Performance estimation of first-order methods for composite convex optimization

François Glineur
Université catholique de Louvain (UCLouvain)

Center for Operations Research and Econometrics and *Information and Communication Technologies, Electronics and Applied Mathematics Institute*

joint work with
Adrien B.M. Taylor and Julien M. Hendrickx

Phi-TAB workshop, September 29th 2016 Telecom ParisTech

A motivating example

Consider the standard projected gradient method for constrained smoooth minimization:

$$\min_{x \in Q} f(x)$$

with

- f convex, smooth with L-Lipschitz gradient
- $Q \subseteq \mathbb{R}^d$ closed convex set

For $i = 0 : N - 1$

$$x_{i+1} = \Pi_Q \left(x_i - \frac{1}{L} \nabla f(x_i) \right)$$

where Π_Q is projection on set Q

Exact convergence rate of projected gradient

$$\min_{x \in Q} f(x) \quad \text{from } x_0 \text{ with } x_{i+1} = \Pi_Q \left(x_i - \frac{1}{L} \nabla f(x_i) \right)$$

It is well known (and proved countless references) that

$$f(x_N) - f(x_*) \leq \frac{L \|x_0 - x_*\|^2}{2N}$$

where x_* is an optimal solution

However this is *not tight*!
For example one has that x_1 actually satisfies

$$f(x_1) - f(x_*) \leq \frac{L \|x_0 - x_*\|^2}{4}$$

(twice better!)

Bouding the error after one gradient step

Convex functions are known to satisfy

$$f(x) \geq f(y) + \nabla f(y)(x - y) \text{ for all } x, y$$

Moreover smooth convex functions actually verify condition $SC(x, y)$

$$f(x) \geq f(y) + \nabla f(y)(x - y) + \frac{\|\nabla f(x) - \nabla f(y)\|^2}{2L} \text{ for all } x, y$$

To simplify we only consider unconstrained case
(but result also holds for projected gradient)

Proof simply relies on writing this condition for three pairs (x, y)

$$(x, y) \in \{(x_0, x_1), (x_*, x_0), (x_*, x_1)\}$$

Proof of the exact convergence rate of projected gradient

$$f(x) \geq f(y) + \nabla f(y)(x - y) + \frac{\|\nabla f(x) - \nabla f(y)\|^2}{2L} \text{ for all } x, y$$

We write

- $(x, y) = (x_0, x_1)$

$$f(x_0) \geq f(x_1) + \nabla f(x_1)(x_0 - x_1) + \frac{\|\nabla f(x_0) - \nabla f(x_1)\|^2}{2L}$$

Proof of the exact convergence rate of projected gradient

$$f(x) \geq f(y) + \nabla f(y)(x - y) + \frac{\|\nabla f(x) - \nabla f(y)\|^2}{2L} \text{ for all } x, y$$

We write

- $(x, y) = (x_0, x_1)$

$$f(x_0) \geq f(x_1) + \nabla f(x_1)(x_0 - x_1) + \frac{\|\nabla f(x_0) - \nabla f(x_1)\|^2}{2L}$$

- $(x, y) = (x_*, x_0)$

$$f(x_*) \geq f(x_0) + \nabla f(x_0)(x_* - x_0) + \frac{\|\nabla f(x_0)\|^2}{2L}$$

Proof of the exact convergence rate of projected gradient

$$f(x) \geq f(y) + \nabla f(y)(x - y) + \frac{\|\nabla f(x) - \nabla f(y)\|^2}{2L} \text{ for all } x, y$$

We write

- $(x, y) = (x_0, x_1)$

$$f(x_0) \geq f(x_1) + \nabla f(x_1)(x_0 - x_1) + \frac{\|\nabla f(x_0) - \nabla f(x_1)\|^2}{2L}$$

- $(x, y) = (x_*, x_0)$

$$f(x_*) \geq f(x_0) + \nabla f(x_0)(x_* - x_0) + \frac{\|\nabla f(x_0)\|^2}{2L}$$

- $(x, y) = (x_*, x_1)$

$$f(x_*) \geq f(x_1) + \nabla f(x_1)(x_* - x_1) + \frac{\|\nabla f(x_1)\|^2}{2L}$$

Proof of the exact convergence rate of projected gradient

$$f(x) \geq f(y) + \nabla f(y)(x - y) + \frac{\|\nabla f(x) - \nabla f(y)\|^2}{2L} \text{ for all } x, y$$

We write

- $(x, y) = (x_0, x_1)$

$$f(x_0) \geq f(x_1) + \nabla f(x_1)(x_0 - x_1) + \frac{\|\nabla f(x_0) - \nabla f(x_1)\|^2}{2L}$$

- $(x, y) = (x_*, x_0)$

$$f(x_*) \geq f(x_0) + \nabla f(x_0)(x_* - x_0) + \frac{\|\nabla f(x_0)\|^2}{2L}$$

- $(x, y) = (x_*, x_1)$

$$f(x_*) \geq f(x_1) + \nabla f(x_1)(x_* - x_1) + \frac{\|\nabla f(x_1)\|^2}{2L}$$

and sum these three inequalities

Proof of the exact convergence rate of gradient

We obtain

$$2f(x_*) \geq 2f(x_1) + \Big(\nabla f(x_1) - \nabla f(x_0)\Big)^T (x_0 - x_*) - 2\nabla f(x_1)^T (x_1 - x_*)$$
$$+ \frac{1}{L}\Big(\|\nabla f(x_0)\|^2 + \|\nabla f(x_1)\|^2 - \nabla f(x_1)^T \nabla f(x_0)\Big)$$

Proof of the exact convergence rate of gradient

We obtain

$$2f(x_*) \geq 2f(x_1) + \Big(\nabla f(x_1) - \nabla f(x_0)\Big)^T (x_0 - x_*) - 2\nabla f(x_1)^T (x_1 - x_*) + \frac{1}{L}\Big(\|\nabla f(x_0)\|^2 + \|\nabla f(x_1)\|^2 - \nabla f(x_1)^T \nabla f(x_0)\Big)$$

which can be rewritten as

$$f(x_1) - f(x_*) \leq -\frac{1}{2}\Big(\nabla f(x_1) - \nabla f(x_0)\Big)^T (x_0 - x_*) + \nabla f(x_1)^T (x_1 - x_*) - \frac{1}{2L}\Big(\|\nabla f(x_0)\|^2 + \|\nabla f(x_1)\|^2 - \nabla f(x_1)^T \nabla f(x_0)\Big)$$

Proof of the exact convergence rate of gradient

Now use definition of the method $x_1 = x_0 - \frac{g_0}{L}$ and let $g_0 = \nabla f(x_0)$, $g_1 = \nabla f(x_1)$ and $d = x_0 - x_*$

Proof of the exact convergence rate of gradient

Now use definition of the method $x_1 = x_0 - \frac{g_0}{L}$ and let $g_0 = \nabla f(x_0)$, $g_1 = \nabla f(x_1)$ and $d = x_0 - x_*$

Our upper bound on $f(x_1) - f(x_*)$

$$
\begin{aligned}
&- \frac{1}{2}\Big(\nabla f(x_1) - \nabla f(x_0)\Big)^T (x_0 - x_*) + \nabla f(x_1)^T (x_1 - x_*) \\
&- \frac{1}{2L}\Big(\|\nabla f(x_0)\|^2 + \|\nabla f(x_1)\|^2 - \nabla f(x_1)^T \nabla f(x_0)\Big)
\end{aligned}
$$

becomes

$$
\begin{aligned}
&- \frac{1}{2}(g_1 - g_0)^T d + g_1^T (d - \frac{g_0}{L}) - \frac{1}{2L}\Big(\|g_0\|^2 + \|g_1\|^2 - g_1^T g_0\Big) \\
&= \frac{1}{2}(g0 + g1)^T d - \frac{1}{2L}\Big(\|g_0\|^2 + \|g_1\|^2 + g_1^T g_0\Big) \\
&= \frac{1}{4}L\|d\|^2 - \frac{1}{4}L\Big\|d - \frac{g0}{L} - \frac{g1}{L}\Big\|^2 - \frac{1}{2L}\Big\|\frac{g0}{L}\Big\|^2 - \frac{1}{2L}\Big\|\frac{g1}{L}\Big\|^2
\end{aligned}
$$

Proof of the exact convergence rate of gradient

We have obtained

$$f(x_1)-f(x_*) \leq \frac{1}{4}L\|d\|^2-\frac{1}{4}L\left\|d-\frac{g0}{L}-\frac{g1}{L}\right\|^2-\frac{1}{2L}\left\|\frac{g0}{L}\right\|^2-\frac{1}{2L}\left\|\frac{g1}{L}\right\|^2$$

which establishes the claim

$$f(x_1) - f(x_*) \leq \frac{1}{4}L\|d\|^2 = \frac{L\|x_0 - x_*\|^2}{4}$$

□

Goal of this talk is to introducing and describe a systematic technique to find this type of proof, called

performance estimation

Performance Estimation: take-home messages

- Worst-case behaviour of first-order methods can be computed exactly using semidefinite optimization

Performance Estimation: take-home messages

- Worst-case behaviour of first-order methods can be computed exactly using semidefinite optimization

- For any fixed-coefficient first-order method after any given number of iterations on the class of smooth convex objective with given parameters (smoothness and strong convexity)

Performance Estimation: take-home messages

- Worst-case behaviour of first-order methods can be computed exactly using semidefinite optimization

- For any fixed-coefficient first-order method
after any given number of iterations
on the class of smooth convex objective with given parameters
(smoothness and strong convexity)

- Methodology provides easy-to-check (but not very intuitive) proofs and explicit examples of worst-case functions

Performance Estimation: take-home messages

- Worst-case behaviour of first-order methods can be computed exactly using semidefinite optimization
- For any fixed-coefficient first-order method after any given number of iterations on the class of smooth convex objective with given parameters (smoothness and strong convexity)
- Methodology provides easy-to-check (but not very intuitive) proofs and explicit examples of worst-case functions
- Very flexible: choice of performance criteria (e.g. objective value, gradient norm); can be extended to constrained, nonsmooth, proximal, linear minimization oracle settings

Outline

Goal

Study methods for smooth unconstrained convex minimization

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

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.
- Compute exact worst-case guarantees after N iterations

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.
- Compute exact worst-case guarantees after N iterations
- Performance criteria include for example

over all functions $f \in \mathcal{F}_{\mu,L}$ and starting points x_0

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.
- Compute exact worst-case guarantees after N iterations
- Performance criteria include for example
  - maximum possible value of $f(x_N) - f^*$

over all functions $f \in \mathcal{F}_{\mu,L}$ and starting points x_0

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.
- Compute exact worst-case guarantees after N iterations
- Performance criteria include for example
  - maximum possible value of $f(x_N) - f^*$
  - maximum possible distance of $\|x_N - x_*\|$

over all functions $f \in \mathcal{F}_{\mu,L}$ and starting points x_0

Goal

Study methods for smooth unconstrained convex minimization

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

- function $f \in \mathcal{F}_{\mu,L}$: μ-strongly convex, L-Lipschitz gradient,
- method: first-order, oracle based, fixed-step coefficients (ability to compute f and ∇f given a x)

- Methods include fixed-step gradient, fast gradient (Nesterov acceleration), heavy ball method, etc.
- Compute exact worst-case guarantees after N iterations
- Performance criteria include for example
  - maximum possible value of $f(x_N) - f^*$
  - maximum possible distance of $\|x_N - x_*\|$
  - maximum possible value of $\max_{0 \leq k \leq N} \|\nabla f(x_k)\|$

over all functions $f \in \mathcal{F}_{\mu,L}$ and starting points x_0

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$
- method $\mathcal{M}$ defined for a number of iterations N

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$
- method $\mathcal{M}$ defined for a number of iterations N
- bound on the initial distance to the solution R

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$
- method $\mathcal{M}$ defined for a number of iterations N
- bound on the initial distance to the solution R
- performance criterion $\mathcal{P}$

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$
- method $\mathcal{M}$ defined for a number of iterations N
- bound on the initial distance to the solution R
- performance criterion $\mathcal{P}$

we want to evalutate worst-case performance $w(\mathcal{F}, R, \mathcal{M}, N, \mathcal{P})$ defined as

$$\sup_{\mathbf{f},\mathbf{x_0},\ldots,\mathbf{x_N},\mathbf{x}_*} \quad \mathcal{P}(\mathcal{O}_f, x_0, \ldots, x_N, x_*) \qquad \text{(PEP)}$$

$$\text{s.t. } f \in \mathcal{F}, x_* \text{ is optimal for } f, \|x_0 - x_*\|_2 \leq R$$

$$x_1, \ldots, x_N \text{ is generated by method } \mathcal{M} \text{ starting from } x_0,$$

Variable f is a function, and infinite-dimensional
No explicit constraint on dimension of domain of function f

Performance estimation problem

Formally, for a given

- problem class $\mathcal{F}$ whose members are equipped with oracle $\mathcal{O}_f$
- method $\mathcal{M}$ defined for a number of iterations N
- bound on the initial distance to the solution R
- performance criterion $\mathcal{P}$

we want to evalutate worst-case performance $w(\mathcal{F}, R, \mathcal{M}, N, \mathcal{P})$ defined as

$$\sup_{\mathbf{f},\mathbf{x_0},\ldots,\mathbf{x_N},\mathbf{x}_*} \quad \mathcal{P}(\mathcal{O}_f, x_0, \ldots, x_N, x_*) \qquad \text{(PEP)}$$

$$\text{s.t. } f \in \mathcal{F}, x_* \text{ is optimal for } f, \|x_0 - x_*\|_2 \leq R$$

$$x_1, \ldots, x_N \text{ is generated by method } \mathcal{M} \text{ starting from } x_0,$$

Variable f is a function, and infinite-dimensional
No explicit constraint on dimension of domain of function f

Solve an optimization problem (PEP)
to estimate performance of an optimization algorithm

A black-box method

First N iterates generated by a first-order black-box method $\mathcal{M}$ (N calls of the oracle), starting from initial x_0 are

$$\begin{aligned} x_1 &= \mathcal{M}_1\left(x_0, \mathcal{O}_f(x_0)\right), \\ x_2 &= \mathcal{M}_2\left(x_0, \mathcal{O}_f(x_0), \mathcal{O}_f(x_1)\right), \\ &\vdots \\ x_N &= \mathcal{M}_N\left(x_0, \mathcal{O}_f(x_0), \ldots, \mathcal{O}_f(x_{N-1})\right). \end{aligned}$$

Only depends on x_0 and the finite list of outputs from the oracle

$\rightarrow$ (PEP) can be reformulated as a finite-dimensional problem

i.e. replace (infnite-dimensional) functional variable $f \in \mathcal{F}$
by a list of N (finite-dimensional) oracle outputs $\mathcal{O}_f(x_k)$

A finite-dimensional reformulation

Since method is oracle based, (PEP) can be reformulated in a finite way using only only iterates $\{x_i\}_{i\in I}$, their function values $\{f_i\}_{i\in I}$ and their gradients $\{g_i\}_{i\in I}$ as

$$\sup_{\{\mathbf{x_i},\mathbf{g_i},\mathbf{f_i}\}_{\mathbf{i}\in\mathbf{I}}} \quad \mathcal{P}\left(\{x_i, g_i, f_i\}_{i\in I}\right), \qquad \text{(f-PEP)}$$

$$\begin{aligned} \text{s.t. } & \text{there exists } f \in \mathcal{F} \text{ such that } \mathcal{O}_f(x_i) = \{f_i, g_i\}\ \forall i \in I, \\ & x_1, \ldots, x_N \text{ is generated by method } \mathcal{M} \text{ from } x_0, \\ & \|x_0 - x_*\|_2 \leq R, \\ & g_* = 0 \end{aligned}$$

Crucial part is the first constraint, which says that $\{x_i, g_i, f_i\}_{i\in I}$ can be interpolated on $\mathcal{F} \rightarrow$ find an equivalent tractable condition for smooth and strongly convex functions

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)
- Nonconvex formulation proposed by Drori and Teboulle $\rightarrow$ they solve a convex relaxation $\rightarrow$ only provides bounds, but shown to be tight for some algorithms (using matching worst-case functions)

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)
- Nonconvex formulation proposed by Drori and Teboulle $\rightarrow$ they solve a convex relaxation $\rightarrow$ only provides bounds, but shown to be tight for some algorithms (using matching worst-case functions)
- Kim and Fessler (2014) identify an optimized first-order methods (optimal coefficients for DT relaxation), shown to be optimal for smooth convex optimization by Drori (2016)

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)
- Nonconvex formulation proposed by Drori and Teboulle $\rightarrow$ they solve a convex relaxation $\rightarrow$ only provides bounds, but shown to be tight for some algorithms (using matching worst-case functions)
- Kim and Fessler (2014) identify an optimized first-order methods (optimal coefficients for DT relaxation), shown to be optimal for smooth convex optimization by Drori (2016)
- Other approach similar in spirit by Lessard, Recht, Packard (2014): integral quadratic constraints, strongly convex case

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)
- Nonconvex formulation proposed by Drori and Teboulle $\rightarrow$ they solve a convex relaxation $\rightarrow$ only provides bounds, but shown to be tight for some algorithms (using matching worst-case functions)
- Kim and Fessler (2014) identify an optimized first-order methods (optimal coefficients for DT relaxation), shown to be optimal for smooth convex optimization by Drori (2016)
- Other approach similar in spirit by Lessard, Recht, Packard (2014): integral quadratic constraints, strongly convex case

- This talk's: an exact formulation for performance estimation of unconstrained optimization algorithms

Brief history of performance estimation

- Concept of performance introduced by Drori and Teboulle (2012, published 2014)
- Nonconvex formulation proposed by Drori and Teboulle $\rightarrow$ they solve a convex relaxation $\rightarrow$ only provides bounds, but shown to be tight for some algorithms (using matching worst-case functions)
- Kim and Fessler (2014) identify an optimized first-order methods (optimal coefficients for DT relaxation), shown to be optimal for smooth convex optimization by Drori (2016)
- Other approach similar in spirit by Lessard, Recht, Packard (2014): integral quadratic constraints, strongly convex case

