Computational NSO

(NonSmooth Optimization)

a tutorial focusing on bundle methods

Claudia Sagastizábal

(visiting researcher)

mailto:sagastiz@impa.br, http://www.impa.br/~sagastiz

SPCOM Tutorial, Adelaide, Feb 8th, 2015

CONTENTS

Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where f is convex but not differentiable at some points

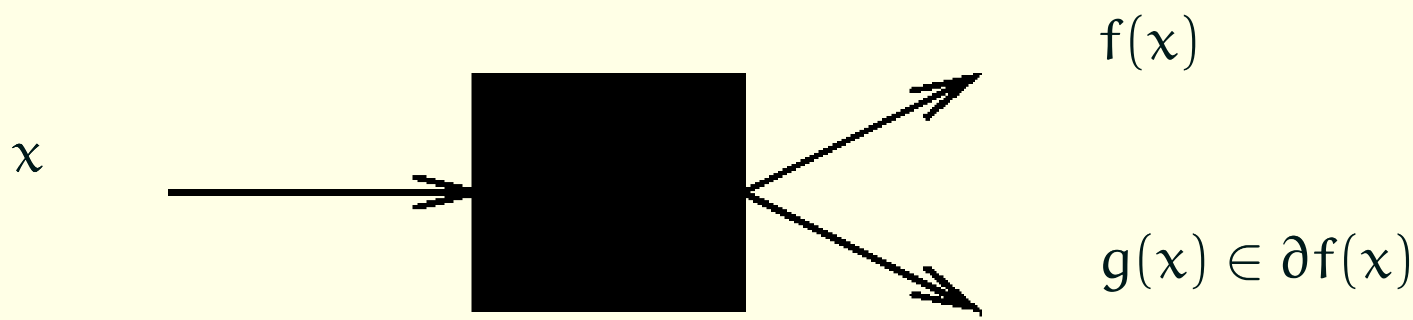
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where f is convex but not differentiable at some points,

we shall define **algorithms** based on information provided by an oracle or “black box”



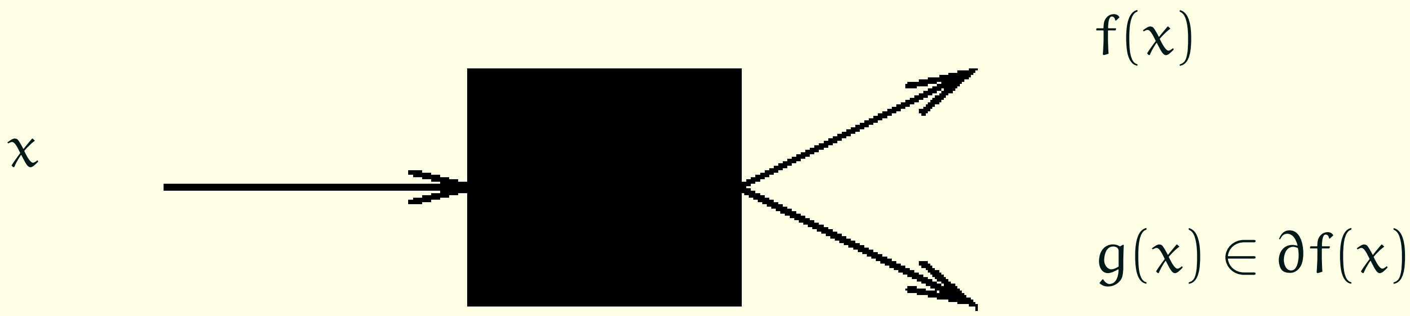
Computational NSO: what does it mean?

For the unconstrained problem

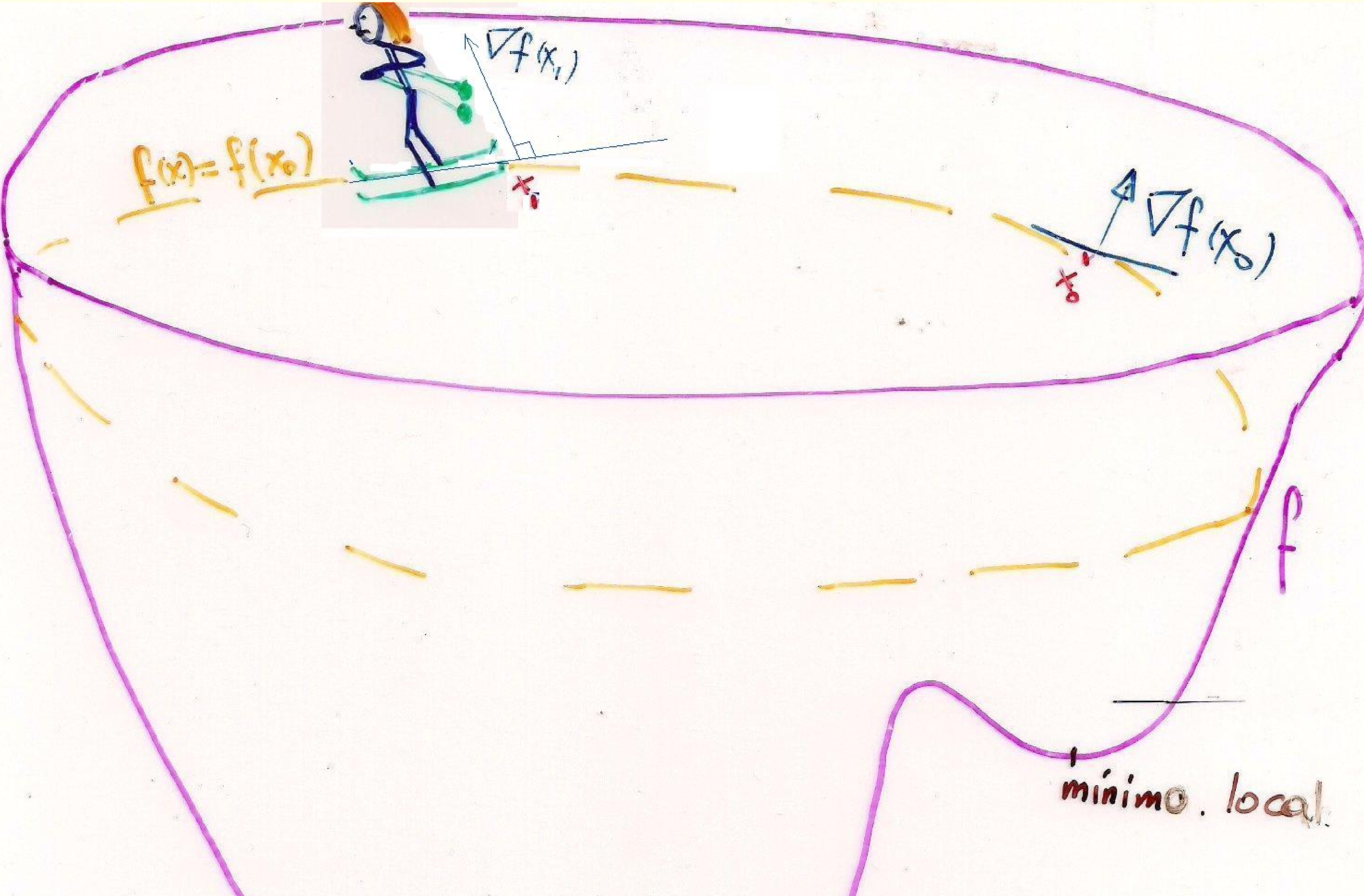
$$\min f(x),$$

where f is convex but not differentiable at some points,

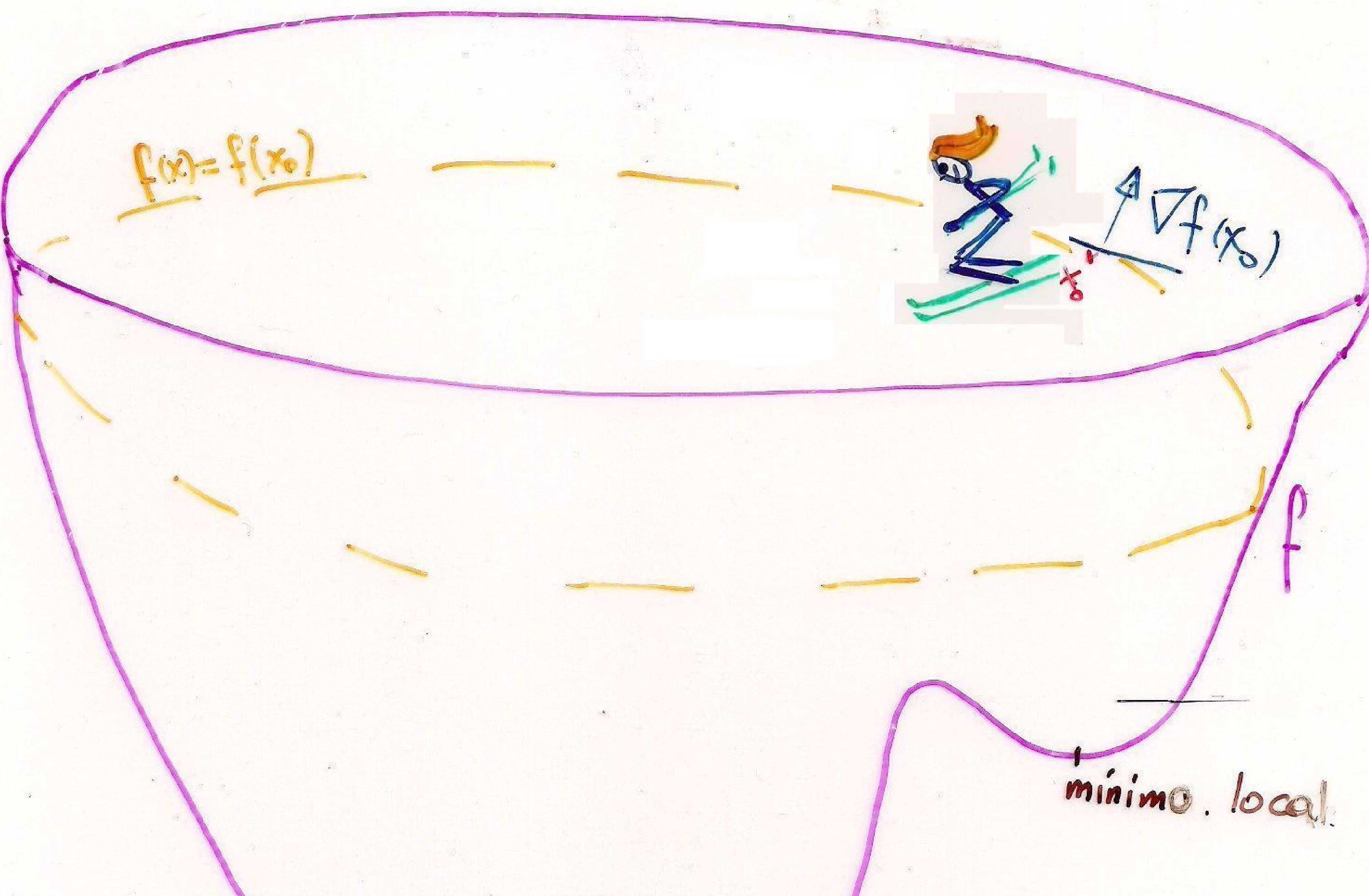
we shall define **algorithms** based on information provided by an oracle or "black box"



Relation with this morning tutorial?

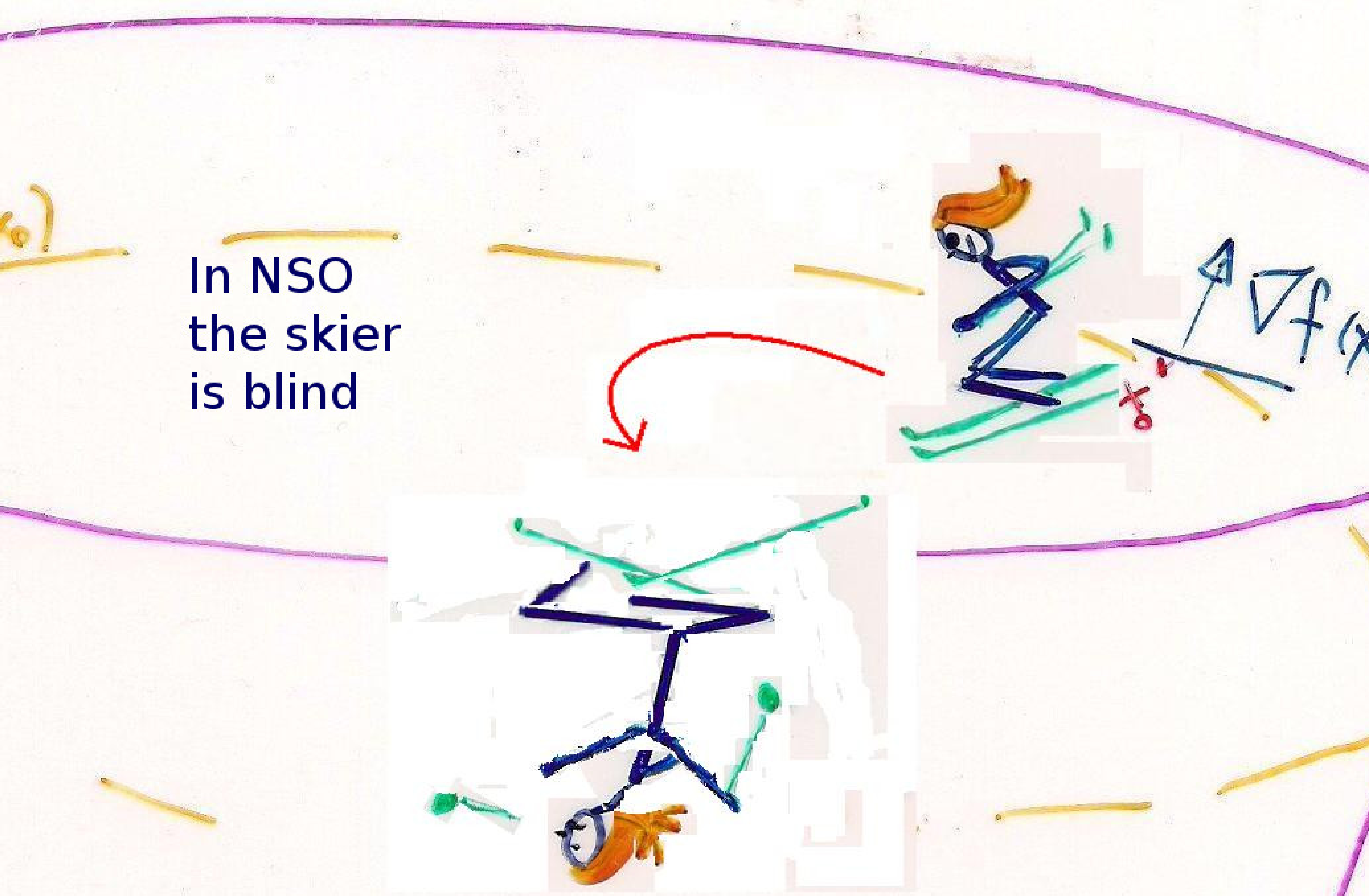


$\nabla f(x_1)$
$f(x)=f(x_0)$
x_0
$\nabla f(x_0)$
x_0
f
mínimo. local.



f(x) = f(x0)
∇f(x0)
x0
f
mínimo . local.

In NSO
the skier
is blind



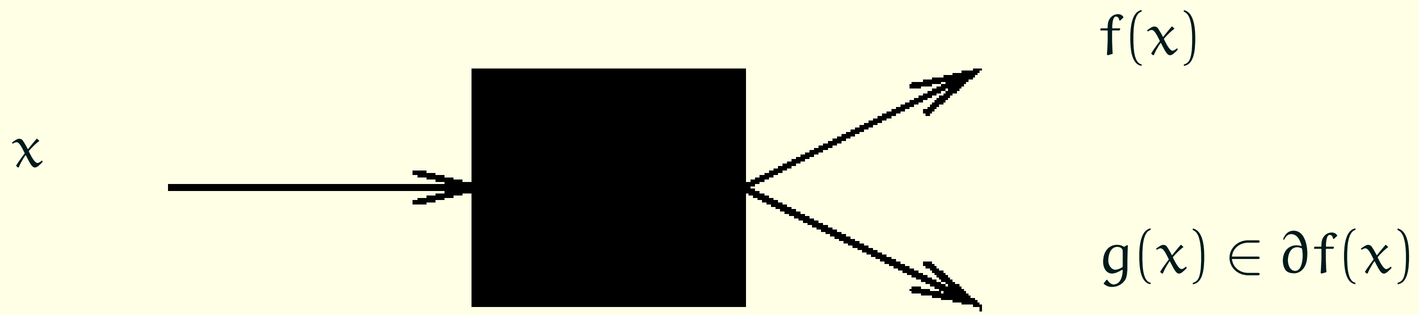
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where f is convex but not differentiable at some points,

we shall define **algorithms** based on information provided by an oracle or "black box"



What do we mean by an algorithm?

An example

source http://comofas.com/

What do we mean by an algorithm?

An example

source http://comofas.com/

What do we mean by an algorithm?

An example

source `http://comofas.com/`

What do we mean by an algorithm?

An example

source http://comofas.com/

What do we mean by an algorithm?

An example

source `http://comofas.com/`

What do we mean by an algorithm?

An example

source http://comofas.com/

What do we mean by an algorithm?

An example

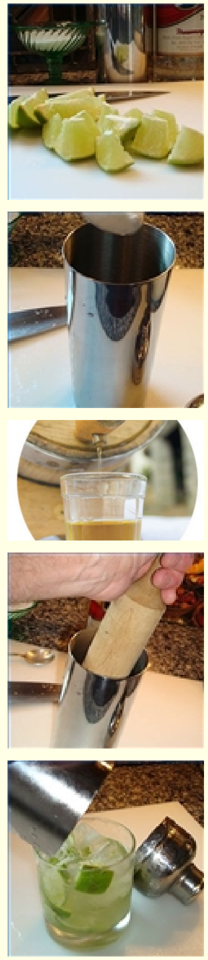
source http://comofas.com/

What do we mean by an algorithm?

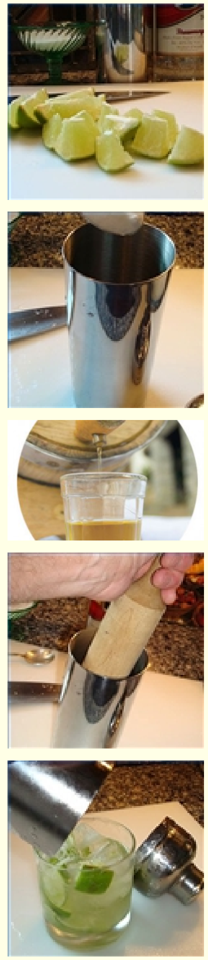
An example

source `http://comofas.com/`

What do we mean by an algorithm?

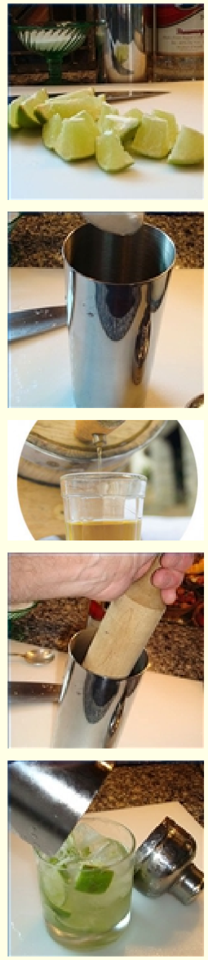


What do we mean by an algorithm?



repeat until . . . ??

What do we mean by an algorithm?



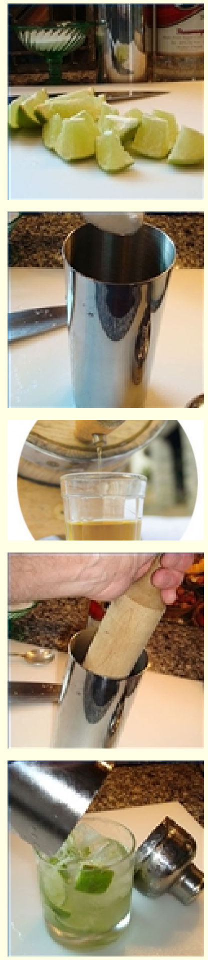
An algorithm

is a sequence of steps

that are repeated

until satisfaction

What do we mean by an algorithm?



An algorithm

is a sequence of steps

that are repeated

until satisfaction

of a stopping test

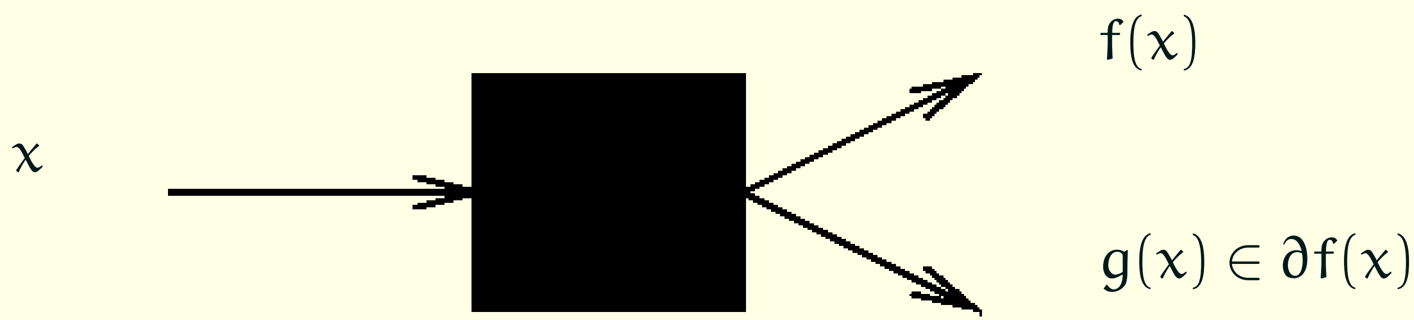
Back to Computational NSO

For the unconstrained problem

$$\min f(x),$$

where f is convex but not differentiable at some points,

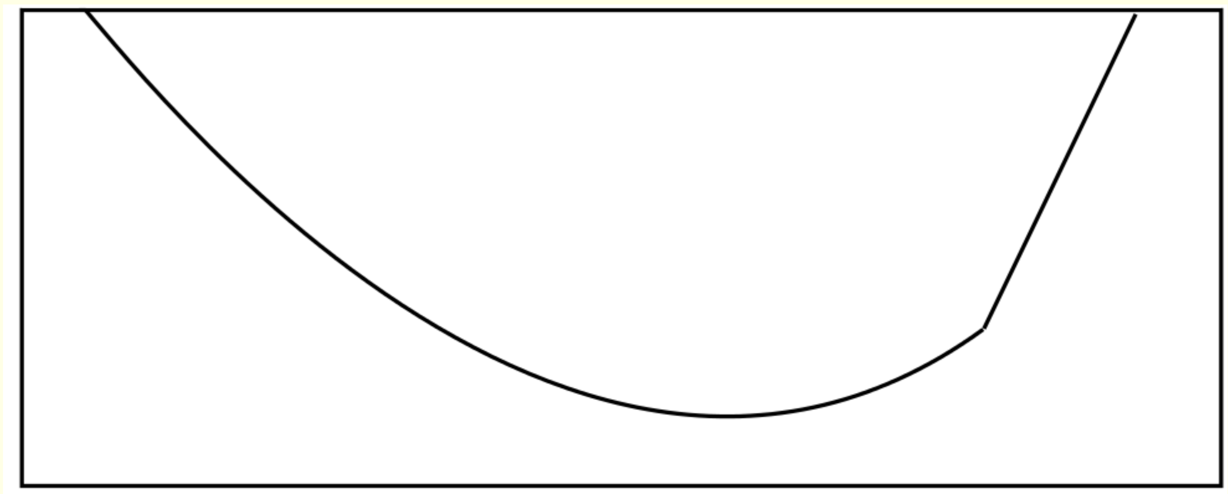
we look for algorithms based on information provided by an oracle or "black box"



endowed with reliable **stopping tests**

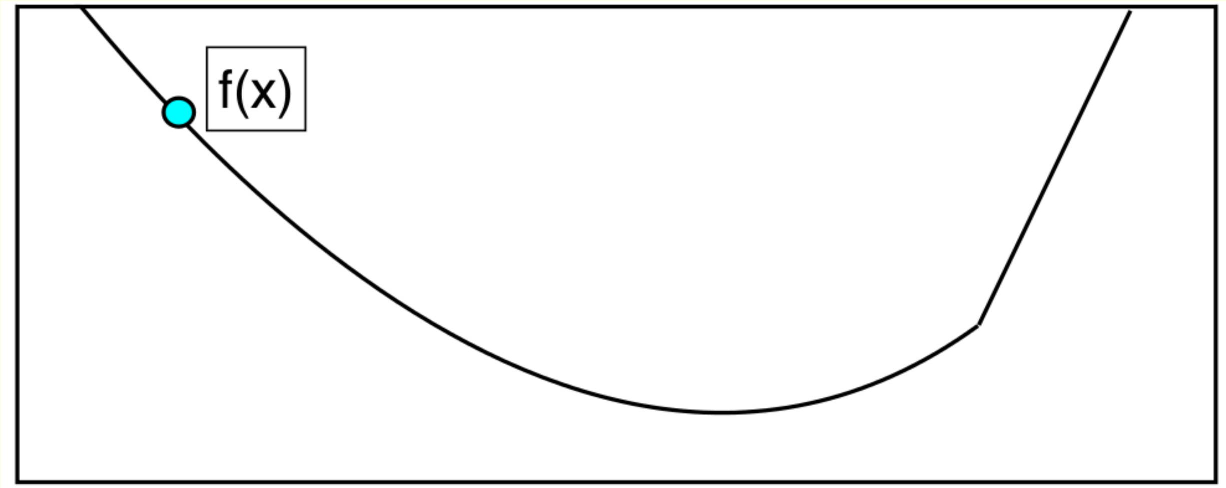
A quick overview of Convex Analysis

An example of a convex nonsmooth function



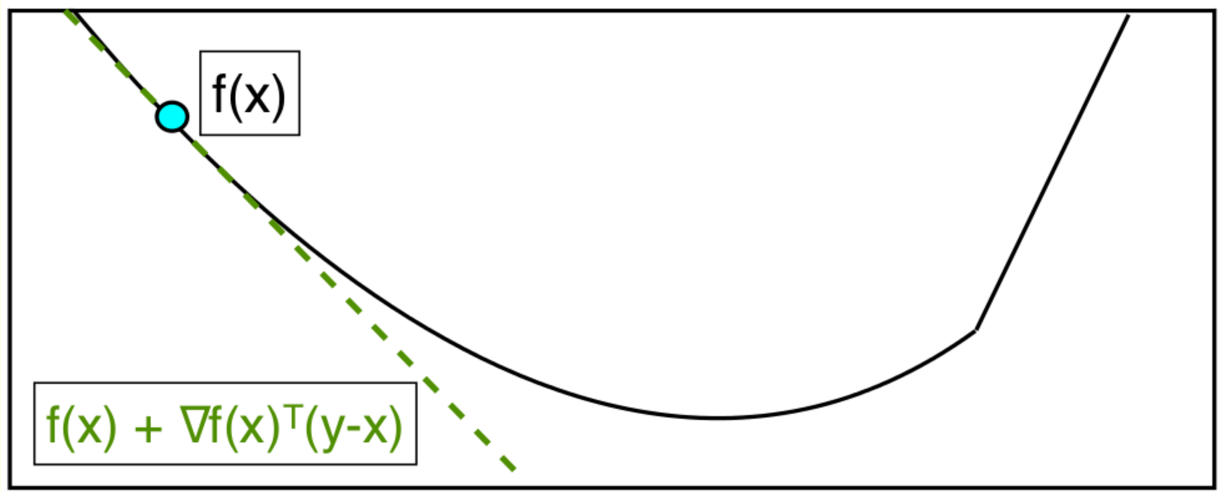
A quick overview of Convex Analysis

An example of a convex nonsmooth function



A quick overview of Convex Analysis

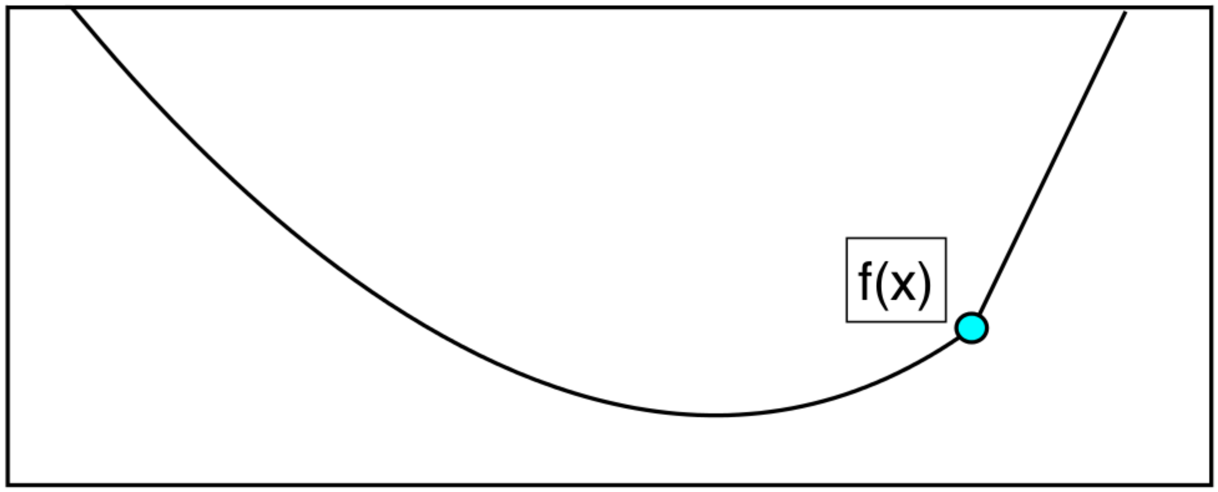
An example of a convex nonsmooth function



$$\begin{aligned} \partial f(x) &= \{\nabla f(x)\} \\ &= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\} \end{aligned}$$

