J. Appl. Numer. Optim. 6 (2024), No. 1, pp. 83-96
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.6.2024.1.05

SOLVING SPLIT FEASIBILITY PROBLEMS VIA BLOCK-WISE FORMULATION

ZHOU WANG, HONGJIN HE*

School of Mathematics and Statistics, Ningbo University, Ningbo, 315211, China

Abstract. The split feasibility problem (SFP), which provides a unified framework to model a wide range of inverse problems, has received much considerable attention in the literature. However, how to efficiently solve SFPs is still an interesting topic. In this paper, we introduce a block-wise formulation for algorithmic design. Specifically, we first introduce an auxiliary variable to formulate the original SFP as a constrained minimization problem with a block structure, which paves a new way to find solutions of SFPs. Then, we show that the employments of some classical gradient-type optimization algorithms produce very simple, yet quite efficient iterative schemes to find a solution of SFPs when the underlying block structure could be exploited. The parallel iterative schemes of the proposed block-wise algorithms are not only efficient to deal with the case that the projections onto the convex sets have explicit representations, but also are possibly valuable for solving large-scale SFPs without explicit projections onto the underlying sets. Some numerical results on synthetic examples support the idea of this paper.

Keywords. CQ algorithm; Convergence rate; Gradient method; Heavy-ball method; Split feasibility problem.

1. INTRODUCTION

The classical Split Feasibility Problem (SFP) was first introduced by Censor and Elfving [1] in 1994, which has been widely used in areas of image reconstructions, intensity-modulated radiation therapy, and statistical learning; see, e.g., [2, 3, 4], to name just a few. Mathematically, the SFP refers to the task of finding a point x^* such that

$$x^* \in \boldsymbol{C} \quad \text{and} \quad Ax^* \in \boldsymbol{Q}, \tag{1.1}$$

where $\boldsymbol{C} \subseteq \mathbb{R}^n$ and $\boldsymbol{Q} \subseteq \mathbb{R}^m$ are two nonempty, closed, and convex sets, respectively, and A is assumed to be an $m \times n$ real matrix. More generally, such a problem has also been studied in Hilbert spaces [5, 6]. Actually, the SFP can be regarded as a natural generalization of the classical Linear Inverse Problem (LIP):

$$Ax = b, \tag{1.2}$$

when $\boldsymbol{C}$ and $\boldsymbol{Q}$ in (1.1) are specified as $\boldsymbol{C} = \mathbb{R}^n$ and $\boldsymbol{Q} = \{b\}$ ($b \in \mathbb{R}^m$), respectively. Comparatively, solving the SFP (1.1) is more difficult than the LIP (1.2) due to the possibly complicated

*Corresponding author.
E-mail address: wangzhmath@163.com (Z. Wang), hehongjin@nbu.edu.cn (H. He).
Received 31 December 2023; Accepted 16 February 2024; Published online 5 March 2024.

convex sets $\boldsymbol{C}$ and $\boldsymbol{Q}$. However, the SFP (1.1) provides a unified treatment framework, which is helpful for us to design customized algorithms for solving some real-world problems [4, 7].

In the literature, one of the most popular solvers is the so-named CQ algorithm, which was originally introduced by Byrne [8, 9]. Specifically, the iterative scheme of the CQ algorithm reads as

$$x^{k+1} = P_{\boldsymbol{C}}\left(x^k - \theta A^\top (I - P_{\boldsymbol{Q}})Ax^k\right), \tag{1.3}$$

where θ is an appropriate stepsize satisfying $\theta \in \left(0, \frac{2}{L}\right)$ with L being the largest eigenvalue of matrix $A^\top A$, and both $P_{\boldsymbol{C}}(\cdot)$ and $P_{\boldsymbol{Q}}(\cdot)$ denote the projections onto the convex sets $\boldsymbol{C}$ and $\boldsymbol{Q}$, respectively. It is interesting that iterative scheme (1.3) can be understood as a specific application of the classical gradient-projection method [6] to the following equivalent formulation of (1.1), i.e.,

$$\min_x \left\{ f(x) := \frac{1}{2}\left\|Ax - P_{\boldsymbol{Q}}(Ax)\right\|^2 \quad | \quad x \in \boldsymbol{C} \right\}. \tag{1.4}$$

To improve the efficiency of solving (1.1), we here refer the reader to [10, 11, 12, 13] for some classical variants of the CQ algorithm based on (1.4). Actually, as shown in [14], the CQ algorithm can also be interpreted as specific applications of some other optimization algorithms, such as partially linearized alternating minimization algorithms, DC (Difference-of-Convex functions) algorithms, fixed-point methods, and majorization-minimization algorithms, when reformulating the SFP (1.1) as different types of minimization problems. With the help of these optimization forms, some regularized SFPs promoting structured (e.g., sparse or low-rank) solutions can be efficiently solved; see, e.g., [4, 7, 15]. Roughly speaking, when taking a revisit on those existing algorithms tailored for SFPs, they can be grouped into two model-based algorithms. The first group is based on one single variable x such as (1.4), and these single-variable model-based algorithms usually enjoy quite simple iterative schemes and run fast as long as the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$ are simple. The other group is developed by the model with two variables, where one auxiliary variable y is introduced for the purpose of separating Ax and $\boldsymbol{Q}$. Such two-variable model-based algorithms usually update two variables in an alternating (Gauss-Seidel) way so that they usually perform well in practice. Overall, the aforementioned algorithms can be regarded as alternating projection algorithms since they must calculate the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$ in a sequential order. Consequently, those algorithms possibly take much computing time to find a solution when the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$ are not easy. Therefore, two natural questions are raised naturally. The first one is that can we combine the ideas of these two groups? If yes, what will be produced by the resulting algorithms? The second one is that can we devise parallel projection algorithms in the sense that we can simultaneously calculate the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$? In this paper, our main goal is to answer the above questions.

