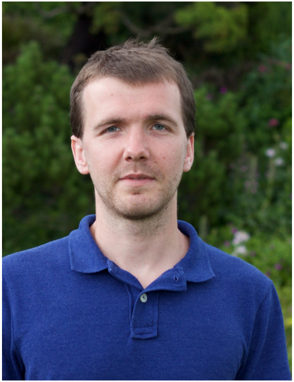
Splitting Algorithms with Forward Steps

Matthew K. Tam

Institute for Numerical and Applied Mathematics
Georg-August-Universität Göttingen

ICCOPT, August 4-8, 2019

Splitting Algorithms with Forward Steps



Joint work with Robert Csetnek and Yura Malitsky

Proximal Gradient Descent

Consider the minimisation problem

$$\min_{x\in\mathcal{H}} f(x) + g(x),$$

where:

- $g : \mathcal{H} \to (-\infty, +\infty]$ is proper, lsc and convex, and
- $f : \mathcal{H} \to \mathbb{R}$ is convex with L-Lipschitz gradient ∇f.

- Solutions characterised by the first order optimality condition:

$$0 \in (A + B)(x) \quad \text{where} \quad A := \partial g \text{ and } B := \nabla f.$$

- Can be solved using proximal gradient descent with $\lambda \in (0, 2/L)$:

$$x_{n+1} := \operatorname{prox}_{\lambda g}(x_k - \lambda \nabla f(x_k)) \quad \forall n \in \mathbb{N}.$$

Proximal Gradient Descent

Consider the minimisation problem

$$\min_{x \in \mathcal{H}} f(x) + g(x),$$

where:

- $g : \mathcal{H} \to (-\infty, +\infty]$ is proper, lsc and convex, and
- $f : \mathcal{H} \to \mathbb{R}$ is convex with L-Lipschitz gradient ∇f.

- Solutions characterised by the first order optimality condition:

$$0 \in (A + B)(x) \quad \text{where} \quad A := \partial g \text{ and } B := \nabla f.$$

- Can be solved using proximal gradient descent with $\lambda \in (0, 2/L)$:

$$x_{n+1} := \text{prox}_{\lambda g}(x_k - \lambda \nabla f(x_k)) \quad \forall n \in \mathbb{N}.$$

The Forward-Backward Method

Abstracted to the framework of monotone operators, the previous minimisation problem becomes the monotone inclusion:

$$\text{find } x \in \mathcal{H} \text{ such that } 0 \in (A+B)(x),$$

where:

- $A : \mathcal{H} \rightrightarrows \mathcal{H}$ is maximally monotone, and
- $B : \mathcal{H} \to \mathcal{H}$ is monotone and L-Lipschitz continuous.

Proximal gradient generalises to the forward-backward algorithm:

$$x_{k+1} := J_{\lambda A}(x_k - \lambda B(x_k)),$$

where $J_{\lambda A} := (I + \lambda A)^{-1}$ is the resolvent of the monotone operator λA.

The Forward-Backward Method

The standard proof of the forward-backward algorithm requires:

- $A = N_C$ and $B = \nabla f$ are both (maximal) monotone operators:

$$\langle x - u, y - v \rangle \geq 0 \quad \forall y \in A(x), \forall v \in A(u).$$

- $B = \nabla f$ is β-cocoercive (equiv. B^{-1} is strongly monotone):

$$\langle x - y, Bx - By \rangle \geq \beta \|Bx - By\|^2,$$

which implies B is $\frac{1}{\beta}$-Lipschitz. The converse is not true in general.

Theorem (Baillon–Haddad)

Let $f : \mathcal{H} \to \mathbb{R}$ be a Fréchet differentiable convex function and let $L > 0$. Then ∇f is L-Lipschitz continuous if and only if ∇f is $(1/L)$-cocoercive.

⋆ Proximal gradient descent **converges** when $B = \nabla f$ is L-Lipschitz because, in this case, the operator B is actually $\frac{1}{L}$-cocoercive!

- If B is merely Lipschitz need (for instance):
  - Chen–Rockafellar 1997 – $A + B$ is strongly monotone.
  - Bello Cruz–Días Millán 2015 – Backtracking procedure.