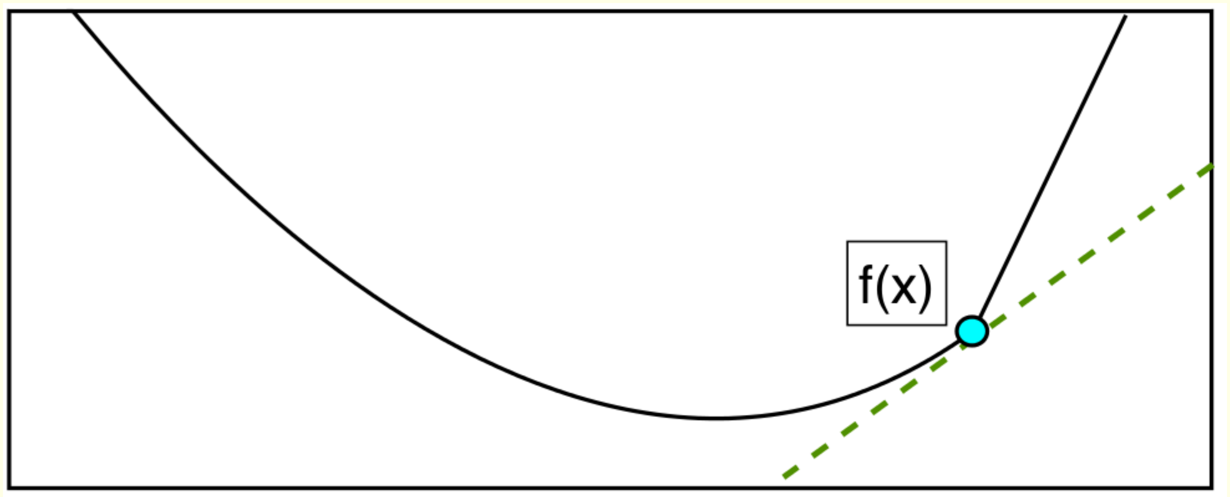
A quick overview of Convex Analysis

An example of a convex nonsmooth function



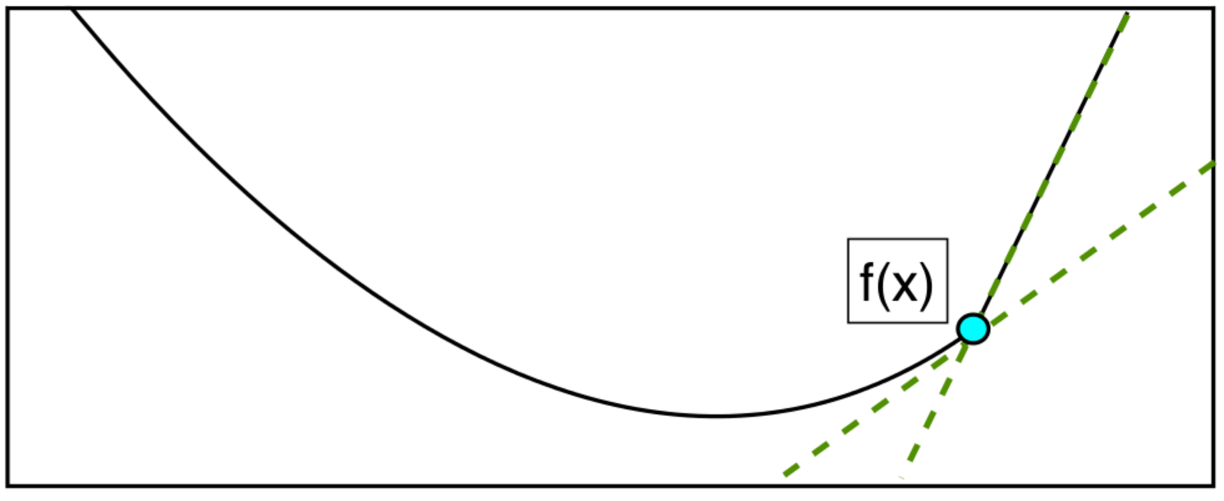
A quick overview of Convex Analysis

An example of a convex nonsmooth function



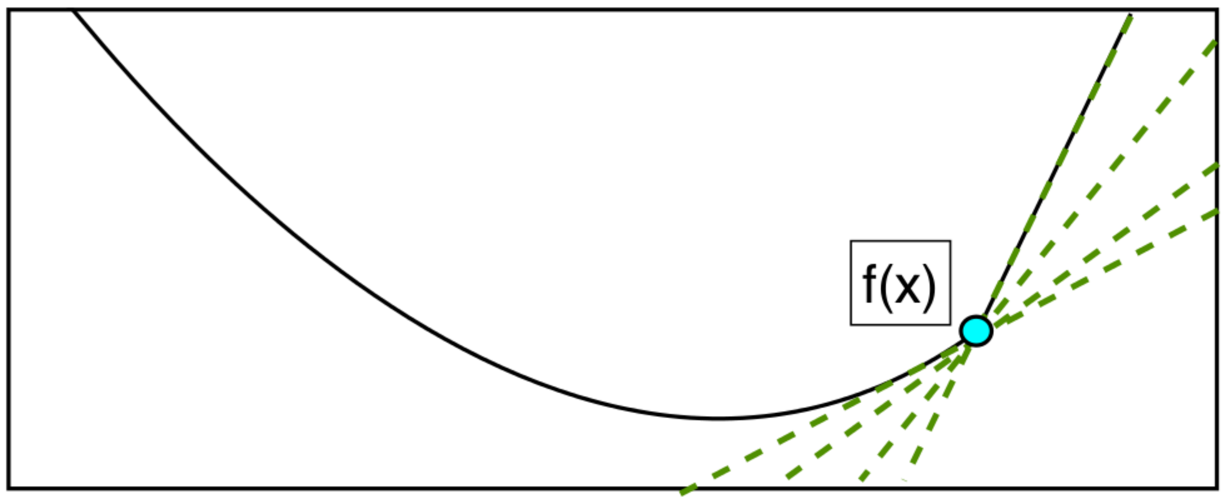
A quick overview of Convex Analysis

An example of a convex nonsmooth function



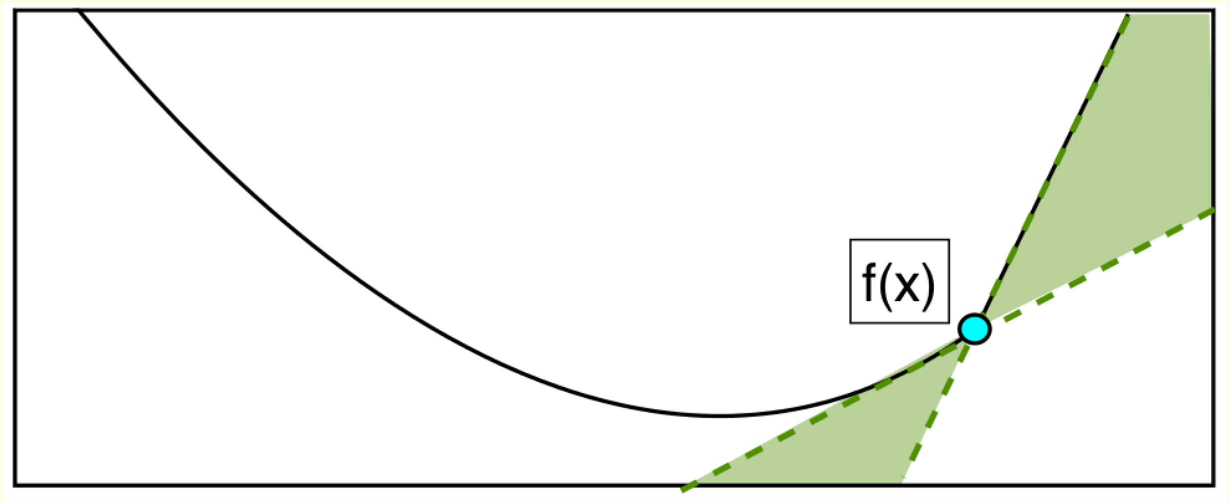
A quick overview of Convex Analysis

An example of a convex nonsmooth function



A quick overview of Convex Analysis

An example of a convex nonsmooth function

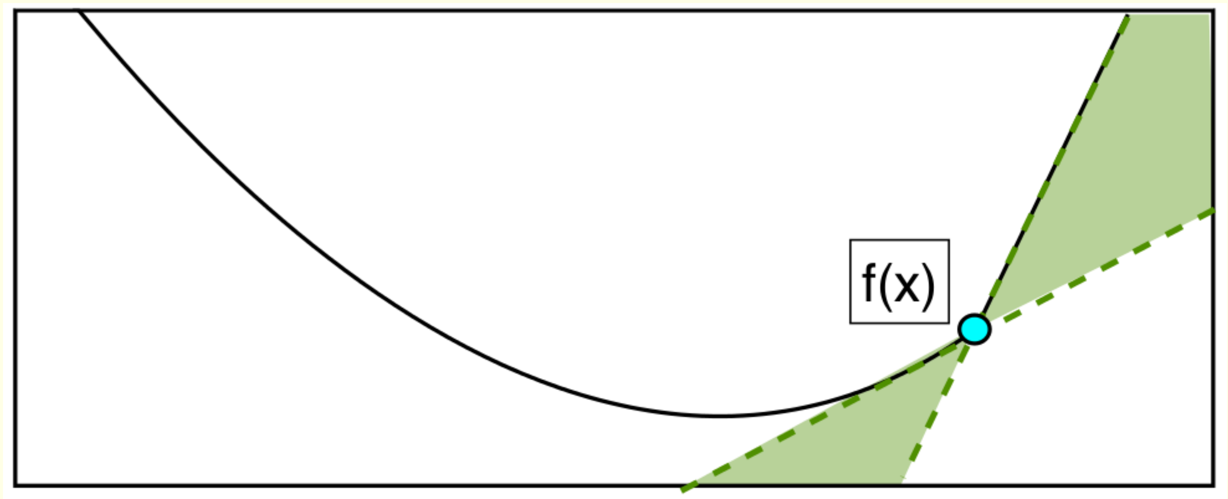


$$\partial f(x) \quad = \quad \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y\}$$

f x

A quick overview of Convex Analysis

An example of a convex nonsmooth function

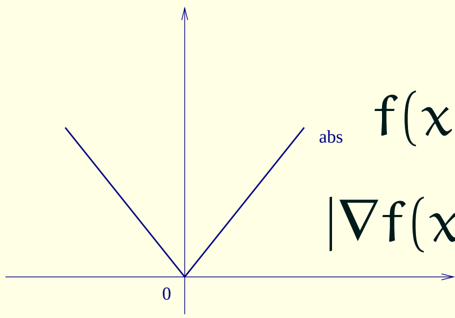


$$\partial f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y\}$$

$$= \{\text{slopes of linearizations supporting } f\text{, tangent at } x\}$$

Why special NSO methods?

Smooth optimization methods **do not work**



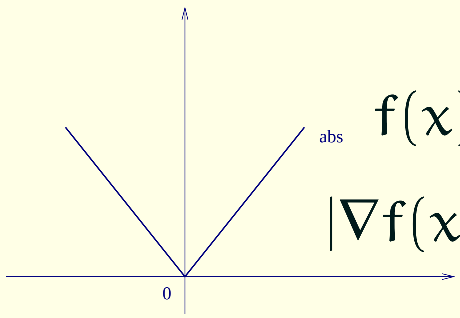
$$f(x) = |x|$$

$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Why special NSO methods?

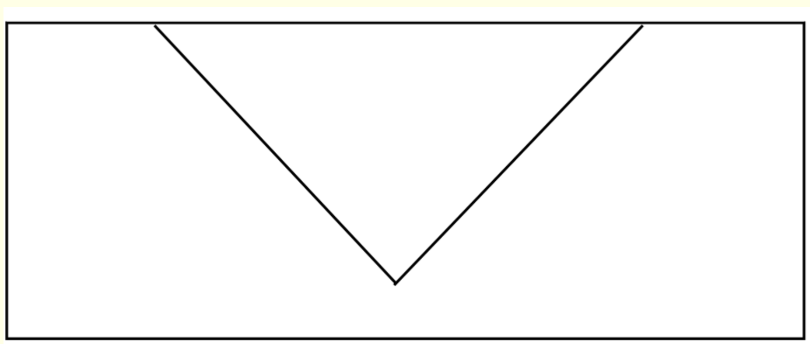
Smooth optimization methods **do not work**



$$f(x) = |x|$$

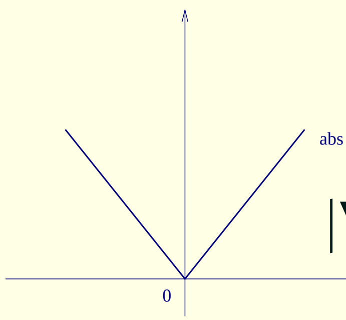
$$|\nabla f(x^k)| = 1, \; \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq \text{TOL}$ $\quad (\leftrightarrow |g(x^k)| \leq \text{TOL})$



Why special NSO methods?

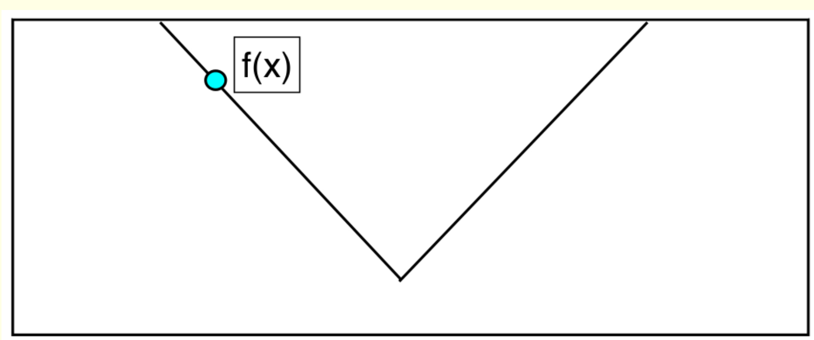
Smooth optimization methods **do not work**



$$f(x) = |x|$$

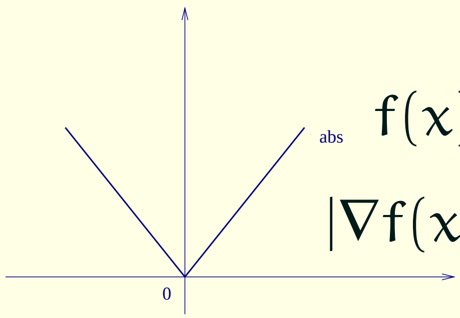
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

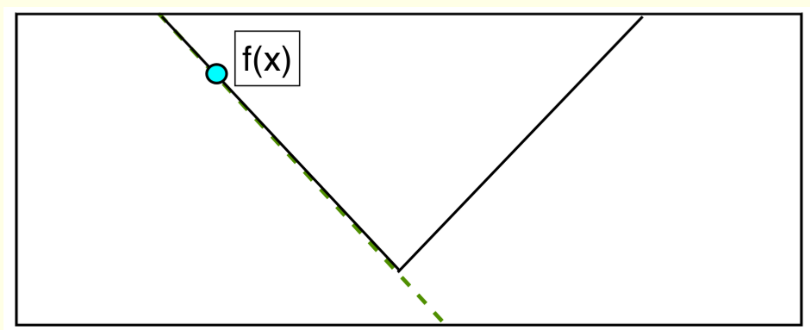
Smooth optimization methods **do not work**



$$f(x) = |x|$$

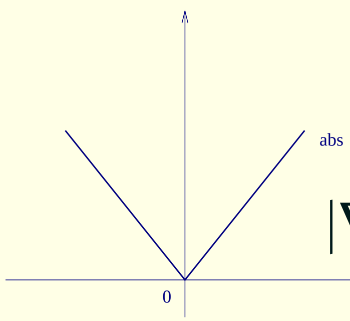
$$|\nabla f(x^k)| = 1, \ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

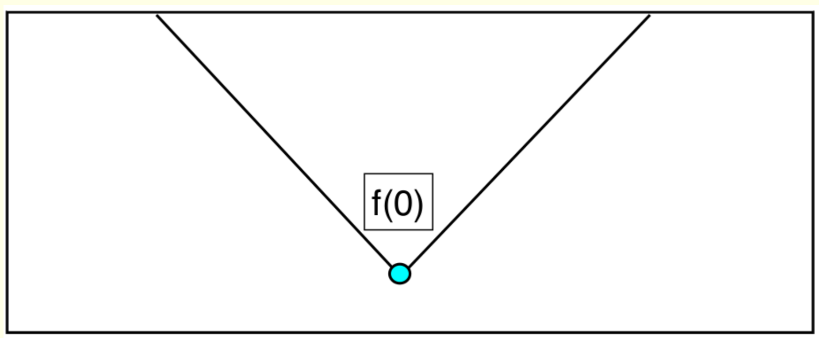
Smooth optimization methods **do not work**



$$f(x) = |x|$$

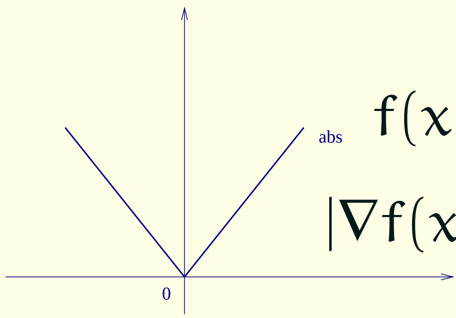
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

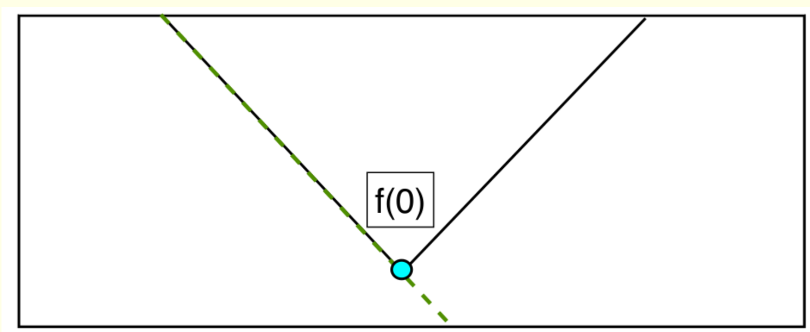
Smooth optimization methods **do not work**



$$f(x) = |x|$$

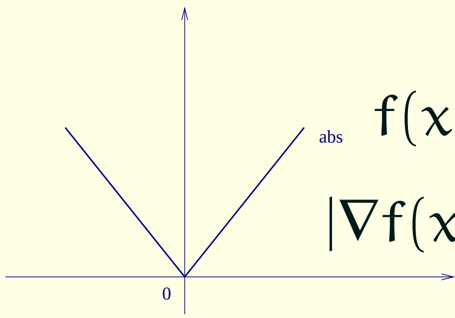
$$|\nabla f(x^k)| = 1, \ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

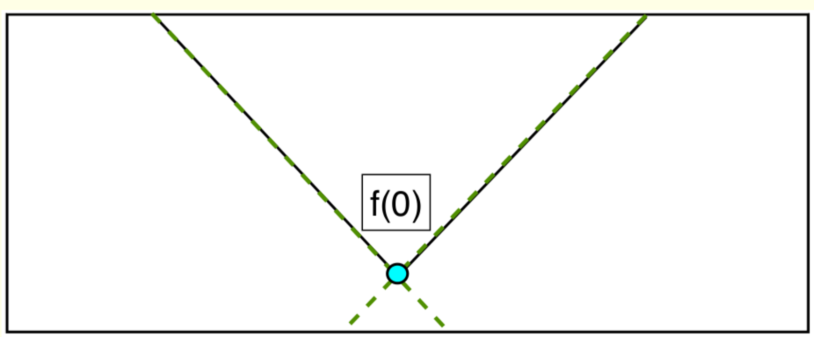
Smooth optimization methods **do not work**



$$f(x) = |x|$$

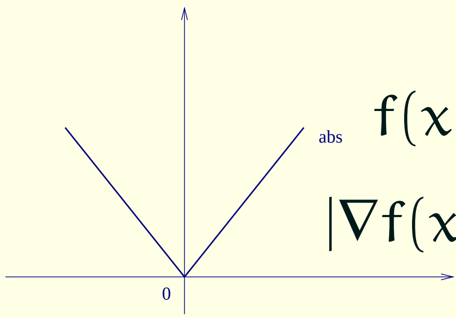
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

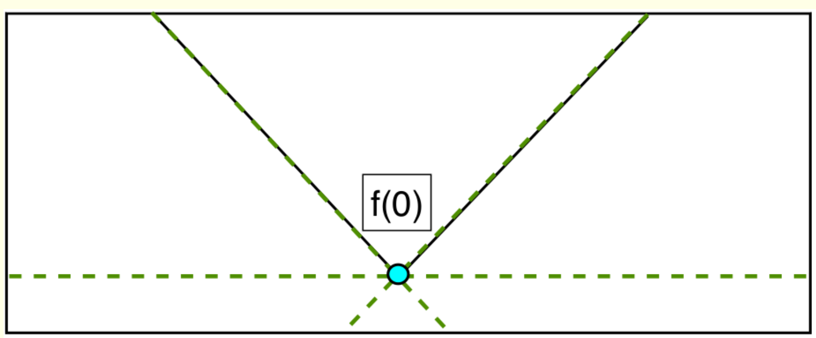
Smooth optimization methods **do not work**



$$f(x) = |x|$$

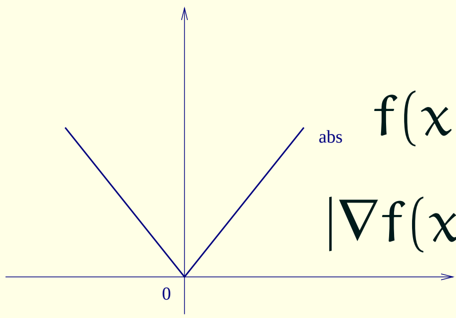
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq \text{TOL}$ $\quad (\leftrightarrow |g(x^k)| \leq \text{TOL})$



Why special NSO methods?

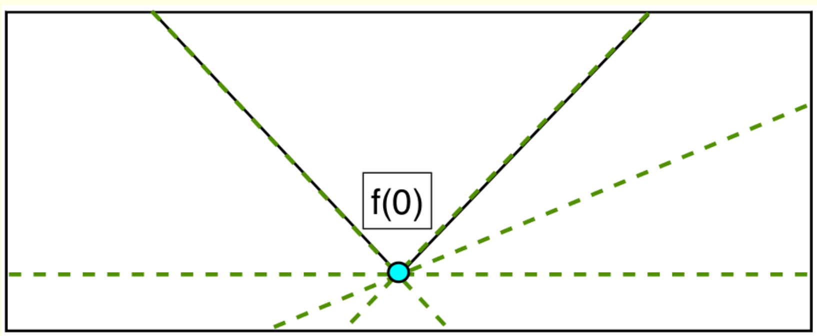
Smooth optimization methods **do not work**



$$f(x) = |x|$$

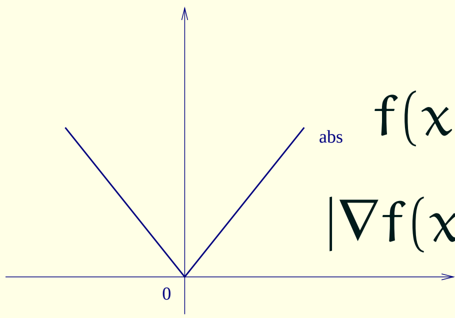
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq \text{TOL}$ $\quad (\leftrightarrow |g(x^k)| \leq \text{TOL})$



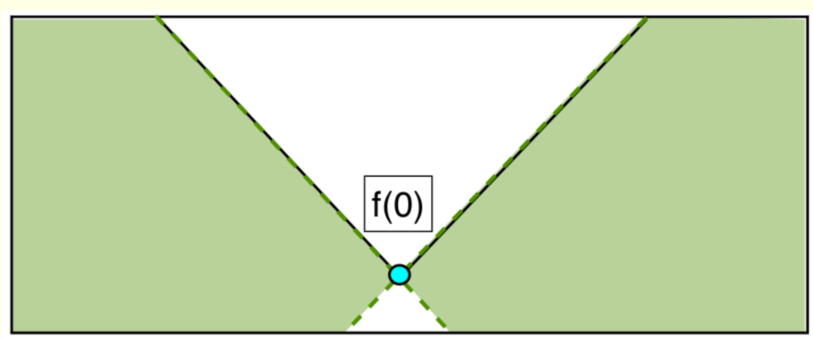
Why special NSO methods?

Smooth optimization methods **do not work**



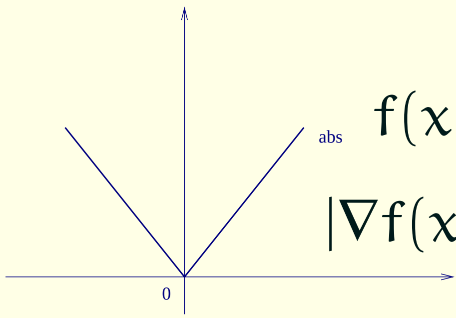
$$
\begin{aligned}
f(x) &= |x| \\
|\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{aligned}
$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$



Why special NSO methods?

Smooth optimization methods **do not work**



$$f(x) = |x|$$

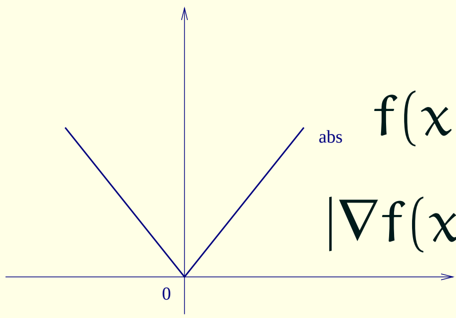
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Finite difference approximations **fail** **(no automatic differentiation)**

Why special NSO methods?

Smooth optimization methods **do not work**



$$f(x) = |x|$$

$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

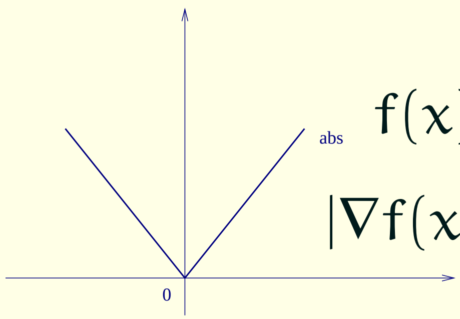
Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

Why special NSO methods?

Smooth optimization methods **do not work**

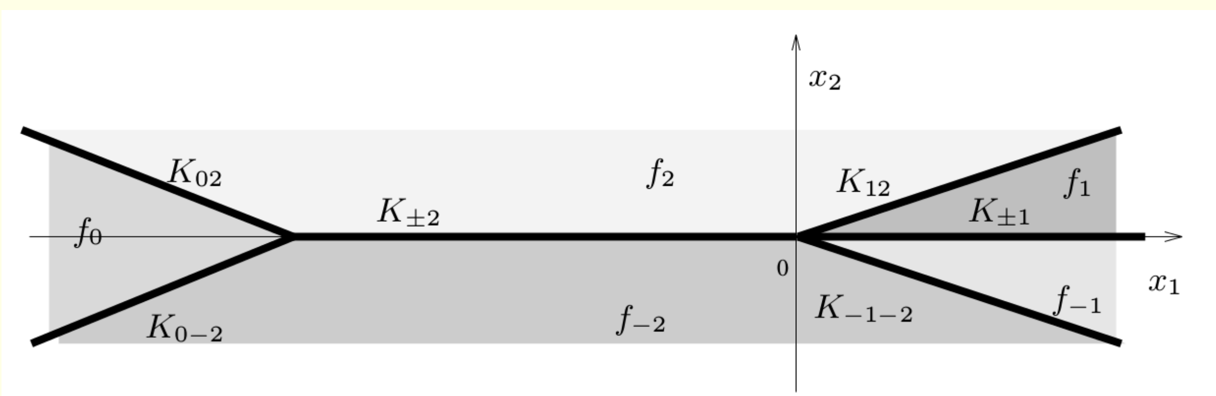


$$f(x) = |x|$$

$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

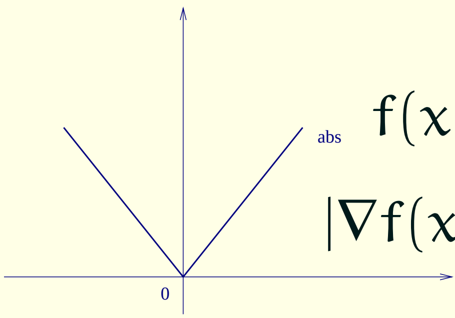
Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Linesearches get trapped in kinks and **fail**



Why special NSO methods?

