Computer-aided analyses for first-order methods
(via semidefinite programming)

Adrien Taylor

SSOPT — June 2019

… great collaborators!

François Glineur
(UCLouvain)

Julien Hendrickx
(UCLouvain)

Etienne de Klerk
(Tilburg & Delft)

Ernest Ryu
(UCLA)

Francis Bach
(Inria/ENS)

Yoel Drori
(Google)

Laurent Lessard
(W-Madison)

Bryan Van Scoy
(W-Madison)

Carolina Bergeling
(Lund)

Pontus Giselsson
(Lund)

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- Drori and Teboulle (2014): worst-case bounds via semidefinite programming.

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- Kim and Fessler (2016): optimized methods from semidefinite programming.

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- ◇ Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- ◇ Kim and Fessler (2016): optimized methods from semidefinite programming.
- ◇ Lessard, Recht, Packard (2016): linear convergence bounds using control theory.

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- ◇ Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- ◇ Kim and Fessler (2016): optimized methods from semidefinite programming.
- ◇ Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- ◇ T., Hendrickx and Glineur (2017): tightness and generalizations.

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- Kim and Fessler (2016): optimized methods from semidefinite programming.
- Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- T., Hendrickx and Glineur (2017): tightness and generalizations.

Taken further by different groups... mostly relying on the same ideas:

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- ◇ Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- ◇ Kim and Fessler (2016): optimized methods from semidefinite programming.
- ◇ Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- ◇ T., Hendrickx and Glineur (2017): tightness and generalizations.

Taken further by different groups... mostly relying on the same ideas:

- ◇ performance estimation problems (PEPs)—"optimization point of view",

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- Kim and Fessler (2016): optimized methods from semidefinite programming.
- Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- T., Hendrickx and Glineur (2017): tightness and generalizations.

Taken further by different groups... mostly relying on the same ideas:

- performance estimation problems (PEPs)—"optimization point of view",
- integral quadratic constraints (IQCs)—"control theoretic point of view".

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- Kim and Fessler (2016): optimized methods from semidefinite programming.
- Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- T., Hendrickx and Glineur (2017): tightness and generalizations.

Taken further by different groups... mostly relying on the same ideas:

- performance estimation problems (PEPs)—"optimization point of view",
- integral quadratic constraints (IQCs)—"control theoretic point of view".

In this presentation: PEPs, through lens of T., Hendrickx and Glineur (2017).

Topic's genealogy and credits

Modern computer-assisted proofs in optimization, "starting points":

- ⋄ Drori and Teboulle (2014): worst-case bounds via semidefinite programming.
- ⋄ Kim and Fessler (2016): optimized methods from semidefinite programming.
- ⋄ Lessard, Recht, Packard (2016): linear convergence bounds using control theory.
- ⋄ T., Hendrickx and Glineur (2017): tightness and generalizations.

Taken further by different groups... mostly relying on the same ideas:

- ⋄ performance estimation problems (PEPs)—"optimization point of view",
- ⋄ integral quadratic constraints (IQCs)—"control theoretic point of view".

In this presentation: PEPs, through lens of T., Hendrickx and Glineur (2017).

All codes used here can be found on github (link at the end).

How do we prove an algorithm works?

How do we prove an algorithm works?

Goal: automate worst-case analyses of optimization algorithms

Take-home messages

Worst-cases are solutions to optimization problems.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Often tractable for first-order methods in convex optimization!

Toy example

Performance estimation

Further examples

Convex interpolation

Conclusions and discussions

Analysis of a gradient step

We want to solve

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

with $f \in \mathcal{F}_{\mu,L}$: class of μ-strongly convex L-smooth functions.

Analysis of a gradient step

We want to solve

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

with $f \in \mathcal{F}_{\mu,L}$: class of μ-strongly convex L-smooth functions.

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

Analysis of a gradient step

We want to solve

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

with $f \in \mathcal{F}_{\mu,L}$: class of μ-strongly convex L-smooth functions.

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

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

Analysis of a gradient step

We want to solve

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

with $f \in \mathcal{F}_{\mu,L}$: class of μ-strongly convex L-smooth functions.

(Gradient method) We decide to use: $x_{k+1} = x_k - \gamma f'(x_k)$.

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

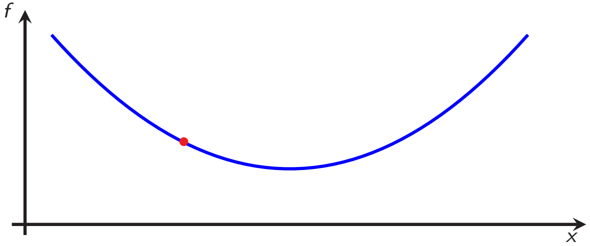
Examples: what about $f(x_N) - f(x_*)$, $\|f'(x_N)\|$, $\|x_N - x_*\|$?

About the assumptions

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:

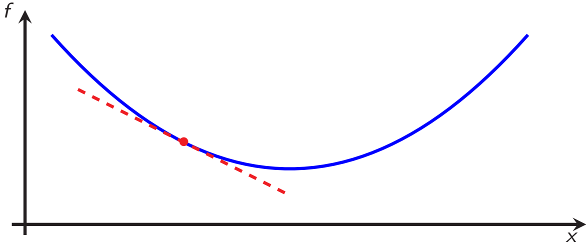
About the assumptions

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:



About the assumptions

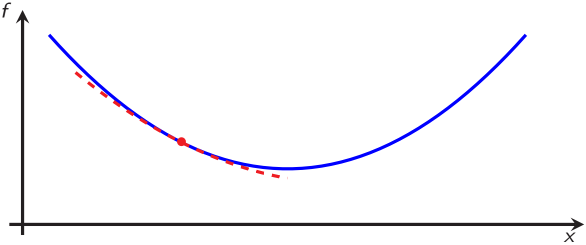
Consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, f is (μ-strongly) convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$ we have:



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

About the assumptions

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:

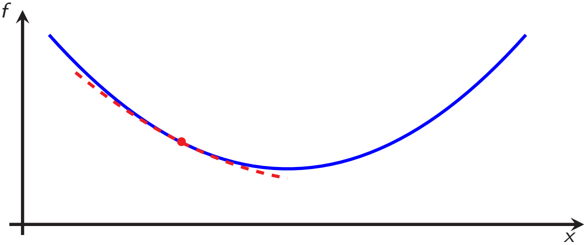


(1) (Convexity) $f(x) \geq f(y) + \langle f'(y), x - y \rangle$,

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

About the assumptions

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:



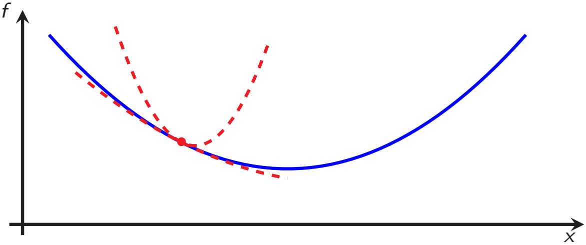
(1) (Convexity) $f(x) \geq f(y) + \langle f'(y), x - y \rangle$,

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

(2) (L-smoothness) $\|f'(x) - f'(y)\| \leq L\|x - y\|$,

About the assumptions

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) $f(x) \geq f(y) + \langle f'(y), x - y \rangle$,

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

(2) (L-smoothness) $\|f'(x) - f'(y)\| \leq L\|x - y\|$,

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

Performance estimation problem

Performance estimation problem for a gradient step

$$\max \left\|f'(x_1)\right\|^2$$

$$\text{s.t. } f \in \mathcal{F}_{\mu,L} \qquad\qquad \text{Functional class}$$

Performance estimation problem

Performance estimation problem for a gradient step

$$\max \left\|f'(x_1)\right\|^2$$

$$\text{s.t. } f \in \mathcal{F}_{\mu,L} \qquad \text{Functional class}$$

$$x_1 = x_0 - \gamma f'(x_0) \qquad \text{Algorithm}$$

Performance estimation problem

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. }\ & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Performance estimation problem

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1;

Performance estimation problem

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max \ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1; parameters: μ, L, γ, R.

Performance estimation problem

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1; parameters: μ, L, γ, R.

Can be solved using semidefinite programming (SDP):

$$\max\left\{\left\|f'(x_1)\right\|^2\right\} = \max\left\{(1-\mu\gamma)^2, (1-L\gamma)^2\right\}\left\|f'(x_0)\right\|^2$$

Performance estimation problem

Performance estimation problem for a gradient step

$$\begin{aligned} \max\ & \left\|f'(x_1)\right\|^2 & \\ \text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\ & x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\ & \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition} \end{aligned}$$

<u>Variables</u>: f, x_0, x_1; <u>parameters</u>: μ, L, γ, R.

Can be solved using semidefinite programming (SDP):

$$\max\left\{\left\|f'(x_1)\right\|^2\right\} = \max\left\{(1-\mu\gamma)^2, (1-L\gamma)^2\right\}\left\|f'(x_0)\right\|^2$$

Matching functions: $f(x) = \frac{\mu}{2}x^2$ and $f(x) = \frac{L}{2}x^2$.

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?

- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

The idea:

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

The idea:

- replace f by its discrete version:

$$f_i = f(x_i),\ g_i = f'(x_i) \quad \forall i \in \{0, 1\}\,.$$

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

The idea:

- replace f by its discrete version:

$$f_i = f(x_i),\ \ g_i = f'(x_i) \quad \forall i \in \{0, 1\}\,.$$

- Require points (x_i, g_i, f_i) to be interpolable by a function $f \in \mathcal{F}_{\mu,L}$.

From infinite to finite dimensional problems

As it is, the previous problem does not seem very practical...

- How to treat the infinite dimensional variable f?
- How to cope with the constraint $f \in \mathcal{F}_{\mu,L}$?

The idea:

- replace f by its discrete version:

$$f_i = f(x_i),\; g_i = f'(x_i) \quad \forall i \in \{0,1\}.$$

- Require points (x_i, g_i, f_i) to be interpolable by a function $f \in \mathcal{F}_{\mu,L}$.
  The new constraint is:

$$\exists f \in \mathcal{F}_{\mu,L}:\; f_i = f(x_i),\; g_i = f'(x_i), \qquad \forall i \in \{0,1\}.$$

From infinite to finite dimensional problems

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \|f'(x_1)\|^2 & & \\
\text{s.t. }\ & f \in \mathcal{F}_{\mu,L} & & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & & \text{Algorithm} \\
& \|f'(x_0)\|^2 = R^2 & & \text{Initial condition}
\end{aligned}
$$

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1;

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$\begin{aligned} \max\ & \left\|f'(x_1)\right\|^2 & \\ \text{s.t. }\ & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\ & x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\ & \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition} \end{aligned}$$

Variables: f, x_0, x_1; new formulation:

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$\begin{aligned} \max\ & \left\|f'(x_1)\right\|^2 & \\ \text{s.t. }\ & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\ & x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\ & \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition} \end{aligned}$$

<u>Variables</u>: f, x_0, x_1; new formulation:

$$\max \|g_1\|^2$$

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$\max \left\|f'(x_1)\right\|^2$$

$$\text{s.t. } f \in \mathcal{F}_{\mu,L} \qquad \text{Functional class}$$

$$x_1 = x_0 - \gamma f'(x_0) \qquad \text{Algorithm}$$

$$\left\|f'(x_0)\right\|^2 = R^2 \qquad \text{Initial condition}$$

Variables: f, x_0, x_1; new formulation:

$$\max \|g_1\|^2$$

$$\text{s.t. } \exists f \in \mathcal{F}_{\mu,L} : \; f_i = f(x_i), \; g_i = f'(x_i), \qquad \forall i \in \{0,1\}$$