When introducing an auxiliary variable y to split Ax and $\boldsymbol{Q}$, we accordingly obtain an equality constraint $Ax - y = 0$. In this way, we immediately obtain an equivalent formulation of (1.1) as follows:

$$\min_u \left\{ \varphi(u) := \frac{1}{2}\|Ax - y\|^2 \equiv \frac{1}{2}\|Bu\|^2 \quad | \quad u \in \Omega := \boldsymbol{C} \times \boldsymbol{Q} \right\}, \tag{1.5}$$

where $B := (A, -I)$ is a block matrix, and $u := (x^\top, y^\top)^\top$, which will be denoted by $u = (x, y)$ for notional simplicity. Clearly, the $\varphi(u)$ given in (1.5) is a standard quadratic function and the set Ω enjoys a Cartesian structure. In this situation, we can gainfully employ the classical

gradient-type methods to solve (1.1), thereby possibly producing some more efficient algorithms. Therefore, we first apply the classical gradient-projection method to (1.5) and obtain a block-wise CQ algorithm (see Algorithm 1, denoted by BCQ). Moreover, inspired by [16], we can employ the popular Nesterov acceleration technique [17] to derive an accelerated block-wise CQ algorithm (see Algorithm 2, denoted by ABCQ). Considering the case where $\boldsymbol{C}$ and/or $\boldsymbol{Q}$ is nonconvex, we further employ the well-known heavy-ball acceleration strategy [18, 19] to develop a fast heavy-ball block-wise CQ algorithm (see Algorithm 3, denoted by HBCQ). It is interesting that the proposed three algorithms not only enjoy quite simple iterative schemes, but also are able to calculate the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$ in a parallel way due to the Cartesian structure of Ω. Comparatively, our proposed algorithms are more suitable to implement the code on parallel machines for dealing with the case where the projections onto $\boldsymbol{C}$ and $\boldsymbol{Q}$ have no explicit forms. Some preliminary computational results on synthetic datasets demonstrate that the proposed algorithms work well in practice.

The rest of this paper is organized as follows. In Section 2, we recall some notations and definitions that are used in this paper. In Section 3, we propose the BCQ and ABCQ algorithms to deal with convex SFPs. In Section 4, we first propose an HBCQ algorithm to solve nonconvex SFPs. Then, we show that the sequence generated by the HBCQ algorithm converges to a critical point of the problem under consideration. In Section 5, we report some preliminary numerical results to support our idea. Finally, we give some concluding remarks to complete this paper in Section 6.

2. Preliminaries

Let $\mathbb{R}^n$ be an n-dimensional Euclidean space. For any two vectors $x, y \in \mathbb{R}^n$, let $\langle x, y\rangle = x^\top y$ represent the standard inner product, where the superscript $\top$ stands for the transpose of a vector or matrix. For a given $x \in \mathbb{R}^n$, we denote by $\|x\|_p$ the standard ℓ_p-norm of x whose value is $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$ for $1 \le p < \infty$. Particularly, we use $\|x\| = \|x\|_2$ for simplicity. For a matrix $A \in \mathbb{R}^{m\times n}$, let $\|A\|$ denote the spectral norm. Additionally, we let $\|\cdot\|_0$ represent the number of nonzero components.

The projection from $\mathbb{R}^n$ onto a nonempty subset $\mathbb{K}$ of $\mathbb{R}^n$, denoted by $P_{\mathbb{K}}(\cdot)$, is defined by

$$P_{\mathbb{K}}(x) = \arg\min\{\|y - x\| \mid y \in \mathbb{K}\}. \tag{2.1}$$

When $\mathbb{K}$ is further assumed to be closed and convex, projection mapping (2.1) is a singleton for any $x \in \mathbb{R}^n$. Moreover, for a nonempty set $\mathbb{K} \subset \mathbb{R}^n$, the distance from a point $x \in \mathbb{R}^n$ to $\mathbb{K}$ is defined by

$$\operatorname{dist}(x, \mathbb{K}) := \inf\{\|x - z\| \mid z \in \mathbb{K}\}.$$

Definition 2.1. Let $h(\cdot) : \mathbb{R}^n \to \mathbb{R}$ be a lower semicontinuous convex function. The subdifferential of h is the set of all subgradients of h at $x \in \mathbb{R}^n$, and is denoted by $\partial h(x)$:

$$\partial h(x) := \{\xi \in \mathbb{R}^n \mid h(y) \ge h(x) + \langle \xi, y - x\rangle, \ \ \forall y \in \mathbb{R}^n\}.$$

Definition 2.2. Let $g(\cdot) : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. Its gradient $\nabla g(\cdot)$ is said to be Lipschitz continuous with constant $L_g > 0$ if

$$\|\nabla g(x) - \nabla g(y)\| \le L_g \|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Clearly, the function $f(x)$ defined in (1.4) and the function $\varphi(u)$ given in (1.5) are Lipschitz continuous with Lipschitz constants $\|A\|^2$ and $\|B\|^2 = \|A\|^2 + 1$, respectively. In this sense, we say that (1.4) and (1.5) are well-defined problems, and the Lipschitz continuity is of benefit for algorithmic design. Moreover, for a general Lipschitz continuous function $f(x)$, we have the following well-known descent lemma.

Lemma 2.1. *Let $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and its gradient ∇f be Lipschitz continuous with constant $L > 0$. Then,*

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2, \quad \forall x, y \in \mathbb{R}^n. \tag{2.2}$$

To end this section, we recall the Kurdyka-Łojasiewicz (KŁ) property [20, 21, 22], which is a fundamental tool for dealing with nonconvex optimization problems.

Definition 2.3 (KŁ property and KŁ function)**.** We say that a proper closed function h satisfies the KŁ property at $\tilde{x} \in \operatorname{dom} \partial h := \{x \in \mathbb{R}^n \mid \partial h(x) \neq \emptyset\}$ if there exist $\eta \in (0, \infty]$, a neighborhood U of $\tilde{x}$, and a function $\Gamma \in \Xi_\eta$ such that, for all x in the intersection

$$U \bigcap \{x \in \mathbb{R}^n : h(\tilde{x}) < h(x) < h(\tilde{x}) + \eta\},$$

it holds that

$$\Gamma'(h(x) - h(\tilde{x}))\operatorname{dist}(0, \partial h(x)) \geq 1,$$

where Ξ_η is the set of all concave continuous functions $\Gamma : [0, \eta) \to [0, \infty)$ being continuously differentiable over $(0, \eta)$ with positive derivatives and satisfying $\Gamma(0) = 0$. Accordingly, if h satisfies the KŁ property at any point of $\operatorname{dom}\partial h$, then h is called KŁ function.

3. The BCQ and ABCQ Algorithms for Convex SFP

In this section, we show concrete iterative schemes of the block-wise CQ (BCQ) algorithm and the accelerated block-wise CQ (ABCQ) algorithm. Then, we present the iteration complexity of both algorithms in accordance with the results in [16].