Smooth optimization methods **do not work**



$$f(x) = |x|$$

$$|\nabla f(x^k)| = 1, \ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$

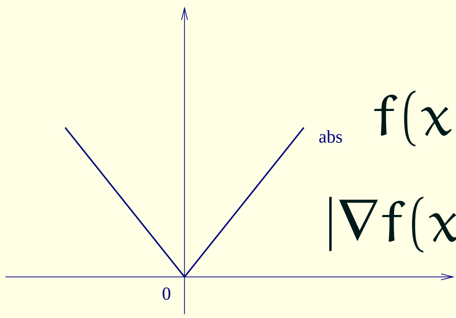
Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

$-g(x^k)$ may **not** provide descent

Why special NSO methods?

Smooth optimization methods **do not work**



$$f(x) = |x|$$

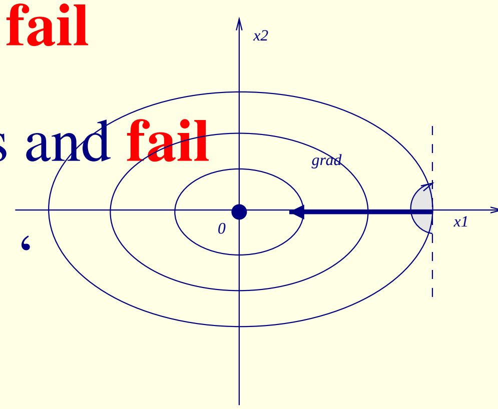
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

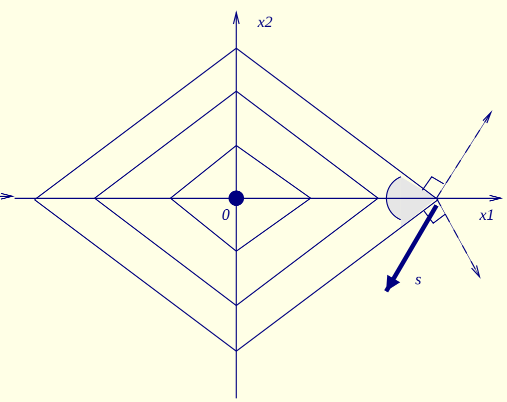
Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\quad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

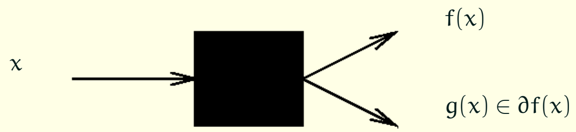
$-g(x^k)$ may **not** provide descent ‘





How is the oracle information used?

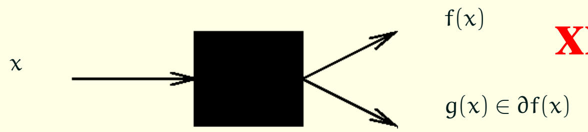
We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

How is the oracle information used?

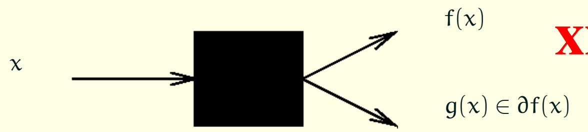
We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

How is the oracle information used?

We look for algorithms based on information provided by an oracle

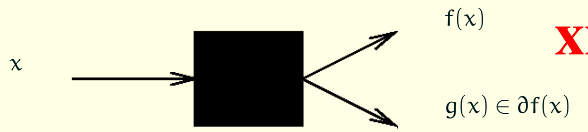


~~endowed with reliable stopping tests~~

Subgradient Methods

How is the oracle information used?

We look for algorithms based on information provided by an oracle



~~endowed with reliable stopping tests~~

Subgradient Methods

0 Choose x^1 and set $k = 1$.

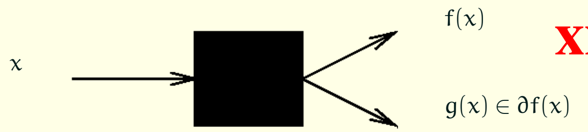
1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

How is the oracle information used?

We look for algorithms based on information provided by an oracle



~~endowed with reliable stopping tests~~

Subgradient Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

Is this a good “recipe”?

Subgradient Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

SG methods are
the algorithmic version
of this road sign

Subgradient Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

SG methods are
the algorithmic version
of this road sign

...something is missing!!!

Subgradient Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

3 Make $k = k + 1$ and loop to 1.

SG methods are

the algorithmic version

of this road sign

not a good recipe

Subgradient Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.

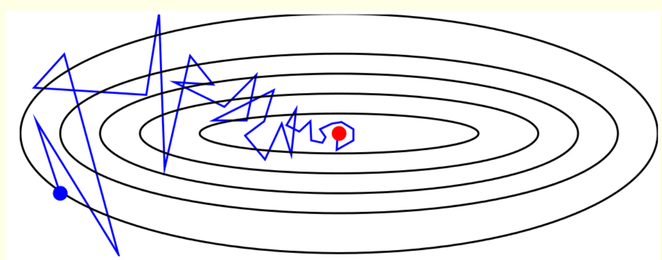
3 Make $k = k + 1$ and loop to 1.

SG methods are

the algorithmic version

of this road sign

not a good recipe



Non-monotone!

Subgradient Methods: why a “not-good” recipe

- Non-monotone functional values, but converges
- because distance to solution set decreases for t_k sufficiently small
- Lacks a stopping test

Subgradient Methods: why a “not-good” recipe

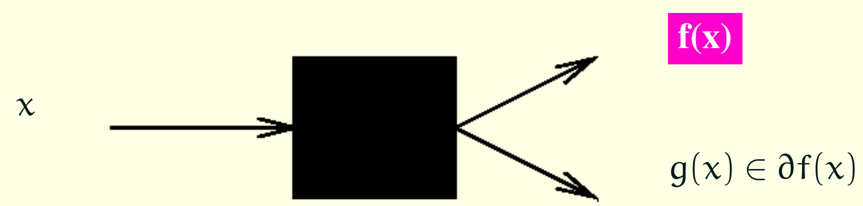
$$\left\{ \begin{array}{l} \text{Non-monotone functional values, but converges} \\ \text{because distance to solution set decreases for } t_k \text{ sufficiently small} \\ \text{Lacks a stopping test} \end{array} \right.$$

…does not use all available information

Subgradient Methods: why a “not-good” recipe

- Non-monotone functional values, but converges
- because distance to solution set decreases for t_k sufficiently small
- Lacks a stopping test

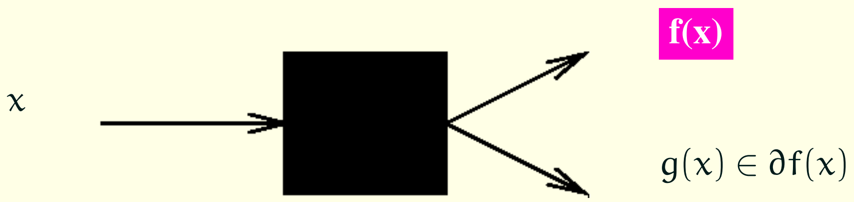
…does not use all **available** information



Subgradient Methods: why a "not-good" recipe

- Non-monotone functional values, but converges
- because distance to solution set decreases for t_k sufficiently small
- Lacks a stopping test

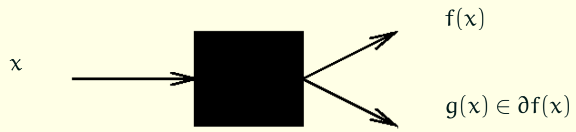
…does not use all **available** information



SG methods are like caipirinha without cachaça

How is the oracle information used?

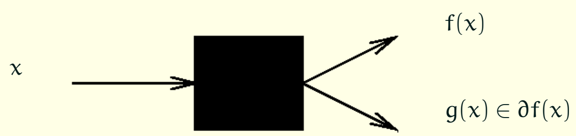
We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

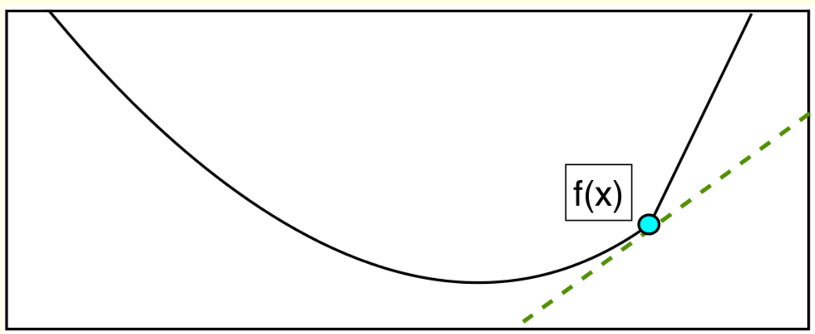
How is the oracle information used?

We look for algorithms based on information provided by an oracle



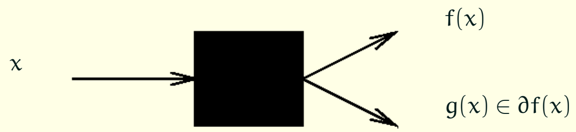
endowed with reliable stopping tests

Black box information defines linearizations



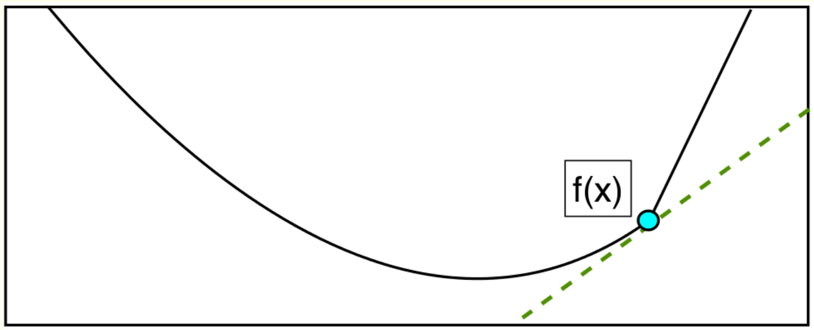
How is the oracle information used?

We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

Black box information defines linearizations

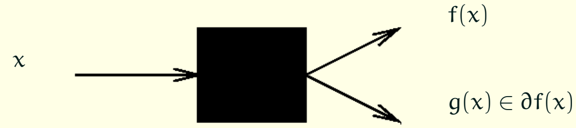


that put together create a **model M** of the function f.

The model is used to define iterates and to put in place a reliable stopping test

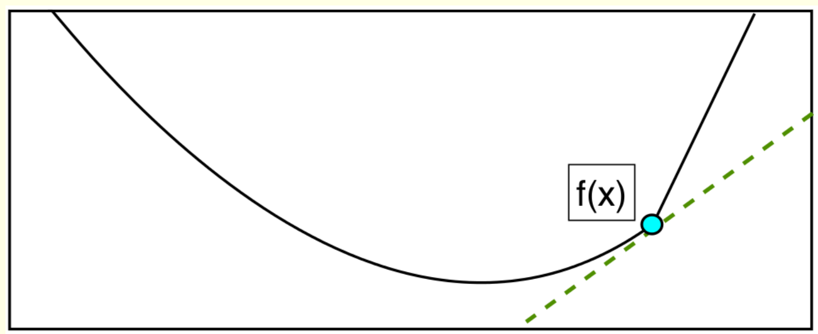
How is the oracle information used?

We look for algorithms based on information provided by an oracle



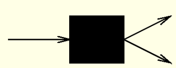
endowed with reliable stopping tests

Black box information defines linearizations



that put together create a **model M** of the function f.

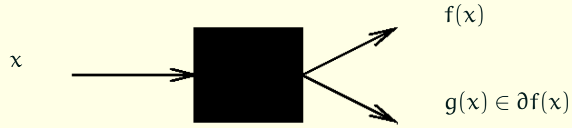
x^i



$$f^i = f(x^i)$$
$$g^i = g(x^i)$$
$$\Longrightarrow \qquad f^i + g^{i\top}(x - x^i)$$

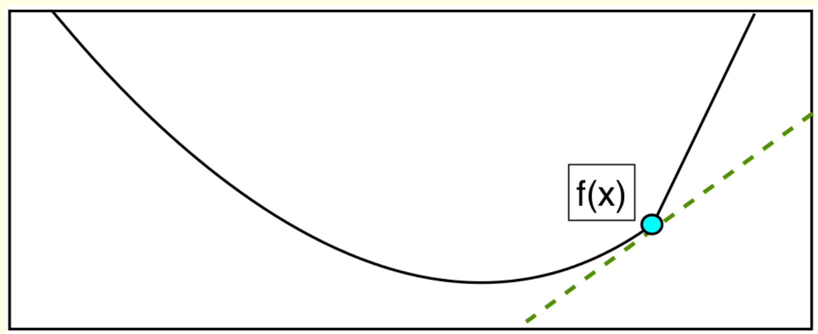
How is the oracle information used?

We look for algorithms based on information provided by an oracle



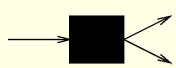
endowed with reliable stopping tests

Black box information defines linearizations



that put together create a **model M** of the function f.

x^i

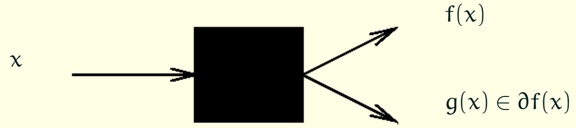


$$f^i = f(x^i)$$
$$g^i = g(x^i)$$

$$\Longrightarrow \mathbf{M}(x) = \max_i \{ f^i + g^{i\top}(x - x^i) \}$$

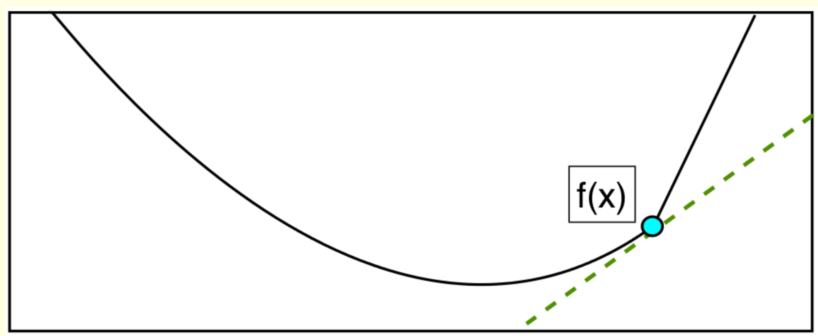
How is the oracle information used?

We look for algorithms based on information provided by an oracle

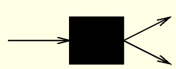


endowed with reliable stopping tests

Black box information defines linearizations



that put together create a **model M** of the function f.

x^i  $f^i = f(x^i)$, $g^i = g(x^i)$ $\Longrightarrow \mathbf{M}(x) = \max_i \{ f^i + {g^i}^\top (x - x^i) \}$

(just an example, many other models are possible)

Cutting-plane methods

To minimize f (unavailable in an explicit manner), minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

Improve the model at each iteration

Cutting-plane methods

To minimize f (unavailable in an explicit manner), minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_{k+1}(x) &= \max_{i \leq k+1} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\left(\mathbf{M}_k(x), f^{k+1} + g^{k+1\top}(x - x^{k+1}) \right) \\ &\quad \textbf{where } x^{k+1} \textbf{ minimizes } \mathbf{M}_k \end{aligned}$$

Cutting-plane methods

To minimize f (unavailable in an explicit manner), minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_{k+1}(x) &= \max_{i \le k+1} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\left(\mathbf{M}_k(x), f^{k+1} + g^{k+1\,\top}(x - x^{k+1}) \right) \\ &\quad \textbf{where } x^{k+1} \textbf{ minimizes } \mathbf{M}_k \end{aligned}$$

Instead of $x^* \in \arg\min f(x)$ at one shot

Cutting-plane methods

To minimize f (unavailable in an explicit manner), minimize its model $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

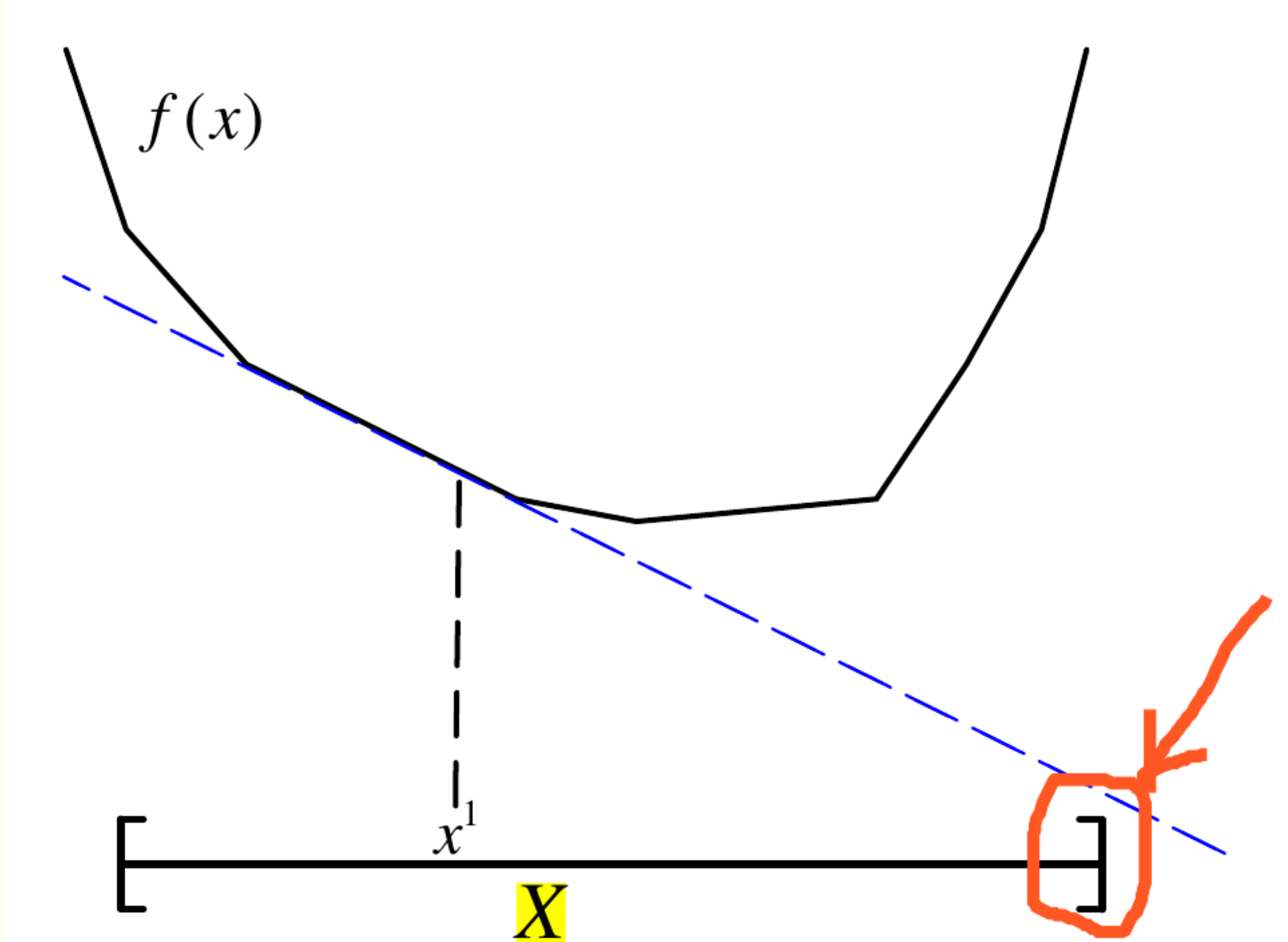
Improve the model at each iteration:

$$\begin{aligned} \mathbf{M}_{k+1}(x) &= \max_{i \le k+1} \left\{ f^i + g^{i\top}(x - x^i) \right\} \\ &= \max\Big(\mathbf{M}_k(x), f^{k+1} + g^{k+1\,\top}(x - x^{k+1}) \Big) \\ &\quad \textbf{where } x^{k+1} \textbf{ minimizes } \mathbf{M}_k \end{aligned}$$

Instead of $x^* \in \arg\min f(x)$ at one shot,

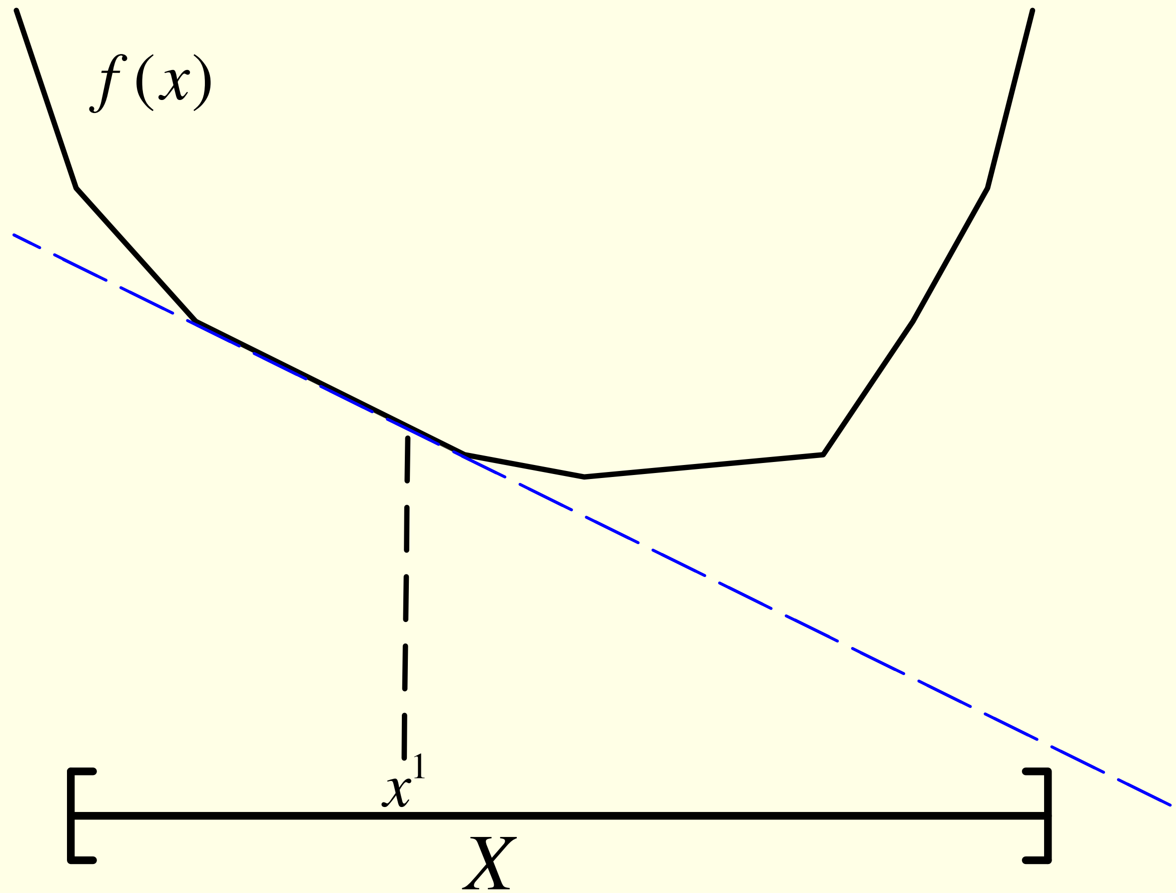
$x^{k+1} \in \arg\min \mathbf{M}_k(x)$ **iteratively**

