method for solving those non-convergent examples in [2].

The first example is solving linear equation

$$\begin{pmatrix}1\\1\\1\end{pmatrix} x_1 + \begin{pmatrix}1\\1\\2\end{pmatrix} x_2 + \begin{pmatrix}1\\2\\2\end{pmatrix} x_3 = \begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{4.1}$$

(4.1) is a special case of (3.5), where

$$A = (A_1, A_2, A_3) = \begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 2\\ 1 & 2 & 2\end{pmatrix}, \quad a = \begin{pmatrix}0\\0\\0\end{pmatrix}.$$

It is easy to verify that A is nonsingular, and the true solution is $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$. Moreover, the corresponding optimal Lagrange multipliers are all 0. Let $\theta_i = 0$ and $C_i = \mathbb{R}$, i=1,2,3 in (3.11), (3.10) or (3.9), we can get the following scheme to solve it. Namely

$$\begin{cases} x_i^{k+1/2} = x_i^k - \lambda A_i^T v^k, \quad i = 1, 2, 3, & \text{(4.2a)}\\ v^{k+1} = v^k + \Big(A_1 x_1^{k+1/2} + A_2 x_2^{k+1/2} + A_3 x_3^{k+1/2} - a\Big), & \text{(4.2b)}\\ x_i^{k+1} = x_i^k - \lambda A_i^T v^{k+1}, \quad i = 1, 2, 3, & \text{(4.2c)} \end{cases}$$

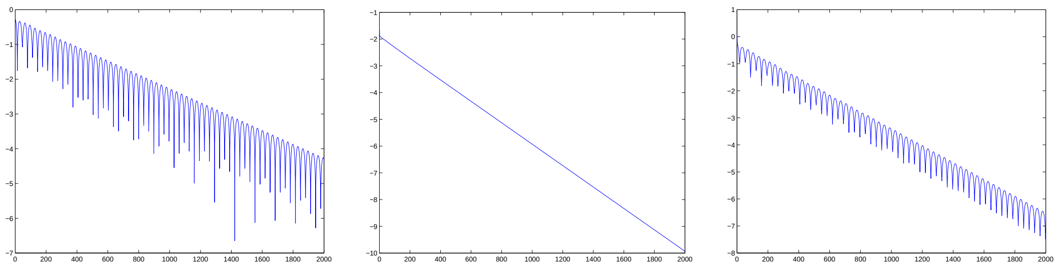

(a) Example 1 (b) Example 2 (c) Example 3

Fig. 4.3. The ℓ_2 norms of errors *versus* iterations for PDFP within 2000 steps.

where $0 < \lambda \le 1/\sum_{i=1}^{3} \lambda_{\max}(A_i^T A_i)$. Substitute $x_i^{k+1/2}$ with x_i^{k+1}, and (4.2) implies

$$\begin{cases} x_i^{k+1} = x_i^k - \lambda A_i^T \Big(A_1 x_1^k + A_2 x_2^k + A_3 x_3^k - a \Big) - \lambda A_i^T v^k, \quad i = 1,2,3, & \text{(4.3a)} \\ v^{k+1} = v^k + \Big(A_1 x_1^{k+1} + A_2 x_2^{k+1} + A_3 x_3^{k+1} - a \Big), & \text{(4.3b)} \end{cases}$$

i.e.,

$$x_i^{k+1} = x_i^k - \lambda A_i^T \sum_{j=0}^{k} \Big(A_1 x_1^j + A_2 x_2^j + A_3 x_3^j - a \Big) - \lambda A_i^T v^0, \quad i = 1,2,3. \tag{4.4}$$

We set $\lambda = 1/\sum_{i=1}^{3} \lambda_{\max}(A_i^T A_i)$ and the initial values of the elements of x_i and v as 1.

The second example is solving

$$\min \quad 0.05x_1^2 + 0.05x_2^2 + 0.05x_3^2 \tag{4.5a}$$

$$\text{st.} \quad \begin{pmatrix}1\\1\\1\end{pmatrix} x_1 + \begin{pmatrix}1\\1\\2\end{pmatrix} x_2 + \begin{pmatrix}1\\2\\2\end{pmatrix} x_3 = \begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{4.5b}$$

(4.5) is also a special case of (3.5). Let $\theta_i = 0.05x_i^2$ and $C_i = \mathbb{R}$, $i = 1,2,3$ in (3.11), we can easily get the algorithm for solving (4.5), where $0 < \lambda \le 1/\sum_{i=1}^{3} \lambda_{\max}(A_i^T A_i)$ and $0 < \gamma < 2/0.1 = 20$. According to the convergence rate theory about PDFP^{2}O given in [3], this algorithm has linear convergence rate, which is also confirmed by Figure 4.3(b) , since $f_1(x) = f_1(x_1, x_2, x_3) = \sum_{i=1}^{3} 0.05x_i^2$ is strongly convex and $BB^T = AA^T$ is positive symmetric definite in (1.4) with $f_3 = 0$. Set $\gamma = 1/0.1 = 10$ and the others setting are the same as the first example.