As mentioned in Section 1, the traditional CQ algorithm is indeed an application of the gradient-projection method to (1.4). So, we similarly apply the gradient-projection method to (1.5), thereby immediately resulting in the following iterative scheme:

$$u^{k+1} = P_\Omega\left(u^k - \frac{1}{\alpha} B^\top B u^k\right), \tag{3.1}$$

where α is an appropriate positive parameter. Note that there are many choices on α to improve the performance of (3.1) in the literature. However, the main purpose of this paper is to show that our block-wise formulation (1.5) is helpful to produce more efficient algorithm than the formulation (1.4). Therefore, we here just consider constant stepsizes for (3.1), and further require $\alpha \geq \|B\|^2 \equiv \|A\|^2 + 1$ to ensure the global convergence due to the fact $BB^\top = AA^\top + I$ and $\|B\|^2 = \|BB^\top\|$ (see [23]), where I is an identity matrix.

By recalling the notation u and B and invoking the Cartesian structure of Ω in (1.5), we immediately obtain a new block-wise CQ algorithm for (1.1), which is called BCQ algorithm and is shown in Algorithm 1.

Since Algorithm 1 is an application of the gradient-projection method, and can also be regarded as a special case of [16, Algorithm 1], we immediately establish the $O(1/k)$ convergence

Algorithm 1 The block-wise CQ (BCQ) algorithm for (1.1).

1: Take starting points $x^0 \in \boldsymbol{C}$ and $y^0 \in \boldsymbol{Q}$. Set $\alpha \geq \|A\|^2 + 1$.
2: **for** $k = 1, 2, \cdots$ **do**
3: $\quad x^{k+1} = P_{\boldsymbol{C}}\left(x^k - \frac{1}{\alpha}(A^\top A x^k - A^\top y^k)\right)$.
4: $\quad y^{k+1} = P_{\boldsymbol{Q}}\left(y^k - \frac{1}{\alpha}(y^k - A x^k)\right)$.
5: **end for**

rate of Algorithm 1. For sake of conciseness of this paper, we here skip the detailed proof. The reader is referred to [16, Theorem 4.3] for details.

Theorem 3.1. *Suppose that $\alpha \geq \|A\|^2 + 1$. Let u^* be an arbitrary solution of (1.5), and let $\{u^k\}$ be a sequence generated by Algorithm 1 (BCQ algorithm). Then, for any $k \geq 1$,*

$$\varphi(u^k) - \varphi(u^*) \leq \frac{\alpha \left\|u^0 - u^*\right\|^2}{2k}.$$

Obviously, the above theorem means that obtaining an ε-optimal solution $u^\star$ takes the number of iterations at most $\lceil \varsigma/\varepsilon \rceil$ such that $\varphi(u^\star) - \varphi(u^*) \leq \varepsilon$, where $\varsigma := \frac{\alpha\|u^0 - u^*\|^2}{2}$. Therefore, we know that Algorithm 1 has the $O(1/k)$ iteration complexity. As shown in [16], the CQ algorithm can be accelerated via the well-known Nesterov acceleration technique [17]. Therefore, we also propose an accelerated variant of Algorithm 1 based upon the block-wise model (1.5). Specifically, the accelerated block-wise CQ (ABCQ) algorithm is described in Algorithm 2.

Algorithm 2 The accelerated block-wise CQ (ABCQ) algorithm for (1.1).

1: Take starting points $x^0, p^1 \in \boldsymbol{C}$ and $y^0, q^1 \in \boldsymbol{Q}$. Set $t_1 = 1$ and $\alpha \geq \|A\|^2 + 1$.
2: **for** $k = 1, 2, \cdots$ **do**
3: $\quad x^k = P_{\boldsymbol{C}}\left(p^k - \frac{1}{\alpha}(A^\top A p^k - A^\top q^k)\right)$.
4: $\quad y^k = P_{\boldsymbol{Q}}\left(q^k - \frac{1}{\alpha}(q^k - A p^k)\right)$.
5: $\quad t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2}$.
6: $\quad p^{k+1} = x^k + \dfrac{t_k - 1}{t_{k+1}}\left(x^k - x^{k-1}\right)$.
7: $\quad q^{k+1} = y^k + \dfrac{t_k - 1}{t_{k+1}}\left(y^k - y^{k-1}\right)$.
8: **end for**

Like Theorem 3.1, we can establish its $O(1/k^2)$ convergence rate from [16, Theorem 4.8]. Also, we skip the proof here for the simplicity of this paper.

Theorem 3.2. *Suppose that $\alpha \geq \|A\|^2 + 1$. Let u^* be an arbitrary solution of (1.5), and let $\{u^k\}$ be a sequence generated by Algorithm 2 (ABCQ algorithm). Then, for any $k \geq 1$,*

$$\varphi(u^k) - \varphi(u^*) \leq \frac{2\alpha \left\|u^0 - u^*\right\|^2}{(k+1)^2}.$$

The above theorem shows that if we would like to obtain an ε-optimal solution $u^\star$ satisfying $\varphi(u^\star)-\varphi(u^*)\leq\varepsilon$, then the number of iterations required by Algorithm 2 is at most $\lceil\widehat{\varsigma}/\sqrt{\varepsilon}-1\rceil$, where $\widehat{\varsigma}:=\sqrt{2\alpha}\|u^0-u^*\|^2$. In other words, Algorithm 1 has the $O(1/k^2)$ iteration complexity.

Remark 3.1. Many real-world problems often pursue some structured (e.g., sparse and low-rank) solutions. Then, attaching an appropriate (possibly nonsmooth) regularization term to (1.1) is a natural way. Accordingly, the regularized SFP is modelled as

$$\min_x \{\phi(x) \mid x\in\boldsymbol{C} \text{ and } Ax\in\boldsymbol{Q}\}. \tag{3.2}$$

