Set-Valued and Variational Analysis (2022) 30:1485–1502
https://doi.org/10.1007/s11228-022-00649-w

Forward-partial inverse-half-forward splitting algorithm for solving monotone inclusions

Luis Briceño-Arias[1] · **Jinjian Chen**[2] · **Fernando Roldán**[1] · **Yuchao Tang**[2]

Received: 1 February 2022 / Accepted: 6 July 2022 / Published online: 31 August 2022

Abstract
In this paper we provide a splitting algorithm for solving coupled monotone inclusions in a real Hilbert space involving the sum of a normal cone to a vector subspace, a maximally monotone, a monotone-Lipschitzian, and a cocoercive operator. The proposed method takes advantage of the intrinsic properties of each operator and generalizes the method of partial inverses and the forward-backward-half forward splitting, among other methods. At each iteration, our algorithm needs two computations of the Lipschitzian operator while the cocoercive operator is activated only once. By using product space techniques, we derive a method for solving a composite monotone primal-dual inclusions including linear operators and we apply it to solve constrained composite convex optimization problems. Finally, we apply our algorithm to a constrained total variation least-squares problem and we compare its performance with efficient methods in the literature.

Keywords Splitting algorithms · Monotone operator theory · Partial inverse · Convex optimization

Mathematics Subject Classification (2010) 47H05 · 47J25 · 49M29 · 65K05 · 90C25

1 Introduction

In this paper we study the numerical resolution of the following inclusion problem. The normal cone to V is denoted by N_V.

All the authors contributed equally to this work.

✉ Luis Briceño-Arias
luis.briceno@usm.cl

Jinjian Chen
400603319001@email.ncu.edu.cn

Fernando Roldán
fernando.roldan@usm.cl

Yuchao Tang
hhaaoo1331@163.com

[1] Department of Mathematics, Universidad Técnica Federico Santa María, Santiago, Chile

[2] Department of Mathematics, Nanchang University, Nanchang, People's Republic of China

Problem 1.1 Let $\mathcal{H}$ be a real Hilbert space and let V be a closed vector subspace of $\mathcal{H}$. Let $A : \mathcal{H} \to 2^{\mathcal{H}}$ be a maximally monotone operator, let $B : \mathcal{H} \to \mathcal{H}$ be a monotone and L-Lipschitzian operator for some $L \in]0, +\infty[$, and let $C : \mathcal{H} \to \mathcal{H}$ be a β-cocoercive operator for some $\beta \in]0, +\infty[$. The problem is to

$$\text{find} \quad x \in \mathcal{H} \quad \text{such that} \quad 0 \in Ax + Bx + Cx + N_V x, \tag{1.1}$$

under the assumption that its solutions set Z is nonempty.

Problem 1.1 models a wide class of problems in engineering including mechanical problems [33, 35, 36], differential inclusions [2, 46], game theory [1, 13], restoration and denoising in image processing [18, 19, 26], traffic theory [9, 32, 34], among others.

In the case when $V = \mathcal{H}$ and the resolvent of B is available, Problem 1.1 can be solved by the algorithms in [27, 28] and, if B is linear, by the algorithm in [39]. Moreover, if the resolvent of B is difficult to compute, Problem 1.1 can be solved by the *forward-backward-half forward* algorithm (FBHF) proposed in [14]. FBHF implement explicit activations of B and C and generalizes the classical forward-backward splitting [40] and Tseng's splitting [50] when $B = 0$ and $C = 0$, respectively.

In the case when $V \neq \mathcal{H}$, a splitting algorithm for solving the case $B = C = 0$ is proposed in [47] using the partial inverse of A with respect to V and extensions for the cases $B = 0$ and $C = 0$ are proposed in [10] and [11], respectively. On the other hand, the algorithms proposed in [4–8, 12, 17, 21–23, 25, 27, 29, 30, 37, 38, 41, 43–45, 51] can solve Problem 1.1 under additional assumptions or without exploiting the vector subspace structure and the intrinsic properties of the operators involved. Indeed, the algorithms in [6–8, 12, 21, 30] need to compute the resolvents of B and C, which are not explicit in general or they can be numerically expensive. In addition, previous methods do not take advantage of the vector subspace structure of Problem 1.1. The schemes proposed in [4, 22, 29, 37] take advantage of the properties of B, but the cocoercivity of C and the vector subspace structure are not leveraged. In fact, the algorithms in [4, 22, 29, 37] may consider $B + C$ as a monotone and Lipschitzian operator and activate it twice by iteration. In contrast, the algorithms in [17, 25, 41, 44, 45] activates $B + C$ only once by iteration, but they need to store in the memory the two past iterations and the step-size is reduced significantly. In addition, the methods proposed in [5, 23, 27, 38, 43, 51] take advantage of the cocoercivity of C, but they do not exploit neither the properties of B nor the vector subspace structure of the problem.

Furthermore, note that Problem 1.1 can be solved by the algorithms proposed in [14, 16] by considering N_V as any maximally monotone operator via product space techniques. These approaches do not exploit the vector subspace structure of the problem and need to update additional auxiliary dual variables at each iteration, which affects their efficiency in large scale problems. Moreover, since $B + C$ is monotone and $(\beta^{-1} + L)$-Lipschitzian, Problem 1.1 can be solved by [11]. However, this implementation needs two computations of C by iteration which affects its efficiency when C is computationally expensive and also may increment drastically the number of iterations to achieve the convergence criterion, as perceived in [14, Section 7.1] in the case $V = \mathcal{H}$.

In this paper we propose a splitting algorithm which fully exploits the vector subspace structure, the cocoercivity of C, and the Lipschitzian property of B. In the particular case when $V = \mathcal{H}$, we recover [14], which generalizes the forward-backward splitting and Tseng's splitting [50]. For general vector subspaces, our algorithm also recovers the methods proposed in [10, 11, 47]. By using standard product space techniques, we apply our algorithm to solve composite primal-dual monotone inclusions including a normal cone to a vector subspace, cocoercive, and Lipschitzian-monotone operators and composite convex

optimization problems under vector subspace constraints. We implement our method in the context of TV-regularized least-squares problems with constraints and we compare its performance with previous methods in the literature including [24]. We observe that, in the case when the matrix in the data fidelity term has large norm values, our implementation is more efficient.

The paper is organized as follows. In Section 2 we set our notation. In Section 3 we provide our main algorithm for solving Problem 1.1 and its proof of convergence. In Section 4 we derive a method for solving a composite monotone primal-dual inclusion, including monotone, Lipschitzian, cocoercive, and bounded linear operators. In this section we also derive an algorithm for solve constrained composite convex optimization problems. Finally, in Section 5 we provide numerical experiments illustrating the efficiency of our proposed method.

2 Notations and Preliminaries

Throughout this paper $\mathcal{H}$ and $\mathcal{G}$ are real Hilbert spaces. We denote their scalar products by $\langle \cdot \mid \cdot \rangle$, the associated norms by $\|\cdot\|$, and by $\rightharpoonup$ the weak convergence. Given a linear bounded operator $L : \mathcal{H} \to \mathcal{G}$, we denote its adjoint by $L^* : \mathcal{G} \to \mathcal{H}$. Id denotes the identity operator on $\mathcal{H}$. Let $D \subset \mathcal{H}$ be non-empty and let $T : D \to \mathcal{H}$. Let $\beta \in]0, +\infty[$. The operator T is β–cocoercive if

$$(\forall x \in D)(\forall y \in D) \quad \langle x - y \mid Tx - Ty \rangle \geq \beta \|Tx - Ty\|^2 \tag{2.1}$$

and it is L–Lipschitzian if

$$(\forall x \in D)(\forall y \in D) \quad \|Tx - Ty\| \leq L\|x - y\|. \tag{2.2}$$

Let $A : \mathcal{H} \to 2^{\mathcal{H}}$ be a set-valued operator. The domain of A is $\operatorname{dom} A = \big\{x \in \mathcal{H} \bigm| Ax \neq \varnothing\big\}$, the range of A is $\operatorname{ran} A = \big\{u \in \mathcal{H} \bigm| (\exists x \in \mathcal{H})\, u \in Ax\big\}$, and the graph of A is $\operatorname{gra} A = \big\{(x, u) \in \mathcal{H} \times \mathcal{H} \bigm| u \in Ax\big\}$. The set of zeros of A is $\operatorname{zer} A = \big\{x \in \mathcal{H} \bigm| 0 \in Ax\big\}$, the inverse of A is $A^{-1} : \mathcal{H} \to 2^{\mathcal{H}} : u \mapsto \big\{x \in \mathcal{H} \bigm| u \in Ax\big\}$, and the resolvent of A is $J_A = (\mathrm{Id} + A)^{-1}$. The operator A is monotone if

$$(\forall (x, u) \in \operatorname{gra} A)(\forall (y, v) \in \operatorname{gra} A) \quad \langle x - y \mid u - v \rangle \geq 0 \tag{2.3}$$

and it is maximally monotone if it is monotone and there exists no monotone operator $B : \mathcal{H} \to 2^{\mathcal{H}}$ such that $\operatorname{gra} B$ properly contains $\operatorname{gra} A$, i.e., for every $(x, u) \in \mathcal{H} \times \mathcal{H}$,

$$(x, u) \in \operatorname{gra} A \quad \Leftrightarrow \quad (\forall (y, v) \in \operatorname{gra} A) \quad \langle x - y \mid u - v \rangle \geq 0. \tag{2.4}$$

