A modified parallel monotone hybrid algorithm for a finite family of $G$-nonexpansive mappings and application to a novel signal recovery

Kunrada Kankam$^a$, Prasit Cholamjiak$^a$, Watcharaporn Cholamjiak$^a$

$^a$School of Science, University of Phayao, Phayao 56000, Thailand.

Abstract

In this work, we aim to prove the convergence of the sequences generated by the shrinking projection method and the parallel monotone method to find a common fixed point of a finite family of $G$-nonexpansive mappings endowed with graphs. We obtain strong convergence results under some mild conditions. We provide numerical examples and give applications to signal recovery. Moreover, numerical experiments of our algorithms which different blurred matrices on the algorithm to show the efficiency and the implementation for signal recovery.

Keywords: Shrinking projection method $G$–nonexpasive mapping Common fixed point Hilbert space, Signal recovery.

2010 MSC: 47H10, 65K05, 94A12.

1. Introduction

Many application in applied science such as the signal recovery, image restoration [18, 19, 20, 31, 32, 33, 34] can be explained by the linear equation system in one dimensional vector as follows:

$$v = F u + \epsilon,$$

(1)

where $F : \mathbb{R}^N \to \mathbb{R}^M (M < N)$ is the blurred matrix, $u \in \mathbb{R}^N$ is an original signal with $k$ nonzero components to be recovered, $\epsilon$ is additive noise and $v \in \mathbb{R}^M$ is the observed signal. It is known that solving the problem (1) can be seen as the well-known regularized least square problem which is called LASSO problem:

$$\min_{u \in \mathbb{R}^N} \left( \frac{1}{2} ||y - Fu||_2^2 + \lambda ||u||_1 \right),$$

(2)
where $\lambda > 0$. Moreover, the problem (2) can be generalized by convex optimization problems as the following form:
\[
\minimize(f(u) + g(u)),
\]
where $u \in \mathcal{H}$, $f, g : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ are two proper, lower-semicontinuous and convex functions, and $\mathcal{H}$ is a real Hilbert space. It is known that the problem (3) is equivalent to the fixed point problem as follows:
\[
u = \text{prox}_{\alpha g}(u - \alpha \nabla f(u)),
\]
where $\alpha > 0$, $\text{prox}_g = (I + \partial g)^{-1}$ is the proximal operator of $g$ and $\partial g$ is subdifferential of $g$ and $\nabla f$ denotes the gradient of $f$. From this point of view, it is known that (4) can generate a classical forward-backward algorithm in the following manner:
\[
u_{n+1} = \text{prox}_{\alpha_n g}(u_n - \alpha_n \nabla f(u_n)),
\]
where $\alpha_n$ is a suitable stepsize. Many mathematicians have used this algorithm (5) to modify in many ways such as the proximal point algorithm [13, 21, 25, 28] and the gradient method [11, 30, 41, 42]. For its applications, there have been modifications of the algorithm (5) in many various areas of science and physics etc., (see [7, 8, 10, 16, 17, 23, 26, 44, 56]).

In signal processing, the sound may be disturbed by many noises. The goal in this paper is to remove noise without knowing the type of noise and different blurred matrices. Here, we aim to focus on the following problem
\[
\min_{u \in \mathbb{R}^N} \left\{ \frac{1}{2} \|F_1 u - v_1\|^2_2 + \lambda_1 \|u\|_1 \right\}, \quad \min_{u \in \mathbb{R}^N} \left\{ \frac{1}{2} \|F_2 u - v_2\|^2_2 + \lambda_2 \|u\|_1 \right\}, \quad \ldots, \quad \min_{u \in \mathbb{R}^N} \left\{ \frac{1}{2} \|F_N u - v_N\|^2_2 + \lambda_N \|u\|_1 \right\},
\]
where $F_i$ is a bounded linear operator, $u$ is original signal and $v_k$ is observed signal with noisy for all $k = 1, 2, ..., N$.

Before we start solving the problem (6), we recall the concept of the fixed point problem of $G$–nonexpansive mapping. Let $\mathcal{K}$ be a nonempty subset of a real Hilbert space $\mathcal{H}$. Let $\Delta$ be the diagonal of the cartesian product $\mathcal{K} \times \mathcal{K}$, i.e., $\Delta = \{(u, u) : u \in \mathcal{K}\}$ and $\mathcal{G}$ be a directed graph such that the set $V(\mathcal{G})$ is vertices coincides of graph $\mathcal{G}$ with $\mathcal{K}$ and the set $E(\mathcal{G})$ is edges of graph $\mathcal{G}$ with $\Delta \subseteq E(\mathcal{G})$. We assume $\mathcal{G}$ has no parallel edge, then the graph $\mathcal{G}$ is the pair $(V(\mathcal{G}), E(\mathcal{G}))$. A mapping $S : \mathcal{K} \to \mathcal{K}$ is said to be

1. **Contraction** if $S$ satisfies the following way: there exists $\alpha \in (0, 1)$ such that
   \[
   \|Su - Sv\| \leq \alpha \|u - v\|,
   \]
   for all $u, v \in \mathcal{K}$.
2. **Nonexpansive** if $S$ satisfies the conditions:
   \[
   \|Su - Sv\| \leq \|u - v\|,
   \]
   for all $u, v \in \mathcal{K}$.
3. **$G$–contraction** if $S$ satisfies the conditions:
   (1) $S$ preserves edges of $\mathcal{G}$, i.e.,
   \[
   (u, v) \in E(\mathcal{G}) \Rightarrow (Su, Sv) \in E(\mathcal{G}),
   \]
   for all $(u, v) \in E(\mathcal{G})$.
   (2) $S$ decreases weights of edges of $\mathcal{G}$ in the following way: there exists $\alpha \in (0, 1)$ such that
   \[
   (u, v) \in E(\mathcal{G}) \Rightarrow \|Su - Sv\| \leq \alpha \|u - v\|,
   \]
   for all $(u, v) \in E(\mathcal{G})$. 
4. **G - nonexpansive** if \( S \) satisfies the conditions:
   
   (1) \( S \) preserves edges of \( G \), i.e.,
   
   \[(u, v) \in E(G) \Rightarrow (Su, Sv) \in E(G),\]
   
   for all \((u, v) \in E(G)\).