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1; new formulation:

$$
\begin{aligned}
\max\ & \|g_1\|^2 \\
\text{s.t. } & \exists f \in \mathcal{F}_{\mu,L} :\ f_i = f(x_i),\ g_i = f'(x_i), \qquad \forall i \in \{0,1\} \\
& x_1 = x_0 - \gamma g_0
\end{aligned}
$$

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$
\begin{aligned}
\max\ & \left\|f'(x_1)\right\|^2 & \\
\text{s.t. } & f \in \mathcal{F}_{\mu,L} & \text{Functional class} \\
& x_1 = x_0 - \gamma f'(x_0) & \text{Algorithm} \\
& \left\|f'(x_0)\right\|^2 = R^2 & \text{Initial condition}
\end{aligned}
$$

Variables: f, x_0, x_1; new formulation:

$$
\begin{aligned}
\max\ & \|g_1\|^2 \\
\text{s.t. } & \exists f \in \mathcal{F}_{\mu,L} : \ f_i = f(x_i),\ g_i = f'(x_i), \qquad \forall i \in \{0,1\} \\
& x_1 = x_0 - \gamma g_0 \\
& \|g_0\|^2 = R^2
\end{aligned}
$$

From infinite to finite dimensional problems

Performance estimation problem for a gradient step

$$\max \left\|f'(x_1)\right\|^2$$

$$\text{s.t. } f \in \mathcal{F}_{\mu,L} \qquad \text{Functional class}$$

$$x_1 = x_0 - \gamma f'(x_0) \qquad \text{Algorithm}$$

$$\left\|f'(x_0)\right\|^2 = R^2 \qquad \text{Initial condition}$$

Variables: f, x_0, x_1; new formulation:

$$\max \|g_1\|^2$$

$$\text{s.t. } \exists f \in \mathcal{F}_{\mu,L}: \; f_i = f(x_i), \; g_i = f'(x_i), \qquad \forall i \in \{0,1\}$$

$$x_1 = x_0 - \gamma g_0$$

$$\|g_0\|^2 = R^2$$

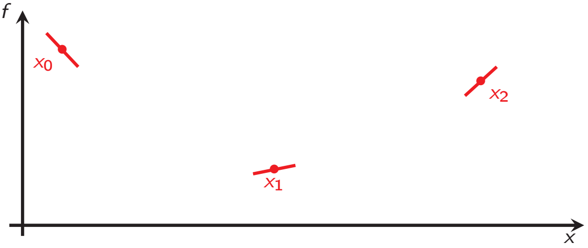
New variables: x_0, x_1, g_0, g_1, f_0, f_1.

Smooth strongly convex interpolation problem

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 problem

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.

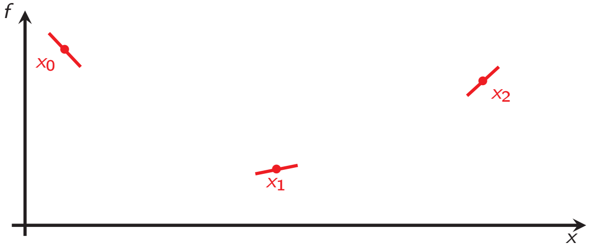


? Possible to find $f \in \mathcal{F}_{\mu,L}$ such that

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

Smooth strongly convex interpolation problem

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: $\forall i, j \in S$

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

Solving PEP via Semidefinite Programming

PEP subject to an existence constraint becomes

$$\begin{aligned} \max\ & \|g_1\|^2 \\ \text{s.t.}\ & \text{interpolation constraint } (i,j)\ \forall i,j \in \{0,1\} \\ & x_1 = x_0 - \gamma g_0 \\ & \|g_0\|^2 = R^2 \end{aligned}$$

under interpolation constraint:

Solving PEP via Semidefinite Programming

PEP subject to an existence constraint becomes

$$\begin{aligned} \max\ & \|g_1\|^2 \\ \text{s.t. }\ & \text{interpolation constraint } (i,j)\ \forall i,j \in \{0,1\} \\ & x_1 = x_0 - \gamma g_0 \\ & \|g_0\|^2 = R^2 \end{aligned}$$

under interpolation constraint:

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

Solving PEP via Semidefinite Programming

PEP subject to an existence constraint becomes

$$\begin{aligned}
\max\ & \|g_1\|^2 \\
\text{s.t. } & \text{interpolation constraint } (i,j)\ \forall i,j \in \{0,1\} \\
& x_1 = x_0 - \gamma g_0 \\
& \|g_0\|^2 = R^2
\end{aligned}$$

under interpolation constraint:

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

Non-convex quadratic program;

Solving PEP via Semidefinite Programming

PEP subject to an existence constraint becomes

$$
\begin{aligned}
\max\ & \|g_1\|^2 \\
\text{s.t. } & \text{interpolation constraint } (i,j)\ \forall i,j \in \{0,1\} \\
& x_1 = x_0 - \gamma g_0 \\
& \|g_0\|^2 = R^2
\end{aligned}
$$

under interpolation constraint:

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

Non-convex quadratic program; can be solved using semidefinite programming.

Semidefinite programming: lifting procedure

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d \times 4}$:

$$P = [x_0 \ x_1 \ g_0 \ g_1]$$

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d\times 4}$:

$$P = [x_0 \; x_1 \; g_0 \; g_1] \updownarrow d,$$

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d \times 4}$:

$$P = [x_0 \; x_1 \; g_0 \; g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d \times 4}$:

$$P = [x_0 \ x_1 \ g_0 \ g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

$$G = P^\top P = \begin{pmatrix} \langle x_0, x_0 \rangle & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle & \langle x_0, g_1 \rangle \\ \langle x_1, x_0 \rangle & \langle x_1, x_1 \rangle & \langle x_1, g_0 \rangle & \langle x_1, g_1 \rangle \\ \langle g_0, x_0 \rangle & \langle g_0, x_1 \rangle & \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle \\ \langle g_1, x_0 \rangle & \langle g_1, x_1 \rangle & \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle \end{pmatrix} \succeq 0.$$

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d\times 4}$:

$$P = [x_0\ x_1\ g_0\ g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

$$G = P^\top P = \begin{pmatrix} \langle x_0, x_0\rangle & \langle x_0, x_1\rangle & \langle x_0, g_0\rangle & \langle x_0, g_1\rangle \\ \langle x_1, x_0\rangle & \langle x_1, x_1\rangle & \langle x_1, g_0\rangle & \langle x_1, g_1\rangle \\ \langle g_0, x_0\rangle & \langle g_0, x_1\rangle & \langle g_0, g_0\rangle & \langle g_0, g_1\rangle \\ \langle g_1, x_0\rangle & \langle g_1, x_1\rangle & \langle g_1, g_0\rangle & \langle g_1, g_1\rangle \end{pmatrix} \succeq 0.$$

Problem from previous slide is linear in G.

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d \times 4}$:

$$P = [x_0 \; x_1 \; g_0 \; g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

$$G = P^\top P = \begin{pmatrix} \langle x_0, x_0 \rangle & \langle x_0, x_1 \rangle & \langle x_0, g_0 \rangle & \langle x_0, g_1 \rangle \\ \langle x_1, x_0 \rangle & \langle x_1, x_1 \rangle & \langle x_1, g_0 \rangle & \langle x_1, g_1 \rangle \\ \langle g_0, x_0 \rangle & \langle g_0, x_1 \rangle & \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle \\ \langle g_1, x_0 \rangle & \langle g_1, x_1 \rangle & \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle \end{pmatrix} \succeq 0.$$

Problem from previous slide is linear in G.

From $G \succeq 0$, we can recover x_0, x_1, g_0 and g_1 (Cholesky factorization):

$$G \succeq 0, \; \operatorname{rank} G \leq d \Leftrightarrow G = P^\top P \text{ with } P \in \mathbb{R}^{d \times 4}.$$

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d\times 4}$:

$$P = [x_0\ x_1\ g_0\ g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

$$G = P^\top P = \begin{pmatrix} \langle x_0, x_0\rangle & \langle x_0, x_1\rangle & \langle x_0, g_0\rangle & \langle x_0, g_1\rangle \\ \langle x_1, x_0\rangle & \langle x_1, x_1\rangle & \langle x_1, g_0\rangle & \langle x_1, g_1\rangle \\ \langle g_0, x_0\rangle & \langle g_0, x_1\rangle & \langle g_0, g_0\rangle & \langle g_0, g_1\rangle \\ \langle g_1, x_0\rangle & \langle g_1, x_1\rangle & \langle g_1, g_0\rangle & \langle g_1, g_1\rangle \end{pmatrix} \succeq 0.$$

Problem from previous slide is linear in G.

From $G \succeq 0$, we can recover x_0, x_1, g_0 and g_1 (Cholesky factorization):

$$G \succeq 0,\ \operatorname{rank} G \leq d \Leftrightarrow G = P^\top P \text{ with } P \in \mathbb{R}^{d\times 4}.$$

Dimension $d \Leftrightarrow \operatorname{rank} G \leq d$. Rank constraint disappears when $d \geq 4$.

Semidefinite programming: lifting procedure

Main difficulty: scalar products...

We stack all variables in a matrix $P \in \mathbb{R}^{d \times 4}$:

$$P = [x_0 \ x_1 \ g_0 \ g_1] \updownarrow d,$$

Introduces Gram matrix $G \succeq 0$, containing all scalar products:

$$G = P^\top P = \begin{pmatrix} \langle x_0, x_0\rangle & \langle x_0, x_1\rangle & \langle x_0, g_0\rangle & \langle x_0, g_1\rangle \\ \langle x_1, x_0\rangle & \langle x_1, x_1\rangle & \langle x_1, g_0\rangle & \langle x_1, g_1\rangle \\ \langle g_0, x_0\rangle & \langle g_0, x_1\rangle & \langle g_0, g_0\rangle & \langle g_0, g_1\rangle \\ \langle g_1, x_0\rangle & \langle g_1, x_1\rangle & \langle g_1, g_0\rangle & \langle g_1, g_1\rangle \end{pmatrix} \succeq 0.$$

Problem from previous slide is linear in G.

From $G \succeq 0$, we can recover x_0, x_1, g_0 and g_1 (Cholesky factorization):

$$G \succeq 0, \ \operatorname{rank} G \leq d \Leftrightarrow G = P^\top P \text{ with } P \in \mathbb{R}^{d \times 4}.$$

Dimension $d \Leftrightarrow \operatorname{rank} G \leq d$. Rank constraint disappears when $d \geq 4$.

SDP without rank constraint $\Leftrightarrow$ find smallest dimension-independent guarantee.

Numerical worst-case computation

Numerical worst-case computation

What can we do so far?

Numerical worst-case computation

What can we do so far?

- ◇ Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,

Numerical worst-case computation

What can we do so far?

- ◇ Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- ◇ encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,

Numerical worst-case computation

What can we do so far?

- ⋄ Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- ⋄ encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- ⋄ impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,

Numerical worst-case computation

What can we do so far?

- Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,
- set up objective: $\|f'(x_1)\|^2 = G_{4,4}$,

Numerical worst-case computation

What can we do so far?

- Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,
- set up objective: $\|f'(x_1)\|^2 = G_{4,4}$,
- your laptop can perform the worst-case analysis:

$$\begin{aligned} \max_{G\in\mathbb{S}^4, f\in\mathbb{R}^2} \; & G_{4,4} \\ \text{such that } f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) &\leq 0, \qquad i,j\in\{0,1\}, \\ G_{3,3} &= R^2, \\ G &\succeq 0. \end{aligned}$$

Numerical worst-case computation

What can we do so far?

- Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,
- set up objective: $\|f'(x_1)\|^2 = G_{4,4}$,
- your laptop can perform the worst-case analysis:

$$
\begin{aligned}
\max_{G \in \mathbb{S}^4, f \in \mathbb{R}^2} \quad & G_{4,4} \\
\text{such that } \quad & f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0, \qquad i,j \in \{0,1\}, \\
& G_{3,3} = R^2, \\
& G \succeq 0.
\end{aligned}
$$

Ex: $\mu = .1$, $L = 1$, $\gamma = .1$, $\|f'(x_0)\|^2 = 1$, d unspecified; then $\|f'(x_1)\|^2 \leq 0.96$.

Numerical worst-case computation

What can we do so far?

- ◇ Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- ◇ encode interpolation constraints as $f_j - f_i + \operatorname{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- ◇ impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,
- ◇ set up objective: $\|f'(x_1)\|^2 = G_{4,4}$,
- ◇ your laptop can perform the worst-case analysis:

$$
\begin{aligned}
\max_{G \in \mathbb{S}^4, f \in \mathbb{R}^2} \; & G_{4,4} \\
\text{such that } f_j - f_i + \operatorname{Tr}\left(GA_{ij}\right) &\leq 0, \qquad i,j \in \{0,1\}, \\
G_{3,3} &= R^2, \\
G &\succeq 0.
\end{aligned}
$$

Ex: $\mu = .1$, $L = 1$, $\gamma = .1$, $\|f'(x_0)\|^2 = 1$, d unspecified; then $\|f'(x_1)\|^2 \leq 0.96$.

Can we translate that into analytical guarantees?

Numerical worst-case computation

What can we do so far?

- ◇ Fix μ, L, R and γ, set up variables $\{f_i\}_i$ and G,
- ◇ encode interpolation constraints as $f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0$ for some A_{ij}'s,
- ◇ impose initial condition: $\|f'(x_0)\|^2 = G_{3,3} = R^2$,
- ◇ set up objective: $\|f'(x_1)\|^2 = G_{4,4}$,
- ◇ your laptop can perform the worst-case analysis:

$$\begin{aligned} \max_{G \in \mathbb{S}^4, f \in \mathbb{R}^2} \quad & G_{4,4} \\ \text{such that } f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0, & \qquad i, j \in \{0, 1\}, \\ G_{3,3} = R^2, & \\ G \succeq 0. & \end{aligned}$$

Ex: $\mu = .1$, $L = 1$, $\gamma = .1$, $\|f'(x_0)\|^2 = 1$, d unspecified; then $\|f'(x_1)\|^2 \leq 0.96$.

Can we translate that into analytical guarantees?

Yes, using Lagrange duality!

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$
\begin{aligned}
f_0 \geq f_1 \quad & + \langle f'(x_1), x_0 - x_1 \rangle + \tfrac{1}{2L} \| f'(x_0) - f'(x_1) \|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 && : \lambda_1 \\
f_1 \geq f_0 \quad & + \langle f'(x_0), x_1 - x_0 \rangle + \tfrac{1}{2L} \| f'(x_0) - f'(x_1) \|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 && : \lambda_2
\end{aligned}
$$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_1 \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_2
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$(1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 \geq \left\|f'(x_1)\right\|^2 + \underbrace{\frac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2},$$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 && : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$(1-\gamma\mu)^2\left\|f'(x_0)\right\|^2 \geq \left\|f'(x_1)\right\|^2 + \underbrace{\frac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geq 0},$$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \qquad : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \qquad : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$\begin{aligned}
(1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 &\geq \left\|f'(x_1)\right\|^2 + \underbrace{\frac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geq 0}, \\
&\geq \left\|f'(x_1)\right\|^2,
\end{aligned}$$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned} f_0 \geq f_1 \quad & + \langle f'(x_1), x_0 - x_1 \rangle + \tfrac{1}{2L} \|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 \end{aligned} \qquad : \lambda_1 = \frac{2}{\gamma}(1 - \mu\gamma)$$

$$\begin{aligned} f_1 \geq f_0 \quad & + \langle f'(x_0), x_1 - x_0 \rangle + \tfrac{1}{2L} \|f'(x_0) - f'(x_1)\|^2 \\ & + \tfrac{\mu}{2(1-\mu/L)} \left\| x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1)) \right\|^2 \end{aligned} \qquad : \lambda_2 = \frac{2}{\gamma}(1 - \mu\gamma)$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$\begin{aligned} (1 - \gamma\mu)^2 \left\| f'(x_0) \right\|^2 & \geq \left\| f'(x_1) \right\|^2 + \underbrace{\frac{2 - \gamma(L + \mu)}{\gamma(L - \mu)} \left\| (1 - \mu\gamma) f'(x_0) - f'(x_1) \right\|^2}_{\geq 0}, \\ & \geq \left\| f'(x_1) \right\|^2, \end{aligned}$$

leading to $\|f'(x_1)\|^2 \leq (1 - \frac{\mu}{L})^2 \|f'(x_0)\|^2$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad &+\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
&+\tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad &+\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
&+\tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 & : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$\begin{aligned}
(1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 &\geq \left\|f'(x_1)\right\|^2 + \underbrace{\frac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geq 0,\text{ or } = 0 \text{ when worst-case is achieved}}, \\
&\geq \left\|f'(x_1)\right\|^2,
\end{aligned}$$

leading to $\|f'(x_1)\|^2 \leq (1-\frac{\mu}{L})^2\|f'(x_0)\|^2$

Dual problem: find a proof

Gradient with $\gamma = \frac{1}{L}$: combine corresponding inequalities

$$\begin{aligned}
f_0 \geq f_1 \quad & +\langle f'(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \qquad : \lambda_1 = \tfrac{2}{\gamma}(1-\mu\gamma) \\
f_1 \geq f_0 \quad & +\langle f'(x_0), x_1 - x_0\rangle + \tfrac{1}{2L}\|f'(x_0) - f'(x_1)\|^2 \\
& + \tfrac{\mu}{2(1-\mu/L)}\left\|x_0 - x_1 - \tfrac{1}{L}(f'(x_0) - f'(x_1))\right\|^2 \qquad : \lambda_2 = \tfrac{2}{\gamma}(1-\mu\gamma)
\end{aligned}$$

Weighted sum with $\lambda_1, \lambda_2 \geq 0$ can be reformulated as

$$\begin{aligned}
(1-\gamma\mu)^2 \left\|f'(x_0)\right\|^2 & \geq \left\|f'(x_1)\right\|^2 + \underbrace{\frac{2-\gamma(L+\mu)}{\gamma(L-\mu)}\left\|(1-\mu\gamma)f'(x_0) - f'(x_1)\right\|^2}_{\geq 0,\ \text{or } = 0 \text{ when worst-case is achieved}}, \\
& \geq \left\|f'(x_1)\right\|^2,
\end{aligned}$$

leading to $\|f'(x_1)\|^2 \leq (1-\frac{\mu}{L})^2\|f'(x_0)\|^2$ (tight).

Toy example

Performance estimation

Further examples

Convex interpolation

Conclusions and discussions

Performance estimation problems

The approach we used for the gradient method can be used for a variety of methods.

Performance estimation problems

The approach we used for the gradient method can be used for a variety of methods.

Some attractive features of the approach:

Performance estimation problems

The approach we used for the gradient method can be used for a variety of methods.

Some attractive features of the approach:

- any primal solution is a lower bound (i.e., a function),

Performance estimation problems

The approach we used for the gradient method can be used for a variety of methods.

Some attractive features of the approach:

- any primal solution is a lower bound (i.e., a function),
- any dual solution is a worst-case guarantee (i.e., a proof),

Performance estimation problems

The approach we used for the gradient method can be used for a variety of methods.

Some attractive features of the approach:

- any primal solution is a lower bound (i.e., a function),
- any dual solution is a worst-case guarantee (i.e., a proof),
- it can be solved using semidefinite programming (SDP).

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

- different types of (smooth or non-smooth) convex functions,

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

- different types of (smooth or non-smooth) convex functions,
- convex indicator and support functions,

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

- different types of (smooth or non-smooth) convex functions,
- convex indicator and support functions,
- non-convex smooth functions,

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

- different types of (smooth or non-smooth) convex functions,
- convex indicator and support functions,
- non-convex smooth functions,
- any class whose interpolation conditions are SDP-representable.

Classes of problems

Constrained and regularized optimization problems can be handled, as well:

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

for different functional classes:

- different types of (smooth or non-smooth) convex functions,
- convex indicator and support functions,
- non-convex smooth functions,
- any class whose interpolation conditions are SDP-representable.

Also works for e.g., monotone inclusion problems.

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,
- inexact gradients methods,

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,
- inexact gradients methods,
- proximal point methods,
- projected and proximal gradients methods,

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,
- inexact gradients methods,
- proximal point methods,
- projected and proximal gradients methods,
- conditional gradient methods,
- splitting methods,
- randomized/stochastic gradient methods,
- distributed/decentralized gradient methods.

Algorithms

The approach can be used to obtain (tight) results for variety of “fixed-step” methods:

- (sub)gradient methods,
- inexact gradients methods,
- proximal point methods,
- projected and proximal gradients methods,
- conditional gradient methods,
- splitting methods,
- randomized/stochastic gradient methods,
- distributed/decentralized gradient methods.

Those includes fast/accelerated variants.

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,
- inexact gradients methods,
- proximal point methods,
- projected and proximal gradients methods,
- conditional gradient methods,
- splitting methods,
- randomized/stochastic gradient methods,
- distributed/decentralized gradient methods.

Those includes fast/accelerated variants.

The approach usually provides upper bounds (no a priori tightness) in other situations.

Algorithms

The approach can be used to obtain (tight) results for variety of "fixed-step" methods:

- (sub)gradient methods,
- inexact gradients methods,
- proximal point methods,
- projected and proximal gradients methods,
- conditional gradient methods,
- splitting methods,
- randomized/stochastic gradient methods,
- distributed/decentralized gradient methods.

Those includes fast/accelerated variants.

The approach usually provides upper bounds (no a priori tightness) in other situations.

In some settings, SDPs scale badly with problem parameters (e.g., stochastic settings).

Convergence measures

Different convergence measures can be taken into account.

Among others:

Convergence measures

Different convergence measures can be taken into account.

Among others:

- $f(x_N) - f(x_\star)$, $\|x_N - x_\star\|^2$, $\|f'(x_N)\|^2$,

Convergence measures

Different convergence measures can be taken into account.

Among others:

- $f(x_N) - f(x_\star)$, $\|x_N - x_\star\|^2$, $\|f'(x_N)\|^2$,
- best iterates on the way:

$$\min_{0\leq i\leq N} f(x_i) - f(x_\star), \quad \min_{0\leq i\leq N} \|x_i - x_\star\|^2, \quad \min_{0\leq i\leq N} \left\|f'(x_i)\right\|^2,$$

Convergence measures

Different convergence measures can be taken into account.

Among others:

- $f(x_N) - f(x_\star)$, $\|x_N - x_\star\|^2$, $\|f'(x_N)\|^2$,
- best iterates on the way:

$$\min_{0\leq i\leq N} f(x_i) - f(x_\star), \quad \min_{0\leq i\leq N} \|x_i - x_\star\|^2, \quad \min_{0\leq i\leq N} \left\|f'(x_i)\right\|^2,$$

- any concave function of f_i's, $\left\langle x_i, g_j \right\rangle$'s, $\|g_i\|^2$'s and $\|x_i\|^2$'s.

Want to give it a try?

Want to give it a try?

Performance EStimation TOolbox (PESTO)

Want to give it a try?

Performance EStimation TOolbox (PESTO)

Purpose: automated worst-case analyses of first-order methods without worrying about modelling steps.

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L}\nabla f(y_i)$$

$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L} \nabla f(y_i)$$
$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{\nabla} f(y_i) - \nabla f(y_i)\| \leq \varepsilon \|\nabla f(y_i)\|.$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L}\nabla f(y_i)$$

$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{\nabla} \mathbf{f}(\mathbf{y_i}) - \nabla f(y_i)\| \leq \varepsilon \|\nabla f(y_i)\|.$$

PESTO example: inexact fast gradient method

Minimize L-smooth convex function $f(x)$:

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

Fast Gradient Method (FGM)

Input: $f \in \mathcal{F}_{0,L}(\mathbb{R}^d)$, $x_0 = y_0 \in \mathbb{R}^d$.

