gipsa-lab

Proximal splitting methods on convex problems with a quadratic term: Relax!

The slides I presented with added comments

Laurent Condat

GIPSA-lab, Univ. Grenoble Alpes, France

Workshop BASP Frontiers, Jan. 2017

Proximal splitting algorithms

A zoo of methods in the literature

ADMM

ISTA

Douglas-Rachford

PDFB

Chambolle-Pock

This is the primal-dual forward-backward algorithm
I proposed, sometimes called ‘Condat-Vu algorithm’

Goal

We consider the problem

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

where

- the $\mathcal{U}_m$ and $\mathcal{X}$ are real Hilbert spaces,
- the functions g_m are convex, from $\mathcal{U}_m$ to $\mathbb{R} \cup \{+\infty\}$,
- the L_m are linear operators from $\mathcal{X}$ to $\mathcal{U}_m$.

Goal

We consider the problem

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

We want full splitting, with individual activation of L_m, L_m^*, the gradient or proximity operator of g_m.

Goal

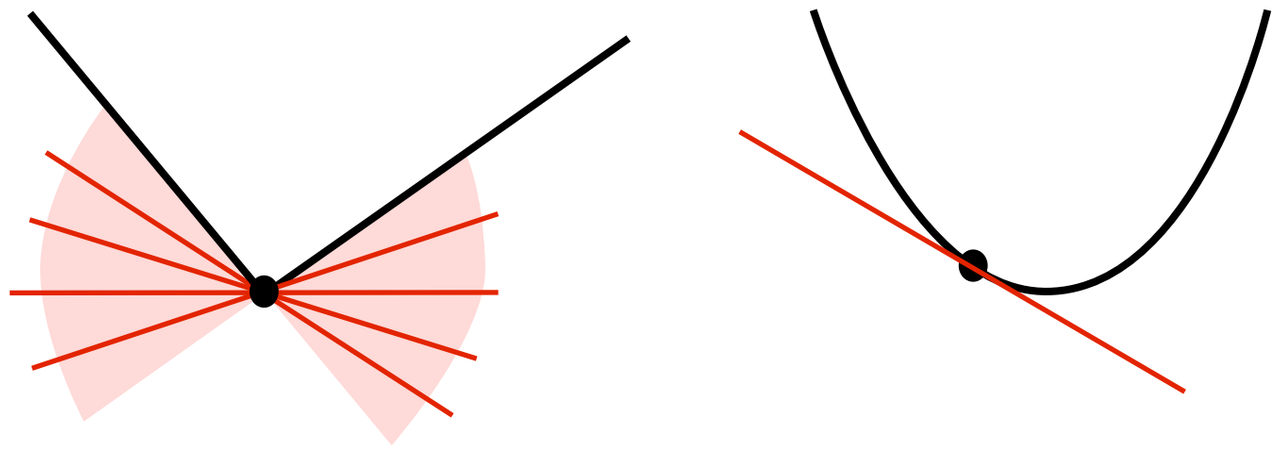
We consider the problem

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

We want full splitting, with individual activation of L_m, L_m^*, the gradient or proximity operator of g_m.

☞ no implicit operation (inner loop or linear system to solve)

The subdifferential



$\partial f : \mathcal{X} \to 2^{\mathcal{X}}$

$\partial f(x)$ is the set of gradients of the affine minorants of f at x.

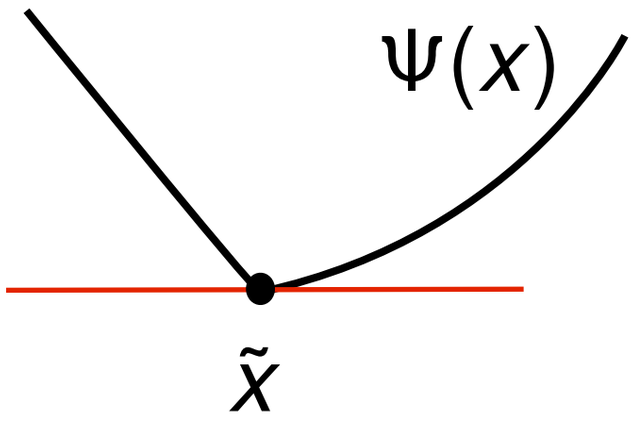
f is smooth at $x \to \partial f(x) = \{\nabla f(x)\}$.

Fermat's rule

$$\tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

$$\equiv$$

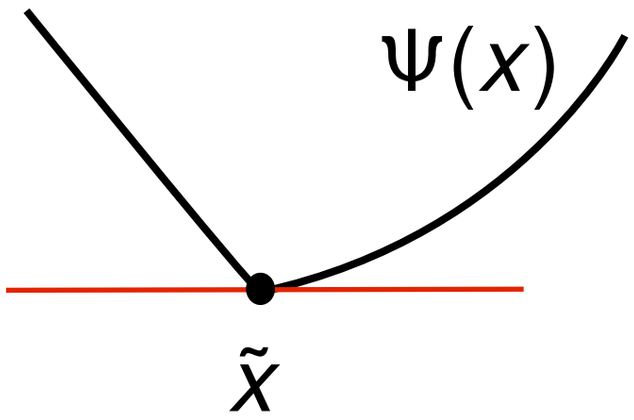
$$0 \in \partial\Psi(\tilde{x})$$



Pierre de Fermat, 1601-1665

Fermat's rule

$$\tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$



$\equiv$

$$0 \in \partial\Psi(\tilde{x})$$

$\equiv$

$$0 \in \sum_{m=1}^{M} L_m^* \partial g_m(L_m \tilde{x})$$

Pierre de Fermat, 1601-1665

Here some qualification constraints are hidden

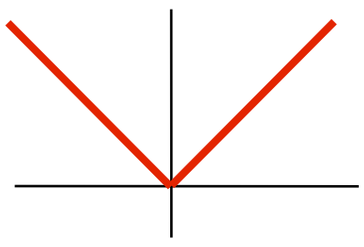
The proximity operator

$$\text{prox}_f : \mathcal{X} \to \mathcal{X}, \; x \mapsto \underset{z \in \mathcal{X}}{\arg\min} \; f(z) + \frac{1}{2}\|z - x\|^2$$