- This talk's: an exact formulation for performance estimation of unconstrained optimization algorithms
- (if time allows: extensions to constrained case with projected gradient, to composite problems with proximal methods, to Frank-Wolfe-type oracles)

Outline

Preview of our main result

- A reformulation as a semidefinite optimization problem (dimension proportional to N)

Preview of our main result

- A reformulation as a semidefinite optimization problem (dimension proportional to N)
- formulation is exact
  $\rightarrow$ optimal value provides the exact worst-case performance

Preview of our main result

- A reformulation as a semidefinite optimization problem
  (dimension proportional to N)
- formulation is exact
  $\rightarrow$ optimal value provides the exact worst-case performance
- any dual feasible solution
  $\rightarrow$ upper bound on the worst-case performance
  can be converted into a standard proof
  (a weighted sum of valid inequalities)

Preview of our main result

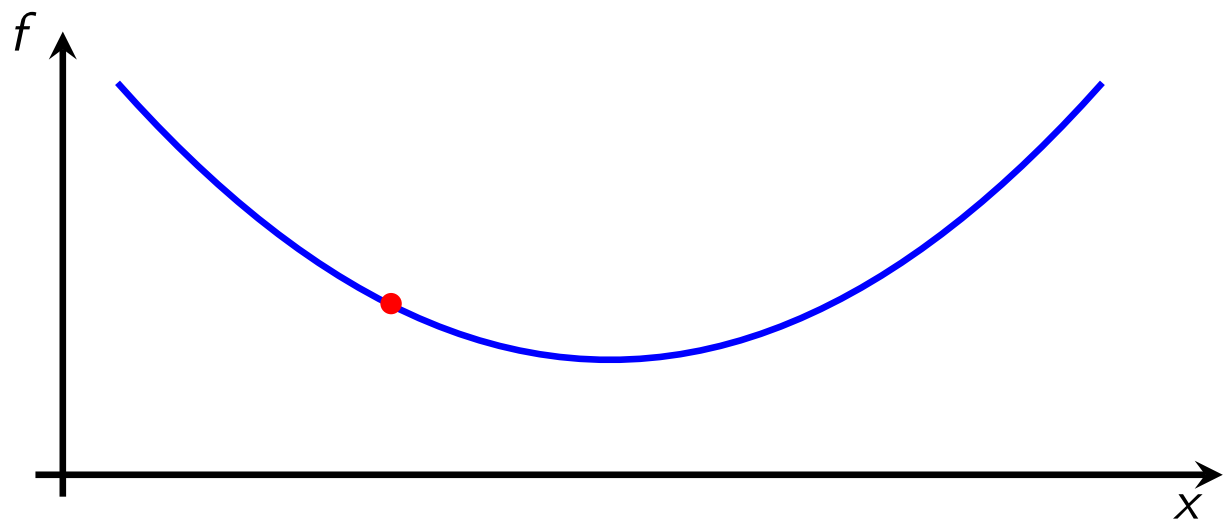
- A reformulation as a semidefinite optimization problem
  (dimension proportional to N)
- formulation is exact
  $\rightarrow$ optimal value provides the exact worst-case performance
- any dual feasible solution
  $\rightarrow$ upper bound on the worst-case performance
  can be converted into a standard proof
  (a weighted sum of valid inequalities)
- any primal feasible solution
  $\rightarrow$ lower bound on the worst case performance
  can be converted into a concrete function

The function class $\mathcal{F}_{\mu,L}$

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:

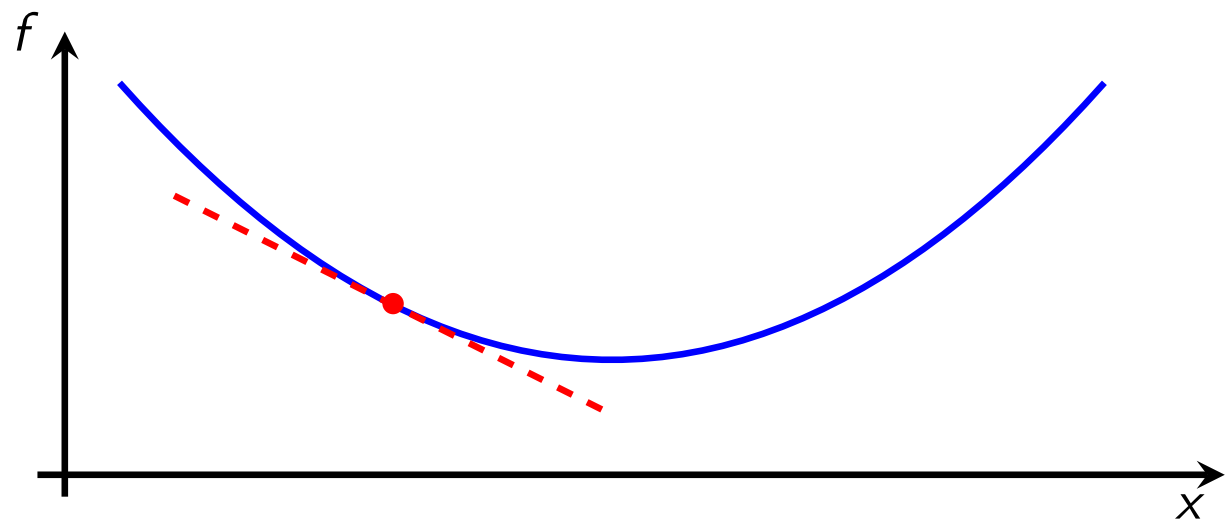
The function class $\mathcal{F}_{\mu,L}$

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:



The function class $\mathcal{F}_{\mu,L}$

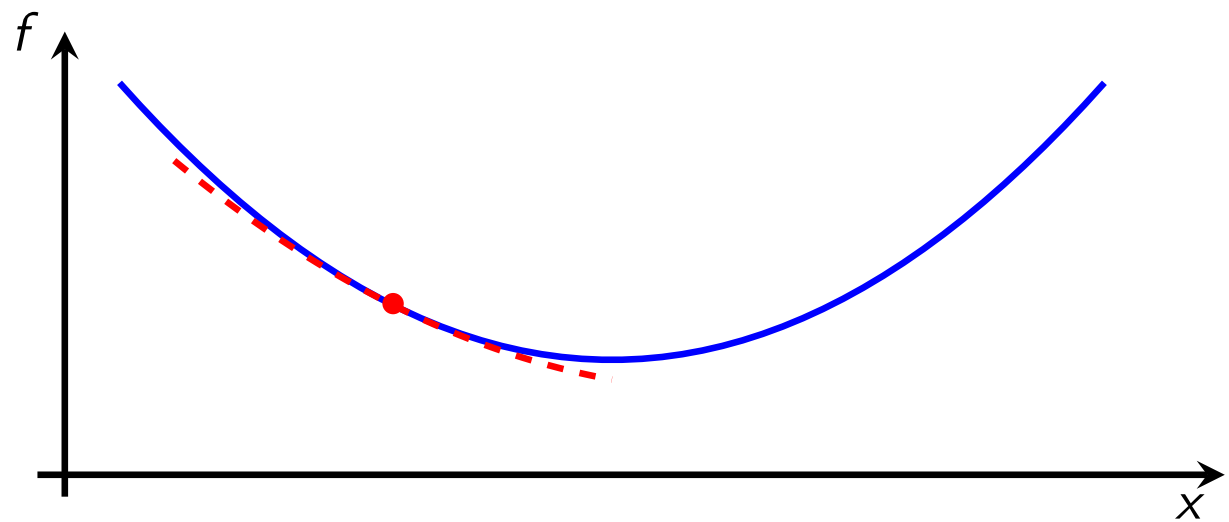
A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:



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

The function class $\mathcal{F}_{\mu,L}$

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:

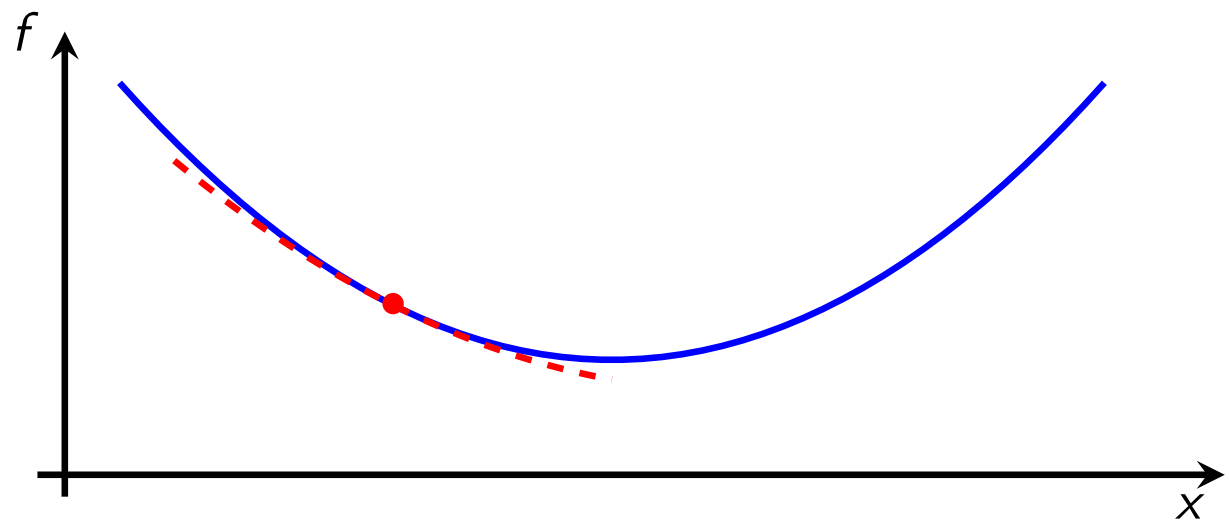


(1) (Convexity) $f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle$,

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

The function class $\mathcal{F}_{\mu,L}$

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:



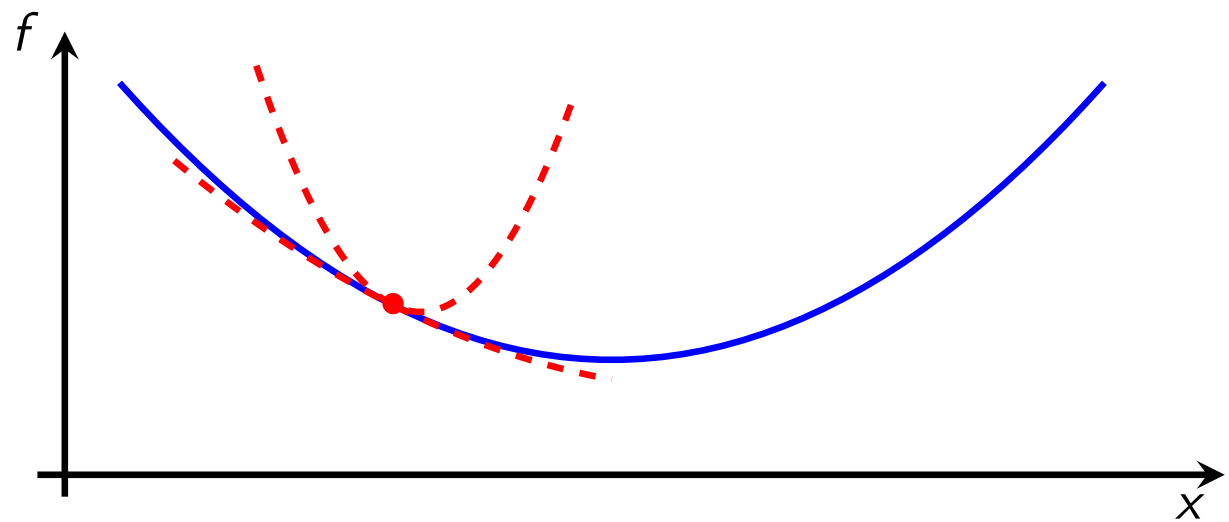
(1) (Convexity) $f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle$,

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

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

The function class $\mathcal{F}_{\mu,L}$

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, f is called (μ-strongly) convex and L-smooth if $\forall x, y \in \mathbb{R}^n$ we have:



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

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

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

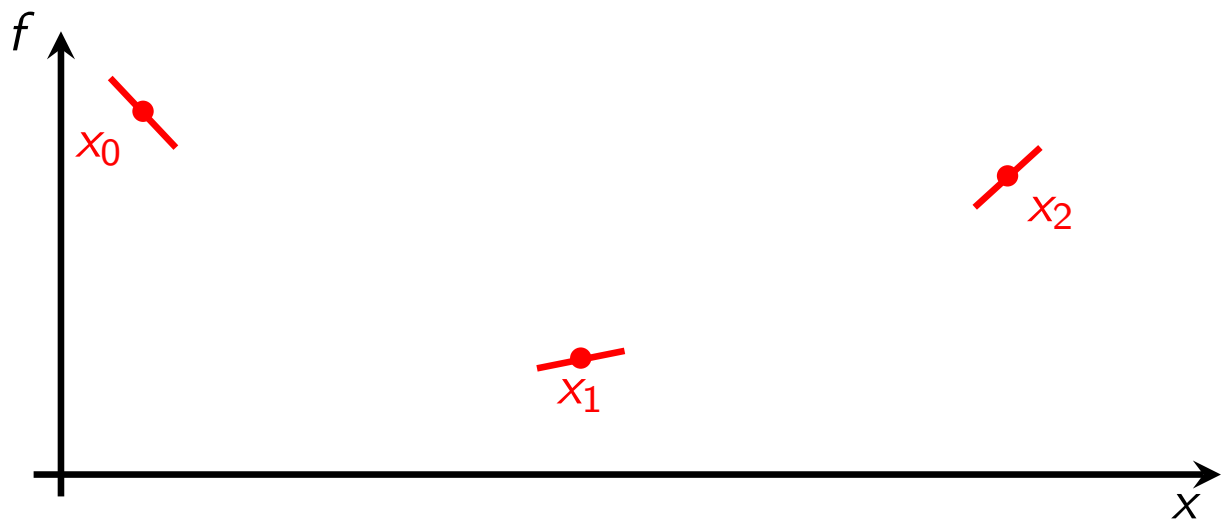
(2b) (L-smoothness) $f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2$.

Smooth strongly convex Interpolation Problem

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.

- Is there $f \in \mathcal{F}_{\mu,L}$ (L-Lipschitz gradient, μ-strongly convex) s.t.

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

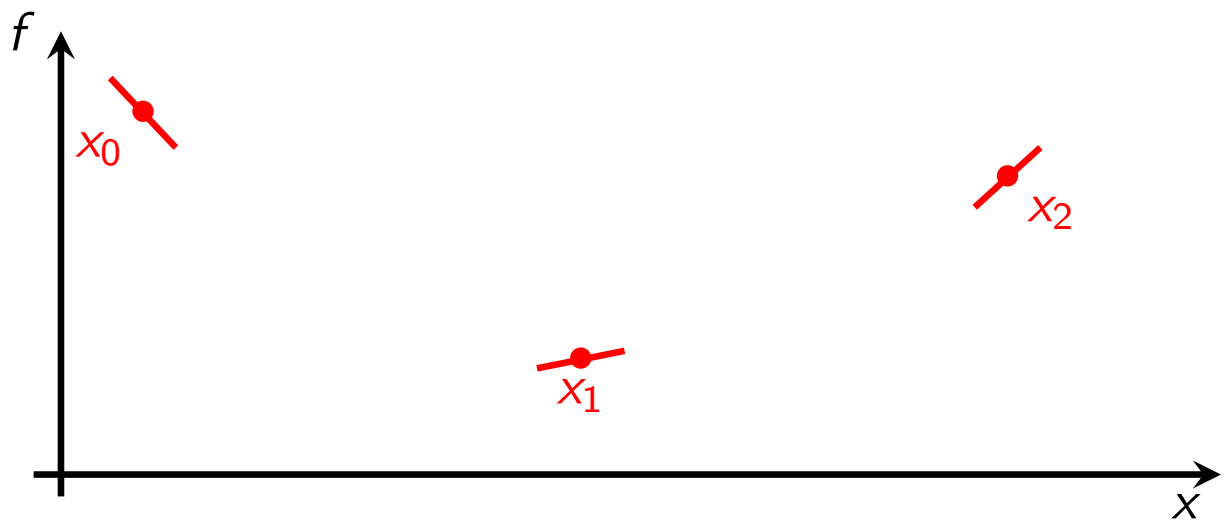


Smooth strongly convex Interpolation Problem

Consider 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.

- Is there $f \in \mathcal{F}_{\mu,L}$ (L-Lipschitz gradient, μ-strongly convex) s.t.