Recently, closely related problems of (3.2) were studied in [4, 7, 16, 24, 25, 26]. Interestingly, we do not worry about the applicability of our algorithms (Algorithms 1 and 2) to (3.2), since we can easily follow the block-wise formulation of (1.5) to transform (3.2) into the following one:

$$\min_u \left\{\varphi(u):=\phi(x)+\frac{\lambda}{2}\|Ax-y\|^2\equiv\phi(x)+\frac{\lambda}{2}\|Bu\|^2 \mid u\in\Omega\right\}, \tag{3.3}$$

where u, B, and Ω are given in (1.5), and $\lambda>0$ is a regularization parameter. In this case, the update of x^{k+1} and y^{k+1} in Algorithm 1 are modified as

$$\begin{cases} x^{k+1}=\arg\min\limits_{x\in\boldsymbol{C}}\left\{\phi(x)+\dfrac{\alpha}{2}\left\|x-\left(x^k-\dfrac{\lambda}{\alpha}\left(A^\top Ax^k-A^\top y^k\right)\right)\right\|^2\right\},\\ y^{k+1}=P_{\boldsymbol{Q}}\left(y^k-\dfrac{\lambda}{\alpha}\left(y^k-Ax^k\right)\right), \end{cases}$$

and Algorithm 2 updates x^k and y^k via

$$\begin{cases} x^{k}=\arg\min\limits_{x\in\boldsymbol{C}}\left\{\phi(x)+\dfrac{\alpha}{2}\left\|x-\left(p^k-\dfrac{\lambda}{\alpha}\left(A^\top Ap^k-A^\top q^k\right)\right)\right\|^2\right\},\\ y^{k}=P_{\boldsymbol{Q}}\left(q^k-\dfrac{\lambda}{\alpha}\left(q^k-Ap^k\right)\right) \end{cases}$$

for solving (3.3).

4. The HBCQ Algorithm for Nonconvex SFP

Considering that nonconvex sets, e.g., $\mathbb{S}:=\{x\in\mathbb{R}^n \mid \|x\|_0\leq s\}$ with s being a positive integer to control the sparsity of x, frequently appear in many real-world problems, in this section, we further introduce a heavy-ball block-wise CQ algorithm for nonconvex SFPs, where $\boldsymbol{C}$ and $\boldsymbol{Q}$ are allowed to be nonconvex sets.

The classical heavy-ball method [18, 19] was originally designed for unconstrained optimization problems. Recently, Ochs [27] extended the heavy-ball method to composite optimization problems, where the objective function is the sum of two (not necessarily) convex functions. Accordingly, by reformulating (1.5) as a composite optimization problem with the help of the indicator function associated to Ω, i.e.,

$$\min\{F(u):=\delta_\Omega(u)+\varphi(u)\}, \tag{4.1}$$

where $\delta_\Omega(\cdot)$ is the indicator function. Then, by setting the $g(x)$ in [27] as an indicator function, a customized application of the heavy-ball method to (1.5) immediately yields the following iterative scheme:

$$u^{k+1} = P_\Omega\left(u^k - \mu B^\top B u^k + \tau(u^k - u^{k-1})\right), \tag{4.2}$$

where μ and τ are appropriate parameters. More concretely, by using the notations of u and B, and invoking the Cartesian structure of Ω, the iterative scheme can be described in Algorithm 3, which is called heavy-ball block-wise CQ algorithm.

Algorithm 3 The heavy-ball block-wise CQ (HBCQ) algorithm for (1.1).

1: Take $x^0, x^1 \in \boldsymbol{C}$ and $y^0, y^1 \in \boldsymbol{Q}$. Set $\tau \in [0, 1/2)$ and $\mu \in \left(0, (1-2\tau)/(\|A\|^2+1)\right)$.
2: **for** $k = 1, 2, \cdots$ **do**
3: $\quad x^{k+1} = P_{\boldsymbol{C}}\left(x^k - \mu\left(A^\top A x^k - A^\top y^k\right) + \tau\left(x^k - x^{k-1}\right)\right)$.
4: $\quad y^{k+1} = P_{\boldsymbol{Q}}\left(y^k - \mu\left(y^k - A x^k\right) + \tau\left(y^k - y^{k-1}\right)\right)$.
5: **end for**

Remark 4.1. The parameter τ is indeed an extrapolation parameter. Clearly, when taking $\tau = 0$, Algorithm 3 immediately reduces to Algorithm 1. On the other hand, according to [27], the extrapolation parameter τ can be further relaxed to $[0, 1)$ for the case where $\boldsymbol{C}$ and $\boldsymbol{Q}$ are convex sets. Additionally, as discussed in Remark 3.1, Algorithm 3 is also applicable to the regularized SFPs (3.2) based on block-wise formulation (3.3).

Below, we give the convergence result of Algorithm 3 according to [27, 28]. Since our model is a special case of the problem considered in [28], we just prove the descent property of the sequence generated by Algorithm 3, and omit the proof of the main theorem, which can be referred to [28].

Before the proof, we first define an auxiliary function

$$H_\sigma(u, v) := \delta_\Omega(u) + \varphi(u) + \sigma\|u - v\|^2, \tag{4.3}$$

where $\sigma > 0$ and v is a given vector. Clearly, when $u = v$, it is clear that $H_\sigma(u, v) = F(u)$.

Lemma 4.1. *Let $\{u^k\}$ be a sequence generated by Algorithm 3. Then, for any $k > 1$, there exists a constant $\gamma > 0$ such that*

$$\gamma\left\|u^k - u^{k-1}\right\|^2 \le H_\sigma(u^k, u^{k-1}) - H_\sigma(u^{k+1}, u^k).$$

Proof. The iterative scheme (4.2) can be written as

$$u^{k+1} \in \arg\min_u \left\{\delta_\Omega(u) + \left\langle \mu B^\top B u^k - \tau(u^k - u^{k-1}), u - u^k\right\rangle + \frac{1}{2}\|u - u^k\|^2\right\}, \tag{4.4}$$

which implies that

$$\delta_\Omega(u^{k+1}) + \left\langle B^\top B u^k - \frac{\tau}{\mu}(u^k - u^{k-1}), u^{k+1} - u^k\right\rangle + \frac{1}{2\mu}\|u^{k+1} - u^k\|^2 \le \delta_\Omega(u^k). \tag{4.5}$$

