Constructive approaches to optimal first-order methods for (strongly) convex minimization

Adrien Taylor

Optimization without borders, Soshi – July 2021

François Glineur
(UCLouvain)

Julien Hendrickx
(UCLouvain)

Etienne de Klerk
(Tilburg & Delft)

Ernest Ryu
(UCLA)

Francis Bach
(Inria/ENS)

Jérôme Bolte
(TSE)

A. d'Aspremont
(CNRS/ENS)

Yoel Drori
(Google)

Mathieu Barré
(Inria/ENS)

A-R. Dragomir
(ENS/TSE)

B. Van Scoy
(W-Madison)

L. Lessard
(W-Madison)

C. Bergeling
(Lund)

P. Giselsson
(Lund)

Yoel Drori

Yoel Drori

Presentation based on joint works:

- ◇ "An optimal gradient method for smooth strongly convex minimization." (2021)
- ◇ "On the oracle complexity of smooth strongly convex minimization." (2021)

François Glineur

Julien Hendrickx

François Glineur

Julien Hendrickx

Introduction based on joint works:

- ◇ "Smooth strongly convex interpolation and exact worst-case performance of first-order methods." (2017)
- ◇ "Exact worst-case performance of first-order methods for composite convex optimization." (2017)

Many (very) related works; much more careful bibliographical treatment in papers.

Many (very) related works; much more careful bibliographical treatment in papers.

- B. Polyak. "Introduction to optimization" (1964)
- Y. Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$." (1983)
- A. Nemirovsky, and B. Polyak. "Iterative methods for solving linear ill-posed problems under precise information." (1984)
- A. Nemirovsky. "Information-based complexity of linear operator equations." (1992)
- A. Nemirovsky. "Information-based complexity of convex programming." (lecture notes, 1995)
- Y. Nesterov. "Introductory Lectures on Convex Optimization." (2003/2018)
- Y. Drori, and M. Teboulle. "Performance of first-order methods for smooth convex minimization: a novel approach." (2014)
- D. Kim, and F. Fessler. "Optimized first-order methods for smooth convex minimization." (2016)
- L. Lessard, B. Recht, and A. Packard. "Analysis and design of optimization algorithms via integral quadratic constraints." (2016)
- B. Van Scoy, R. Freeman, K. Lynch. "The fastest known globally convergent first-order method for minimizing strongly convex functions." (2017)
- D. Kim, and F. Fessler. "Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions." (2021)

Take-home messages

Worst-cases are solutions to optimization problems

Acceleration/“optimal” methods by optimizing worst-cases

On worst-case analyses

Step-sizes optimization

Constructing lower bounds

Software

Concluding remarks

Analysis of a first-order method

Say we aim to solve

$$\min_{x \in \mathbb{R}^{\boldsymbol{d}}} f(x)$$

where f is μ-strongly convex and L-smooth.

Analysis of a first-order method

Say we aim to solve

$$\min_{x \in \mathbb{R}^{\boldsymbol{d}}} f(x)$$

where f is μ-strongly convex and L-smooth.

(Given first-order method) We decide to use:

$$\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned} \tag{FOM}$$

for some coefficients $\{h_{i,j}\}$.

Analysis of a first-order method

(Given first-order method) We decide to use:

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
\tag{FOM}
$$

for some coefficients $\{h_{i,j}\}$.

Analysis of a first-order method

(Given first-order method) We decide to use:

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
\tag{FOM}
$$

for some coefficients $\{h_{i,j}\}$.

Question 1: what *a priori* guarantees after N iterations?

Analysis of a first-order method

(Given first-order method) We decide to use:

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
\tag{FOM}
$$

for some coefficients $\{h_{i,j}\}$.

Question 1: what *a priori* guarantees after N iterations?

Examples: how small should $f(x_N) - f(x_\star)$, $\|f'(x_N)\|$, $\|x_N - x_\star\|$ be?

Analysis of a first-order method

(Given first-order method) We decide to use:

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
\tag{FOM}
$$

for some coefficients $\{h_{i,j}\}$.

Question 1: what *a priori* guarantees after N iterations?

Examples: how small should $f(x_N) - f(x_\star)$, $\|f'(x_N)\|$, $\|x_N - x_\star\|$ be?

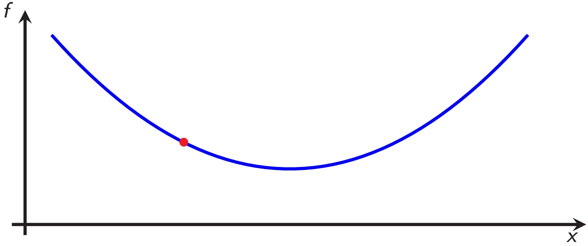
Question 2: how to choose the step-sizes $\{h_{i,j}\}$?

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:

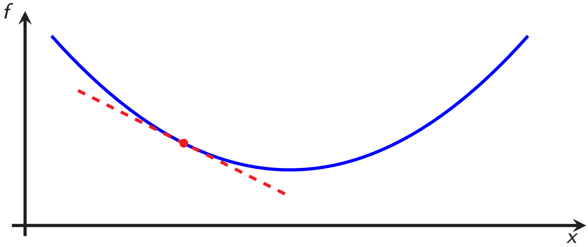
Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



Smooth strongly convex functions

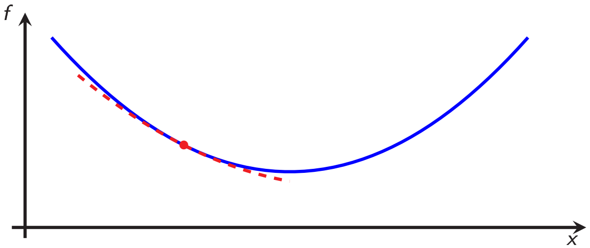
Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:

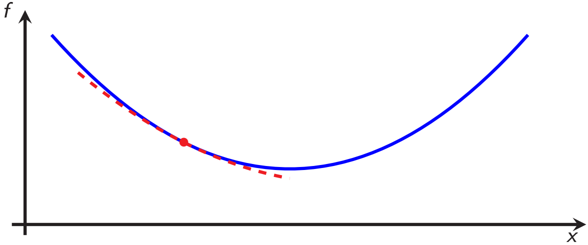


(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2$,

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



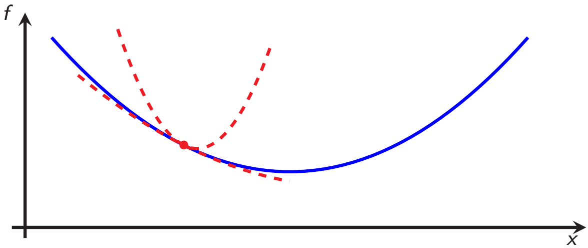
(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2$,

(2) (L-smoothness) $f(x) \leqslant f(y) + \langle f'(y), x - y \rangle + \frac{L}{2} \|x - y\|^2$.

Smooth strongly convex functions

Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



(1) (Convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle f'(y), x - y \rangle + \frac{\mu}{2}\|x - y\|^2$,

(2) (L-smoothness) $f(x) \leqslant f(y) + \langle f'(y), x - y \rangle + \frac{L}{2}\|x - y\|^2$.

Convergence rate of a gradient step

Convergence rate of a gradient step

Toy example: What is the smallest τ such that:

$$\|x_1 - x_\star\|^2 \leqslant \tau \|x_0 - x_\star\|^2,$$

for all

- L-smooth and μ-strongly convex function f (notation $f \in \mathcal{F}_{\mu,\mathrm{L}}$),
- x_0, and x_1 generated by gradient step $x_1 = x_0 - h_{1,0} f'(x_0)$,
- $x_\star = \underset{x}{\operatorname{argmin}}\, f(x)$?

Convergence rate of a gradient step

Toy example: What is the smallest τ such that:

$$\|x_1 - x_\star\|^2 \leqslant \tau \|x_0 - x_\star\|^2,$$

for all

- L-smooth and μ-strongly convex function f (notation $f \in \mathcal{F}_{\mu,\mathrm{L}}$),
- x_0, and x_1 generated by gradient step $x_1 = x_0 - h_{1,0} f'(x_0)$,
- $x_\star = \operatorname{argmin}_x f(x)$?

Final goal: optimize τ (as a function of the step-size, $h_{1,0}$)

Convergence rate of a gradient step

Toy example: What is the smallest τ such that:

$$\|x_1 - x_\star\|^2 \leqslant \tau \|x_0 - x_\star\|^2,$$

for all

- L-smooth and μ-strongly convex function f (notation $f \in \mathcal{F}_{\mu,\mathrm{L}}$),
- x_0, and x_1 generated by gradient step $x_1 = x_0 - h_{1,0} f'(x_0)$,
- $x_\star = \underset{x}{\operatorname{argmin}}\, f(x)$?

Final goal: optimize τ (as a function of the step-size, $h_{1,0}$)

First: let's compute τ!