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



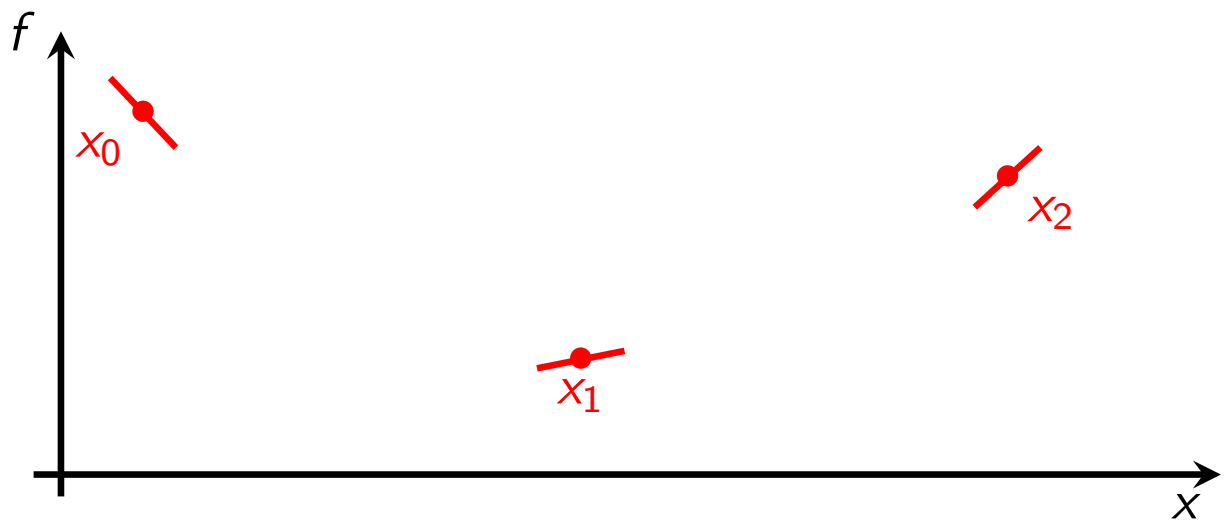
- We want necessary and sufficient conditions for existence of f

Smooth strongly convex Interpolation Problem

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.

- Is there $f \in \mathcal{F}_{\mu,L}$ (L-Lipschitz gradient, μ-strongly convex) s.t.

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



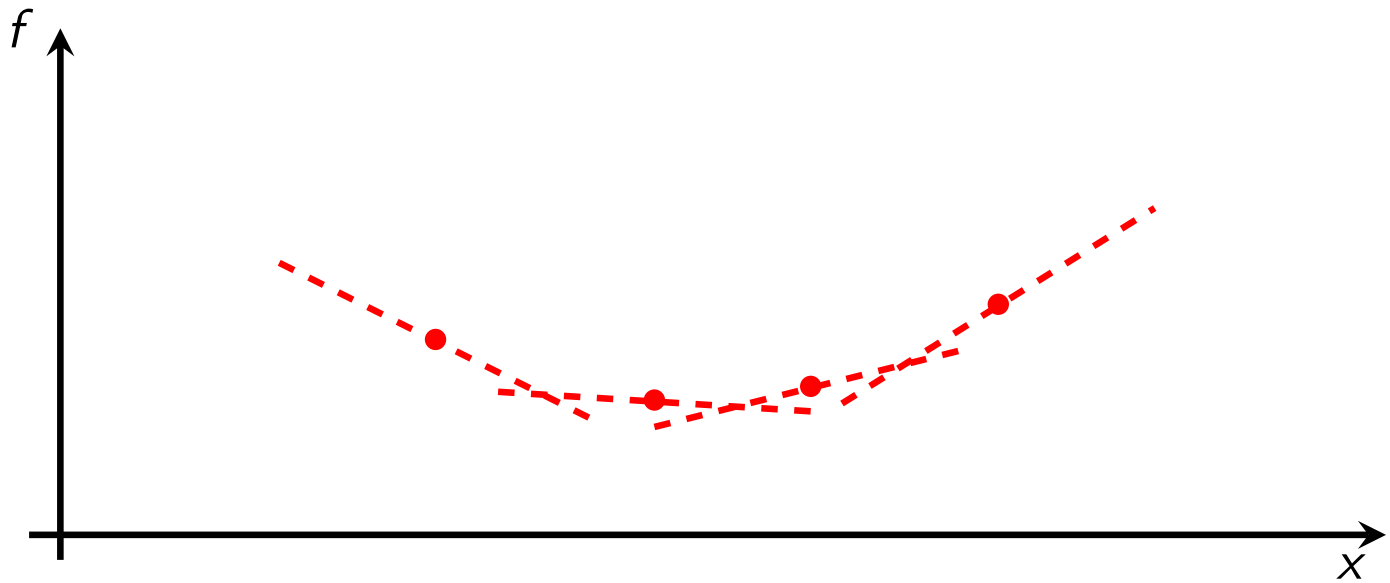
- We want necessary and sufficient conditions for existence of f
- These conditions will appear as a constraints in our PEP formulation

Simple case: convex interpolation ($L = \infty, \mu = 0$)

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) ?

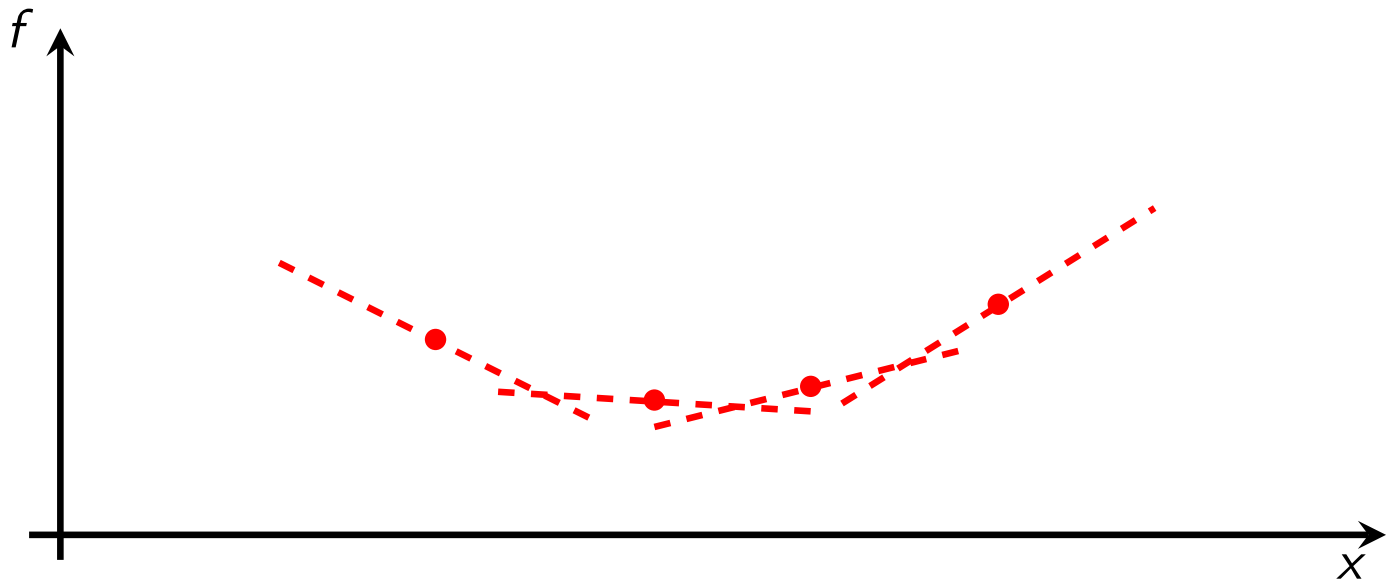
Simple case: convex interpolation ($L = \infty, \mu = 0$)

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) ?



Simple case: convex interpolation ($L = \infty, \mu = 0$)

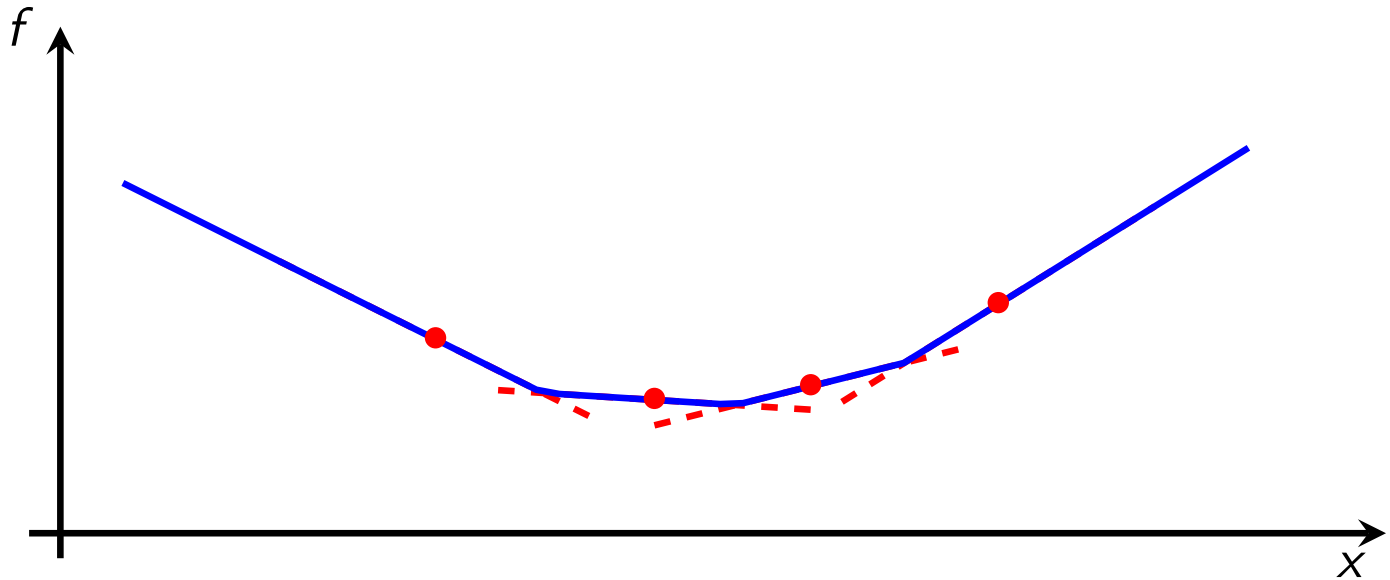
Conditions for $\{(x_i, g_i, f_i)\}_{i \in S}$ to be interpolable by a function $f \in \mathcal{F}_{0,\infty}$ (proper, closed and convex function) ?



Conditions $f_i \geq f_j + g_j^T (x_i - x_j)$ is necessary

Simple case: convex interpolation ($L = \infty, \mu = 0$)

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 $f_i \geq f_j + g_j^T (x_i - x_j)$ is necessary and sufficient

Explicit construction of a (piecewise linear) interpolating function:

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

Not unique.

Next case: smooth convex interpolation ($L < \infty, \mu = 0$)

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

Next case: smooth convex interpolation ($L < \infty, \mu = 0$)

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

First attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i,j \in S, \qquad \text{(C1)}$$
$$||g_i - g_j||_2 \leq L||x_i - x_j||_2.$$

Next case: smooth convex interpolation ($L < \infty, \mu = 0$)

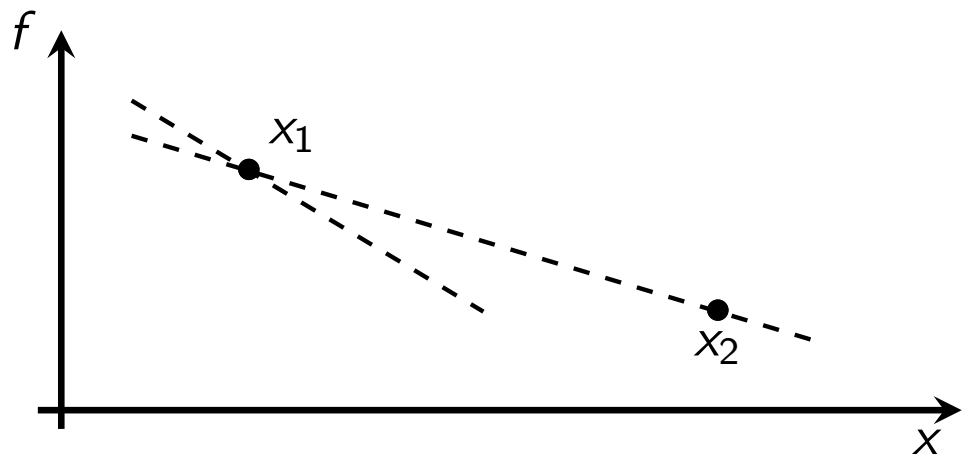
Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

First attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i,j \in S, \qquad \text{(C1)}$$
$$||g_i - g_j||_2 \leq L||x_i - x_j||_2.$$

but not sufficient!

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



satisfies (C1) with $L = 1$ but cannot be differentiable...

Next case: smooth convex interpolation ($L < \infty, \mu = 0$)

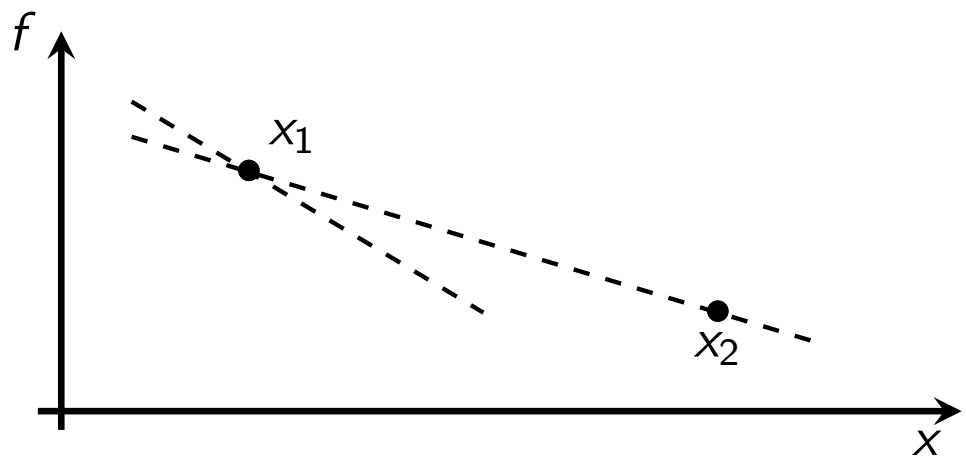
Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

First attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i,j \in S, \qquad \text{(C1)}$$
$$||g_i - g_j||_2 \leq L||x_i - x_j||_2.$$

but not sufficient!

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



satisfies (C1) with $L = 1$ but cannot be differentiable...

(of course conditions work if set S is whole domain)

Smooth convex interpolation ($L < \infty, \mu = 0$)

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

Second attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i, j \in S, \qquad \text{(C2)}$$

$$f_i \leq f_j + g_j^T(x_i - x_j) + \frac{L}{2}||x_i - x_j||_2^2.$$

Smooth convex interpolation ($L < \infty, \mu = 0$)

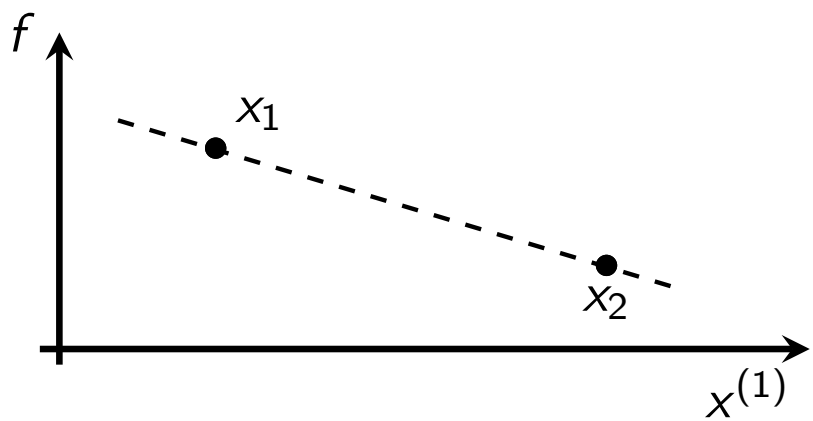
Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

Second attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i, j \in S, \qquad \text{(C2)}$$
$$f_i \leq f_j + g_j^T(x_i - x_j) + \frac{L}{2}||x_i - x_j||_2^2.$$

but not sufficient!

$$(x_1, g_1, f_1) = \left(\begin{pmatrix}0\end{pmatrix}, \begin{pmatrix}1\end{pmatrix}, 0\right)$$
$$(x_2, g_2, f_2) = \left(\begin{pmatrix}1\end{pmatrix}, \begin{pmatrix}1\end{pmatrix}, 1\right)$$



Smooth convex interpolation ($L < \infty, \mu = 0$)

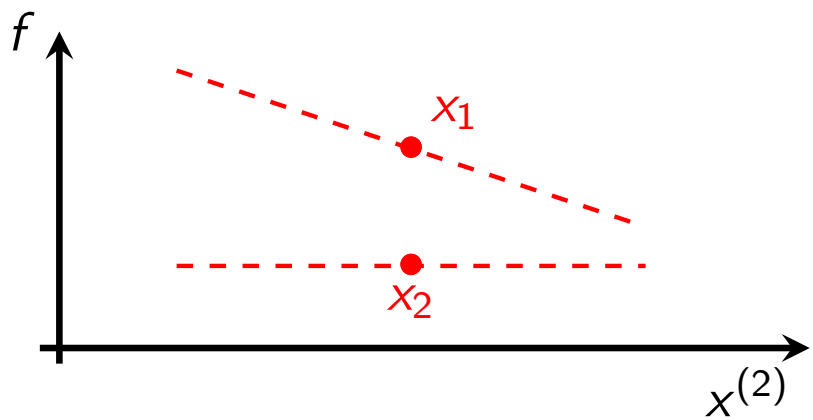
Generalization to smooth interpolation ? Interpolation by a function $f \in \mathcal{F}_{0,L}$ (proper, closed and convex function with L-Lipschitz gradient).

Second attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i,j \in S, \qquad \text{(C2)}$$
$$f_i \leq f_j + g_j^T(x_i - x_j) + \frac{L}{2}||x_i - x_j||_2^2.$$

but not sufficient!

$$(x_1, g_1, f_1) = \left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, 0\right)$$
$$(x_2, g_2, f_2) = \left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\\alpha\end{pmatrix}, 1\right)$$



satisfies (C2) but do not even satisfy the basic conditions (C1)...

Smooth convex interpolation ($L < \infty, \mu = 0$)

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

Second attempt: following set conditions is necessary

$$f_i \geq f_j + g_j^T(x_i - x_j), \qquad i,j \in S, \qquad \text{(C2)}$$
$$f_i \leq f_j + g_j^T(x_i - x_j) + \frac{L}{2}||x_i - x_j||_2^2.$$

but not sufficient!

$$(x_1, g_1, f_1) = \left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, 0\right)$$
$$(x_2, g_2, f_2) = \left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\\alpha\end{pmatrix}, 1\right)$$

satisfies (C2) but do not even satisfy the basic conditions (C1)...

conditions (C2) may also satisfy previous example for some L...