On the other hand, applying Lemma 2.1 to $\varphi(u)$ yields

$$\varphi(u^{k+1}) \le \varphi(u^k) + \left\langle B^\top B u^k, u^{k+1} - u^k\right\rangle + \frac{L}{2}\|u^{k+1} - u^k\|^2, \tag{4.6}$$

where $L := \|A\|^2 + 1$. Consequently, combining (4.5) and (4.6) yields

$$\begin{aligned}
&\varphi(u^{k+1}) + \delta_\Omega(u^{k+1}) \\
&\le \varphi(u^k) + \delta_\Omega(u^k) + \left\langle \frac{\tau}{\mu}(u^k - u^{k-1}), u^{k+1} - u^k \right\rangle + \frac{L\mu - 1}{2\mu}\|u^{k+1} - u^k\|^2 \\
&\le \varphi(u^k) + \delta_\Omega(u^k) + \frac{\tau}{2\mu}\|u^k - u^{k-1}\|^2 + \frac{\tau}{2\mu}\|u^{k+1} - u^k\|^2 + \frac{L\mu - 1}{2\mu}\|u^{k+1} - u^k\|^2 \\
&= \varphi(u^k) + \delta_\Omega(u^k) + \left(\frac{\tau}{2\mu} + \gamma\right)\|u^k - u^{k-1}\|^2 - \gamma\|u^k - u^{k-1}\|^2 + \frac{\tau + L\mu - 1}{2\mu}\|u^{k+1} - u^k\|^2 \\
&= \varphi(u^k) + \delta_\Omega(u^k) + \sigma\|u^k - u^{k-1}\|^2 - \gamma\|u^k - u^{k-1}\|^2 - \sigma\|u^{k+1} - u^k\|^2 \qquad (4.7)
\end{aligned}$$

where the second inequality comes from the fact $\langle a, b\rangle \le \frac{1}{2}\left(\|a\|^2 + \|b\|^2\right)$ and the last equality follows from the notations $\sigma = \frac{\tau}{2\mu} + \gamma$ with $\gamma = \frac{1-2\tau-L\mu}{2\mu}$. Noting that $\tau \in [0,1)$ and $\mu \in \left(0, (1-2\tau)/(\|A\|^2+1)\right)$, it is easy to verify $\gamma > 0$ with $L := \|A\|^2 + 1$. As a result, rewriting (4.7) immediately leads to

$$\varphi(u^{k+1}) + \delta_\Omega(u^{k+1}) + \sigma\|u^{k+1} - u^k\|^2 \le \varphi(u^k) + \delta_\Omega(u^k) + \sigma\|u^k - u^{k-1}\|^2 - \gamma\|u^k - u^{k-1}\|^2,$$

which, together with (4.3), concludes

$$\gamma\left\|u^k - u^{k-1}\right\|^2 \le H_\sigma(u^k, u^{k-1}) - H_\sigma(u^{k+1}, u^k). \qquad (4.8)$$

Therefore, we complete the proof. □

For simplicity, we denote $w := (w_u, w_v)^\top \in \partial H_\sigma(u, v)$, where

$$w_u \in \partial\delta_\Omega(u) + \nabla\varphi(u) + 2\sigma(u - v) \quad \text{and} \quad w_v \in -2\sigma(u - v). \qquad (4.9)$$

Lemma 4.2. *Let $\{u^k\}$ be a sequence generated by Algorithm 3. Then, for any $k > 1$, there exists a constant c such that*

$$\|w^{k+1}\| \le c\left(\|u^k - u^{k-1}\| + \|u^{k+1} - u^k\|\right).$$

Proof. Recalling the notation in (4.9), we have

$$w^{k+1} := (w_u^{k+1}, w_v^{k+1})^\top \in \partial H_\sigma(u^{k+1}, u^k), \qquad (4.10)$$

where $w_u^{k+1} \in \partial\delta_\Omega(u^{k+1}) + \nabla\varphi(u^{k+1}) + 2\sigma(u^{k+1} - u^k)$ and $w_v^{k+1} \in -2\sigma(u^{k+1} - u^k)$. Writing the optimality condition of (4.4) yields that

$$0 \in \partial\delta_\Omega(u^{k+1}) + \nabla\varphi(u^k) - \frac{\tau}{\mu}(u^k - u^{k-1}) + \frac{1}{\mu}(u^{k+1} - u^k). \qquad (4.11)$$

Combining (4.10) and (4.11) leads to

$$\begin{aligned}
\|w^{k+1}\| &\le \|w_u^{k+1}\| + \|w_v^{k+1}\| \\
&\le \|\nabla\varphi(u^{k+1}) - \nabla\varphi(u^k)\| + \left(\frac{1}{\mu} + 4\sigma\right)\|u^{k+1} - u^k\| + \frac{\tau}{\mu}\|u^k - u^{k-1}\| \\
&\le \frac{\mu L + 1 + 4\mu\sigma}{\mu}\|u^{k+1} - u^k\| + \frac{\tau}{\mu}\|u^k - u^{k-1}\|, \qquad (4.12)
\end{aligned}$$

where the last inequality follows from the Lipschitz continuous of $\nabla\varphi$ with Lipschitz constant $L = \|A\|^2 + 1$. Therefore, we conclude from (4.12) that there exists a constant $c := \min\left\{\frac{\mu L+1+4\mu\sigma}{\mu}, \frac{\tau}{\mu}\right\}$ such that

$$\|w^{k+1}\| \leq c(\|u^k - u^{k-1}\| + \|u^{k+1} - u^k\|).$$

Therefore, we complete the proof. □

Lemma 4.3. *Assume that the sequence $\{u^k\}$ generated by Algorithm 3 is bounded. Then, there exists a subsequence $\{u^{k_j}\}_{j\in N}$ such that, as $j \to \infty$,*

$$(u^{k_j+1}, u^{k_j}) \to (u^*, u^*), \quad H_\sigma(u^{k_j+1}, u^{k_j}) \to H_\sigma(u^*, u^*),$$

where u^* is a critical point of (4.1).

Proof. Note that $\{u^k\}$ is assumed to be bounded. Therefore, there exist a subsequence $\{u^{k_j+1}\}_{j\in N}$ of $\{u^k\}$ and a cluster point u^* such that $\lim_{j\to\infty} u^{k_j} = u^*$. Now, we show that u^* is a critical point of (4.1). We first define

$$V^j = -\nabla\varphi(u^{k_j-1}) + \frac{\tau}{\mu}(u^{k_j-1} - u^{k_j-2}) + \frac{1}{2\mu}(u^{k_j-1} - u^{k_j}) + \nabla\varphi(u^{k_j}).$$