   (2) \( S \) non-decreases weights of edges of \( G \) in the following way:
   
   \[(u, v) \in E(G) \Rightarrow \|Su - Sv\| \leq \|u - v\|,\]
   
   for all \((u, v) \in E(G)\).

   The set of all fixed points of \( S \) is denoted by \( F(S) \), i.e., \( F(S) = \{z \in K : Sz = z\} \). If we set \( Su = \text{prox}_{\alpha f}(u - \alpha \nabla f(u)) \) where \( \alpha \in (0, 2/L) \) and \( L \) is the Lipschitz constant of the gradient of functions \( f \), then \( S \) is nonexpansive. It is known that if \( S \) is nonexpansive, then \( S \) is \( G \)-nonexpansive. This is the reason that why we interested in studying \( G \)-nonexpansive mapping.

   In 2008, the Banach’s contraction principle was studied and extended to complete metric spaces endowed with a graph by Jachymski [15]. In 2012, Aleomraninejad et al. [1] introduced some iterative scheme for \( G \)-contraction with \( G \)-nonexpansive mappings in Banach spaces endowed with a graph. Recently, Alfuradan [3] studied the existence of fixed point and proved the convergence result of monotone nonexpansive mapping on a Banach space endowed with a directed graph. Since 2012, the Browders convergence theorem for \( G \)-nonexpansive mapping in a Hilbert space with a directed graph, weak and strong convergence of some iterations for \( G \)-nonexpansive mappings were discussed by many authors (see for example [1, 2, 3, 39, 40]).

   Finding a common solution to a problem system is very useful in real-world problems. Many authors [9, 12, 27, 48] have proposed many algorithms to solve it. One of that is a parallel monotone hybrid algorithm was proposed for solving common fixed point problems of a finite family of quasi \( \phi \)-nonexpansive mappings \( (S_i)_{i=1}^N \) in Banach spaces by Anh and Hieu [4, 5]. It was modified by using parallel methods and the Shrinking projection method [24]. It can be seen in Hilbert spaces as follows:

   \[
   \begin{cases}
   u_0 \in K, \\
   v_n^k = \alpha_n u_n + (1 - \alpha_n) S_k u_n, k = 1, 2, ..., N, \\
   k_n = \arg\max ||v_n^k - u_n|| : k = 1, 2, ..., N, \\
   \bar{v}_n := v_n^{k_n}, \\
   C_{n+1} = \{ \bar{v}_n : ||\bar{v}_n - \bar{v}_n|| \leq ||\bar{v}_n - u_n||\}, \\
   u_{n+1} = P_{C_{n+1}} u_n, n \geq 1,
   \end{cases}
   \tag{7}
   \]

   where \( 0 < \alpha_n < 1, \limsup_{n \to \infty} \alpha_n < 1 \). It can be seen that at the \( n \)-th iteration step of the algorithm (7), \( \bar{v}_n \) is chosen from the parallel of \( v_n^k \) for \( k = 1, 2, ..., N \). For the final step, the projection on to \( C_{n+1} \) which can be more easily performed then existing algorithms specially when the number of variational inequalities \( N \) is large. The parallel algorithm can solve the problem which is divided into sub-problem and are executed in parallel to get individual outputs which are combined together to get the final desired output, so it have been used to solve many problems (see [9, 14, 37]).

   Motivated by the previous works, we proposed a new algorithm with the modified parallel monotone algorithm for finding a common fixed point. Using the Shrinking projection method, we obtain the strong convergence theorem under suitable conditions in Hilbert spaces endowed with a directed graph. Further, we give an example and numerical experiments for supporting our main results. Finally, we use our proposed algorithm for applying to solve the signal recovery problem [6].

2. **Main results**

   In this section, we prove strong convergence theorems of the modified parallel hybrid algorithm for a finite family of \( G \)-nonexpansive mappings in real Hilbert spaces. Throughout this paper, we denote \( u_n \to u \) and \( u_n \rightharpoonup u \) as strong and weak convergence of \( \{u_n\} \) to \( u \), respectively.
Thus, we have \( p \) and convex subset of \( \mathcal{K} \) (Step 2).

On the other hand, as \( \Omega \) is nonexpansive mapping for all \( k = 1, 2, \ldots, N \) such that \( \Omega := \bigcap_{k=1}^{N} F(S_k) \neq \emptyset \), \( \Omega \) is closed and \( F(S_k) \times F(S_k) \subseteq E(\mathcal{G}) \) for all \( k = 1, 2, \ldots, N \). Take \( C_0 = C_1 \) and \( u_1 \in \mathcal{K} \) arbitrarily and

**Step 1:** Compute

\[
v_n = \alpha_n^0 u_n + \sum_{k=1}^{N} \alpha_n^k S_k u_n.
\]

**Step 2:** Calculate

\[C_{n+1} = \{ t \in C_n : \|t - v_n\| \leq \|t - u_n\| \}\]
and

\[u_{n+1} = P_{C_{n+1}} u_0, n \geq 1\]

where \( \{ \alpha_n^k \} \) is a sequence in \([0, 1]\) for all \( k = 0, 1, \ldots, N \) and \( \sum_{k=0}^{N} \alpha_n^k = 1 \).