A different approach

Idea: reduce smooth convex interpolation to convex interpolation.

A different approach

Idea: reduce smooth convex interpolation to convex interpolation.

Using two basic operations to transform the problem:

A different approach

Idea: reduce smooth convex interpolation to convex interpolation.

Using two basic operations to transform the problem:

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

  $(x_i, g_i, f_i)_{i \in S}$ is $\mathcal{F}_{0,L}$-interp. $\Leftrightarrow$ $(g_i, x_i, x_i^T g_i - f_i)_{i \in S}$ is $\mathcal{F}_{1/L,\infty}$-interp.

A different approach

Idea: reduce smooth convex interpolation to convex interpolation.

Using two basic operations to transform the problem:

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

  $(x_i, g_i, f_i)_{i \in S}$ is $\mathcal{F}_{0,L}$-interp. $\Leftrightarrow$ $(g_i, x_i, x_i^T g_i - f_i)_{i \in S}$ is $\mathcal{F}_{1/L,\infty}$-interp.

- Minimal curvature subtraction:
  $f(x)$ μ-strongly convex $\Leftrightarrow$ $f(x) - \frac{\mu}{2}||x||_2^2$ convex

  Since $\nabla\left(f(x) - \frac{\mu}{2}||x||^2\right) = \nabla f(x) - \mu x$ we have

$$(x_i, g_i, f_i)_{i \in S} \text{ is } \mathcal{F}_{\mu,L}\text{-interpolable}$$
$$\Leftrightarrow (x_i, g_i - \mu x_i, f_i - \frac{\mu}{2}||x_i||^2)_{i \in S} \text{ is } \mathcal{F}_{0,L-\mu}\text{-interpolable}$$

Reminder: Conjugation

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

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

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

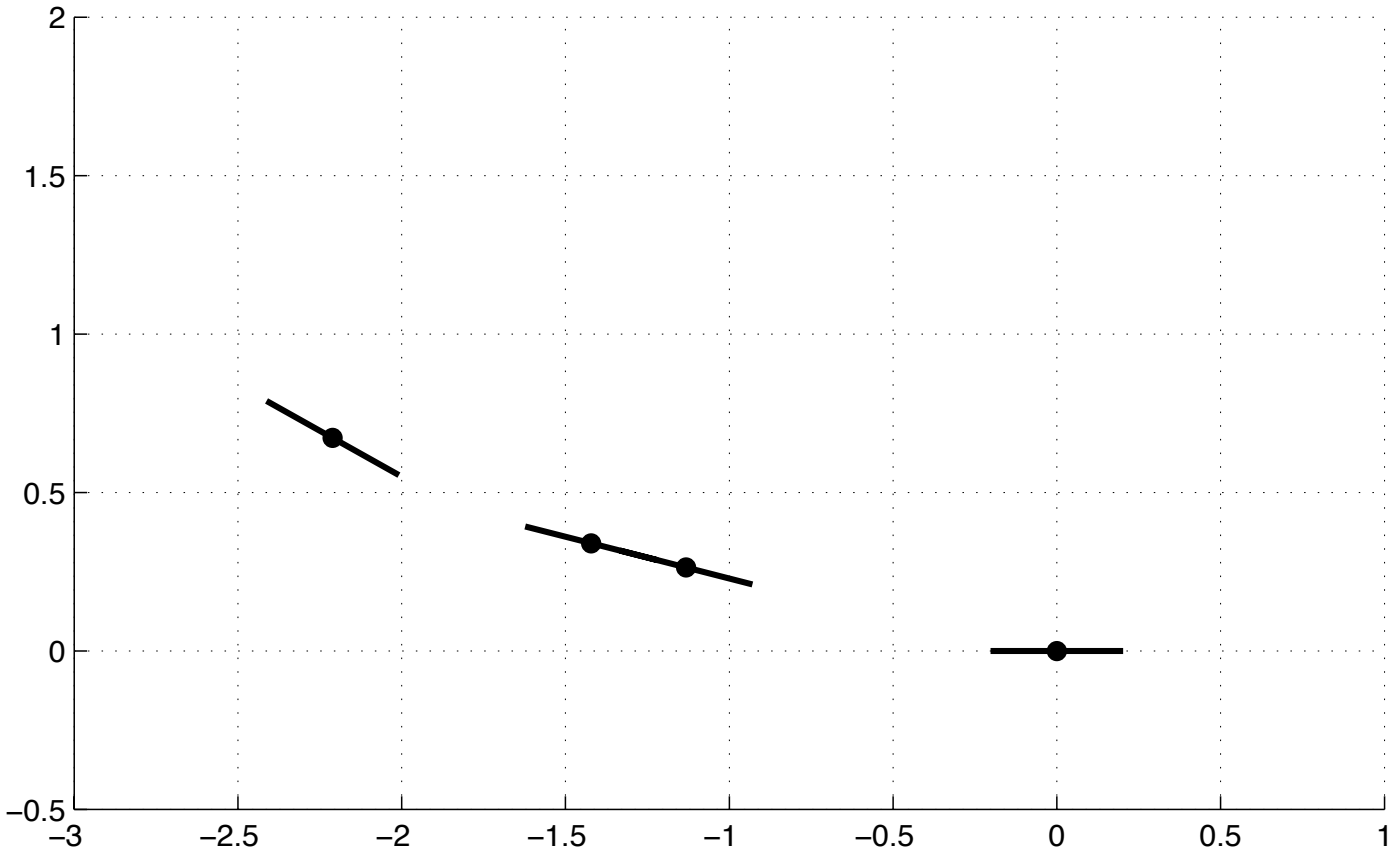
For $f \in \mathcal{F}_{0,\infty}$, we have a one-to-one correspondence between f and f^*, and the following propositions are equivalent:

(a) $f(x) + f^*(g) = g^T x$,
(b) $g \in \partial f(x)$,
(c) $x \in \partial f^*(g)$.

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

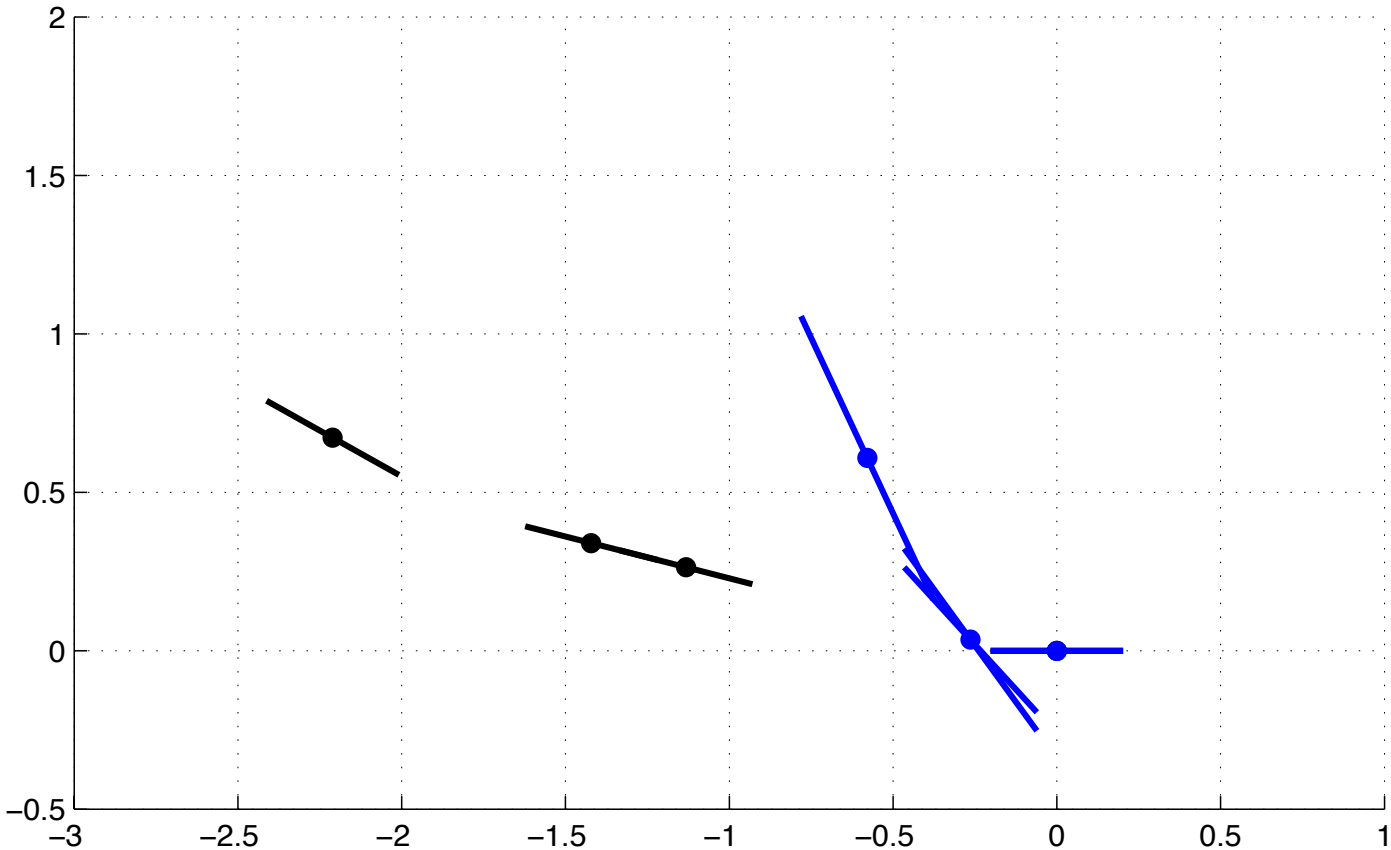
$$(x_i, g_i, f_i)_{i \in S} \text{ is } \mathcal{F}_{0,L}\text{-interpolable}$$
$$\Leftrightarrow (g_i, x_i, x_i^T g_i - f_i)_{i \in S} \text{ is } \mathcal{F}_{1/L,\infty}\text{-interpolable}$$

Example: Smooth Convex Interpolation



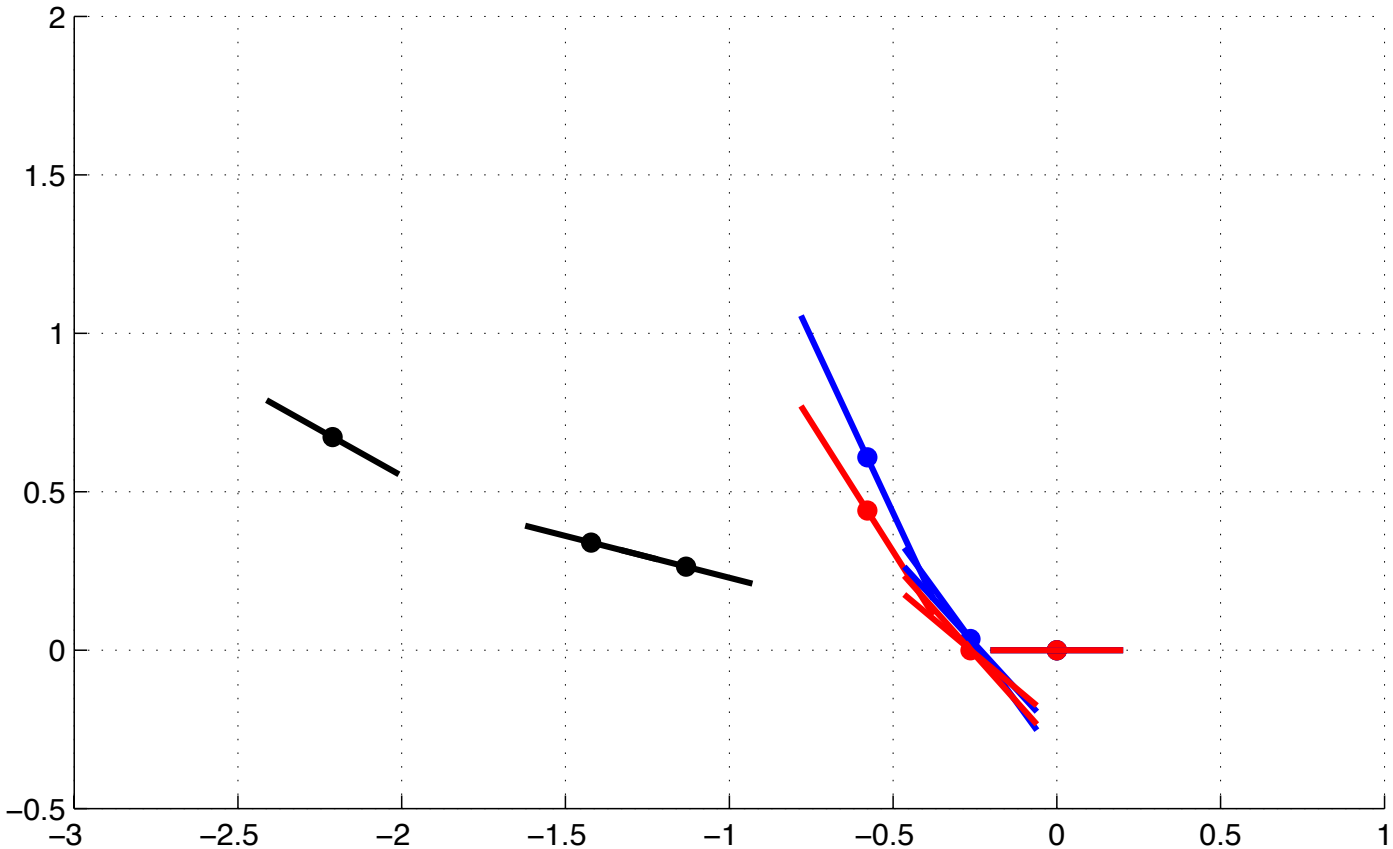
Interpolate $\{(x_i, g_i, f_i)\}_{i \in S}$ by $f \in \mathcal{F}_{0,L}$

Example: Smooth Convex Interpolation



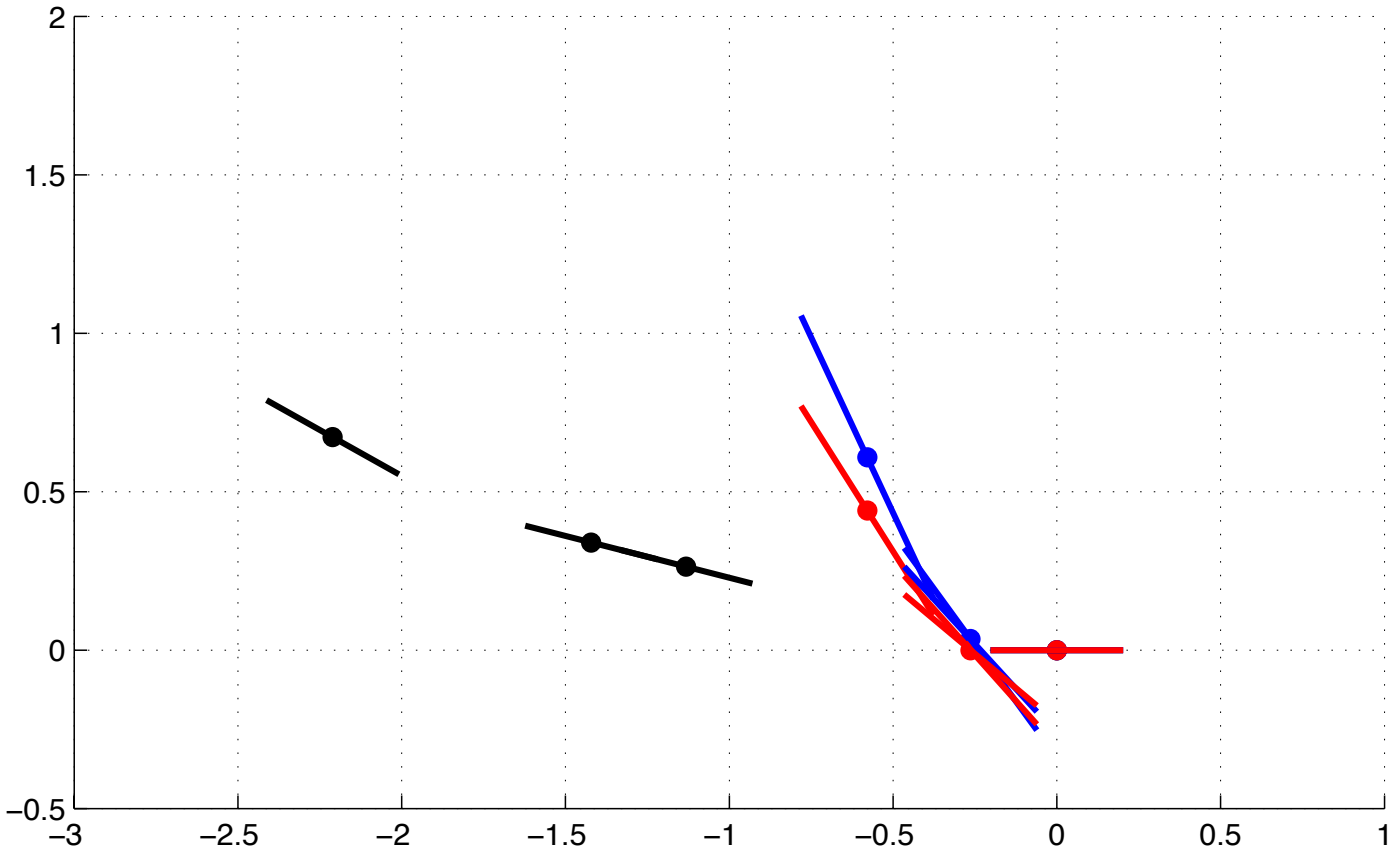
$\Leftrightarrow$ interpolate $\left\{(g_i, x_i, x_i^T g_i - f_i)\right\}_{i \in S}$ by $f^* \in \mathcal{F}_{1/L,\infty}$.

