ORF522 – Linear and Nonlinear Optimization

16. Operator splitting algorithms

Bartolomeo Stellato — Fall 2024

Recap

Summary of monotone and cocoercive operators

$$\begin{tikzcd}
\begin{array}{c}\textbf{Monotone}\\ (T(x)-T(y))^T(x-y)\geq 0\end{array} & \begin{array}{c}\textbf{Lipschitz}\\ \|F(x)-F(y)\|\leq L\|x-y\|\end{array} \\
\begin{array}{c}\textbf{Strongly monotone}\\ (T(x)-T(y))^T(x-y)\geq \mu\|x-y\|^2\end{array} \arrow[u, "\mu=0"'] \arrow[r, leftrightarrow, "F=T^{-1}"'] & \begin{array}{c}\textbf{Cocoercive}\\ (F(x)-F(y))^T(x-y)\geq \mu\|F(x)-F(y)\|^2\end{array} \arrow[u, "L=1/\mu"'] \arrow[d, leftrightarrow, "G=I-2\mu F"'] \\
 & \begin{array}{c}\textbf{Nonexpansive}\\ \|G(x)-G(y)\|\leq\|x-y\|\end{array}
\end{tikzcd}$$

Strong convexity is the dual of smoothness

f is μ-strongly convex $\iff$ f^* is $(1/\mu)$-smooth

Proof

$$
\begin{aligned}
f \quad \mu\text{-strongly convex} \quad &\iff \quad \partial f \quad \mu\text{-strongly monotone} \\
&\iff \quad (\partial f)^{-1} = \partial f^* \quad \mu\text{-cocoercive} \\
&\iff \quad f^* \quad (1/\mu)\text{-smooth} \qquad \blacksquare
\end{aligned}
$$

Remark: strong convexity and (strong) smoothness are **dual**

Resolvent of subdifferential: proximal operator

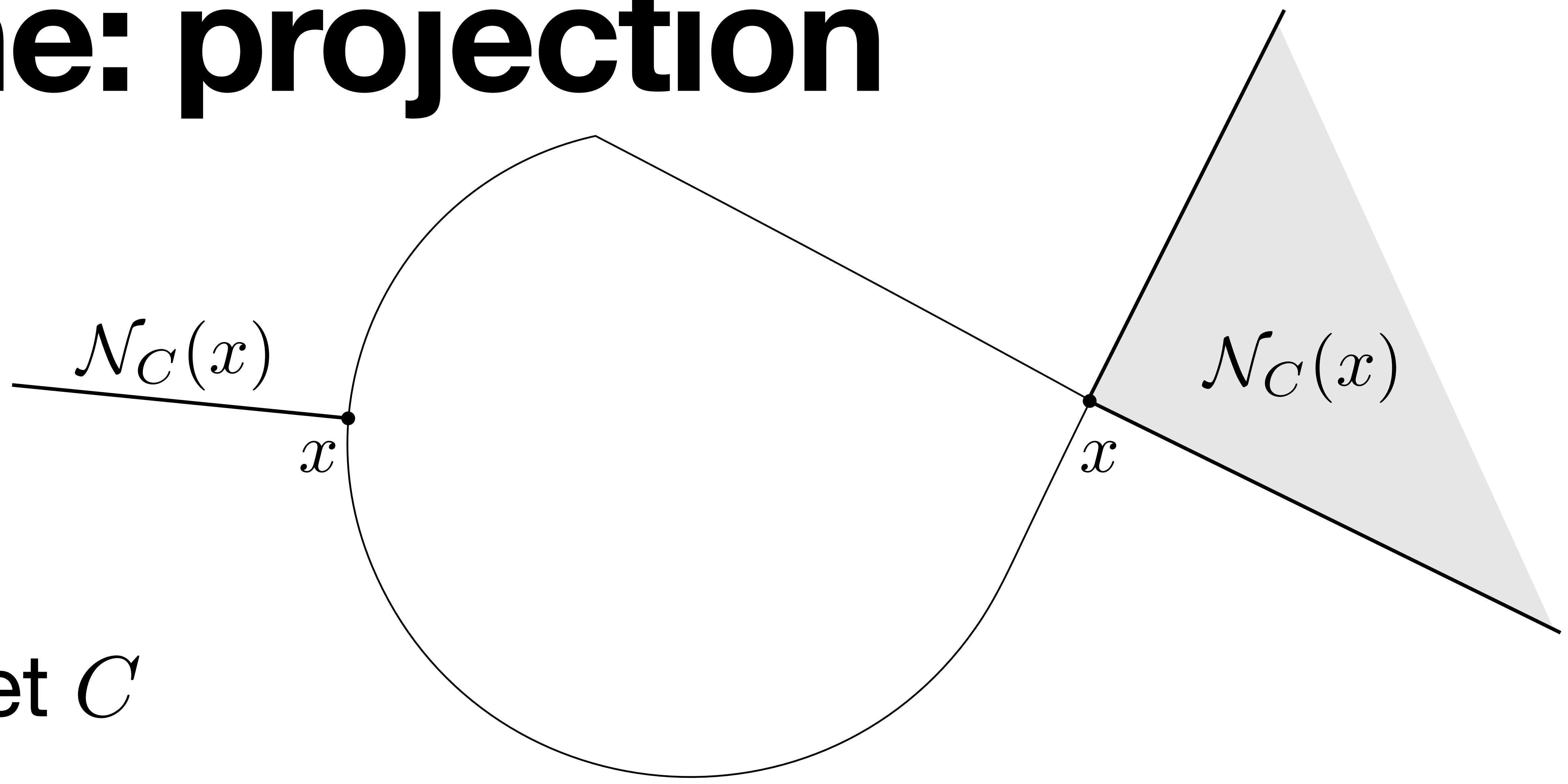
$$\mathbf{prox}_f = R_{\partial f} = (I + \partial f)^{-1}$$