**Theorem 2.2.** Let \( \{ u_n \} \) be generated by Algorithm 2.1. Assume that the following conditions hold:

1. \( \{ u_n \} \) dominates \( p \) for all \( p \in F \) and if there exists a subsequence \( \{ u_{n_i} \} \) of \( \{ u_n \} \) such that \( u_{n_i} \rightharpoonup w \in \mathcal{K} \), then \( (u_{n_i}, w) \in E(\mathcal{G}) \);

2. \( \lim_{n \to \infty} \alpha_n^0 \alpha_n^k > 0 \) for all \( k = 1, 2, \ldots, N \).

Then \( \{ u_n \} \) convergence strongly to \( w = P_{\Omega} u_1 \).

**Proof.** We split the proof into five steps.

**Step 1.** Show that \( P_{C_{n+1}} \) is well-defined for every \( u_1 \in \mathcal{H} \). As shown in Theorem 3.2 of Tiammee et al., \( F(S_k) \) is convex for all \( k = 1, 2, \ldots, N \). It follows from our assumption that \( \Omega \) is closed and convex. Hence, \( P_{\Omega} u_1 \) is well-defined. We see that \( C_1 = \mathcal{K} \) is closed and convex. Assume that \( C_n \) is closed and convex. From the definition of \( C_{n+1} \) and Lemma 1.3 in [22], we get \( C_{n+1} \) is closed and convex. Let \( p \in \Omega \). Since \( \{ u_n \} \) dominates \( p \) and \( S_k \) is edge-preserving for all \( k = 1, 2, \ldots, N \), we have \( (S_k u_n, p) \in E(\mathcal{G}) \) for all \( k = 1, 2, \ldots, N \). This shows that \( (v_n, p) = (\alpha_n^0 u_n + \sum_{k=1}^{N} \alpha_n^k S_k u_n, p) \in E(\mathcal{G}) \) by \( E(\mathcal{G}) \) is convex, we get

\[
\|v_n - p\| = \|\alpha_n^0 u_n + \sum_{k=1}^{N} \alpha_n^k S_k u_n - p\|
\leq \alpha_n^0 \|u_n - p\| + \sum_{k=1}^{N} \alpha_n^k \|S_k u_n - p\|
\leq \|u_n - p\|.
\]

Thus, we have \( p \in C_{n+1} \). Therefore \( \Omega \cap C_{n+1} \). This implies that \( P_{C_{n+1}} u_1 \) is well-defined.

**Step 2.** Show that \( \lim_{n \to \infty} \|u_n - u_1\| \) exists. By the property of the metric projection \( P_{\Omega} \) when \( \Omega \) is a nonempty, closed and convex subset of \( \mathcal{H} \), then there exists a unique \( v \in \Omega \) such that \( v = P_{\Omega} u_1 \). From \( u_{n+1} \in C_n \), for all \( n \geq 1 \), we get

\[
\|P_{C_n} u_1 - u_1\| \leq \|u_{n+1} - u_1\|, \quad \forall n \geq 1.
\]

(8)

On the other hand, as \( \Omega \subset C_n \), we obtain

\[
\|u_n - u_1\| \leq \|v - u_1\|, \quad \forall n \geq 1.
\]

(9)
It follows from (8) and (9) that the sequence \( \{u_n\} \) is bounded and nondecreasing. Therefore \( \lim_{n \to \infty} \|u_n - u_1\| \) exists.

**Step 3.** Show that \( u_n \to w \in \mathcal{K} \) as \( n \to \infty \). For \( j > n \), by the definition of \( C_n \), since \( u_j = P_{C_j} u_1 \in C_j \subset C_n \), it is from the metric projection that
\[
\|u_j - u_n\|^2 \leq \|u_j - u_1\|^2 - \|u_n - u_1\|^2.
\]
Since \( \lim_{n \to \infty} \|u_n - u_1\| \) exists, by Step 2, we have \( u_j \to u_n \) as \( n \to \infty \) i.e., \( \{u_n\} \) is a Cauchy sequence. Hence, there exists \( w \in \mathcal{K} \) such that \( u_n \to w \) as \( n \to \infty \). In particular, we have
\[
\lim_{n \to \infty} \|u_{n+1} - u_n\| = 0. \tag{10}
\]

**Step 4.** Show that \( w \in \Omega \). Since \( u_{n+1} \in C_{n+1} \subset C_n \), it follows from (10) that
\[
\|v_n - u_n\| \leq \|v_n - u_{n+1}\| + \|u_{n+1} - u_n\| \leq 2\|u_{n+1} - u_n\| \to 0 \tag{11}
\]
as \( n \to \infty \). For \( p \in \Omega \), it follows from Lemma 2.5 in [6] and \( \{u_n\} \) dominates \( p \) that
\[
\|v_n - p\|^2 = \|a_n^0 u_n + \sum_{k=1}^{N} a_n^k S_k u_n - p\|^2 \\
\leq a_n^0 \|u_n - p\|^2 + \sum_{k=1}^{N} a_n^k \|S_k u_n - p\|^2 - \sum_{k=1}^{N} a_n^0 a_n^k \|S_k u_n - u_n\|^2 \\
\leq \|u_n - p\|^2 - \sum_{k=1}^{N} a_n^0 a_n^k \|S_k u_n - u_n\|^2.
\]
This implies that
\[
\sum_{k=1}^{N} a_n^0 a_n^k \|S_k u_n - u_n\|^2 \leq \|u_n - p\|^2 - \|v_n - p\|^2.
\]
It follows from the assumption (2) and (11), we obtain
\[
\lim_{n \to \infty} \|S_k u_n - u_n\| = 0
\]
for all \( k = 1, 2, ..., N \). From \( u_n \to w \) as \( n \to \infty \), the assumption (1) and Lemma 6 in [35], we have \( w \in \Omega \).