Example: Smooth Convex Interpolation



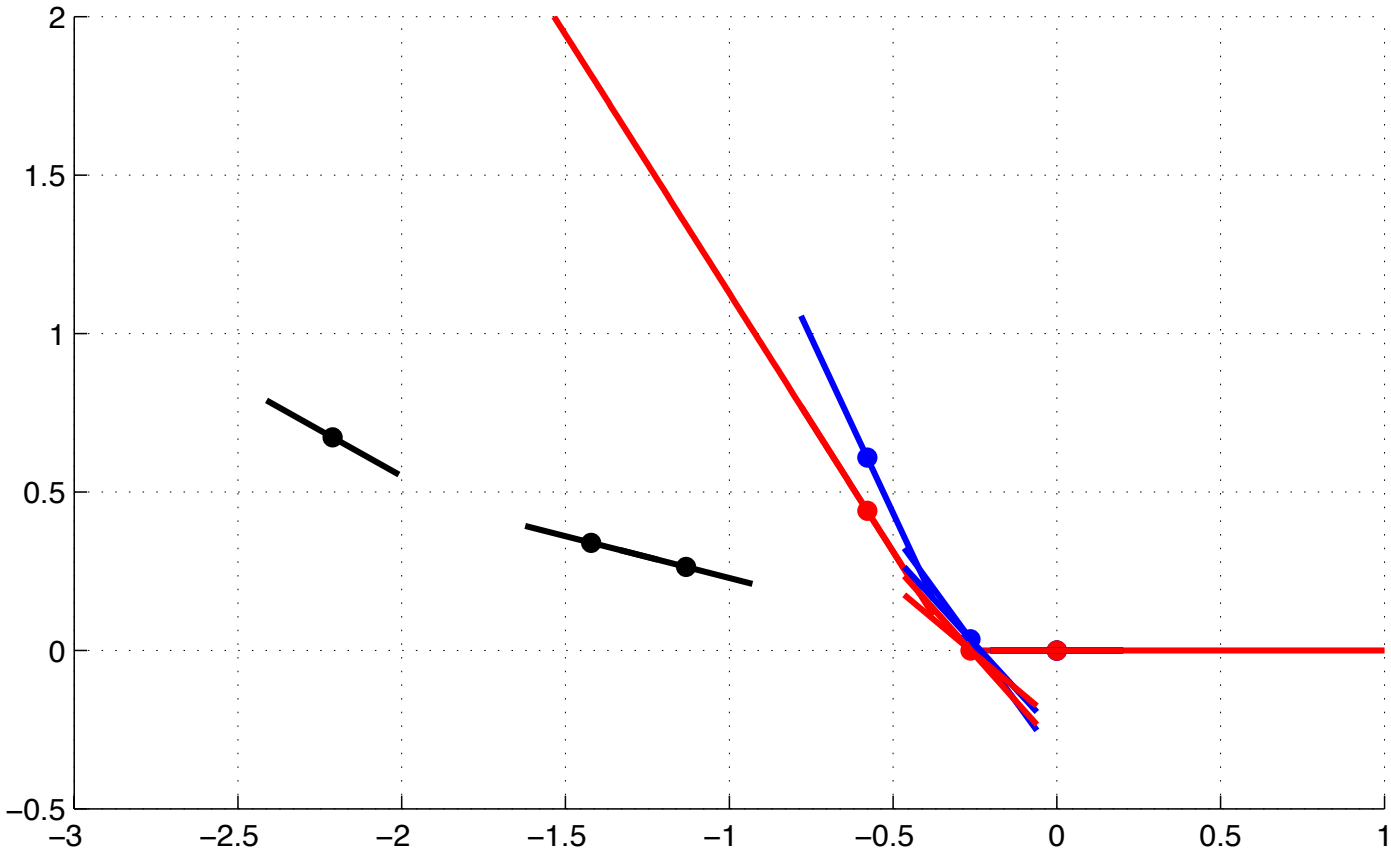
$\Leftrightarrow$ interpolate $\left\{\left(g_i, x_i - \frac{g_i}{L}, x_i^T g_i - f_i - \frac{||g_i||_2^2}{2L}\right)\right\}_{i \in S}$ by $\tilde{f} \in \mathcal{F}_{0,\infty}$.

Example: Smooth Convex Interpolation



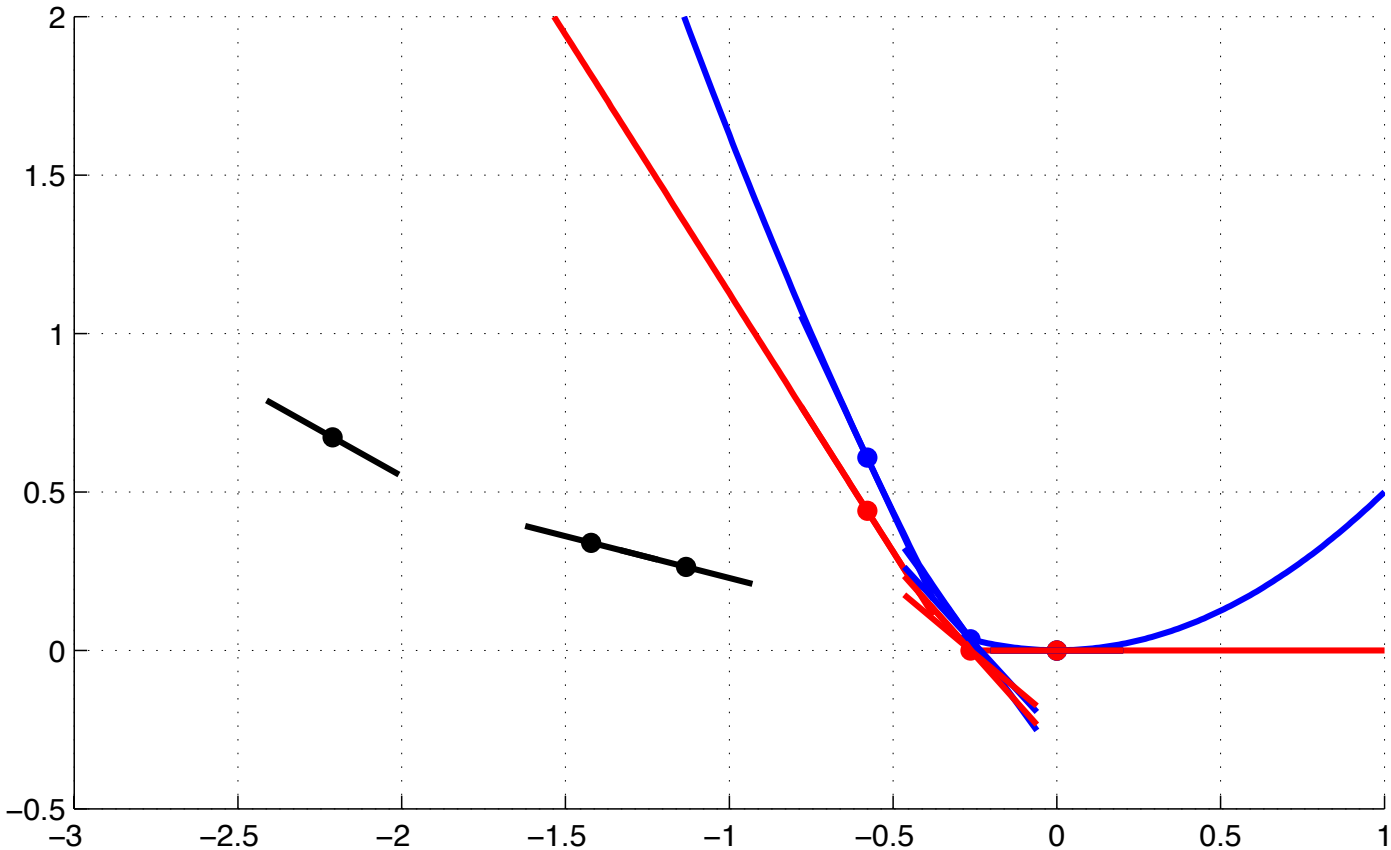
$\Leftrightarrow$ interpolate $\left\{(\tilde{x}_i, \tilde{g}_i, \tilde{f}_i)\right\}_{i \in S}$ by $\tilde{f} \in \mathcal{F}_{0,\infty}$.

Example: Smooth Convex Interpolation



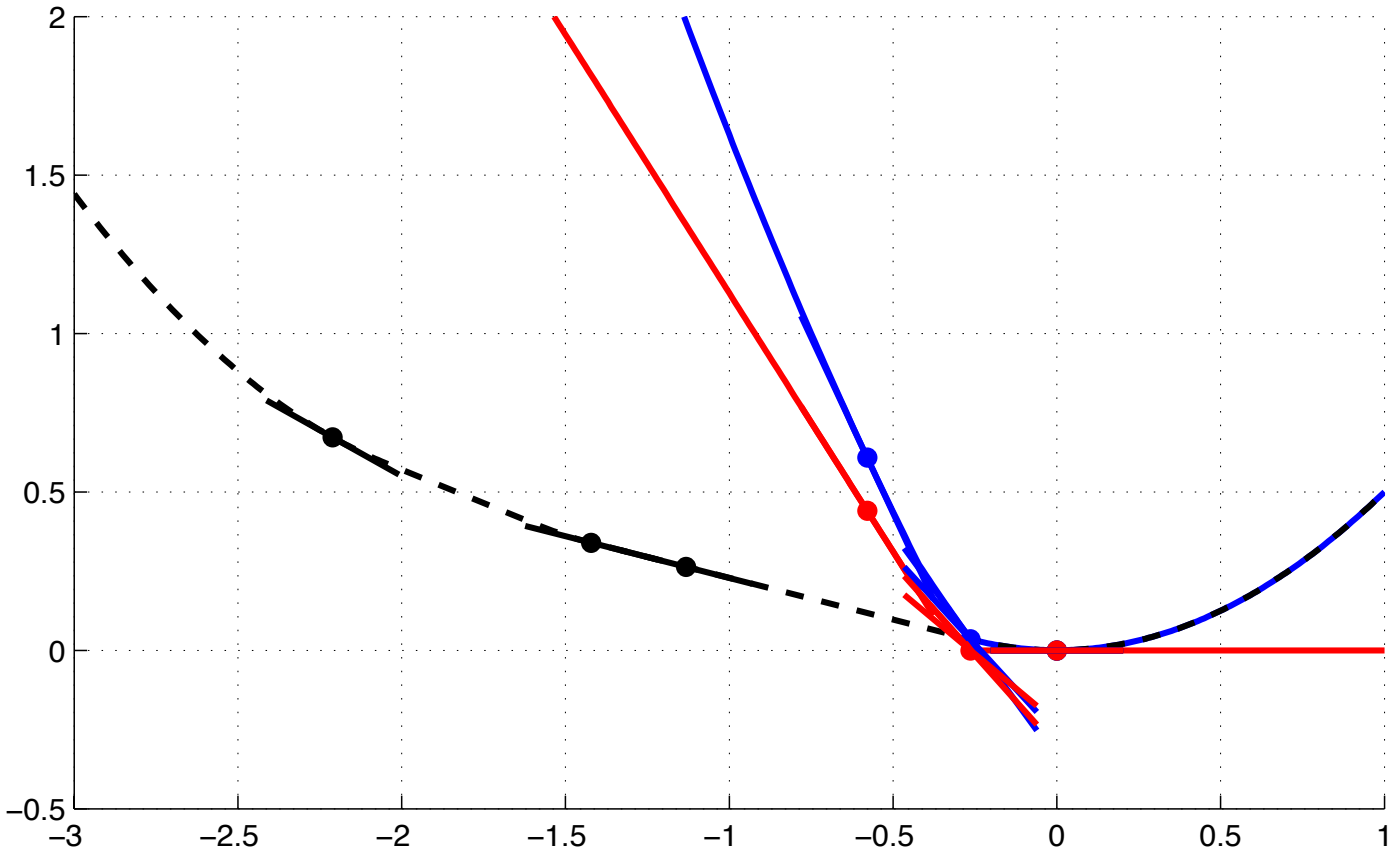
$$\tilde{f}(x) = \max_j \left\{ \tilde{f}_j + \tilde{g}_j^T (x - \tilde{x}_j) \right\}$$

Example: Smooth Convex Interpolation



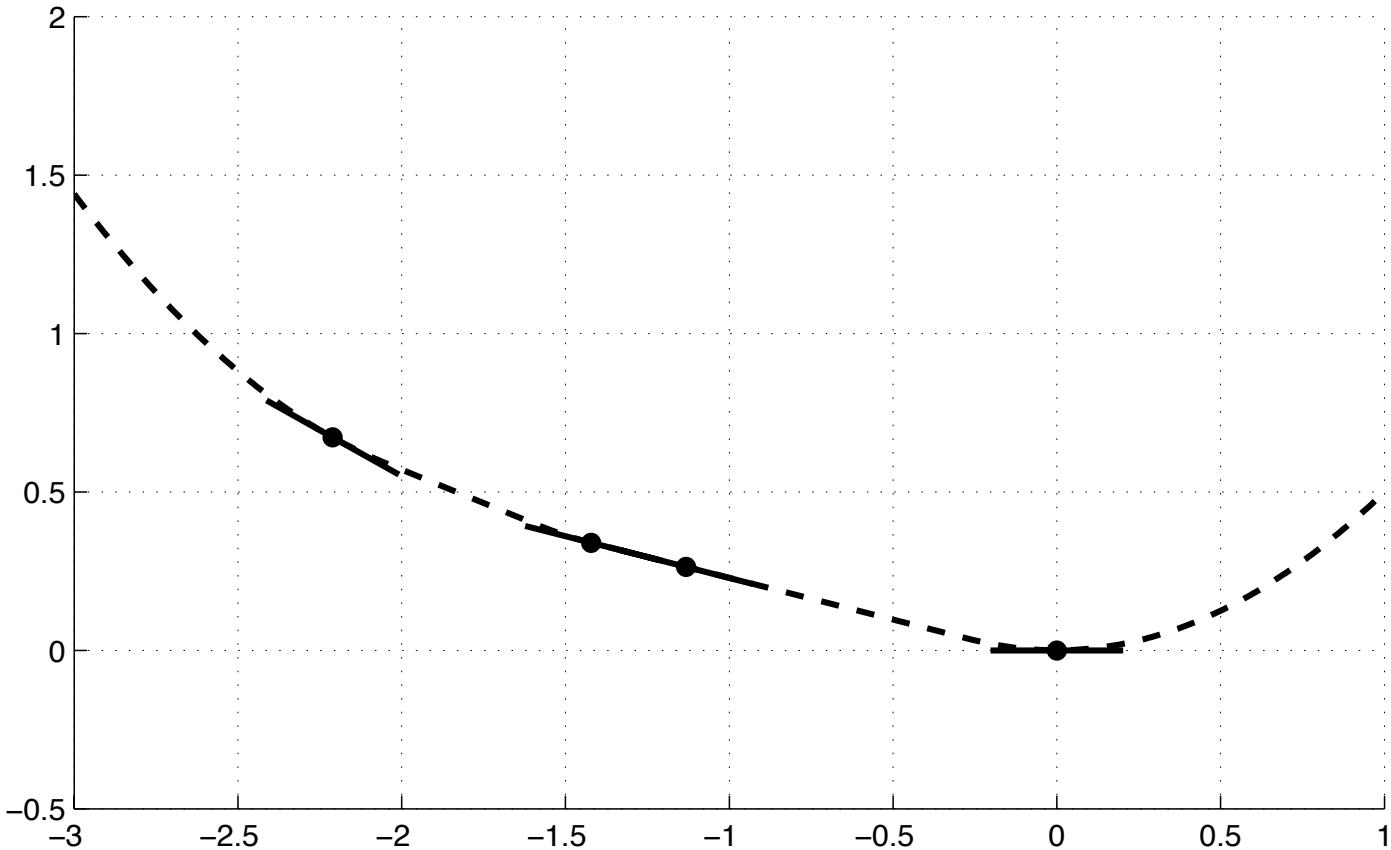
$$f^*(x) = \max_j \left\{ \tilde{f}_j + \tilde{g}_j^T (x - \tilde{x}_j) \right\} + \frac{||x||_2^2}{2L}$$

Example: Smooth Convex Interpolation



$$f(x) = \left(\max_j \left\{ \tilde{f}_j + \tilde{g}_j^T (x - \tilde{x}_j) \right\} + \frac{||x||_2^2}{2L} \right)^*$$

Example: Smooth Convex Interpolation



$$f(x) = \left(\max_j \left\{ \tilde{f}_j + \tilde{g}_j^T (x - \tilde{x}_j) \right\} + \frac{||x||_2^2}{2L} \right)^*$$