Proof

Let $z = \mathbf{prox}_f(x)$, then

$$\begin{aligned} z &= \underset{u}{\operatorname{argmin}}\, f(u) + \frac{1}{2}\|u - x\|^2 \\ \iff &\; 0 \in \partial f(z) + z - x \qquad \text{(optimality conditions)} \\ \iff &\; x \in (I + \partial f)(z) \\ \iff &\; z = (I + \partial f)^{-1}(x) \qquad \blacksquare \end{aligned}$$

Resolvent of normal cone: projection



$$R_{\partial \mathcal{I}_C} = \Pi_C(x)$$

Proof

Let $f = \mathcal{I}_C$, the indicator function of a convex set C

Recall: $\partial \mathcal{I}_C(x) = \mathcal{N}_C(x)$ **normal cone operator**

$$u = (I + \partial \mathcal{I}_C)^{-1}(x) \quad \iff \quad u = \operatorname*{argmin}_{z} \; \mathcal{I}_C(u) + (1/2)\|z - x\|^2 = \Pi_C(x) \qquad \blacksquare$$

$\mathcal{N}_C$ monotone $\implies$ Π_C nonexpansive

Proof of monotonicity

$$u \in \mathcal{N}_C(x) \;\Rightarrow\; u^T(z - x) \leq 0, \; \forall z \in C \;\Rightarrow\; u^T(y - x) \leq 0$$

$$v \in \mathcal{N}_C(y) \;\Rightarrow\; v^T(z - y) \leq 0, \; \forall z \in C \;\Rightarrow\; v^T(x - y) \leq 0$$

$\longrightarrow$ add to obtain monotonicity $\blacksquare$

Resolvent and Cayley operators

The **resolvent** of operator A is defined as

$$R_A = (I + A)^{-1}$$

The **Cayley (reflection) operator** of A is defined as

$$C_A = 2R_A - I = 2(I + A)^{-1} - I$$

Properties

- If A is maximal monotone, $\mathbf{dom}\, R_A = \mathbf{dom}\, C_A = \mathbf{R}^n$ (Minty's theorem)
- If A is **monotone**, R_A and C_A are **nonexpansive** (thus functions)
- **Zeros** of A are **fixed points** of R_A and C_A

Key result we can solve $0 \in A(x)$ by finding fixed points of C_A or R_A

Today's lecture

[LSMO][PA][PMO]

Operator splitting algorithms

- Requirements to build contraction
- Algorithms
  - Proximal point method
  - Forward-backward splitting
  - Douglas-Rachford splitting

Building contractions

Forward step contractions

Given T L-Lipschitz and μ-strongly monotone, then $I - \gamma T$ converges linearly at rate $\sqrt{1 - 2\gamma\mu + \gamma^2 L^2}$, with optimal step $\gamma = \mu / L^2$.