By (4.5), we have

$$\delta_\Omega(u^{k_j}) + \left\langle B^\top B u^{k_j-1} - \frac{\tau}{\mu}(u^{k_j-1} - u^{k_j-2}), u^{k_j} - u^{k_j-1} \right\rangle + \frac{1}{2\mu}\|u^{k_j} - u^{k_j-1}\|^2 \leq \delta_\Omega(u^{k_j-1}),$$

or equivalently,

$$\delta_\Omega(u^{k_j}) + \left\langle -B^\top B u^{k_j-1} + \frac{\tau}{\mu}(u^{k_j-1} - u^{k_j-2}) + \frac{1}{2\mu}(u^{k_j-1} - u^{k_j}), u^{k_j-1} - u^{k_j} \right\rangle \leq \delta_\Omega(u^{k_j-1}),$$

which, together with Definition 2.1 and the notion of $\varphi(u)$, implies

$$-\nabla\varphi(u^{k_j-1}) + \frac{\tau}{\mu}(u^{k_j-1} - u^{k_j-2}) + \frac{1}{2\mu}(u^{k_j-1} - u^{k_j}) \in \partial\delta_\Omega(u^{k_j}).$$

Consequently, we obtain

$$(u^{k_j}, V_j) \in \mathrm{Graph}(\partial F) := \{(u,t) \in \mathbb{R}^n \times \mathbb{R}^n | t \in \partial F(u)\}.$$

Furthermore, it holds that $\lim_{j\to\infty} u^{k_j} = u^*$. Due to the Lipschitz continuity of $\nabla\varphi$ and

$$\|V_j - 0\| \leq \frac{\tau}{\mu}\|u^{k_j-1} - u^{k_j-2}\| + \frac{1}{2\mu}\|u^{k_j-1} - u^{k_j}\| + \|\nabla\varphi(u^{k_j}) - \nabla\varphi(u^{k_j-1})\|,$$

we have $\lim_{j\to\infty} V_j = 0$. By the closure property of the subdifferential ∂F, we have $(u^*, 0) \in \mathrm{Graph}(\partial F)$, which means that u^* is a critical point of F. The closeness of Ω also implies $u^* \in \Omega$. Thus we can obtain $\lim_{j\to\infty} \delta_\Omega(u^{k_j}) = \delta_\Omega(u^*)$. Moreover, since φ is continuous, we have $\lim_{j\to\infty} \varphi(u^{k_j}) = \varphi(u^*)$. Then $\lim_{j\to\infty} F(u^{k_j}) = F(u^*)$. Summing inequality (4.8) from $k = 1$ to n, we obtain

$$\begin{aligned}\sum_{k=1}^{n} \gamma\|u^k - u^{k-1}\|^2 &\leq \sum_{k=1}^{n} \Big(H_\sigma(u^k, u^{k-1}) - H_\sigma(u^{k+1}, u^k)\Big) \\ &= H_\sigma(u^1, u^0) - H_\sigma(u^{n+1}, u^n) \leq H_\sigma(u^1, u^0) < \infty,\end{aligned}$$

where the second inequality follows from the nonnegativity of $H_\sigma(u,v)$, and the last one holds by u^0 and u^1 belonging to Ω. It follows that $\|u^k-u^{k-1}\|\to 0$ as $k\to\infty$. Consequently, we arrive at

$$\lim_{j\to\infty} H_\sigma(u^{k_j+1},u^{k_j}) = \lim_{j\to\infty}(F(u^{k_j+1})+\sigma\|u^{k_j+1}-u^{k_j}\|) = H_\sigma(u^*,u^*) = F(u^*).$$

The proof is complete. □

With the preparation of Lemmas 4.1-4.3, we have the following main convergence theorem of Algorithm 3, which can be proved by [28, Theorem 5.4], and its detailed proof is skipped here for the conciseness of this paper.

Theorem 4.1. *Let both $\boldsymbol{C}$ and $\boldsymbol{Q}$ be nonempty, closed, and semi-algebraic sets, and suppose that the sequence $\{u^k\}$ generated by Algorithm 3 is bounded. Then, $\{u^k\}$ has finite length, i.e., $\sum_{k=1}^{\infty}\left\|u^k-u^{k-1}\right\| < \infty$, and converges to a critical point u^* of (4.1).*

Proof. Note that $\boldsymbol{C}$ and $\boldsymbol{Q}$ are assumed to be semi-algebraic sets. Then, we have that the indicator function δ_Ω associated to Ω in (4.1) is semi-algebraic, which naturally implies that $F(u)$ and $H_\sigma(u,v)$ satisfy the KŁ property. Therefore, with the help of Lemmas 4.1-4.3, we can follow the proof of [28, Theorem 5.4] to show that the sequence $\{u^k\}$ generated by Algorithm 3 converges to a critical point of (4.1). Here, we skip the remainder proof for simplicity. □

5. Numerical Experiments

In this section, we report some numerical results to support the idea of this paper. Note that Algorithm 3 is also applicable to convex SFPs, we here only conduct the numerical performance of our proposed block-wise algorithms on solving convex SFPs. For notational convenience, we denote Algorithms 1, 2, and 3 by BCQ, ABCQ, and HBCQ, respectively. We also compare them with the original CQ algorithm (CQ for short, see (1.3)) and the accelerated CQ algorithm (ACQ for short, see [16]). All algorithms are implemented in MATLAB 2021a and all experiments are conducted on a 64-bit Windows personal computer with Intel(R) Core(TM) i5-10210U CPU@2.11GHz and 8GB of RAM.

We consider a synthetic SFP studied in [16]. Here, the problem is constructed by setting $\boldsymbol{C} := \{x\in\mathbb{R}^n \,|\, \|x\|\le 50\}$ and $\boldsymbol{Q} := \{z\in\mathbb{R}^m \,|\, l_i \le z_i \le u_i,\ i=1,2,\ldots,m\}$, where l_i and u_i are uniformly distributed in $(-20,-10)$ and $(50,100)$, respectively. The matrix A is also randomly generated by the following MATLAB scripts:

$$\mathsf{A = rand(m,n);\ [Q,\sim] = qr(A);\ S = 2000*rand(n,1);\ A = Q*diag(S)*Q'.}$$