We denote by $\Gamma_0(\mathcal{H})$ the class of proper lower semicontinuous convex functions $f : \mathcal{H} \to]-\infty, +\infty]$. Let $f \in \Gamma_0(\mathcal{H})$. The Fenchel conjugate of f is defined by $f^* : u \mapsto \sup_{x \in \mathcal{H}}(\langle x \mid u \rangle - f(x))$, which is a function in $\Gamma_0(\mathcal{H})$, the subdifferential of f is the maximally monotone operator

$$\partial f : x \mapsto \big\{u \in \mathcal{H} \bigm| (\forall y \in \mathcal{H})\, f(x) + \langle y - x \mid u \rangle \leq f(y)\big\},$$

we have that $(\partial f)^{-1} = \partial f^*$, and that $\operatorname{zer} \partial f$ is the set of minimizers of f, which is denoted by $\arg\min_{x \in \mathcal{H}} f$. We denote by

$$\operatorname{prox}_f : x \mapsto \operatorname*{arg\,min}_{y\in\mathcal{H}} \Big(f(y) + \frac{1}{2}\|x-y\|^2 \Big). \tag{2.5}$$

We have $\operatorname{prox}_f = J_{\partial f}$. Moreover, it follows from [3, Theorem 14.3] that

$$(\forall \gamma > 0) \quad \operatorname{prox}_{\gamma f} + \gamma \operatorname{prox}_{f^*/\gamma} \circ (\mathrm{Id}/\gamma) = \mathrm{Id}. \tag{2.6}$$

Given a non-empty closed convex set $C \subset \mathcal{H}$, we denote by P_C the projection onto C, by $\iota_C \in \Gamma_0(\mathcal{H})$ the indicator function of C, which takes the value 0 in C and $+\infty$ otherwise, and by $N_C = \partial(\iota_C)$ the normal cone to C. The partial inverse of A with respect to a closed vector subspace V of $\mathcal{H}$, denoted by A_V, is defined by

$$(\forall (x,y) \in \mathcal{H}^2) \quad y \in A_V x \quad \Leftrightarrow \quad (P_V y + P_{V^\perp} x) \in A(P_V x + P_{V^\perp} y). \tag{2.7}$$

Note that $A_{\mathcal{H}} = A$ and $A_{\{0\}} = A^{-1}$. For further properties of monotone operators, non-expansive mappings, and convex analysis, the reader is referred to [3].

The following is a simplified version of the algorithm proposed in [14, Theorem 2.3].

Proposition 2.1 [14, *Theorem* 2.3] *Let* $\widehat{L} \in]0, +\infty[$, *let* $\hat{\beta} \in]0, +\infty[$, *let* $\mathcal{A} : \mathcal{H} \to 2^{\mathcal{H}}$ *be a maximally monotone operator, let* $\mathcal{B} : \mathcal{H} \to \mathcal{H}$ *be monotone and* $\hat{L}$*-Lipschitzian, and let* $\mathcal{C} : \mathcal{H} \to \mathcal{H}$ *be a* $\hat{\beta}$*-cocoercive operator. Suppose that* $\operatorname{zer}(\mathcal{A}+\mathcal{B}+\mathcal{C}) \neq \varnothing$ *and set*

$$\hat{\chi} = \frac{4\hat{\beta}}{1+\sqrt{1+16\hat{\beta}^2\hat{L}^2}} \in]0, \min\Big\{2\hat{\beta}, \frac{1}{\hat{L}}\Big\}[, \tag{2.8}$$

let $(\lambda_n)_{n\in\mathbb{N}}$ *be a sequence in* $[\varepsilon, \hat{\chi}-\varepsilon]$, *for some* $\varepsilon \in]0, \hat{\chi}/2[$. *Moreover, let* $z_0 \in \mathcal{H}$ *and consider the following recurrence*

$$\left\lfloor \begin{aligned} &\text{for } n = 0, 1, 2, \ldots \\ &s_n = J_{\lambda_n \mathcal{A}}\big(z_n - \lambda_n(\mathcal{B}+\mathcal{C})z_n\big) \\ &z_{n+1} = s_n + \lambda_n(\mathcal{B}z_n - \mathcal{B}s_n). \end{aligned} \right. \tag{2.9}$$

Then, $(z_n)_{n\in\mathbb{N}}$ *converges weakly to some* $\overline{z} \in \operatorname{zer}(\mathcal{A}+\mathcal{B}+\mathcal{C})$.

Observe that (2.9) reduces to forward-backward splitting when $\mathcal{B} = 0$ (and $L = 0$), and to a version of Tseng's splitting when $\mathcal{C} = 0$ (and $\beta \to +\infty$) [12, 50].

3 Main Result

The following is our main algorithm, whose convergence is proved in Theorem 3.2 below.

Algorithm 3.1 In the context of Problem 1.1, let $(x_0, y_0) \in V \times V^\perp$, let $\gamma \in]0, +\infty[$, and let $(\lambda_n)_{n\in\mathbb{N}}$ be a sequence in $]0, +\infty[$. Consider the recurrence

$$\left\lfloor\begin{array}{l}\text{for } n=0,1,2,\ldots\\ \text{find } (p_n,q_n)\in\mathcal{H}^2 \text{ such that } x_n+\gamma y_n-\lambda_n\gamma P_V(B+C)x_n=p_n+\gamma q_n\\ \qquad\text{and } \dfrac{P_Vq_n}{\lambda_n}+P_{V^\perp}q_n\in A\bigg(P_Vp_n+\dfrac{P_{V^\perp}p_n}{\lambda_n}\bigg),\\ x_{n+1}=P_Vp_n+\lambda_n\gamma P_V(Bx_n-BP_Vp_n),\\ y_{n+1}=P_{V^\perp}q_n.\end{array}\right. \tag{3.1}$$

Note that (3.1) involves only one activation of C, two of B, and three projections onto V at each iteration.

Theorem 3.2 *In the context of Problem* 1.1, *set*

$$\chi=\frac{4\beta}{1+\sqrt{1+16\beta^2L^2}}\in\left]0,\min\left\{2\beta,\frac{1}{L}\right\}\right[, \tag{3.2}$$

let $\gamma\in]0,+\infty[$, *and let* $(\lambda_n)_{n\in\mathbb{N}}$ *be a sequence in* $[\varepsilon,\chi/\gamma-\varepsilon]$ *for some* $\varepsilon\in]0,\chi/(2\gamma)[$. *Moreover, let* $(x_0,y_0)\in V\times V^\perp$ *and let* $(x_n)_{n\in\mathbb{N}}$ *and* $(y_n)_{n\in\mathbb{N}}$ *be the sequences generated by Algorithm* 3.1. *Then* $(x_n)_{n\in\mathbb{N}}$ *and* $(y_n)_{n\in\mathbb{N}}$ *are sequences in* V *and* $V^\perp$, *respectively, and there exist* $\bar{x}\in Z$ *and* $\bar{y}\in V^\perp\cap(A\bar{x}+P_V(B+C)\bar{x})$ *such that* $x_n\rightharpoonup\bar{x}$ *and* $y_n\rightharpoonup\bar{y}$.

Proof Define

$$\begin{cases}\mathcal{A}_\gamma=(\gamma A)_V:\mathcal{H}\to 2^{\mathcal{H}}\\ \mathcal{B}_\gamma=\gamma P_V\circ B\circ P_V:\mathcal{H}\to\mathcal{H}\\ \mathcal{C}_\gamma=\gamma P_V\circ C\circ P_V:\mathcal{H}\to\mathcal{H}.\end{cases} \tag{3.3}$$

It follows from [11, Proposition 3.1(i)&(ii)] that $\mathcal{A}_\gamma$ is maximally monotone and that $\mathcal{B}_\gamma$ is monotone and γL-Lipschitzian. Moreover, $\mathcal{C}_\gamma$ is β/γ-cocoercive in view of [10, Proposition 5.1(ii)]. Since C is β^{-1}-Lipschitzian, $B+C$ is $(\beta^{-1}+L)$-Lipschitzian, and (3.3) and the linearity of P_V yield

$$\mathcal{B}_\gamma+\mathcal{C}_\gamma=\gamma P_V\circ(B+C)\circ P_V. \tag{3.4}$$

Therefore, [11, Proposition 3.1(iii)] implies that $\hat{x}\in\mathcal{H}$ is a solution to Problem 1.1 if and only if

$$\begin{aligned}\hat{x}\in V\quad\text{and}\quad&\big(\exists\hat{y}\in V^\perp\cap(A\hat{x}+B\hat{x}+C\hat{x})\big)\\ &\hat{x}+\gamma\big(\hat{y}-P_{V^\perp}(B+C)\hat{x}\big)\in\operatorname{zer}(\mathcal{A}_\gamma+\mathcal{B}_\gamma+\mathcal{C}_\gamma).\end{aligned} \tag{3.5}$$