Cutting-plane methods

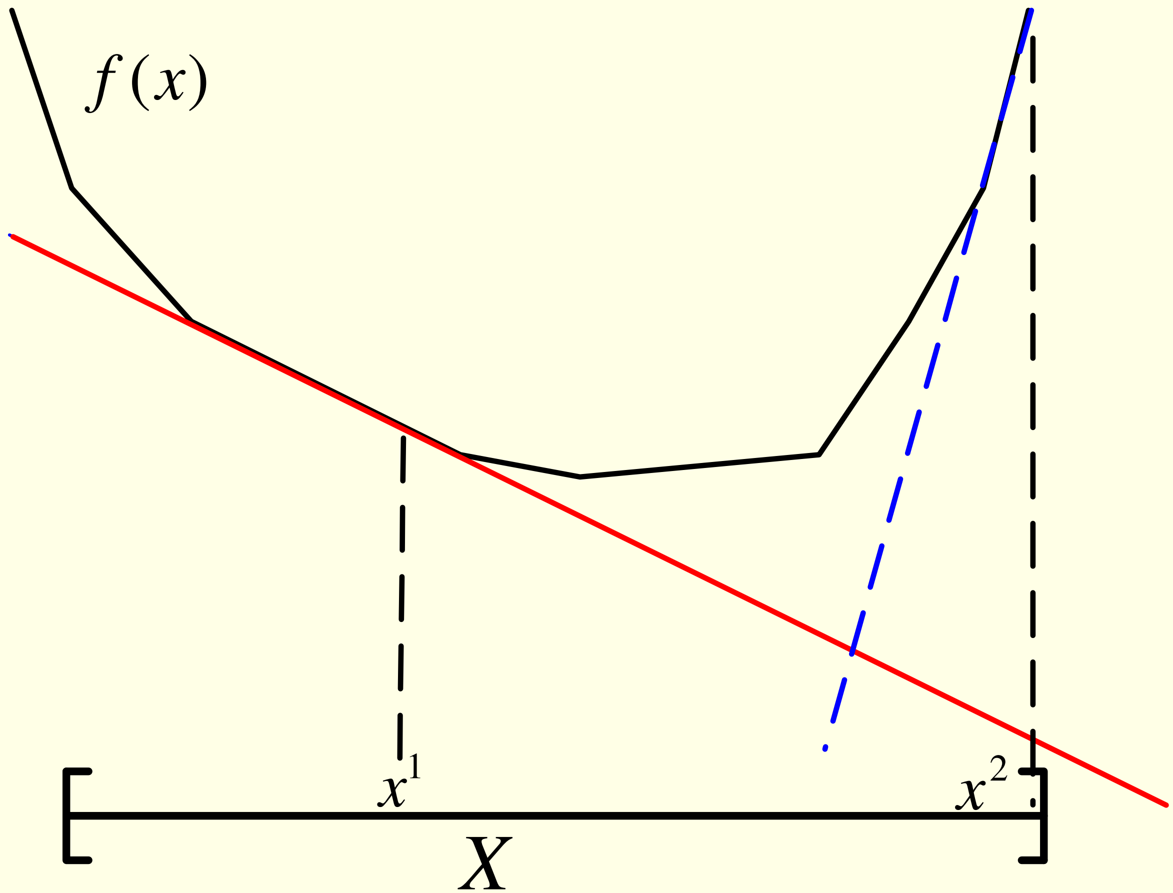


Artificial bounding at least for the first iterations

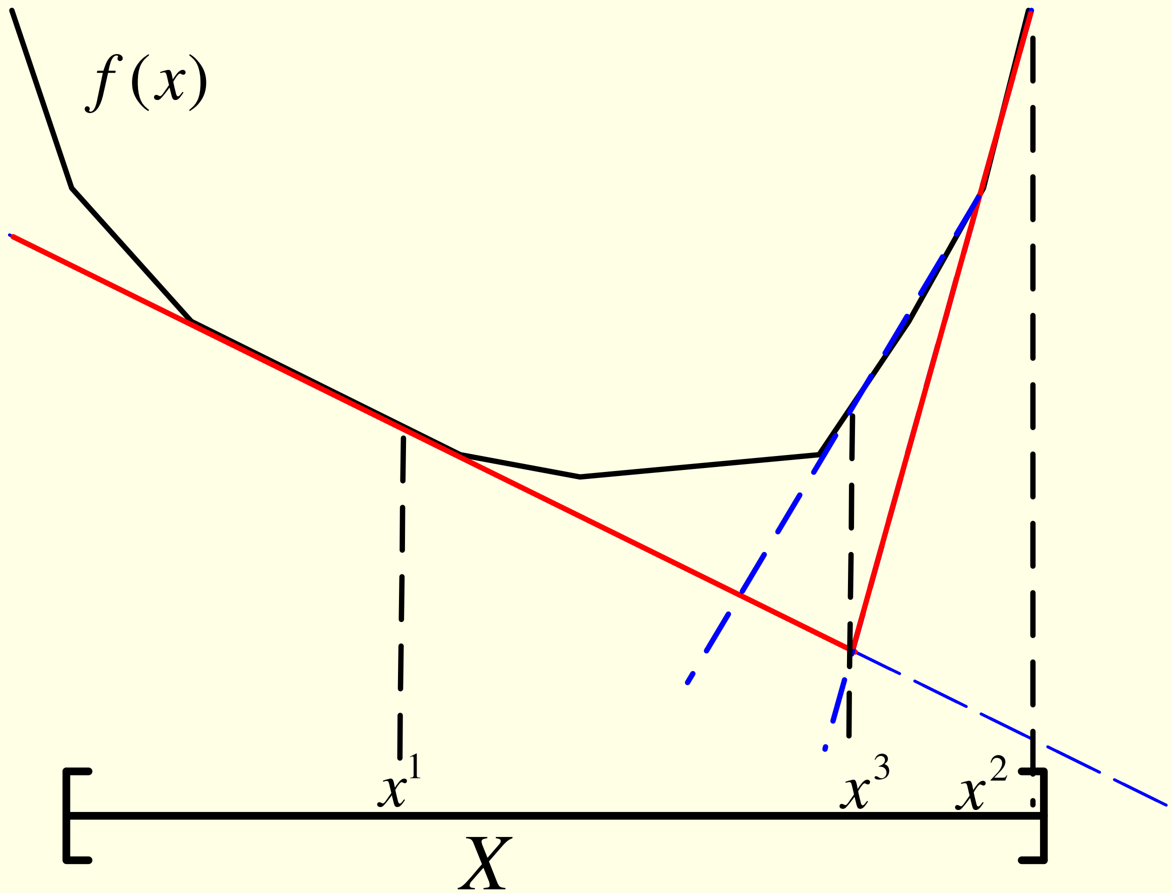
Cutting-plane methods



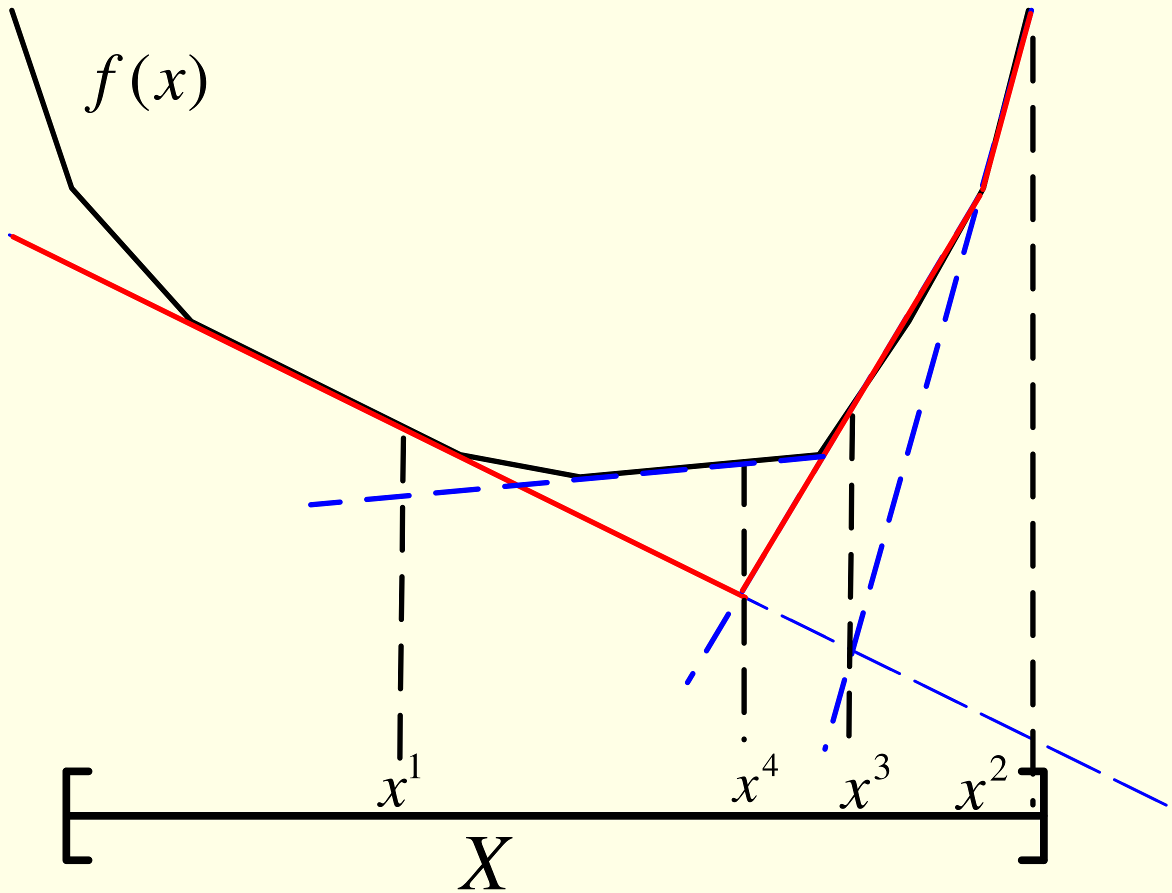
Cutting-plane methods



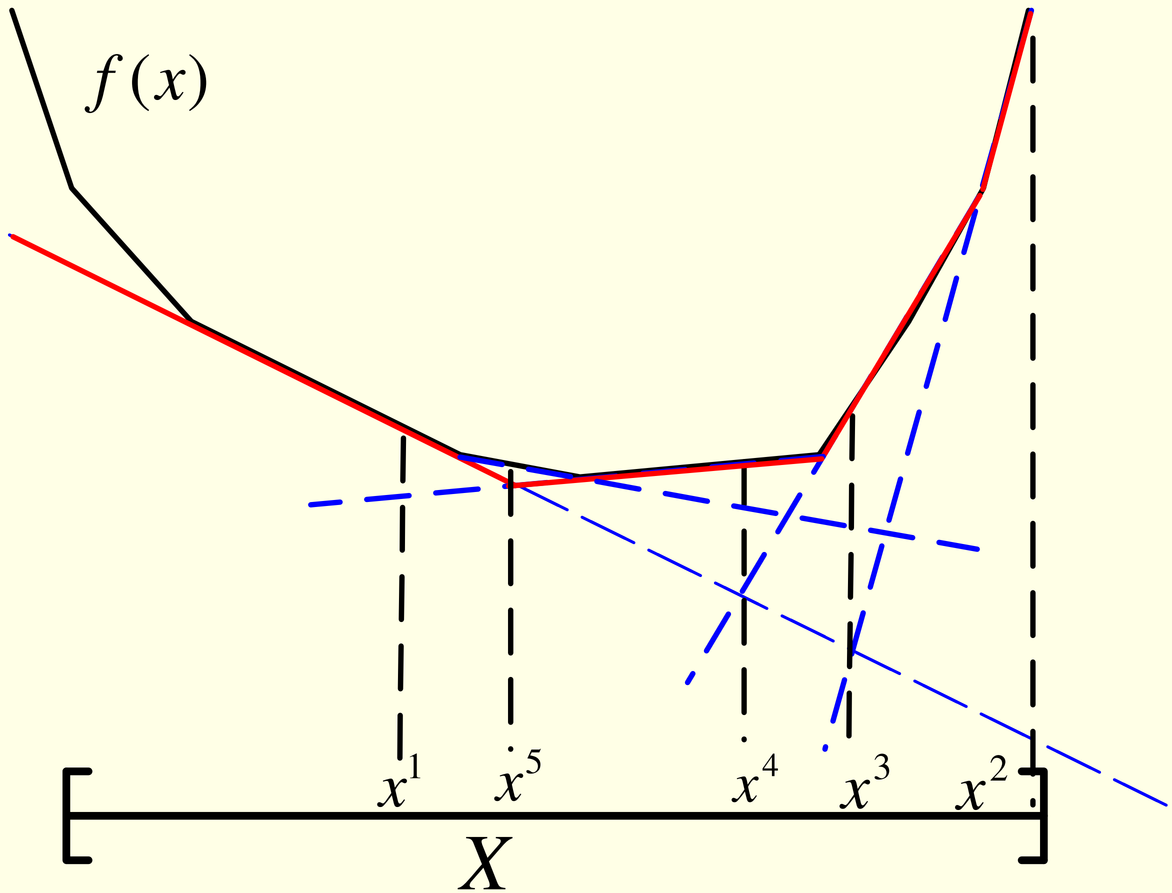
Cutting-plane methods



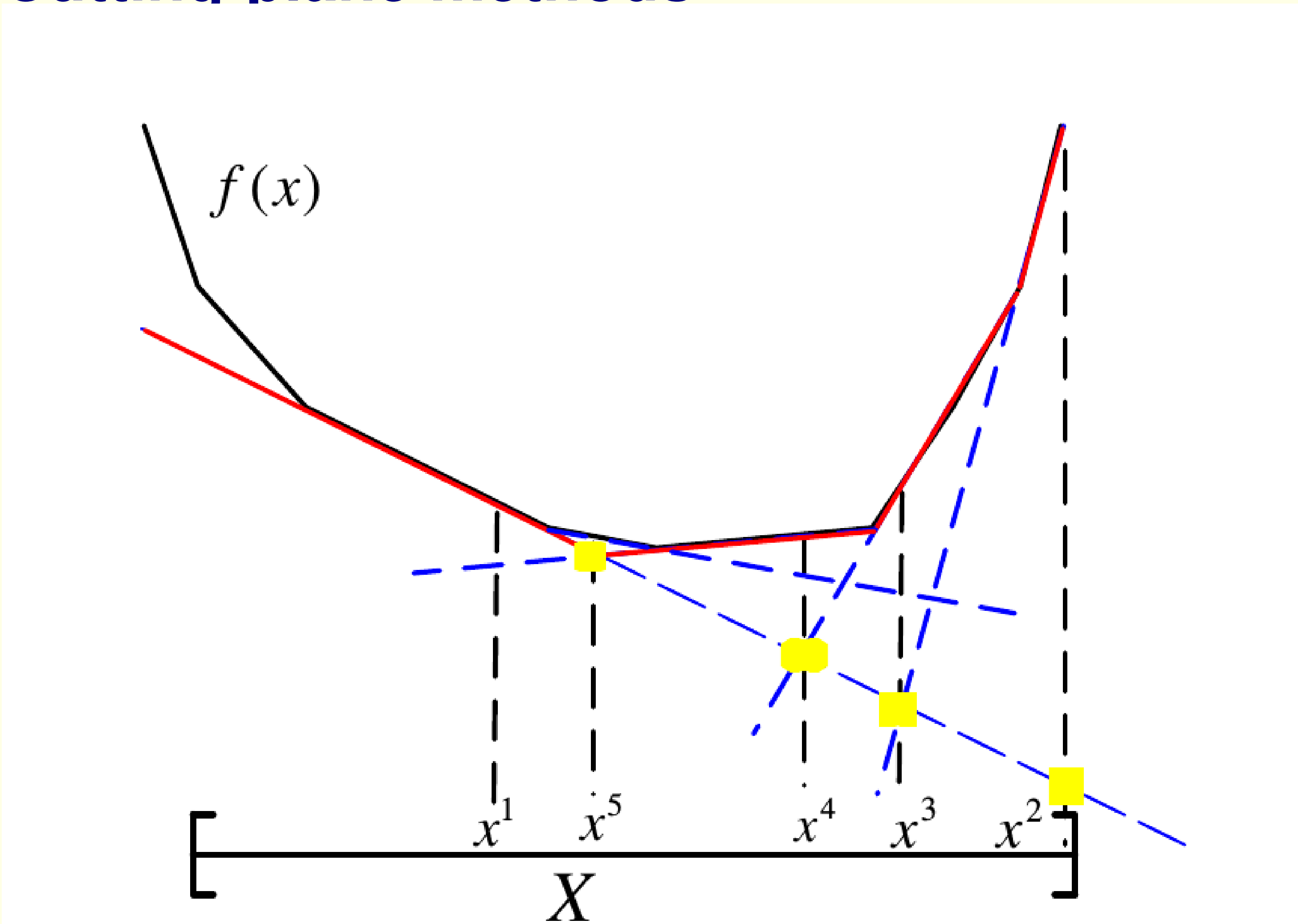
Cutting-plane methods



Cutting-plane methods

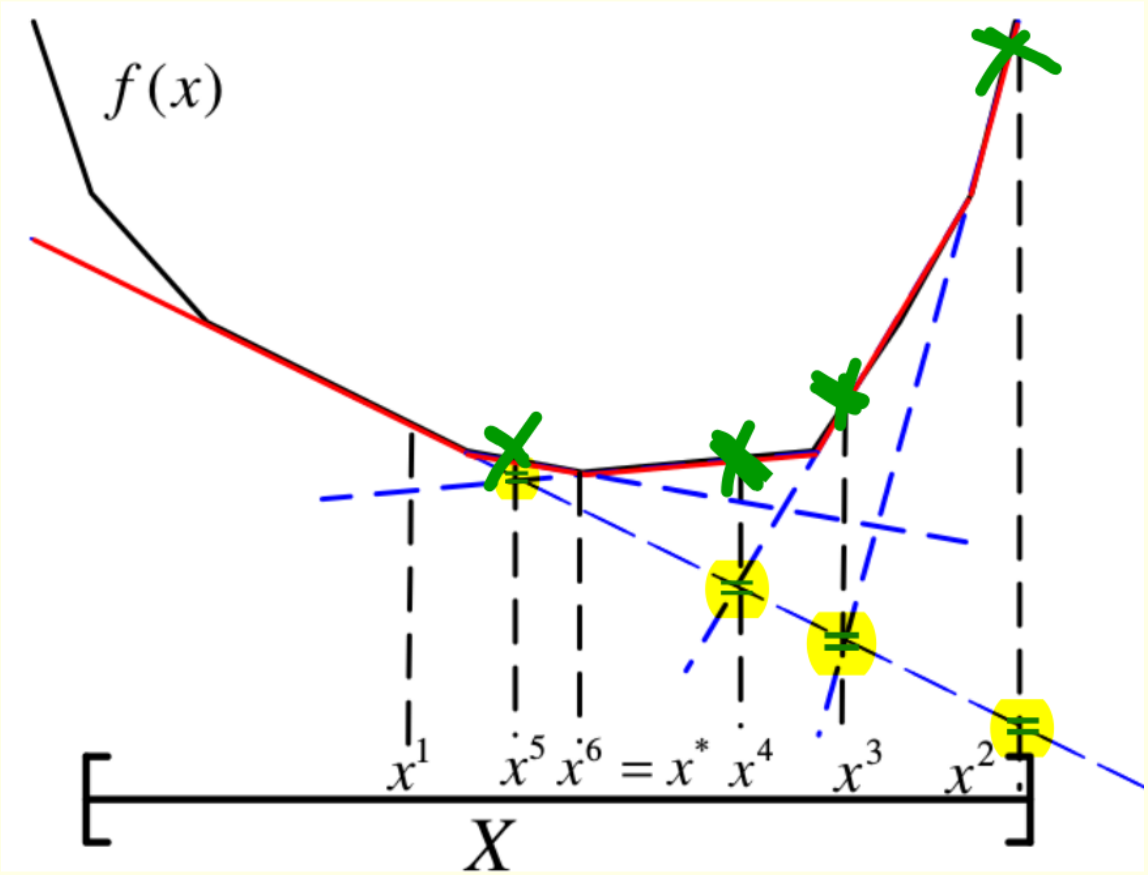


Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases

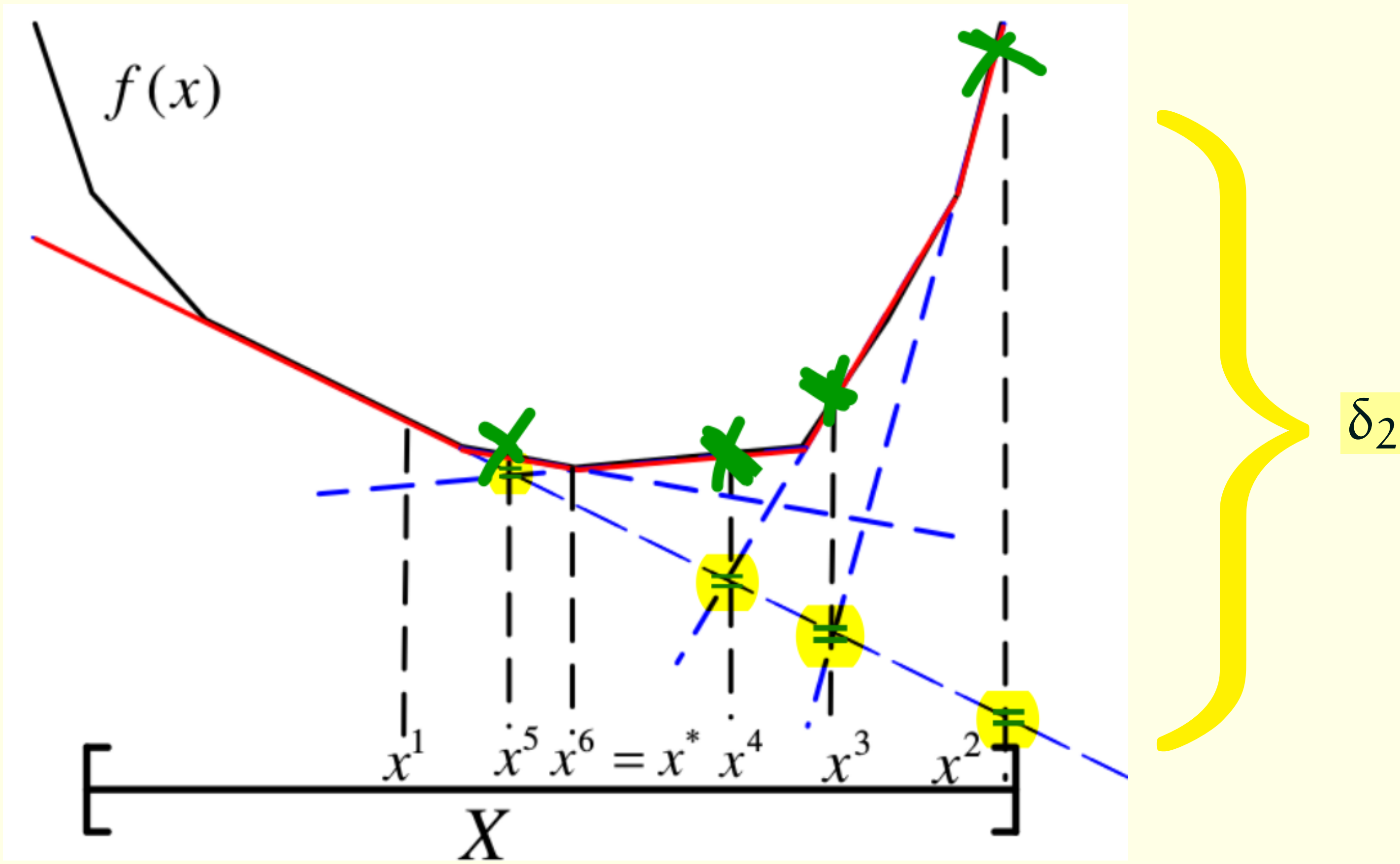
Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases but not necessarily the functional values:

$f(x^5) > f(x^4)$

Cutting-plane methods



$\{\mathbf{M}_k(x^{k+1})\}$ increases but not necessarily the functional values: $f(x^5) > f(x^4)$. Stopping test measures $\delta_k := f(x^k) - \mathbf{M}_{k-1}(x^k)$

Cutting-plane Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k.

2 Compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 $\mathbf{M}_{k+1}(\cdot) = \max\Big(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\Big)$, $k = k+1$, loop to 1.

Cutting-plane Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k. **If** $f(x^k) - \mathbf{M}_{k-1}(x^k) \leq \mathrm{tol}$ **STOP**

2 Compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, $k = k+1$, loop to 1.

CP methods are

an improved algorithmic version

of the Aussie sign

a better recipe

Cutting-plane Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k. **If** $f(x^k) - \mathbf{M}_{k-1}(x^k) \leq \text{tol}$ **STOP**

2 Compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, $k = k+1$, loop to 1.

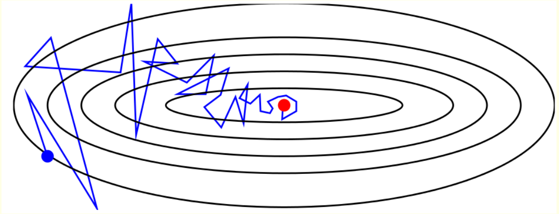
CP methods are

an improved algorithmic version

of the Aussie sign

a better recipe

converges, but can stall and



Cutting-plane Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k. **If** $f(x^k) - \mathbf{M}_{k-1}(x^k) \leq \mathtt{tol}$ **STOP**

2 Compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, $k = k+1$, loop to 1.

CP methods are

an improved algorithmic version

of the Aussie sign

a better recipe

CP methods are like caipirinha with a few drops of cachaça

Cutting-plane Methods

0 Choose x^1 and set $k = 1$.

1 Call the oracle at x^k. **If** $f(x^k) - \mathbf{M}_{k-1}(x^k) \leq \mathrm{tol}$ **STOP**

2 Compute $x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$

3 $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, $k = k+1$, loop to 1.

CP methods are

an improved algorithmic version

of the Aussie sign

a better recipe

CP methods are like caipirinha with a few drops of cachaça

can be improved!

Cutting-plane Methods: why not the best recipe

$$\begin{cases} \text{Non-monotone functional values, but converges} \\ \text{because } \liminf\left(f(x^k) - \mathbf{M}_{k-1}(x^k)\right) \to 0 \\ \text{Has a stopping test, but LP size grows indefinitely} \\ \text{eventually numerical errors prevail.} \end{cases}$$

$x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$ with $\begin{array}{l} \mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\} \\ \text{and } X \text{ polyhedral} \end{array}$

Cutting-plane Methods: why not the best recipe

$$\left\{\begin{array}{l}\text{Non-monotone functional values, but converges}\\ \text{because } \liminf\left(f(x^k)-\mathbf{M}_{k-1}(x^k)\right)\to 0\\ \text{Has a stopping test, but LP size grows indefinitely}\\ \text{eventually numerical errors prevail.}\end{array}\right.$$

$x^{k+1}\in\arg\min_X \mathbf{M}_k(x)$ with $\begin{array}{l}\mathbf{M}_k(x)=\max_{i\le k}\{f^i+g^{i\top}(x-x^i)\}\\ \text{and } X \text{ polyhedral}\end{array}$

is equivalent to solving a linear programming problem

$$\left\{\begin{array}{ll}\min & r\\ \text{s.t.} & r\in\mathbb{R}, x\in X\\ & r\ge f^i+g^{i\top}(x-x^i) \text{ for } i\le k\end{array}\right.$$

Cutting-plane Methods: why not the best recipe

$$\begin{cases} \text{Non-monotone functional values, but converges} \\ \text{because } \liminf\left(f(x^k) - \mathbf{M}_{k-1}(x^k)\right) \to 0 \\ \text{Has a stopping test, but LP size grows indefinitely} \\ \text{eventually } \textbf{numerical errors} \text{ prevail.} \end{cases}$$

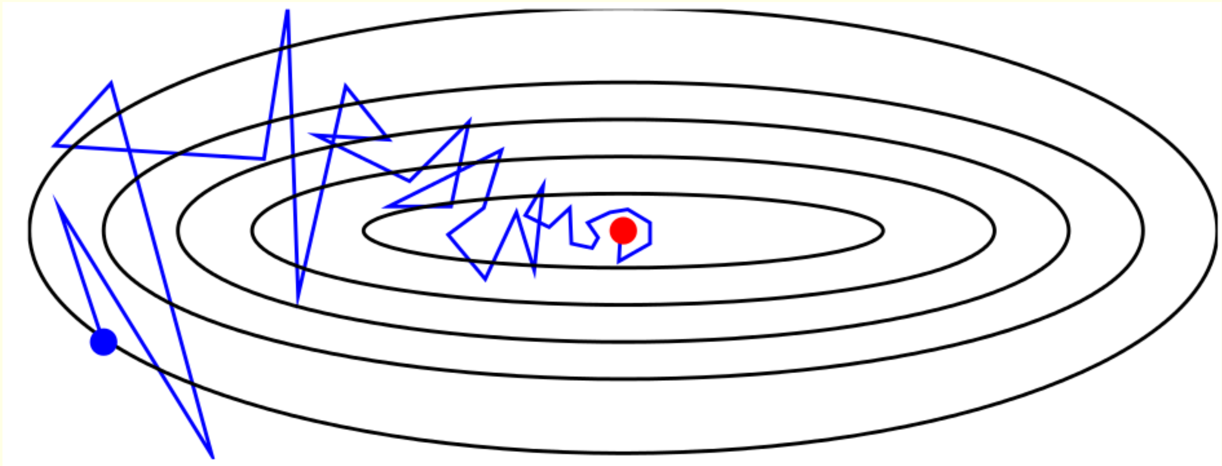
$x^{k+1} \in \arg\min_X \mathbf{M}_k(x)$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ and X polyhedral

is equivalent to solving a linear programming problem

$$\begin{cases} \min & r \\ \text{s.t.} & r \in \mathbb{R}, x \in X \\ & r \geq f^i + g^{i\top}(x - x^i) \text{ for } i \leq \ \textbf{k grows with iterations} \end{cases}$$

Ingredients for the best recipe

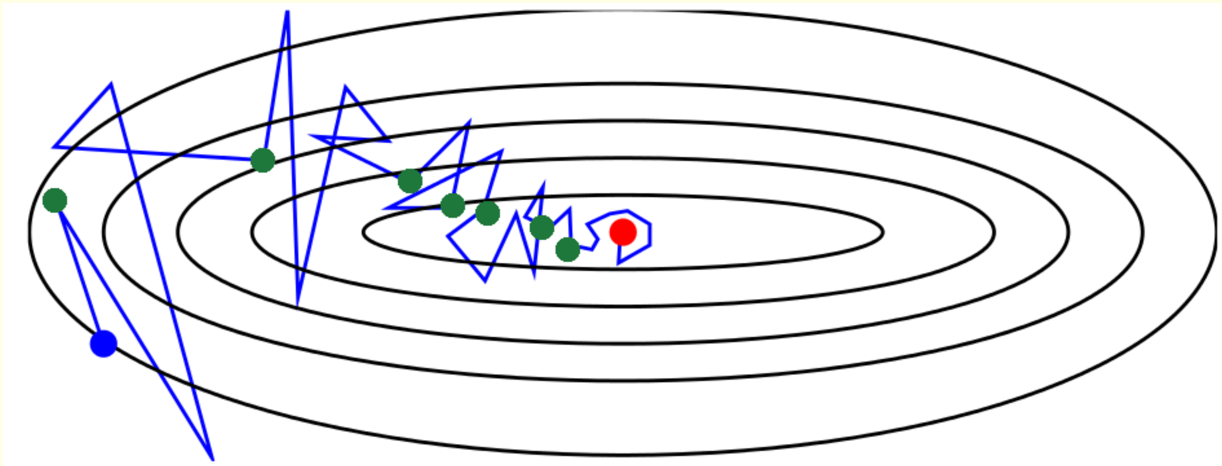
- CP brings in the concept of a model, which gives a stopping test (δ^k)
- CP still non-monotone



Monotonicity defeats instability and oscillations

Ingredients for the best recipe

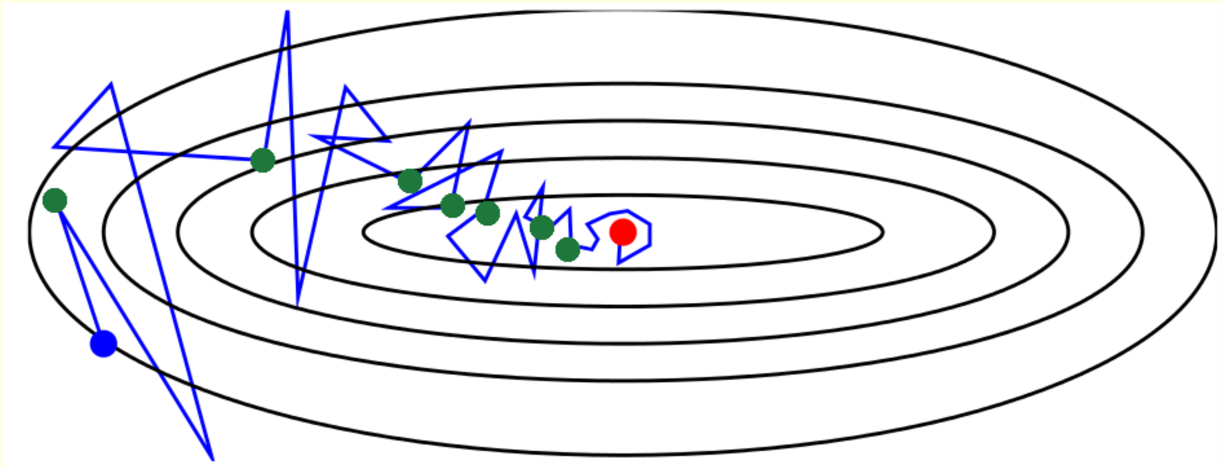
- CP brings in the concept of a model, which gives a stopping test (δ^k)
- CP still non-monotone



Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges

Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test (δ^k)
- CP still non-monotone

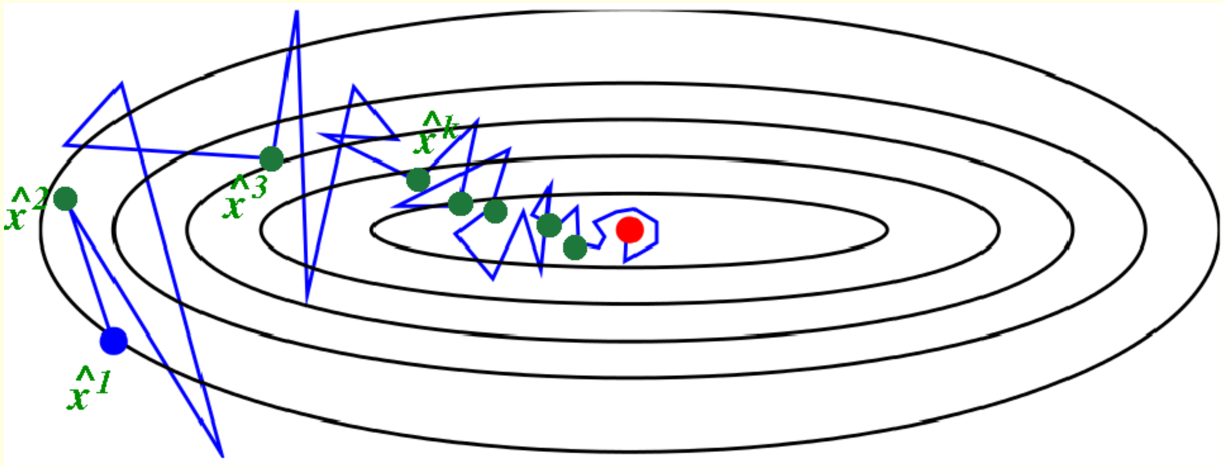


Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges

- Bundle Methods select green-spot iterates using a descent rule

Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test (δ^k)
- CP still non-monotone