Convergence rate of a gradient step

Toy example: What is the smallest τ such that:

$$\|x_1 - x_\star\|^2 \leqslant \tau \|x_0 - x_\star\|^2,$$

for all

- L-smooth and μ-strongly convex function f (notation $f \in \mathcal{F}_{\mu,\mathrm{L}}$),
- x_0, and x_1 generated by gradient step $x_1 = x_0 - h_{1,0} f'(x_0)$,
- $x_\star = \operatorname*{argmin}_x f(x)$?

Final goal: optimize τ (as a function of the step-size, $h_{1,0}$)

First: let's compute τ!

$$\begin{aligned} \tau = \max_{f, x_0, x_1, x_\star} \; & \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} & \\ \text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\ & x_1 = x_0 - h_{1,0} f'(x_0) & \text{Algorithm} \\ & f'(x_\star) = 0 & \text{Optimality of } x_\star \end{aligned}$$

Convergence rate of a gradient step

Toy example: What is the smallest τ such that:

$$\|x_1 - x_\star\|^2 \leqslant \tau \|x_0 - x_\star\|^2,$$

for all

- L-smooth and μ-strongly convex function f (notation $f \in \mathcal{F}_{\mu,\mathrm{L}}$),
- x_0, and x_1 generated by gradient step $x_1 = x_0 - h_{1,0} f'(x_0)$,
- $x_\star = \underset{x}{\operatorname{argmin}}\, f(x)$?

Final goal: optimize τ (as a function of the step-size, $h_{1,0}$)

First: let's compute τ!

$$\begin{aligned}
\tau = \max_{f, x_0, x_1, x_\star} \quad & \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} & \\
\text{s.t.} \quad & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - h_{1,0} f'(x_0) & \text{Algorithm} \\
& f'(x_\star) = 0 & \text{Optimality of } x_\star
\end{aligned}$$

Variables: f, x_0, x_1, $x_\star$; parameters: μ, L, $h_{1,0}$.

Sampled version

Sampled version

- Performance estimation problem:

$$\begin{aligned}
\max_{f, x_0, x_1, x_\star} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\
\text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex,} \\
& x_1 = x_0 - h_{1,0} f'(x_0) \\
& f'(x_\star) = 0.
\end{aligned}$$

Sampled version

- Performance estimation problem:

$$\max_{f, x_0, x_1, x_\star} \quad \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2}$$
$$\text{subject to} \quad f \text{ is } L\text{-smooth and } \mu\text{-strongly convex,}$$
$$x_1 = x_0 - h_{1,0} f'(x_0)$$
$$f'(x_\star) = 0.$$

- Variables: f, x_0, x_1, $x_\star$.

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1, x_\star} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex,} \\ & x_1 = x_0 - h_{1,0} f'(x_0) \\ & f'(x_\star) = 0. \end{aligned}$$

- Variables: f, x_0, x_1, $x_\star$.
- Sampled version: f is only used at x_0 and $x_\star$ (no need to sample other points)

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1, x_\star} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex,} \\ & x_1 = x_0 - h_{1,0} f'(x_0) \\ & f'(x_\star) = 0. \end{aligned}$$

- Variables: f, x_0, x_1, $x_\star$.
- Sampled version: f is only used at x_0 and $x_\star$ (no need to sample other points)

$$\begin{aligned} \max_{\substack{x_0, x_1, x_\star \\ g_0, g_\star \\ f_0, f_\star}} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\ \text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, \star \\ g_i = f'(x_i) & i = 0, \star \end{cases} \\ & x_1 = x_0 - h_{1,0} g_0 \\ & g_\star = 0. \end{aligned}$$

Sampled version

- Performance estimation problem:

$$\begin{aligned} \max_{f, x_0, x_1, x_\star} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\ \text{subject to} \quad & f \text{ is } L\text{-smooth and } \mu\text{-strongly convex,} \\ & x_1 = x_0 - h_{1,0} f'(x_0) \\ & f'(x_\star) = 0. \end{aligned}$$

- Variables: f, x_0, x_1, $x_\star$.
- Sampled version: f is only used at x_0 and $x_\star$ (no need to sample other points)

$$\begin{aligned} \max_{\substack{x_0, x_1, x_\star \\ g_0, g_\star \\ f_0, f_\star}} \quad & \frac{\|x_1 - x_0\|^2}{\|x_0 - x_\star\|^2} \\ \text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, \star \\ g_i = f'(x_i) & i = 0, \star \end{cases} \\ & x_1 = x_0 - h_{1,0} g_0 \\ & g_\star = 0. \end{aligned}$$

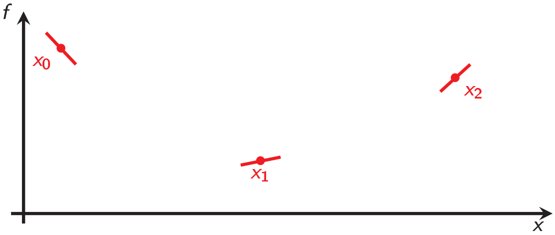
- Variables: x_0, x_1, $x_\star$, g_0, $g_\star$, f_0, $f_\star$.

Smooth strongly convex interpolation (or extension)

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.

Smooth strongly convex interpolation (or extension)

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i \in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.

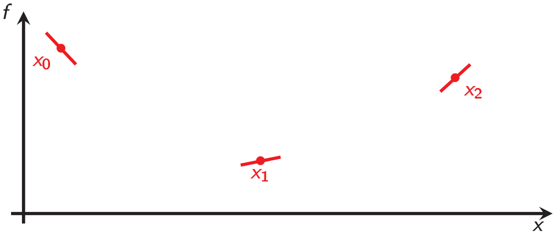


? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

Smooth strongly convex interpolation (or extension)

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.



? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

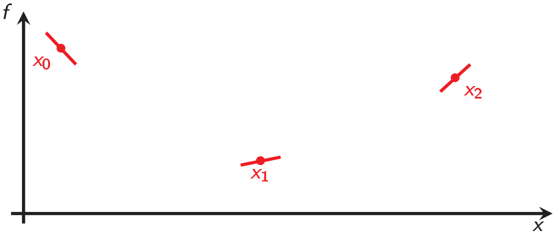
$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

- Necessary and sufficient condition: $\forall i, j \in S$

$$f_i \geqslant f_j + \langle g_j, x_i - x_j \rangle + \frac{1}{2L} \|g_i - g_j\|^2 + \frac{\mu}{2(1-\mu/L)} \|x_i - x_j - \frac{1}{L}(g_i - g_j)\|^2.$$

Smooth strongly convex interpolation (or extension)

Consider an index set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, (sub)gradients g_i and function values f_i.



? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

- Necessary and sufficient condition: $\forall i, j \in S$

$$f_i \geqslant f_j + \langle g_j, x_i - x_j \rangle + \tfrac{1}{2L}\left\|g_i - g_j\right\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_i - x_j - \tfrac{1}{L}(g_i - g_j)\right\|^2.$$

- Simpler example: pick $\mu = 0$ and $L = \infty$ (just convexity):

$$f_i \geqslant f_j + \langle g_j, x_i - x_j \rangle.$$

Replace constraints

Replace constraints

- Interpolation conditions allow removing red constraints

$$
\begin{aligned}
\max_{\substack{x_0, x_1, x_\star \\ g_0, g_\star \\ f_0, f_\star}} \quad & \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} \\
\text{subject to} \quad & \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, \star \\ g_i = f'(x_i) & i = 0, \star \end{cases} \\
& x_1 = x_0 - h_{1,0} g_0 \\
& g_\star = 0,
\end{aligned}
$$

Replace constraints

- Interpolation conditions allow removing red constraints

$$\max_{\substack{x_0, x_1, x_\star \\ g_0, g_\star \\ f_0, f_\star}} \quad \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}$$

$$\text{subject to} \quad \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0, \star \\ g_i = f'(x_i) & i = 0, \star \end{cases}$$

$$x_1 = x_0 - h_{1,0} g_0$$

$$g_\star = 0,$$

- replacing them by

$$f_\star \geqslant f_0 + \langle g_0, x_\star - x_0 \rangle + \tfrac{1}{2L}\|g_\star - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_\star - x_0 - \tfrac{1}{L}(g_\star - g_0)\right\|^2$$

$$f_0 \geqslant f_\star + \langle g_\star, x_0 - x_\star \rangle + \tfrac{1}{2L}\|g_0 - g_\star\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_\star - \tfrac{1}{L}(g_0 - g_\star)\right\|^2.$$

Replace constraints

- Interpolation conditions allow removing red constraints

$$\max_{\substack{x_0,x_1,x_\star\\ g_0,g_\star\\ f_0,f_\star}} \quad \frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2}$$

$$\text{subject to} \quad \exists f \in \mathcal{F}_{\mu,L} \text{ such that } \begin{cases} f_i = f(x_i) & i = 0,\star \\ g_i = f'(x_i) & i = 0,\star \end{cases}$$

$$x_1 = x_0 - h_{1,0} g_0$$

$$g_\star = 0,$$

- replacing them by

$$f_\star \geqslant f_0 + \langle g_0, x_\star - x_0\rangle + \tfrac{1}{2L}\|g_\star - g_0\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_\star - x_0 - \tfrac{1}{L}(g_\star - g_0)\right\|^2$$

$$f_0 \geqslant f_\star + \langle g_\star, x_0 - x_\star\rangle + \tfrac{1}{2L}\|g_0 - g_\star\|^2 + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_\star - \tfrac{1}{L}(g_0 - g_\star)\right\|^2.$$

- Same optimal value (no relaxation); but still non-convex quadratic problem.