The Forward-Backward Method

The standard proof of the forward-backward algorithm requires:

- $A = N_C$ and $B = \nabla f$ are both (maximal) monotone operators:

$$\langle x - u, y - v \rangle \geq 0 \quad \forall y \in A(x), \forall v \in A(u).$$

- $B = \nabla f$ is β-cocoercive (equiv. B^{-1} is strongly monotone):

$$\langle x - y, Bx - By \rangle \geq \beta \|Bx - By\|^2,$$

  which implies B is $\frac{1}{\beta}$-Lipschitz. The converse is not true in general.

Theorem (Baillon–Haddad)

Let $f \colon \mathcal{H} \to \mathbb{R}$ be a Fréchet differentiable convex function and let $L > 0$. Then ∇f is L-Lipschitz continuous if and only if ∇f is $(1/L)$-cocoercive.

⋆ Proximal gradient descent **converges** when $B = \nabla f$ is L-Lipschitz because, in this case, the operator B is actually $\frac{1}{L}$-cocoercive!

- If B is merely Lipschitz need (for instance):
  - Chen–Rockafellar 1997 – $A + B$ is strongly monotone.
  - Bello Cruz–Días Millán 2015 – Backtracking procedure.

The Forward-Backward Method

The standard proof of the forward-backward algorithm requires:

- $A = N_C$ and $B = \nabla f$ are both (maximal) monotone operators:

$$\langle x - u, y - v \rangle \geq 0 \quad \forall y \in A(x), \forall v \in A(u).$$

- $B = \nabla f$ is β-cocoercive (equiv. B^{-1} is strongly monotone):

$$\langle x - y, Bx - By \rangle \geq \beta \|Bx - By\|^2,$$

which implies B is $\frac{1}{\beta}$-Lipschitz. The converse is not true in general.

Theorem (Baillon–Haddad)

Let $f\colon \mathcal{H} \to \mathbb{R}$ be a Fréchet differentiable convex function and let $L > 0$. Then ∇f is L-Lipschitz continuous if and only if ∇f is $(1/L)$-cocoercive.

⋆ Proximal gradient descent **converges** when $B = \nabla f$ is L-Lipschitz because, in this case, the operator B is actually $\frac{1}{L}$-cocoercive!

- If B is merely Lipschitz need (for instance):
  - Chen–Rockafellar 1997 – $A + B$ is strongly monotone.
  - Bello Cruz–Días Millán 2015 – Backtracking procedure.

The Forward-Backward Method

The standard proof of the forward-backward algorithm requires:

- $A = N_C$ and $B = \nabla f$ are both (maximal) monotone operators:

$$\langle x - u, y - v \rangle \geq 0 \quad \forall y \in A(x), \forall v \in A(u).$$

- $B = \nabla f$ is β-cocoercive (equiv. B^{-1} is strongly monotone):

$$\langle x - y, Bx - By \rangle \geq \beta \|Bx - By\|^2,$$

which implies B is $\frac{1}{\beta}$-Lipschitz. The converse is not true in general.

Theorem (Baillon–Haddad)

Let $f \colon \mathcal{H} \to \mathbb{R}$ be a Fréchet differentiable convex function and let $L > 0$. Then ∇f is L-Lipschitz continuous if and only if ∇f is $(1/L)$-cocoercive.

⋆ Proximal gradient descent **converges** when $B = \nabla f$ is L-Lipschitz because, in this case, the operator B is actually $\frac{1}{L}$-cocoercive!

- If B is merely Lipschitz need (for instance):
  - Chen–Rockafellar 1997 – $A + B$ is strongly monotone.
  - Bello Cruz–Días Millán 2015 – Backtracking procedure.

Nonsmooth Convex Minimisation

Consider the minimisation problem

$$\min_{x \in \mathcal{H}} f(x) + g(Kx),$$

where:

- $f, g : \mathcal{H} \to (-\infty, +\infty]$ are proper, lsc and convex.
- $K : \mathcal{H} \to \mathcal{H}$ is a linear, bounded operator with adjoint K^*.

Using Fenchel duality, this can be cast as "$0 \in (A+B)(z)$" with

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \Bigg(\underbrace{\begin{bmatrix} \partial g & 0 \\ 0 & \partial f^* \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} 0 & K^* \\ -K & 0 \end{bmatrix}}_{B} \Bigg) \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{z} \subseteq \mathcal{H} \times \mathcal{H}.$$

The operator B is $\|K\|$-Lipschitz continuous but **not** cocoercive.