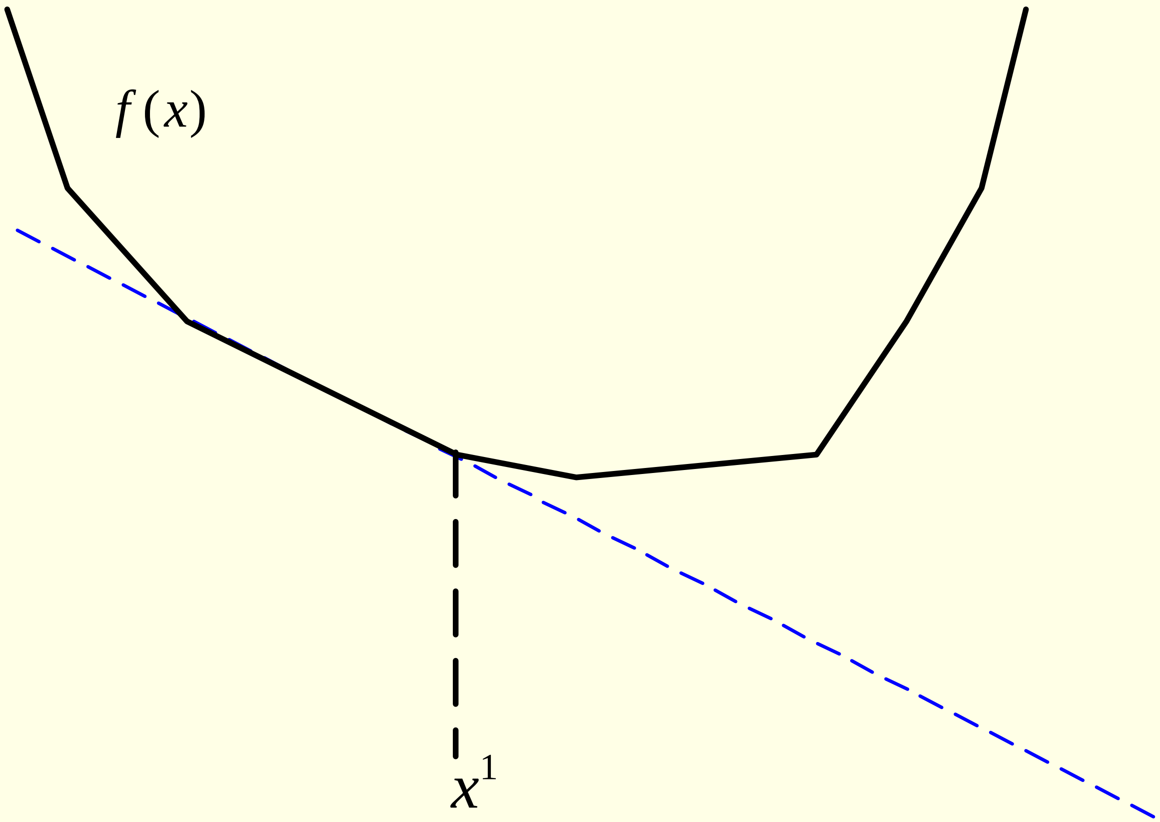
Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges

- Bundle Methods select green-spot iterates using a descent rule

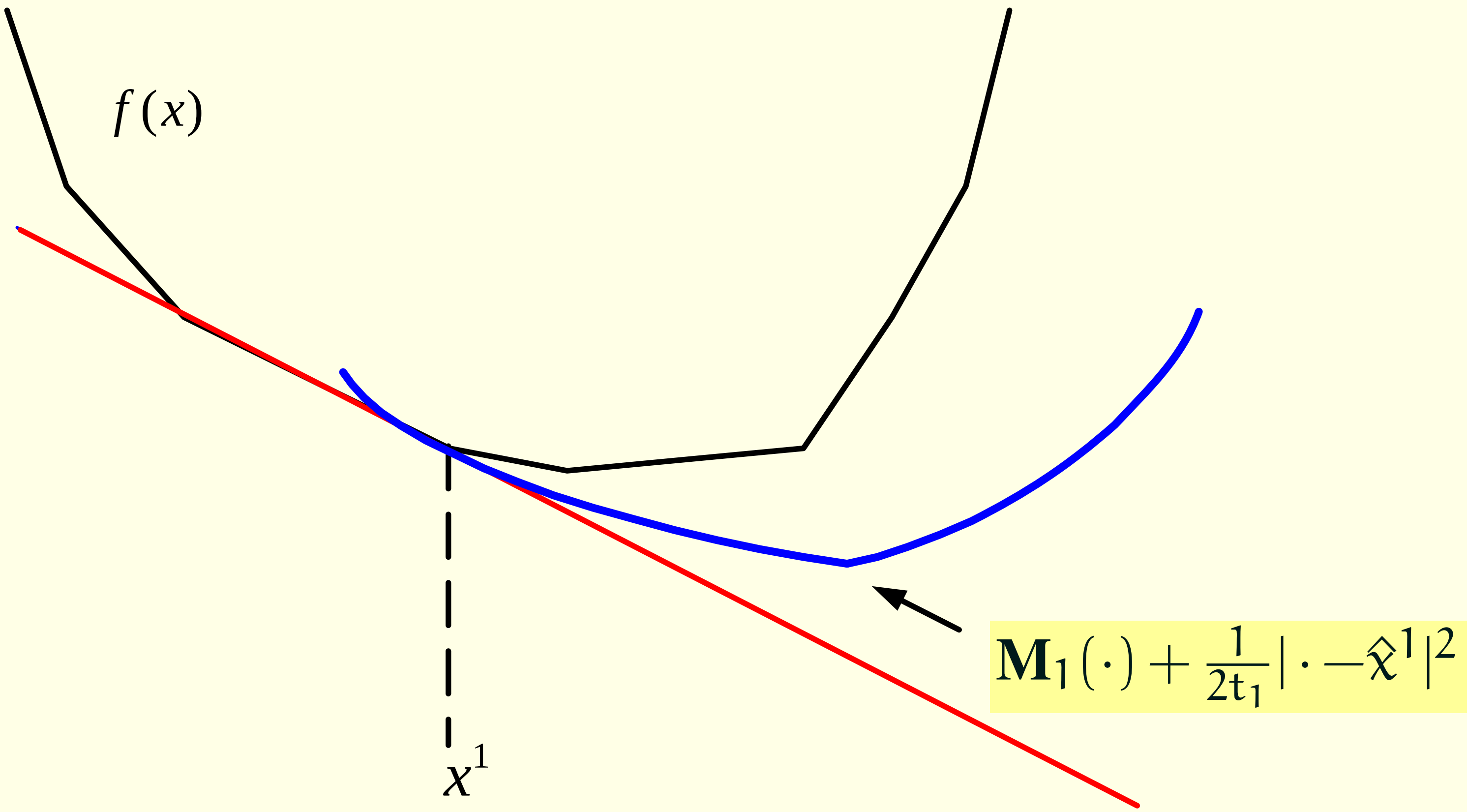
  $f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k$ where δ_k **is a positive quantity** $< f(\hat{x}^k)$

  limit points of the **serious-step** subsequence $\{\hat{\mathbf{x}}^k\}$ minimize f

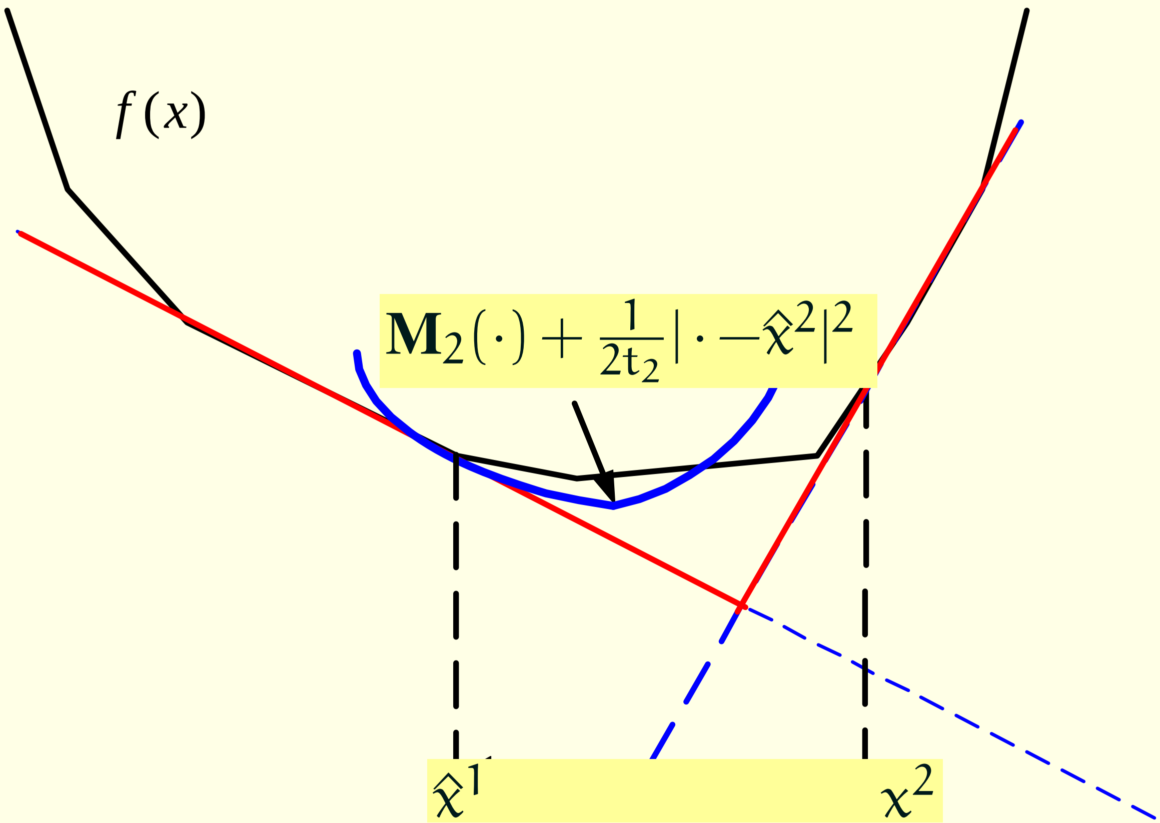
Bundle Methods



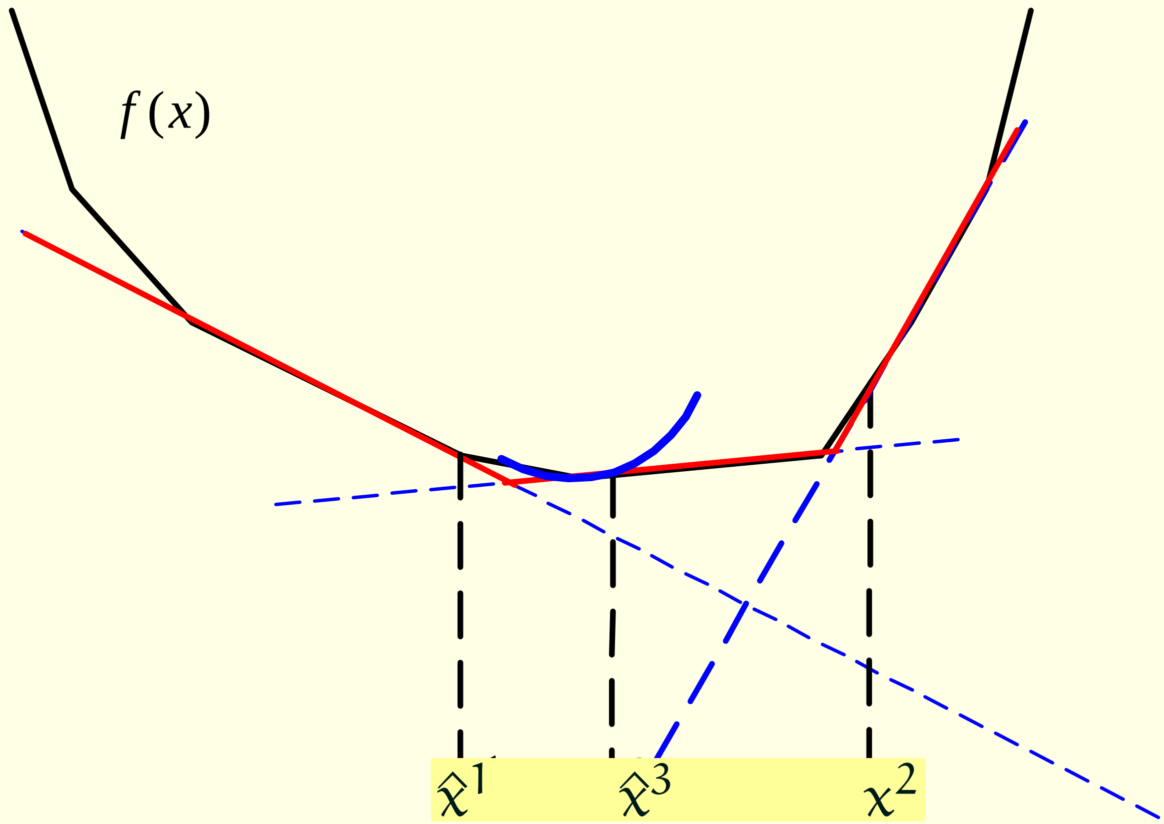
Bundle Methods



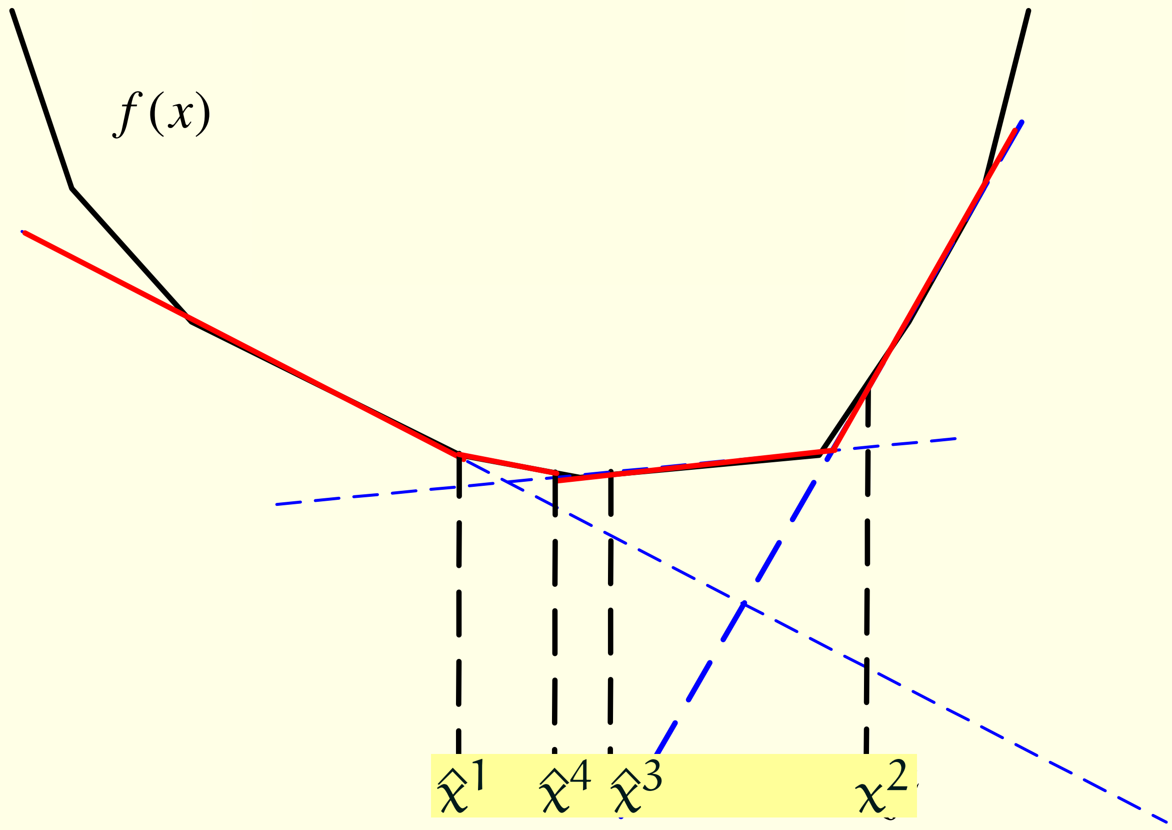
Bundle Methods



Bundle Methods



Bundle Methods



Bundle Methods

0 Choose x^1, set $k = 1$, and let $\hat{x}^1 = x^1$.

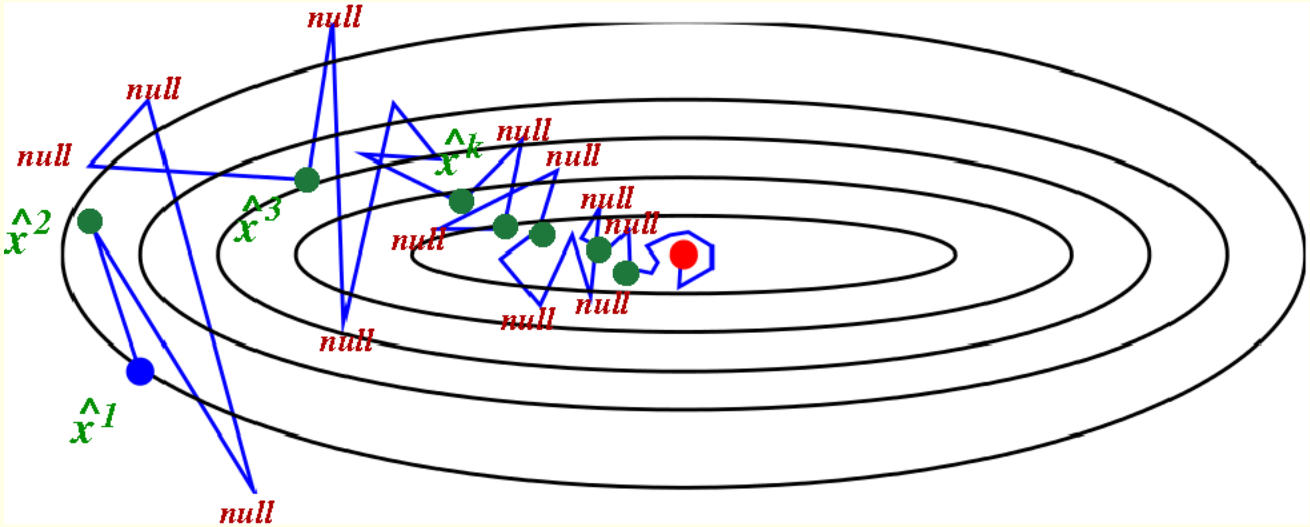
1 Compute $x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$

2 If $\delta_{\mathbf{k}} := f(\hat{x}^k) - \mathbf{M}_k(x^{k+1}) \leq \mathrm{tol}$ STOP

3 Call the oracle at x^{k+1}.

If $f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k$, set $\hat{x}^{k+1} = x^{k+1}$ • (Serious Step)
Otherwise, maintain $\hat{x}^{k+1} = \hat{x}^k$ (Null Step)

4 Define $\mathbf{M}_{k+1}$, t_{k+1}, make $k = k + 1$, and loop to 1.



Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge f^i + g^{i\top}(x - x^i) \text{ for } i \le k \end{cases}$$

A posteriori, the solution remains the same if ...

Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge f^i + g^{i\top}(x - x^i) \text{ for } \mathbf{i \le k} \end{cases}$$

A posteriori, the solution remains the same if all, or …

Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\Big(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\Big)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge f^i + g^{i\top}(x - x^i) \text{ for } \textbf{active i's} \end{cases}$$

A posteriori, the solution remains the same if all, or active, or ...

Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\Big(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\Big)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge \sum_{\mathbf{i}} \bar{\alpha}^{\mathbf{i}}(f^i + g^{i\top}(x - x^i)) \end{cases}$$

A posteriori, the solution remains the same if all, or active, or the **optimal convex combination**

Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge \sum_i \bar{\alpha}^i (f^i + g^{i\top}(x - x^i)) \end{cases}$$

A posteriori, the solution remains the same if all, or active, or the optimal convex combination

$$\textbf{BM}\ \mathbf{M}_{k+1}(\cdot) = \max\left(\begin{array}{l} \mathbf{M}_k(\cdot) \\ \max_{active} \\ \text{aggregate} \end{array}, f^k + g^{k\top}(\cdot - x^k)\right)$$

Bundle Methods

Unlike **CP** $\mathbf{M}_{k+1}(\cdot) = \max\Big(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\Big)$,

now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \le k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

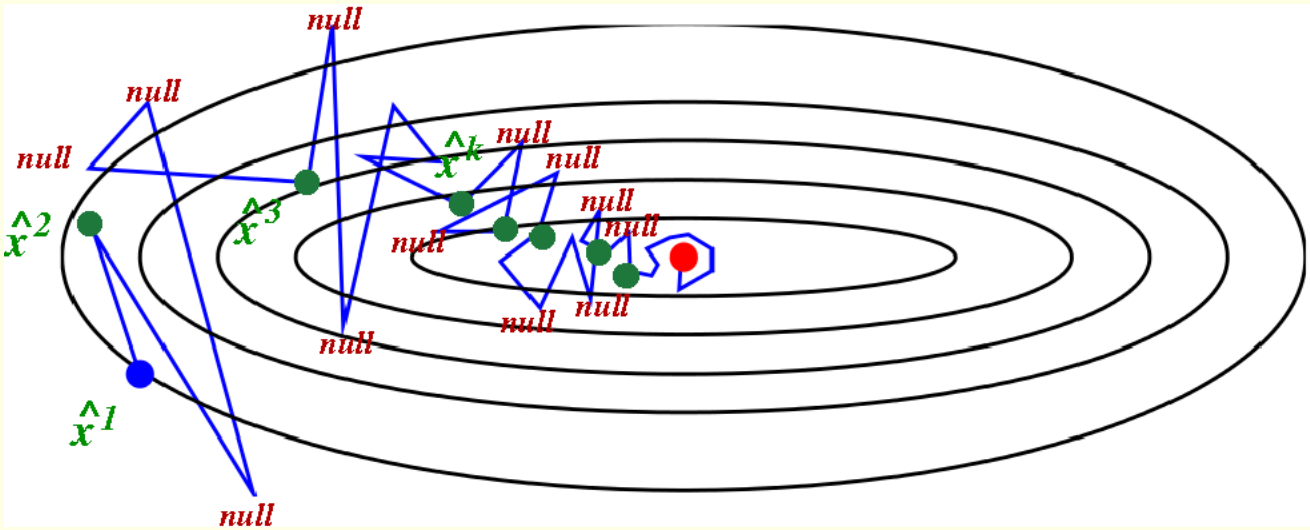
$$\begin{cases} \min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\ \text{s.t.} & r \ge \sum_i \bar{\alpha}^i (f^i + g^{i\top}(x - x^i)) \end{cases}$$

Same solution if all, or active, or the optimal convex combination

$$\textbf{BM} \quad \mathbf{M}_{k+1}(\cdot) = \max\left(\begin{array}{l} \mathbf{M}_k(\cdot) \\ \max_{active} \\ \text{aggregate} \end{array}, f^k + g^{k\top}(\cdot - x^k)\right)$$

Bundle Compression: QP with 2 constraints

Bundle Methods

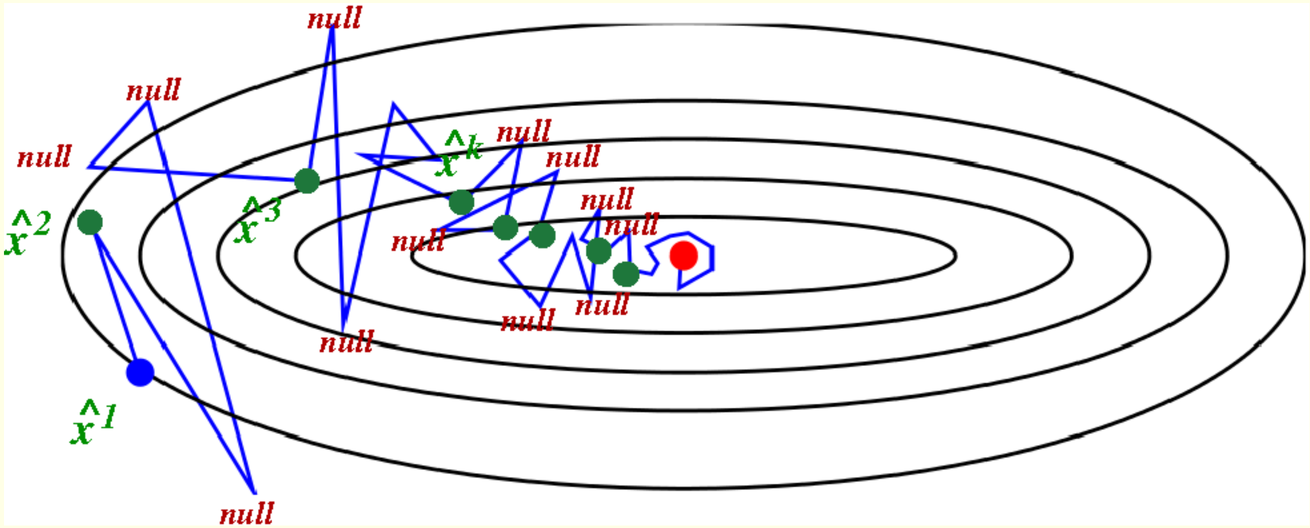


When k $\rightarrow \infty$, the algorithm generates two subsequences.

Convergence analysis addresses the mutually exclusive situations

- either the SS subsequence is infinite
- or there is a last SS, followed by infinitely many null steps

Bundle Methods



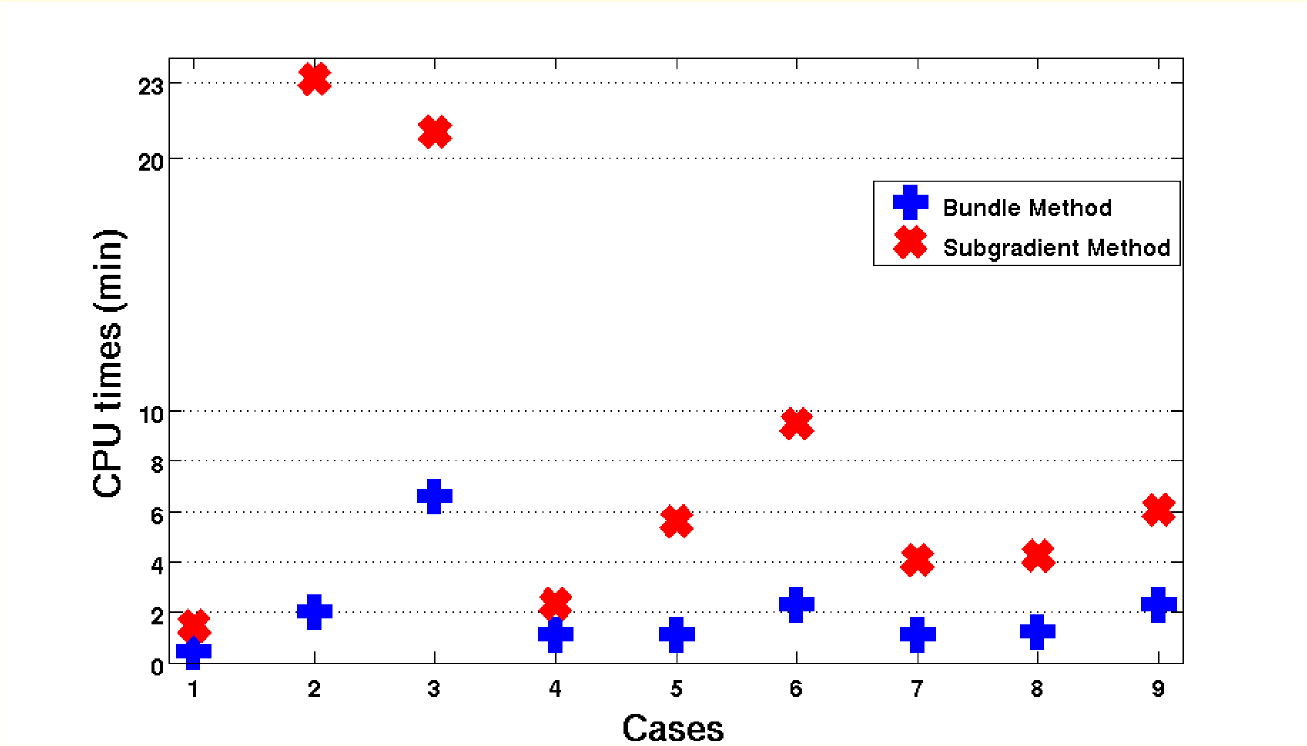
When $k \to \infty$, the algorithm generates two subsequences.