Semidefinite lifting

Semidefinite lifting

◇ Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|x_0 - x_\star\|^2 & \langle g_0, x_0 - x_\star\rangle \\ \langle g_0, x_0 - x_\star\rangle & \|g_0\|^2 \end{bmatrix}, \quad F = f_0 - f_\star,$$

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|x_0 - x_\star\|^2 & \langle g_0, x_0 - x_\star\rangle \\ \langle g_0, x_0 - x_\star\rangle & \|g_0\|^2 \end{bmatrix}, \quad F = f_0 - f_\star,$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & \frac{G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2}}{G_{1,1}} \\ \text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\ & -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\ & \\ & G \succcurlyeq 0, \end{aligned}$$

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|x_0 - x_\star\|^2 & \langle g_0, x_0 - x_\star\rangle \\ \langle g_0, x_0 - x_\star\rangle & \|g_0\|^2 \end{bmatrix}, \quad F = f_0 - f_\star,$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\ \text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\ & -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0, \end{aligned}$$

(using an an homogeneity argument and substituting x_1 and $g_\star$).

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|x_0 - x_\star\|^2 & \langle g_0, x_0 - x_\star \rangle \\ \langle g_0, x_0 - x_\star \rangle & \|g_0\|^2 \end{bmatrix}, \quad F = f_0 - f_\star,$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\ \text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\ & -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0, \end{aligned}$$

(using an an homogeneity argument and substituting x_1 and $g_\star$).

- Assuming $x_0, x_\star, g_0 \in \mathbb{R}^d$ with $d \geqslant 2$, same optimal value as original problem!

Semidefinite lifting

- Using the new variables $G \succcurlyeq 0$ and F

$$G = \begin{bmatrix} \|x_0 - x_\star\|^2 & \langle g_0, x_0 - x_\star\rangle \\ \langle g_0, x_0 - x_\star\rangle & \|g_0\|^2 \end{bmatrix}, \quad F = f_0 - f_\star,$$

- previous problem can be reformulated as a 2×2 SDP

$$\begin{aligned} \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\ \text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\ & -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\ & G_{1,1} = 1 \\ & G \succcurlyeq 0, \end{aligned}$$

(using an an homogeneity argument and substituting x_1 and $g_\star$).

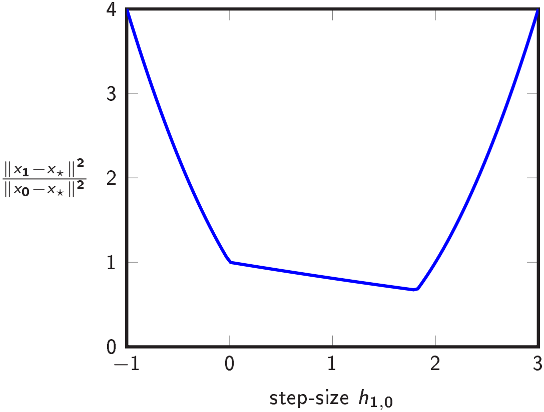
- Assuming $x_0, x_\star, g_0 \in \mathbb{R}^d$ with $d \geqslant 2$, same optimal value as original problem!
- For $d = 1$ same optimal value by adding $\operatorname{rank}(G) \leqslant 1$.

Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of $h_{1,0}$.

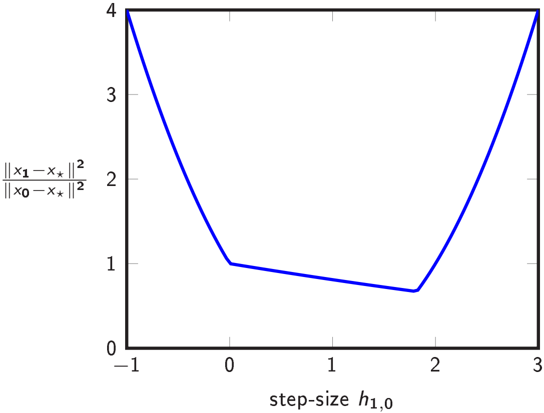
Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of $h_{1,0}$.



Solving the SDP...

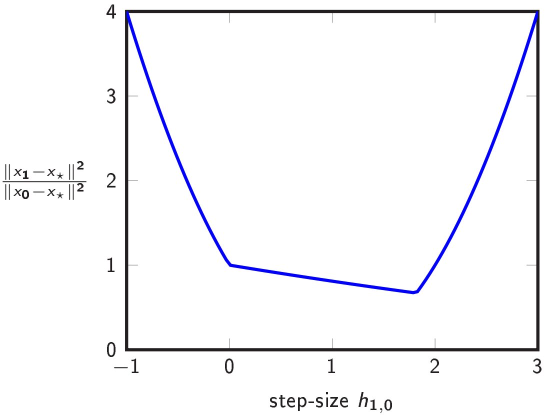
Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of $h_{1,0}$.



- ◇ Observation: numerics match $\max\{(1 - h_{1,0}L)^2, (1 - h_{1,0}\mu)^2\}$.

Solving the SDP...

Fix $L = 1$, $\mu = .1$ and solve the SDP for a few values of $h_{1,0}$.



◇ Observation: numerics match $\max\{(1 - h_{1,0}L)^2, (1 - h_{1,0}\mu)^2\}$.

◇ We recover the celebrated $\frac{2}{L+\mu}$ as the optimal step-size.

Translation to worst-case guarantees

◇ Summary: we can compute for the smallest $\tau(h_{1,0})$ such that

$$\|x_1 - x_\star\|^2 \leq \tau(h_{1,0})\|x_0 - x_\star\|^2$$

is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - h_{1,0} f'(x_0)$.

Translation to worst-case guarantees

- Summary: we can compute for the smallest $\tau(h_{1,0})$ such that

$$\|x_1 - x_\star\|^2 \leq \tau(h_{1,0})\|x_0 - x_\star\|^2$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - h_{1,0} f'(x_0)$.
- Feasible points to the previous SDP correspond to lower bounds on $\tau(h_{1,0})$.

Translation to worst-case guarantees

- Summary: we can compute for the smallest $\tau(h_{1,0})$ such that

$$\|x_1 - x_\star\|^2 \leq \tau(h_{1,0})\|x_0 - x_\star\|^2$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,\mathrm{L}}$, and $x_1 = x_0 - h_{1,0} f'(x_0)$.

- Feasible points to the previous SDP correspond to lower bounds on $\tau(h_{1,0})$.

- Note: many things can be said on such problems (obtaining rigorous proof, etc.)
  - Many details & references in `https://francisbach.com/computer-aided-analyses/`
  - Performance-Estimation-Toolbox (PESTO) on Github.

Translation to worst-case guarantees

- Summary: we can compute for the smallest $\tau(h_{1,0})$ such that

$$\|x_1 - x_\star\|^2 \leq \tau(h_{1,0})\|x_0 - x_\star\|^2$$

  is satisfied for all $x_0 \in \mathbb{R}^d$, $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}$, and $x_1 = x_0 - h_{1,0}f'(x_0)$.

- Feasible points to the previous SDP correspond to lower bounds on $\tau(h_{1,0})$.
- Note: many things can be said on such problems (obtaining rigorous proof, etc.)
  - Many details & references in `https://francisbach.com/computer-aided-analyses/`
  - Performance-Estimation-Toolbox (PESTO) on Github.
- For now: what about minimizing $\tau(h_{1,0})$? And for more complicated methods?

Primal problem

Primal problem

- Recall primal problem, with step-size optimization

$$
\begin{aligned}
\min_{h_{1,0}} \ \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\
\text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\
& -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

Primal problem

- Recall primal problem, with step-size optimization

$$
\begin{aligned}
\min_{h_{1,0}} \ \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\
\text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\
& -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

- "Simple" minimization problem by dualizing inner maximization.

Primal problem

- Recall primal problem, with step-size optimization

$$
\begin{aligned}
\min_{h_{1,0}} \; \max_{G,\,F} \quad & G_{1,1} + h_{1,0}^2 G_{2,2} - 2h_{1,0} G_{1,2} \\
\text{subject to} \quad & F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{L}{L-\mu} G_{1,2} \leqslant 0 \\
& -F + \tfrac{L\mu}{2(L-\mu)} G_{1,1} + \tfrac{1}{2(L-\mu)} G_{2,2} - \tfrac{\mu}{L-\mu} G_{1,2} \leqslant 0 \\
& G_{1,1} = 1 \\
& G \succcurlyeq 0.
\end{aligned}
$$

- "Simple" minimization problem by dualizing inner maximization.
- Introduce dual variables λ_1, λ_2 and τ for the linear constraints, and dualize.