Proof

strongly monotone

Lipschitz

$$
\begin{aligned}
\|(I - \gamma T)(x) - (I - \gamma T)(y)\|^2 &= \|x - y + \gamma T(x) - \gamma T(y)\|^2 \\
&= \|x - y\|^2 - 2\gamma (T(x) - T(y))^T (x - y) + \gamma^2 \|T(x) - T(y)\|^2 \\
&\leq (1 - 2\gamma\mu + \gamma^2 L^2)\|x - y\|^2
\end{aligned}
$$

■

Remarks

- It applies to **gradient descent** with L-smooth and μ-strongly convex f
- Better rate in gradient descent lecture. We can get it by bounding derivative: $\|D(I - \gamma \nabla^2 f(x))\|_2 \leq \max\{|1 - \gamma L|, |1 - \gamma\mu|\}$. Optimal step $\gamma = 2/(\mu + L)$ and factor $(\mu/L - 1)(\mu/L + 1)$.

Resolvent contractions

If A is μ-strongly monotone, then

$$R_A = (I + A)^{-1}$$

is a contraction with Lipschitz parameter $1/(1+\mu)$

Proof

A μ-strongly monotone $\implies$ $(I + A)$ $(1+\mu)$-strongly monotone

$\implies$ $R_A = (I + A)^{-1}$ $(1+\mu)$-cocoercive

$\implies$ R_A $(1/(1+\mu))$-Lipschitz ■

Cayley contractions

If A is μ-strongly monotone and L-Lipschitz, then

$$C_{\gamma A} = 2R_{\gamma A} - I = 2(I + \gamma A)^{-1} - I$$

is a contraction with factor $\sqrt{1 - 4\gamma \frac{\mu}{(1+\gamma L)^2}}$

Remark need also Lipschitz condition

Proof [Page 20, A Primer on Monotone Operator Methods]

If, in addition, $A = \partial f$ where f is CCP, then $C_{\gamma A}$ converges with a **better** factor $(\sqrt{\mu/L} - 1)/(\sqrt{\mu/L} + 1)$ and optimal step $\gamma = 1/\sqrt{\mu L}$

Proof
[Linear Convergence and Metric Selection for Douglas-Rachford Splitting and ADMM, Giselsson and Boyd]

Requirements for contractions

	Operator A	**Function** f ($A = \partial f$)
Forward step $I - \gamma A$	μ-strongly monotone L-Lipschitz	μ-strongly convex L-smooth
Resolvent $R_A = (I + A)^{-1}$	μ-strongly monotone	μ-strongly convex L-smooth
Cayley $C_A = 2(I + A)^{-1} - I$	μ-strongly monotone L-Lipschitz	μ-strongly convex L-smooth

faster convergence

Key to contractions: strong monotonicity/convexity

Proximal point method

Proximal point method

Resolvent iterations

$$x^{k+1} = R_A(x^k) = (I + A)^{-1}(x^k)$$

Many traditional algorithms are **proximal point method** with a specific A

If $A = \partial t f$, we get **proximal minimization algorithm**

$$x^{k+1} = \mathbf{prox}_{tf}(x^k) = \operatorname*{argmin}_z \left(tf(z) + \frac{1}{2}\|z - x^k\|_2^2 \right)$$

Proximal minimization properties

- R_A is $1/2$ averaged: $R_A = (1/2)I + (1/2)C_A \quad \Longrightarrow \quad R_{t\partial f}$ converges $\forall t$
- $\mathbf{fix}\, R_{\partial t f}$ are zeros of ∂f: **optimal solutions**
- If f μ-strongly convex, $R_{\partial t f}$ contraction: **linear convergence**
- Useful only if you can evaluate $\mathbf{prox}_{tf}$ efficiently