Nonsmooth Convex Minimisation

Consider the minimisation problem

$$\min_{x\in\mathcal{H}} f(x)+g(Kx),$$

where:

- $f,g:\mathcal{H}\to(-\infty,+\infty]$ are proper, lsc and convex.
- $K:\mathcal{H}\to\mathcal{H}$ is a linear, bounded operator with adjoint K^*.

Using Fenchel duality, this can be cast as "$0\in(A+B)(z)$" with

$$\begin{pmatrix}0\\0\end{pmatrix}\in\left(\underbrace{\begin{bmatrix}\partial g & 0\\ 0 & \partial f^*\end{bmatrix}}_{A}+\underbrace{\begin{bmatrix}0 & K^*\\ -K & 0\end{bmatrix}}_{B}\right)\underbrace{\begin{pmatrix}x\\y\end{pmatrix}}_{z}\subseteq\mathcal{H}\times\mathcal{H}.$$

The operator B is $\|K\|$-Lipschitz continuous but **not** cocoercive.

Nonsmooth Convex Minimisation

Consider the minimisation problem

$$\min_{x \in \mathcal{H}} f(x) + g(Kx),$$

where:

- $f, g : \mathcal{H} \to (-\infty, +\infty]$ are proper, lsc and convex.
- $K : \mathcal{H} \to \mathcal{H}$ is a linear, bounded operator with adjoint K^*.

Using Fenchel duality, this can be cast as "$0 \in (A + B)(z)$" with

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \left(\underbrace{\begin{bmatrix} \partial g & 0 \\ 0 & \partial f^* \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} 0 & K^* \\ -K & 0 \end{bmatrix}}_{B} \right) \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{z} \subseteq \mathcal{H} \times \mathcal{H}.$$

The operator B is $\|K\|$-Lipschitz continuous but **not** cocoercive.

Saddle Point Problems

Consider the saddle point problem

$$\min_{x\in\mathcal{H}} \max_{y\in\mathcal{H}} g(x) + \Phi(x,y) - f(y),$$

where:

- $f, g \colon \mathcal{H} \to (-\infty, +\infty]$ are proper, lsc and convex.
- $\Phi \colon \mathcal{H}\times\mathcal{H} \to \mathbb{R}$ is convex-concave with Lipschitz gradient.

First-order optimality condition yields "$0 \in (A+B)(z)$" with

$$\begin{pmatrix}0\\0\end{pmatrix} \in \underbrace{\begin{pmatrix}\partial g(x)\\ \partial f(y)\end{pmatrix}}_{A(z)} + \underbrace{\begin{pmatrix}\nabla_x\Phi(x,y)\\ -\nabla_y\Phi(x,y)\end{pmatrix}}_{B(z)} \subseteq \mathcal{H}\times\mathcal{H}.$$

Again, the operator B is Lipschitz but **not** cocoercive.

Saddle Point Problems

Consider the saddle point problem

$$\min_{x \in \mathcal{H}} \max_{y \in \mathcal{H}} g(x) + \Phi(x, y) - f(y),$$

where:

- $f, g \colon \mathcal{H} \to (-\infty, +\infty]$ are proper, lsc and convex.
- $\Phi \colon \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is convex-concave with Lipschitz gradient.

First-order optimality condition yields "$0 \in (A + B)(z)$" with

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \underbrace{\begin{pmatrix} \partial g(x) \\ \partial f(y) \end{pmatrix}}_{A(z)} + \underbrace{\begin{pmatrix} \nabla_x \Phi(x, y) \\ -\nabla_y \Phi(x, y) \end{pmatrix}}_{B(z)} \subseteq \mathcal{H} \times \mathcal{H}.$$

Again, the operator B is Lipschitz but **not** cocoercive.

Forward-backward Splitting with Cocoercivity

In summary:

	Cocoercivity?	Lipschitz?
Smooth + nonsmooth minimisation	✓	✓
Nonsmooth + nonsmooth minimisation	✗	✓
Saddle point problems	✗	✓

Goal: Splitting algorithms that:

- Only use $J_{\lambda A}$ (backward step) and B (forward step).
- Converge when B is Lipschitz (but not necessarily cocoercive).