Dual problem

- Dual problem is

$$\begin{aligned} \min_{\tau,\lambda_1,\lambda_2\geqslant 0} \ & \tau \\ \text{subject to } S = & \begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0 \\ 0 = & \lambda_1-\lambda_2. \end{aligned}$$

Dual problem

- Dual problem is

$$
\begin{aligned}
&\min_{\tau,\lambda_1,\lambda_2\geqslant 0} \tau\\
&\text{subject to } S = \begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0\\
&\qquad\qquad 0=\lambda_1-\lambda_2.
\end{aligned}
$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate

Dual problem

- Dual problem is

$$\min_{\tau,\lambda_1,\lambda_2\geqslant 0} \tau$$

$$\text{subject to } S=\begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix}\preccurlyeq 0$$

$$0=\lambda_1-\lambda_2.$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate
- Direct consequence: for any $\tau\geqslant 0$ we have

$$\|x_1-x_\star\|^2\leq\tau\|x_0-x_\star\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d, \text{ all } d\in\mathbb{N},$$
$$\text{with } x_1=x_0-h_{1,0}f'(x_0).$$

$$\Uparrow$$

$$\exists\lambda\geq 0:\ \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix}\preccurlyeq 0$$

Dual problem

- Dual problem is

$$\min_{\tau,\lambda_1,\lambda_2\geqslant 0} \tau$$

$$\text{subject to } S = \begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

$$0=\lambda_1-\lambda_2.$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$).
- Direct consequence: for any $\tau \geqslant 0$ we have

$$\|x_1-x_\star\|^2 \leq \tau\|x_0-x_\star\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d, \text{ all } d\in\mathbb{N},$$
$$\text{with } x_1 = x_0 - h_{1,0}f'(x_0).$$

$$\Uparrow$$

$$\exists\lambda\geq 0: \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

Dual problem

- Dual problem is

$$\min_{\tau,\lambda_1,\lambda_2\geqslant 0} \tau$$

$$\text{subject to } S = \begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

$$0 = \lambda_1 - \lambda_2.$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$).
- Direct consequence: for any $\tau \geqslant 0$ we have

$$\|x_1 - x_\star\|^2 \leq \tau\|x_0 - x_\star\|^2 \text{ for all } f \in \mathcal{F}_{\mu,L}, \text{ all } x_0 \in \mathbb{R}^d, \text{ all } d \in \mathbb{N},$$
$$\text{with } x_1 = x_0 - h_{1,0} f'(x_0).$$

$$\Uparrow$$

$$\exists\lambda \geq 0 : \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

- Strong duality holds (existence of a Slater point): any valid worst-case convergence rate $\equiv$ valid dual feasible point ($\Downarrow$) .

Dual problem

- Dual problem is

$$\min_{\tau,\lambda_1,\lambda_2\geqslant 0} \tau$$

$$\text{subject to } S = \begin{bmatrix} \frac{\mu+L(\lambda_1\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda_1(\mu+L)}{2(L-\mu)} & \frac{\lambda_1}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

$$0 = \lambda_1 - \lambda_2.$$

- Weak duality: any dual feasible point $\equiv$ valid worst-case convergence rate ($\Uparrow$).
- Direct consequence: for any $\tau \geqslant 0$ we have

$$\|x_1-x_\star\|^2 \leq \tau\|x_0-x_\star\|^2 \text{ for all } f\in\mathcal{F}_{\mu,L}, \text{ all } x_0\in\mathbb{R}^d, \text{ all } d\in\mathbb{N},$$
$$\text{with } x_1 = x_0 - h_{1,0}f'(x_0).$$

$$\Updownarrow$$

$$\exists\lambda\geq 0 : \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \preccurlyeq 0$$

- Strong duality holds (existence of a Slater point): any valid worst-case convergence rate $\equiv$ valid dual feasible point ($\Downarrow$) : hence "$\Updownarrow$".

Optimizing the step-sizes

Optimizing the step-sizes

- In this case ($N = 1$), optimizing over step-size $h_{1,0}$ remains convex!

Optimizing the step-sizes

- In this case ($N = 1$), optimizing over step-size $h_{1,0}$ remains convex!
- Indeed:

$$\begin{aligned} \min_{\tau,\lambda\geqslant 0} \quad & \tau \\ \text{subject to} \quad & \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \succcurlyeq 0. \end{aligned}$$

Optimizing the step-sizes

- In this case ($N = 1$), optimizing over step-size $h_{1,0}$ remains convex!
- Indeed:

$$\min_{\tau,\lambda\geqslant 0,h_{1,0}} \tau$$
$$\text{subject to } \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \succcurlyeq 0.$$

Optimizing the step-sizes

- In this case ($N = 1$), optimizing over step-size $h_{1,0}$ remains convex!
- Indeed:

$$\min_{\tau,\lambda\geqslant 0,h_{1,0}} \tau$$

$$\text{subject to } \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} \\ h_{1,0}-\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu}-h_{1,0}^2 \end{bmatrix} \succcurlyeq 0.$$

- Optimization jointly over $h_{1,0}$ "for free" (still linear SDP via Schur complement).

$$\min_{\tau,\lambda\geqslant 0,h_{1,0}} \tau$$

$$\text{subject to } \begin{bmatrix} \frac{\mu+L(\lambda\mu-1)}{L-\mu}+\tau & -\frac{\lambda(\mu+L)}{2(L-\mu)} & 1 \\ -\frac{\lambda(\mu+L)}{2(L-\mu)} & \frac{\lambda}{L-\mu} & -h_{1,0} \\ 1 & -h_{1,0} & 1 \end{bmatrix} \succcurlyeq 0.$$

Optimizing the step-sizes

- Recall first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
$$

Optimizing the step-sizes

- Recall first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\ \vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
$$

- idea: solve minimization problem over

$$
\min_{\{h_{i,j}\}} \tau(\{h_{i,j}\}),
$$

where $\tau(.)$ can be computed via $(N+1) \times (N+1)$ SDP.

Optimizing the step-sizes

- Recall first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
$$

- idea: solve minimization problem over

$$\min_{\{h_{i,j}\}} \tau(\{h_{i,j}\}),$$

  where $\tau(.)$ can be computed via $(N+1) \times (N+1)$ SDP.
- Using similar ideas and dualization: minimax transformed to minimization.

Optimizing the step-sizes

- Recall first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
$$

- idea: solve minimization problem over

$$\min_{\{h_{i,j}\}} \tau(\{h_{i,j}\}),$$

where $\tau(.)$ can be computed via $(N+1) \times (N+1)$ SDP.

- Using similar ideas and dualization: minimax transformed to minimization.
- This time, more complicated, and nonconvex (example with $N = 2$ next slide).

Optimizing the step-sizes

- Recall first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i),
\end{aligned}
$$

- idea: solve minimization problem over

$$\min_{\{h_{i,j}\}} \tau(\{h_{i,j}\}),$$

  where $\tau(.)$ can be computed via $(N+1) \times (N+1)$ SDP.
- Using similar ideas and dualization: minimax transformed to minimization.
- This time, more complicated, and nonconvex (example with $N = 2$ next slide).
- Idea: use numerical inspiration to find tractable relaxations/reformulations.

Optimizing the step-sizes

Optimizing the step-sizes

- When $N = 2$, the problem becomes

$$\min_{\substack{\tau,\lambda_1,\ldots,\lambda_6 \geqslant 0\\ \{h_{i,j}\}}} \tau$$

subject to

Optimizing the step-sizes

- When $N = 2$, the problem becomes

$$\begin{aligned} \min_{\substack{\tau, \lambda_1, \ldots, \lambda_6 \geqslant 0 \\ \{h_{i,j}\}}} \; & \tau \\ \text{subject to } & \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{1,2} & S_{2,2} & S_{2,3} \\ S_{1,3} & S_{2,3} & S_{3,3} \end{bmatrix} \succcurlyeq 0 \\ & \begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 - \lambda_5 \\ -\lambda_1 + \lambda_3 + \lambda_4 - \lambda_6 \end{bmatrix} = 0, \end{aligned}$$

Optimizing the step-sizes

◇ When $N = 2$, the problem becomes

$$\begin{aligned} \min_{\substack{\tau,\lambda_1,\ldots,\lambda_6 \geqslant 0 \\ \{h_{i,j}\}}} \; & \tau \\ \text{subject to } & \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{1,2} & S_{2,2} & S_{2,3} \\ S_{1,3} & S_{2,3} & S_{3,3} \end{bmatrix} \succcurlyeq 0 \\ & \begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 - \lambda_5 \\ -\lambda_1 + \lambda_3 + \lambda_4 - \lambda_6 \end{bmatrix} = 0, \end{aligned}$$

for some $S_{1,1}, S_{1,2}, S_{1,3}, S_{2,2}, S_{2,3}, S_{3,3}$ (functions of $\tau, \lambda_1, \ldots, \lambda_6$ and $\{h_{i,j}\}$).