Convergence analysis addresses the mutually exclusive situations

- either the SS subsequence is infinite **(limit point minimizes f)**
- or there is a last SS, followed by infinitely many null steps **(last SS minimizes f and null$\to$ last SS)**

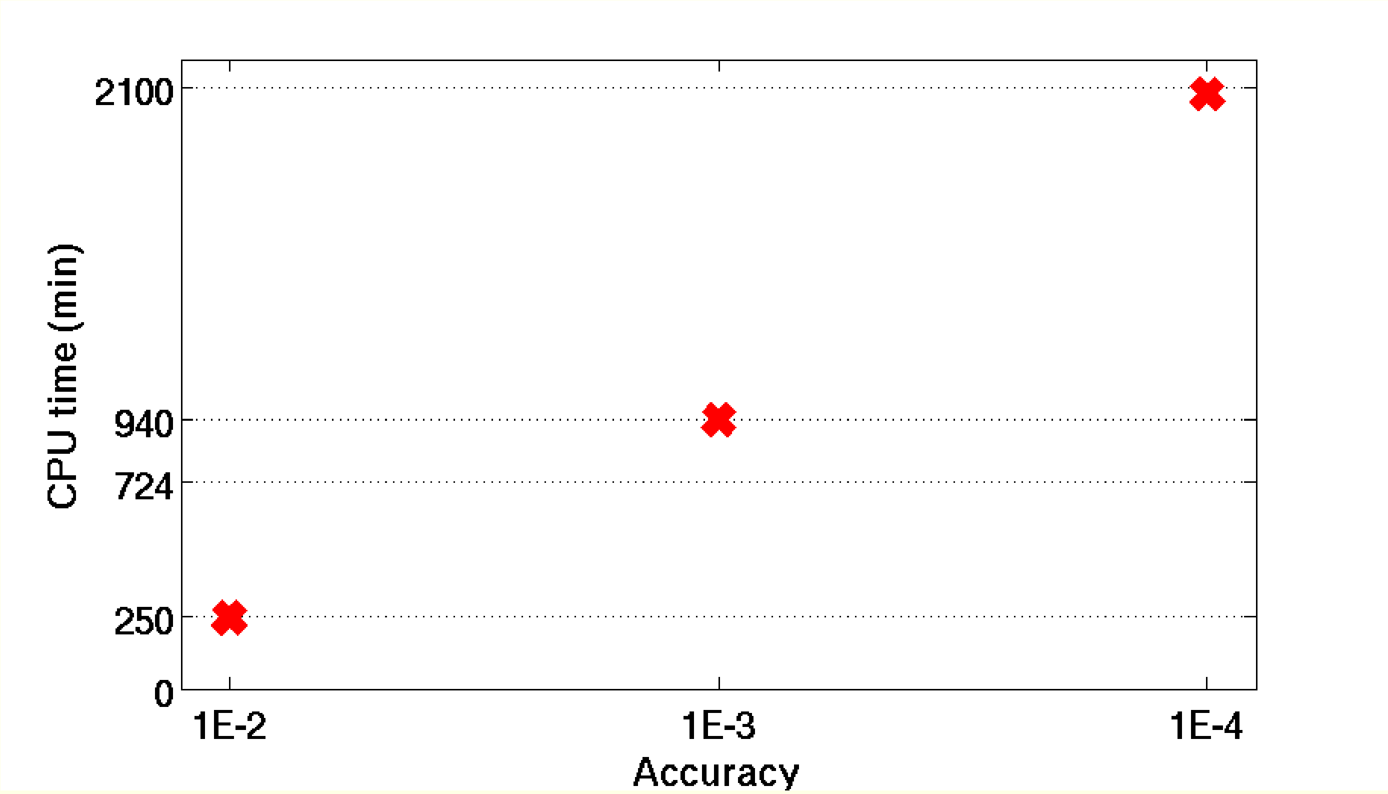
Comparing the methods: bundle and SG

Typical performance on a battery of Unit Commitment problems



Comparing the methods: bundle and CP

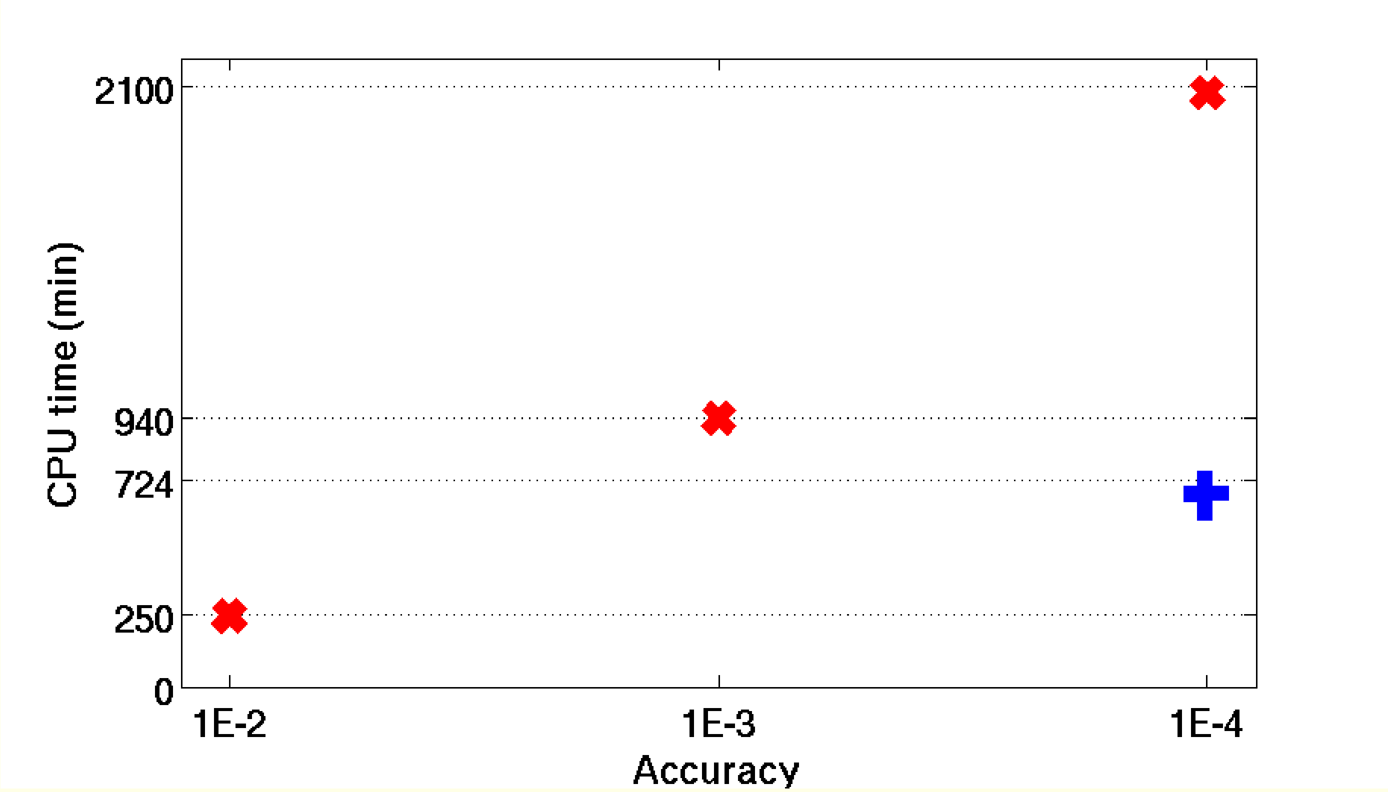
On a battery of probabilistically constrained problems



X CP is fast to reach a few digits of accuracy, then stalls

Comparing the methods: bundle and CP

On a battery of probabilistically constrained problems



x CP is fast to reach a few digits of accuracy, then stalls

+ Bundle is consistently 3 times faster

Comparing the methods

SG ok if low precision -for instance in combinatorial optimization

CP ok if not many iterations -usually not the case

Bundle ok if f complex and high precision is required

Comparing the methods

SG ok if low precision -for instance in combinatorial optimization

CP ok if not many iterations -usually not the case

Bundle ok if f complex and high precision is required

a good recipe

Can we do any better??

Can we do any better??

YES, WE CAN

Bundle Methods with on-demand accuracy

the new generation

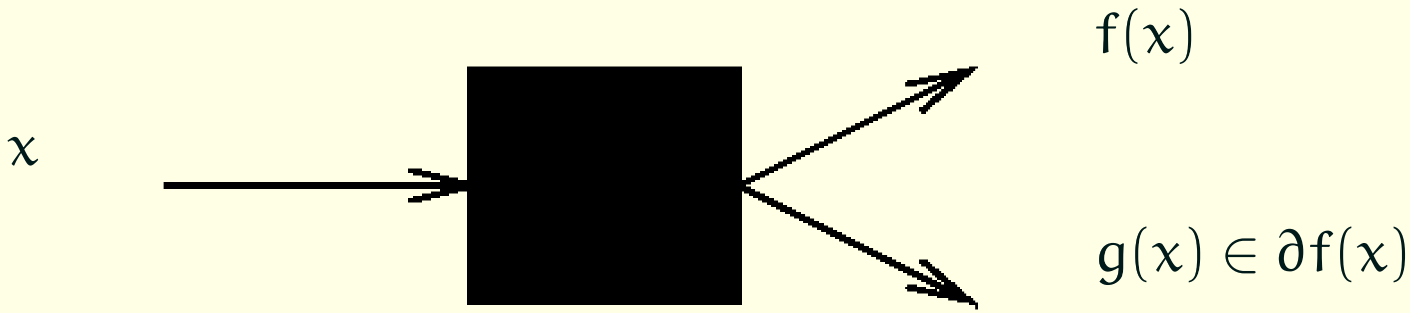
(or the perfect caipirinha)

First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method



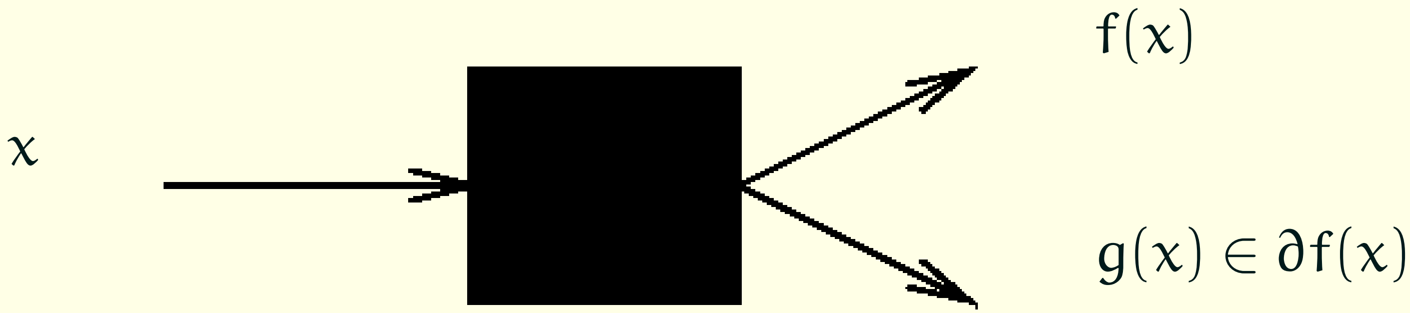
is doomed to slow convergence speed: complexity is $O(\frac{1}{\sqrt{k}})$ k iterations

First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method



is doomed to slow convergence speed: complexity is $O(\frac{1}{\sqrt{k}})$ k iterations

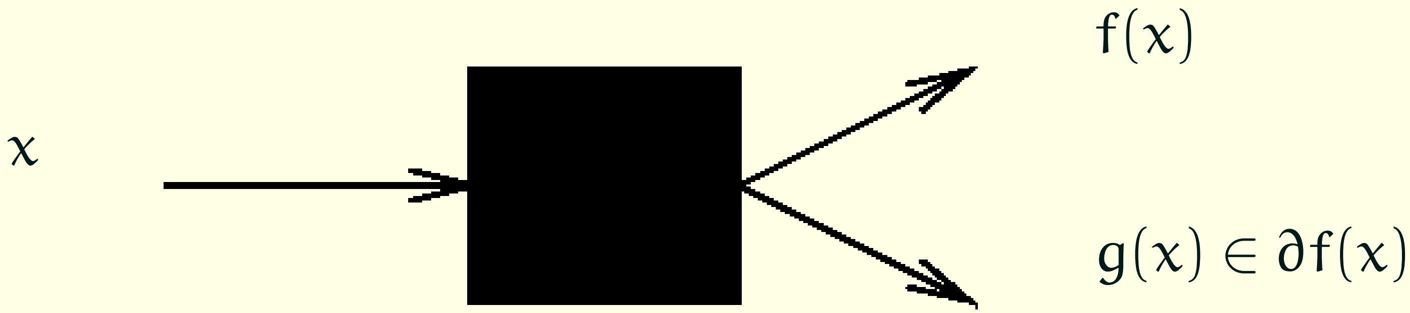
Better performance possible by exploiting structure

First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method



is doomed to slow convergence speed: complexity is $O(\frac{1}{\sqrt{k}})$ k iterations

Better performance possible by exploiting structure

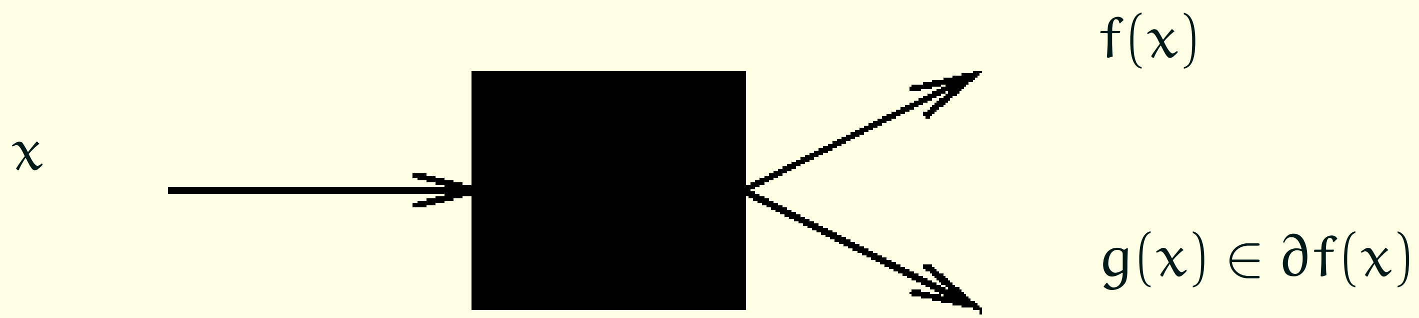
For instance, for strongly convex f complexity drops to $O(\frac{1}{k})$

First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method



is doomed to slow convergence speed: complexity is $O(\frac{1}{\sqrt{k}})$ k iterations

Note: complexity results assume black box always called as above

How does structure appear?

– Explicitly

as a sum

as a composition

– Implicitly

U-Lagrangian

VU-decomposition

partly smooth functions

How does structure appear?

– Explicitly

  as a sum

  as a composition

– Implicitly

  U-Lagrangian

  VU-decomposition

  partly smooth functions

How does structure appear?

– Explicitly

as a sum

as a composition

} $\neq$ **black boxes**

– Implicitly

U-Lagrangian

VU-decomposition

partly smooth functions

How does structure appear?

– Explicitly

as a sum

as a composition

$\neq$ **black boxes**

– Implicitly

U-Lagrangian

VU-decomposition

partly smooth functions

digging tools

Explicit Structure:

Opening the Black Box

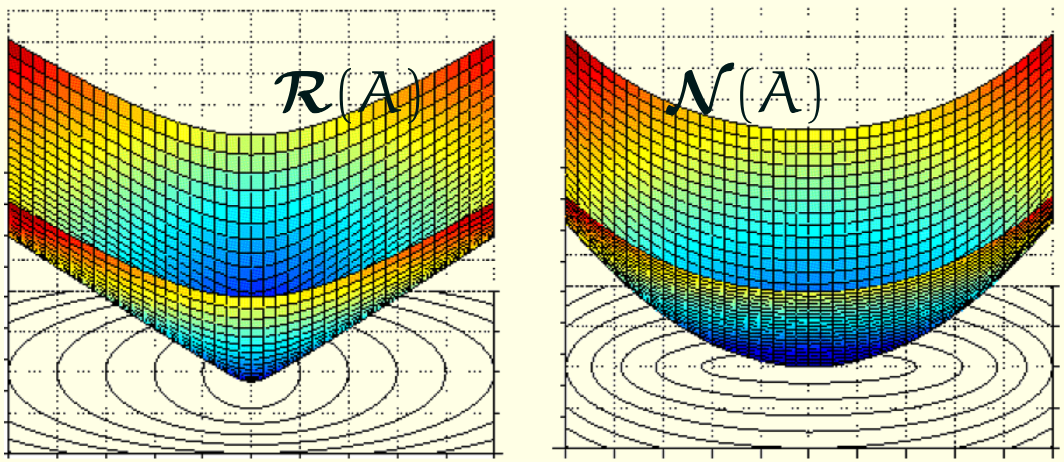
A convex partly nonsmooth function

For $x \in \mathbb{R}^n$, given matrices $A \succeq 0$, $B \succ 0$,

$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

has a unique minimizer at 0.

On $\mathcal{N}(A)$ the function is not differentiable, and the first term vanishes: $f|_{\mathcal{N}(A)}$ looks smooth.



This function has several interesting structures

If no structure at all

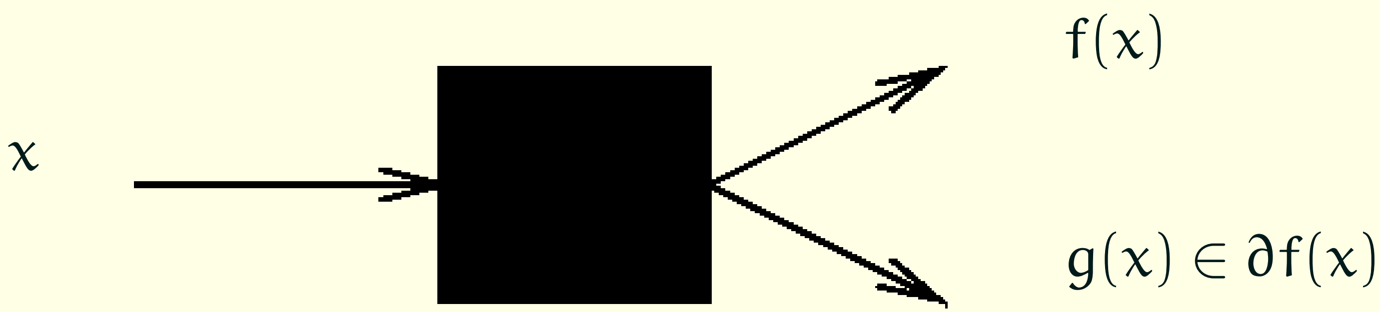
$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

This function has several interesting structures

If no structure at all

$$f(x) = \sqrt{x^\top A x} + x^\top B x$$

This defines the **black box**:



This function has several interesting structures

Sum structure

$$f(x) = f_1(x) + f_2(x) \text{ with } \begin{cases} f_1(x) = \sqrt{x^\top A x} \\ f_2(x) = x^\top B x \end{cases}$$

This function has several interesting structures

Sum structure

$$f(x) = f_1(x) + f_2(x) \text{ with } \begin{cases} f_1(x) = \sqrt{x^\top A x} \\ f_2(x) = x^\top B x \end{cases}$$

This defines a **sum black box**:

x

$f_1(x), f_2(x)$

$g_j(x) \in \partial f_j(x)_{j=1,2}$

This function has several interesting structures

Composite structure

$$f(x) = (h \circ c)(x) \text{ with } \begin{cases} c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1} \\ h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1} \end{cases}$$

for C smooth and h positively homogeneous

This function has several interesting structures

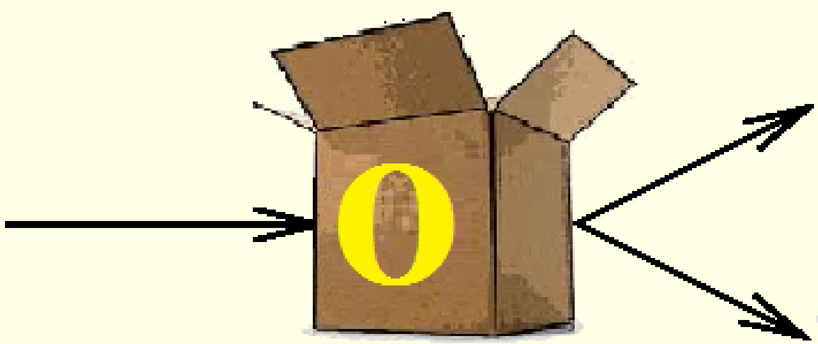
Composite structure

$$f(x) = (h \circ c)(x) \text{ with } \begin{cases} c(x) = (x, x^\top Bx) \in \mathbb{R}^{n+1} \\ h(C) = \sqrt{C_{1:n}^\top A C_{1:n}} + C_{n+1} \end{cases}$$

for C smooth and h positively homogeneous

This defines a **composite black box**:

x



$C := c(x)$ and $h(C)$

Jacobian $Dc(x)$ and

$G(C) \in \partial h(C)$

This function has several interesting structures

Inexact information

Suppose not all of A/B is known/accessible,

so that only **estimates** are available for f

This function has several interesting structures

Inexact information

Suppose not all of A/B is known/accessible,

so that only **estimates** are available for f

This defines a **noisy black box**:

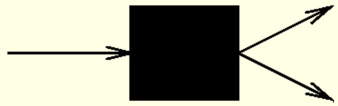
x

$f_x \approx f(x)$

$g_x \approx g(x) \in \partial f(x)$

Structured models for f

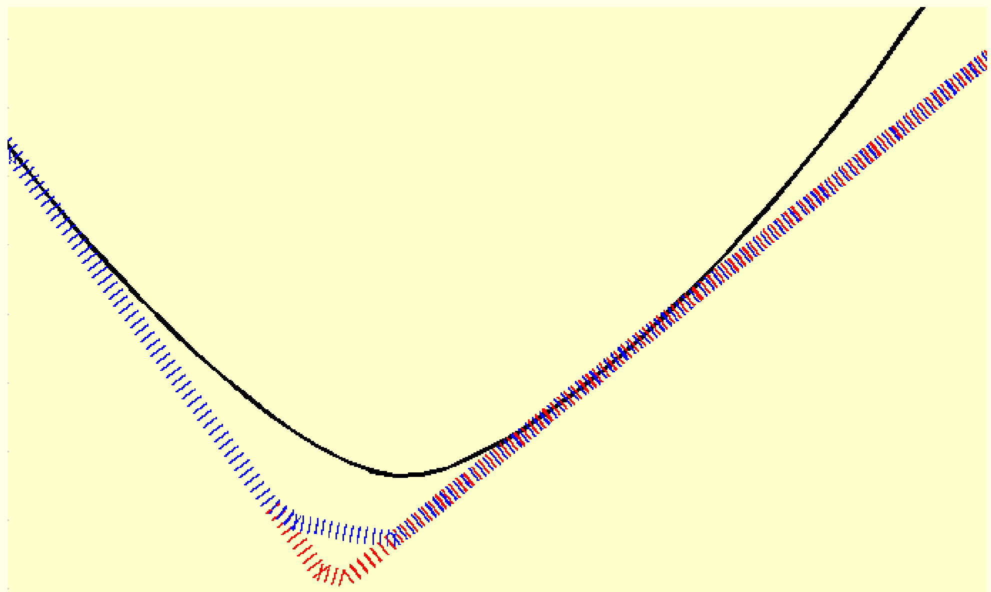
No structure



$$\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$$
$$= \max_i \left\{ (f_1^i + f_2^i) + (g_1^i + g_2^i)^\top (x - x^i) \right\}$$

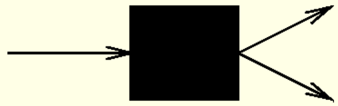
Sum structure

$$\mathbf{M}(x) = \max_i \left\{ f_1^i + g_1^{i\,\top}(x - x^i) \right\} + \max_i \left\{ f_2^i + g_2^{i\,\top}(x - x^i) \right\}$$



Structured models for f

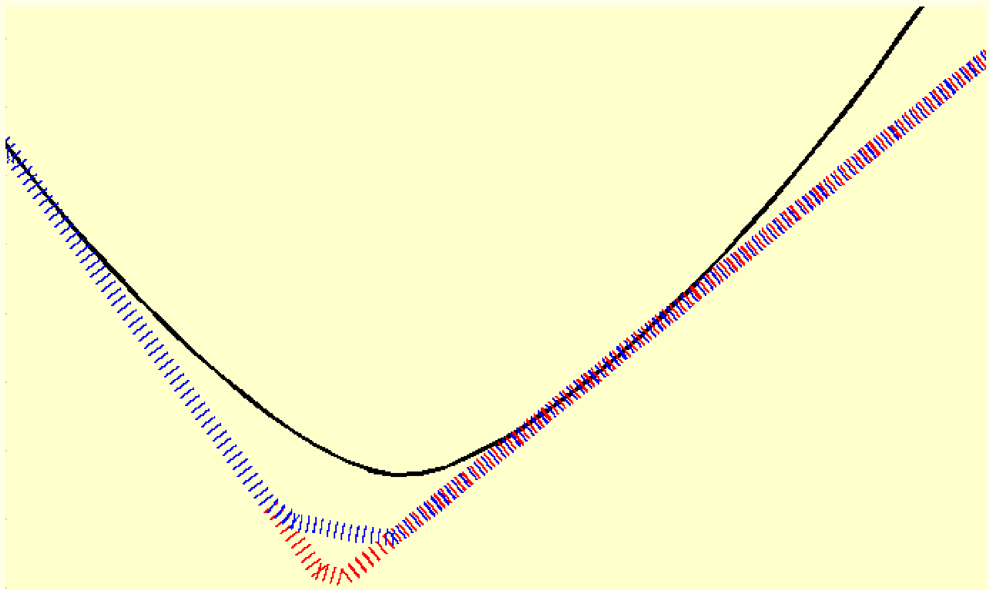
No structure



$$\mathbf{M}(x) = \max_i \left\{ f^i + {g^i}^\top (x - x^i) \right\}$$
$$= \max_i \left\{ (f_1^i + f_2^i) + (g_1^i + g_2^i)^\top (x - x^i) \right\}$$

Sum structure

$$\mathbf{M}(x) = \max_i \left\{ f_1^i + {g_1^i}^\top (x - x^i) \right\}$$
$$+ \max_i \left\{ f_2^i + {g_2^i}^\top (x - x^i) \right\}$$