Now, since $x_0\in V$ and $y_0\in V^\perp$, it follows from Algorithm 3.1 that $(x_n)_{n\in\mathbb{N}}$ and $(y_n)_{n\in\mathbb{N}}$ are sequences in V and $V^\perp$, respectively. In addition, from Algorithm 3.1 and [11, Proposition 3.1(i)] we deduce that

$$(\forall n\in\mathbb{N})\quad J_{\lambda_n\mathcal{A}_\gamma}\big(x_n+\gamma y_n-\lambda_n\gamma P_V(B+C)x_n\big)=P_Vp_n+\gamma P_{V^\perp}q_n. \tag{3.6}$$

For every $n\in\mathbb{N}$, set $z_n=x_n+\gamma y_n$ and set $s_n=P_Vp_n+\gamma P_{V^\perp}q_n$. Hence, for every $n\in\mathbb{N}$, $P_Vs_n=P_Vp_n$, $P_{V^\perp}s_n=\gamma P_{V^\perp}q_n$, and (3.6) and (3.4) yield

$$\begin{aligned} s_n &= J_{\lambda_n \mathcal{A}_\gamma}(x_n + \gamma y_n - \lambda_n \gamma P_V(B+C)x_n) \\ &= J_{\lambda_n \mathcal{A}_\gamma}(z_n - \lambda_n \gamma P_V(B+C)P_V z_n) \\ &= J_{\lambda_n \mathcal{A}_\gamma}\big(z_n - \lambda_n(\mathcal{B}_\gamma + \mathcal{C}_\gamma)z_n\big). \end{aligned} \tag{3.7}$$

Thus, from Algorithm 3.1 we deduce that, for every $n \in \mathbb{N}$,

$$\begin{aligned} z_{n+1} &= x_{n+1} + \gamma y_{n+1} \\ &= P_V p_n + \lambda_n \gamma P_V(Bx_n - BP_V p_n) + \gamma P_{V^\perp} q_n \\ &= P_V s_n + \lambda_n(\gamma P_V B P_V z_n - \gamma P_V B P_V s_n) + P_{V^\perp} s_n \\ &= s_n + \lambda_n(\mathcal{B}_\gamma z_n - \mathcal{B}_\gamma s_n). \end{aligned} \tag{3.8}$$

Therefore, we obtain from (3.7) and (3.8) that

$$\left\lfloor \begin{aligned} &\text{for } n = 0, 1, 2, \ldots \\ &s_n = J_{\lambda_n \mathcal{A}_\gamma}\big(z_n - \lambda_n(\mathcal{B}_\gamma + \mathcal{C}_\gamma)z_n\big) \\ &z_{n+1} = s_n + \lambda_n(\mathcal{B}_\gamma z_n - \mathcal{B}_\gamma s_n). \end{aligned} \right. \tag{3.9}$$

Altogether, by setting $\hat{\beta} = \beta/\gamma$ and $\hat{L} = \gamma L$, we have $\hat{\chi} = \chi/\gamma$ and Proposition 2.1 asserts that there exists $\overline{z} \in \operatorname{zer}(\mathcal{A}_\gamma + \mathcal{B}_\gamma + \mathcal{C}_\gamma)$ such that $z_n \rightharpoonup \overline{z}$. Furthermore, by setting $\overline{x} = P_V \overline{z}$ and $\overline{y} = P_{V^\perp}\overline{z}/\gamma$, we have $-(\mathcal{B}_\gamma + \mathcal{C}_\gamma)(\overline{x} + \gamma\overline{y}) \in \mathcal{A}_\gamma(\overline{x} + \gamma\overline{y})$, which, in view of (3.3), is equivalent to $-P_V(B+C)\overline{x} + \overline{y} \in A\overline{x}$. Therefore, by defining $\hat{y} = \overline{y} + P_{V^\perp}(B+C)\overline{x} \in V^\perp \cap (A\overline{x} + B\overline{x} + C\overline{x})$, we have $\overline{x} + \gamma(\hat{y} - P_{V^\perp}(B+C)\overline{x}) \in \operatorname{zer}(\mathcal{A}_\gamma + \mathcal{B}_\gamma + \mathcal{C}_\gamma)$ and (3.5) implies that $\overline{x} \in Z$ and that $\overline{y} \in V^\perp \cap (A\overline{x} + P_V(B+C)\overline{x})$. Moreover, from the weakly continuity of P_V and $P_{V^\perp}$, we obtain $x_n = P_V z_n \rightharpoonup P_V \overline{z} = \overline{x}$ and $y_n = P_{V^\perp} z_n/\gamma \rightharpoonup P_{V^\perp}\overline{z}/\gamma = \overline{y}$, which completes the proof.

The sequence $(\lambda_n)_{n\in\mathbb{N}}$ in Algorithm 3.1 can be manipulated in order to accelerate the convergence. However, as in [10, 11, 48], the inclusion in (3.1) is not always easy to solve. The following result provides a particular case of our method, in which this inclusion can be explicitly computed in terms of the resolvent of A.

Corollary 3.3 *In the context of Problem* 1.1, *let* $(x_0, y_0) \in V \times V^\perp$, *let* $\chi \in]0, +\infty[$ *be the constant defined in* (3.2), *let* $\gamma \in]0, \chi[$, *and let* $(x_n)_{n\in\mathbb{N}}$ *and* $(y_n)_{n\in\mathbb{N}}$ *be the sequences generated by the recurrence*

$$\left\lfloor \begin{aligned} &\text{for } n = 0, 1, 2, \ldots \\ &p_n = J_{\gamma A}\big(x_n + \gamma y_n - \gamma P_V(B+C)x_n\big) \\ &r_n = P_V p_n \\ &x_{n+1} = r_n + \gamma P_V(Bx_n - Br_n) \\ &y_{n+1} = y_n - \frac{p_n - r_n}{\gamma}. \end{aligned} \right. \tag{3.10}$$

Then, there exist $\overline{x} \in Z$ *and* $\overline{y} \in V^\perp \cap (A\overline{x} + P_V(B+C)\overline{x})$ *such that* $x_n \rightharpoonup \overline{x}$ *and* $y_n \rightharpoonup \overline{y}$.

Proof Note that (3.10) implies that $(x_n)_{n\in\mathbb{N}}$ and $(y_n)_{n\in\mathbb{N}}$ are sequences in V and $V^\perp$, respectively. Fix $n \in \mathbb{N}$ and set $q_n = (x_n + \gamma y_n - \gamma P_V(B+C)x_n - p_n)/\gamma$. Hence, we obtain from (3.10) that $p_n + \gamma q_n = x_n + \gamma y_n - \gamma P_V(B+C)x_n$,

that $q_n \in Ap_n$, that $x_{n+1} = P_V p_n + \gamma P_V(Bx_n - BP_V p_n)$, and that $y_{n+1} = y_n - (p_n - P_V p_n)/\gamma = y_n - P_{V^\perp} p_n/\gamma = P_{V^\perp} q_n$. Therefore, (3.10) is a particular case of Algorithm 3.1 when $\lambda_n \equiv 1 \in]0, \chi/\gamma[$ and the result hence follows from Theorem 3.2.

Remark 3.4

1. Note that, in the case when $C = 0$, (3.10) reduces to the method proposed in [11]. Observe that in this case we can take $\beta \to +\infty$ which yields $\chi \to 1/L$.
2. Note that, in the case when $B = 0$, (3.10) reduces to the method proposed in [10]. In this case, we can take $L \to 0$, which yields $\chi \to 2\beta$.
3. In the case when $V = \mathcal{H}$, (3.10) reduces to the algorithm proposed in [14] (see also Proposition 2.1).

4 Applications

In this section we tackle the following composite primal-dual monotone inclusion.

Problem 4.1 Let H be a real Hilbert space, let V be a closed vector subspace of H, let $\mathsf{A} : \mathsf{H} \to 2^{\mathsf{H}}$ be maximally monotone, let $\mathsf{M} : \mathsf{H} \to \mathsf{H}$ be monotone and μ-Lipschitzian, for some $\mu \in]0, +\infty[$, let $\mathsf{C} : \mathsf{H} \to \mathsf{H}$ be ζ-cocoercive, for some $\zeta \in]0, +\infty[$, and let m be a strictly positive integer. For every $i \in \{1, \ldots, m\}$, let G_i be a real Hilbert space, let $\mathsf{B}_i : \mathsf{G}_i \to 2^{\mathsf{G}_i}$ be maximally monotone, let $\mathsf{N}_i : \mathsf{G}_i \to 2^{\mathsf{G}_i}$ be monotone and such that N_i^{-1} is ν_i-Lipschitzian, for some $\nu_i \in]0, +\infty[$, let $\mathsf{D}_i : \mathsf{G}_i \to 2^{\mathsf{G}_i}$ be maximally monotone and δ_i -strongly monotone, for some $\delta_i \in]0, +\infty[$, and let $\mathsf{L}_i : \mathsf{H} \to \mathsf{G}_i$ be a nonzero bounded linear operator. The problem is to

$$\begin{aligned} &\text{find} \quad \overline{\mathsf{x}} \in \mathsf{H}, \overline{\mathsf{u}}_1 \in \mathsf{G}_1, \ldots, \overline{\mathsf{u}}_m \in \mathsf{G}_m \quad \text{such that} \\ &\qquad \begin{cases} 0 \in \mathsf{A}\overline{\mathsf{x}} + \mathsf{M}\overline{\mathsf{x}} + \mathsf{C}\overline{\mathsf{x}} + \sum_{i=1}^m \mathsf{L}_i^* \overline{\mathsf{u}}_i + N_{\mathsf{V}}\overline{\mathsf{x}} \\ 0 \in \big(\mathsf{B}_1^{-1} + \mathsf{N}_1^{-1} + \mathsf{D}_1^{-1}\big)\overline{\mathsf{u}}_1 - \mathsf{L}_1\overline{\mathsf{x}} \\ \quad \vdots \\ 0 \in \big(\mathsf{B}_m^{-1} + \mathsf{N}_m^{-1} + \mathsf{D}_m^{-1}\big)\overline{\mathsf{u}}_m - \mathsf{L}_m\overline{\mathsf{x}}, \end{cases} \end{aligned} \tag{4.1}$$

under the assumption that the solution set $\boldsymbol{Z}$ to (4.1) is nonempty.