For $i = 0 : N-1$

$$x_{i+1} = y_i - \frac{1}{L}\tilde{\nabla}\mathbf{f}(\mathbf{y_i})$$

$$y_{i+1} = x_{i+1} + \frac{i-1}{i+2}(x_{i+1} - x_i)$$

What if inexact gradient used instead? Relative inaccuracy model:

$$\|\tilde{\nabla}\mathbf{f}(\mathbf{y_i}) - \nabla f(y_i)\| \le \varepsilon \|\nabla f(y_i)\|.$$

PESTO example: inexact fast gradient method

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

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0       = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();         % xs is an optimal point, and fs=F(xs)
P.InitialCondition((x0-xs)^2 <= 1);  % Add an initial condition ||x0-xs||^2<= 1

% (3) Algorithm
N = 7; % number of iterations

x    = cell(N+1,1); % we store the iterates in a cell for convenience
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

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

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```

PESTO example: inexact fast gradient method

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

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0       = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();         % xs is an optimal point, and fs=F(xs)
```

```
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```

```
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

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

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```

PESTO example: inexact fast gradient method

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

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0      = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();        % xs is an optimal point, and fs=F(xs)
```

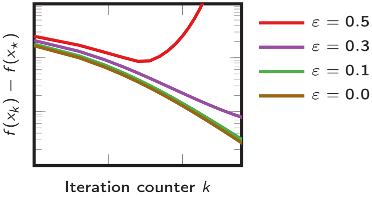
```
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```

```
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

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

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```



PESTO example: inexact fast gradient method

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

% (1) Set up the objective function
param.mu = 0;      % strong convexity parameter
param.L  = 1;      % Smoothness parameter

F=P.DeclareFunction('SmoothStronglyConvex',param); % F is the objective function

% (2) Set up the starting point and initial condition
x0       = P.StartingPoint();        % x0 is some starting point
[xs, fs] = F.OptimalPoint();         % xs is an optimal point, and fs=F(xs)
```

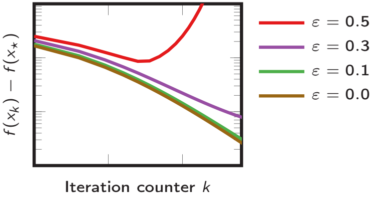
```matlab
x{1} = x0;
y    = x0;
eps  = .1;
for i = 1:N
    d      = inexactsubgradient(y, F, eps);
    x{i+1} = y - 1/param.L * d;
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end
```

```matlab
    y      = x{i+1} + (i-1)/(i+2) * (x{i+1} - x{i});
end

% (4) Set up the performance measure
[g, f] = F.oracle(x{N+1});    % g=grad F(x), f=F(x)
P.PerformanceMetric(f - fs); % Worst-case evaluated as F(x)-F(xs)