Optimizing the step-sizes

◇ When $N = 2$, the problem becomes

$$\min_{\substack{\tau,\lambda_1,\ldots,\lambda_6 \geqslant 0\\ \{h_{i,j}\}}} \tau$$

$$\text{subject to} \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{1,2} & S_{2,2} & S_{2,3} \\ S_{1,3} & S_{2,3} & S_{3,3} \end{bmatrix} \succcurlyeq 0$$

$$\begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 - \lambda_5 \\ -\lambda_1 + \lambda_3 + \lambda_4 - \lambda_6 \end{bmatrix} = 0,$$

for some $S_{1,1}, S_{1,2}, S_{1,3}, S_{2,2}, S_{2,3}, S_{3,3}$ (functions of $\tau, \lambda_1, \ldots, \lambda_6$ and $\{h_{i,j}\}$).

◇ In particular

$$S_{1,2} = -\frac{L\lambda_3 - 2(L-\mu)h_{2,0} + \mu\lambda_1 + L\mu(\lambda_2+\lambda_5)h_{1,0}}{L-\mu}$$

$$S_{2,2} = \frac{-2(\mu\lambda_6 + L\lambda_4)h_{1,0} - 2(L-\mu)h_{2,0}^2 + L\mu(\lambda_2+\lambda_4+\lambda_5+\lambda_6)h_{1,0}^2 + \lambda_1+\lambda_3+\lambda_4+\lambda_6}{L-\mu}$$

Optimizing the step-sizes

- When $N = 2$, the problem becomes

$$\min_{\substack{\tau, \lambda_1, \ldots, \lambda_6 \geqslant 0 \\ \{h_{i,j}\}}} \tau$$

$$\text{subject to } \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{1,2} & S_{2,2} & S_{2,3} \\ S_{1,3} & S_{2,3} & S_{3,3} \end{bmatrix} \succcurlyeq 0$$

$$\begin{bmatrix} \lambda_1 + \lambda_2 - \lambda_3 - \lambda_5 \\ -\lambda_1 + \lambda_3 + \lambda_4 - \lambda_6 \end{bmatrix} = 0,$$

for some $S_{1,1}, S_{1,2}, S_{1,3}, S_{2,2}, S_{2,3}, S_{3,3}$ (functions of $\tau, \lambda_1, \ldots, \lambda_6$ and $\{h_{i,j}\}$).

- In particular

$$S_{1,2} = -\frac{L\lambda_3 - 2(L-\mu)h_{2,0} + \mu\lambda_1 + L\mu(\lambda_2+\lambda_5)h_{1,0}}{L-\mu}$$

$$S_{2,2} = \frac{-2(\mu\lambda_6 + L\lambda_4)h_{1,0} - 2(L-\mu)h_{2,0}^2 + L\mu(\lambda_2+\lambda_4+\lambda_5+\lambda_6)h_{1,0}^2 + \lambda_1+\lambda_3+\lambda_4+\lambda_6}{L-\mu}$$

- LMI remains convex in some step-sizes ($h_{2,0}$ and $h_{2,1}$) but not in the others.

Optimizing the step-sizes, numerically

- ◇ Even for $N = 2$ the problem does not seem “that simple”.

Optimizing the step-sizes, numerically

- ◇ Even for $N = 2$ the problem does not seem "that simple".
- ◇ Our approach:

  (i) "relax & reformulate" (details in the paper #1)

  (ii) a constructive technique (based on similar ingredients) for generating algo-independent lower bounds (see paper #2)

Optimizing the step-sizes, numerically

- Even for $N = 2$ the problem does not seem "that simple".
- Our approach:
  - (i) "relax & reformulate" (details in the paper #1)
  - (ii) a constructive technique (based on similar ingredients) for generating algo-independent lower bounds (see paper #2)
- It provides a recipe for designing methods, numerically.

Optimizing the step-sizes, numerically

- Even for $N = 2$ the problem does not seem "that simple".
- Our approach:
  - (i) "relax & reformulate" (details in the paper #1)
  - (ii) a constructive technique (based on similar ingredients) for generating algo-independent lower bounds (see paper #2)
- It provides a recipe for designing methods, numerically.
- Allows obtaining analytical solutions to the design problem, in some cases.

Optimizing the step-sizes, numerically

- Even for $N = 2$ the problem does not seem "that simple".
- Our approach:
  - (i) "relax & reformulate" (details in the paper #1)
  - (ii) a constructive technique (based on similar ingredients) for generating algo-independent lower bounds (see paper #2)
- It provides a recipe for designing methods, numerically.
- Allows obtaining analytical solutions to the design problem, in some cases.
- Recall again the first-order method of interest

$$
\begin{aligned}
x_1 &= x_0 - h_{1,0} f'(x_0) \\
x_2 &= x_1 - h_{2,0} f'(x_0) - h_{2,1} f'(x_1) \\
x_3 &= x_2 - h_{3,0} f'(x_0) - h_{3,1} f'(x_1) - h_{3,2} f'(x_2) \\
&\vdots \\
x_N &= x_{N-1} - \sum_{i=0}^{N-1} h_{N,i} f'(x_i).
\end{aligned}
$$

Numerical examples I

Example for $L = 1$ and $\mu = .1$

Numerical examples I

Example for $L = 1$ and $\mu = .1$

- For $N = 1$, we obtain $\frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.6694$ with corresponding step-sizes

$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

Numerical examples I

Example for $L = 1$ and $\mu = .1$

- ◇ For $N = 1$, we obtain $\frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.6694$ with corresponding step-sizes

$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

- ◇ For $N = 2$, we obtain $\frac{\|x_2 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.3769$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & \\ 0.2038 & 2.4961 \end{bmatrix}.$$

Numerical examples I

Example for $L = 1$ and $\mu = .1$

- For $N = 1$, we obtain $\frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.6694$ with corresponding step-sizes

$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

- For $N = 2$, we obtain $\frac{\|x_2 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.3769$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & \\ 0.2038 & 2.4961 \end{bmatrix}.$$

- For $N = 3$, we obtain $\frac{\|x_3 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.1932$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & & \\ 0.1142 & 1.8380 & \\ 0.0642 & 0.4712 & 2.8404 \end{bmatrix}.$$

Numerical examples I

Example for $L = 1$ and $\mu = .1$

- For $N = 1$, we obtain $\frac{\|x_1 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.6694$ with corresponding step-sizes

$$[h^\star_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

- For $N = 2$, we obtain $\frac{\|x_2 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.3769$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & \\ 0.2038 & 2.4961 \end{bmatrix}.$$

- For $N = 3$, we obtain $\frac{\|x_3 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.1932$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & & \\ 0.1142 & 1.8380 & \\ 0.0642 & 0.4712 & 2.8404 \end{bmatrix}.$$

- For $N = 4$, we obtain $\frac{\|x_4 - x_\star\|^2}{\|x_0 - x_\star\|^2} \leq 0.0944$ with

$$[h^\star_{i,j}] = \begin{bmatrix} 1.5466 & & & \\ 0.1142 & 1.8380 & & \\ 0.0331 & 0.2432 & 1.9501 & \\ 0.0217 & 0.1593 & 0.6224 & 3.0093 \end{bmatrix}.$$

Numerical examples II

What about different performance measure? Example $\frac{f(x_N)-f_\star}{f(x_0)-f_\star}$ and $L=1$, $\mu=.1$.

Numerical examples II

What about different performance measure? Example $\frac{f(x_N)-f_\star}{f(x_0)-f_\star}$ and $L = 1$, $\mu = .1$.

◇ For $N = 1$, we obtain $\frac{f(x_1)-f_\star}{f(x_0)-f_\star} \leq 0.6694$ with step-size

$$[h_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

Numerical examples II

What about different performance measure? Example $\frac{f(x_N)-f_\star}{f(x_0)-f_\star}$ and $L=1$, $\mu=.1$.

- For $N=1$, we obtain $\frac{f(x_1)-f_\star}{f(x_0)-f_\star} \leq 0.6694$ with step-size

$$[h_{i,j}] = \begin{bmatrix} 1.8182 \end{bmatrix}.$$

- For $N=2$, we obtain $\frac{f(x_2)-f_\star}{f(x_0)-f_\star} \leq 0.3554$ with

$$[h_{i,j}] = \begin{bmatrix} 2.0095 & \\ 0.4229 & 2.0095 \end{bmatrix}.$$

Numerical examples II

What about different performance measure? Example $\frac{f(x_N)-f_\star}{f(x_0)-f_\star}$ and $L=1$, $\mu=.1$.