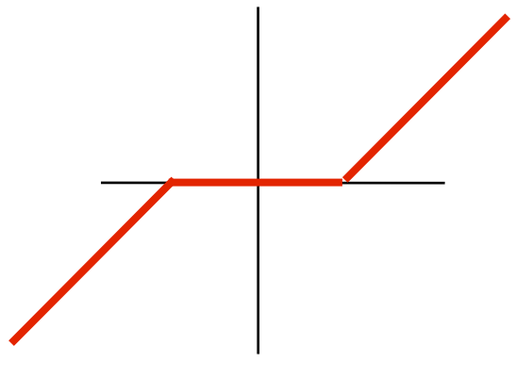
The proximity operator

$$\text{prox}_f : \mathcal{X} \to \mathcal{X},\ x \mapsto \underset{z \in \mathcal{X}}{\arg\min}\ f(z) + \tfrac{1}{2}\|z - x\|^2$$

Explicit forms of prox_f for a large class of functions.



$f(x) = |x|$



$$\text{prox}_f(x) = \text{sgn}(x)\max(|x| - 1, 0)$$

The proximity operator

$$\text{prox}_f : \mathcal{X} \to \mathcal{X},\ x \mapsto \underset{z \in \mathcal{X}}{\arg\min}\ f(z) + \frac{1}{2}\|z - x\|^2$$

$$\forall \gamma > 0,\ \boxed{\text{prox}_{\gamma f} = (\text{Id} + \gamma \partial f)^{-1}}$$

Iterative algorithms

Principle: design an algorithm which iterates

$x^{(i+1)} = T(x^{(i)})$

for some operator $T : \mathcal{X} \to \mathcal{X}$

Iterative algorithms

Principle: design an algorithm which iterates

$x^{(i+1)} = T(x^{(i)})$

for some operator $T : \mathcal{X} \to \mathcal{X}$

Convergence: $\|x^{(i)} - \tilde{x}\| \to 0$, for some solution $\tilde{x}$.

Iterative algorithms

Principle: design an algorithm which iterates

$$x^{(i+1)} = T(x^{(i)})$$

for some **nonexpansive** operator $T : \mathcal{X} \to \mathcal{X}$

i.e. $\forall x, y, \ \|T(x) - T(y)\| \leq \|x - y\|$,

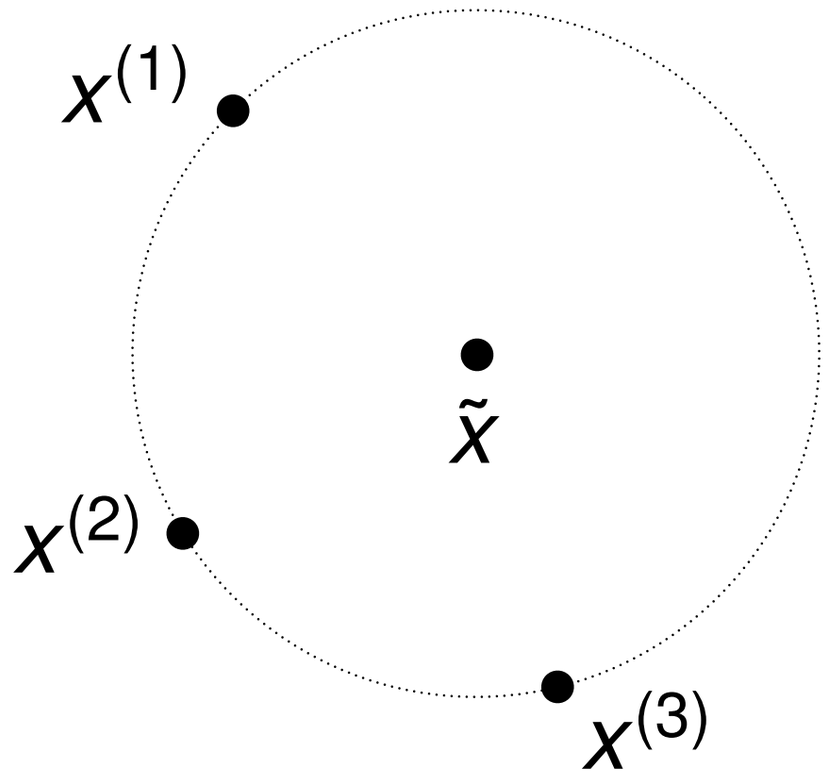
and such that the set S of solutions is the set of **fixed points** of T, i.e. $T(\tilde{x}) = \tilde{x}$.