Larger

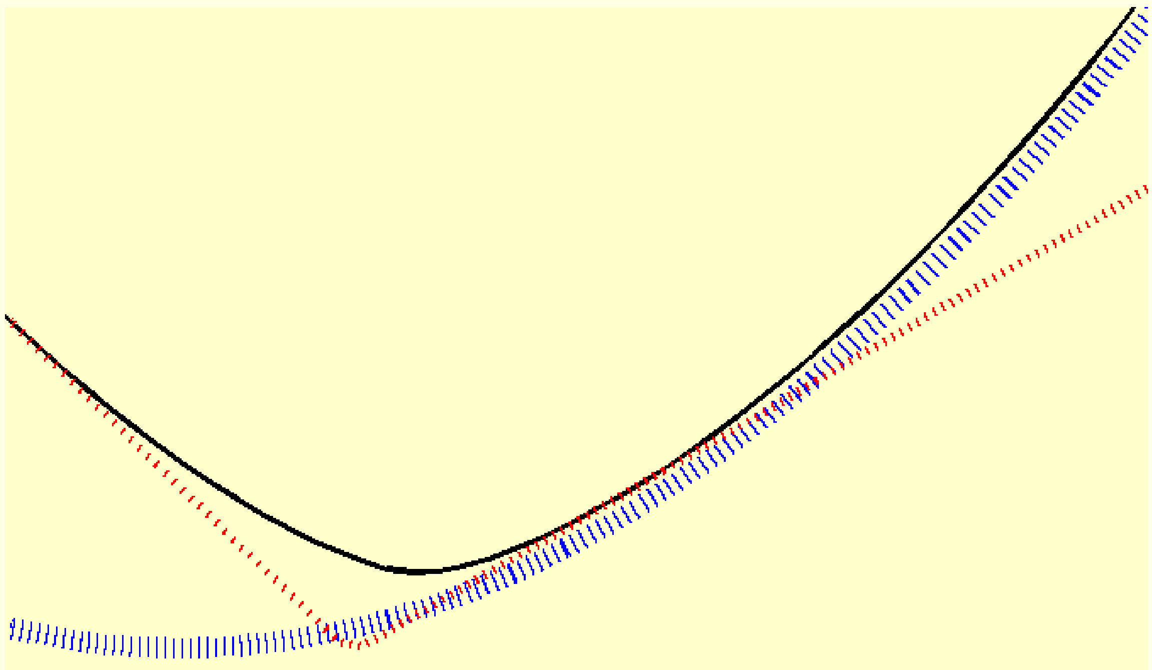
QP

Structured models for f

Composite structure

$$\mathbf{M}(x) = \max_i \left\{ G^{i\top}\Big(c(\hat{x}) + Dc(\hat{x})(x - \hat{x})\Big)\right\}$$

$$\approx h(c(\hat{x}) + Dc(\hat{x})(x - \hat{x}))$$

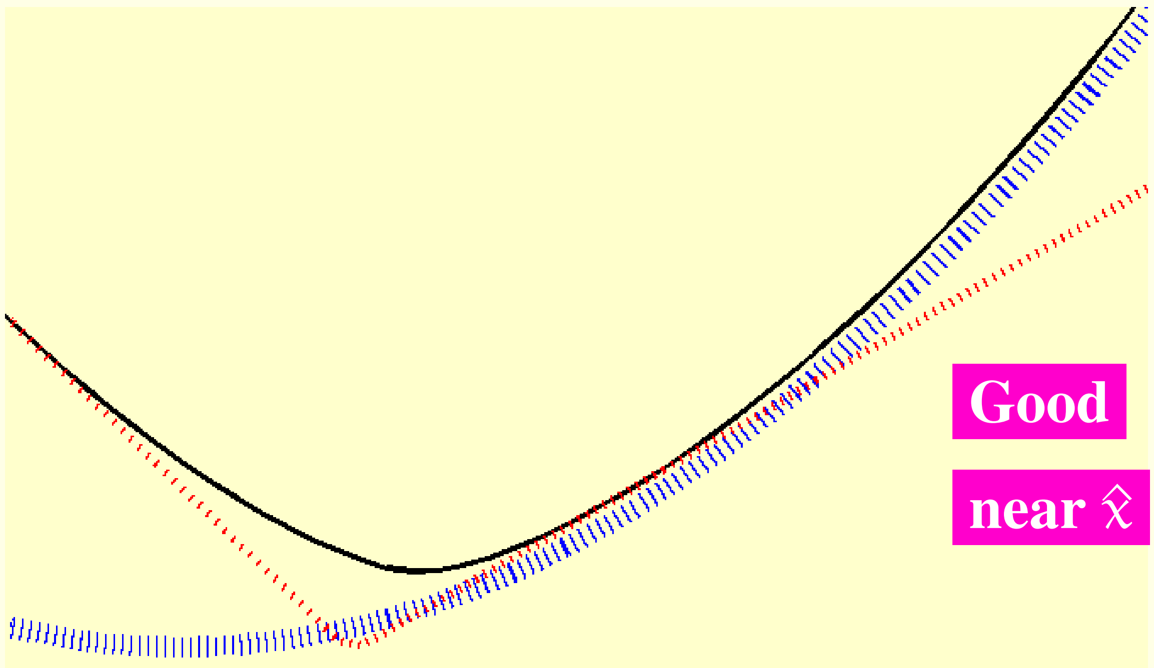


Structured models for f

Composite structure

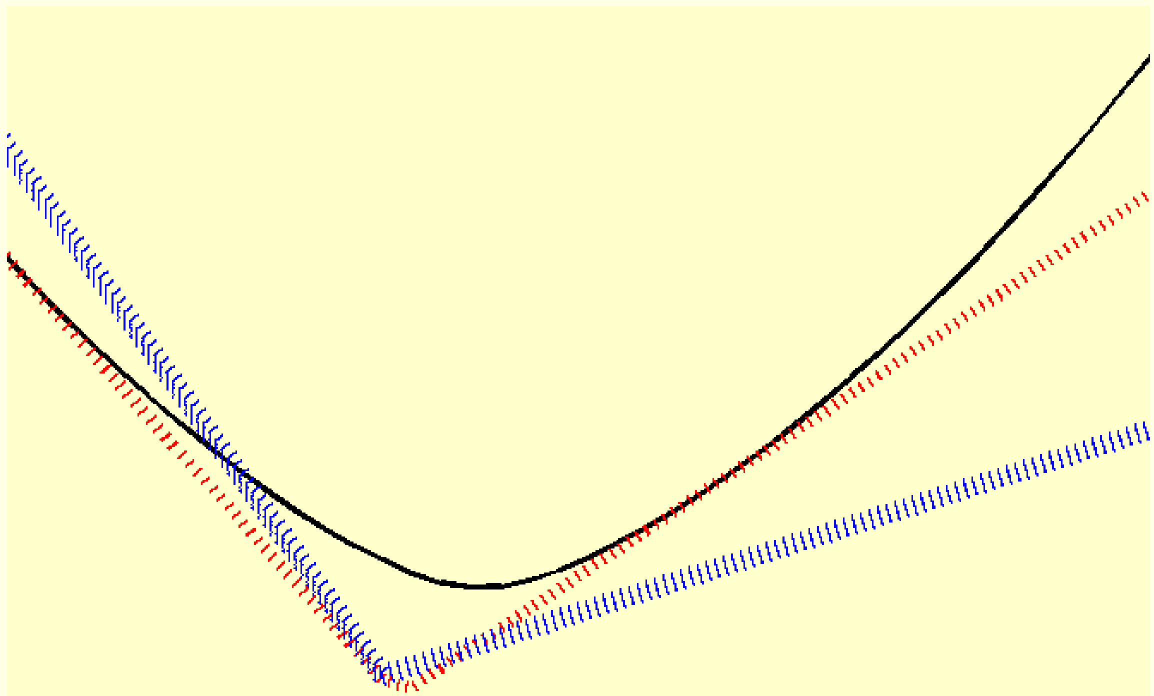
$$\mathbf{M}(x) = \max_i \left\{ G^{i\top}\Big(c(\hat{x}) + Dc(\hat{x})(x - \hat{x})\Big) \right\}$$

$$\approx h(c(\hat{x}) + Dc(\hat{x})(x - \hat{x}))$$



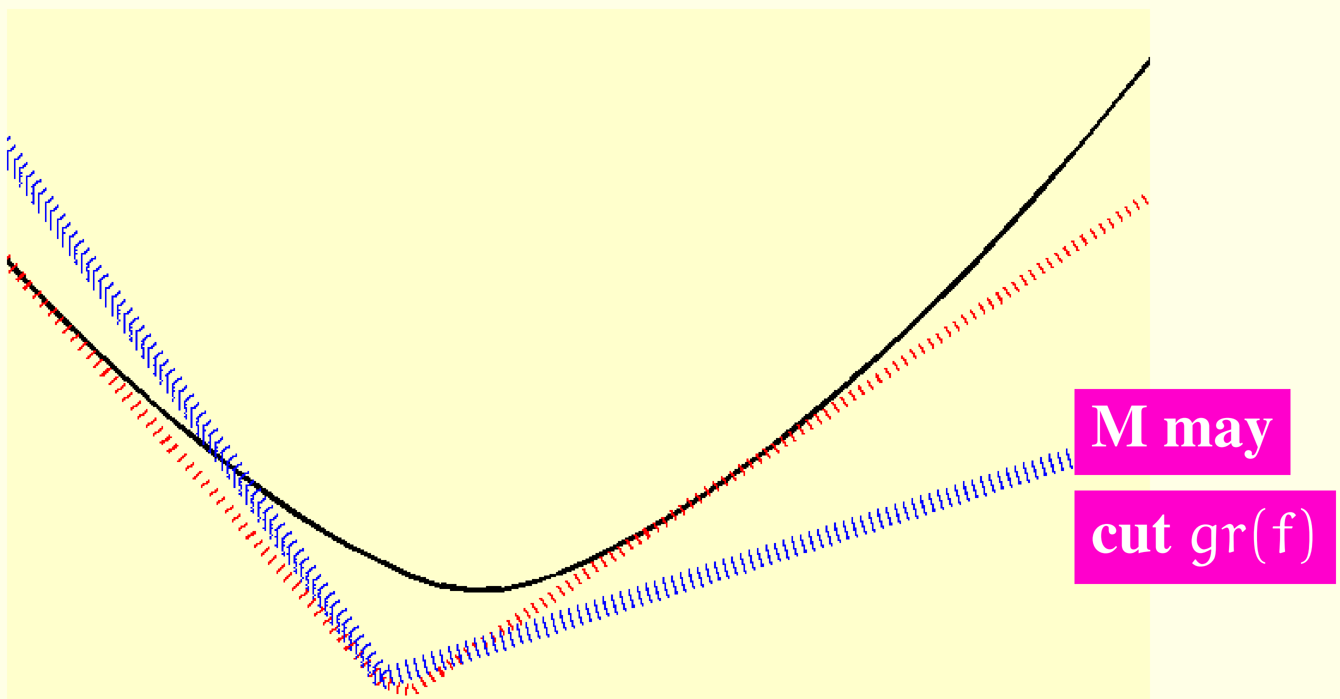
Inexact models for f

Inexact information  $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$



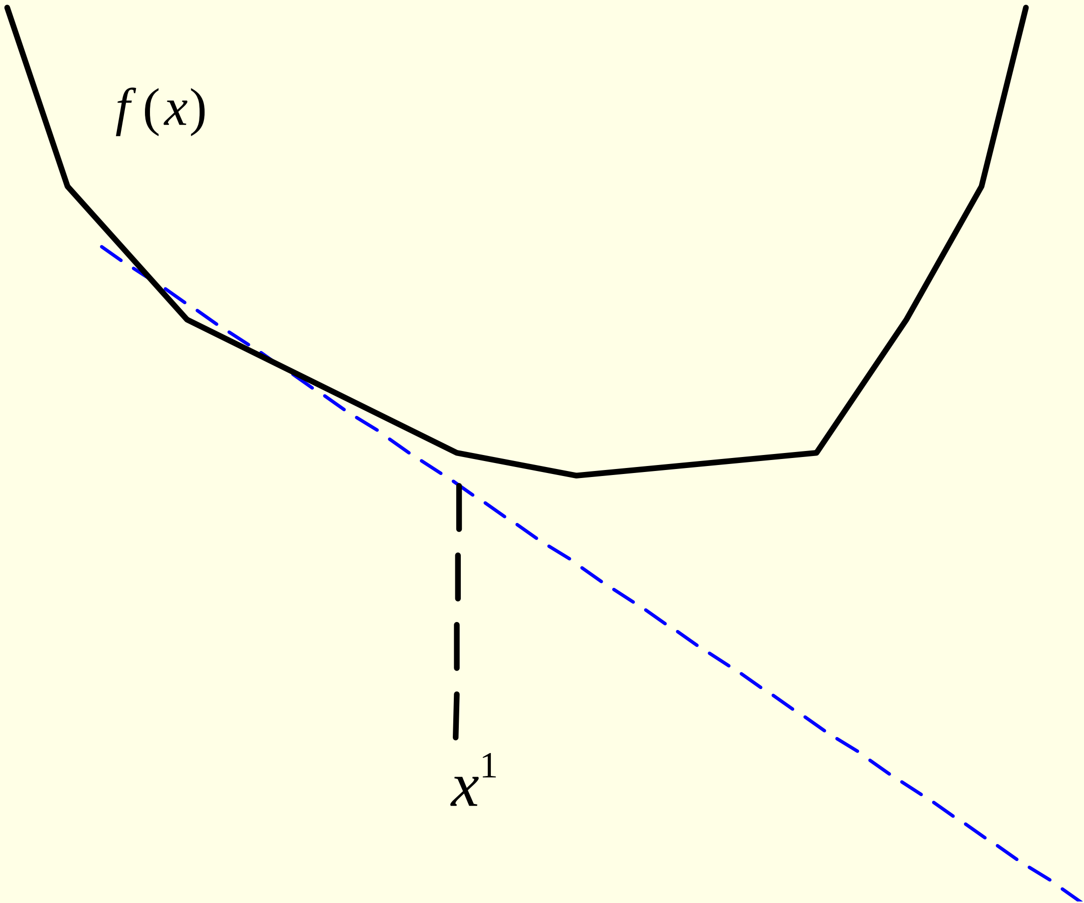
Inexact models for f

Inexact information  $\mathbf{M}(x) = \max_i \left\{ f^i + g^{i\top}(x - x^i) \right\}$

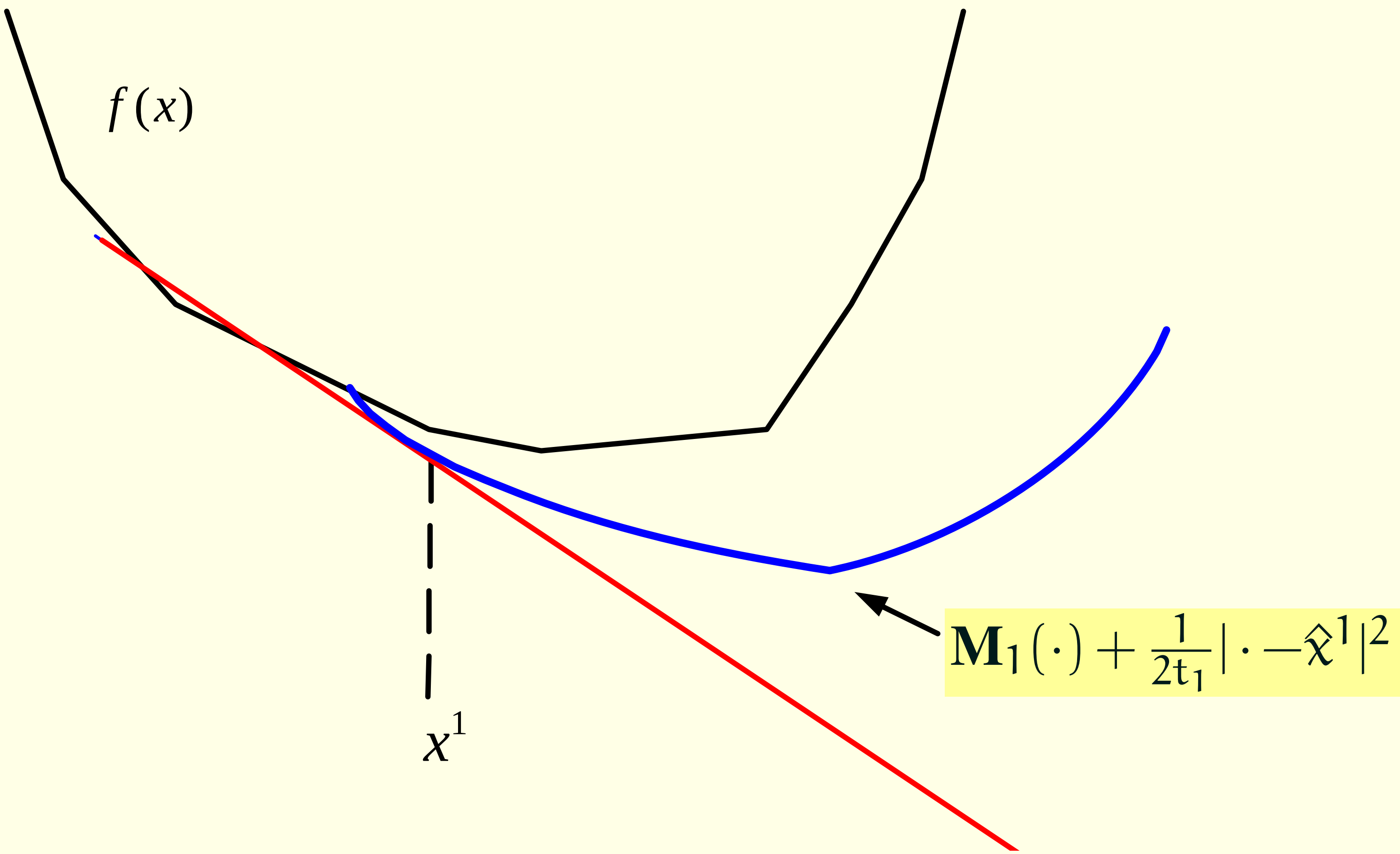


excessive noise is attenuated via stepsize t_k

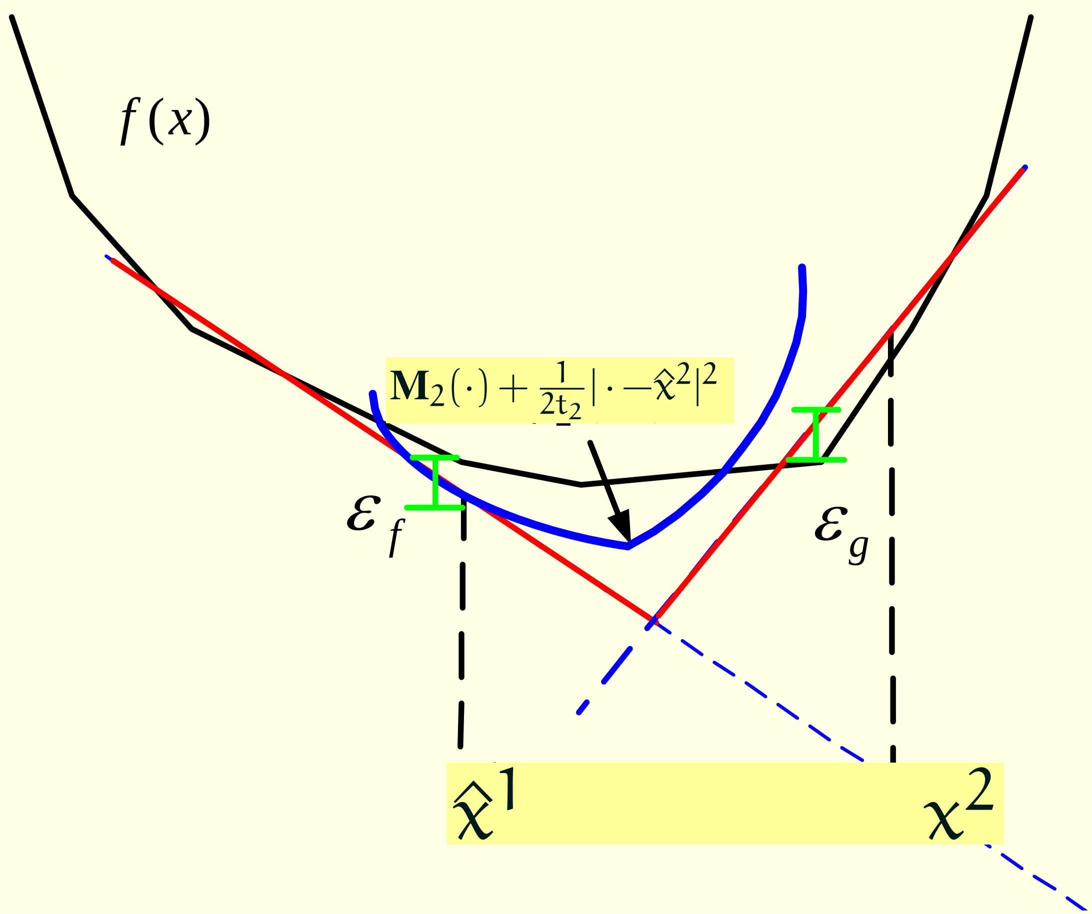
Bundle Methods with Inexact Information



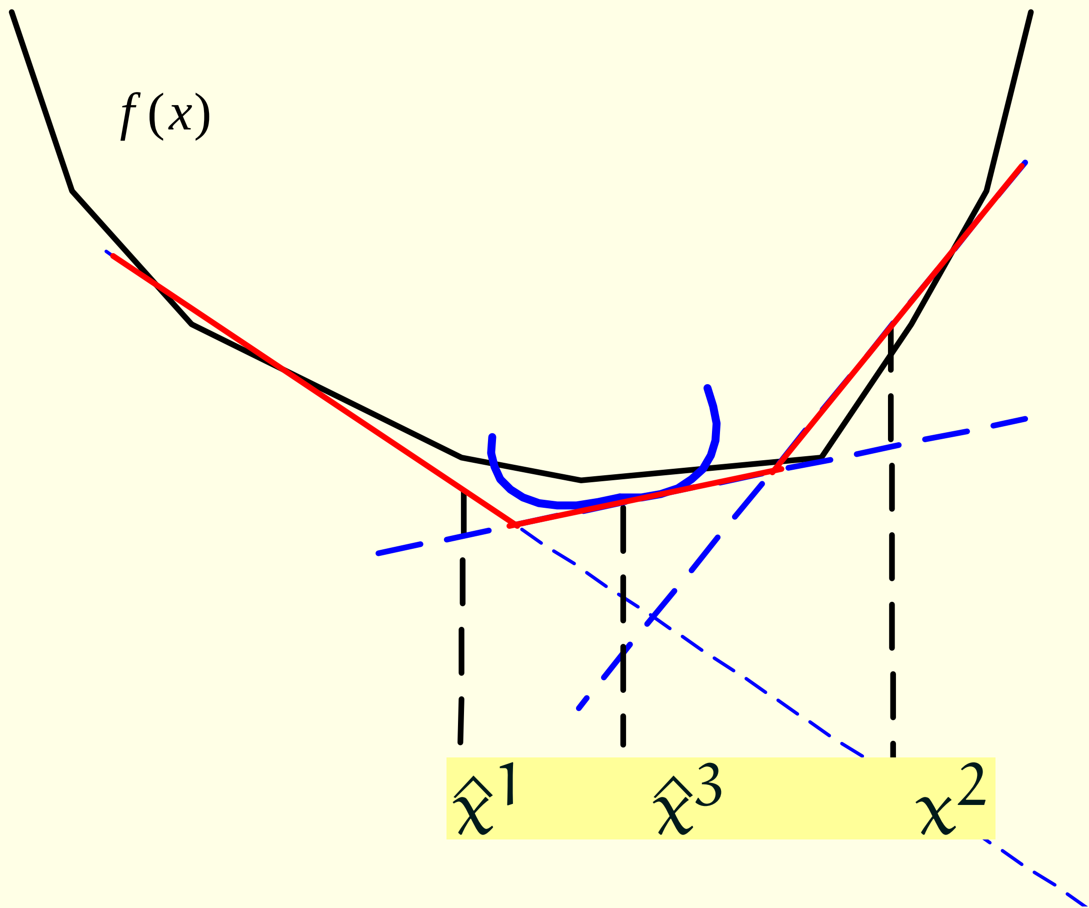
Bundle Methods with Inexact Information



Bundle Methods with Inexact Information



Bundle Methods with Inexact Information

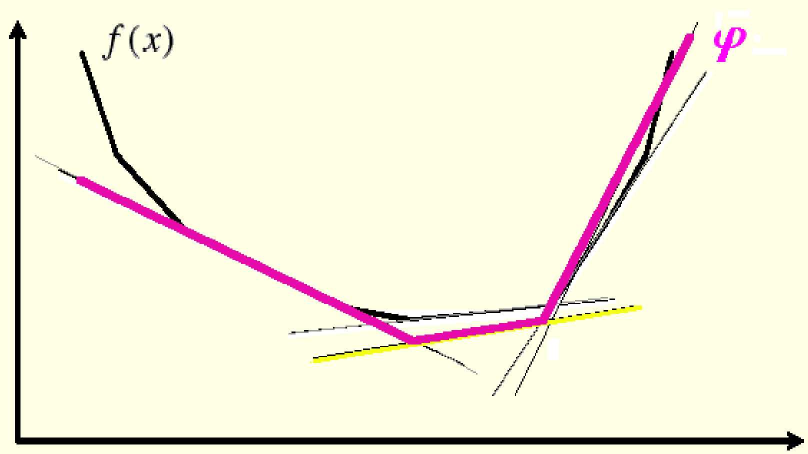


Controlling the impact of noise

$$x^{k+1} = \arg\min_x \mathbf{M}(x) + \frac{1}{2t_k}|x - \hat{x}|^2$$

now linearizations may be inexact:

x^j →  $f^j = f_{x^j}$, $g^j = g_{x^j}$ $\Longrightarrow \mathbf{M}(x) = \max_{j \le i}\left\{f^j + {g^j}^\top(x - x^j)\right\}$



and the model may be "wrong"

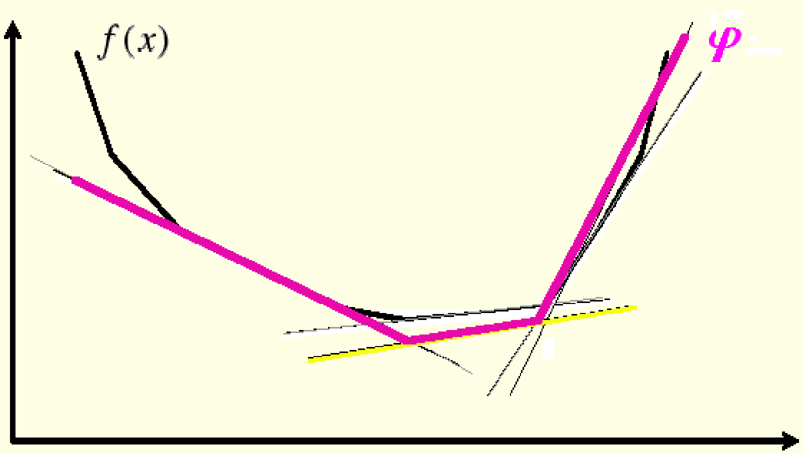
If too wrong: noise needs to be attenuated

Controlling the impact of noise

$$x^{k+1} = \arg\min_x \mathbf{M}(x) + \frac{1}{2t_k}|x - \hat{x}|^2$$

now linearizations may be inexact:

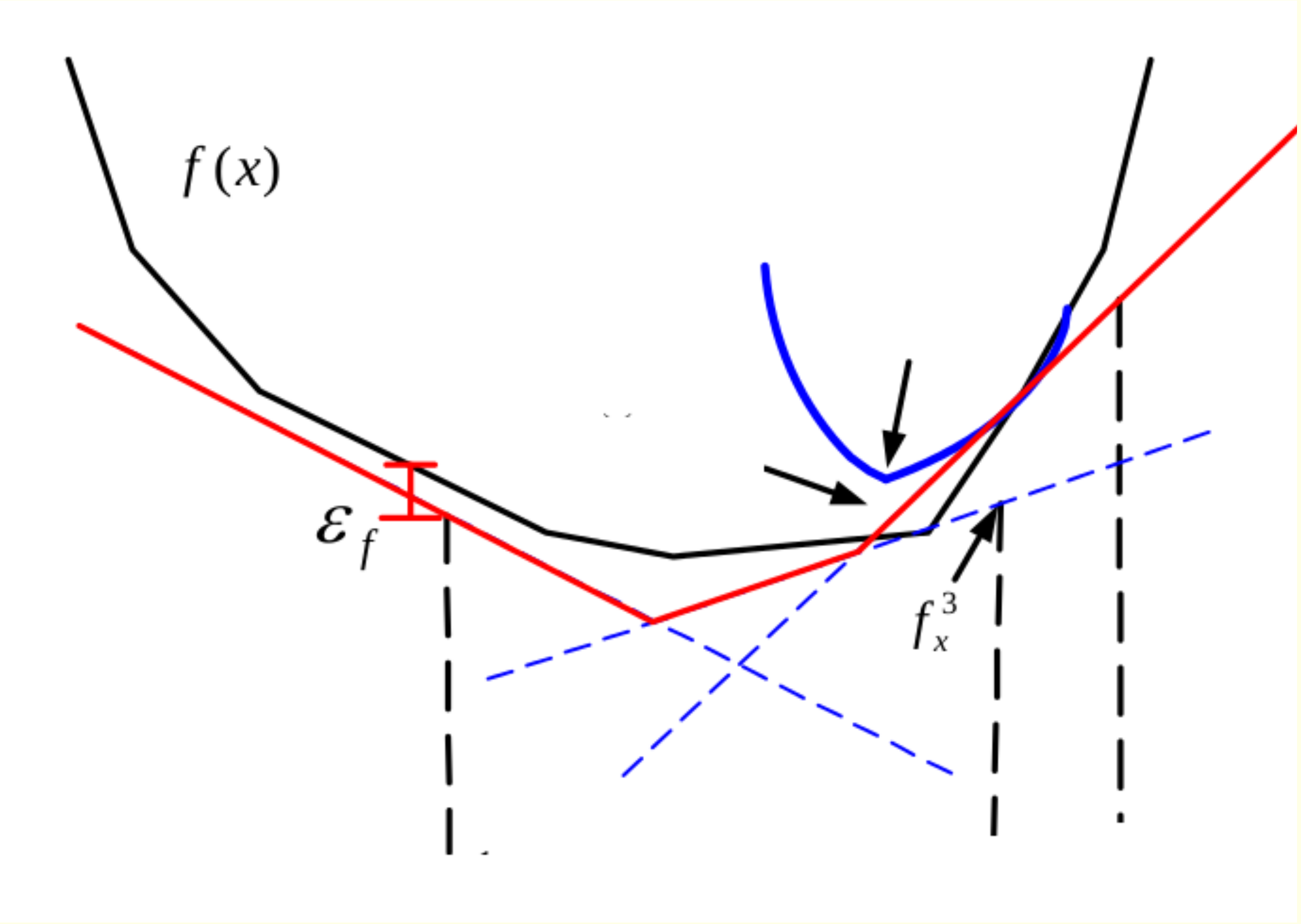
x^j  $f^j = f_{x^j}$, $g^j = g_{x^j}$ $\Longrightarrow \mathbf{M}(x) = \max_{j \le i}\left\{f^j + {g^j}^\top(x - x^j)\right\}$