Method of multipliers

minimize $\quad f(x)$

subject to $\quad Ax = b$

Lagrangian

$$L(x, y) = f(x) + y^T(Ax - b)$$

Dual problem

maximize $\quad g(y) = -(f^*(-A^T y) + y^T b)$

Multiplier to residual map operator

$T(y) = b - Ax$, where $x = \operatorname{argmin}_z L(z, y)$ $\longrightarrow$ $T(y) = \partial(-g)$

Therefore, $\quad \partial(-g)(y) = b - Ax, \qquad 0 \in \partial f(x) + A^T y$

Solve the dual with proximal point method

$$y^{k+1} = R_{t\partial(-g)}(y^k)$$

Method of multipliers

Derivation

Solve the dual with proximal point method

$$y^{k+1} = R_{t\partial(-g)}(y^k)$$

where $\quad \partial(-g)(y) = b - Ax, \qquad$ with x such that $\quad 0 \in \partial f(x) + A^T y$

Resolvent reformulation

$$\begin{aligned} y^{k+1} = R_{t\partial(-g)}(y^k) &\iff y^{k+1} + t\partial(-g)(y^{k+1}) = y^k \\ &\iff y^{k+1} + t(b - Ax^{k+1}) = y^k, \quad \text{with} \quad 0 \in \partial f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

x^{k+1} minimizes the **augmented Lagrangian** $L_t(x, y^{k+1})$

$$0 \in \partial f(x^{k+1}) + A^T(y^k + t(Ax^{k+1} - b))$$

$$\implies \quad x^{k+1} \in \underset{x}{\operatorname{argmin}}\, f(x) + (y^k)^T(Ax - b) + (t/2)\|Ax - b\|^2 = \underset{x}{\operatorname{argmin}}\, L_t(x, y^k)$$

Method of multipliers (augmented Lagrangian method)

Primal

minimize $f(x)$

subject to $Ax = b$

Dual

maximize $g(y) = -(f^*(-A^Ty) + y^Tb)$

Iterates

$$y^{k+1} = R_{t\partial(-g)}(y^k)$$

$$\downarrow$$

$$x^{k+1} \in \underset{x}{\operatorname{argmin}}\, L_t(x, y^k)$$

$$y^{k+1} = y^k + t(Ax^{k+1} - b)$$

Properties

- Always converges with CCP f for any $t > 0$
- If f L-smooth

  f^* and g are μ-strongly convex

  $R_{\partial(-g)}$ is a contraction: **linear convergence**
- If f strictly convex ($>$), then argmin has a unique solution ($\in$ becomes $=$)
- Useful when f L-smooth and A sparse

Method of multipliers dual feasibility

minimize	$f(x)$
subject to	$Ax = b$

$$
\begin{aligned}
x^{k+1} &\in \operatorname*{argmin}_{x} L_t(x, y^k) \\
y^{k+1} &= y^k + t(Ax^{k+1} - b)
\end{aligned}
$$

Optimality conditions (primal and dual feasibility)

$$Ax - b, \qquad \partial f(x) + A^T y \ni 0$$

From x^{k+1} update

$$
\begin{aligned}
0 &\in \partial f(x^{k+1}) + A^T y^k + tA^T(Ax^{k+1} - b) \\
&= \partial f(x^{k+1}) + A^T y^{k+1}
\end{aligned}
$$