**Step 5.** Show that \( w = P_{\Omega} u_1 \). By the property of the metric projection \( P_{C_n} u_1 \), we have
\[
\langle u_1 - P_{C_n} u_1, P_{C_n} u_1 - p \rangle \geq 0, \quad \forall p \in C_n. \tag{12}
\]
By taking the limit in (12), we obtain
\[
\langle u_1 - w, w - p \rangle \geq 0, \quad \forall p \in C_n.
\]
Since \( \Omega \subset C_n \), so \( w = P_{\Omega} u_1 \). This completes the proof. \( \square \)

We know that if \( S \) is nonexpansive, that \( S \) is \( \mathcal{G} \)-nonexpansive. Applying from Theorem 2.1, we obtain the following corollary.

**Corollary 2.3.** Let \( \mathcal{K} \) be a nonempty closed and convex subset of a real Hilbert space \( \mathcal{H} \). Let \( S_k : \mathcal{K} \to \mathcal{K} \) be a nonexpansive mapping for all \( k = 1, 2, ..., N \) such that \( \Omega := \bigcap_{k=1}^{N} F(S_k) \neq \emptyset \). Let \( \{u_n\} \) be a sequence generated by
\[
\begin{align*}
\left\{ \begin{array}{l}
u_1 \in \mathcal{K}, \\
v_n = a_n^0 u_n + \sum_{k=1}^{N} a_n^k S_k u_n,
\end{array} \right. \\
C_{n+1} = \{ t \in C_n : \| t - v_n \| \leq \| t - u_n \| \}, \\
u_{n+1} = P_{C_{n+1}} u_0, n \geq 1,
\end{align*}
\]
(13)
where \( \{ \alpha_k^n \} \) is a sequence in \([0, 1]\) for all \( k = 0, 1, ..., N \) and \( \sum_{k=0}^{N} \alpha_k^n = 1 \) with \( \liminf_{n \to \infty} \alpha_0^n \alpha_k^n > 0 \) for all \( k = 1, 2, ..., N \).

Then \( \{ u_n \} \) convergence strongly to \( w = P_\Omega u_1 \).

### 3. Numerical Experiments

In this section, we provide the numerical experiments and apply the convex minimization problem to signal restoration problem. All experiments and visualizations are performed on a computer (Intel(R) Core(TM) i7-2600 16 GB RAM/Windows 10/64-bit) with MATLAB 2022a.

In this experiments, firstly, we give an example in Euclidian space \( \mathbb{R}^3 \) which shows numerical experiment for supporting our main theorem.

**Example 3.1.** Let \( \mathcal{H} = \mathbb{R}^3 \) and \( \mathcal{K} = [0, \infty) \times (-\infty, 5] \times [-5, 5] \). Assume that \((u, v) \in E(G)\) if and only if \( 1 \leq u_1, v_1, u_2, v_2 \leq 0, -1.5 \leq u_3, v_3 \leq -0.5 \) for all \( u = (u_1, u_2, u_3), v = (v_1, v_2, v_3) \in \mathcal{K} \). Define mappings \( S_1, S_2, S_3 : \mathcal{K} \to \mathcal{K} \) by

\[
S_1 u = \left( \log \frac{u_1}{3} + 3, -1, -1 \right);
S_2 u = \left( 3, -1, \frac{\tan(u_3 + 1)}{3} - 1 \right);
S_3 u = \left( 3, e^{u_2+1} - \frac{3}{2}, -1 \right)
\]

for all \( u = (u_1, u_2, u_3) \in \mathcal{K} \). It is easy to check that \( S_1, S_2 \) and \( S_3 \) are \( G \)-nonexpansive such that \( F(S_1) \cap F(S_2) \cap F(S_3) = \{(3, -1, -1)\} \). On the other hand, \( S_1 \) is nonexpansive since for \( u = (0.23, -5, 5) \) and \( v = (0.15, -5, 5) \). This implies that \( \|S_1 u - S_1 v\| > 0.1 > \|u - v\| \). \( S_2 \) is not nonexpansive since for \( u = (10, -7, 0.1) \) and \( v = (10, -7, 0.29) \). We have \( \|S_2 u - S_2 v\| > 0.3 > \|u - v\| \). Moreover, \( S_3 \) is not nonexpansive since for \( u = (0.5, 0.48, 1) \) and \( v = (0.5, 0.49, 1) \). We have \( \|S_3 u - S_3 v\| > 0.5 > \|u - v\| \). We use the mean squared error (MSE) to measure quantitatively, which is defined by

\[
\text{MSE} = \frac{1}{N} \|u^k - u_*\|^2 < \varepsilon,
\]

where \( u^k \) is an estimated point of \( u_* \). In our experiment, we give cases as follows in Table 1. The numerical results are reported by Table 2.
Table 1: Choose parameters $\alpha$ for $S_i$, $i = 1, 2, 3$ and stop condition(Cauchy error) < $10^{-9}$.