Here replace y by a fixed point and you get Fejer-monotonicity: at every iteration you get closer to the solution set

Iterative algorithms

But nonexpansiveness is not sufficient :

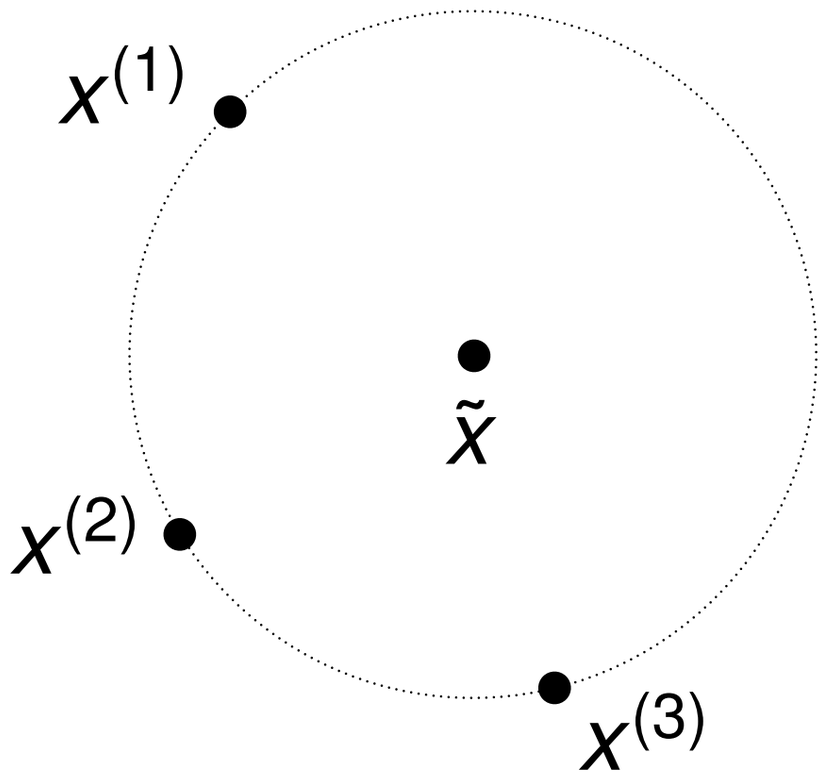
for instance, $\mathcal{X} = \mathbb{R}^2$ and T is a rotation.



Iterative algorithms

But nonexpansiveness is not sufficient :

for instance, $\mathcal{X} = \mathbb{R}^2$ and T is a rotation.



we need a bit more
than nonexpansiveness.

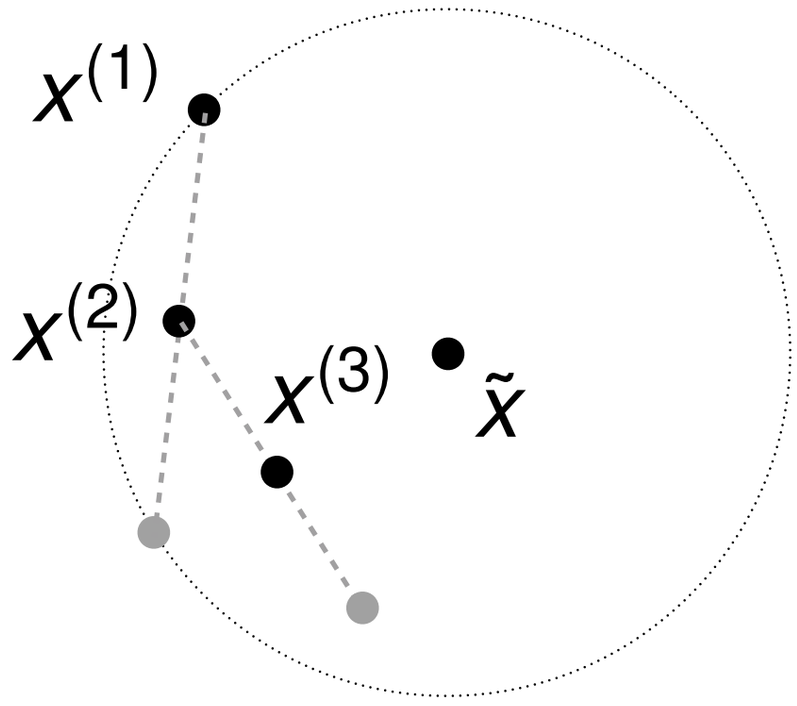
"firmly nonexpansive" = 1/2-averaged

Iterative algorithms

But nonexpansiveness is not sufficient :

for instance, $\mathcal{X} = \mathbb{R}^2$ and T is a rotation.

Definition: T is **α-averaged** if
$T = \alpha T' + (1 - \alpha)\mathrm{Id}$,
for some $0 < \alpha < 1$ and
nonexpansive op. T'.



Iterative algorithms

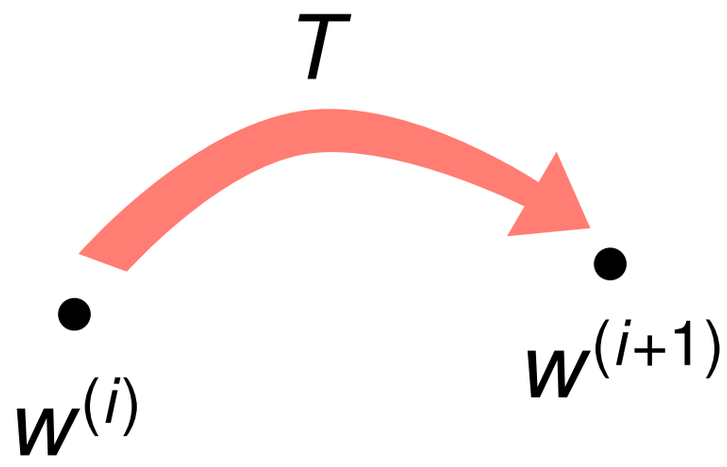
Theorem (Krasnoselskii–Mann)
If T is $\alpha-$averaged, for some $0 < \alpha < 1$,
and a fixed point of T exists,
then the iteration $x^{(i+1)} = T(x^{(i)})$
converges to some fixed point $\tilde{x}$ of T.