Note that, if $(\overline{\mathsf{x}}, \overline{\mathsf{u}}_1, \ldots, \overline{\mathsf{u}}_m) \in \boldsymbol{Z}$ then $\overline{\mathsf{x}}$ solves the primal inclusion

$$\text{find} \quad \overline{\mathsf{x}} \in \mathsf{H} \quad \text{such that} \quad 0 \in \mathsf{A}\overline{\mathsf{x}} + \mathsf{M}\overline{\mathsf{x}} + \mathsf{C}\overline{\mathsf{x}} + \sum_{i=1}^m \mathsf{L}_i^*\Big(\big(\mathsf{B}_i \square \mathsf{N}_i \square \mathsf{D}_i\big)\mathsf{L}_i\overline{\mathsf{x}}\Big) + N_{\mathsf{V}}\overline{\mathsf{x}} \tag{4.2}$$

and $(\overline{\mathsf{u}}_1, \ldots, \overline{\mathsf{u}}_m)$ solves the dual inclusion

$$\begin{aligned} &\text{find } \overline{\mathsf{u}}_1 \in \mathsf{G}_1, \ldots, \overline{\mathsf{u}}_m \in \mathsf{G}_m \text{ such that} \\ &\qquad (\exists \mathsf{x} \in \mathsf{H}) \quad \begin{cases} -\sum_{i=1}^m \mathsf{L}_i^* \overline{\mathsf{u}}_i \in \mathsf{A}\mathsf{x} + \mathsf{M}\mathsf{x} + \mathsf{C}\mathsf{x} + N_{\mathsf{V}}\mathsf{x} \\ (\forall i \in \{1, \ldots, m\}) \quad \overline{\mathsf{u}}_i \in \big(\mathsf{B}_i \square \mathsf{N}_i \square \mathsf{D}_i\big)\mathsf{L}_i\mathsf{x}. \end{cases} \end{aligned} \tag{4.3}$$

In the case when $\mathsf{V} = \mathsf{H}$, $\mathsf{C} = 0$, and, for every $i \in \{1, \ldots, m\}$, $\mathsf{D}_i^{-1} = 0$, this problem can be solved by algorithms in [20, 22] by using Tseng's splitting [50] in a suitable product space.

In the case when $\mathsf{V} = \mathsf{H}$, $\mathsf{M} = 0$, and, for every $i \in \{1, \ldots, m\}$, $\mathsf{N}_i^{-1} = 0$, this problem can be solved by algorithms in [23, 51] by using forward-backward splitting in a suitable product space. Since $\mathsf{M} + \mathsf{C}$ and $(\mathsf{N}_i^{-1} + \mathsf{D}_i^{-1})_{1 \leq i \leq m}$ are monotone and Lipschitzian and N_{V} is maximally monotone, Problem 4.1 can be solved by the algorithms in [20, 22]. However, these methods do not exploit the cocoercivity or the vector subspace structure of Problem 4.1. Other algorithms as those in [16, 21, 38] provide alternatives for solving Problem 4.1, but any of them exploit its vector subspace and cocoercive structure. In the case when $\mathsf{M} = 0$, and, for every $i \in \{1, \ldots, m\}$, $\mathsf{N}_i^{-1} = 0$, the algorithm in [15] exploits the vector subspace structure of Problem 4.1 by using the partial inverse of A with respect to V. The following result provides a fully split algorithm to solve Problem 4.1 in its full generality. It is obtained by using (3.10) in a suitable product space, which exploits the vector subspace structure and which activates each cocoercive operator only once by iteration.

Proposition 4.2 *Consider the framework of Problem* 4.1 *and set*

$$L = \max\{\mu, \nu_1, \ldots, \nu_m\} + \sqrt{\sum_{i=1}^{m} \|\mathsf{L}_i\|^2} \quad \text{and} \quad \beta = \min\{\zeta, \delta_1, \ldots, \delta_m\}. \tag{4.4}$$

Let $\mathsf{x}_0 \in \mathsf{V}$, *let* $\mathsf{y}_0 \in \mathsf{V}^{\top}$, *for every* $i \in \{1, \ldots, m\}$, *let* $\mathsf{u}_{i,0} \in \mathsf{G}_i$, *set* $\gamma \in]0, \chi[$, *where* χ *is defined in* (3.2), *and consider the routine*

$$\left\lfloor \begin{array}{l} \text{for } n = 0, 1, 2, \ldots \\ \mathsf{p}_n = J_{\gamma \mathsf{A}}\Big(\mathsf{x}_n + \gamma \mathsf{y}_n - \gamma P_{\mathsf{V}}\Big((\mathsf{M} + \mathsf{C})\mathsf{x}_n + \sum_{i=1}^{m} \mathsf{L}_i^* \mathsf{u}_{i,n}\Big)\Big) \\ \mathsf{q}_n = P_{\mathsf{V}} \mathsf{p}_n \\ \left\lfloor \begin{array}{l} \text{for } i = 1, \ldots, m \\ \mathsf{r}_{i,n} = J_{\gamma \mathsf{B}_i^{-1}}\Big(\mathsf{u}_{i,n} - \gamma\big((\mathsf{N}_i^{-1} + \mathsf{D}_i^{-1})\mathsf{u}_{i,n} - \mathsf{L}_i \mathsf{x}_n\big)\Big) \\ \mathsf{u}_{i,n+1} = \mathsf{r}_{i,n} - \gamma\big(\mathsf{N}_i^{-1}\mathsf{r}_{i,n} - \mathsf{N}_i^{-1}\mathsf{u}_{i,n} - \mathsf{L}_i(\mathsf{q}_n - \mathsf{x}_n)\big) \end{array} \right. \\ \mathsf{x}_{n+1} = \mathsf{q}_n - \gamma P_{\mathsf{V}}\Big(\mathsf{M}\mathsf{q}_n - \mathsf{M}\mathsf{x}_n + \sum_{i=1}^{m} \mathsf{L}_i^*(\mathsf{r}_{i,n} - \mathsf{u}_{i,n})\Big) \\ \mathsf{y}_{n+1} = \mathsf{y}_n - \dfrac{\mathsf{p}_n - \mathsf{q}_n}{\gamma}. \end{array} \right. \tag{4.5}$$

Then, $(\mathsf{x}_n)_{n \in \mathbb{N}}$ *is a sequence in* V *and there exists* $(\overline{\mathsf{x}}, \overline{\mathsf{u}}_1, \ldots, \overline{\mathsf{u}}_m) \in \boldsymbol{Z}$ *such that* $\mathsf{x}_n \rightharpoonup \overline{\mathsf{x}}$ *and, for every* $i \in \{1, \ldots, m\}$, $\mathsf{u}_{i,n} \rightharpoonup \overline{\mathsf{u}}_i$.

Proof Set $\mathcal{H} = \mathsf{H} \oplus \mathsf{G}_1 \oplus \cdots \oplus \mathsf{G}_m$ and define

$$\begin{cases} A : \mathcal{H} \to 2^{\mathcal{H}} : (x, u_1, \ldots, u_m) \mapsto Ax \times B_1^{-1}u_1 \times \cdots \times B_m^{-1}u_m \\ B : \mathcal{H} \to \mathcal{H} : (x, u_1, \ldots, u_m) \mapsto \big(Mx + \sum_{i=1}^m L_i^* u_i, N_1^{-1}u_1 - L_1 x, \ldots, N_m^{-1}u_m - L_m x\big) \\ C : \mathcal{H} \to \mathcal{H} : (x, u_1, \ldots, u_m) \mapsto (Cx, D_1^{-1}u_1, \ldots, D_m^{-1}u_m) \\ V = \big\{(x, u_1, \ldots, u_m) \in \mathcal{H} \;\big|\; x \in V\big\}. \end{cases} \tag{4.6}$$

Then, A is maximally monotone and B is monotone and L-Lipschitzian [22, eq.(3.11)], C is β-cocoercive [51, eq.(3.12)], and V is a closed vector subspace of $\mathcal{H}$. Therefore, Problem 4.1 is a particular instance of Problem 1.1. Moreover, we have from [3, Proposition 23.18] that

$$\begin{cases} (\forall \gamma > 0) \quad J_{\gamma A} : (x, u_1, \ldots, u_m) \mapsto (J_{\gamma A}x, J_{\gamma B_1^{-1}}u_1, \ldots, J_{\gamma B_m^{-1}}u_m) \\ P_V : (x, u_1, \ldots, u_m) \mapsto (P_V x, u_1, \ldots, u_m). \end{cases} \tag{4.7}$$

Altogether, by defining

$$(\forall n \in \mathbb{N}) \quad \begin{cases} x_n = (x_n, u_{1,n}, \ldots, u_{m,n}) \\ y_n = (y_n, 0, \ldots, 0) \\ p_n = (p_n, r_{1,n}, \ldots, r_{m,n}) \\ q_n = (q_n, s_{1,n}, \ldots, s_{m,n}), \end{cases} \tag{4.8}$$

(4.5) is a particular case of (3.1) and the convergence follows from Corollary 3.3.