For the algorithmic parameters, we set $\theta = 1.8/\|A\|^2$ for CQ, $\beta = \|A\|^2$ for ACQ as the default value in [16], $\alpha = \|A\|^2+1$ for BCQ and ABCQ, $\tau = 0.85$ and $\mu = 1/(\|A\|^2+1)$ for HBCQ. In our experiments, we consider two cases on the setting of matrix A. The first one corresponds to the case where A is a square matrix, i.e., $A\in\mathbb{R}^{n\times n}$. The other one is that A is a rectangle matrix, i.e., $A\in\mathbb{R}^{m\times n}$. For each case, we further consider six scenarios with different sizes. Specifically, when A is a square matrix, we consider $n = \{500,\ 800,\ 1000,\ 1500,\ 2000,\ 2500\}$. When A is a rectangle matrix, we consider $(m,n) = (512i, 1024i)$ with $i = 1,2,\ldots,6$. For the sake of fairness, we take

$$\max\left\{\left\|x^k - P_{\boldsymbol{C}}(x^k)\right\|, \left\|Ax^k - P_{\boldsymbol{Q}}(Ax^k)\right\|\right\} \le 10^{-6}$$

as the termination rule for all algorithms. Besides, considering the randomness of the data, we report the averaged performance of 10 random trails for each problem. All results are summarized in Table 1 and 2, where 'Iter' represents the number of iterations and 'Time' stands for the computing time in seconds. Clearly, we can see from Tables 1 and 2 that the proposed block-wise algorithms (BCQ, ABCQ, and HBCQ) perform well in practice. In particular, BCQ runs much faster than the original CQ algorithm, even we take a smaller step size for BCQ. On the other hand, ABCQ also performs better than ACQ. Although the HBCQ is an accelerated algorithm, it requires a little more iterations than the ABCQ. In our opinion, the main reason is that the Nesterov acceleration is an optimal accelerated strategy. However, the HBCQ still works better than the original ACQ for the case where A is a square matrix, while it has almost the same results for the case where A is a rectangle matrix. From these computational results, we can see that our block-wise algorithms are beneficial for improving the efficiency of solving SFPs.

TABLE 1. Numerical results for synthetic example with square matrix A.

	CQ	ACQ	BCQ	ABCQ	HBCQ
(m,n)	Iter / Time	Iter / Time	Iter / Time	Iter / Time	Iter / Time
$(500,500)$	486.2 / 0.025	81.1 / 0.007	382.1 / 0.019	50.5 /0.004	69.6 / 0.009
$(800,800)$	697.4 / 0.135	103.3 / 0.030	455.8 / 0.098	55.0 / 0.018	71.1 / 0.024
$(1000,1000)$	611.4 / 0.450	121.7 / 0.136	371.8 / 0.288	51.0 / 0.056	82.2 / 0.093
$(1500,1500)$	974.2 / 2.098	196.6 / 0.638	536.9 / 1.157	63.6 / 0.203	75.0 / 0.244
$(2000,2000)$	786.6 / 3.025	170.1 / 0.987	388.0 / 1.503	50.5 / 0.296	76.8 / 0.450
$(2500,2500)$	918.5 / 6.084	131.1 / 1.324	559.8 / 3.638	61.6 / 0.602	89.6 / 0.886

TABLE 2. Numerical results for synthetic example with rectangle matrix A.

	CQ	ACQ	BCQ	ABCQ	HBCQ
(m,n)	Iter / Time	Iter / Time	Iter / Time	Iter / Time	Iter / Time
$(512,1024)$	43.1 / 0.006	17.4 / 0.004	9.3 / 0.002	6.0 / 0.002	20.0 / 0.004
$(1024,2048)$	51.2 / 0.061	22.0 / 0.039	13.3 / 0.015	7.1 / 0.012	25.1 / 0.045
$(1536,3072)$	57.9 / 0.189	28.9 / 0.142	16.6 / 0.056	8.7 / 0.045	27.9 / 0.140
$(2048,4096)$	58.3 / 0.367	28.6 / 0.269	16.8 / 0.107	8.3 / 0.079	30.7 / 0.289
$(2560,5120)$	57.6 / 0.649	31.7 / 0.522	18.0 / 0.188	8.7 / 0.141	32.7 / 0.520
$(3072,6144)$	57.6 / 1.120	30.7/ 0.907	19.0 / 0.362	9.1 / 0.265	34.8 / 1.027

To see more detailed performance of our proposed algorithms on these random datasets, we further report the maximum, minimum and averaged iterations and computing time in Figures 1 and 2. It can be easily seen from these plots in Figures 1 and 2 that our block-wise algorithms (BCQ, ABCQ, and HBCQ) runs more stable than the original CQ and ACQ. Therefore, these results demonstrate that our block-wise formulation is helpful for solving SFPs, which also provides a new insight to solve some structured optimization problems.

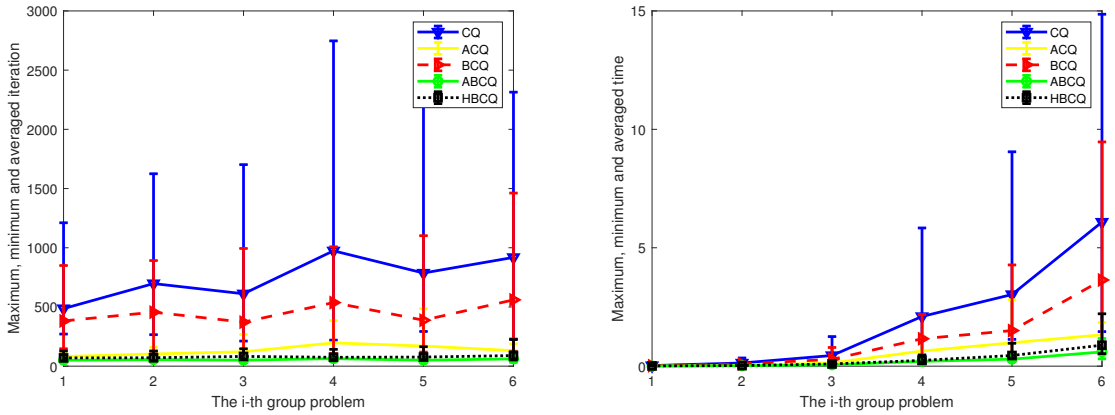

FIGURE 1. The maximum, minimum and averaged iterations and computing time for synthetic example with square matrix A.