The third example given in [2] is more sophisticated. It can be described as

$$\min \quad 0.5x_1^2 \tag{4.6a}$$

$$\text{st.} \quad \begin{pmatrix}1\\1\\1\end{pmatrix} x_1 + \begin{pmatrix}1\\1\\1\end{pmatrix} x_2 + \begin{pmatrix}1\\1\\2\end{pmatrix} x_3 + \begin{pmatrix}1\\2\\2\end{pmatrix} x_4 = \begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{4.6b}$$

The feasible region of (4.6) is not a singleton, and the objective function is only related with x_1. The optimal solution of (4.6) is $x_i = 0$, $i = 1,2,3,4$. Similar to (3.11), let $N_1 = 0$, $N = 4$, $\theta_1 = 0.5x_1^2$, $\theta_i = 0$, $i = 2,3,4$, $C_i = \mathbb{R}$, $i = 1,2,3,4$ in (3.2), we can easily derive the algorithm for solving (4.6), where $0 < \lambda \le 1/\sum_{i=1}^{4} \lambda_{\max}(A_i^T A_i)$ and $0 < \gamma < 2$. We set $\lambda = 1/\sum_{i=1}^{4} \lambda_{\max}(A_i^T A_i)$ and $\gamma = 1$. The others setting are same as the first example.

The errors with respect to the true solution within 2000 steps are given in Figure 4.3. The convergence phenomenon are very interesting for the three examples. For Example 2, the method converges without oscillation; but for the other two ones, the iteration solutions converge with high oscillations. In what follows, we will give a rigorous explanation to reveal the numerical insights. First of all, we have by a direct manipulation that the PDFP in these examples can be expressed as

$$\begin{pmatrix} v^{k+1} \\ x^{k+1} \end{pmatrix} = \begin{pmatrix} I - \lambda A A^T & A(I-\gamma F) \\ -\lambda A^T(I - \lambda A^T A) & (I - \lambda A^T A)(I - \gamma F))A^T \end{pmatrix} \begin{pmatrix} v^k \\ x^k \end{pmatrix} + \begin{pmatrix} -a \\ \lambda A^T a \end{pmatrix}, \tag{4.7}$$

where $F = \mathrm{diag}(0,0,0)$, $F = \mathrm{diag}(1,1,1)$ and $F = \mathrm{diag}(1,0,0,0)$ for these three examples, respectively. For simplicity, we denote by T the iteration matrix in the above iteration method. Therefore, the corresponding errors satisfy that

$$e^k = \mathrm{T}^k e^0,$$

where $e^k = [(v^k - v^*)^T, (x^k - x^*)^T]^T$ stands for the k-th iteration error to the true solution.

By direct calculations, we know the eigenvalues of T for Examples 1, 2, 3 for the given λ and γ are (0.0284 + 0.1661i, 0.0284 - 0.1661i, 0.9908 + 0.0954i, 0.9908 - 0.0954i, 0.9808 + 0.1373i, 0.9808 - 0.1373i), (0, 0, 0, 0.0284, 0.9808, 0.9908) and (0, 0.0992 + 0.1427i, 0.0992 - 0.1427i, 0.9833 + 0.1273i, 0.9833 - 0.1273i, 0.9889 + 0.0881i, 0.9889 - 0.0881i), respectively. We can find that the largest modulus of the eigenvalues of the iteration matrices for the three examples are less than 1, so they are all convergent linearly. However, for Example 2, the eigenvalue with the largest modulus is a real number, while the eigenvalues with the largest modulus are a pair of conjugate complex numbers for the other two examples. Then, by the eigenbasis representation of e^0 and standard Power method calculation, we know that for Example 2, the ℓ_2 norms of errors converge in proportional to cr^k, where r is the largest eigenvalue of T; for the other two examples, the ℓ_2 norms of errors behave like $cr^k|\cos(k\varphi + \psi)|$, with $r\mathrm{e}^{\pm \mathrm{i}\varphi}$ denoting the two conjugate complex eigenvalues of T with the largest modulus r and ψ a constant. Hence, we can observe the phenomenon mentioned before.