Remark 4.3 In the particular case when $V = H$ and $C = D_1^{-1} = \cdots = D_m^{-1} = 0$, Proposition 4.2 recovers the main result in [22, Theorem 3.1] in the error-free case. By including non-standard metrics in the space $\mathcal{H}$ as in [14], we can also recover [15] when $M = N_1^{-1} = \cdots = N_m^{-1} = 0$ and [51] if we additionally assume that $V = H$, but we preferred to avoid this generalization for simplicity.

We now provide two important examples of Problem 4.1 and Proposition 4.2 in the context of convex optimization.

Example 4.4 Suppose that $A = \partial f$, $M = N_1^{-1} = \cdots = N_m^{-1} = 0$, $C = \nabla h$, for every $i \in \{1, \ldots, m\}$, $D_i = \partial \ell_i$ and $B_i = \partial g_i$, where $f \in \Gamma_0(H)$, $h : H \to \mathbb{R}$ is convex differentiable with ζ^{-1}-Lipschitzian gradient, for every $i \in \{1, \ldots, m\}$, $\ell_i \in \Gamma_0(G_i)$ is ν_i-strongly convex and $g_i \in \Gamma_0(G_i)$. Then under the qualification condition [22, Proposition 4.3(i)]

$$(0, \ldots, 0) \in \operatorname{sri}\Big(\times_{i=1}^m \big(L_i(V \cap \operatorname{dom} f) - (\operatorname{dom} g_i + \operatorname{dom} \ell_i)\big)\Big), \tag{4.9}$$

Problem 4.1 is equivalent to

$$\min_{x \in V}\Big(f(x) + h(x) + \sum_{i=1}^m (g_i \square \ell_i)(L_i x)\Big), \tag{4.10}$$

which, in view of Proposition 4.2, can be solved by the algorithm

$$
\begin{bmatrix}
\text{for } n = 0, 1, 2, \ldots \\
\mathsf{p}_n = \operatorname{prox}_{\gamma \mathsf{f}}\Big(\mathsf{x}_n + \gamma \mathsf{y}_n - \gamma P_{\mathsf{V}}\Big(\nabla \mathsf{h}(\mathsf{x}_n) + \sum_{i=1}^{m} \mathsf{L}_i^* \mathsf{u}_{i,n}\Big)\Big) \\
\mathsf{q}_n = P_{\mathsf{V}} \mathsf{p}_n \\
\begin{bmatrix}
\text{for } i = 1, \ldots, m \\
\mathsf{r}_{i,n} = \operatorname{prox}_{\gamma \mathsf{g}_i^*}\big(\mathsf{u}_{i,n} - \gamma(\nabla \ell_i^*(\mathsf{u}_{i,n}) - \mathsf{L}_i \mathsf{x}_n)\big) \\
\mathsf{u}_{i,n+1} = \mathsf{r}_{i,n} + \gamma \mathsf{L}_i(\mathsf{q}_n - \mathsf{x}_n)
\end{bmatrix} \\
\mathsf{x}_{n+1} = \mathsf{q}_n - \gamma P_{\mathsf{V}}\Big(\sum_{i=1}^{m} \mathsf{L}_i^*(\mathsf{r}_{i,n} - \mathsf{u}_{i,n})\Big) \\
\mathsf{y}_{n+1} = \mathsf{y}_n - \dfrac{\mathsf{p}_n - \mathsf{q}_n}{\gamma},
\end{bmatrix}
\tag{4.11}
$$

where $\mathsf{x}_0 \in V$, $\mathsf{y}_0 \in V^{\perp}$, for every $i \in \{1, \ldots, m\}$, $\mathsf{u}_{i,0} \in \mathsf{G}_i$, $L = \sqrt{\sum_{i=1}^{m} \|\mathsf{L}_i\|^2}$, $\beta = \min\{\zeta, \delta_i, \ldots, \delta_m\}$, χ is defined in (3.2), and $\gamma \in]0, \chi[$. Observe that the algorithm (4.11) exploits the cocoercivity of $\nabla \mathsf{h}$ and $(\nabla \ell_i^*)_{1 \le i \le m}$ by implementing them only once by iteration a difference of [22, Theorem 4.2], which needs to implement them twice by iteration.

Example 4.5 Consider the convex minimization problem

$$
\min_{x \in \mathcal{H}} \big(f(x) + g(\mathcal{L}x) + h(\mathcal{A}x)\big), \tag{4.12}
$$

where $\mathcal{H}$, $\mathcal{G}$, and $\mathcal{K}$ are real Hilbert spaces, $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{G})$, $\mathcal{L} : \mathcal{H} \to \mathcal{G}$, $\mathcal{A} : \mathcal{H} \to \mathcal{K}$, $h : \mathcal{K} \to \mathbb{R}$ is convex, differentiable with β^{-1}-Lipschitzian gradient, and suppose that

$$
0 \in \operatorname{sri}(\mathcal{L}\ \operatorname{dom} f - \operatorname{dom} g). \tag{4.13}
$$

Note that $h \circ \mathcal{A}$ is convex, differentiable, and $\nabla(h \circ \mathcal{A}) = \mathcal{A}^* \circ \nabla h \circ \mathcal{A}$ is $\beta^{-1}\|\mathcal{A}\|^2$– Lipschitzian. Then, (4.12) can be solved by the primal-dual algorithm proposed in [24, 51], whose convergence is guaranteed under the assumption

$$
\sigma \|\mathcal{L}\|^2 \le \frac{1}{\tau} - \frac{\|\mathcal{A}\|^2}{2\beta}, \tag{4.14}
$$

where $\tau > 0$ and $\sigma > 0$ are primal and dual step-sizes, respectively. Observe that, when $\|\mathcal{A}\|$ is large, this method is forced to choose small primal and dual step-sizes in order to ensure convergence. To overcome this inconvenient, we propose the following formulation

$$
\min_{\mathsf{x} \in \mathsf{V}} \big(\mathsf{f}(\mathsf{x}) + \mathsf{h}(\mathsf{x}) + \mathsf{g}(\mathsf{L}\mathsf{x})\big), \tag{4.15}
$$

where

$$\begin{cases} \mathsf{H} = \mathcal{H} \oplus \mathcal{K} \\ \mathsf{G} = \mathcal{G} \\ \mathsf{T} : \mathsf{x} = (x, w) \mapsto \mathcal{A}x - w \\ \mathsf{V} = \ker \mathsf{T} \\ \mathsf{f} : \mathsf{x} = (x, w) \mapsto f(x) \\ \mathsf{g} = g \\ \mathsf{L} : \mathsf{x} = (x, w) \mapsto \mathcal{L}x \\ \mathsf{h} : \mathsf{x} = (x, w) \mapsto h(w). \end{cases} \tag{4.16}$$

Since in this case (4.9) reduces to (4.13), (4.12) is a particular instance of (4.10) when $m = 1$ and $\ell_1 = 0$. Therefore, in view of [3, Example 29.19], (4.12) can be solved by the routine in (4.11) which, on this setting, reduces to:

$$\begin{bmatrix} \text{for } n = 0,1,2,\ldots \\ p_{1,n} = \operatorname{prox}_{\gamma f}\Big(x_n + \gamma y_{1,n} - \gamma\big(\mathcal{L}^* u_n - \mathcal{A}^*\mathcal{B}\big(\mathcal{A}\mathcal{L}^* u_n - \nabla h(w_n)\big)\big)\Big) \\ p_{2,n} = w_n + \gamma y_{2,n} - \gamma\big(\nabla h(w_n) + \mathcal{B}\big(\mathcal{A}\mathcal{L}^* u_n - \nabla h(w_n)\big)\big) \\ q_{1,n} = p_{1,n} - \mathcal{A}^*\mathcal{B}\big(\mathcal{A}p_{1,n} - p_{2,n}\big) \\ q_{2,n} = p_{2,n} + \mathcal{B}\big(\mathcal{A}p_{1,n} - p_{2,n}\big) \\ r_n = \operatorname{prox}_{\gamma g^*}\big(u_n + \gamma \mathcal{L}x_n\big) \\ u_{n+1} = r_n + \gamma \mathcal{L}(q_{1,n} - x_n) \\ x_{n+1} = q_{1,n} - \gamma\big(\mathcal{L}^*(r_n - u_n) - \mathcal{A}^*\mathcal{B}\mathcal{A}\mathcal{L}^*(r_n - u_n)\big) \\ w_{n+1} = q_{2,n} - \gamma\mathcal{B}\mathcal{A}\mathcal{L}^*(r_n - u_n) \\ y_{1,n+1} = y_{1,n} - \dfrac{p_{1,n+1} - q_{1,n+1}}{\gamma} \\ y_{2,n+1} = y_{2,n} - \dfrac{p_{2,n+1} - q_{2,n+1}}{\gamma}, \end{bmatrix} \tag{4.17}$$

where $\mathcal{B} = (\mathrm{Id} + \mathcal{A}\mathcal{A}^*)^{-1}$ can be computed only once before the loop, $(x_0, w_0) \in V$, $(y_{1,0}, y_{2,0}) \in V^\perp$, $u_0 \in \mathcal{G}$, $L = \|\mathcal{L}\|$, χ is defined in (3.2), and $\gamma \in \left]0, \chi\right[$.