Most general: smooth strongly convex interpolation

Theorem [Taylor, Hendrickx, G. 2016]
Set $\{(x_i, g_i, f_i)\}_{i \in S}$ is $\mathcal{F}_{\mu,L}$-interpolable if and only

$$f_i - f_j - g_j^\top (x_i - x_j) \geq \frac{1}{2(1-\mu/L)} \left(\frac{1}{L} \|g_i - g_j\|_2^2 \cdots \right.$$
$$\left. + \mu \|x_i - x_j\|_2^2 - 2\frac{\mu}{L} (g_j - g_i)^\top (x_j - x_i) \right)$$

holds for every pair of indices $i \in I$ and $j \in S$

Most general: smooth strongly convex interpolation

Theorem [Taylor, Hendrickx, G. 2016]
Set $\{(x_i, g_i, f_i)\}_{i \in S}$ is $\mathcal{F}_{\mu,L}$-interpolable if and only

$$f_i - f_j - g_j^\top (x_i - x_j) \geq \frac{1}{2(1-\mu/L)} \left(\frac{1}{L} \|g_i - g_j\|_2^2 \cdots \right.$$
$$\left. + \mu \|x_i - x_j\|_2^2 - 2\frac{\mu}{L} (g_j - g_i)^\top (x_j - x_i) \right)$$

holds for every pair of indices $i \in I$ and $j \in S$

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

$$f_j \geq f_i + g_i^T (x_j - x_i) + \frac{1}{2L} ||g_i - g_j||_2^2 \qquad \forall i, j \in S$$

which we now know are also sufficient for interpolation

Most general: smooth strongly convex interpolation

Theorem [Taylor, Hendrickx, G. 2016]
Set $\{(x_i, g_i, f_i)\}_{i \in S}$ is $\mathcal{F}_{\mu,L}$-interpolable if and only

$$f_i - f_j - g_j^\top (x_i - x_j) \geq \frac{1}{2(1-\mu/L)} \left(\frac{1}{L} \|g_i - g_j\|_2^2 \cdots \right.$$
$$\left. + \mu \|x_i - x_j\|_2^2 - 2\frac{\mu}{L} (g_j - g_i)^\top (x_j - x_i) \right)$$

holds for every pair of indices $i \in I$ and $j \in S$

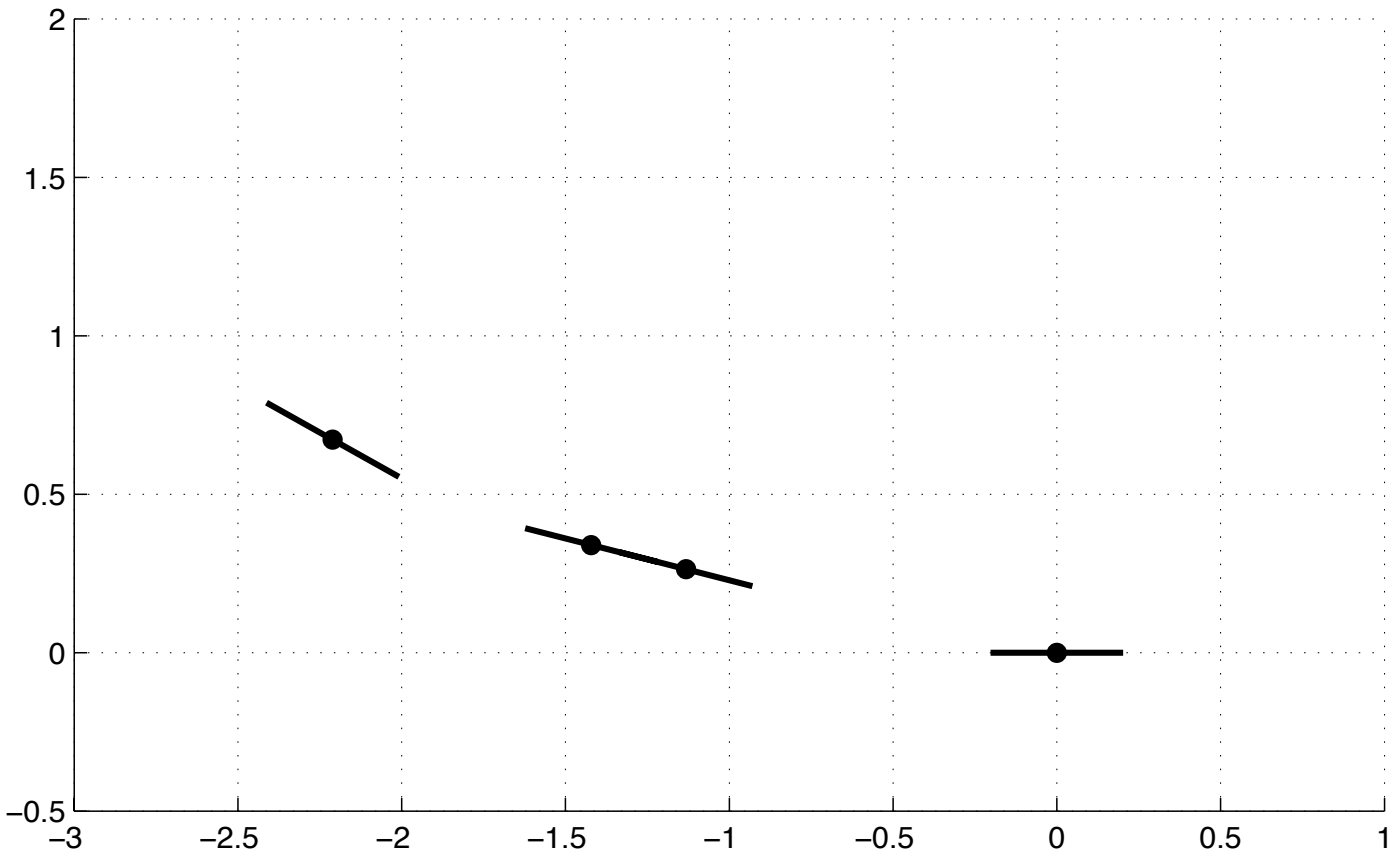
When $\mu = 0$, those conditions reduce to the well-known necessary

$$f_j \geq f_i + g_i^T (x_j - x_i) + \frac{1}{2L} ||g_i - g_j||_2^2 \qquad \forall i, j \in S$$

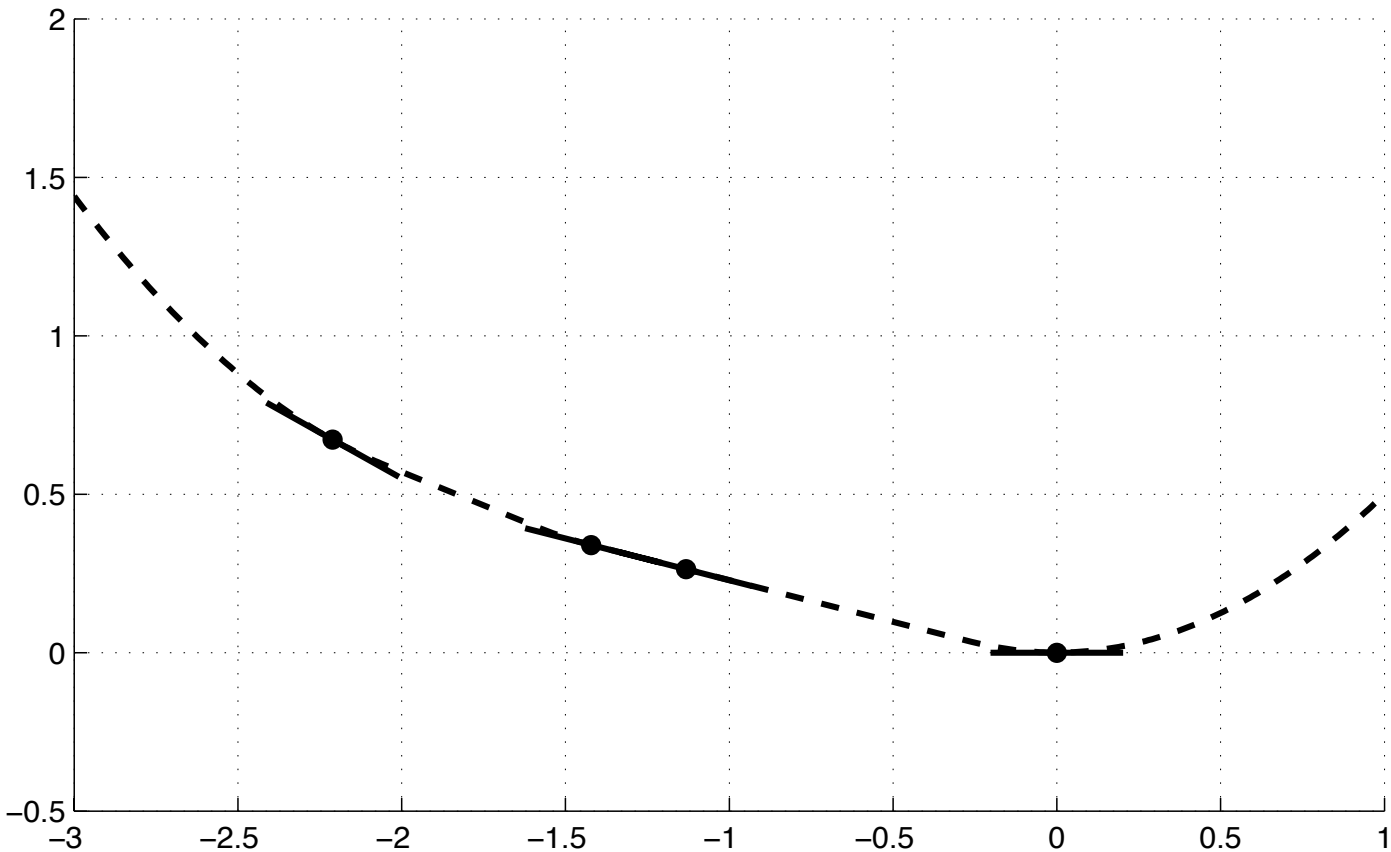
which we now know are also sufficient for interpolation

Proof provides an explicit interpolating function
(piecewise linear-quadratic, not unique)

Example



Example



A semidefinite optimization formulation

Let $I = \{0, 1, \ldots, N, *\}$, assume w.l.o.g. $f^* = 0$, $x^* = 0$, $g^* = 0$

Our performance estimation problem is now

$$\sup_{\{x_i, g_i, f_i\}_{i \in I \setminus \{*\}}} \quad \mathcal{P}\left(\{x_i, g_i, f_i\}_{i \in I}\right), \qquad \text{(f-PEP2)}$$

$$\text{such that } \{x_i, g_i, f_i\}_{i \in I} \text{ is } \mathcal{F}_{\mu,L}\text{-interpolable},$$

$$x_1, \ldots, x_N \text{ is generated by method } \mathcal{M},$$

$$\|x_0 - x_*\|_2 \leq R.$$

- Drori and Teboulle (2014) obtain upper bounds with a dual of this nonconvex problem (after relaxation / reformulation)
- We want an exact and convex reformulation

Fixed-step first-order algorirhms

- In order to deal with the method constraint, we only consider fixed-step first-order methods

$$x_i = x_0 - \frac{1}{L} \sum_{k=0}^{i-1} h_{i,k} g_k$$

where $h_{i,k}$ and fixed constants

Fixed-step first-order algorirhms

- In order to deal with the method constraint, we only consider fixed-step first-order methods

$$x_i = x_0 - \frac{1}{L} \sum_{k=0}^{i-1} h_{i,k} g_k$$

  where $h_{i,k}$ and fixed constants
- Iterates are defined by linear expressions involving past oracle outputs

Fixed-step first-order algorirhms

- In order to deal with the method constraint, we only consider fixed-step first-order methods

$$x_i = x_0 - \frac{1}{L} \sum_{k=0}^{i-1} h_{i,k} g_k$$

  where $h_{i,k}$ and fixed constants
- Iterates are defined by linear expressions involving past oracle outputs
- Many classic, standard black-box methods can be reformulated in this way (including methods based on multiple sequences)

Fixed-step first-order algorirhms

- In order to deal with the method constraint, we only consider fixed-step first-order methods

$$x_i = x_0 - \frac{1}{L} \sum_{k=0}^{i-1} h_{i,k} g_k$$

  where $h_{i,k}$ and fixed constants
- Iterates are defined by linear expressions involving past oracle outputs
- Many classic, standard black-box methods can be reformulated in this way (including methods based on multiple sequences)
- Recursively express all iterates in terms of gradients (and initial iterate x_0)

Fixed-step first-order algorirhms

- In order to deal with the method constraint, we only consider fixed-step first-order methods

$$x_i = x_0 - \frac{1}{L} \sum_{k=0}^{i-1} h_{i,k} g_k$$

  where $h_{i,k}$ and fixed constants
- Iterates are defined by linear expressions involving past oracle outputs
- Many classic, standard black-box methods can be reformulated in this way (including methods based on multiple sequences)
- Recursively express all iterates in terms of gradients (and initial iterate x_0)
- This leads to tractable linear equalities in our formulation, involving variables x_i and g_i only

Simple example: PEP for gradient method

$$\max_{f,x_0,\ldots,x_N,x^*} f(x_N) - f^*,$$

subject to the constraints

$f \in \mathcal{F}_{0,L}$ i.e. convex and has L-Lipschitz gradient

$\nabla f(x^*) = 0,$

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

$||x_0 - x^*||_2^2 \leq R^2.$

can be reformulated as finite-dimensional problem using our $\mathcal{F}_{0,L}$-interpolation conditions

Formulation for performance optimization

Worst-case estimation problem translates into an exact finite problem

$$\max\ f_N - f^*$$

$$\text{s.t. } f_j \geq f_i + g_i^T(x_j - x_i) + \frac{1}{2L}||g_i - g_j||_2^2 \qquad \forall i \neq j \in \{0, ..., N, *\}$$

$$x_{i+1} = x_i - \frac{h}{L}g_i \qquad \forall i \in \{0, ..., N-1\}$$

$$(x_0 - x^*)^T(x_0 - x^*) \leq R^2$$

- Only remaining difficulty: scalar products, i.e. nonconvex quadratic constraints

Formulation for performance optimization

Worst-case estimation problem translates into an exact finite problem

$$\max\ f_N - f^*$$

$$\text{s.t. } f_j \geq f_i + g_i^T(x_j - x_i) + \frac{1}{2L}||g_i - g_j||_2^2 \qquad \forall i \neq j \in \{0, ..., N, *\}$$

$$x_{i+1} = x_i - \frac{h}{L}g_i \qquad \forall i \in \{0, ..., N-1\}$$

$$(x_0 - x^*)^T(x_0 - x^*) \leq R^2$$

- Only remaining difficulty: scalar products, i.e. nonconvex quadratic constraints
- Solution: Gram matrix $G \succeq 0$, containing all scalar products which turns the problem turns into an equivalent semidefinite optimization problem

Formulation for performance optimization

Example: $N = 1$ iteration, assume wlog $x^* = g^* = 0$, and choose

$$P = (x_0 \mid x_1 \mid g_0 \mid g_1)$$

$$G = P^T P = \begin{pmatrix} x_0^T x_0 & x_0^T x_1 & x_0^T g_0 & x_0^T g_1 \\ x_1^T x_0 & x_1^T x_1 & x_1^T g_0 & x_1^T g_1 \\ g_0^T x_0 & g_0^T x_1 & g_0^T g_0 & g_0^T g_1 \\ g_1^T x_0 & g_1^T x_1 & g_1^T g_0 & g_1^T g_1 \end{pmatrix} \succeq 0.$$

Formulation for performance optimization

Example: $N = 1$ iteration, assume wlog $x^* = g^* = 0$, and choose

$$P = (x_0 \mid x_1 \mid g_0 \mid g_1)$$

$$G = P^T P = \begin{pmatrix} x_0^T x_0 & x_0^T x_1 & x_0^T g_0 & x_0^T g_1 \\ x_1^T x_0 & x_1^T x_1 & x_1^T g_0 & x_1^T g_1 \\ g_0^T x_0 & g_0^T x_1 & g_0^T g_0 & g_0^T g_1 \\ g_1^T x_0 & g_1^T x_1 & g_1^T g_0 & g_1^T g_1 \end{pmatrix} \succeq 0.$$

Our interpolating constraints become linear in the elements of G

Formulation for performance optimization

Example: $N = 1$ iteration, assume wlog $x^* = g^* = 0$, and choose

$$P = (x_0 \mid x_1 \mid g_0 \mid g_1)$$

$$G = P^T P = \begin{pmatrix} x_0^T x_0 & x_0^T x_1 & x_0^T g_0 & x_0^T g_1 \\ x_1^T x_0 & x_1^T x_1 & x_1^T g_0 & x_1^T g_1 \\ g_0^T x_0 & g_0^T x_1 & g_0^T g_0 & g_0^T g_1 \\ g_1^T x_0 & g_1^T x_1 & g_1^T g_0 & g_1^T g_1 \end{pmatrix} \succeq 0.$$

Our interpolating constraints become linear in the elements of G

From $G \succeq 0$, we can recover x_0, x_1, g_0 and g_1.

Formulation for performance optimization

Example: $N = 1$ iteration, assume wlog $x^* = g^* = 0$, and choose

$$P = (x_0 \mid x_1 \mid g_0 \mid g_1)$$

$$G = P^T P = \begin{pmatrix} x_0^T x_0 & x_0^T x_1 & x_0^T g_0 & x_0^T g_1 \\ x_1^T x_0 & x_1^T x_1 & x_1^T g_0 & x_1^T g_1 \\ g_0^T x_0 & g_0^T x_1 & g_0^T g_0 & g_0^T g_1 \\ g_1^T x_0 & g_1^T x_1 & g_1^T g_0 & g_1^T g_1 \end{pmatrix} \succeq 0.$$

Our interpolating constraints become linear in the elements of G
From $G \succeq 0$, we can recover x_0, x_1, g_0 and g_1.