- For $N=1$, we obtain $\frac{f(x_1)-f_\star}{f(x_0)-f_\star}\leq 0.6694$ with step-size

$$[h_{i,j}] = \begin{bmatrix}1.8182\end{bmatrix}.$$

- For $N=2$, we obtain $\frac{f(x_2)-f_\star}{f(x_0)-f_\star}\leq 0.3554$ with

$$[h_{i,j}] = \begin{bmatrix}2.0095 & \\ 0.4229 & 2.0095\end{bmatrix}.$$

- For $N=3$, we obtain $\frac{f(x_3)-f_\star}{f(x_0)-f_\star}\leq 0.1698$ with

$$[h_{i,j}] = \begin{bmatrix}1.9470 & & \\ 0.4599 & 2.2406 & \\ 0.1705 & 0.4599 & 1.9470\end{bmatrix}.$$

Numerical examples II

What about different performance measure? Example $\frac{f(x_N)-f_\star}{f(x_0)-f_\star}$ and $L=1$, $\mu=.1$.

- ◇ For $N=1$, we obtain $\frac{f(x_1)-f_\star}{f(x_0)-f_\star} \leq 0.6694$ with step-size

$$[h_{i,j}] = \begin{bmatrix}1.8182\end{bmatrix}.$$

- ◇ For $N=2$, we obtain $\frac{f(x_2)-f_\star}{f(x_0)-f_\star} \leq 0.3554$ with

$$[h_{i,j}] = \begin{bmatrix}2.0095 & \\ 0.4229 & 2.0095\end{bmatrix}.$$

- ◇ For $N=3$, we obtain $\frac{f(x_3)-f_\star}{f(x_0)-f_\star} \leq 0.1698$ with

$$[h_{i,j}] = \begin{bmatrix}1.9470 & & \\ 0.4599 & 2.2406 & \\ 0.1705 & 0.4599 & 1.9470\end{bmatrix}.$$

- ◇ For $N=4$, we obtain $\frac{f(x_4)-f_\star}{f(x_0)-f_\star} \leq 0.0789$ with

$$[h_{i,j}] = \begin{bmatrix}1.9187 & & & \\ 0.4098 & 2.1746 & & \\ 0.1796 & 0.5147 & 2.1746 & \\ 0.0627 & 0.1796 & 0.4098 & 1.9187\end{bmatrix}.$$

Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,

Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),

Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),

Numerical examples III

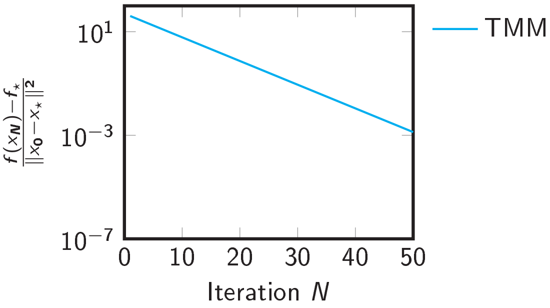
Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

- ◇ worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- ◇ worst-case performance of optimized method (numerically generated),
- ◇ conjugate-gradient based method (numerically generated),
- ◇ Lower complexity bound (numerically generated).

Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

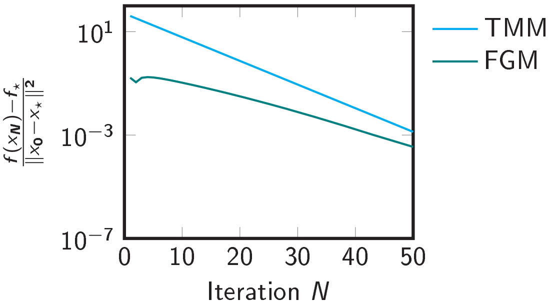
- ◇ worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- ◇ worst-case performance of optimized method (numerically generated),
- ◇ conjugate-gradient based method (numerically generated),
- ◇ Lower complexity bound (numerically generated).



Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

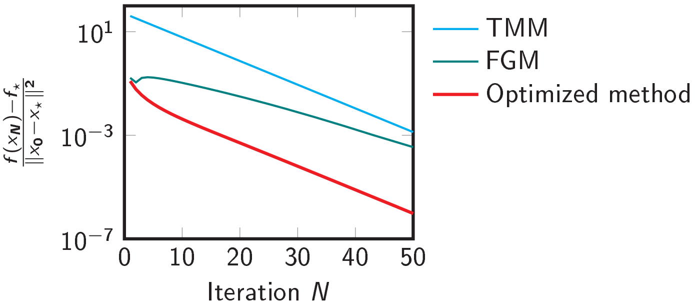
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

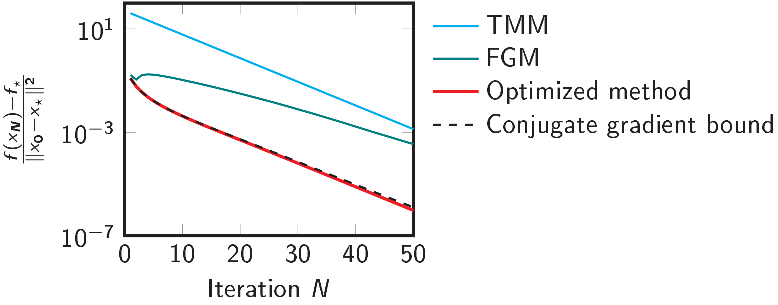
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L=1$ and $\mu=.01$. We compare

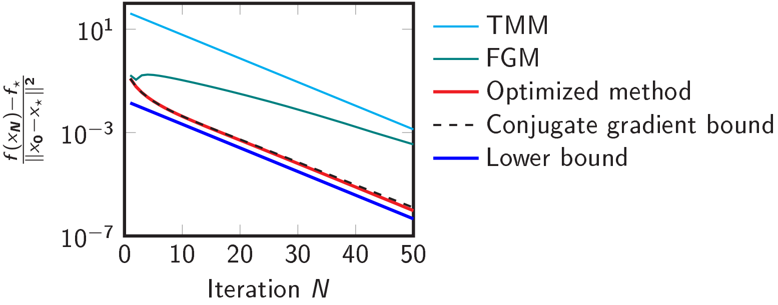
- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Numerical examples III

Worst-case performance $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $L = 1$ and $\mu = .01$. We compare

- worst-case performance of known methods, namely Fast Gradient Method (FGM), Triple Momentum Method (TMM) computed using PEPs,
- worst-case performance of optimized method (numerically generated),
- conjugate-gradient based method (numerically generated),
- Lower complexity bound (numerically generated).



Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.

Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.
- The method referred to as "Information-Theoretic Exact Method" (ITEM)

$$
\begin{aligned}
y_k &= (1 - \beta_k) z_k + \beta_k \left(y_{k-1} - \frac{1}{L} f'(y_{k-1}) \right) \\
z_{k+1} &= (1 - \tfrac{\mu}{L}\delta_k) z_k + \tfrac{\mu}{L}\delta_k \left(y_k - \frac{1}{\mu} f'(y_k) \right),
\end{aligned}
$$

for some sequences $\{\beta_k\}$, $\{\delta_k\}$ (depending on μ, L, and k).

Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.
- The method referred to as "Information-Theoretic Exact Method" (ITEM)

$$y_k = (1-\beta_k) z_k + \beta_k \left(y_{k-1} - \frac{1}{L} f'(y_{k-1}) \right)$$
$$z_{k+1} = (1 - \tfrac{\mu}{L}\delta_k) z_k + \tfrac{\mu}{L}\delta_k \left(y_k - \frac{1}{\mu} f'(y_k) \right),$$

  for some sequences $\{\beta_k\}$, $\{\delta_k\}$ (depending on μ, L, and k).
- The worst-case guarantee matches exactly a lower complexity bound.

Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.
- The method referred to as "Information-Theoretic Exact Method" (ITEM)

$$y_k = (1 - \beta_k) z_k + \beta_k \left(y_{k-1} - \frac{1}{L} f'(y_{k-1}) \right)$$
$$z_{k+1} = (1 - \tfrac{\mu}{L}\delta_k) z_k + \tfrac{\mu}{L}\delta_k \left(y_k - \frac{1}{\mu} f'(y_k) \right),$$

for some sequences $\{\beta_k\}$, $\{\delta_k\}$ (depending on μ, L, and k).

- The worst-case guarantee matches exactly a lower complexity bound.
- Worst-case guarantee of order

$$\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2} = O\left(\left(1 - \sqrt{\tfrac{\mu}{L}} \right)^{2N} \right).$$

Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.
- The method referred to as "Information-Theoretic Exact Method" (ITEM)

$$y_k = (1-\beta_k)z_k + \beta_k\left(y_{k-1} - \frac{1}{L}f'(y_{k-1})\right)$$
$$z_{k+1} = (1 - \tfrac{\mu}{L}\delta_k)z_k + \tfrac{\mu}{L}\delta_k\left(y_k - \frac{1}{\mu}f'(y_k)\right),$$

  for some sequences $\{\beta_k\}$, $\{\delta_k\}$ (depending on μ, L, and k).
- The worst-case guarantee matches exactly a lower complexity bound.
- Worst-case guarantee of order

$$\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2} = O\left(\left(1 - \sqrt{\tfrac{\mu}{L}}\right)^{2N}\right).$$

- Asymptotically, this method corresponds to the Triple Momentum Method by Van Scoy et al. (2017).