Over-relaxation

An algorithm:

$w^{(i+1)} = T(w^{(i)})$

with T α-averaged

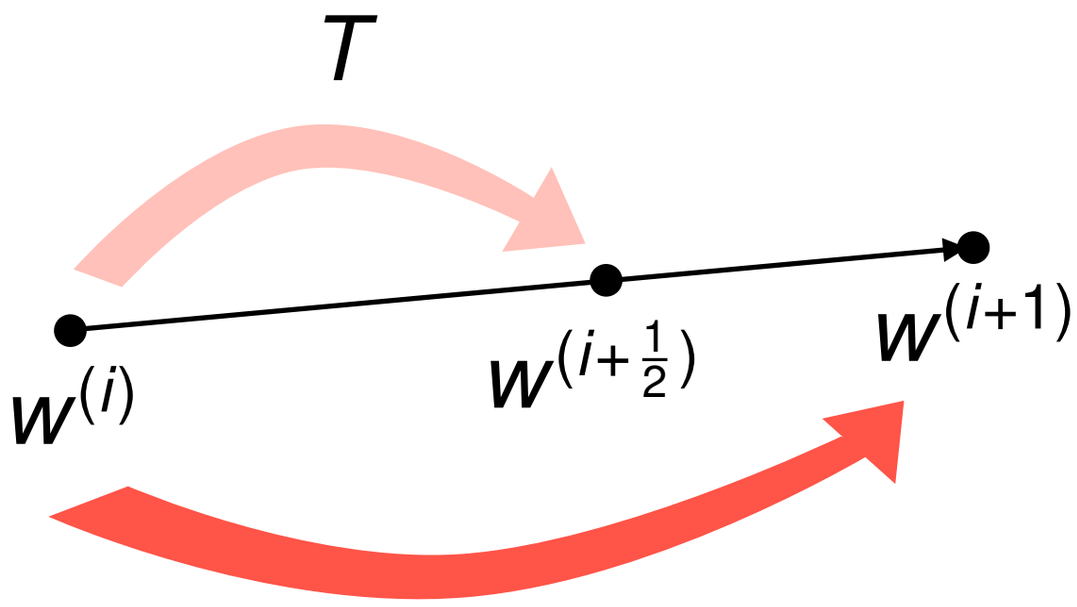


the solution is around here!

Its over-relaxed variant:

$$\begin{cases} w^{(i+\frac{1}{2})} = T(w^{(i)}) \\ w^{(i+1)} = w^{(i)} + \rho(w^{(i+\frac{1}{2})} - w^{(i)}) \end{cases}$$

with $1 < \rho < 1/\alpha$

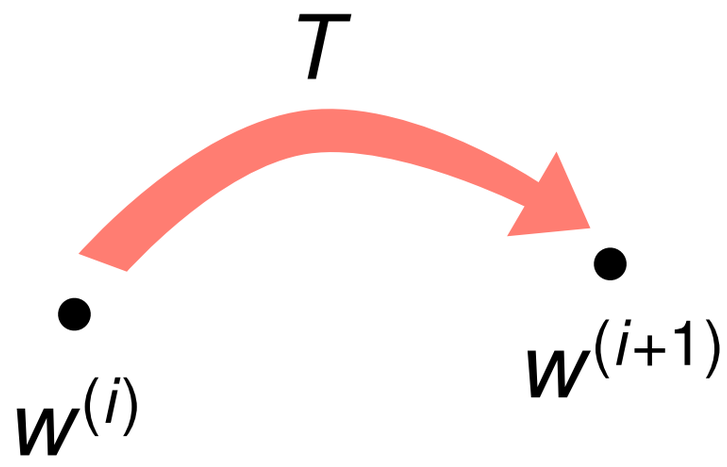


Over-relaxation

An algorithm:

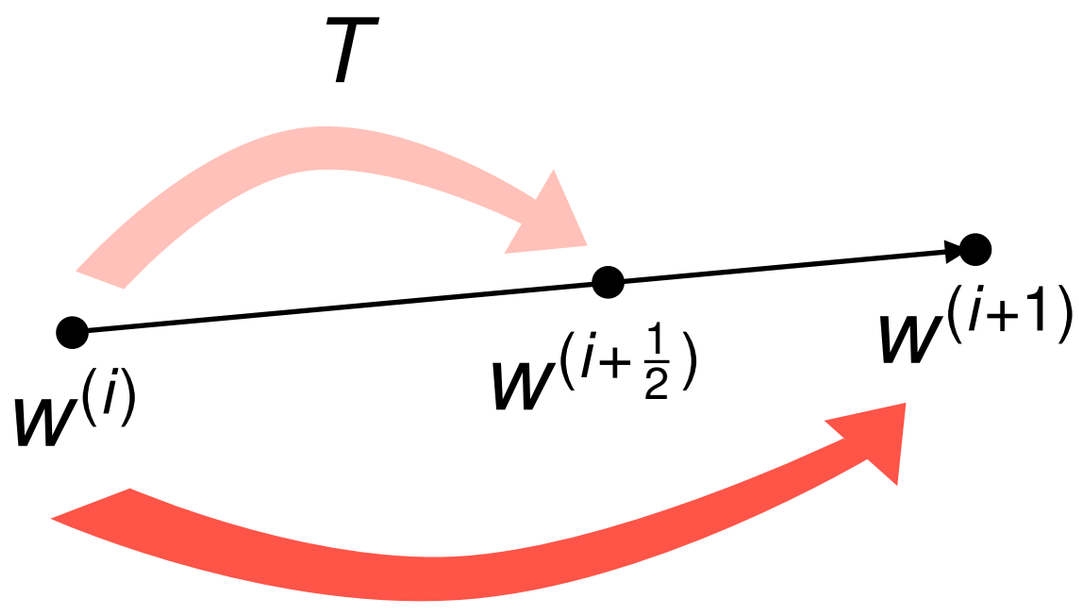
$w^{(i+1)} = T(w^{(i)})$

with T α-averaged



☞ a firmly nonexpansive operator

can be over-relaxed with $1 < \rho < 2$



Goal

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

by iterating $x^{(i+1)} = T(x^{(i)})$
for an α-averaged operator T.

Goal

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \Psi(x) = \sum_{m=1}^{M} g_m(L_m x) \right\}$$

by iterating $x^{(i+1)} = T(x^{(i)})$
for an α-averaged operator T.

☞ How to build T from the

L_m, L_m^*, gradient or proximity operator of g_m ?

Forward-backward splitting

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ h(x) + f(x) \right\}$$

where h is smooth with β-Lipschitz cont. gradient.

☞ the **forward-backward** iteration

$$x^{(i+1)} = \text{prox}_{\tau f}\left(x^{(i)} - \tau \nabla h(x^{(i)})\right)$$

converges to a solution $\tilde{x}$, if $0 < \tau < \frac{2}{\beta}$.

Forward-backward splitting

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ h(x) + f(x) \right\}$$

where h is smooth with β-Lipschitz cont. gradient.

☞ the over-relaxed **forward-backward** iteration

$$\left\lfloor \begin{array}{l} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau \nabla h(x^{(i)})\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \end{array} \right.$$

converges to a solution $\tilde{x}$, if $0 < \tau < \frac{2}{\beta}$, $1 \leq \rho < 2 - \frac{\tau\beta}{2}$.

Forward-backward splitting

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ h(x) + f(x) \right\}$$

where h is smooth with β-Lipschitz cont. gradient.

☞ the over-relaxed **forward-backward** iteration

$$\left\lfloor \begin{array}{l} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau \nabla h(x^{(i)})\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \end{array} \right.$$

converges to a solution $\tilde{x}$, if $0 < \tau < \frac{2}{\beta}$, $1 \leq \rho < 2 - \frac{\tau\beta}{2}$.

For instance, $\tau = \frac{1}{\beta}$, $\rho = 1.49$

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

where f and g have simple prox.
$\rightarrow$ calls to prox_f and prox_g.

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

$$\equiv$$

find $x \in \mathcal{X}$ such that

$$0 \in \partial f(x) + \partial g(x)$$

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

$$\equiv$$

$$\text{find } x \in \mathcal{X} \text{ such that}$$

$$0 \in \partial f(x) + \partial g(x)$$

$$\equiv$$

$$\text{find } (x, u) \in \mathcal{X} \times \mathcal{X} \text{ such that}$$

$$\begin{cases} -u \in \partial f(x) \\ u \in \partial g(x) \end{cases}$$

The ‘dual’ variable u can just be viewed as an auxiliary variable.

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

$$\equiv$$

find $x \in \mathcal{X}$ such that

$$0 \in \partial f(x) + \partial g(x)$$

$$\equiv$$

find $(x, u) \in \mathcal{X} \times \mathcal{X}$ such that

$$\begin{cases} -u \in \partial f(x) \\ x \in (\partial g)^{-1}(u) \end{cases}$$

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

$$\equiv$$

$$\text{find } x \in \mathcal{X} \text{ such that}$$

$$0 \in \partial f(x) + \partial g(x)$$

$$\equiv$$

$$\text{find } (x, u) \in \mathcal{X} \times \mathcal{X} \text{ such that}$$

$$\begin{cases} 0 \in \partial f(x) + u \\ 0 \in (\partial g)^{-1}(u) - x \end{cases}$$

Minimization of 2 functions

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(x) \right\}$$

$$\equiv$$

$$\text{find } x \in \mathcal{X} \text{ such that}$$

$$0 \in \partial f(x) + \partial g(x)$$

$$\equiv$$

$$\text{find } (x, u) \in \mathcal{X} \times \mathcal{X} \text{ such that}$$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x) + u \\ -x + (\partial g)^{-1}(u) \end{pmatrix}$$

Primal-dual monotone inclusion

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \Big\{ f(x) + g(x) \Big\}$$

$$\equiv$$

$$\text{find } x \in \mathcal{X} \text{ such that}$$

$$0 \in \partial f(x) + \partial g(x)$$

$$\equiv$$

$$\text{find } (x, u) \in \mathcal{X} \times \mathcal{X} \text{ such that}$$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x) + u \\ -x + (\partial g)^{-1}(u) \end{pmatrix} \longrightarrow \text{maximally monotone operator}$$

The proximal point algorithm

To solve the monotone inclusion $0 \in M(\tilde{w})$,

the proximal point algorithm is:

$$w^{(i+1)} = (P^{-1}M + \mathrm{Id})^{-1}(w^{(i)})$$

$$\Leftrightarrow \boxed{0 \in M(w^{(i+1)}) + P(w^{(i+1)} - w^{(i)})}$$

Convergence, for any linear, self-adjoint, strongly positive, P.