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

% (6) Evaluate the output
double(f - fs)   % worst-case objective function accuracy
```



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

Toy example

Performance estimation

Further examples

Convex interpolation

Conclusions and discussions

Steepest descent with inexact search directions

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

with $f \in \mathcal{F}_{\mu,L}$ (L-smooth μ-strongly convex).

Steepest descent with inexact search directions

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

with $f \in \mathcal{F}_{\mu,L}$ (L-smooth μ-strongly convex).

Relative error model:

$$\|\nabla f(x_i) - d_i\| \leq \varepsilon \|\nabla f(x_i)\| \quad i = 0, 1, \ldots, \tag{1}$$

Steepest descent with inexact search directions

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

with $f \in \mathcal{F}_{\mu,L}$ (L-smooth μ-strongly convex).

Relative error model:

$$\|\nabla f(x_i) - d_i\| \leq \varepsilon \|\nabla f(x_i)\| \quad i = 0, 1, \ldots, \tag{1}$$

Noisy gradient descent method with exact line search

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^n)$, $x_0 \in \mathbb{R}^n$, $0 \leq \varepsilon < 1$.

for $i = 0, 1, \ldots$

Select any seach direction d_i that satisfies (1);

$\gamma = \text{argmin}_{\gamma \in \mathbb{R}} f\left(x_i - \gamma d_i\right)$

$x_{i+1} = x_i - \gamma d_i$

Steepest descent with inexact search directions

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

with $f \in \mathcal{F}_{\mu,L}$ (L-smooth μ-strongly convex).

Relative error model:

$$\|\nabla f(x_i) - d_i\| \leq \varepsilon \|\nabla f(x_i)\| \quad i = 0, 1, \ldots, \tag{1}$$

Noisy gradient descent method with exact line search

Input: $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^n)$, $x_0 \in \mathbb{R}^n$, $0 \leq \varepsilon < 1$.

for $i = 0, 1, \ldots$

Select any seach direction d_i that satisfies (1);

$\gamma = \text{argmin}_{\gamma \in \mathbb{R}} f(x_i - \gamma d_i)$

$x_{i+1} = x_i - \gamma d_i$

Worst-case behavior: (de Klerk, Glineur, T. 2017)

$$f(x_{i+1}) - f_* \leq \left(\frac{1 - \kappa_\varepsilon}{1 + \kappa_\varepsilon} \right)^2 (f(x_i) - f_*) \quad i = 0, 1, \ldots$$

where $\kappa_\varepsilon = \frac{\mu}{L} \frac{(1-\varepsilon)}{(1+\varepsilon)}$.

Steepest descent with inexact search directions

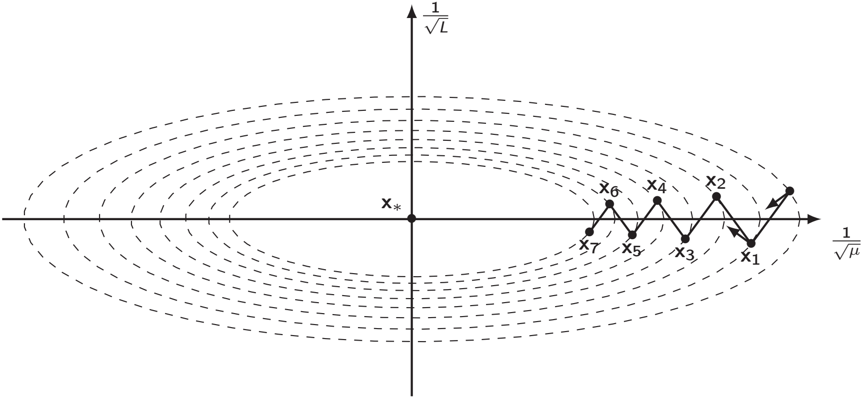
Quadratic worst-case function:

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \lambda_i x_i^2 \quad \text{where} \quad 0 < \mu = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_n = L.$$

Steepest descent with inexact search directions

Quadratic worst-case function:

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \lambda_i x_i^2 \quad \text{where} \quad 0 < \mu = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n = L.$$



What does the proof look like?

Aggregate the constraints

What does the proof look like?

Aggregate the constraints

$$f_0 \geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{1}{2L}\|g_0 - g_1\|^2 + \frac{\mu}{2\left(1 - \frac{\mu}{L}\right)}\|x_0 - x_1 - (g_0 - g_1)/L\|^2$$

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

$$f_\star \geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{1}{2L}\|g_\star - g_1\|^2 + \frac{\mu}{2\left(1 - \frac{\mu}{L}\right)}\|x_\star - x_1 - (g_\star - g_1)/L\|^2$$

$$0 = \langle g_0, g_1 \rangle$$

$$0 = \langle g_1, x_1 - x_0 \rangle$$

What does the proof look like?

Aggregate the constraints

$$
\begin{aligned}
f_0 &\geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{1}{2L}\|g_0 - g_1\|^2 + \frac{\mu}{2\left(1 - \frac{\mu}{L}\right)}\|x_0 - x_1 - (g_0 - g_1)/L\|^2 \\
f_\star &\geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{1}{2L}\|g_\star - g_0\|^2 + \frac{\mu}{2\left(1 - \frac{\mu}{L}\right)}\|x_\star - x_0 - (g_\star - g_0)/L\|^2 \\
f_\star &\geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{1}{2L}\|g_\star - g_1\|^2 + \frac{\mu}{2\left(1 - \frac{\mu}{L}\right)}\|x_\star - x_1 - (g_\star - g_1)/L\|^2 \\
0 &= \langle g_0, g_1 \rangle \\
0 &= \langle g_1, x_1 - x_0 \rangle
\end{aligned}
$$

with multipliers

What does the proof look like?

Aggregate the constraints

$$
\begin{aligned}
f_0 &\geq f_1 + \langle g_1, x_0 - x_1 \rangle + \frac{1}{2L}\|g_0 - g_1\|^2 + \frac{\mu}{2\left(1-\frac{\mu}{L}\right)}\|x_0 - x_1 - (g_0 - g_1)/L\|^2 \\
f_\star &\geq f_0 + \langle g_0, x_\star - x_0 \rangle + \frac{1}{2L}\|g_\star - g_0\|^2 + \frac{\mu}{2\left(1-\frac{\mu}{L}\right)}\|x_\star - x_0 - (g_\star - g_0)/L\|^2 \\
f_\star &\geq f_1 + \langle g_1, x_\star - x_1 \rangle + \frac{1}{2L}\|g_\star - g_1\|^2 + \frac{\mu}{2\left(1-\frac{\mu}{L}\right)}\|x_\star - x_1 - (g_\star - g_1)/L\|^2 \\
0 &= \langle g_0, g_1 \rangle \\
0 &= \langle g_1, x_1 - x_0 \rangle
\end{aligned}
$$

with multipliers

$$
y_1 = \frac{L-\mu}{L+\mu}, \quad y_2 = 2\mu\frac{(L-\mu)}{(L+\mu)^2}, \quad y_3 = \frac{2\mu}{L+\mu}, \quad y_4 = \frac{2}{L+\mu}, \quad y_5 = 1.
$$

What does the proof look like?

Resulting inequality:

$$\begin{aligned} f_1 - f_\star \quad \leq \quad & \left(\tfrac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star) \\ & - \tfrac{\mu L(L+3\mu)}{2(L+\mu)^2} \left\| x_0 - \tfrac{L+\mu}{L+3\mu} x_1 - \tfrac{2\mu}{L+3\mu} x_\star - \tfrac{3L+\mu}{L^2+3\mu L} g_0 - \tfrac{L+\mu}{L^2+3\mu L} g_1 \right\|^2 \\ & - \tfrac{2L\mu^2}{L^2+2L\mu-3\mu^2} \left\| x_1 - x_\star - \tfrac{(L-\mu)^2}{2\mu L(L+\mu)} g_0 - \tfrac{L+\mu}{2\mu L} g_1 \right\|^2. \end{aligned}$$

What does the proof look like?

Resulting inequality:

$$\begin{aligned} f_1 - f_\star \quad \leq \quad & \left(\tfrac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star) \\ & - \tfrac{\mu L(L+3\mu)}{2(L+\mu)^2} \left\| x_0 - \tfrac{L+\mu}{L+3\mu} x_1 - \tfrac{2\mu}{L+3\mu} x_\star - \tfrac{3L+\mu}{L^2+3\mu L} g_0 - \tfrac{L+\mu}{L^2+3\mu L} g_1 \right\|^2 \\ & - \tfrac{2L\mu^2}{L^2+2L\mu-3\mu^2} \left\| x_1 - x_\star - \tfrac{(L-\mu)^2}{2\mu L(L+\mu)} g_0 - \tfrac{L+\mu}{2\mu L} g_1 \right\|^2. \end{aligned}$$

Last two terms nonpositive, so

$$f_1 - f_\star \leq \left(\frac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star).$$

What does the proof look like?

Resulting inequality:

$$
\begin{aligned}
f_1 - f_\star \quad \leq \quad & \left(\tfrac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star) \\
& - \tfrac{\mu L(L+3\mu)}{2(L+\mu)^2} \left\| x_0 - \tfrac{L+\mu}{L+3\mu} x_1 - \tfrac{2\mu}{L+3\mu} x_\star - \tfrac{3L+\mu}{L^2+3\mu L} g_0 - \tfrac{L+\mu}{L^2+3\mu L} g_1 \right\|^2 \\
& - \tfrac{2L\mu^2}{L^2+2L\mu-3\mu^2} \left\| x_1 - x_\star - \tfrac{(L-\mu)^2}{2\mu L(L+\mu)} g_0 - \tfrac{L+\mu}{2\mu L} g_1 \right\|^2.
\end{aligned}
$$

Last two terms nonpositive, so

$$
f_1 - f_\star \leq \left(\frac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star).
$$

One actually has equality at optimality, due to the quadratic example.

Optimized gradient methods

Smooth convex minimization setting:

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

with f being L-smooth and convex, with black-box oracle $f'(.)$ available.

Optimized gradient methods

Smooth convex minimization setting:

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

with f being L-smooth and convex, with black-box oracle $f'(.)$ available.

Lower bound for large-scale setting ($d \geq N+2$) by Drori (2017):

$$f(x_N) - f(x_\star) \geq \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2},$$

with $\theta_0 = 1$, and:

$$\theta_{i+1} = \begin{cases} \frac{1+\sqrt{4\theta_i^2+1}}{2} & \text{if } i \leq N-2, \\ \frac{1+\sqrt{8\theta_i^2+1}}{2} & \text{if } i = N-1. \end{cases}$$

Optimized gradient methods

Smooth convex minimization setting:

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

with f being L-smooth and convex, with black-box oracle $f'(.)$ available.

Lower bound for large-scale setting ($d \geq N+2$) by Drori (2017):

$$f(x_N) - f(x_\star) \geq \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2} = O(N^{-2}),$$

with $\theta_0 = 1$, and:

$$\theta_{i+1} = \begin{cases} \frac{1+\sqrt{4\theta_i^2+1}}{2} & \text{if } i \leq N-2, \\ \frac{1+\sqrt{8\theta_i^2+1}}{2} & \text{if } i = N-1. \end{cases}$$

Optimized gradient methods

Smooth convex minimization setting:

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

with f being L-smooth and convex, with black-box oracle $f'(.)$ available.

Lower bound for large-scale setting ($d \geq N+2$) by Drori (2017):

$$f(x_N) - f(x_\star) \geq \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2} = O(N^{-2}),$$

with $\theta_0 = 1$, and:

$$\theta_{i+1} = \begin{cases} \frac{1+\sqrt{4\theta_i^2+1}}{2} & \text{if } i \leq N-2, \\ \frac{1+\sqrt{8\theta_i^2+1}}{2} & \text{if } i = N-1. \end{cases}$$

Coherent with historical lower bounds (Nemirovski & Yudin 1983) and optimal methods (Nemirovski 1982), (Nesterov 1983).

Optimized gradient methods

Three methods with the same (optimal) worst-case behavior

Greedy First-order Method (GFOM)

Inputs: f, x_0, N.

For $i = 1, 2, \ldots$

$$x_i = \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ f(x) : \; x \in x_0 + \mathrm{span}\{f'(x_0), \ldots, f'(x_{i-1})\} \right\}.$$

Worst-case guarantee:

$$f(x_N) - f(x_\star) \le \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2}.$$

See (Drori & T. 2018).

Optimized gradient methods

Three methods with the same (optimal) worst-case behavior

Optimized gradient method with exact line-search

Inputs: f, x_0, N.

For $i = 1, \ldots, N$