$\longrightarrow$ (x^{k+1}, y^{k+1})

dual feasible

primal feasible in the limit, i.e. $Ax^k - b \to 0$

Forward-backward splitting

Operator splitting

Main idea

We would like to solve

$$0 \in F(x), \qquad F \text{ maximal monotone}$$

Split the operator

$$F = A + B, \qquad A \text{ and } B \text{ are maximal monotone}$$

Solve by evaluating

$$R_A = (I + A)^{-1} \qquad\qquad C_A = 2R_A - I$$

or

$$R_B = (I + B)^{-1} \qquad\qquad C_B = 2R_B - I$$

Useful when R_A and R_B are cheaper than R_F

Forward-backward splitting

Goal

Find x such that $0 \in A(x) + B(x)$

Rewrite optimality condition

$$\begin{aligned} 0 \in (A+B)(x) &\iff 0 \in t(A+B)(x) \\ &\iff 0 \in (I + tB)(x) - (I - tA)(x) \\ &\iff (I + tB)(x) \ni (I - tA)(x) \\ &\iff x = (I + tB)^{-1}(I - tA)(x) \\ &\iff x = R_{tB}(I - tA)(x) \end{aligned}$$

Iterations

$$x^{k+1} = R_{tB}(I - tA)(x^k)$$

Forward-backward splitting

Properties

Iterations

$$x^{k+1} = \underbrace{R_{tB}}_{\text{resolvent}}\underbrace{(I - tA}_{\text{forward step}})(x^k)$$

Properties

- R_{tB} is $1/2$ averaged
- If A is μ-cocoercive then $I - 2\mu A$ is nonexpansive
  $\Rightarrow I - tA$ is averaged for $t \in (0, 2\mu)$
- Therefore forward-backward splitting converges
- If either A or B is strongly monotone, then **linear convergence**

Proximal gradient descent as forward-backward splitting

minimize $\quad f(x) + g(x)$

f is L-smooth
g is nonsmooth but proxable

Therefore, ∇f is $(1/L)$-cocoercive and ∂g maximal monotone

Proximal gradient descent

$$\begin{aligned} x^{k+1} &= R_{t\partial g}(I - t\nabla f)(x^k) \\ &= \mathbf{prox}_{tg}(x^k - t\nabla f(x^k)) \end{aligned}$$

Remarks

- Converges for $t \in (0, 2/L)$
- If either f or g strongly convex **linear convergence**
- If $g = \mathcal{I}_C$, then it's projected gradient descent