Operator Splitting without Cocoercivity

Theorem (Tseng 2000)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0 \in \mathcal{H}$, let $\lambda \in (0, \frac{1}{L})$, and set

$$y_k = J_{\lambda A}(x_k - \lambda B(x_k))$$
$$x_{k+1} = y_k - \lambda B(y_k) + \lambda B(x_k).$$

Then (x_n) and (y_n) both converge weakly to a point $x \in (A+B)^{-1}(0)$.

- Requires one backward and two forward evaluations per iteration.
- Maximum stepsize is half that of the forward-backward algorithm.
- Fejér monotone. In fact, (x_k) satisfies

$$\|x_{k+1} - x\|^2 + \epsilon \|x_k - y_k\|^2 \le \|x_k - x\|^2 .$$

- Variant of Tseng's method studied by Combettes–Pesquet 2012.

Another approaches to the same problem:

- Eckstein–Johnstone 2019 – Projective splitting w/forward steps.
- Bang Cong Vu's talk from yesterday.

Operator Splitting without Cocoercivity

Theorem (Tseng 2000)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0 \in \mathcal{H}$, let $\lambda \in (0, \frac{1}{L})$, and set

$$\begin{aligned} y_k &= J_{\lambda A}(x_k - \lambda B(x_k)) \\ x_{k+1} &= y_k - \lambda B(y_k) + \lambda B(x_k). \end{aligned}$$

Then (x_n) and (y_n) both converge weakly to a point $x \in (A+B)^{-1}(0)$.

- Requires one backward and two forward evaluations per iteration.
- Maximum stepsize is half that of the forward-backward algorithm.
- Fejér monotone. In fact, (x_k) satisfies

$$\|x_{k+1} - x\|^2 + \epsilon \|x_k - y_k\|^2 \leq \|x_k - x\|^2 .$$

- Variant of Tseng's method studied by Combettes–Pesquet 2012.

Another approaches to the same problem:

- Eckstein–Johnstone 2019 – Projective splitting w/forward steps.
- Bang Cong Vu's talk from yesterday.

Forward-Reflected-Backward Splitting

Theorem (Malitsky–T.)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{2L}\right)$, and set

$$x_{k+1} = J_{\lambda A}\big(x_k - 2\lambda B(x_k) + \lambda B(x_{k-1})\big). \tag{1}$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $x \in \mathcal{H}$ such that $0 \in (A+B)(x)$.

- Converges under the exact same assumptions as Tseng's method.
- Requires one backward and **one** forward evaluation per iteration.
- The maximal permissible stepsize is half that of Tseng's method.
- Linesearch procedure when B is locally Lipschitz. (1) becomes:

$$x_{k+1} = J_{\lambda A}\big(x_k - (\lambda_k + \lambda_{k-1})B(x_k) + \lambda_{k-1}B(x_{k-1})\big).$$

Forward-Reflected-Backward Splitting

Theorem (Malitsky–T.)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{2L}\right)$, and set

$$x_{k+1} = J_{\lambda A}\big(x_k - 2\lambda B(x_k) + \lambda B(x_{k-1})\big). \tag{1}$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $x \in \mathcal{H}$ such that $0 \in (A+B)(x)$.

- Converges under the exact same assumptions as Tseng's method.
- Requires one backward and **one** forward evaluation per iteration.
- The maximal permissible stepsize is half that of Tseng's method.
- Linesearch procedure when B is locally Lipschitz. (1) becomes:

$$x_{k+1} = J_{\lambda A}\big(x_k - (\lambda_k + \lambda_{k-1})B(x_k) + \lambda_{k-1}B(x_{k-1})\big).$$

Forward-Reflected-Backward Splitting

Theorem (Malitsky–T.)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{2L}\right)$, and set

$$x_{k+1} = J_{\lambda A}\big(x_k - 2\lambda B(x_k) + \lambda B(x_{k-1})\big). \tag{1}$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $x \in \mathcal{H}$ such that $0 \in (A+B)(x)$.