$$y_i = \left(1 - \frac{1}{\theta_i}\right) x_{i-1} + \frac{1}{\theta_i} x_0$$

$$d_i = \left(1 - \frac{1}{\theta_i}\right) f'(x_{i-1}) + \frac{1}{\theta_i} \left(2 \sum_{j=0}^{i-1} \theta_j f'(x_j)\right)$$

$$\alpha = \operatorname*{argmin}_{\alpha \in \mathbb{R}} f(y_i + \alpha d_i)$$

$$x_i = y_i + \alpha d_i$$

Worst-case guarantee:

$$f(x_N) - f(x_\star) \leq \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2}.$$

See (Drori & T. 2018).

Optimized gradient methods

Three methods with the same (optimal) worst-case behavior

Optimized gradient method

Inputs: f, x_0, N.

For $i = 1, \ldots, N$

$$y_i = x_{i-1} - \frac{1}{L} f'(x_{i-1})$$

$$z_i = x_0 - \frac{2}{L} \sum_{j=0}^{i-1} \theta_j f'(x_j)$$

$$x_i = \left(1 - \frac{1}{\theta_i}\right) y_i + \frac{1}{\theta_i} z_i$$

Worst-case guarantee:

$$f(x_N) - f(x_\star) \leq \frac{L\|x_0 - x_\star\|^2}{2\theta_N^2}.$$

See (Drori & Teboulle 2014), (Kim & Fessler 2016), (Drori & T. 2018).

A few more examples

Warning for the next few slides:

A few more examples

Warning for the next few slides:

- ◇ the expressions are horrible,

A few more examples

Warning for the next few slides:

- ◇ the expressions are horrible,
- ◇ barely obtainable by hand,

A few more examples

Warning for the next few slides:

- ◇ the expressions are horrible,
- ◇ barely obtainable by hand,
- ◇ computer-generated (Mathematica or Matlab),

A few more examples

Warning for the next few slides:

- ⋄ the expressions are horrible,
- ⋄ barely obtainable by hand,
- ⋄ computer-generated (Mathematica or Matlab),
- ⋄ verifiable by hand (possibly long algebraic proofs).

A few more examples

Warning for the next few slides:

- the expressions are horrible,
- barely obtainable by hand,
- computer-generated (Mathematica or Matlab),
- verifiable by hand (possibly long algebraic proofs).

Intuitions can be developed, but this is another story ☺

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\operatorname*{find}_{x \in \mathbb{R}^d} \; 0 \in A(x) + B(x).$$

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\underset{x \in \mathbb{R}^d}{\text{find}} \; 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\operatorname*{find}_{x \in \mathbb{R}^d} \ 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Reformulate problem via fixed-point of some $T : \mathbb{R}^d \to \mathbb{R}^d$. One particular choice:

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\underset{x \in \mathbb{R}^d}{\text{find}} \; 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Reformulate problem via fixed-point of some $T : \mathbb{R}^d \to \mathbb{R}^d$. One particular choice:

- ◇ let $J_A = (I + A)^{-1}$ and $J_B = (I + B)^{-1}$ being resolvents of A and B,

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\operatorname*{find}_{x \in \mathbb{R}^d} \; 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Reformulate problem via fixed-point of some $T : \mathbb{R}^d \to \mathbb{R}^d$. One particular choice:

- ◇ let $J_A = (I + A)^{-1}$ and $J_B = (I + B)^{-1}$ being resolvents of A and B,
- ◇ let $T = I - \theta J_B + \theta J_A(2J_B - I)$ (overrelaxed Douglas-Rachford operator).

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\operatorname*{find}_{x \in \mathbb{R}^d} \; 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Reformulate problem via fixed-point of some $T : \mathbb{R}^d \to \mathbb{R}^d$. One particular choice:

- ◇ let $J_A = (I + A)^{-1}$ and $J_B = (I + B)^{-1}$ being resolvents of A and B,
- ◇ let $T = I - \theta J_B + \theta J_A(2J_B - I)$ (overrelaxed Douglas-Rachford operator).

Example: Douglas-Rachford for solving

$$\min_{x \in \mathbb{R}^d} f(x) + h(x),$$

Douglas-Rachford Splitting

Let $A : \mathbb{R}^d \to 2^{\mathbb{R}^d}$, $B : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ (point-to-set maps) be maximally monotone;

$$\operatorname*{find}_{x\in\mathbb{R}^d} \; 0 \in A(x) + B(x).$$

Examples: $A(x) = \partial f(x)$ and $B(x) = \partial h(x)$ for two convex functions f and h.

Reformulate problem via fixed-point of some $T : \mathbb{R}^d \to \mathbb{R}^d$. One particular choice:

- ⋄ let $J_A = (I + A)^{-1}$ and $J_B = (I + B)^{-1}$ being resolvents of A and B,
- ⋄ let $T = I - \theta J_B + \theta J_A(2J_B - I)$ (overrelaxed Douglas-Rachford operator).

Example: Douglas-Rachford for solving

$$\min_{x\in\mathbb{R}^d} f(x) + h(x),$$

amount to iterate:

$$\begin{aligned}
x_{k+1} &= \operatorname{argmin}_{x\in\mathbb{R}^d}\{h(x) + \tfrac{1}{2}\|x - w_k\|^2\}\\
y_{k+1} &= \operatorname{argmin}_{y\in\mathbb{R}^d}\{f(y) + \tfrac{1}{2}\|y - 2x_{k+1} + w_k\|^2\}\\
w_{k+1} &= w_k + \theta(y_{k+1} - x_{k+1})
\end{aligned}$$

Douglas-Rachford Splitting

Question: When is this T a contraction? What is the smallest ρ such that

$$\|Tx - Ty\| \leq \rho \|x - y\|,$$

for all $x, y \in \mathbb{R}^d$?

Douglas-Rachford Splitting

Question: When is this T a contraction? What is the smallest ρ such that

$$\|Tx - Ty\| \leq \rho \|x - y\|,$$

for all $x, y \in \mathbb{R}^d$?

Related previous works:

(Douglas & Rachford 1956), (Lions & Mercier 1979), (Giselsson & Boyd 2017), (Giselsson 2017), (Davis & Yin 2017), (Moursi & Vandenberghe 2018), and many others; gentle introduction to monotone operators (Ryu & Boyd 2016).

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \le \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right.$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho \|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta \frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ & \\ & \end{cases}$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta\frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ \left|1 - \theta\frac{1+\mu\beta}{(\mu+1)(\beta+1)}\right| & \text{if } \mu\beta - \mu - \beta > 0, \text{ and } \theta \leq 2\frac{\mu^2+\beta^2+\mu\beta+\mu+\beta-\mu^2\beta^2}{\mu^2+\beta^2+\mu^2\beta+\mu\beta^2+\mu+\beta-2\mu^2\beta^2}, \\ & \\ & \end{cases}$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta\frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ \left|1 - \theta\frac{1+\mu\beta}{(\mu+1)(\beta+1)}\right| & \text{if } \mu\beta - \mu - \beta > 0, \text{ and } \theta \leq 2\frac{\mu^2+\beta^2+\mu\beta+\mu+\beta-\mu^2\beta^2}{\mu^2+\beta^2+\mu^2\beta+\mu\beta^2+\mu+\beta-2\mu^2\beta^2}, \\ |1 - \theta| & \text{if } \theta \geq 2\frac{\mu\beta+\mu+\beta}{2\mu\beta+\mu+\beta}, \\ & \end{cases}$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta\frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ \left|1 - \theta\frac{1+\mu\beta}{(\mu+1)(\beta+1)}\right| & \text{if } \mu\beta - \mu - \beta > 0, \text{ and } \theta \leq 2\frac{\mu^2+\beta^2+\mu\beta+\mu+\beta-\mu^2\beta^2}{\mu^2+\beta^2+\mu^2\beta+\mu\beta^2+\mu+\beta-2\mu^2\beta^2}, \\ |1 - \theta| & \text{if } \theta \geq 2\frac{\mu\beta+\mu+\beta}{2\mu\beta+\mu+\beta}, \\ \left|1 - \theta\frac{\mu}{\mu+1}\right| & \text{if } \mu\beta + \mu - \beta < 0, \text{ and } \theta \leq 2\frac{(\mu+1)(\beta-\mu-\mu\beta)}{\beta+\mu\beta-\mu-\mu^2-2\mu^2\beta}, \end{cases}$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta\frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ \left|1 - \theta\frac{1+\mu\beta}{(\mu+1)(\beta+1)}\right| & \text{if } \mu\beta - \mu - \beta > 0, \text{ and } \theta \leq 2\frac{\mu^2+\beta^2+\mu\beta+\mu+\beta-\mu^2\beta^2}{\mu^2+\beta^2+\mu^2\beta+\mu\beta^2+\mu+\beta-2\mu^2\beta^2}, \\ |1 - \theta| & \text{if } \theta \geq 2\frac{\mu\beta+\mu+\beta}{2\mu\beta+\mu+\beta}, \\ \left|1 - \theta\frac{\mu}{\mu+1}\right| & \text{if } \mu\beta + \mu - \beta < 0, \text{ and } \theta \leq 2\frac{(\mu+1)(\beta-\mu-\mu\beta)}{\beta+\mu\beta-\mu-\mu^2-2\mu^2\beta}, \\ X & \text{otherwise,} \end{cases}$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B β-cocoercive.

We have $\|Tx - Ty\| \leq \rho\|x-y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \left|1 - \theta\frac{\beta}{\beta+1}\right| & \text{if } \mu\beta - \mu + \beta < 0, \text{ and } \theta \leq 2\frac{(\beta+1)(\mu-\beta-\mu\beta)}{\mu+\mu\beta-\beta-\beta^2-2\mu\beta^2}, \\ \left|1 - \theta\frac{1+\mu\beta}{(\mu+1)(\beta+1)}\right| & \text{if } \mu\beta - \mu - \beta > 0, \text{ and } \theta \leq 2\frac{\mu^2+\beta^2+\mu\beta+\mu+\beta-\mu^2\beta^2}{\mu^2+\beta^2+\mu^2\beta+\mu\beta^2+\mu+\beta-2\mu^2\beta^2}, \\ |1-\theta| & \text{if } \theta \geq 2\frac{\mu\beta+\mu+\beta}{2\mu\beta+\mu+\beta}, \\ \left|1 - \theta\frac{\mu}{\mu+1}\right| & \text{if } \mu\beta + \mu - \beta < 0, \text{ and } \theta \leq 2\frac{(\mu+1)(\beta-\mu-\mu\beta)}{\beta+\mu\beta-\mu-\mu^2-2\mu^2\beta}, \\ X & \text{otherwise,} \end{cases}$$

with

$$X = \frac{\sqrt{2-\theta}}{2}\sqrt{\frac{\left((2-\theta)\mu(\beta+1)-\theta\beta(\mu-1)\right)\left((2-\theta)\beta(\mu+1)-\theta\mu(\beta-1)\right)}{(2-\theta)\mu\beta(\mu+1)(\beta+1)-\theta\mu^2\beta^2}}.$$

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B L-Lipschitz and monotone.

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B L-Lipschitz and monotone.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

Douglas-Rachford Splitting

Assumptions: A μ-strongly monotone, B L-Lipschitz and monotone.

We have $\|Tx - Ty\| \leq \rho\|x - y\|$ for all $x, y \in \mathcal{H}$ with:

$$\rho = \begin{cases} \dfrac{\theta+\sqrt{\frac{(2(\theta-1)\mu+\theta-2)^2+L^2(\theta-2(\mu+1))^2}{L^2+1}}}{2(\mu+1)} & \text{if } (a), \\ \left|1 - \theta\frac{L+\mu}{(\mu+1)(L+1)}\right| & \text{if } (b), \\ \sqrt{\frac{(2-\theta)}{4\mu(L^2+1)}\frac{\left(\theta(L^2+1)-2\mu(\theta+L^2-1)\right)\left(\theta\left(1+2\mu+L^2\right)-2(\mu+1)\left(L^2+1\right)\right)}{2\mu(\theta+L^2-1)-(2-\theta)(1-L^2)}} & \text{otherwise,} \end{cases}$$