5 Numerical Experiments

In this section we consider the following optimization problem

$$\min_{y^0 \le x \le y^1} \left(\frac{\alpha_1}{2}\|\mathcal{A}x - z\|^2 + \alpha_2\|\nabla x\|_1\right), \tag{5.1}$$

where $y^0 = (\eta_i^0)_{1\le i\le N}$, $y^1 = (\eta_i^1)_{1\le i\le N}$ are vectors in $\mathbb{R}^N$, α_1 and α_2 are in $]0, +\infty[$, $\mathcal{A} \in \mathbb{R}^{K\times N}$, $z \in \mathbb{R}^K$, and $\nabla : \mathbb{R}^N \to \mathbb{R}^{N-1} : (\xi_i)_{1\le i\le N} \mapsto (\xi_{i+1} - \xi_i)_{1\le i\le N-1}$ is the discrete

Let $x_0 \in \mathbb{R}^N$ and $u_0 \in \mathbb{R}^{N-1}$, let $(\sigma, \tau, \rho) \in]0, +\infty[^3$, and fix $\epsilon_0 > \varepsilon > 0$.
while $\epsilon_n > \varepsilon$ **do**
  $p_{n+1} = P_C\big(x_n - \tau(\alpha_1 \mathcal{A}^\top(\mathcal{A}x_n - z) + \nabla^\top u_n)\big)$
  $q_{n+1} = \sigma(\mathrm{Id} - \mathrm{prox}_{\alpha_2\|\cdot\|_1/\sigma})(u_n/\sigma + \nabla(2p_{n+1} - x_n))$
  $x_{n+1} = x_n + \rho(p_{n+1} - x_n)$
  $u_{n+1} = u_n + \rho(q_{n+1} - u_n)$
  $\epsilon_{n+1} = \mathcal{R}\big((x_{n+1}, u_{n+1}), (x_n, u_n)\big)$
end while
return (x_{n+1}, u_{n+1})

Algorithm 1 Condat-Vũ [24, 51]

Set $\mathcal{B} = (\mathrm{Id} + \mathcal{A}\mathcal{A}^\top)^{-1}$, let $(x_0, w_0) \in (\ker \mathsf{T})^2$, $(y_{1,0}, y_{2,0}) \in (\ker \mathsf{T}^\perp)^2$, $u_0 \in \mathbb{R}^K$, let $\gamma \in]0, +\infty[$, and fix $\epsilon_0 > \varepsilon > 0$.
while $\epsilon_n > \varepsilon$ **do**
  $p_{1,n} = P_C\big(x_n + \gamma y_{1,n} - \gamma\big(\nabla^\top u_n - \mathcal{A}^\top\mathcal{B}\big(\mathcal{A}\nabla^\top u_n - \alpha_1(w_n - z)\big)\big)\big)$
  $p_{2,n} = w_n + \gamma y_{2,n} - \gamma\big(\alpha_1(w_n - z) + \mathcal{B}\big(\mathcal{A}\nabla^\top u_n - \alpha_1(w_n - z)\big)\big)$
  $q_{1,n} = p_{1,n} - \mathcal{A}^\top\mathcal{B}(\mathcal{A}p_{1,n} - p_{2,n})$
  $q_{2,n} = p_{2,n} + \mathcal{B}\big(\mathcal{A}p_{1,n} - p_{2,n}\big)$
  $r_n = \gamma(\mathrm{Id} - \mathrm{prox}_{\alpha_2\|\cdot\|_1/\gamma})(u_n/\gamma + \nabla x_n)$
  $u_{n+1} = r_n + \gamma\nabla(q_{1,n} - x_n)$
  $x_{n+1} = q_{1,n} - \gamma\big(\nabla^\top(r_n - u_n) - \mathcal{A}^\top\mathcal{B}\mathcal{A}\nabla^\top(r_n - u_n)\big)$
  $w_{n+1} = q_{2,n} - \gamma\mathcal{B}\mathcal{A}\nabla^\top(r_n - u_n)$
  $y_{1,n+1} = y_{1,n} - (p_{1,n+1} - q_{1,n+1})/\gamma$
  $y_{2,n+1} = y_{2,n} - (p_{2,n+1} - q_{2,n+1})/\gamma$
  $\epsilon_{n+1} = \mathcal{R}\big((x_{n+1}, w_{n+1}, y^1_{n+1}, y^2_{n+1}), (x_n, w_n, y^1_n, y^2_n)\big)$
end while
return $(x_{n+1}, w_{n+1}, y^1_{n+1}, y^2_{n+1})$

Algorithm 2 Forward-partial inverse-half-forward splitting (FPIHF)

gradient. This problem appears when computing the fusion estimator in fused LASSO problems [31, 42, 49].

Note that (5.1) can be written equivalently as (4.12), where

$$\begin{cases} \mathcal{H} = \mathbb{R}^N \\ f = \iota_C \\ C = \times_{i=1}^N [\eta_i^0, \eta_i^1] \\ g = \alpha_2\|\cdot\|_1 \\ h = \alpha_1\|\cdot - z\|^2/2 \\ L = \nabla \end{cases} \tag{5.2}$$

Since $f \in \Gamma_0(\mathbb{R}^N)$, $g \in \Gamma_0(\mathbb{R}^{N-1})$, h is convex, differentiable, $\nabla h = \alpha_1(\mathrm{Id} - z)$ is α_1–Lipschitzian, $\|L\| = 2$, and (4.13) is trivially satisfied, (5.1) is a particular instance of Example 4.5. Hence, (5.1) can be solved by the algorithm in [24, 51] (called Condat-Vũ),

by (4.17) (called FPIHF), and by [11] (called FPIF), which are compared in this section. In this context, the Algorithm Condat-Vũ [24, 51] reduces to Algorithm 1.

Observe that $P_C : (\xi_i)_{1\leq i\leq N} \mapsto (\max\{\min\{\xi_i, \eta_i^1\}, \eta_i^0\})_{1\leq i\leq N}$. The convergence of Algorithm 1 is guaranteed if

$$\sigma\|\mathcal{L}\|^2 \leq \frac{1}{\tau} - \frac{\alpha_1\|\mathcal{A}\|^2}{2} \quad \text{and} \quad \rho \in]0, \delta[\,, \quad \text{where } \delta = 2 - \frac{\alpha_1\|\mathcal{A}\|^2}{2(\frac{1}{\tau} - \sigma\|\mathcal{L}\|^2)}. \tag{5.3}$$

Note that, the larger is $\alpha_1\|\mathcal{A}\|^2$, the smaller should be τ and σ in order to achieve convergence. On the other hand, by considering T defined in (4.16) and Example 4.5, the algorithm in (4.17) reduces to Algorithm 2, whose convergence is guaranteed if the step-size γ satisfies

$$0 < \gamma < \chi = \frac{4}{\alpha_1 + \sqrt{\alpha_1^2 + 64}}. \tag{5.4}$$

Observe that the condition for the step-size γ in (5.4) does not depend on $\|\mathcal{A}\|$.

The FPIF algorithm proposed in [11] for solving (5.1) differs from Algorithm 2 in the fact that the cocoercive gradient $\nabla h : x \mapsto x - z$ is implemented twice by iteration. Indeed, the algorithm consider the monotone Lipschitzian operator $(x, u, w) \mapsto (\nabla^\top u, \nabla x, \alpha_1(w - z))$, whose Lipschitz constant follows from

$$\begin{aligned}
|||\big(\nabla^\top u_1, \nabla x_1, \alpha_1(w_1 - z)\big) - \big(\nabla^\top u_2, \nabla x_2, \alpha_1(w_2 - z)\big)|||^2 \\
&= \|\nabla^\top(u_1 - u_2)\|^2 + \|\nabla(x_1 - x_2)\|^2 + \alpha_1^2\|w_1 - w_2\|^2 \\
&\leq \|\nabla^\top\|^2\|u_1 - u_2\|^2 + \|\nabla\|^2\|x_1 - x_2\|^2 + \alpha_1^2\|w_1 - w_2\|^2 \\
&\leq \max\{\|\nabla\|^2, \alpha_1^2\}|||(x_1 - x_2, u_1 - u_2, w_1 - w_2)|||^2.
\end{aligned}$$