<table>
<thead>
<tr>
<th>Initial point</th>
<th>Cases</th>
<th>Inputting</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0.01,-1.29,-7.38)$</td>
<td>C1</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C6</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C7</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>$(1,-0.9,-0.1)$</td>
<td>C8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C9</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C10</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C11</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
<tr>
<td></td>
<td>C12</td>
<td>$a_0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_3$</td>
</tr>
</tbody>
</table>
Table 2: The convergence behavior of Table 1.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Inputting operators</th>
<th>S₁₅</th>
<th>S₂₅</th>
<th>S₃₅</th>
<th>S₁S₂₅</th>
<th>S₁S₃₅</th>
<th>S₂S₃₅</th>
<th>S₁S₂S₃₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Time</td>
<td>0.0585</td>
<td>0.0699</td>
<td>0.0074</td>
<td>0.0073</td>
<td>0.0069</td>
<td>0.0076</td>
<td>0.0074</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>55</td>
<td>53</td>
<td>34</td>
<td>46</td>
<td>41</td>
<td>58</td>
<td>48</td>
</tr>
<tr>
<td>C2</td>
<td>Time</td>
<td>0.0491</td>
<td>0.0088</td>
<td>0.0073</td>
<td>0.0068</td>
<td>0.0073</td>
<td>0.0070</td>
<td>0.0074</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>53</td>
<td>53</td>
<td>39</td>
<td>50</td>
<td>43</td>
<td>46</td>
<td>55</td>
</tr>
<tr>
<td>C3</td>
<td>Time</td>
<td>0.0571</td>
<td>0.0067</td>
<td>0.0072</td>
<td>0.0082</td>
<td>0.0073</td>
<td>0.0082</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>53</td>
<td>56</td>
<td>34</td>
<td>48</td>
<td>47</td>
<td>53</td>
<td>39</td>
</tr>
<tr>
<td>C4</td>
<td>Time</td>
<td>0.0454</td>
<td>0.0068</td>
<td>0.0073</td>
<td>0.0069</td>
<td>0.0086</td>
<td>0.0070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>59</td>
<td>53</td>
<td>39</td>
<td>46</td>
<td>45</td>
<td>47</td>
<td>40</td>
</tr>
<tr>
<td>C5</td>
<td>Time</td>
<td>0.0499</td>
<td>0.0071</td>
<td>0.0071</td>
<td>0.0074</td>
<td>0.0080</td>
<td>0.0070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>53</td>
<td>54</td>
<td>34</td>
<td>42</td>
<td>54</td>
<td>63</td>
<td>45</td>
</tr>
<tr>
<td>C6</td>
<td>Time</td>
<td>0.0499</td>
<td>0.0074</td>
<td>0.0072</td>
<td>0.0069</td>
<td>0.0074</td>
<td>0.0067</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>52</td>
<td>55</td>
<td>34</td>
<td>41</td>
<td>48</td>
<td>44</td>
<td>42</td>
</tr>
<tr>
<td>C7</td>
<td>Time</td>
<td>0.0513</td>
<td>0.0068</td>
<td>0.0103</td>
<td>0.0106</td>
<td>0.0084</td>
<td>0.0073</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>71</td>
<td>63</td>
<td>50</td>
<td>54</td>
<td>55</td>
<td>59</td>
<td>47</td>
</tr>
<tr>
<td>C8</td>
<td>Time</td>
<td>0.0503</td>
<td>0.0071</td>
<td>0.0071</td>
<td>0.0068</td>
<td>0.0074</td>
<td>0.0081</td>
<td>0.0070</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>66</td>
<td>61</td>
<td>47</td>
<td>50</td>
<td>54</td>
<td>50</td>
<td>46</td>
</tr>
<tr>
<td>C9</td>
<td>Time</td>
<td>0.0545</td>
<td>0.0079</td>
<td>0.0086</td>
<td>0.0073</td>
<td>0.0076</td>
<td>0.0078</td>
<td>0.0086</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>81</td>
<td>73</td>
<td>54</td>
<td>66</td>
<td>70</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>C10</td>
<td>Time</td>
<td>0.0549</td>
<td>0.0077</td>
<td>0.0073</td>
<td>0.0074</td>
<td>0.0073</td>
<td>0.0075</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>83</td>
<td>75</td>
<td>57</td>
<td>66</td>
<td>65</td>
<td>60</td>
<td>58</td>
</tr>
<tr>
<td>C11</td>
<td>Time</td>
<td>0.0548</td>
<td>0.0069</td>
<td>0.0071</td>
<td>0.0075</td>
<td>0.0073</td>
<td>0.0077</td>
<td>0.0069</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>84</td>
<td>74</td>
<td>54</td>
<td>66</td>
<td>75</td>
<td>77</td>
<td>66</td>
</tr>
<tr>
<td>C12</td>
<td>Time</td>
<td>0.0497</td>
<td>0.0069</td>
<td>0.0100</td>
<td>0.0079</td>
<td>0.0079</td>
<td>0.0075</td>
<td>0.0073</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>84</td>
<td>75</td>
<td>55</td>
<td>69</td>
<td>95</td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>C13</td>
<td>Time</td>
<td>0.0489</td>
<td>0.0071</td>
<td>0.0078</td>
<td>0.0077</td>
<td>0.0077</td>
<td>0.0073</td>
<td>0.0077</td>
</tr>
<tr>
<td></td>
<td>*Iter</td>
<td>161</td>
<td>146</td>
<td>114</td>
<td>128</td>
<td>148</td>
<td>126</td>
<td>129</td>
</tr>
</tbody>
</table>

The results are presented in Table 2, where the CPU time and number of iterations for all cases under the three operators S₁, S₂ and S₃ by using the main algorithm. It is shown that in the CPU time and number of iterations of proposed algorithm decrease when α set by case C7 and C8, it has an effect on the number of iterations for input many mapping Sᵢ.

Next, we apply the Algorithm 2.1 to solve the LASSO problem in signal recovery (6) by setting

\[ S_{ij}u = \text{prox}_{\lambda_{ij}}(u_n - A_i \nabla f_i(u_n)) \]

In our experiment, the sparse vector \( u \in \mathbb{R}^N \) is generated from uniform distribution in the interval [-2,2] with \( n \) nonzero elements. The matrix \( A_i \in \mathbb{R}^{M \times N} \) is generated from a normal distribution with mean zero and invariance one. The observation \( v_i \) is generated by with Gaussian noise white signal-to-noise ratio (SNR).

In what follows, let the initial point is picked randomly. Let the step size \( \alpha_n^{k} = 1/4 \) in Algorithm 2.1. The numerical results are shown in Table 3.
Table 3: The computational result for solving the LASSO problem (6).