- Converges under the exact same assumptions as Tseng's method.
- Requires one backward and **one** forward evaluation per iteration.
- The maximal permissible stepsize is half that of Tseng's method.
- Linesearch procedure when B is locally Lipschitz. (1) becomes:

$$x_{k+1} = J_{\lambda A}\big(x_k - (\lambda_k + \lambda_{k-1})B(x_k) + \lambda_{k-1}B(x_{k-1})\big).$$

Forward-Reflected-Backward Splitting

Theorem (Malitsky–T.)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz with $(A+B)^{-1}(0) \neq \emptyset$. Let $\lambda \in \left(0, \frac{1}{2L}\right)$. Given $x_0 \in \mathcal{H}$, define the sequence $(x_k)_{k\in\mathbb{N}}$ according to

$$x_{k+1} = J_{\lambda A}\big(x_k - 2\lambda B(x_k) + \lambda B(x_{k-1})\big) \quad \forall k \in \mathbb{N}.$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

Proof sketch. Let $x \in (A+B)^{-1}(0)$ and consider $(\varphi_k)_{k\in\mathbb{N}} \subseteq \mathbb{R}$ given by

$$\varphi_k := \|x_k - x\|^2 + 2\lambda \langle B(x_k) - B(x_{k-1}), x - x_k\rangle + \frac{1}{2}\|x_k - x_{k-1}\|^2.$$

Then $\varphi_k \geq \frac{1}{2}\|x_k - x\|^2$ and there exists an $\varepsilon > 0$ such that

$$\varphi_{k+1} + \varepsilon \|x_{k+1} - x_k\|^2 \leq \varphi_k.$$

Forward-Reflected-Backward Splitting

Theorem (Malitsky–T.)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz with $(A+B)^{-1}(0) \neq \emptyset$. Let $\lambda \in \left(0, \frac{1}{2L}\right)$. Given $x_0 \in \mathcal{H}$, define the sequence $(x_k)_{k\in\mathbb{N}}$ according to

$$x_{k+1} = J_{\lambda A}\big(x_k - 2\lambda B(x_k) + \lambda B(x_{k-1})\big) \quad \forall k \in \mathbb{N}.$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

Proof sketch. Let $x \in (A+B)^{-1}(0)$ and consider $(\varphi_k)_{k\in\mathbb{N}} \subseteq \mathbb{R}$ given by

$$\varphi_k := \|x_k - x\|^2 + 2\lambda \langle B(x_k) - B(x_{k-1}), x - x_k \rangle + \frac{1}{2}\|x_k - x_{k-1}\|^2.$$

Then $\varphi_k \geq \frac{1}{2}\|x_k - x\|^2$ and there exists an $\varepsilon > 0$ such that

$$\varphi_{k+1} + \varepsilon \|x_{k+1} - x_k\|^2 \leq \varphi_k.$$

Are Larger Stepsizes Always Better?

Consider the monotone inclusion

$$0 \in (A+B)(x) \subseteq \mathbb{R}^n \times \mathbb{R}^n,$$

where $A(x_1, x_2) = (0, 0)$ and $B(x_1, x_2) = (x_2, -x_1)$. Then:

- Zero is the unique solution of the problem.
- B is 1-Lipschitz and monotone, but not cocoercive.

- Tseng's method: For $\lambda \in (0, 1)$, converges Q-linearly with rate

$$\rho(\lambda) := \sqrt{1 - \lambda^2 + \lambda^4} < 1.$$

  Thus, best convergence rate is $\rho(\lambda) = \sqrt{3}/2$ with $\lambda = 1/\sqrt{2}$.
- FoRB: For $\lambda \approx 1/2$, converges with rate given by $\rho(\lambda) \approx \sqrt{2}/2$.

Larger stepsizes are **not** necessarily better.

Shadow Douglas–Rachford Splitting

Theorem (Csetnek–Malitsky–T. 2019)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{3L}\right)$ and set

$$x_{k+1} = J_{\lambda A}\big(x_k - \lambda B(x_k)\big) - \lambda\big(B(x_k) - B(x_{k-1})\big).$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

- Converges under the exact same assumptions as Tseng's method.
- Requires one backward and **one** forward evaluation per iteration.
- **Open question:** How to incorporate a linesearch procedure?

Shadow Douglas–Rachford Splitting

Theorem (Csetnek–Malitsky–T. 2019)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{3L}\right)$ and set

$$x_{k+1} = J_{\lambda A}\big(x_k - \lambda B(x_k)\big) - \lambda\big(B(x_k) - B(x_{k-1})\big).$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

- Converges under the exact same assumptions as Tseng's method.
- Requires one backward and **one** forward evaluation per iteration.
- **Open question:** How to incorporate a linesearch procedure?