With w=(x,u), be careful that convergence is on the pair (x,u) and with respect to a distorted <.,P.> norm. So, the variable x alone does not get closer to the solution at every iteration, it can oscillate.

Douglas-Rachford splitting

☞ design an algorithm such that, $\forall i \in \mathbb{N}$,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + (\partial g)^{-1}(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

We need to find this linear operator P to have a primal-dual PPA like on the previous slide

Douglas-Rachford algorithm

☞ design an algorithm such that, $\forall i \in \mathbb{N}$,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & -\mathrm{Id} \\ -\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

☞ Douglas–Rachford iteration:

$$\left\lfloor \begin{array}{ll} x^{(i+1)} & = \mathrm{prox}_{\tau f}\left(x^{(i)} - \tau u^{(i)}\right) \\ u^{(i+1)} & = \mathrm{prox}_{g^*/\tau}\left(u^{(i)} + \frac{1}{\tau}(2x^{(i+1)} - x^{(i)})\right) \end{array} \right.$$

The linear operator in orange is positive, not strongly positive, but the convergence proofs can be extended to this case.

Douglas-Rachford algorithm

☞ design an algorithm such that, $\forall i \in \mathbb{N}$,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & -\mathrm{Id} \\ -\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

☞ Douglas–Rachford iteration:

$$\left\lfloor \begin{array}{lll} x^{(i+1)} & = \mathrm{prox}_{\tau f}\left(s^{(i)}\right) \\ z^{(i+1)} & = \mathrm{prox}_{\tau g}\left(2x^{(i+1)} - s^{(i)}\right) \\ s^{(i+1)} & = s^{(i)} + z^{(i+1)} - x^{(i+1)} \end{array} \right.$$

Douglas-Rachford algorithm

Over-relaxed Douglas–Rachford iteration:

$$\left\lfloor \begin{array}{rcl} x^{(i+1)} & = & \mathrm{prox}_{\tau f}\left(s^{(i)}\right) \\ z^{(i+1)} & = & \mathrm{prox}_{\tau g}\left(2x^{(i+1)} - s^{(i)}\right) \\ s^{(i+1)} & = & s^{(i)} + \rho(z^{(i+1)} - x^{(i+1)}) \end{array} \right.$$

Convergence with $1 \leq \rho < 2$.

☞ take $\rho = 1.9$!

Over-relaxation does not cost anything here!

Douglas-Rachford algorithm, version 2

Switching the update order $\rightarrow$ different iteration:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & +\mathrm{Id} \\ +\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

Douglas-Rachford algorithm, version 2

Switching the update order $\rightarrow$ different iteration:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & +\mathrm{Id} \\ +\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

$\equiv$ switching the roles of the primal and dual variables / problems

Douglas-Rachford algorithm, version 2

Switching the update order $\rightarrow$ different iteration:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & +\mathrm{Id} \\ +\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

$$\left\lfloor \begin{array}{ll} u^{(i+1)} & = \mathrm{prox}_{g^*/\tau}\left(u^{(i)} + \frac{1}{\tau}x^{(i)}\right) \\ x^{(i+1)} & = \mathrm{prox}_{\tau f}\left(x^{(i)} - \tau(2u^{(i+1)} - u^{(i)})\right) \end{array} \right.$$

ADMM

Switching the update order $\rightarrow$ different iteration:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & +\mathrm{Id} \\ +\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

$$\Leftrightarrow \quad \begin{cases} u^{(i+1)} &= \mathrm{prox}_{g^*/\tau}\left(u^{(i)} + \frac{1}{\tau}x^{(i)}\right) \\ x^{(i+1)} &= \mathrm{prox}_{\tau f}\left(x^{(i)} - \tau(2u^{(i+1)} - u^{(i)})\right) \end{cases}$$

$$\equiv \quad \begin{cases} x^{(i)} &= \mathrm{prox}_{\tau f}\left(z^{(i)} - \tau u^{(i)}\right) \\ z^{(i+1)} &= \mathrm{prox}_{\tau g}\left(x^{(i)} + \tau u^{(i)}\right) \\ u^{(i+1)} &= u^{(i)} + \frac{1}{\tau}(x^{(i)} - z^{(i+1)}) \end{cases}$$

This is ADMM

ADMM

Switching the update order $\rightarrow$ different iteration:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + u^{(i+1)} \\ -x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & +\mathrm{Id} \\ +\mathrm{Id} & \tau\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

$$\equiv \left\lfloor \begin{array}{ll} z^{(i+1)} & = \mathrm{prox}_{\tau g}\left(s^{(i)}\right) \\ x^{(i+1)} & = \mathrm{prox}_{\tau f}\left(2z^{(i+1)} - s^{(i)}\right) \\ s^{(i+1)} & = s^{(i)} + x^{(i+1)} - z^{(i+1)} \end{array} \right.$$

This is Douglas-Rachford like in slide 26, just with f and g exchanged.