<table>
<thead>
<tr>
<th>Cases</th>
<th>Size</th>
<th>Inputting</th>
<th>( m = 10 )</th>
<th>( m = 20 )</th>
<th>( m = 40 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( A_i )</td>
<td>Time</td>
<td>Iter</td>
<td>Time</td>
</tr>
<tr>
<td>1-A1</td>
<td>( M = 512 )</td>
<td>( A_1 )</td>
<td>15.7565</td>
<td>7573</td>
<td>14.5070</td>
</tr>
<tr>
<td>1-A2</td>
<td>( N = 256 )</td>
<td>( A_2 )</td>
<td>13.4425</td>
<td>7141</td>
<td>15.1371</td>
</tr>
<tr>
<td>1-A3</td>
<td>( A_3 )</td>
<td>16.0106</td>
<td>7398</td>
<td>12.7354</td>
<td>6917</td>
</tr>
<tr>
<td>1-A12</td>
<td>( A_1 A_2 )</td>
<td>0.6685</td>
<td>1521</td>
<td>0.6461</td>
<td>1489</td>
</tr>
<tr>
<td>1-A13</td>
<td>( A_1 A_3 )</td>
<td>0.8720</td>
<td>1672</td>
<td>0.6313</td>
<td>1411</td>
</tr>
<tr>
<td>1-A23</td>
<td>( A_2 A_3 )</td>
<td>0.8203</td>
<td>1594</td>
<td>0.6999</td>
<td>1440</td>
</tr>
<tr>
<td>1-A123</td>
<td>( A_1 A_2 A_3 )</td>
<td>0.1480</td>
<td>481</td>
<td>0.1172</td>
<td>441</td>
</tr>
<tr>
<td>2-A1</td>
<td>( M = 1024 )</td>
<td>( A_1 )</td>
<td>87.6031</td>
<td>12791</td>
<td>85.5404</td>
</tr>
<tr>
<td>2-A2</td>
<td>( N = 512 )</td>
<td>( A_2 )</td>
<td>99.7328</td>
<td>13691</td>
<td>109.2934</td>
</tr>
<tr>
<td>2-A3</td>
<td>( A_3 )</td>
<td>6.0985</td>
<td>2634</td>
<td>8.8468</td>
<td>2819</td>
</tr>
<tr>
<td>2-A12</td>
<td>( A_1 A_2 )</td>
<td>8.9298</td>
<td>2587</td>
<td>6.7377</td>
<td>2654</td>
</tr>
<tr>
<td>2-A13</td>
<td>( A_1 A_3 )</td>
<td>9.8777</td>
<td>2383</td>
<td>8.3323</td>
<td>2923</td>
</tr>
<tr>
<td>2-A23</td>
<td>( A_2 A_3 )</td>
<td>1.8444</td>
<td>617</td>
<td>1.2883</td>
<td>698</td>
</tr>
<tr>
<td>3-A1</td>
<td>( M = 512 )</td>
<td>( A_1 )</td>
<td>13.4005</td>
<td>7094</td>
<td>16.1148</td>
</tr>
<tr>
<td>3-A2</td>
<td>( N = 256 )</td>
<td>( A_2 )</td>
<td>13.8487</td>
<td>7258</td>
<td>16.6190</td>
</tr>
<tr>
<td>3-A3</td>
<td>( A_3 )</td>
<td>11.6239</td>
<td>6647</td>
<td>15.8769</td>
<td>7606</td>
</tr>
<tr>
<td>3-A12</td>
<td>( A_1 A_2 )</td>
<td>0.5997</td>
<td>1390</td>
<td>0.6758</td>
<td>1489</td>
</tr>
<tr>
<td>3-A13</td>
<td>( A_1 A_3 )</td>
<td>0.7106</td>
<td>1569</td>
<td>0.7120</td>
<td>1460</td>
</tr>
<tr>
<td>3-A23</td>
<td>( A_2 A_3 )</td>
<td>0.8484</td>
<td>1600</td>
<td>1.2158</td>
<td>1816</td>
</tr>
<tr>
<td>3-A123</td>
<td>( A_1 A_2 A_3 )</td>
<td>0.1400</td>
<td>461</td>
<td>0.1888</td>
<td>539</td>
</tr>
</tbody>
</table>

The Table 3 we set Case1-2 by input \( A_i, i = 1, 2, 3, \) SNR=40 and in Case 3 input \( A_1, \) SNR=40, \( A_2, \) SNR=60 and \( A_3, \) SNR=70. It is shown that the recovered signal by inputting \( A_i, i = 1, 2, 3 \) has less number of iterations and CPU time than inputting \( A_i, i = 1, 2 \) and \( A_i, i = 1 \) for all cases. Next, we give some numerical experiments for two cases in Table 3 to illustrate the convergence behavior of cases in comparison. We plot the number of iterations versus MSE< \( 10^{-5} \) are shown in Figure 11 and the original signal, observation data and recovered signal are shown in Figure 1, Figure 2, Figure 3-5 respectively.

Figure 1: The original signal size \( N = 512, M = 256 \) and 20 spikes.
The different types of three blurred matrices are shown in Figure 2.

Figure 2: The measured values with $A_i, i = 1, 2, 3$, SNR=40.

Figure 3: The recovered signal by Table 3 with $m = 20$ in case 2-A1 (12693 Iter, CPU=85.5404), case 2-A2 (13979 Iter, CPU=102.9699), case 2-A3 (14155 Iter, CPU=109.2934), respectively.
Next, we provide the signal recovery by different type of inputting SNR. The original signal, observation data and recovered signal are shown in Figure 6, Figure 7, Figure 8-10, respectively.