5. Conclusion

We extend the ideas of a primal-dual fixed point algorithm PDFP to solve separable multi-block minimization problems with or without linear constraints. The variants of PDFP are fully splitting and therefore easy to implement. Moreover, the algorithms are parallel naturally, so they are very suitable for solving large-scale problems from real-world models. Through numerical experiments, we can see that treating smooth functions as parts of $f_2 \circ B$ leads to better convergence and partial inverse can be viewed as preconditioning for a good balance of convergence speed and computational cost. The convergence condition on the parameter γ is an arbitrary positive number, while the choice might heavily affect the convergent speed, which may make parameter choosing a difficult problem in practice. Therefore the proper decomposition of the smooth functions and non-smooth functions, and explicit or implicit schemes should depend on the properties and computation balances in real applications. Moreover, for problems with constraints that classic ADMM may fail to converge, PDFP algorithm can be also a choice with the guarantee of theoretical convergence. Finally, we point out that this article only discusses synchronous parallel realization of the proposed algorithms. Actually, it is possible and valuable to design the related asynchronous algorithms. We refer the reader to [15] and references therein for details along this line.

Acknowledgements. The authors thank the referee for helpful and valuable comments which greatly improves the early version of the article. P. Chen was partially supported by the PhD research startup foundation of Taiyuan University of Science and Technology (No. 20132024). J. Huang was partially supported by NSFC (No. 11571237). X. Zhang was partially supported by NSFC (No. 91330102 and GZ1025) and 973 program (No. 2015CB856004).

References

[1] Chambolle A, Pock T, A first-order primal-dual algorithm for convex problems with applications to imaging, *Journal of Mathematical Imaging and Vision*, **40**:1 (2011), 120-145.

[2] Chen C, He B, Ye Y, et al., The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent, *Mathematical Programming*, (2014), 1-23.

[3] Chen P, Huang J, Zhang X, A primal-dual fixed point algorithm for convex separable minimization with applications to image restoration, *Inverse Problems*, **29**:2 (2013), 025011.

[4] Chen P, Huang J, Zhang X, A primal-dual fixed point algorithm based on proximity operator for convex set constrained separable problem, *Journal of Nanjing Normal University (Natural Science Edition)*, **36**:3 (2013), 1-5.

[5] Chen P, Huang J, Zhang X, A primal-dual fixed point algorithm for minimization of the sum of three convex separable functions, arXiv preprint arXiv: 1512.09235, 2015.

[6] Clason C, Kunisch K, A duality-based approach to elliptic control problems in non-reflexive Banach spaces, *ESAIM: Control, Optimisation and Calculus of Variations*, **17**:01 (2011), 243-266.

[7] Combettes P L, Wajs V R, Signal recovery by proximal forward-backward splitting, *Multiscale Modeling & Simulation*, **4**:4 (2005), 1168-1200.

[8] Condat L, A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms, *Journal of Optimization Theory and Applications*, **158**:2 (2013), 460-479.

[9] Deng W, Lai M J, Peng Z, et al., Parallel multi-block ADMM with $O(1/k)$ convergence, arXiv preprint arXiv: 1312.3040, 2013.

[10] Davis D, Yin W, A three-operator splitting scheme and its optimization applications, arXiv preprint arXiv: 1504.01032, 2015.

[11] He B, Tao M, Yuan X, Alternating direction method with Gaussian back substitution for separable convex programming, *SIAM Journal on Optimization*, **22**:2 (2012), 313-340.

[12] He B S, Xu H K, Yuan X M, On the proximal Jacobian decomposition of ALM for multiple-block separable convex minimization problems and its relationship to ADMM. 2013-11-21]. http://www.optimization-online.org/D B_FILE/2013/11/4142. pdf, 2013.

[13] Li Q, Shen L, Xu Y, et al., Multi-step fixed-point proximity algorithms for solving a class of optimization problems arising from image processing, *Advances in Computational Mathematics*, **41**:2 (2015), 387-422.

[14] Li Q, Zhang N, Fast proximity-gradient algorithms for structured convex optimization problems, Applied and Computational Harmonic Analysis, 2015.

[15] Peng Z, Wu T, Xu Y, et al., Coordinate friendly structures, algorithms and applications, arXiv preprint arXiv: 1601.00863, 2016.

[16] Tang Y C, Zhu C X, Wen M, et al., A splitting primal-dual proximity algorithm for solving composite optimization problems, arXiv preprint arXiv: 1507.08413, 2015.

[17] Zhang X, Burger M, Osher S, A unified primal-dual algorithm framework based on Bregman iteration, *Journal of Scientific Computing*, **46**:1 (2011), 20-46.