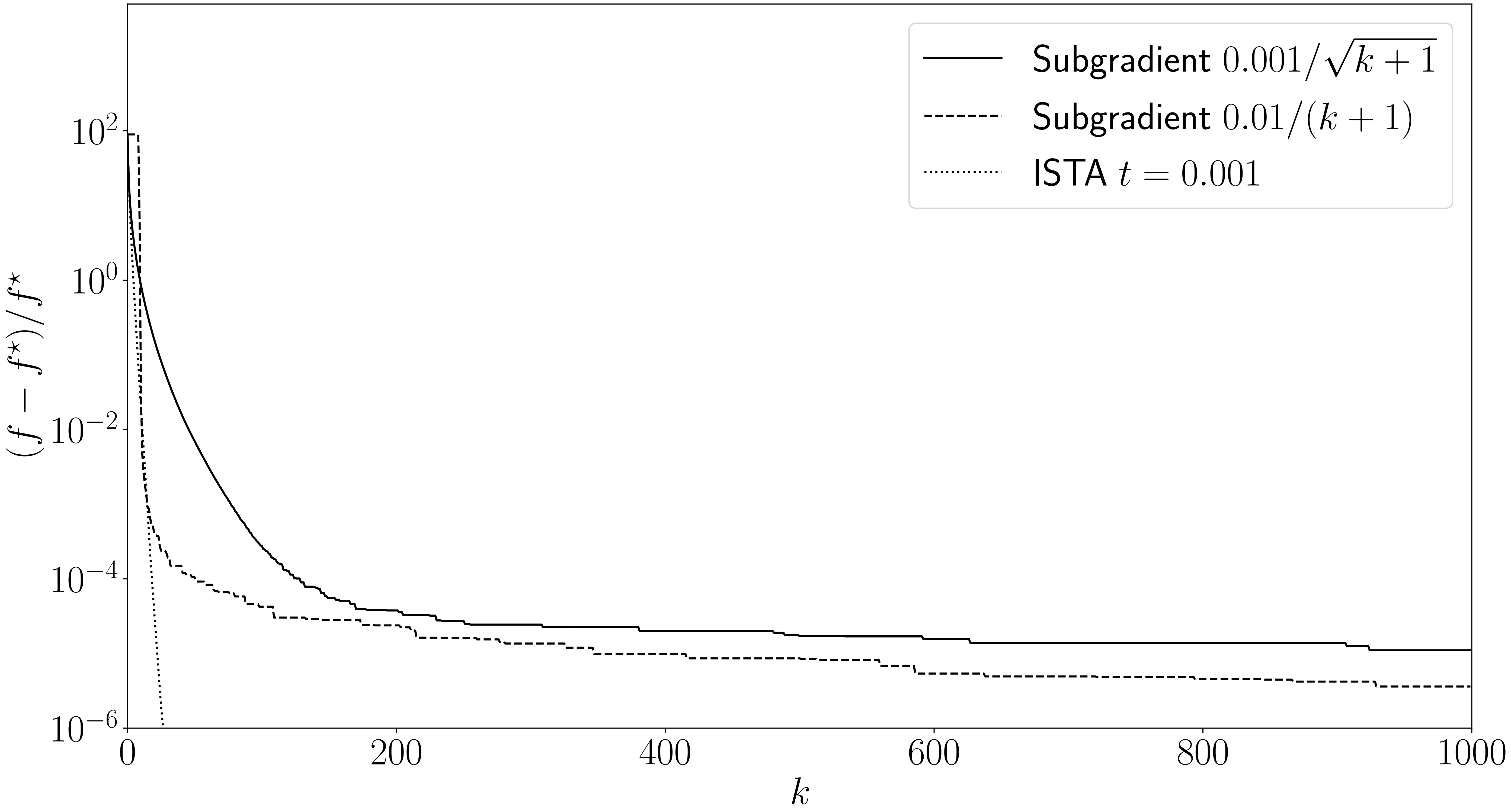
Example: Lasso with linear convergence

Iterative Soft Thresholding Algorithm (ISTA)

minimize $\underbrace{(1/2)\|Ax - b\|_2^2}_{f(x)} + \underbrace{\lambda\|x\|_1}_{g(x)}$