with

(a) $\mu\frac{-(2(\theta-1)\mu+\theta-2)+L^2(\theta-2(1+\mu))}{\sqrt{(2(\theta-1)\mu+\theta-2)^2+L^2(\theta-2(\mu+1))^2}} \leq \sqrt{L^2+1}$,

(b) $L < 1$, $\mu > \frac{L^2+1}{(L-1)^2}$, and $\theta \leq \frac{2(\mu+1)(L+1)(\mu+\mu L^2-L^2-2\mu L-1)}{2\mu^2-\mu+\mu L^3-L^3-3\mu L^2-L^2-2\mu^2 L-\mu L-L-1}$.

Toy example

Performance estimation

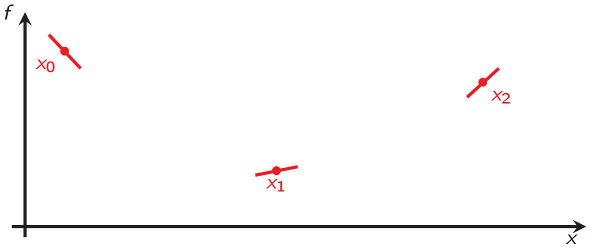
Further examples

Convex interpolation

Conclusions and discussions

Smooth strongly convex interpolation

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.

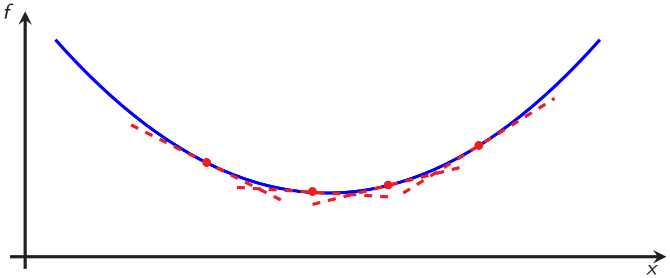
$$f(x_i) = f_i, \quad \text{and} \quad g_i \in \partial f(x_i), \qquad \forall i \in S.$$

Special case: convex interpolation

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) ?

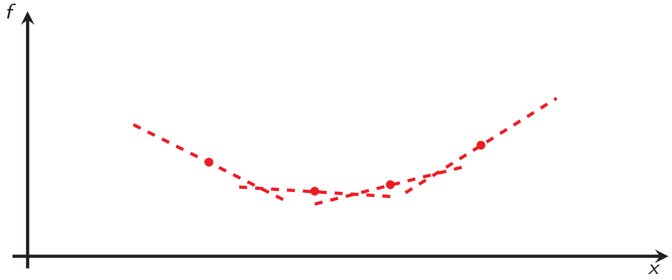
Special case: convex interpolation

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

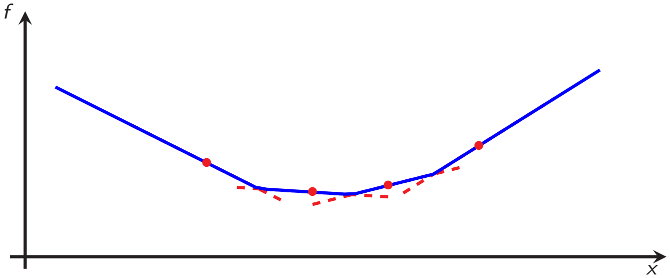
Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



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

Special case: convex interpolation

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:

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

Not unique.

Smooth convex interpolation

Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

Smooth convex interpolation

Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

Counter-example 1: what about the conditions:

$$\begin{aligned} & f_i \geq f_j + \langle g_j, x_i - x_j \rangle, && i, j \in S, \\ & \|g_i - g_j\| \leq L \|x_i - x_j\|. && \end{aligned} \tag{C1}$$

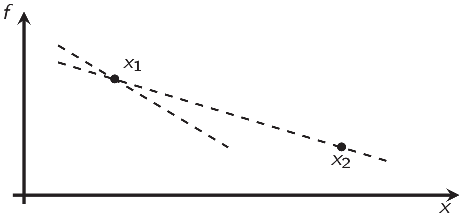
Smooth convex interpolation

Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

Counter-example 1: what about the conditions:

$$\begin{aligned} f_i &\geq f_j + \langle g_j, x_i - x_j \rangle, \qquad\qquad i,j \in S, \\ ||g_i - g_j|| &\leq L||x_i - x_j||. \end{aligned} \tag{C1}$$

$(x_1, g_1, f_1) = (-1, -2, 1)$
$(x_2, g_2, f_2) = (0, -1, 0)$



satisfies (C1) but cannot be differentiable...

An approach to smooth convex interpolation

Idea: reduce smooth convex interpolation to convex interpolation.

Basic operations needed in order to transform the problem:

- Conjugation: f is closed, proper and convex, then:
  f L-Lipschitz gradient $\Leftrightarrow f^*$ $\frac{1}{L}$-strongly convex.

- Minimal curvature subtraction:
  $f(x)$ μ-strongly convex $\Leftrightarrow f(x) - \frac{\mu}{2}\|x\|^2$ convex.

Conjugation (1): Definition

Consider a proper function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, the (Legendre-Fenchel) conjugate of f is defined as:

$$f^*(y) = \sup_{x \in \mathbb{R}^d} \langle y, x \rangle - f(x),$$

with $f^* \in \mathcal{F}_{\mathbf{0},\infty}$ (proper, closed and convex).

Conjugation (2): Useful properties

For $f \in \mathcal{F}_{0,\infty}$, we have a one-to-one correspondence between f and f^*, and the following propositions are equivalent:

(a) $f(x) + f^*(g) = \langle g, x \rangle$,

(b) $g \in \partial f(x)$,

(c) $x \in \partial f^*(g)$.

Conjugation (2): Useful properties

For $f \in \mathcal{F}_{0,\infty}$, we have a one-to-one correspondence between f and f^*, and the following propositions are equivalent:

(a) $f(x) + f^*(g) = \langle g, x \rangle$,

(b) $g \in \partial f(x)$,

(c) $x \in \partial f^*(g)$.

For $f \in \mathcal{F}_{0,\infty}$, we have: $f \in \mathcal{F}_{0,L} \Leftrightarrow f^* \in \mathcal{F}_{1/L,\infty}$.

Conjugation (2): Useful properties

For $f \in \mathcal{F}_{0,\infty}$, we have a one-to-one correspondence between f and f^*, and the following propositions are equivalent:

(a) $f(x) + f^*(g) = \langle g, x \rangle$,

(b) $g \in \partial f(x)$,

(c) $x \in \partial f^*(g)$.

For $f \in \mathcal{F}_{0,\infty}$, we have: $f \in \mathcal{F}_{0,L} \Leftrightarrow f^* \in \mathcal{F}_{1/L,\infty}$.

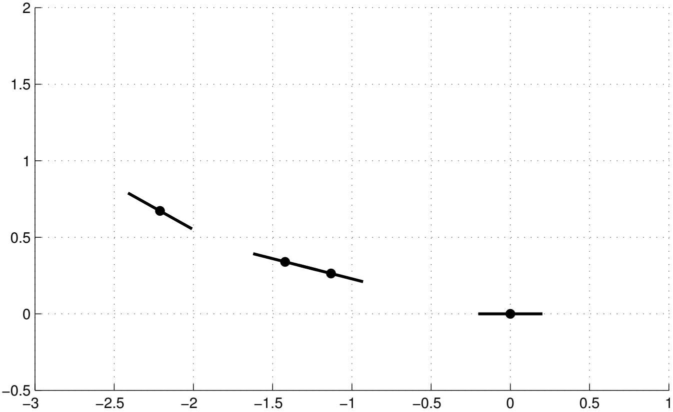
Intuition:

- upper bounds become lower bounds; let $f, u \in \mathcal{F}_{0,\infty}$, we have:

$$f(x) \leq u(x) \text{ for all } x \in \mathbb{R}^d \Leftrightarrow u^*(g) \leq f^*(g) \text{ for all } g \in \mathbb{R}^d.$$

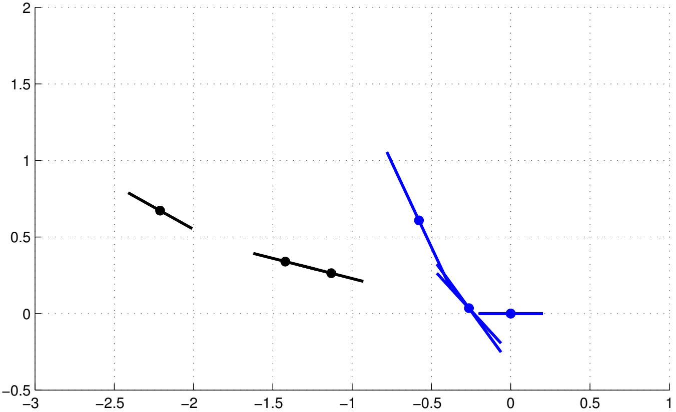
- Conjugate of quadratics are quadratics.

Example: Smooth Convex Interpolation



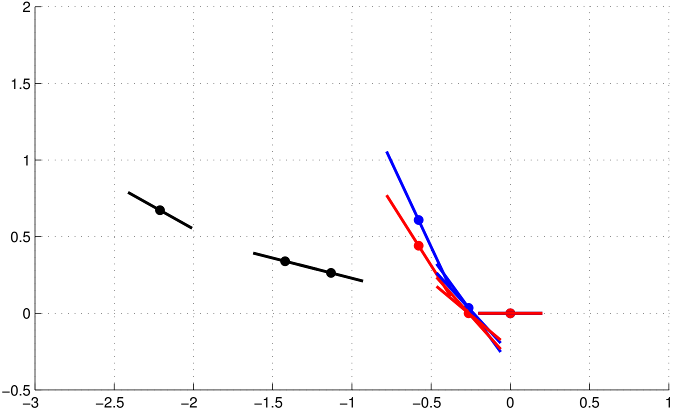
Interpolate $\{(x_i, g_i, f_i)\}_{i \in S}$ by $f \in \mathcal{F}_{0,L}$

Example: Smooth Convex Interpolation



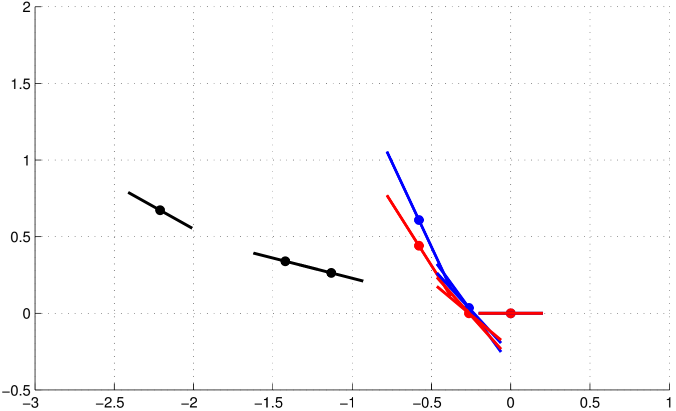
$\Leftrightarrow$ interpolate $\{(g_i, x_i, \langle x_i, g_i\rangle - f_i)\}_{i\in S}$ by $f^* \in \mathcal{F}_{1/L,\infty}$.

Example: Smooth Convex Interpolation



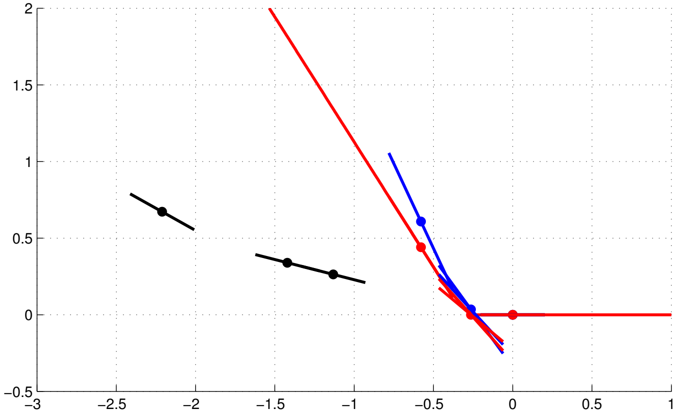
$\Leftrightarrow$ interpolate $\left\{\left(g_i, x_i - \frac{g_i}{L}, \langle x_i, g_i\rangle - f_i - \frac{\|g_i\|^2}{2L}\right)\right\}_{i\in S}$ by $\tilde{f} \in \mathcal{F}_{0,\infty}$.

Example: Smooth Convex Interpolation



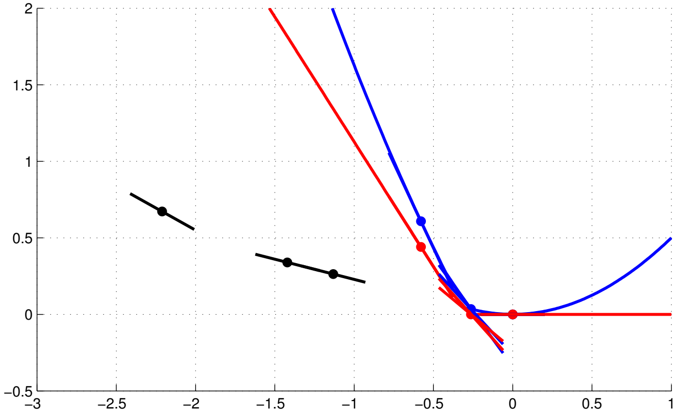
$\Leftrightarrow$ interpolate $\left\{(\tilde{x}_i, \tilde{g}_i, \tilde{f}_i)\right\}_{i\in S}$ by $\tilde{f} \in \mathcal{F}_{0,\infty}$.

Example: Smooth Convex Interpolation



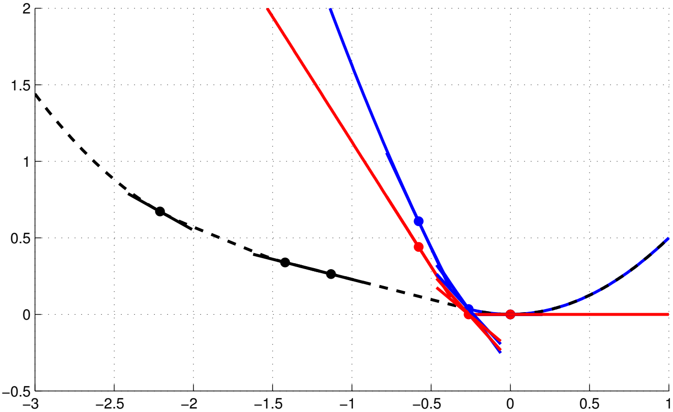
$\tilde{f}(x) = \max_j \left\{ \tilde{f}_j + \langle \tilde{g}_j, x - \tilde{x}_j \rangle \right\}$

Example: Smooth Convex Interpolation



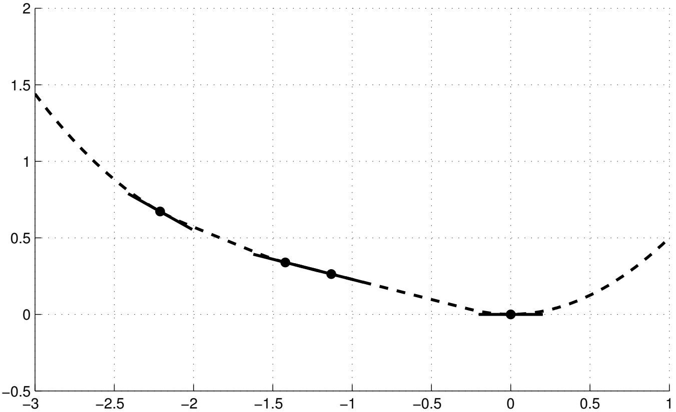
$f^*(x) = \max_j \left\{ \tilde{f}_j + \langle \tilde{g}_j, x - \tilde{x}_j \rangle \right\} + \frac{\|x\|^2}{2L}$

Example: Smooth Convex Interpolation



$$f(x) = \left(\max_j \left\{\tilde{f}_j + \langle \tilde{g}_j, x - \tilde{x}_j \rangle\right\} + \frac{\|x\|^2}{2L}\right)^*$$

Example: Smooth Convex Interpolation



$$f(x) = \left(\max_j \left\{\tilde{f}_j + \langle \tilde{g}_j, x - \tilde{x}_j \rangle\right\} + \frac{\|x\|^2}{2L}\right)^*$$

Conclusion: iff conditions

Using the same reasoning:

The set $\{(x_i, g_i, f_i)\}_{i\in S}$ is interpolable by a function $f \in \mathcal{F}_{\mu,L}$ (proper, closed, μ-strongly convex with L-Lipschitz gradient) iff:

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

When $\mu = 0$, those conditions transforms to the well-known

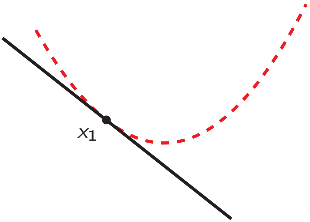
$$f_j \geq f_i + \langle g_i, x_j - x_i\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \qquad \forall i, j \in S.$$

Interpretation: compatible upper and lower bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \| g_i - g_j \|^2 \qquad \forall i, j \in S$$

characterize compatibility between upper and lower bounds.

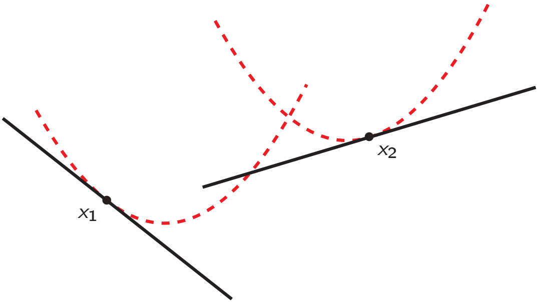


Interpretation: compatible upper and lower bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \|g_i - g_j\|^2 \qquad \forall i, j \in S$$

characterize compatibility between upper and lower bounds.



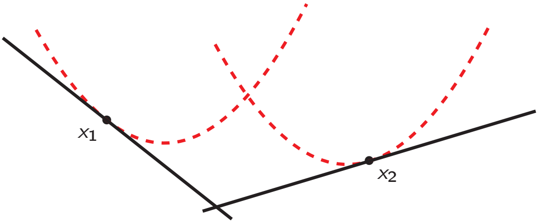
x_1 and x_2 are not compatible.

Interpretation: compatible upper and lower bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \| g_i - g_j \|^2 \qquad \forall i, j \in S$$

characterize compatibility between upper and lower bounds.



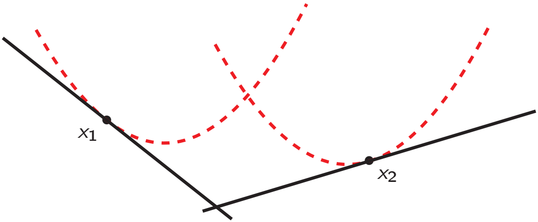
x_1 and x_2 are compatible.

Interpretation: convex hull of upper bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \|g_i - g_j\|^2 \qquad \forall i, j \in S$$

ensure you can perform the following interpolation procedure:

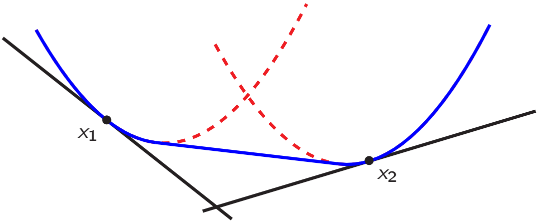


Interpretation: convex hull of upper bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \|g_i - g_j\|^2 \qquad \forall i, j \in S$$

ensure you can perform the following interpolation procedure:

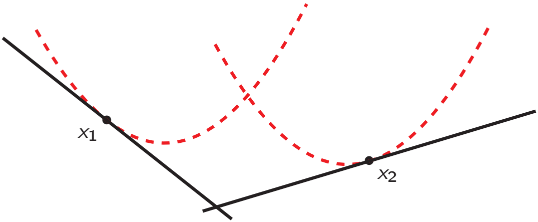


Interpretation: smoothed lower bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \|g_i - g_j\|^2 \qquad \forall i, j \in S$$

ensure you can perform the following interpolation procedure:

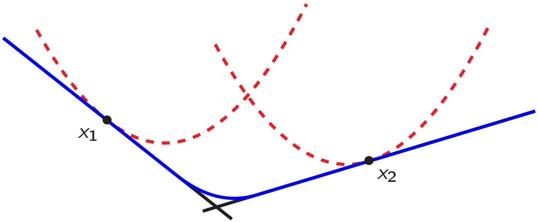


Interpretation: smoothed lower bounds

Smooth convex interpolation conditions

$$f_j \geq f_i + \langle g_i, x_j - x_i \rangle + \frac{1}{2L} \|g_i - g_j\|^2 \qquad \forall i, j \in S$$

ensure you can perform the following interpolation procedure:



Some opinions on PEPs

Pros/cons of PEPs

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.

- ☹ SDPs typically become prohibitively large (with N and generalizations),

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),
- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.
- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),
- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.
- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

  examples in (Drori & Teboulle 2014), (Drori 2014), (Kim & Fessler 2016 2018), (Shi & Liu 2017), (de Klerk et al. 2017), etc.

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.

- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

  examples in (Drori & Teboulle 2014), (Drori 2014), (Kim & Fessler 2016 2018), (Shi & Liu 2017), (de Klerk et al. 2017), etc.

- ☹ proofs (may be) hard to generalize (e.g., to handle projections, backtracking),

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.

- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

  examples in (Drori & Teboulle 2014), (Drori 2014), (Kim & Fessler 2016 2018), (Shi & Liu 2017), (de Klerk et al. 2017), etc.

- ☹ proofs (may be) hard to generalize (e.g., to handle projections, backtracking),

  examples in (Kim & Fessler 2016 2018).