Shadow Douglas–Rachford Splitting

Theorem (Csetnek–Malitsky–T. 2019)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz with $(A+B)^{-1}(0) \neq \emptyset$. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{3L}\right)$ and set

$$x_{k+1} = J_{\lambda A}\big(x_k - \lambda B(x_k)\big) - \lambda\big(B(x_k) - B(x_{k-1})\big).$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

Proof sketch. Let $x \in (A+B)^{-1}$. Set $y_k = \lambda B(x_k)$ and $y = \lambda B(x)$. Then

$$\|(x_{k+1}+y_k)-(x+y)\|^2 + \left(\frac{1}{3}+\epsilon\right)\|x_{k+1}-x_k\|^2 \leq \|(x_k+y_{k-1})-(x+y)\|^2 + \frac{1}{3}\|x_k - x_{k-1}\|^2.$$

Shadow Douglas–Rachford Splitting

Theorem (Csetnek–Malitsky–T. 2019)

Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximally monotone and $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz with $(A+B)^{-1}(0) \neq \emptyset$. Let $x_0, x_{-1} \in \mathcal{H}$, let $\lambda \in \left(0, \frac{1}{3L}\right)$ and set

$$x_{k+1} = J_{\lambda A}\big(x_k - \lambda B(x_k)\big) - \lambda\big(B(x_k) - B(x_{k-1})\big).$$

Then $(x_k)_{k\in\mathbb{N}}$ converges weakly to some $\overline{x} \in \mathcal{H}$ such that $0 \in (A+B)(\overline{x})$.

Proof sketch. Let $x \in (A+B)^{-1}$. Set $y_k = \lambda B(x_k)$ and $y = \lambda B(x)$. Then

$$\|(x_{k+1}+y_k) - (x+y)\|^2 + \left(\frac{1}{3}+\epsilon\right)\|x_{k+1}-x_k\|^2 \leq \|(x_k+y_{k-1}) - (x+y)\|^2 + \frac{1}{3}\|x_k - x_{k-1}\|^2.$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \tag{DR}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \tag{S-DR}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_{k+1} - y_k$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_{k+1} - y_k),$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \qquad \text{(DR)}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \qquad \text{(S-DR)}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_{k+1} - y_k$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_{k+1} - y_k),$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \tag{DR}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \tag{S-DR}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_{k+1} - y_k$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_{k+1} - y_k),$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \tag{DR}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \tag{S-DR}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_{k+1} - y_k$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_{k+1} - y_k),$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \tag{DR}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \tag{S-DR}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_k - y_{k-1}$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_k - y_{k-1}),$$

Shadow Douglas–Rachford Splitting

Douglas–Rachford splitting for "$0 \in (A+B)(z)$":

$$z_{k+1} = z_k + J_{\lambda A}(2J_{\lambda B}(z_k) - z_k) - J_{\lambda B}(z_k).$$

Substitute $z(t) = z_k$ and $\dot{z}(t) = z_{k+1} - z_k$, gives the dynamical system

$$\dot{z}(t) = J_{\lambda A}(2J_{\lambda B}(z(t)) - z(t)) - J_{\lambda B}(z(t)). \tag{DR}$$

Express in terms of the shadow trajectory "$x(t) = J_{\lambda B}(z(t))$":

$$\begin{cases} \dot{x}(t) = J_{\lambda A}(x(t) - y(t)) - x(t) - \dot{y}(t), \\ y(t) = \lambda B(x(t)), \end{cases} \tag{S-DR}$$

where we note that $x(t) = (I + \lambda B)^{-1}(z(t)) \iff x(t) + y(t) = z(t)$.

Discretising with $\dot{x}(t) = x_{k+1} - x_k$ and $\dot{y}(t) = y_k - y_{k-1}$ gives

$$x_{k+1} = J_{\lambda A}(x_t - y_k) - (y_k - y_{k-1}),$$

which is exactly the iteration from the previous slide.