Function f has d variables $\Leftrightarrow \operatorname{rank}(G) \leq d$
Formulation is exact and dimension-free in large-scale case, i.e.

$$N \ll d \quad \text{(actually when } 2N + 2 \leq d\text{)}$$

Final semidefinite formulation for performance optimization

Assuming the performance criterion idepends linearly on function values f_i and elements of G ($g_i^T g_j$ and $x_0^T g_j$) we now have

$$\begin{aligned} \max_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} \quad & b^T f + \mathrm{Tr}(CG) && \text{(sdp-PEP)} \\ \text{s.t.} \quad & f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0, && \text{for all } i, j \in I, \\ & \mathrm{Tr}\left(GA_R\right) - R^2 \leq 0, \\ & G \succeq 0. \end{aligned}$$

- Constant data matrices A_{ij}, and A_R that depend on method $\mathcal{M}$ (i.e. coefficients $h_{i,k}$) and function class parameters L, μ

Final semidefinite formulation for performance optimization

Assuming the performance criterion idepends linearly on function values f_i and elements of G ($g_i^T g_j$ and $x_0^T g_j$) we now have

$$\begin{aligned} \max_{G \in \mathbb{S}^{N+2}, f \in \mathbb{R}^{N+1}} \quad & b^T f + \mathrm{Tr}(CG) && \text{(sdp-PEP)} \\ \text{s.t.} \quad & f_j - f_i + \mathrm{Tr}\left(GA_{ij}\right) \leq 0, && \text{for all } i, j \in I, \\ & \mathrm{Tr}\left(GA_R\right) - R^2 \leq 0, \\ & G \succeq 0. \end{aligned}$$

- Constant data matrices A_{ij}, and A_R that depend on method $\mathcal{M}$ (i.e. coefficients $h_{i,k}$) and function class parameters L, μ
- Exact formulation, matrix variable G is size $N+2$, has $\mathcal{O}(N^2)$ linear constraints

Obtaining a proof for the exact worst-case performance

Solution of (primal) provides an explicit function attaining the worst-case performance (using interpolation of the solution)
$\rightarrow$ computed value is a lower bound on worst-case performance

Obtaining a proof for the exact worst-case performance

Solution of (primal) provides an explicit function attaining the worst-case performance (using interpolation of the solution)
$\rightarrow$ computed value is a lower bound on worst-case performance

To obtain a proof that computed value is an upper bound on the worst-case performance, use solution of the dual problem

Obtaining a proof for the exact worst-case performance

Solution of (primal) provides an explicit function attaining the worst-case performance (using interpolation of the solution)
$\rightarrow$ computed value is a lower bound on worst-case performance

To obtain a proof that computed value is an upper bound on the worst-case performance, use solution of the dual problem

$$\inf_{S,\lambda_{ij},\tau} \ \tau R^2 \ \text{ such that } \ \tau A_R - C + \sum_{i,j\in I} \lambda_{ij} A_{ij} = S, \quad \text{(d-sdp-PEP)}$$

$$\sum_{i,j\in I} \lambda_{ij}(u_j - u_i) = b,$$

$$S \succeq 0,$$

$$\lambda_{ij} \geq 0, \qquad i,j \in I,$$

$$\tau \geq 0,$$

to derive a bound in the spirit of the introductory example

Obtaining a proof for the exact worst-case performance

- Dual solution will provide a coefficient λ_{ij} for each interpolating condition
  $\rightarrow$ weights for the list of inequalities valid for each pairs of iterates

  In the example: $\lambda_{01} = \lambda_{*0} = \lambda_{*1} = \frac{1}{2}$ and $\lambda_{10} = 0$, leading to a sum of three valid inequalities $SC(i,j) \geq 0$

$$\frac{1}{2}SC(0,1) + \frac{1}{2}SC(*,0) + \frac{1}{2}SC(*,1) \geq 0$$

- Dual solution will provide a positive semidefinite slack matrix S
  $\rightarrow$ quadratic form (in the x_i and g_i) expressing the slack for the weighted sum of valid inequalities will be nonnegative

  In the example $0 \leq \frac{1}{2}SC(0,1) + \frac{1}{2}SC(*,0) + \frac{1}{2}SC(*,1) = \frac{1}{4}L\|d\|^2 - \Big(f(x_1) - f(x_*)\Big) - QUAD(x,g)$ with

$$QUAD(x,g) = \frac{L}{4}\left\|d - \frac{g0}{L} - \frac{g1}{L}\right\|^2 + \frac{1}{2L}\left\|\frac{g0}{L}\right\|^2 + \frac{1}{2L}\left\|\frac{g1}{L}\right\|^2 \geq 0$$

Introduction to performance estimation

A convex formulation for performance estimation

Performance estimation of standard algorithms

- Gradient method
- Other methods and criteria

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)
- In most cases we can identify the analytical form of a primal solution valid for all N (hence an explicit function f)

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)
- In most cases we can identify the analytical form of a primal solution valid for all N (hence an explicit function f)
- This gives us rigorous lower bounds on the worst-case performance for all N

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)
- In most cases we can identify the analytical form of a primal solution valid for all N (hence an explicit function f)
- This gives us rigorous lower bounds on the worst-case performance for all N

- Dependence on some constants can be derived a priori using homogeneity considerations

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)
- In most cases we can identify the analytical form of a primal solution valid for all N (hence an explicit function f)
- This gives us rigorous lower bounds on the worst-case performance for all N

- Dependence on some constants can be derived a priori using homogeneity considerations
- In several cases we can also identify the analytical form of a dual solution with matching objective function for all $N \rightarrow$ rigorous proof of the worst-case

Performance estimation of standard algorithms

- We compute numerical results from our formulation (for values of N up to a few dozens/hundreds)
- In most cases we can identify the analytical form of a primal solution valid for all N (hence an explicit function f)
- This gives us rigorous lower bounds on the worst-case performance for all N

- Dependence on some constants can be derived a priori using homogeneity considerations
- In several cases we can also identify the analytical form of a dual solution with matching objective function for all $N \rightarrow$ rigorous proof of the worst-case
- If no analytical dual solution: (one-sided) conjectures strongly supported by numerical evidence

Gradient method: analytical results

Exact convergence rates for projected gradient method with step-length γ

$$x_{i+1} = \Pi_Q \left(x_i - \gamma \nabla f(x_i) \right) \qquad \text{(with } 0 < \gamma < \frac{2}{L}\text{)}$$

applied to smooth strongly convex function in $\mathcal{F}_{\mu,L}$ ($\mu > 0$):

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

where $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

Proofs rely on same principle as introductory example (simple but non-intuitive)

Gradient method: analytical results

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- Only previously known in some special cases

Gradient method: analytical results

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- Only previously known in some special cases
- Given μ and L optimal steplength is $\gamma^* = \frac{2}{L+\mu}$ and $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$

Gradient method: analytical results

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- Only previously known in some special cases
- Given μ and L optimal steplength is $\gamma^* = \frac{2}{L+\mu}$ and $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$
- Worst-case functions are 1D quadratics $\frac{\mu}{2}x^2$ and $\frac{L}{2}x^2$

Gradient method: analytical results

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- Only previously known in some special cases
- Given μ and L optimal steplength is $\gamma^* = \frac{2}{L+\mu}$ and $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$
- Worst-case functions are 1D quadratics $\frac{\mu}{2}x^2$ and $\frac{L}{2}x^2$
- Same rates also hold for proximal gradient

Gradient method: analytical results

Linear convergence for all performance criteria, with same rate $\rho = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$

$$\begin{aligned} \|x_k - x_*\|^2 &\leq \rho^k \|x_0 - x_*\|^2 \\ \|\nabla f(x_k)\|^2 &\leq \rho^k \|\nabla f(x_0)\|^2 \\ f(x_k) - f(x_*) &\leq \rho^k \big(f(x_0) - f(x_*)\big) \end{aligned}$$

- Only previously known in some special cases
- Given μ and L optimal steplength is $\gamma^* = \frac{2}{L+\mu}$ and $\rho^* = \left(\frac{L-\mu}{L+\mu}\right)^2$
- Worst-case functions are 1D quadratics $\frac{\mu}{2}x^2$ and $\frac{L}{2}x^2$
- Same rates also hold for proximal gradient
- When $\mu = 0$ we obtain $\rho = 1$, i.e. no convergence, which is tight (!) $\rightarrow$ other convergence results needed to describe convergence

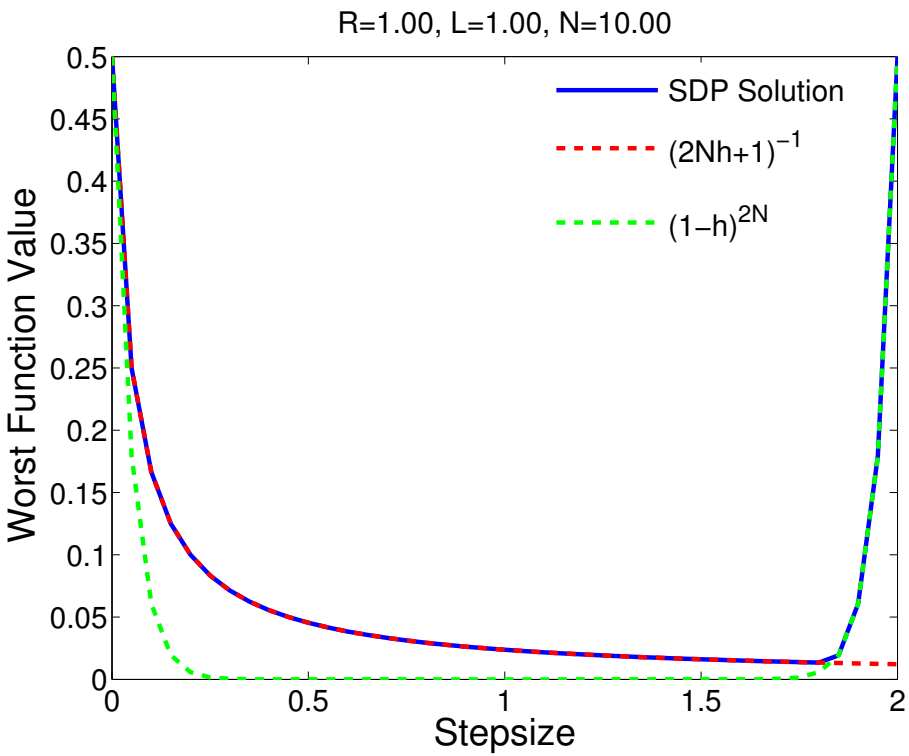
Gradient method: numerical results

Choose $\mu = 0$, criterion $f_N - f^*$ and gradient method with step-size h

$$x_{i+1} = x_i - \frac{h}{L} g_i.$$

Our results match, for all tested $0 < h < 2$ and $1 \leq N \leq 100$

$$\max f(x_N) - f^* = \frac{LR^2}{2} \max\left(\frac{1}{2Nh+1}, (1-h)^{2N}\right)$$



Also conjectured by Drori and Teboulle, 2014

Suggests two worst-case regimes

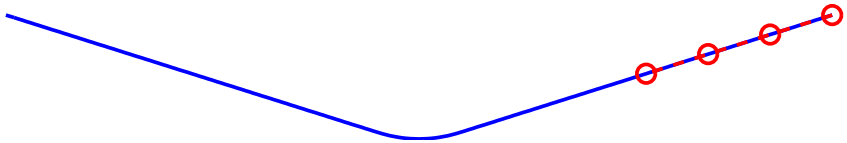
Class theoretical bound for $h = 1$ from literature is

$$\frac{LR^2}{2}\frac{1}{N} \text{ (2× worse)}$$

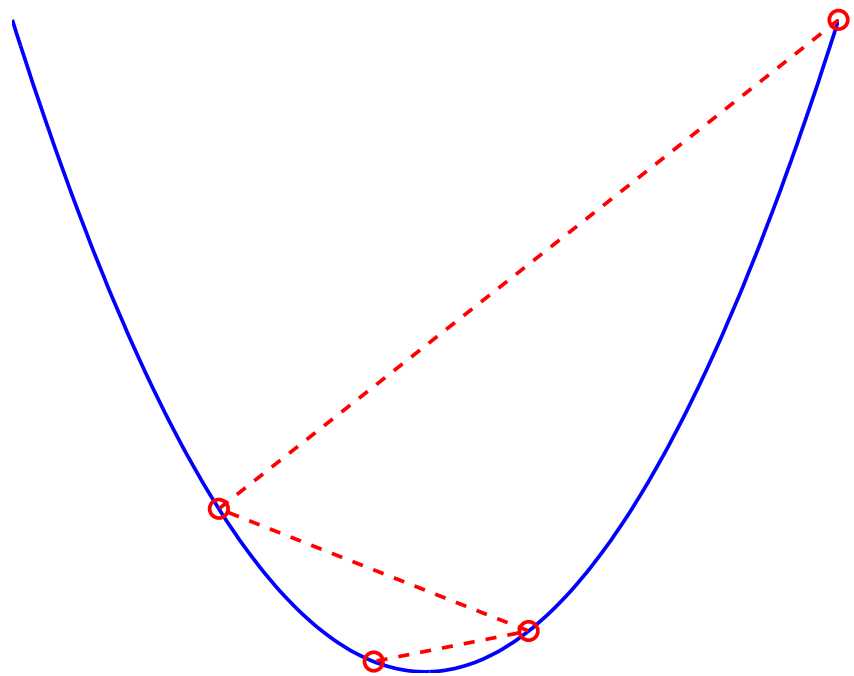
Intuition for each term in the bound

Each term corresponds to an explicit worst-case function
(which one is active depends on h and N)

These are very simple: 1D and piecewise linear-quadratic



Stays on linear part until last iteration



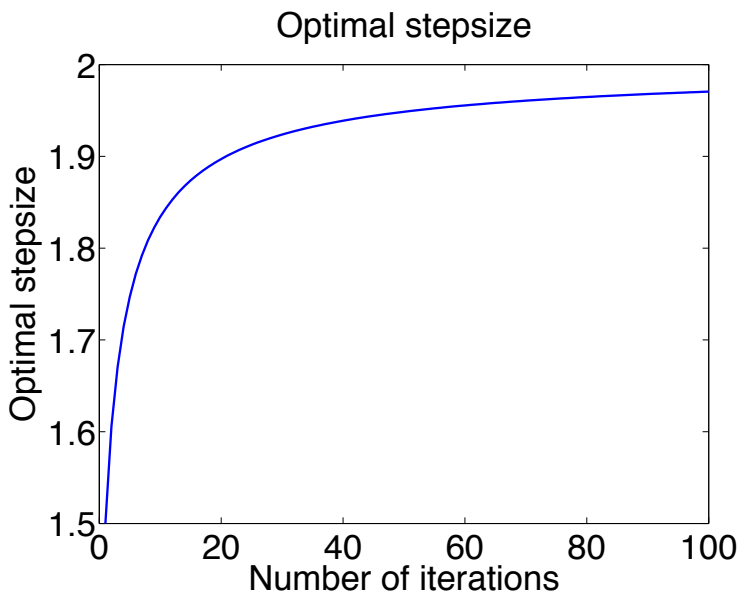
Purely quadratic, controlled overshooting at each iteration

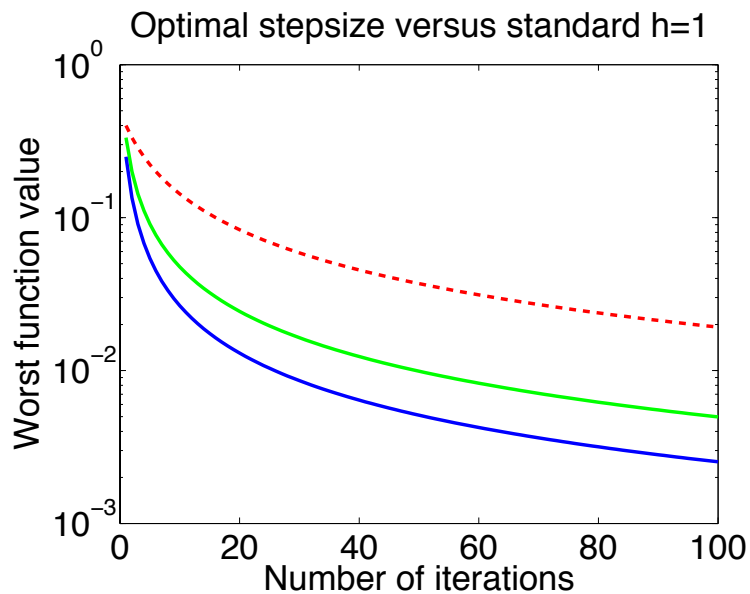
Optimal gradient step-size

Using the conjectured worst-case, compute optimal $h_{\text{opt}}(N)$

$$2-\frac{\log 4N}{2N} \sim 1+(1+4N)^{-1/(2N)} \leq h_{\text{opt}}(N) \leq 1+(1+2N)^{-1/(2N)} \sim 2-\frac{\log 2N}{2N}.$$

(equalize both terms in bounds, no closed-form solution)





(red: $h = 1$ classic bound, green: $h = 1$ exact bound, blue: optimal h exact bound)

Optimal step-size tends quickly to 2

Exact bound for h_{opt} approximately twice better than for $h = 1$

A few words about numerics

Problems solved with YALMIP+MOSEK, verified with interval-arithmetic VSDP toolbox

N	h_{opt}	Conjecture	DT relaxation	Rel. error	SDP-PEP	Rel. error
1	1.5000	$LR^2/8.00$	$LR^2/8.00$	0.00	$LR^2/8.00$	7e-09
2	1.6058	$LR^2/14.85$	$LR^2/14.54$	2e-02	$LR^2/14.85$	5e-09
5	1.7471	$LR^2/36.94$	$LR^2/32.57$	1e-01	$LR^2/36.94$	1e-08
10	1.8341	$LR^2/75.36$	$LR^2/59.80$	3e-01	$LR^2/75.36$	3e-08
20	1.8971	$LR^2/153.77$	$LR^2/109.58$	4e-01	$LR^2/153.77$	6e-08
30	1.9238	$LR^2/232.85$	$LR^2/156.23$	5e-01	$LR^2/232.85$	7e-08
40	1.9388	$LR^2/312.21$	$LR^2/201.10$	6e-01	$LR^2/312.21$	3e-08
50	1.9486	$LR^2/391.72$	$LR^2/244.70$	6e-01	$LR^2/391.72$	1e-07
100	1.9705	$LR^2/790.22$	$LR^2/451.72$	7e-01	$LR^2/790.22$	1e-07

Table : Gradient Method with $\mu = 0$, worst-case computed with relaxation from Drori and Teboulle and worst-case obtained by exact formulation (SDP-PEP) for the criterion $f(x_N) - f^*$. Error is measured relatively to the conjectured result. Results obtained with MOSEK.