Figure 4: The recovered signal by Table 3 with $m = 20$ in case 2-A12 (2819 Iter, CPU=6.8648), case 2-A13 (2654 Iter, CPU=6.7377), case 2-A23 (2923 Iter, CPU=8.3323), respectively.

Figure 5: The recovered signal by Table 3 with $m = 20$ in case 2-A123 (698 Iter, CPU=1.2883).

Figure 6: The original signal size $N = 1024$, $M = 512$ and 40 spikes.
The different types of three blurred matrices are shown in Figure 7.

Figure 7: The measured values with input $A_1$, SNR=40, $A_2$, SNR=60, $A_3$, SNR=70, respectively.

Figure 8: The recovered signal by Table [3] with $m = 40$ in case 3-A1 (11103 Iter, CPU=32.4653), case 3-A2 (11543 Iter, CPU=34.5990), case 3-A3 (12264 Iter, CPU=40.2163), respectively.
Figure 9: The recovered signal by Table 3 with $m = 40$ in case 3-A12 (1671 Iter, CPU=0.7678), case 3-A13 (1385 Iter, CPU=0.5935), case 3-A23 (1428 Iter, CPU=0.9180), respectively.

Figure 10: The recovered signal by Table 3 with $m = 40$ in case 3-A123 (474 Iter, CPU=0.5131).

Figure 11: The MSE of Algorithm 2.1 with Table 3 with case 2, $m = 20$ and case 3, $m = 40$, respectively.

From Table 3 and Figure 11, we see that the CPU time and the numbers of iterations of $A_i$, $i = 1, 2, 3$ of Algorithm 2.1 are better than inputting $A_i$, $i = 1, 2$ and $A_i$, $i = 1$ of all cases for solving the LASSO problem in signal recovery.
Next, we provide a comparison among Algorithms 2.1, MSP algorithm [29] and NMTS algorithm [43]. Let the stepsize $\alpha_i = 1/4$ in Algorithms 2.1, and let $\alpha_n = \beta_n = \gamma_n = 0.1$ in SP algorithm [29] and NMTS algorithm [43]. The numerical results are shown in Table 4.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Size</th>
<th>Inputting</th>
<th>$m = 10$</th>
<th>$m = 20$</th>
<th>$m = 40$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>time</td>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td>1</td>
<td>$M = 512$</td>
<td>Algorithm 2.1</td>
<td>0.1480</td>
<td>0.1172</td>
<td>0.1437</td>
</tr>
<tr>
<td></td>
<td>$N = 256$</td>
<td>MSP</td>
<td>0.2044</td>
<td>0.1738</td>
<td>0.2451</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMTS</td>
<td>0.1979</td>
<td>0.1598</td>
<td>0.2357</td>
</tr>
<tr>
<td>2</td>
<td>$M = 1024$</td>
<td>Algorithm 2.1</td>
<td>1.844</td>
<td>1.2832</td>
<td>1.2076</td>
</tr>
<tr>
<td></td>
<td>$N = 512$</td>
<td>MSP</td>
<td>1.5547</td>
<td>1.9789</td>
<td>1.6240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMTS</td>
<td>2.2776</td>
<td>2.431</td>
<td>2.3807</td>
</tr>
<tr>
<td>3</td>
<td>$M = 512$</td>
<td>Algorithm 2.1</td>
<td>14.8005</td>
<td>10.1481</td>
<td>14.331</td>
</tr>
<tr>
<td></td>
<td>$N = 256$</td>
<td>MSP</td>
<td>1.5547</td>
<td>1.9789</td>
<td>1.6240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMTS</td>
<td>2.2776</td>
<td>2.431</td>
<td>2.3807</td>
</tr>
<tr>
<td>4</td>
<td>$M = 1024$</td>
<td>Algorithm 2.1</td>
<td>1.844</td>
<td>1.2832</td>
<td>1.2076</td>
</tr>
<tr>
<td></td>
<td>$N = 512$</td>
<td>MSP</td>
<td>1.5744</td>
<td>1.6933</td>
<td>1.5577</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMTS</td>
<td>1.8619</td>
<td>2.5307</td>
<td>2.5962</td>
</tr>
</tbody>
</table>

Next, we give two cases in Table 3 to show the efficiency of Algorithm 2.1 in case A123 with the other algorithms by MSE versus the number of iterations.

The original signal are shown in Figure 12-15, the observation signal are shown in Figure 13-16 and the recovered signal are shown in Figure 14-17, respectively.

Figure 12: The original signal size $N = 512$, $M = 256$ and 10 spikes.
Figure 13: The measured values with $A_i, i = 1, 2, 3$, SNR=40.

Figure 14: The recovered signal by Table 4 case 1 with $m = 10$ in Algorithm 2.1 (481 Iter, CPU=0.1480), MSP algorithm (669 Iter, CPU=0.2044), NMTS algorithm (1233 Iter, CPU=0.1979), respectively.
Figure 15: The original signal size $N = 1024$, $M = 512$ and 40 spikes.

Figure 16: The measured values with input $A_1$, SNR=40, $A_2$, SNR=60, $A_3$, SNR=70, respectively.
Figure 17: The recovered signal by Table 4 case 4 with $m = 40$ in Algorithm 2.1 (624 Iter, CPU = 1.1695), MSP algorithm (860 Iter, CPU = 1.5577), NMTS algorithm (1233 Iter, CPU = 2.5862), respectively.

Figure 18: The MSE of Algorithm 2.1 by Table 4 with case 1, $m = 10$ and case 4, $m = 40$, respectively.

From Table 4 and Figure 18, we see that the CPU time and the numbers of iterations of Algorithm 2.1 are better than those of MSP algorithm and NMTS algorithm.

Acknowledgement

K. Kankam was supported by School of Science, University of Phayao under grant no. PBSC65023. W. Cholamjiak was supported by the National Research Council of Thailand (NRCT), Thailand Science Research and Innovation Fund, and University of Phayao (Grant No. FF65-RIM072; FF65-UoE002).