Proximal gradient descent

$$x^{k+1} = S_{\lambda t}\left(x^k - tA^T(Ax^k - b)\right)$$



Example

randomly generated $A \in \mathbf{R}^{500 \times 300}$

$\Rightarrow \quad \nabla^2 f = A^T A \succ 0$

$\Rightarrow \quad f$ strongly convex

linear convergence

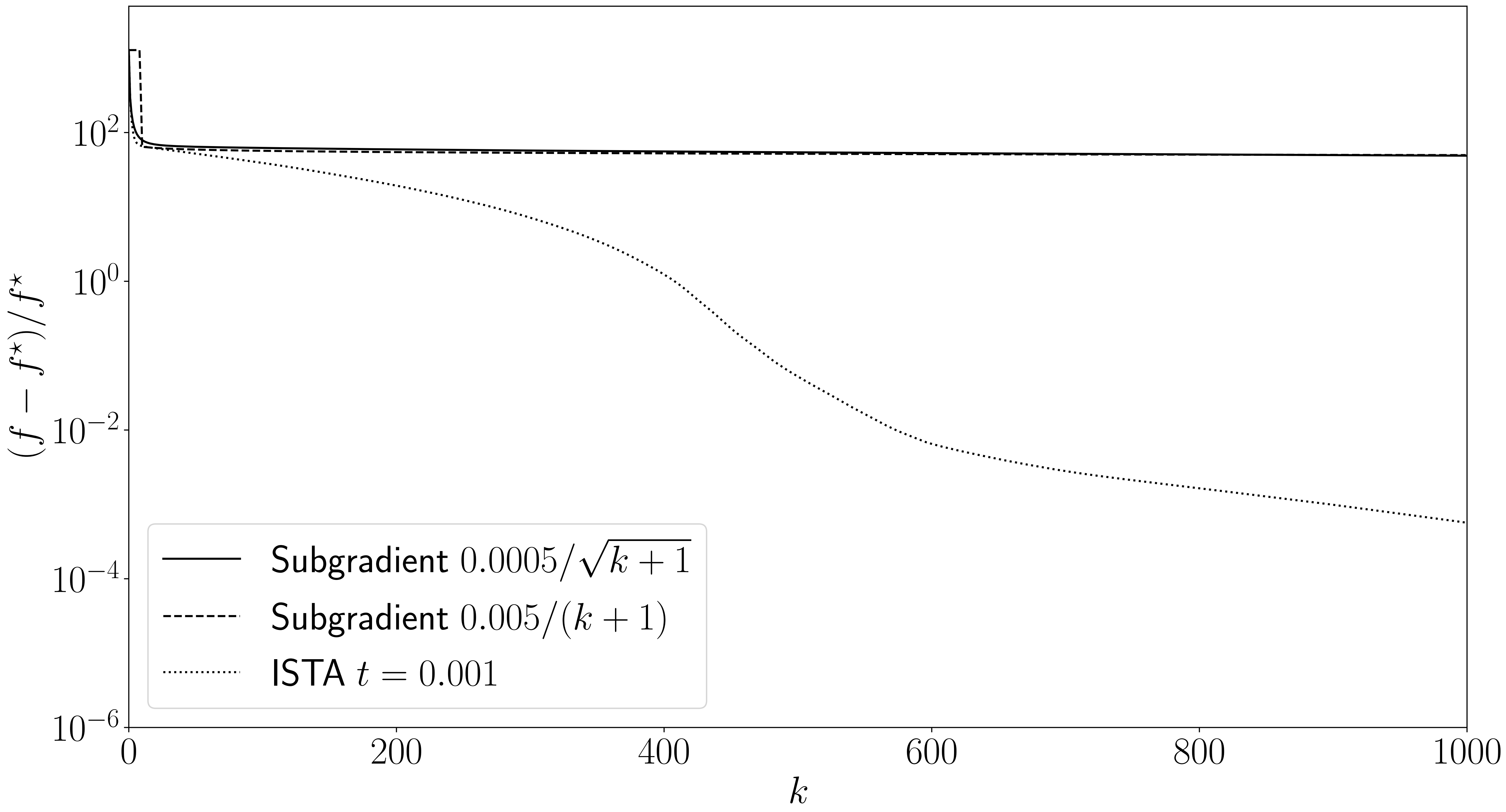
Example: Lasso without linear convergence

Iterative Soft Thresholding Algorithm (ISTA)

$$\text{minimize} \quad \underbrace{(1/2)\|Ax - b\|_2^2}_{f(x)} + \underbrace{\lambda\|x\|_1}_{g(x)}$$

Proximal gradient descent

$$x^{k+1} = S_{\lambda t}\left(x^k - tA^T(Ax^k - b)\right)$$



Example

randomly generated $A \in \mathbf{R}^{300 \times 500}$

$\Rightarrow \quad \nabla^2 f = A^T A \succeq 0$

$\Rightarrow \quad f$ not strongly convex

sublinear convergence

Douglas-Rachford splitting

Operator splitting

Main idea

We would like to solve

$$0 \in F(x), \qquad F \text{ maximal monotone}$$

Split the operator

$$F = A + B, \qquad A \text{ and } B \text{ are maximal monotone}$$

Solve by evaluating

$$R_A = (I + A)^{-1} \qquad C_A = 2R_A - I$$

or

$$R_B = (I + B)^{-1} \qquad C_B = 2R_B - I$$

Useful when R_A and R_B are cheaper than R_F

Splitting Cayley iterations

Key result

$$0 \in A(x) + B(x) \quad \Longleftrightarrow \quad C_A C_B(z) = z, \quad x = R_B(z)$$

Goal

Apply C_A and C_B sequentially instead of computing R_{A+B} directly

Splitting Cayley iterations

Proof of key result

$$\left.\begin{aligned} C_AC_B(z) &= z \\ x &= R_B(z) \end{aligned}\right. \longrightarrow \left.\begin{aligned} x &= R_B(z) \\ \tilde{z} &= 2x - z \\ \tilde{x} &= R_A(\tilde{z}) \\ z &= 2\tilde{x} - \tilde{z} \end{aligned}\right. \xrightarrow{\text{combine}} \tilde{x} = x$$

last equation

$$2x = z + \tilde{z}$$

Since $x = R_B(z)$, we have $z \in x + B(x)$

Since $\tilde{x} = R_A(\tilde{z})$, we have $\tilde{z} \in \tilde{x} + A(\tilde{x}) = x + A(x)$