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),

- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.

- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

  examples in (Drori & Teboulle 2014), (Drori 2014), (Kim & Fessler 2016 2018), (Shi & Liu 2017), (de Klerk et al. 2017), etc.

- ☹ proofs (may be) hard to generalize (e.g., to handle projections, backtracking),

  examples in (Kim & Fessler 2016 2018).

- ☺ allows reaching proofs that could barely be obtained by hand,

Some opinions on PEPs

Pros/cons of PEPs

- ☺ Worst-case guarantees *cannot be improved*,

  details in (T, Hendrickx & Glineur 2017),
- ☺ fair amount of generalizations (finite sums, constraints, prox, etc.),

  details in (T, Hendrickx & Glineur 2017); (Drori 2014), etc.
- ☹ SDPs typically become prohibitively large (with N and generalizations),
- ☹ proofs (may be) quite involved and hard to intuit,

  examples in (Drori & Teboulle 2014), (Drori 2014), (Kim & Fessler 2016 2018), (Shi & Liu 2017), (de Klerk et al. 2017), etc.
- ☹ proofs (may be) hard to generalize (e.g., to handle projections, backtracking),

  examples in (Kim & Fessler 2016 2018).
- ☺ allows reaching proofs that could barely be obtained by hand,
- ☺ easy to try via Performance EStimation TOolbox (PESTO).

Concluding remarks

Performance estimation's philosophy

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),

Concluding remarks

Performance estimation's philosophy

- ◇ numerically allows to obtain tight bounds (rigorous baselines),
- ◇ helps designing analytical proofs (reduces to linear combinations of inequalities),

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

  before trying to prove your crazy-algorithm works; give PEP a try!

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

  before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

- Second-order methods?
- Interpolation and formulations with non-Euclidean geometries?
- How to generically end up with optimal schemes and lower-bounds?
- Beyond worst-case analyses?
- And many others...

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

- Second-order methods?
- Interpolation and formulations with non-Euclidean geometries?
- How to generically end up with optimal schemes and lower-bounds?
- Beyond worst-case analyses?
- And many others...

Three recent works:

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

  before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

- Second-order methods?
- Interpolation and formulations with non-Euclidean geometries?
- How to generically end up with optimal schemes and lower-bounds?
- Beyond worst-case analyses?
- And many others...

Three recent works:

- *"Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions"* (COLT 2019, with F. Bach),

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

  before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

- Second-order methods?
- Interpolation and formulations with non-Euclidean geometries?
- How to generically end up with optimal schemes and lower-bounds?
- Beyond worst-case analyses?
- And many others...

Three recent works:

- *"Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions"* (COLT 2019, with F. Bach),
- *"Operator Splitting Performance Estimation: Tight contraction factors and optimal parameter selection"* (2018, with E. Ryu, C. Bergeling and P. Giselsson),

Concluding remarks

Performance estimation's philosophy

- numerically allows to obtain tight bounds (rigorous baselines),
- helps designing analytical proofs (reduces to linear combinations of inequalities),
- fast prototyping:

  before trying to prove your crazy-algorithm works; give PEP a try!

Open questions:

- Second-order methods?
- Interpolation and formulations with non-Euclidean geometries?
- How to generically end up with optimal schemes and lower-bounds?
- Beyond worst-case analyses?
- And many others...

Three recent works:

- *"Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions"* (COLT 2019, with F. Bach),
- *"Operator Splitting Performance Estimation: Tight contraction factors and optimal parameter selection"* (2018, with E. Ryu, C. Bergeling and P. Giselsson),
- *"Efficient first-order methods for convex minimization: a constructive approach"* (MP 2019, with Y. Drori).

Take-home messages

Worst-cases are solutions to optimization problems.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Take-home messages

Worst-cases are solutions to optimization problems.

Sometimes, those optimization problems are tractable.

Often tractable for first-order methods in convex optimization!

Thanks! Questions?

www.di.ens.fr/∼ataylor/

AdrienTaylor/Performance-Estimation-Toolbox on Github

References I

[1] N. Bansal and A. Gupta.
Potential-function proofs for first-order methods.
preprint arXiv:1712.04581, 2017.

[2] A. Beck and M. Teboulle.
A fast iterative shrinkage-thresholding algorithm for linear inverse problems.
SIAM Journal on Imaging Sciences, 2(1):183–202, 2009.

[3] D. Davis and W. Yin.
A three-operator splitting scheme and its optimization applications.
Set-Valued and Variational Analysis, 25(4):829–858, 2017.

[4] E. de Klerk, F. Glineur, and A. B. Taylor.
On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions.
Optimization Letters, 11(7):1185–1199, 2017.

[5] J. Douglas and H. H. Rachford.
On the numerical solution of heat conduction problems in two and three space variables.
Transactions of the American Mathematical Society, 82:421–439, 1956.

[6] Y. Drori.
Contributions to the Complexity Analysis of Optimization Algorithms.
PhD thesis, Tel-Aviv University, 2014.

[7] Y. Drori.
The exact information-based complexity of smooth convex minimization.
Journal of Complexity, 39:1–16, 2017.

References II

[8] Y. Drori and A. B. Taylor.
Efficient first-order methods for convex minimization: a constructive approach.
Mathematical Programming (to appear), 2019.

[9] Y. Drori and M. Teboulle.
Performance of first-order methods for smooth convex minimization: a novel approach.
Mathematical Programming, 145(1-2):451–482, 2014.

[10] P. Giselsson.
Tight global linear convergence rate bounds for Douglas-Rachford splitting.
Journal of Fixed Point Theory and Applications, 19(4):2241–2270, 2017.

[11] P. Giselsson and S. Boyd.
Diagonal scaling in Douglas-Rachford splitting and ADMM.
In *53rd IEEE Conference on Decision and Control*, pages 5033–5039, Los Angeles, CA, Dec. 2014.

[12] P. Giselsson and S. Boyd.
Linear convergence and metric selection for Douglas-Rachford splitting and ADMM.
IEEE Transactions on Automatic Control, 62(2):532–544, 2017.

[13] D. Kim and J. A. Fessler.
Optimized first-order methods for smooth convex minimization.
Mathematical Programming, 159(1-2):81–107, 2016.

[14] D. Kim and J. A. Fessler.
Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions.
preprint arXiv:1803.06600, 2018.

References III

[15] L. Lessard, B. Recht, and A. Packard.
Analysis and design of optimization algorithms via integral quadratic constraints.
SIAM Journal on Optimization, 26(1):57–95, 2016.

[16] P. L. Lions and B. Mercier.
Splitting algorithms for the sum of two nonlinear operators.
SIAM Journal on Numerical Analysis, 16(6):964–979, 1979.

[17] W. M. Moursi and L. Vandenberghe.
Douglas-Rachford splitting for a Lipschitz continuous and a strongly monotone operator.
arXiv preprint arXiv:1805.09396, 2018.

[18] A. S. Nemirovski.
Orth-method for smooth convex optimization.
Izvestia AN SSSR, Tekhnicheskaya Kibernetika, 2:937–947, 1982.

[19] A. S. Nemirovski and D. B. Yudin.
Problem complexity and method efficiency in optimization.
Willey-Interscience, New York, 1983.

[20] Y. Nesterov.
A method of solving a convex programming problem with convergence rate $O(1/k^2)$.
Soviet Mathematics Doklady, 27:372–376, 1983.

[21] Y. Nesterov.
Introductory Lectures on Convex Optimization : a Basic Course.
Applied optimization. Kluwer Academic Publishing, 2004.

References IV

[22] Y. Nesterov.
Lectures on Convex Optimization.
Springer Optimization and Its Applications. Springer International Publishing, 2018.

[23] E. K. Ryu and S. Boyd.
Primer on monotone operator methods.
Appl. Comput. Math., 15:3–43, 2016.

[24] E. K. Ryu, A. B. Taylor, C. Bergeling, and P. Giselsson.
Operator splitting performance estimation: Tight contraction factors and optimal parameter selection.
preprint arXiv:1812.00146, 2018.

[25] A. B. Taylor and F. Bach.
Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions.
Conference on Learning Theory (COLT), 2019.

[26] A. B. Taylor, J. M. Hendrickx, and F. Glineur.
Exact worst-case performance of first-order methods for composite convex optimization.
SIAM Journal on Optimization, 27(3):1283–1313, 2017.

[27] A. B. Taylor, J. M. Hendrickx, and F. Glineur.
Performance Estimation Toolbox (PESTO): automated worst-case analysis of first-order optimization methods.
In *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1278–1283, 2017.

References V

[28] A. B. Taylor, J. M. Hendrickx, and F. Glineur.
Smooth strongly convex interpolation and exact worst-case performance of first-order methods.
Mathematical Programming, 161(1-2):307–345, 2017.

[29] A. B. Taylor, J. M. Hendrickx, and F. Glineur.
Exact worst-case convergence rates of the proximal gradient method for composite convex minimization.
Journal of Optimization Theory and Applications, 178(2):455–476, 2018.

[30] A. C. Wilson, B. Recht, and M. I. Jordan.
A Lyapunov analysis of momentum methods in optimization.
preprint arXiv:1611.02635, 2016.