$\equiv$ switching the roles of f and g in Douglas–Rachford

Generalized Douglas-Rachford

$$\text{Find } (\tilde{x}, \tilde{z}) \in \operatorname*{Arg\,min}_{(x,z)} \Big\{ f(x) + g(z) \Big\} \quad \text{s.t. } Lx + Kz = 0$$

☞ Douglas–Rachford iteration:

$$\left\lfloor \begin{array}{l} x^{(i+1)} \in \arg\min_x f(x) + \frac{1}{2\tau} \| Lx - s^{(i)} \|^2 \\ z^{(i+1)} \in \arg\min_z g(z) + \frac{1}{2\tau} \| Kz + (2Lx^{(i+1)} - s^{(i)}) \|^2 \\ s^{(i+1)} = s^{(i)} - \rho(Lx^{(i+1)} + Kz^{(i+1)}) \end{array} \right.$$

☞ take $\rho = 1.9$!

The ADMM is usually expressed with 1 or 2 linear operators, L and K here. We can write it in this completely equivalent Douglas-Rachford form. The advantage is that we can easily over-relax!

Chambolle-Pock algorithm

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(Lx) \right\}$$

☞ Design an algorithm such that, $\forall i \in \mathbb{N}$,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} \partial f(x^{(i+1)}) + L^* u^{(i+1)} \\ -Lx^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix} + \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} & -L^* \\ -L & \frac{1}{\sigma}\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

Once we understand that Douglas-Rachford is just a primal-dual PPA, there is no price to pay for introducing a linear operator L in the problem. It is the same as the L in the previous slide (with K=-Id) but this time we split. Other said, the Chambolle-Pock algorithm is a preconditioned ADMM/Douglas-Rachford.

Chambolle-Pock algorithm

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(Lx) \right\}$$

☞ Chambolle–Pock algorithm:

$$\left\lfloor \begin{array}{l} x^{(i+1)} = \text{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)}\right) \\ u^{(i+1)} = \text{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+1)} - x^{(i)})\right) \end{array} \right.$$

Chambolle-Pock algorithm

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(Lx) \right\}$$

☞ Chambolle–Pock algorithm:

$$\left\lfloor \begin{array}{l} x^{(i+1)} = \text{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)}\right) \\ u^{(i+1)} = \text{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+1)} - x^{(i)})\right) \end{array} \right.$$

Convergence if $\tau\sigma\|L\|^2 \leq 1$

☞ set $\sigma = 1/(\tau\|L\|^2)$

This condition is proved in my paper JOTA'13

This way, there is only one parameter, tau, to tune, and we recover Douglas-Rachford exactly when L=Id (slide 25)

Chambolle-Pock algorithm

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(Lx) \right\}$$

Over-relaxed Chambolle–Pock algorithm:

$$\left\lfloor \begin{array}{l} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)}\right) \\ u^{(i+\frac{1}{2})} = \text{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+\frac{1}{2})} - x^{(i)})\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \\ u^{(i+1)} = u^{(i)} + \rho(u^{(i+\frac{1}{2})} - u^{(i)}) \end{array} \right.$$

Chambolle-Pock algorithm

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ f(x) + g(Lx) \right\}$$

Over-relaxed Chambolle–Pock algorithm:

$$\left\lfloor \begin{array}{l} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)}\right) \\ u^{(i+\frac{1}{2})} = \text{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+\frac{1}{2})} - x^{(i)})\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \\ u^{(i+1)} = u^{(i)} + \rho(u^{(i+\frac{1}{2})} - u^{(i)}) \end{array} \right.$$

Convergence with $1 \leq \rho < 2$.

☞ take $\rho = 1.9$!

Preconditioned Chambolle-Pock algo.

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \frac{1}{2} \|Ax - y\|^2 + f(x) + g(Lx) \right\}$$

☞ primal-dual PPA:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} A^* A x^{(i+1)} - A^* y + \partial f(x^{(i+1)}) + L^* u^{(i+1)} \\ -L x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix}$$

$$+ \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} - A^* A & -L^* \\ -L & \frac{1}{\sigma}\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

Preconditioned Chambolle-Pock algo.

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \frac{1}{2} \|Ax - y\|^2 + f(x) + g(Lx) \right\}$$

☞ primal-dual PPA:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \begin{pmatrix} A^* A x^{(i+1)} - A^* y + \partial f(x^{(i+1)}) + L^* u^{(i+1)} \\ -L x^{(i+1)} + \partial g^*(u^{(i+1)}) \end{pmatrix}$$

$$+ \begin{pmatrix} \frac{1}{\tau}\mathrm{Id} - A^* A & -L^* \\ -L & \frac{1}{\sigma}\mathrm{Id} \end{pmatrix} \begin{pmatrix} x^{(i+1)} - x^{(i)} \\ u^{(i+1)} - u^{(i)} \end{pmatrix}$$

☞ convergence if $\tau < \dfrac{1}{\|A\|^2}$ and $\sigma = \dfrac{\frac{1}{\tau} - \|A\|^2}{\|L\|^2}$

Preconditioned Chambolle-Pock algo.