Analytical solutions

- It turns out that for $\frac{\|x_N - x_\star\|^2}{\|x_0 - x_\star\|^2}$, we can also solve the problem analytically.
- The method referred to as "Information-Theoretic Exact Method" (ITEM)

$$
\begin{aligned}
y_k &= (1-\beta_k)z_k + \beta_k\left(y_{k-1} - \frac{1}{L}f'(y_{k-1})\right)\\
z_{k+1} &= (1-\tfrac{\mu}{L}\delta_k)z_k + \tfrac{\mu}{L}\delta_k\left(y_k - \frac{1}{\mu}f'(y_k)\right),
\end{aligned}
$$

for some sequences $\{\beta_k\}$, $\{\delta_k\}$ (depending on μ, L, and k).

- The worst-case guarantee matches exactly a lower complexity bound.
- Worst-case guarantee of order

$$
\frac{\|z_N - z_\star\|^2}{\|z_0 - z_\star\|^2} = O\left(\left(1 - \sqrt{\tfrac{\mu}{L}}\right)^{2N}\right).
$$

- Asymptotically, this method corresponds to the Triple Momentum Method by Van Scoy et al. (2017).
- All details can be found in (T. & Drori, 2021), and (Drori & T., 2021).

A few observations/limitations

Were we lucky? Some pieces might be missing!

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- ◇ $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu = 0$: optimized gradient method (OGM, Kim & Fessler 2016),

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- $\frac{\|f'(x_N)\|^2}{f(x_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- $\frac{\|f'(x_N)\|^2}{f(x_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

Relation to quadratics? When specifying f to be quadratic, similar known methods

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- $\frac{\|f'(x_N)\|^2}{f(x_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

Relation to quadratics? When specifying f to be quadratic, similar known methods

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$ (via Chebyshev polynomials),

A few observations/limitations

Were we lucky? Some pieces might be missing!

- ◇ Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- ◇ Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- ◇ $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- ◇ $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- ◇ $\frac{\|f'(x_N)\|^2}{f(x_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

Relation to quadratics? When specifying f to be quadratic, similar known methods

- ◇ $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$ (via Chebyshev polynomials),
- ◇ $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$ (via Chebyshev polynomials), asymptotically Polyak's Heavy-Ball

A few observations/limitations

Were we lucky? Some pieces might be missing!

- Why/when are optimal step-sizes $\{h^\star_{i,j}\}$ independent of horizon N?
- Why/when can the optimal method be expressed efficiently? (eg. using second order recursions)

The situation seems quite involved in general, apart from a few cases

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$: optimized gradient method (OGM, Kim & Fessler 2016),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$: information-theoretic exact method (ITEM, T & Drori 2021),
- $\frac{\|f'(x_N)\|^2}{f(x_0)-f_\star}$ with $\mu=0$: OGM for gradient (OGM-G, Kim & Fessler 2021).

Relation to quadratics? When specifying f to be quadratic, similar known methods

- $\frac{f(x_N)-f_\star}{\|x_0-x_\star\|^2}$ with $\mu=0$ (via Chebyshev polynomials),
- $\frac{\|x_N-x_\star\|^2}{\|x_0-x_\star\|^2}$ (via Chebyshev polynomials), asymptotically Polyak's Heavy-Ball
- see e.g.: A. Nemirovsky's "Information-based complexity of convex programming." (lecture notes, 1995)

On worst-case analyses

Step-sizes optimization

Constructing lower bounds

Software

Concluding remarks

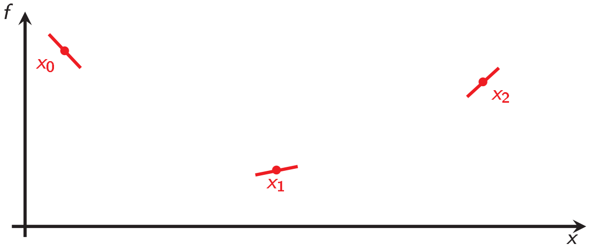
Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!

Reminder: smooth strongly convex interpolation/extension

Consider a set S, and its associated values $\{(x_i, g_i, f_i)\}_{i\in S}$ with coordinates x_i, subgradients g_i and function values f_i.



? Possible to find a $f \in \mathcal{F}_{\mu,L}$ s.t.

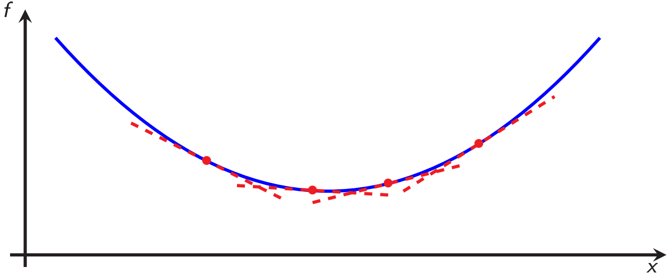
$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?

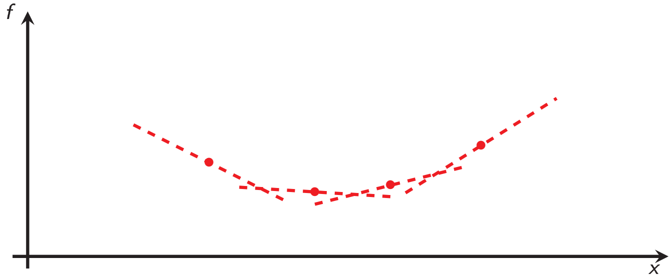
Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i\in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Special case: convex interpolation problem

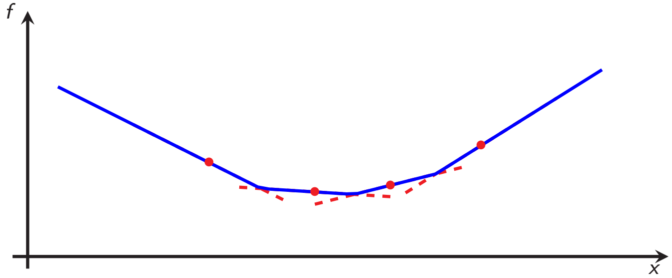
Conditions for $\{(x_i, g_i, f_i)\}_{i\in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Conditions $f_i \geq f_j + \langle g_j, x_i - x_j\rangle$ is nec.

Special case: convex interpolation problem

Conditions for $\{(x_i, g_i, f_i)\}_{i\in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Conditions $f_i \geq f_j + \langle g_j, x_i - x_j \rangle$ is nec. and suff.

Explicit construction:

$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\},$$

Not unique.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- ◇ For obtaining *tight* SDP representation of the worst-case computation problem.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- Similar constructions for smooth (possibly strongly) convex functions.

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \left\langle g_j, x - x_j \right\rangle \right\}.$$
- Similar constructions for smooth (possibly strongly) convex functions.
- We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- Similar constructions for smooth (possibly strongly) convex functions.
- We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- Similar constructions for smooth (possibly strongly) convex functions.
- We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;
  - those constraints fit into a SDP;

Shape of lower complexity bounds

Role of extension/interpolation results, so far?

- For obtaining *tight* SDP representation of the worst-case computation problem.
- We can infer shapes for the worst-case functions!
  - Why? Let's flashback into the interpolation/extension problem!
- Example: (ccp) convex minimization, worst-case problems can be assumed to have the form
$$f(x) = \max_j \left\{ f_j + \langle g_j, x - x_j \rangle \right\}.$$
- Similar constructions for smooth (possibly strongly) convex functions.
- We can use that for generating algorithm-independent lower bounds numerically (with a few additional ingredients)
  - idea: impose a few additional constraints on the structure so that any black-box first-order method has the "same information" at each iteration;
  - those constraints fit into a SDP;
  - such functions are sometimes referred to as being "zero-chain".

Avoiding semidefinite programming modeling steps?

- ◇ Performance Estimation Toolbox (PESTO) available on

  AdrienTaylor/Performance-Estimation-Toolbox

  Contains about 50 examples.
- ◇ Python version should be available during summer.