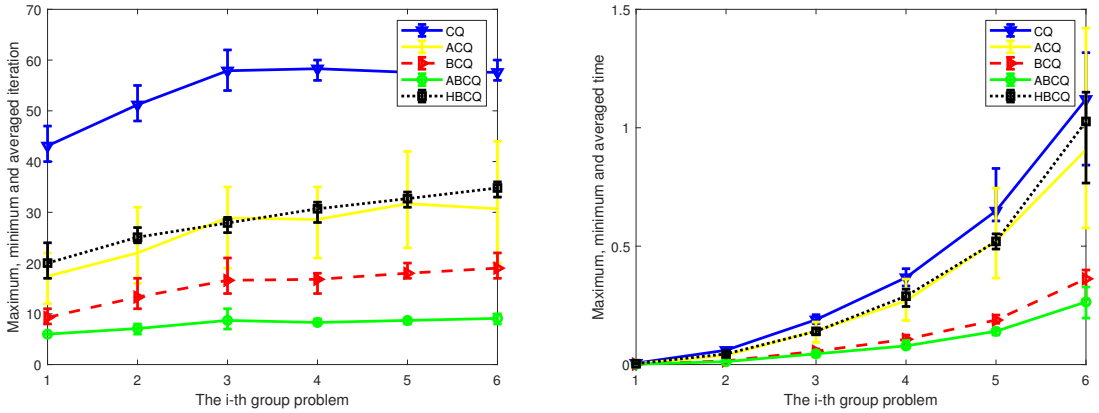

FIGURE 2. The maximum, minimum and averaged iterations and computing time for synthetic example with rectangle matrix A.

6. CONCLUSIONS

We considered the classical split feasibility problem (SFP), which is widely used in modeling inverse problems. In the past decades, many solvers have been developed, and most of them can be regarded as sequential (Gauss-Seidel) algorithms, which perform well on the cases where $\boldsymbol{C}$ and $\boldsymbol{Q}$ have explicit projections. In this paper, we introduced three new parallel (Jacobian) algorithm from a block-wise insight. Comparatively, our algorithms are able to exploit the advantages of modern parallel machines, which possibly are of benefit to improving the efficiency of solving large-scale SFPs with complicate structure. As an extension, we also studied a general accelerated algorithm, named by heavy-ball block-wise CQ algorithm, for solving nonconvex split feasibility problems. Some preliminary numerical results demonstrated that the proposed block-wise algorithms are efficient, which support the numerical improvements brought by the block-wise formulation for solving SFPs.

Acknowledgments

This research was supported in part by Zhejiang Provincial Natural Science Foundation of China (No. LZ24A010001), and Ningbo Natural Science Foundation (No. 2023J014).

REFERENCES

[1] Y. Censor, T. Elfving, A multiprojection algorithm using Bregman projections in a product space, Numer. Algorithms 8 (1994), 221-239.
[2] Y. Censor, T. Bortfeld, B. Martin, A. Trofimov, A unified approach for inversion problems in intensity-modulated radiation therapy, Phys. Med. Biol. 51 (2006), 2353-2365.
[3] Y. Censor, T. Elfving, N. Kopf, T. Bortfeld, The multiple-sets split feasibility problem and its applications for inverse problems, Inverse Probl. 21 (2005), 2071-2084.
[4] H.J. He, H.-K. Xu, Splitting methods for split feasibility problems with application to Dantzig selectors, Inverse Probl. 33 (2017), 055003.
[5] X. Qin, A. Petrusel, J.-C. Yao, CQ iterative algorithms for fixed points of nonexpansive mappings and split feasibility problems in Hilbert spaces, J. Nonlinear Convex Anal. 19 (2018), 157-165.
[6] H.-K. Xu, Iterative methods for the split feasibility problem in infinite-dimensional Hilbert spaces, Inverse Probl. 26 (2010), 105018.
[7] H.J. He, C. Ling, H.-K. Xu, An implementable splitting algorithm for the ℓ_1-norm regularized split feasibility problem, J. Sci. Comput. 67 (2016), 281-298.
[8] C.L. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, Inverse Probl. 20 (2004), 103-120.
[9] C.L. Byrne, Iterative oblique projection onto convex sets and the split feasibility problem, Inverse Probl. 18 (2002), 441-453.
[10] G. López, V. Martin-Marquez, F.H. Wang, H.K. Xu, Solving the split feasibility problem without prior knowledge of matrix norms, Inverse Probl. 28 (2012), 085004.
[11] B. Qu, N.-H. Xiu, A note on the CQ algorithm for the split feasibility problem, Inverse Probl. 21 (2005), 1655-1665.
[12] Q.Z. Yang, The relaxed CQ algorithm solving the split feasibility problem, Inverse Probl. 20 (2004), 1261-1266.
[13] H. Zhang, Y. Wang, A new CQ method for solving split feasibility problem, Front. Math. China 5 (2010), 37-46.
[14] T. Lu, L. Zhao, H. He, A multi-view on the CQ algorithm for split feasibility problems: from optimization lens, J. Appl. Numer. Optim. 2 (2020), 387-399.
[15] X. Mao, H.J. He, H.K. Xu, A partially proximal linearized alternating minimization method for finding Dantzig selectors, Comput. Appl. Math. 40 (2021), Article No. 62.
[16] P. Chen, H.J. He, Y.-C. Liou, C.-F. Wen, Convergence rate of the CQ algorithm for split feasibility problems, J. Nonlinear Convex Anal. 19 (2018), 381-395.
[17] Y. Nesterov, A method of solving a convex programming problem with convergence rate $O(1/k^2)$, Soviet Mathematics Doklady, 27 (1983), 372-376.
[18] B.T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (1964), 1-17.
[19] S. Zavriev, F. Kostyuk, Heavy-ball method in nonconvex optimization problems, Comput. Math. Model. 4 (1993), 336-341.
[20] H. Attouch, J. Bolte, On the convergence of the proximal algorithm for nonsmooth functions involving analytic features, Math. Program. 116 (2009), 5-16.
[21] H. Attouch, J. Bolte, B.F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods, Math. Program. 137 (2013), 91-129.
[22] J. Bolte, A. Daniilidis, A. Lewis, The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems, SIAM J. Optim. 17 (2007), 1205-1223.
[23] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, 2012.

[24] H.K. Xu, Iterative methods for the split feasibility problem in infinite dimensional Hilbert spaces, Inverse Probl. 26 (2010), 105018.
[25] L.C. Ceng, Q.H. Ansari, J.C. Yao, Relaxed extragradient methods for finding minimum-norm solutions of the split feasibility problem, Nonlinear Anal. 75 (2012), 2116-2125.
[26] Y. Censor, A. Gibali, S. Reich, Algorithms for the split variational inequality problem, Numer. Algorithms 59 (2012), 301-323.
[27] P. Ochs, Local convergence of the heavy-ball method and iPiano for non-convex optimization, J. Optim. Theory Appl. 177 (2018), 153-180.
[28] P. Ochs, Long term motion analysis for object level grouping and nonsmooth optimization methods, PhD thesis, Albert-Ludwigs-Universität Freiburg, 2015.