An Application: Optimistic Gradient Descent Ascent

Recall, the inclusion associated with the saddle point problem:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \underbrace{\begin{pmatrix} \partial g(x) \\ \partial f(y) \end{pmatrix}}_{A(z)} + \underbrace{\begin{pmatrix} \nabla_x \Phi(x,y) \\ -\nabla_y \Phi(x,y) \end{pmatrix}}_{B(z)} \subseteq \mathcal{H} \times \mathcal{H}.$$

Applying the forward reflected backward method yields

$$\begin{cases} x_{k+1} = \text{prox}_{\lambda g}\big(x_k - 2\lambda \nabla_x \Phi(x_k, y_k) + \lambda \nabla_x \Phi(x_k, y_k)\big) \\ y_{k+1} = \text{prox}_{\lambda f}\big(y_k + 2\lambda \nabla_y \Phi(x_k, y_k) - \lambda \nabla_y \Phi(x_k, y_k)\big). \end{cases}$$

In the case when $f = g = 0$, FoRB spltting:

- Coincides with the shadow Douglas–Rachford method.
- Coincides with optimistic gradient descent ascent (OGDA) method from ML used for training generative adverserial networks (GANs) (Daskalaski et al, 2018).

An Application: Optimistic Gradient Descent Ascent

Recall, the inclusion associated with the saddle point problem:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \underbrace{\begin{pmatrix} \partial g(x) \\ \partial f(y) \end{pmatrix}}_{A(z)} + \underbrace{\begin{pmatrix} \nabla_x \Phi(x,y) \\ -\nabla_y \Phi(x,y) \end{pmatrix}}_{B(z)} \subseteq \mathcal{H} \times \mathcal{H}.$$

Applying the forward reflected backward method yields

$$\begin{cases} x_{k+1} = \operatorname{prox}_{\lambda g}\big(x_k - 2\lambda \nabla_x \Phi(x_k, y_k) + \lambda \nabla_x \Phi(x_k, y_k)\big) \\ y_{k+1} = \operatorname{prox}_{\lambda f}\big(y_k + 2\lambda \nabla_y \Phi(x_k, y_k) - \lambda \nabla_y \Phi(x_k, y_k)\big). \end{cases}$$

In the case when $f = g = 0$, FoRB splitting:

- Coincides with the shadow Douglas–Rachford method.
- Coincides with optimistic gradient descent ascent (OGDA) method from ML used for training generative adverserial networks (GANs) (Daskalaski et al, 2018).

An Application: Optimistic Gradient Descent Ascent

Recall, the inclusion associated with the saddle point problem:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \underbrace{\begin{pmatrix} \partial g(x) \\ \partial f(y) \end{pmatrix}}_{A(z)} + \underbrace{\begin{pmatrix} \nabla_x \Phi(x,y) \\ -\nabla_y \Phi(x,y) \end{pmatrix}}_{B(z)} \subseteq \mathcal{H} \times \mathcal{H}.$$

Applying the forward reflected backward method yields

$$\begin{cases} x_{k+1} = \operatorname{prox}_{\lambda g}\big(x_k - 2\lambda \nabla_x \Phi(x_k, y_k) + \lambda \nabla_x \Phi(x_k, y_k)\big) \\ y_{k+1} = \operatorname{prox}_{\lambda f}\big(y_k + 2\lambda \nabla_y \Phi(x_k, y_k) - \lambda \nabla_y \Phi(x_k, y_k)\big). \end{cases}$$

In the case when $f = g = 0$, FoRB spltting:

- Coincides with the shadow Douglas–Rachford method.
- Coincides with optimistic gradient descent ascent (OGDA) method from ML used for training generative adverserial networks (GANs) (Daskalaski et al, 2018).

Concluding Remarks

- Two simple modification of the forward-backward algorithm allows for the assumption of cocoercivity to avoided.
- Only require one evaluation of B per iteration (Tseng needs two).

Open questions and directions for further reserach:

- Do there exist a useful fixed point interpretation of the methods?
- Is there a continuous dynamical system associated with the FoRB?
- Can a linesearch be incorporated into the shadow DR method?

A forward-backward splitting method for monotone inclusions without cocoercivity with Y. Malitsky. `arXiv:1808.04162`.

Shadow Douglas-Rachford splitting for monotone inclusions with E.R. Csetnek and Malitsky. *Appl Math & Optim*, p. 1–14, 2019.