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \tfrac{1}{2} \|Ax - y\|^2 + f(x) + g(Lx) \right\}$$

☞ preconditioned Chambolle–Pock algorithm:

$$\left\lfloor \begin{aligned}
x^{(i+\frac{1}{2})} &= \operatorname{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)} - \tau(A^*Ax^{(i)} - A^*y)\right) \\
u^{(i+\frac{1}{2})} &= \operatorname{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+\frac{1}{2})} - x^{(i)})\right) \\
x^{(i+1)} &= x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \\
u^{(i+1)} &= u^{(i)} + \rho(u^{(i+\frac{1}{2})} - u^{(i)})
\end{aligned} \right.$$

This is exactly the algorithm I proposed (JOTA ’13), and exactly to solve this kind of regularized inverse problems. But if you view it as a preconditioned Chambolle-Pock algorithm instead of a primal-dual forward-backward, you can over-relax more!

Preconditioned Chambolle-Pock algo.

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \frac{1}{2}\|Ax - y\|^2 + f(x) + g(Lx) \right\}$$

☞ preconditioned Chambolle–Pock algorithm:

$$\left\lfloor \begin{array}{l} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau L^* u^{(i)} - \tau(A^*Ax^{(i)} - A^*y)\right) \\ u^{(i+\frac{1}{2})} = \text{prox}_{\sigma g^*}\left(u^{(i)} + \sigma L(2x^{(i+\frac{1}{2})} - x^{(i)})\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \\ u^{(i+1)} = u^{(i)} + \rho(u^{(i+\frac{1}{2})} - u^{(i)}) \end{array} \right.$$

Convergence with $1 \leq \rho < 2$.

☞ take $\rho = 1.9$!

With the conditions in my paper you cannot choose 1.9. This is specific to the smooth term in blue being quadratic.

Preconditioned PPA

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \frac{1}{2} \|Ax - y\|^2 + f(x) \right\}$$

$$\left\lfloor \begin{aligned} x^{(i+\frac{1}{2})} &= \text{prox}_{\tau f}\left(x^{(i)} - \tau(A^* A x^{(i)} - A^* y)\right) \\ x^{(i+1)} &= x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \end{aligned} \right.$$

Convergence if $\tau < \dfrac{1}{\|A\|^2}$ and $1 \leq \rho < 2$

Instead of $\tau = \frac{1.99}{\|A\|^2}$, $\rho = 1$,

try $\tau = \frac{0.99}{\|A\|^2}$, $\rho = 1.9$!

Here also, the forward-backward can be viewed as just preconditioning the proximity operator.

Preconditioned PPA

$$\text{Find } \tilde{x} \in \operatorname*{Arg\,min}_{x \in \mathcal{X}} \left\{ \frac{1}{2} \|Ax - y\|^2 + f(x) \right\}$$

$$\begin{cases} x^{(i+\frac{1}{2})} = \text{prox}_{\tau f}\left(x^{(i)} - \tau(A^* A x^{(i)} - A^* y)\right) \\ x^{(i+1)} = x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}) \end{cases}$$

Convergence if $\tau < \dfrac{1}{\|A\|^2}$ and $1 \leq \rho < 2$

Instead of $\tau = \frac{1.99}{\|A\|^2}$, $\rho = 1$,

try $\tau = \frac{0.99}{\|A\|^2}$, $\rho = 1.9$!

there is a continuum between these two regimes (proof unfinished)

Conclusion

- You can and **should** over-relax the algorithms, even more when the smooth function is quadratic.

RELAX!

Conclusion

- You can and **should** over-relax the algorithms, even more when the smooth function is quadratic.

- No time to explain:
  - Product-space trick to generalize $g \circ L$ to $\sum_{m=1}^{M} g_m \circ L_m$
  - Applications to other splitting methods
  - Design an algorithm = play LEGO
  - ...

See the two bonus slides for the Chambolle-Pock algorithm (one of the two forms, the other one in my paper JOTA '13, algorithm 5.2).

The product-space trick

$$\underset{x \in \mathcal{X}}{\text{minimize}} \; \sum_{m=1}^{M} g_m(L_m x)$$

$$\equiv \; \underset{x \in \mathcal{X}}{\text{minimize}} \; \mathbf{g}(\mathbf{L}x)$$

with $\mathbf{g}$: $\mathcal{U}_1 \oplus \cdots \oplus \mathcal{U}_M \to \mathbb{R} \cup \{+\infty\}$

$$(z_1, \ldots, z_M) \mapsto \sum_{m=1}^{M} g_m(z_m)$$

and $\mathbf{L}$: $\mathcal{X} \to \mathcal{U}_1 \oplus \cdots \oplus \mathcal{U}_M$

$$x \mapsto (L_1 x, \ldots, L_M x)$$

Chambolle-Pock algorithm

$$\underset{x \in \mathcal{X}}{\text{minimize}} \left\{ f(x) + \sum_{m=1}^{M} g_m(L_m x) \right\}$$

Over-relaxed Chambolle-Pock algo.

Iterate, for $i \in \mathbb{N}$

$$x^{(i+\frac{1}{2})} := \text{prox}_{\tau f}\left(x^{(i)} - \tau \sum_{m=1}^{M} L_m^* u_m^{(i)}\right),$$

$$x^{(i+1)} := x^{(i)} + \rho(x^{(i+\frac{1}{2})} - x^{(i)}),$$

For $m = 1, \dots, M$,

$$u_m^{(i+\frac{1}{2})} := \text{prox}_{\sigma g_m^*}\left(u_m^{(i)} + \sigma L_m(2x^{(i+\frac{1}{2})} - x^{(i)})\right),$$

$$u_m^{(i+1)} := u_m^{(i)} + \rho(u_m^{(i+\frac{1}{2})} - u_m^{(i)}).$$