More results for gradient method

- Strongly convex case, with condition number $\kappa = \mu/L$

$$\max\ f(x_N) - f_* = \frac{LR^2}{2} \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-2N}}, (1-h)^{2N}\right)$$

More results for gradient method

- Strongly convex case, with condition number $\kappa = \mu/L$

$$\max\ f(x_N) - f_* = \frac{LR^2}{2} \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-2N}}, (1-h)^{2N}\right)$$

- Residual gradient norm $\|\nabla f(x_N)\|$, smooth case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{1}{Nh+1}, |1-h|^N\right)$$

More results for gradient method

- Strongly convex case, with condition number $\kappa = \mu/L$

$$\max\ f(x_N) - f_* = \frac{LR^2}{2} \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-2N}}, (1-h)^{2N}\right)$$

- Residual gradient norm $\|\nabla f(x_N)\|$, smooth case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{1}{Nh+1}, |1-h|^N\right)$$

- Residual gradient norm, strongly convex case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-N}}, |1-h|^N\right)$$

More results for gradient method

- Strongly convex case, with condition number $\kappa = \mu/L$

$$\max\ f(x_N) - f_* = \frac{LR^2}{2} \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-2N}}, (1-h)^{2N}\right)$$

- Residual gradient norm $\|\nabla f(x_N)\|$, smooth case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{1}{Nh+1}, |1-h|^N\right)$$

- Residual gradient norm, strongly convex case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-N}}, |1-h|^N\right)$$

- Simple $1D$ piecewise linear-quadratic solutions in all cases (different for each case)

More results for gradient method

- Strongly convex case, with condition number $\kappa = \mu/L$

$$\max\ f(x_N) - f_* = \frac{LR^2}{2} \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-2N}}, (1-h)^{2N}\right)$$

- Residual gradient norm $\|\nabla f(x_N)\|$, smooth case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{1}{Nh+1}, |1-h|^N\right)$$

- Residual gradient norm, strongly convex case

$$\max\ \|\nabla f(x_N)\|_2 = LR \max\left(\frac{\kappa}{(\kappa-1)+(1-\kappa h)^{-N}}, |1-h|^N\right)$$

- Simple $1D$ piecewise linear-quadratic solutions in all cases (different for each case)
- All results lead to optimal step-sizes

Nesterov's accelerated gradient method

Belongs to the class of fixed-step methods
Algorithm:

Initialization: $x_1 = x_0 - \frac{g_0}{L}$, $t_1 = 1$:

For $i = 1 : N - 2$

$$t_{i+1} = \frac{1 + \sqrt{1 + 4t_i^2}}{2}$$

$$x_{i+1} = x_i - \left(1 + \frac{t_i - 1}{t_{i+1}}\right) \frac{g_i}{L} + \frac{t_i - 1}{t_{i+1}} (x_i - x_{i-1}) + \frac{t_i - 1}{t_{i+1}} \frac{g_{i-1}}{L}$$

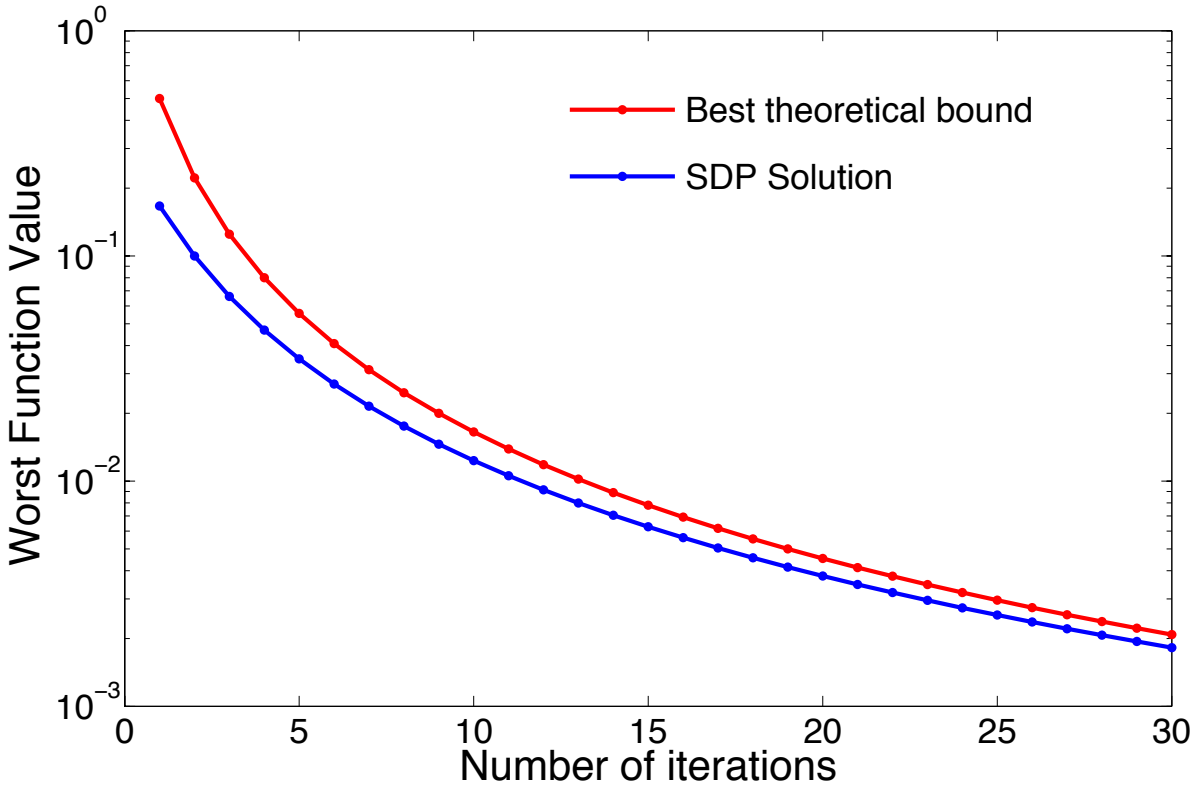
Termination: $x_N = x_{N-1} - \frac{g_{N-1}}{L}$

(nonstandard description, obtained from standard two-sequence algorithm)

Nesterov's accelerated gradient method

Accelerated gradient with $L = 1$, $R = 1$ and $\mu = 0$

Known theoretical bound: $f_N - f^* \leq \frac{2LR^2}{(N+1)^2}$



Relatively modest improvement ($\approx 15\%$ better)

Similar results for the recent optimized method of Kim and Fessler

Smallest gradient norm among all iterates

- Dual methods care about gradient norm ($\leftarrow$ primal feasibility)

Smallest gradient norm among all iterates

- Dual methods care about gradient norm ($\leftarrow$ primal feasibility)
- For accelerated gradient methods, objective function accuracy is $\mathcal{O}(\frac{1}{k^2})$ but norm of residual gradient $\|\nabla f(x_N)\|$ is only $\mathcal{O}(\frac{1}{k})$

Smallest gradient norm among all iterates

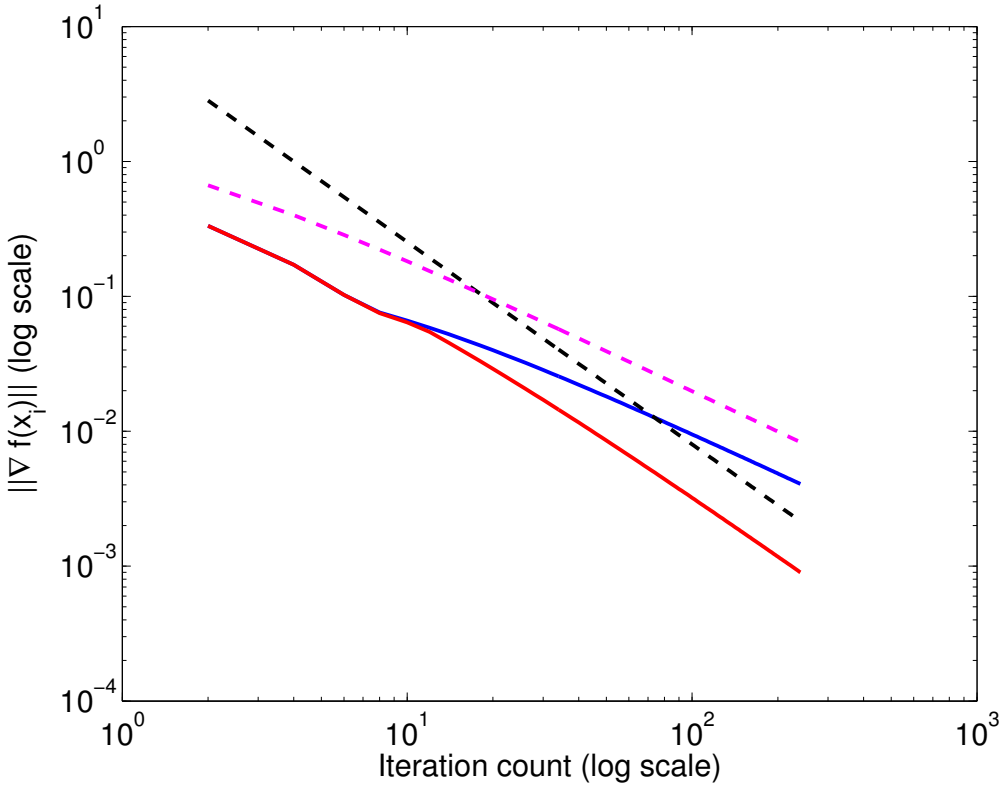
- Dual methods care about gradient norm ($\leftarrow$ primal feasibility)
- For accelerated gradient methods, objective function accuracy is $\mathcal{O}(\frac{1}{k^2})$ but norm of residual gradient $\|\nabla f(x_N)\|$ is only $\mathcal{O}(\frac{1}{k})$
- However this norm is not necessarily decreasing monotonically!

Smallest gradient norm among all iterates

- Dual methods care about gradient norm ($\leftarrow$ primal feasibility)
- For accelerated gradient methods, objective function accuracy is $\mathcal{O}(\frac{1}{k^2})$ but norm of residual gradient $\|\nabla f(x_N)\|$ is only $\mathcal{O}(\frac{1}{k})$
- However this norm is not necessarily decreasing monotonically!
- We can compute worst-case performance for $\min_{0\leq i\leq N} ||\nabla f(x_i)||_2^2$ which is still semidefinite-representable

Smallest gradient norm among all iterates

- Dual methods care about gradient norm ($\leftarrow$ primal feasibility)
- For accelerated gradient methods, objective function accuracy is $\mathcal{O}(\frac{1}{k^2})$ but norm of residual gradient $\|\nabla f(x_N)\|$ is only $\mathcal{O}(\frac{1}{k})$
- However this norm is not necessarily decreasing monotonically!
- We can compute worst-case performance for $\min_{0\leq i\leq N} ||\nabla f(x_i)||_2^2$ which is still semidefinite-representable



Suggests $\mathcal{O}(\frac{1}{k^{3/2}})$ rate for accelerated gradient (red) (previously known only for modified ad hoc method)

Open whether $\mathcal{O}(\frac{1}{k^2})$ achievable by fixed-step method without regularization

(dashed: standard bound ; blue: exact last iterate ; red: exact best iter.)

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation

Key observations:

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps

Key observations:

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization

Key observations:

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization
- linear minimization oracles (i.e. Frank-Wolfe-type methods)

Key observations:

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization
- linear minimization oracles (i.e. Frank-Wolfe-type methods)
- Computes worst-case over all possible convex feasible domains

Key observations:

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization
- linear minimization oracles (i.e. Frank-Wolfe-type methods)
- Computes worst-case over all possible convex feasible domains

Key observations:

- proximal steps can be exactly formulated as

$$x_+ = \mathrm{prox}_L(x) \quad \Leftrightarrow \quad x_+ - g_+ = x \text{ and } g_+ \in \partial f(x_+)$$

  which is a linear condition involving iterates and oracle outputs

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization
- linear minimization oracles (i.e. Frank-Wolfe-type methods)
- Computes worst-case over all possible convex feasible domains

Key observations:

- proximal steps can be exactly formulated as

$$x_+ = \mathrm{prox}_L(x) \quad \Leftrightarrow \quad x_+ - g_+ = x \text{ and } g_+ \in \partial f(x_+)$$

  which is a linear condition involving iterates and oracle outputs
- Projected gradient = proximal gradient with indicator function

Recent extensions of our formulation

We can also handle

- first-order methods for smooth constrained convex optimization i.e. express projection steps in our formulation
- proximal algorithms i.e. express proximal steps
- composite minimization
- linear minimization oracles (i.e. Frank-Wolfe-type methods)
- Computes worst-case over all possible convex feasible domains

Key observations:

- proximal steps can be exactly formulated as

$$x_+ = \mathrm{prox}_L(x) \quad \Leftrightarrow \quad x_+ - g_+ = x \text{ and } g_+ \in \partial f(x_+)$$

  which is a linear condition involving iterates and oracle outputs
- Projected gradient = proximal gradient with indicator function
- New interpolation conditions: indicator functions, functions with bounded subgradients, etc.

Thank you for your attention!

- Worst-case behaviour of first-order methods can be computed exactly using semidefinite optimization
- For any fixed-coefficient first-order method after any given number of iterations on the class of smooth convex objective with given parameters (smoothness and strong convexity)
- Methodology provides easy-to-check (but not very intuitive) proofs and explicit examples of worst-case functions
- Very flexible: choice of performance criteria (e.g. objective value, gradient norm); can be extended to constrained, nonsmooth, proximal, linear minimization oracle settings

Thank you for your attention!

Paper (unconstrained case):

Smooth strongly convex interpolation and exact worst-case performance of first-order methods, Adrien B. Taylor, Julien M. Hendrickx, François Glineur, Mathematical Programming, Springer, May 2016, dx.doi.org/10.1007/s10107-016-1009-3

Preprint (constrained and proximal case):

http://arxiv.org/abs/1512.07516

MATLAB toolbox for performance estimation of first-order methods (in the standard and composite cases):
http://perso.uclouvain.be/adrien.taylor/

Analytical results for gradient method contained in a preprint that will appear soon on arxiv

Use of PEP in practice (1)

How to use the toolbox for projected gradient methods ?

Problem settings

```
>> P.L=1; P.mu=0; P.R=1; P.Proj=1;
```

Algorithm settings

```
>> A.name='GM'; A.N=5; A.stepsize=1;
```

Criterion choice: $f_N - f_*$

```
>> C.name='Obj';
```

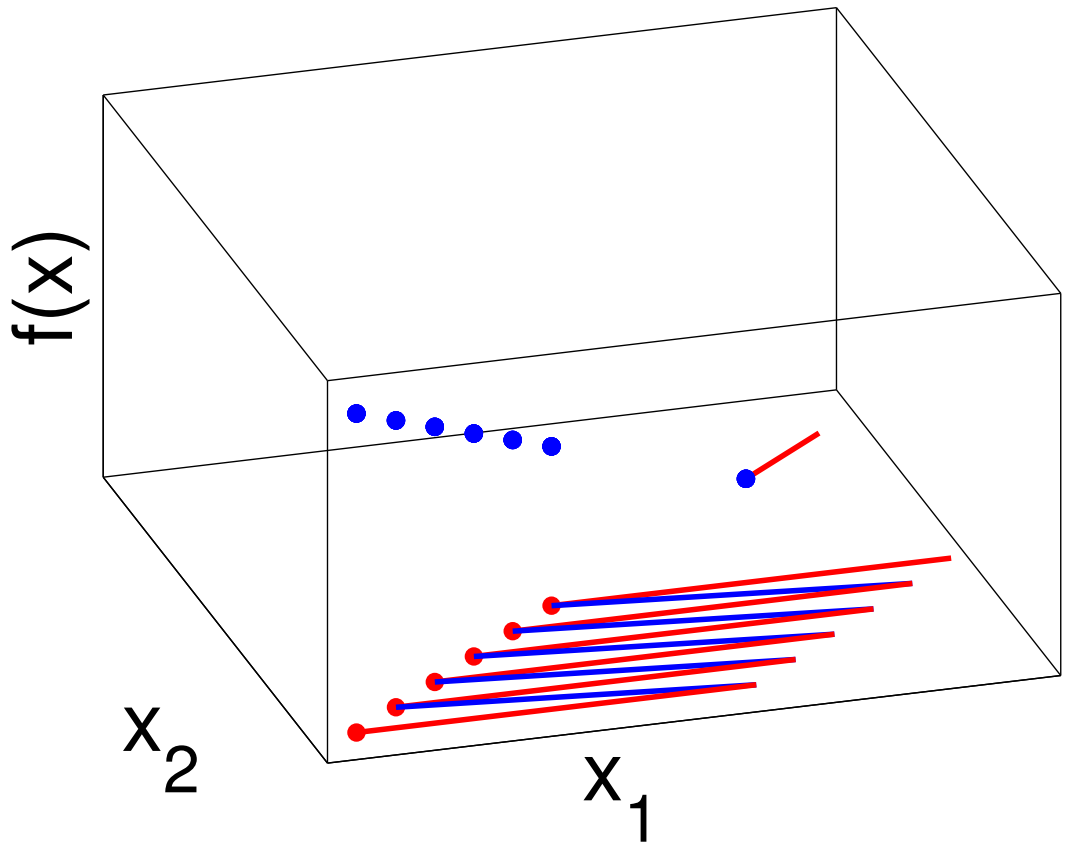
Choice of the solver, default solver settings

```
>> S.solver='mosek';
```

Solve the PEP

```
>> [val, Sol, Prob]=pep_fixedsteps(P,A,C,S);
```

Use of PEP in practice (2)



- Red lines: explicit gradient steps,
- Blue lines: projection steps.