By adding them, we obtain $z + \tilde{z} \in 2x + A(x) + B(x)$

Therefore, $0 \in A(x) + B(x)$ ■

Note the arguments also holds the other way but we do not need it

Peaceman-Rachford and Douglas Rachford splitting

Peaceman-Rachford splitting

$$w^{k+1} = C_A C_B(w^k)$$

It does not converge in general (product of nonexpansive).
Need C_A or C_B to be a contraction

Douglas-Rachford splitting (averaged iterations)

$$w^{k+1} = (1/2)(I + C_A C_B)(w^k)$$

- **Always converges** when $0 \in A(x) + B(x)$ has a solution
- If A or B strongly monotone and Lipschitz, then $C_A C_B$ is a contraction: **linear convergence**
- This method traces back to the 1950s

Douglas-Rachford splitting

$$w^{k+1} = (1/2)(I + C_A C_B)(w^k) \longrightarrow$$

Iterations

$$\begin{aligned} z^{k+1} &= R_B(w^k) \\ \tilde{w}^{k+1} &= 2z^{k+1} - w^k \\ x^{k+1} &= R_A(\tilde{w}^{k+1}) \\ w^{k+1} &= w^k + x^{k+1} - z^{k+1} \end{aligned}$$

Last update (averaging) follows from:

$$\begin{aligned} w^{k+1} &= (1/2)w^k + (1/2)(2x^{k+1} - \tilde{w}^{k+1}) \\ &= (1/2)w^k + x^{k+1} - (1/2)(2z^{k+1} - w^k) \\ &= w^k + x^{k+1} - z^{k+1} \end{aligned}$$

Simplified iterations of Douglas-Rachford splitting

DR iterations

(simplify two inner steps)

$$z^{k+1} = R_B(w^k)$$

$$w^{k+1} = w^k + R_A(2z^{k+1} - w^k) - z^{k+1}$$

1 Swap iterations and counter

$$w^{k+1} = w^k + R_A(2z^k - w^k) - z^k$$

$$z^{k+1} = R_B(w^{k+1})$$

2 Introduce x^{k+1}

$$x^{k+1} = R_A(2z^k - w^k)$$

$$w^{k+1} = w^k + x^{k+1} - z^k$$

$$z^{k+1} = R_B(w^{k+1})$$

3 Update w^{k+1} at the end

$$x^{k+1} = R_A(2z^k - w^k)$$

$$z^{k+1} = R_B(w^k + x^{k+1} - z^k)$$

$$w^{k+1} = w^k + x^{k+1} - z^k$$

4 Define $u^k = w^k - z^k$

$$x^{k+1} = R_A(z^k - u^k)$$

$$z^{k+1} = R_B(x^{k+1} + u^k)$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Douglas-Rachford splitting

Simplified iterations

$$x^{k+1} = R_A(z^k - u^k)$$
$$z^{k+1} = R_B(x^{k+1} + u^k)$$
$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Residual: $x^{k+1} - z^{k+1}$

running sum of residuals
u^k

Interpretation as integral control

Remarks

- *many* ways to rearrange the D-R algorithm
- Equivalent to many other algorithms (proximal point, Spingarn's partial inverses, Bregman iterative methods, etc.)
- Need very little to converge: A, B maximal monotone
- Splitting A and B, we can uncouple and evaluate R_A and R_B separately

Operator splitting algorithms

Today, we learned to:

- **Construct** contractions and **understand** their requirements
- **Apply** the proximal point method to the “multiplier to residual” mapping obtaining the Method of Multipliers (Augmented Lagrangian)
- **Derive** proximal gradient from forward-backward splitting
- **Split** operators to obtain simpler averaged iterations with Douglas-Rachford splitting

Next lecture

- Alternating Direction Method of Multipliers