Therefore, the convergence of FPIF is guaranteed if $\gamma \in]0, 1/\max\{\|\nabla\|, \alpha_1\}[$, and, as in Algorithm 2, this condition does not depend on $\|\mathcal{A}\|$. In order to compare Condat-Vũ, FPIHF, and FPIF, we set $\alpha_1 = 5$ and $\alpha_2 = 0.5$ and we consider $\mathcal{A} = \kappa \cdot \mathrm{rand}(N, K)$, $y^0 = -1.5 \cdot \mathrm{rand}(N)$, $y^1 = 1.5 \cdot \mathrm{rand}(N)$, and $z = \mathrm{randn}(N)$, where $\kappa \in \{1/5, 1/10, 1/20, 1/30\}$, $N \in \{600, 1200, 2400\}$, $K \in \{N/3, N/2, 2N/3\}$, and $\mathrm{rand}(\cdot,\cdot)$ and $\mathrm{randn}(\cdot,\cdot)$ are functions in MATLAB generating matrices/vectors with uniformly and normal distributed entries, respectively. For each value of κ, N, and K, we generate 20 random realizations for $\mathcal{A}$, z, y^0, and y^1. Note that the average value of $\|\mathcal{A}\|$ increases as κ increase (see Fig. 1 for $K = N/2$), which affects Algorithm 1 in view of (5.3). We also set $\rho = 0.99 \cdot \delta$, where δ is defined in (5.3). In this setting, from (5.4) we deduce that the convergence of FPIHF is guaranteed for $\gamma < \chi \approx 0.2771$. On the other hand, since $\max\{\|\nabla\|, \alpha_1\} = \alpha_1 = 5$, the convergence of FPIF is guaranteed for $\gamma < 0.2$.

In Tables 1, 2, 3, 4 we provide the average time and number of iterations to achieve a tolerance $\varepsilon = 10^{-6}$ for each algorithm under study. In the case when an algorithm exceeds 50000 iterations in all cases, we write "⊠" in both columns. From these tables we can observe that when κ increases (and therefore, $\|\mathcal{A}\|$ increases), Condat-Vũ reduces its performance and does not converge within 50000 iterations for big dimensions and large values of κ. Moreover, the number of iterations of FPIHP is considerably lower than its competitors but with expensive computational time by iteration. This can be explained by the fact that FPIHP needs to compute three projections onto the kernel of $(x, w) \mapsto \mathcal{A}x - w$ at each iteration. We can also perceive that, at exception of some

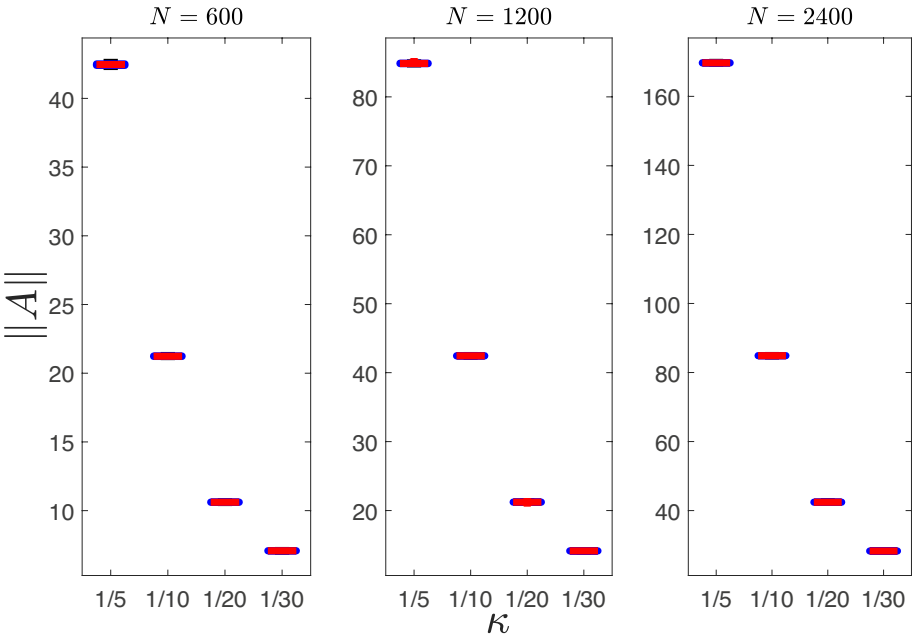

Fig. 1 Box plot for the norm of the 20 random realizations of $\mathcal{A}$, $N \in \{600, 1200, 2400\}$, $K = N/2$

Table 1 Comparison of Condat-Vũ, FPIF, and FPIHF for the case $\kappa = 1/30$

		$K = N/3$		$K = N/2$		$K = 2N/3$	
N	Algorithm	Av. time (s)	Av. iter	Av. time (s)	Av. iter	Av. time (s)	Av. iter
600	Condat-Vũ	0.89	11059	0.80	10047	0.76	9666
	FPIF	3.46	17454	3.91	14353	7.20	17430
	FPIHF	0.99	4851	1.24	4442	1.73	3996
1200	Condat-Vũ	11.32	17321	10.55	16129	10.54	16082
	FPIF	25.52	19930	32.37	13788	51.54	16443
	FPIHF	7.07	5425	13.76	5838	23.83	7570
2400	Condat-Vũ	74.17	34059	70.14	32216	69.48	31963
	FPIF	95.55	17747	138.67	16074	190.06	17216
	FPIHF	43.08	7961	64.68	7464	70.64	6369

cases, the partial inverse-based algorithms increase their computational time to achieve convergence when K is larger. This can be explained by the fact that the dimension of matrix $\mathcal{B}$ is larger as K is larger, and it has to be implemented three times by iteration.

When $\kappa = 1/30$, we observe from Table 1, that FPIHP and Condat-Vũ are competitive and both are more efficient than FPIF. When $\kappa = 1/20$, we observe from Table 2 that FPIHP outperforms Condat-Vũ and FPIF for large dimensions. When $\kappa = 1/10$, we observe from Table 3 that FPIHP is the best algorithm at exception of the smallest

Table 2 Comparison of Condat-Vũ, FPIF, and FPIHF for the case $\kappa = 1/20$

		$K = N/3$		$K = N/2$		$K = 2N/3$	
N	Algorithm	Av. time (s)	Av. iter	Av. time (s)	Av. iter	Av. time (s)	Av. iter
600	Condat-Vũ	0.86	10752	0.81	10263	0.87	10992
	FPIF	2.67	13381	3.91	14204	5.88	14258
	FPIHF	0.97	4725	0.82	2900	1.63	3747
1200	Condat-Vũ	13.91	21209	13.35	20359	12.51	19118
	FPIF	23.30	18142	45.16	19222	52.60	16773
	FPIHF	9.07	6943	20.53	8689	10.91	3458
2400	Condat-Vũ	103.92	47673	98.92	45543	91.33	41996
	FPIF	89.77	16659	132.60	15374	145.58	13181
	FPIHF	32.27	5957	45.35	5234	83.48	7539

Table 3 Comparison of Condat-Vũ, FPIF, and FPIHF for the case $\kappa = 1/10$

		$K = N/3$		$K = N/2$		$K = 2N/3$	
N	Algorithm	Av. time (s)	Av. iter	Av. time (s)	Av. iter	Av. time (s)	Av. iter
600	Condat-Vũ	1.43	18233	1.30	16747	1.25	15577
	FPIF	3.56	18040	3.01	11057	5.17	12389
	FPIHF	1.11	5414	1.30	4696	1.49	3436
1200	Condat-Vũ	30.19	46078	26.98	41243	24.05	36849
	FPIF	25.61	19916	30.70	13095	40.57	12960
	FPIHF	6.96	5343	10.16	4294	17.79	5657
2400	Condat-Vũ	☒	☒	☒	☒	☒	☒
	FPIF	98.90	18363	129.27	14975	172.05	15609
	FPIHF	28.90	5349	46.74	5391	60.61	5484

Table 4 Comparison of Condat-Vũ, FPIF, and FPIHF for the case $\kappa = 1/5$

		$K = N/3$		$K = N/2$		$K = 2N/3$	
N	Algorithm	Av. time (s)	Av. iter	Av. time (s)	Av. iter	Av. time (s)	Av. iter
600	Condat-Vũ	3.76	48078	3.27	40998	2.58	33226
	FPIF	2.68	13527	3.31	11945	4.14	9840
	FPIHF	0.50	2428	0.64	2263	0.79	1780
1200	Condat-Vũ	☒	☒	☒	☒	☒	☒
	FPIF	21.26	16535	27.29	11627	35.55	11399
	FPIHF	7.23	5529	5.72	2424	10.25	3257
2400	Condat-Vũ	☒	☒	☒	☒	☒	☒
	FPIF	88.51	16392	124.71	14444	139.69	12653
	FPIHF	23.95	4414	35.51	4102	41.38	3773

dimensional case in which it is competitive with Condat-Vũ. The latter does not converge within 50000 for dimension $N = 2400$. When $\kappa = 1/5$, FPIHP is the more efficient algorithm in all the cases under study, as it is illustrated in Table 4. Moreover, Condat-Vũ converge before 50000 iterations only in the lower dimensional case when $N = 600$. We conclude that, for higher values of $\|\mathcal{A}\|$ and larger dimensions, is more convenient to implement FPIHP.