Example: convergence of Douglas-Rachford

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}     = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation

for k = 1 : N
    x{k}    = proximal_step(w{k},B,lambda);
    y{k}    = proximal_step(2*x{k}-w{k},A,lambda);
    w{k+1}  = w{k}-theta*(x{k}-y{k});

    xp{k}   = proximal_step(wp{k},B,lambda);
    yp{k}   = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}   = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)    % worst-case contraction factor
```

Example: convergence of Douglas-Rachford

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}     = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation

x{k}     = proximal_step(w{k},B,lambda);
y{k}     = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}   = w{k}-theta*(x{k}-y{k});
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)    % worst-case contraction factor
```

Example: convergence of Douglas-Rachford

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}    = P.StartingPoint(); wp{1}   = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;       % step size (in the resolvents)
theta  = .9;         % overrelaxation
```

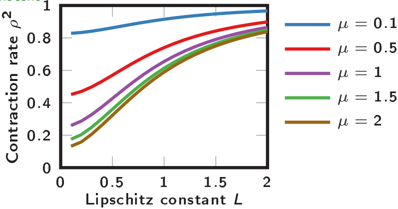
```
x{k}     = proximal_step(w{k},B,lambda);
y{k}     = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}   = w{k}-theta*(x{k}-y{k});
```

```
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)    % worst-case contraction factor
```



Example: convergence of Douglas-Rachford

```
% (0) Initialize an empty PEP
P=pep();

N = 1;
% (1) Set up the class of monotone inclusions
paramA.L  =  1; paramA.mu = 0; % A is 1-Lipschitz and 0-strongly monotone
paramB.mu = .1;                % B is .1-strongly monotone

A = P.DeclareFunction('LipschitzStronglyMonotone',paramA);
B = P.DeclareFunction('StronglyMonotone',paramB);

w  = cell(N+1,1);   wp = cell(N+1,1);
x  = cell(N,1);     xp = cell(N,1);
y  = cell(N,1);     yp = cell(N,1);

% (2) Set up the starting points
w{1}    = P.StartingPoint(); wp{1}    = P.StartingPoint();
P.InitialCondition((w{1}-wp{1})^2<=1);

% (3) Algorithm
lambda = 1.3;        % step size (in the resolvents)
theta  = .9;          % overrelaxation
```

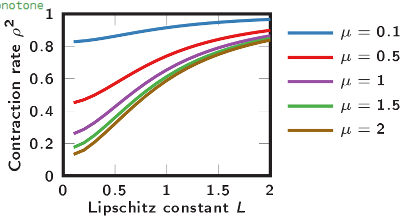
```
x{k}     = proximal_step(w{k},B,lambda);
y{k}     = proximal_step(2*x{k}-w{k},A,lambda);
w{k+1}   = w{k}-theta*(x{k}-y{k});
```

```
    xp{k}    = proximal_step(wp{k},B,lambda);
    yp{k}    = proximal_step(2*xp{k}-wp{k},A,lambda);
    wp{k+1}    = wp{k}-theta*(xp{k}-yp{k});
end

% (4) Set up the performance measure: ||z0-z1||^2
P.PerformanceMetric((w{k+1}-wp{k+1})^2);

% (5) Solve the PEP
P.solve()

% (6) Evaluate the output
double((w{k+1}-wp{k+1})^2)    % worst-case contraction factor
```



- ✔ fast prototyping ($\sim$ 20 effective lines)
- ✔ quick analyses ($\sim$ 10 minutes)
- ✔ computer-aided proofs (multipliers)

Current library of examples within PESTO

Includes... but not limited to

- subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- proximal point algorithm,
- projected and proximal gradient, accelerated/momentum versions,
- steepest descent, greedy/conjugate gradient methods,
- Douglas-Rachford/three operator splitting,
- Frank-Wolfe/conditional gradient,
- inexact gradient/fast gradient,
- Krasnoselskii-Mann and Halpern fixed-point iterations,
- mirror descent,
- stochastic methods: SAG, SAGA, SGD and variants.

Current library of examples within PESTO

Includes... but not limited to

- ◇ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ◇ proximal point algorithm,
- ◇ projected and proximal gradient, accelerated/momentum versions,
- ◇ steepest descent, greedy/conjugate gradient methods,
- ◇ Douglas-Rachford/three operator splitting,
- ◇ Frank-Wolfe/conditional gradient,
- ◇ inexact gradient/fast gradient,
- ◇ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ◇ mirror descent,
- ◇ stochastic methods: SAG, SAGA, SGD and variants.

Current library of examples within PESTO

Includes... but not limited to

- ◇ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ◇ proximal point algorithm,
- ◇ projected and proximal gradient, accelerated/momentum versions,
- ◇ steepest descent, greedy/conjugate gradient methods,
- ◇ Douglas-Rachford/three operator splitting,
- ◇ Frank-Wolfe/conditional gradient,
- ◇ inexact gradient/fast gradient,
- ◇ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ◇ mirror descent,
- ◇ stochastic methods: SAG, SAGA, SGD and variants.

PESTO contains most of the recent PEP-related advances (including techniques by other groups). Clean updated references in user manual.

Current library of examples within PESTO

Includes... but not limited to

- ◇ subgradient, gradient, heavy-ball, fast gradient, optimized gradient methods,
- ◇ proximal point algorithm,
- ◇ projected and proximal gradient, accelerated/momentum versions,
- ◇ steepest descent, greedy/conjugate gradient methods,
- ◇ Douglas-Rachford/three operator splitting,
- ◇ Frank-Wolfe/conditional gradient,
- ◇ inexact gradient/fast gradient,
- ◇ Krasnoselskii-Mann and Halpern fixed-point iterations,
- ◇ mirror descent,
- ◇ stochastic methods: SAG, SAGA, SGD and variants.

PESTO contains most of the recent PEP-related advances (including techniques by other groups). Clean updated references in user manual.

Among others, see works by Drori, Teboulle, Kim, Fessler, Ryu, Lieder, Lessard, Recht, Packard, Van Scoy, Cyrus, Gu, Yang, etc.

On worst-case analyses

Step-sizes optimization

Constructing lower bounds

Software

Concluding remarks

Concluding remarks

Performance estimation's philosophy

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- overall: principled approach (definition of worst-case),

Concluding remarks

Performance estimation's philosophy

- ◇ numerically allows obtaining tight bounds (rigorous baselines),
- ◇ overall: principled approach (definition of worst-case),
- ◇ computer-assisted design of numerical methods.

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- overall: principled approach (definition of worst-case),
- computer-assisted design of numerical methods.

Difficulties:

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- overall: principled approach (definition of worst-case),
- computer-assisted design of numerical methods.

Difficulties:

- suffers from standard caveats of worst-case analyses,

Concluding remarks

Performance estimation's philosophy

- numerically allows obtaining tight bounds (rigorous baselines),
- overall: principled approach (definition of worst-case),
- computer-assisted design of numerical methods.

Difficulties:

- suffers from standard caveats of worst-case analyses,
- closed-form solutions might be involved.

Concluding remarks

Performance estimation's philosophy

- ⋄ numerically allows obtaining tight bounds (rigorous baselines),
- ⋄ overall: principled approach (definition of worst-case),
- ⋄ computer-assisted design of numerical methods.

Difficulties:

- ⋄ suffers from standard caveats of worst-case analyses,
- ⋄ closed-form solutions might be involved.

Note: many links with theory on quadratics (Chebyshev methods).

Take-home messages

Worst-cases are solutions to optimization problems

Acceleration/“optimal” methods by optimizing worst-cases

Design of theoretical methods via numerical experiments

Short bibliography

Short bibliography

Presentation mainly based on

- T., Y. Drori. "An optimal gradient method for smooth strongly convex minimization." (2021)
- Y. Drori, T. "On the oracle complexity of smooth strongly convex minimization." (2021)
- T., J. Hendrickx, F. Glineur. "Smooth strongly convex interpolation and exact worst-case performance of first-order methods." (2017)

Short bibliography

Presentation mainly based on

- ◇ T., Y. Drori. "An optimal gradient method for smooth strongly convex minimization." (2021)
- ◇ Y. Drori, T. "On the oracle complexity of smooth strongly convex minimization." (2021)
- ◇ T., J. Hendrickx, F. Glineur. "Smooth strongly convex interpolation and exact worst-case performance of first-order methods." (2017)

Many (very) related works; much more careful bibliographical treatment in papers.

- ◇ Y. Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$." (1983)
- ◇ A. Nemirovsky, and B. Polyak. "Iterative methods for solving linear ill-posed problems under precise information." (1984)
- ◇ A. Nemirovsky. "Information-based complexity of linear operator equations." (1992)
- ◇ A. Nemirovsky. "Information-based complexity of convex programming." (lecture notes, 1995)
- ◇ Y. Nesterov. "Introductory Lectures on Convex Optimization." (2003/2018)
- ◇ Y. Drori, and M. Teboulle. "Performance of first-order methods for smooth convex minimization: a novel approach." (2014)
- ◇ D. Kim, and F. Fessler. "Optimized first-order methods for smooth convex minimization." (2017)
- ◇ B. Van Scoy, R. Freeman, K. Lynch. "The fastest known globally convergent first-order method for minimizing strongly convex functions" (2017)
- ◇ D. Kim, and F. Fessler. "Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions." (2021)

Thanks! Questions?

www.di.ens.fr/∼ataylor/

AdrienTaylor/Performance-Estimation-Toolbox on Github