References

B. Martinet, Brèv `ev communication. Régularisation d’inéquations variationnelles par approximations successives. Revue
K. Kankam, N. Pholasa, P. Cholamjiak, Hybrid forward-backward algorithms using linesearch rule for minimization prob-
[20]
K. Kankam, P. Cholamjiak, Strong convergence of the forward–backward splitting algorithms via linesearches in Hilbert
[18]
[17]
S.A. Khan, W. Cholamjiak, K.R. Kazmi, An inertial forward-backward splitting method for solving combination of equi-
[16]
[15]
J. Jachymski, The contraction principle for mappings on a metric space with a graph. Proceedings of the American Mathe-
[14]
[13]
O. Guler, On the convergence of the proximal point algorithm for convex minimization. SIAM Journal on Control and
[12]
[11]
[10]
Q. Dong, D. Jiang, P. Cholamjiak, Y. Shehu, A strong convergence result involving an inertial forward-backward algorithm
[9]
[8]
[7]
C. Martinez-Yanes, H.K. Xu, Strong convergence of the CQ method for fixed point iteration processes. Nonlinear Analysis:
[6]
[5]
[4]
[3]
[2]
K. Sitthithakerngkiet, J. Deepho, P. Kumam, Modified hybrid steepest method for the split feasibility problem in image
[28]
[27]
[25]
[24]
[23]
[22]
[21]
[20]
[19]
[18]
[17]
[16]
[15]
[14]
[13]
[12]
[11]
[10]
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]

C.E. Chidume, J.N. Ezeora, Krasnoselskii-type algorithm for family of multi-valued strictly pseudo-contractive mappings.

P. Cholamjiak, A generalized forward-backward splitting method for solving quasi inclusion problems in Banach spaces.

W. Cholamjiak, P. Cholamjiak, S. Suantai, An inertial forward-backward splitting method for solving inclusion problems in

P. Cholamjiak, S. Suantai, P. Sunthrayuth, An explicit parallel algorithm for solving variational inclusion problem and fixed

Q. Dong, D. Jiang, P. Cholamjiak, Y. Shehu, A strong convergence result involving an inertial forward-backward algorithm

J.C. Dunn, Convexity, monotonicity, and gradient processes in Hilbert space. Journal of Mathematical Analysis and Applica-

A.G. Gebrie, R. Wangkeeree, Parallel proximal method of solving split system of fixed point set constraint minimization

O. Guler, On the convergence of the proximal point algorithm for convex minimization. SIAM Journal on Control and
Optimization, 29(2) (1991) 403-419.

D.V. Hieu, Parallel and cyclic hybrid subgradient extragradient methods for variational inequalities. Afrika Matematika,

J. Jachymski, The contraction principle for mappings on a metric space with a graph. Proceedings of the American Mathemat-

S.A. Khan, S. Suantai, W. Cholamjiak, Shrinking projection methods involving inertial forward-backward splitting methods
for inclusion problems. Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas, 113(2)

S.A. Khan, W. Cholamjiak, K.R. Kazmi, An inertial forward-backward splitting method for solving combination of equi-

K. Kankam, N. Pholasa, P. Cholamjiak, On convergence and complexity of the modified forward-backward method involv-

K. Kankam, P. Cholamjiak, Strong convergence of the forward–backward splitting algorithms via linesearches in Hilbert

K. Kankam, N. Pholasa, P. Cholamjiak, Hybrid forward-backward algorithms using linesearch rule for minimization prob-

B. Martinet, Brèv `ev communication. Régularisation d’inéquations variationnelles par approximations successives. Revue
fran`aise d’informatique et de recherche opérationnelle. Série rouge, 4(R3) (1907) 154-158.

C. Martinez-Yanes, H.K. Xu, Strong convergence of the CQ method for fixed point iteration processes. Nonlinear Analysis:

T. Minh Tuyen, R. Promkam, P. Sunthrayuth, Strong convergence of a generalized forward-backward splitting method in

K. Nakajo, W. Takahashi, Strong convergence theorems for nonexpansive mappings and nonexpansive semigroups. Journal


N. Pholasa, P. Cholamjiaka, Y. J. Cho, Modified forward-backward splitting methods for accretive operators in Banach

S. Reich, T.M. Tuyen, N. M. Trang, Parallel iterative methods for solving the split common fixed point problem in Hilbert

R.T. Rockafellar, Monotone operators and the proximal point algorithm. SIAM Journal on Control and Optimization, 14(5)
(1976) 877-898.

P. Sridarat, R. Suparaturatorn, S. Suantai, Y. J. Cho, Convergence analysis of SP-iteration for G-nonexpansive mappings

K. Sitthithakerngkit, J. Deepho, P. Kumam, Modified hybrid steepest method for the split feasibility problem in image

S. Suantai, P. Jailoka, A. Hanjing, An accelerated viscosity forward-backward splitting algorithm with the linesearch process

S. Suantai, K. Kankam, P. Cholamjiak, A novel forward-backward algorithm for solving convex minimization problem in

S. Suantai, K. Kankam, P. Cholamjiak, A projected forward-backward algorithm for constrained minimization with appli-
cations to image inpainting. mathematics, 9(8) (2021) 890.

S. Suantai, M.A. Noor, K. Kankam, P. Cholamjiak, Novel forward–backward algorithms for optimization and applications

S. Suantai, M. Donganont, W. Cholamjiak, Hybrid methods for a countable family of G-nonexpansive mappings in Hilbert

S. Suantai, P. Cholamjiak, P. Sunthrayuth, Iterative methods with perturbations for the sum of two accretive operators
in q-uniformly smooth Banach spaces. Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A.