Acknowledgements The first author thanks the support of Centro de Modelamiento Matemático (CMM), ACE210010 and FB210005, BASAL funds for centers of excellence, grant FONDECYT 1190871, and grant Redes 180032 from ANID-Chile. The third author thanks the support of ANID-Subdirección de Capital Humano/Doctorado Nacional/2018-21181024 and by the Dirección de Postgrado y Programas from UTFSM through Programa de Incentivos a la Iniciación Científica (PIIC).

References

1. Attouch, H., Cabot, A.: Convergence of a relaxed inertial forward-backward algorithm for structured monotone inclusions. Appl. Math. Optim. **80**, 547–598 (2019)
2. Aubin, J.-P., Frankowska, H.: Set-Valued Analysis. Modern Birkhäuser Classics, Birkhäuser Boston Inc, Boston, MA (2009)
3. Bauschke, H.H., Combettes, P.L.: Convex Analysis and Monotone Oerator Theory in Hilbert Spaces, CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, Springer, Cham, second ed., (2017)
4. Boţ, R.I., Csetnek, E.R.: An inertial forward-backward-forward primal-dual splitting algorithm for solving monotone inclusion problems. Numer. Algorithms **71**, 519–540 (2016)
5. Boţ, R.I., Csetnek, E.R.: ADMM for monotone operators: convergence analysis and rates. Adv. Comput. Math. **45**, 327–359 (2019)
6. Boţ, R.I., Csetnek, E.R., Heinrich, A.: A primal-dual splitting algorithm for finding zeros of sums of maximal monotone operators. SIAM J. Optim. **23**, 2011–2036 (2013)
7. Boţ, R.I., Csetnek, E.R., Hendrich, C.: Inertial Douglas-Rachford splitting for monotone inclusion problems. Appl. Math. Comput. **256**, 472–487 (2015)
8. Boţ, R.I., Hendrich, C.: A Douglas-Rachford type primal-dual method for solving inclusions with mixtures of composite and parallel-sum type monotone operators. SIAM J. Optim. **23**, 2541–2565 (2013)
9. Briceño, L., Cominetti, R., Cortés, C.E., Martínez, F.: An integrated behavioral model of land use and transport system: a hyper-network equilibrium approach. Netw. Spat. Econ. **8**, 201–224 (2008)
10. Briceño-Arias, L.M.: Forward-Douglas-Rachford splitting and forward-partial inverse method for solving monotone inclusions. Optimization **64**, 1239–1261 (2015)
11. Briceño-Arias, L.M.: Forward-partial inverse-forward splitting for solving monotone inclusions. J. Optim. Theory Appl. **166**, 391–413 (2015)
12. Briceño-Arias, L.M., Combettes, P.L.: A monotone + skew splitting model for composite monotone inclusions in duality. SIAM J. Optim. **21**, 1230–1250 (2011)
13. Briceño-Arias, L.M., Combettes, P.L.: Monotone Operator Methods for Nash Equilibria in Non-Potential Games. In: Bailey, D.H., Bauschke, H.H., Borwein, P., Garvan, F., Thera, M., Vanderwerff, J., Wolkowicz, H. (eds.) Computational and Analytical Mathematics, vol. 50, pp. 143–159. Springer Proceedings in Mathematics & Statistics, New York, (2013)
14. Briceño-Arias, L.M., Davis, D.: Forward-backward-half forward algorithm for solving monotone inclusions. SIAM J. Optim. **28**, 2839–2871 (2018)
15. Briceño-Arias, L.M., Deride, J., López-Rivera, S., Silva, F.J.: A primal-dual partial inverse splitting for constrained monotone inclusions: Applications to stochastic programming and mean field games, (2022). https://arxiv.org/abs/2007.01983
16. Bùi, M.N., Combettes, P.L.: Multivariate monotone inclusions in saddle form. Math. Oper. Res. 47, 1082–1109 (2022)
17. Cevher, V., Vũ, B.C.: A reflected forward-backward splitting method for monotone inclusions involving Lipschitzian operators. Set-Valued Var. 29, 163–174 (2021)
18. Chambolle, A., Lions, P.-L.: Image recovery via total variation minimization and related problems. Numer. Math. **76**, 167–188 (1997)

19. Colas, J., Pustelnik, N., Oliver, C., Abry, P., Géminard, J.-C., Vidal, V.: Nonlinear denoising for characterization of solid friction under low confinement pressure. Phys. Rev. E 100, 032803 (2019)
20. Combettes, P.L.: Systems of structured monotone inclusions: duality, algorithms, and applications. SIAM J. Optim. **23**, 2420–2447 (2013)
21. Combettes, P.L., Eckstein, J.: Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions. Math. Program. **168**, 645–672 (2018)
22. Combettes, P.L., Pesquet, J.-C.: Primal-dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum type monotone operators. Set-Valued Var. Anal. **20**, 307–330 (2012)
23. Combettes, P.L., Vũ, B.C.: Variable metric forward-backward splitting with applications to monotone inclusions in duality. Optimization **63**, 1289–1318 (2014)
24. Condat, L.: A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. J. Optim. Theory Appl. **158**, 460–479 (2013)
25. Csetnek, E., Malitsky, Y., Tam, M.: Shadow Douglas-Rachford splitting for monotone inclusions. Appl. Math. Optim. **80**, 665–678 (2019)
26. Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Comm. Pure Appl. Math. **57**, 1413–1457 (2004)
27. Davis, D., Yin, W.: A three-operator splitting scheme and its optimization applications. Set-Valued Var. Anal. **25**, 829–858 (2017)
28. Dong, Y.: Weak convergence of an extended splitting method for monotone inclusions. J. Global Optim. **79**, 257–277 (2021)
29. Dũng, D., Vũ, B.C.: A splitting algorithm for system of composite monotone inclusions. Vietnam J. Math. **43**, 323–341 (2015)
30. Eckstein, J.: A simplified form of block-iterative operator splitting and an asynchronous algorithm resembling the multi-block alternating direction method of multipliers. J. Optim. Theory Appl. **173**, 155–182 (2017)
31. Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. Ann. Appl. Stat. **1**, 302–332 (2007)
32. Fukushima, M.: The primal Douglas-Rachford splitting algorithm for a class of monotone mappings with application to the traffic equilibrium problem. Math. Program. **72**, 1–15 (1996)
33. Gabay, D.: Chapter IX Applications of the method of multipliers to variational inequalities. In: Fortin, M., Glowinski, R. (eds.) Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, pp. 299–331. Studies in Mathematics and Its Applications, Elsevier (1983)
34. Gafni, E.M., Bertsekas, D.P.: Two-metric projection methods for constrained optimization. SIAM J. Control Optim. **22**, 936–964 (1984)
35. Glowinski, R., Marroco, A.: Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite, d'une classe de problemes de dirichlet non lineares. Revue Francaise d'Automatique, Informatique et Recherche Operationelle **9**, 41–76 (1975)
36. Goldstein, A.A.: Convex programming in Hilbert space. Bull. Amer. Math. Soc. **70**, 709–710 (1964)
37. Johnstone, P.R., Eckstein, J.: Projective splitting with forward steps only requires continuity. Optim. Lett. **14**, 229–247 (2020)
38. Johnstone, P.R., Eckstein, J.: Single-forward-step projective splitting: exploiting cocoercivity. Comput. Optim. Appl. **78**, 125–166 (2021)
39. Latafat, P., Patrinos, P.: Asymmetric forward-backward-adjoint splitting for solving monotone inclusions involving three operators. Comput. Optim. Appl. **68**, 57–93 (2017)
40. Lions, P.-L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**, 964–979 (1979)
41. Malitsky, Y., Tam, M.K.: A forward-backward splitting method for monotone inclusions without cocoercivity. SIAM J. Optim. **30**, 1451–1472 (2020)
42. Ohishi, M., Fukui, K., Okamura, K., Itoh, Y., Yanagihara, H.: Coordinate optimization for generalized fused Lasso. Comm. Statist. Theory Methods **50**, 5955–5973 (2021)
43. Raguet, H., Fadili, J., Peyré, G.: A generalized forward-backward splitting. SIAM J. Imaging Sci. **6**, 1199–1226 (2013)
44. Rieger, J., Tam, M.K.: Backward-forward-reflected-backward splitting for three operator monotone inclusions. Appl. Math. Comput. **381**, 125248, 10 (2020)
45. Ryu, E.K., Vũ, B.C.: Finding the forward-Douglas-Rachford-forward method. J. Optim. Theory Appl. **184**, 858–876 (2020)

46. Showalter, R.E.: Monotone Operators in Banach Space and Nonlinear Partial Differential Equations. Mathematical Surveys and Monographs, vol. 49. American Mathematical Society, Providence, RI (1997)
47. Spingarn, J.E.: Partial inverse of a monotone operator. Appl. Math. Optim. **10**, 247–265 (1983)
48. Spingarn, J.E.: Applications of the method of partial inverses to convex programming: decomposition. Math. Program. **32**, 199–223 (1985)
49. Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K.: Sparsity and smoothness via the fused lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. **67**, 91–108 (2005)
50. Tseng, P.: A modified forward-backward splitting method for maximal monotone mappings. SIAM J. Control Optim. **38**, 431–446 (2000)
51. Vũ, B.C.: A splitting algorithm for dual monotone inclusions involving cocoercive operators. Adv. Comput. Math. **38**, 667–681 (2013)