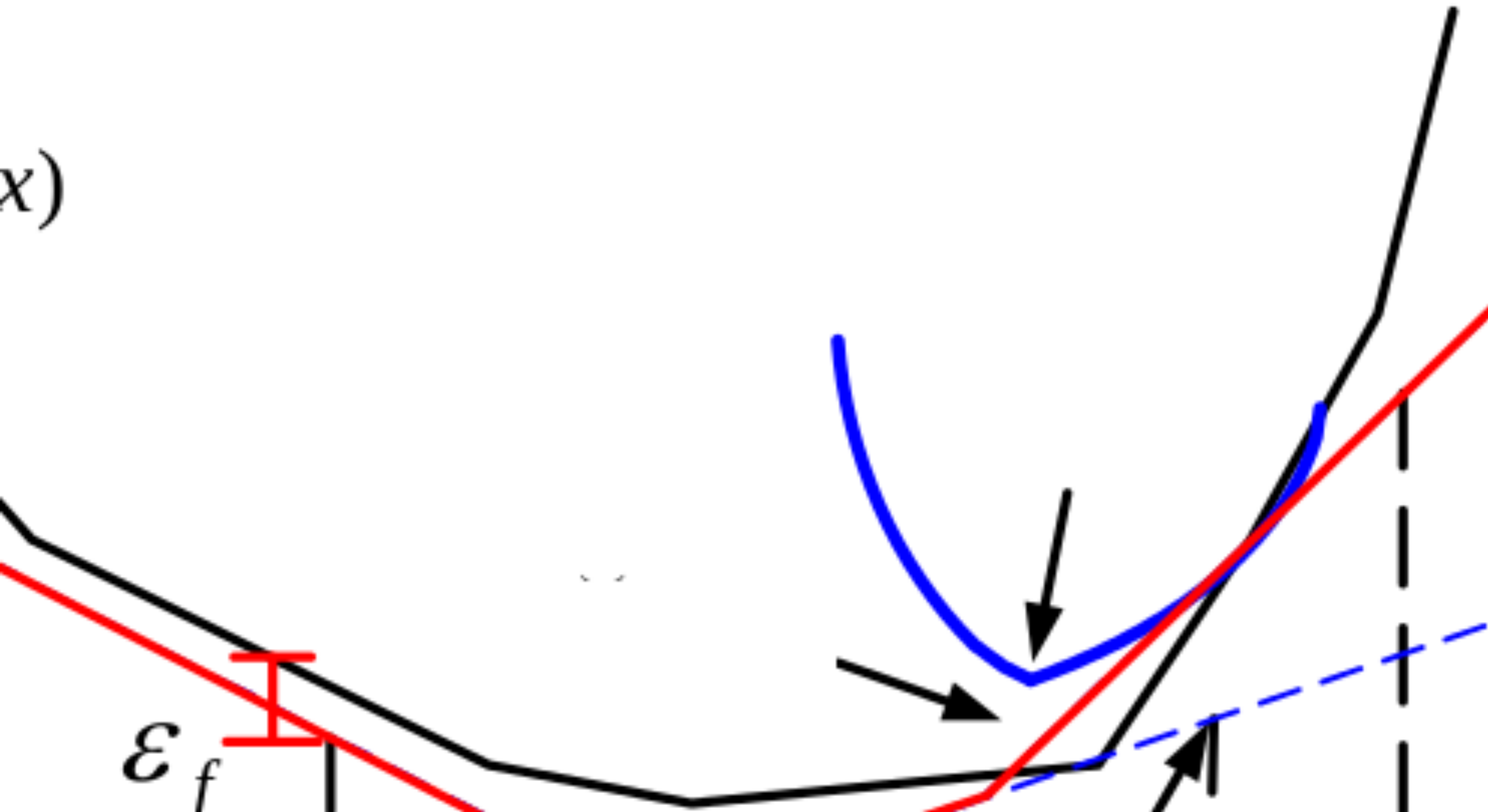
and the model may be "wrong"

Noise attenuated by increasing t, hence lowering QP value

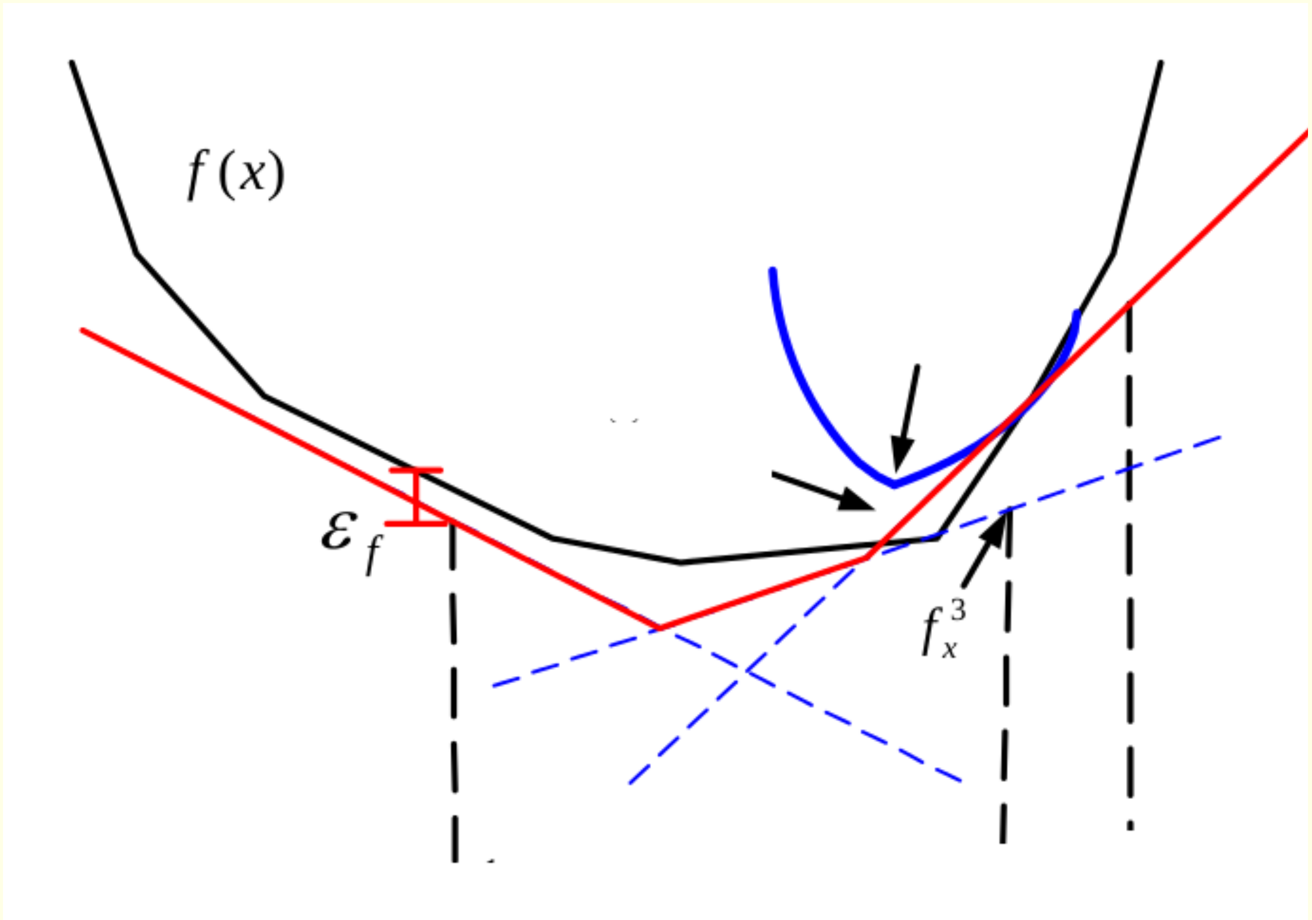
Detecting excessive noise by checking δ_k



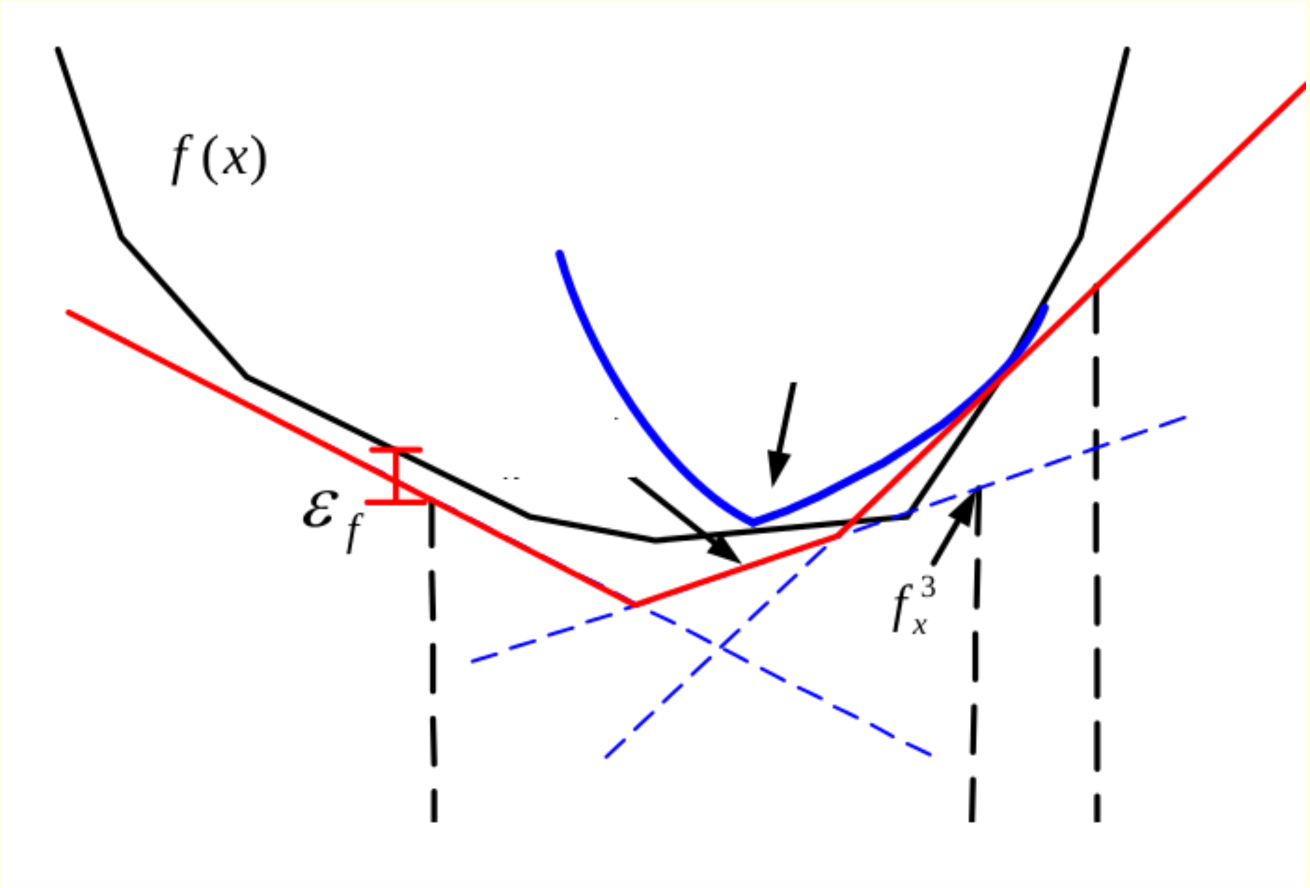
Detecting excessive noise: $\delta_k < 0$



Detecting excessive noise by checking δ_k



Detecting excessive noise by checking δ_k



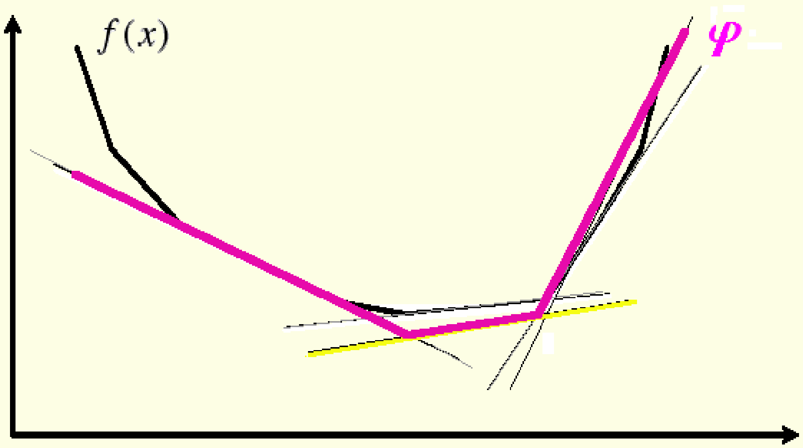
Controlling the impact of noise:

oracles with on-demand accuracy

$$x^{k+1} = \arg\min_x \mathbf{M}(x) + \frac{1}{2t_k}|x - \hat{x}|^2$$

now linearizations may be inexact:

x^j  $f^j = f_{x^j}$, $g^j = g_{x^j}$ $\Longrightarrow \mathbf{M}(x) = \max_{j \le i}\left\{f^j + g^{j\top}(x - x^j)\right\}$



If we have the ability of computing f_x/g_x

with more or less accuracy

compute (asympt.) exactly SS

and do not waste time in Null

On-demand accuracy scheme

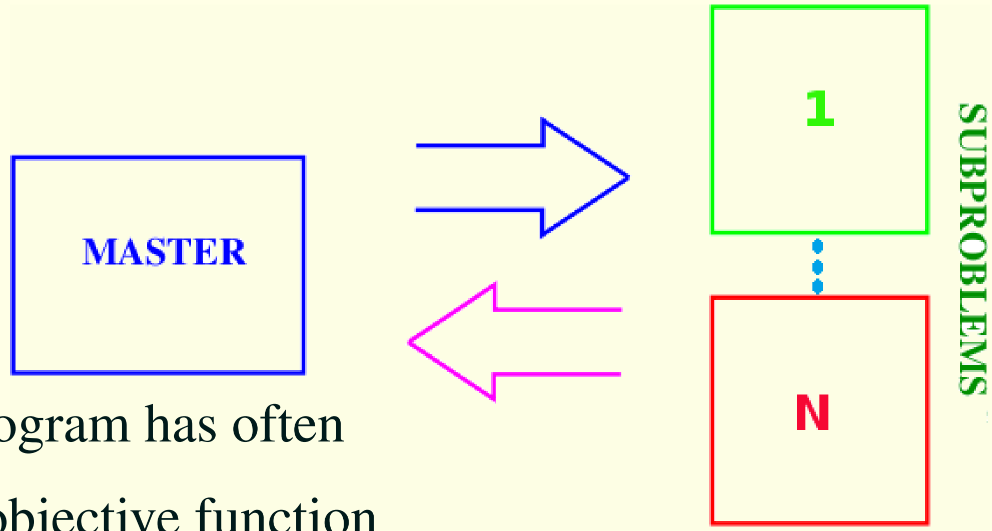
Explicit structure, induced by some **decomposition** method

by Lagrangian relaxation

by Benders decomposition

Principle: if a problem is difficult to solve directly,

solve instead a sequence of easier subproblems.



The master program has often

a nonsmooth objective function

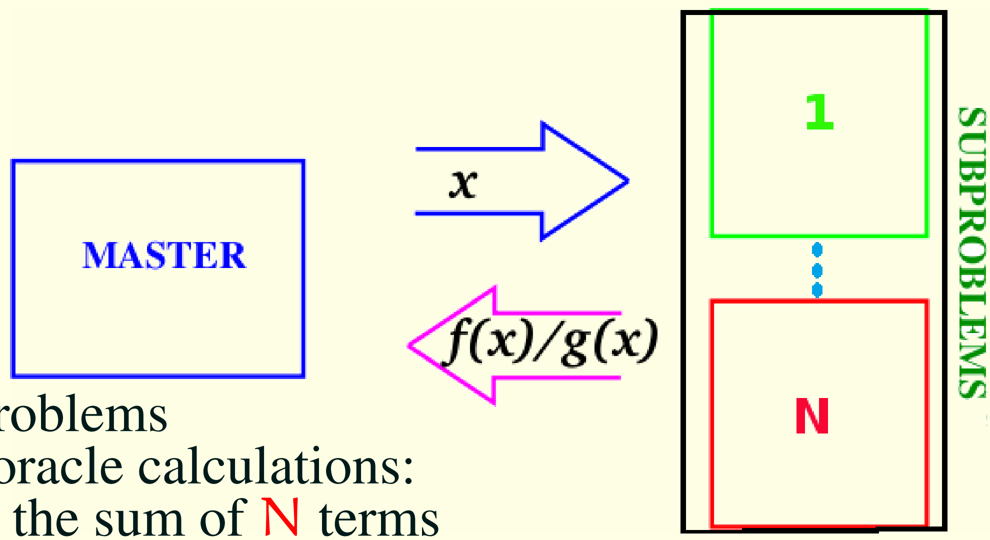
On-demand accuracy scheme

Explicit structure, induced by some **decomposition** method

by Lagrangian relaxation

by Benders decomposition

Principle: if a problem is difficult to solve directly,
solve instead a sequence of easier subproblems.



Separate subproblems
allow for fast oracle calculations:
f/g defined as the sum of N terms

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \begin{cases} \max & \displaystyle\sum_{j \in J} -\mathcal{C}^j(p^j) \\ & p^j \in \mathcal{P}^j, j \in J \\ & \displaystyle\sum_{j \in J} g^j(p^j) = 0 \end{cases}
$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \begin{cases} \min \quad \displaystyle\sum_{j\in J} \mathcal{C}^j(p^j) \\ \qquad\quad p^j \in \mathcal{P}^j, j \in J \\ \qquad\quad \displaystyle\sum_{j\in J} g^j(p^j) = 0 \end{cases} \qquad \longleftarrow x
$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \begin{cases} \max \quad \displaystyle\sum_{j \in J} -\mathcal{C}^j(p^j) \\ \qquad\quad p^j \in \mathcal{P}^j, j \in J \\ \qquad\quad \boxed{\displaystyle\sum_{j \in J} g^j(p^j) = 0} \end{cases} \quad \boxed{\leftarrow x}
$$

often exhibit separable structure after dualization

Lagrangian Relaxation Example

Real-life optimization problems

$$(\texttt{primal}) \quad \begin{cases} \max & \displaystyle\sum_{j\in J} -\mathcal{C}^j(p^j) \\ & p^j \in \mathcal{P}^j, j \in J \\ & \boxed{\displaystyle\sum_{j\in J} g^j(p^j) = 0} \qquad \boxed{\leftarrow x} \end{cases}$$

often exhibit separable structure after dualization

$$(\texttt{dual}) \quad \min_x \sum_{j\in J} \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$$

Lagrangian Relaxation Example

Real-life optimization problems

$$
(\texttt{primal}) \quad \begin{cases} \max & \sum_{j\in J} -\mathcal{C}^j(p^j) \\ & p^j \in \mathcal{P}^j, j \in J \\ & \boxed{\sum_{j\in J} g^j(p^j) = 0} \quad \boxed{\leftarrow x} \end{cases}
$$

often exhibit separable structure after dualization

$$
(\texttt{dual}) \quad \min_x \sum_{j\in J} f^j(x)
$$

$$
f^j(x) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}
$$

Energy management problems

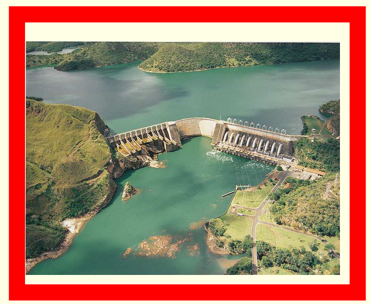
Typically, evaluating $f^j(x) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$

corresponds to local subproblems, related to one power plant, requiring sometimes heavy calculations

Energy management problems

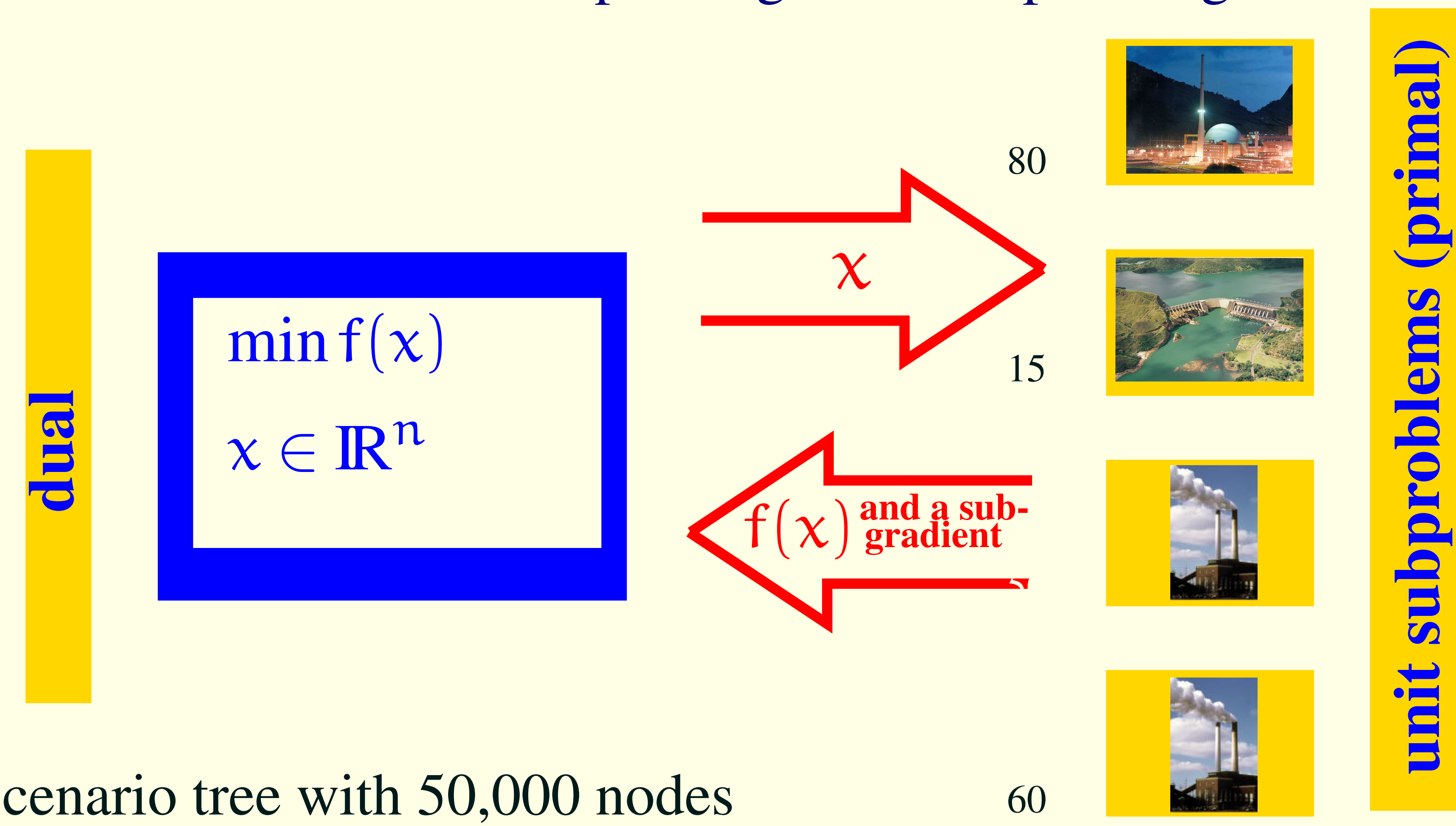
Typically, evaluating $f^j(x) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$

corresponds to local subproblems, related to one power plant, requiring sometimes heavy calculations

One subgradient for free: $g^j(p^j(x))$ once a solution $p^j(x)$ is available

Often, most of the CPU time is spent in the oracle calculations. For mid-term power generation planning:

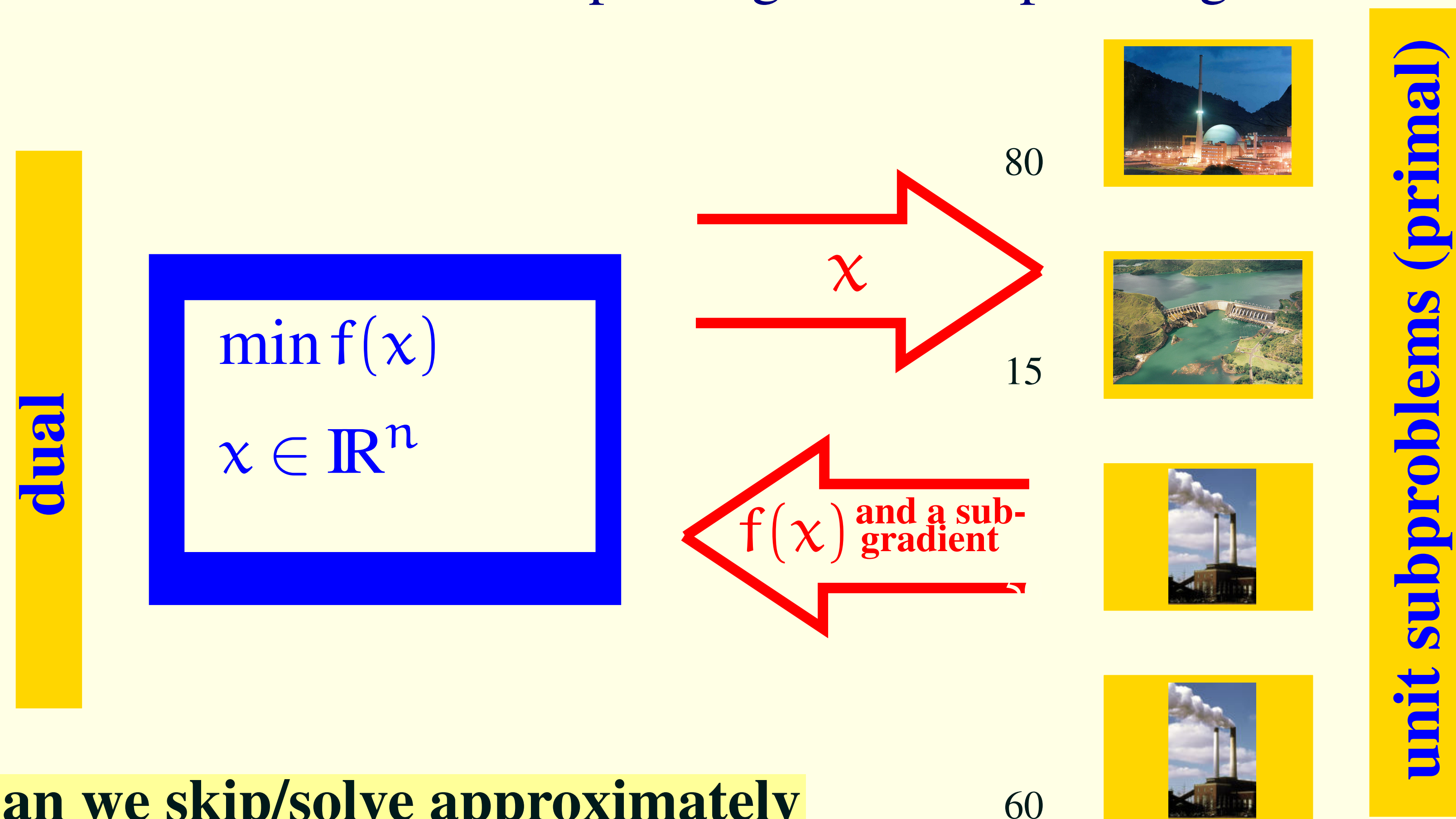


Scenario tree with 50,000 nodes

Nuclear subproblems are LPs with 100,000 variables

and 300,000 constraints, consuming **99%** total running time

Often, most of the CPU time is spent in the oracle calculations. For mid-term power generation planning:

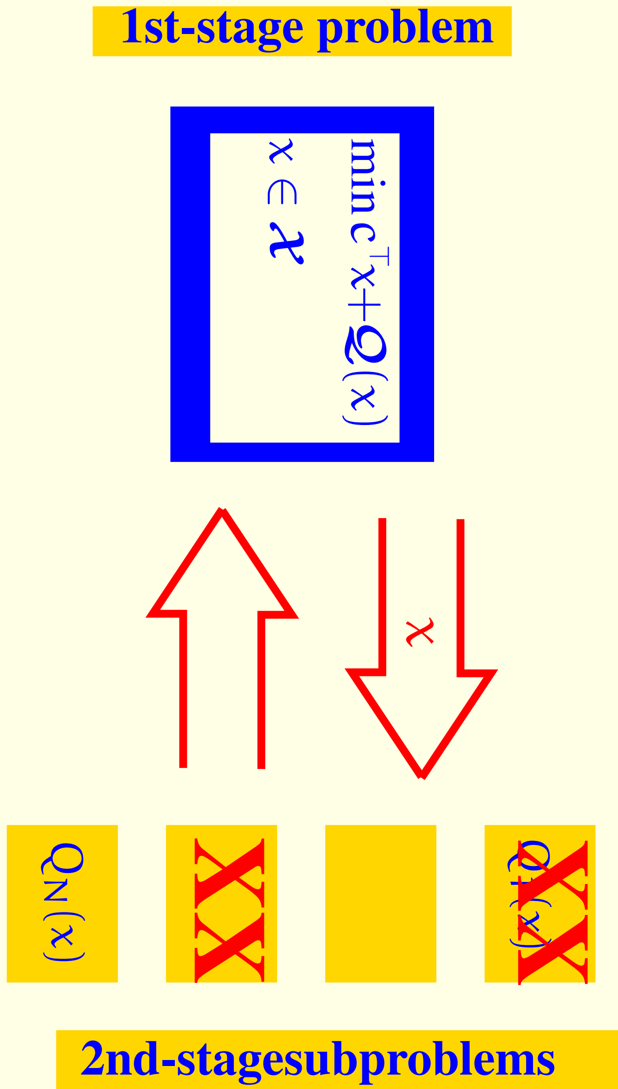


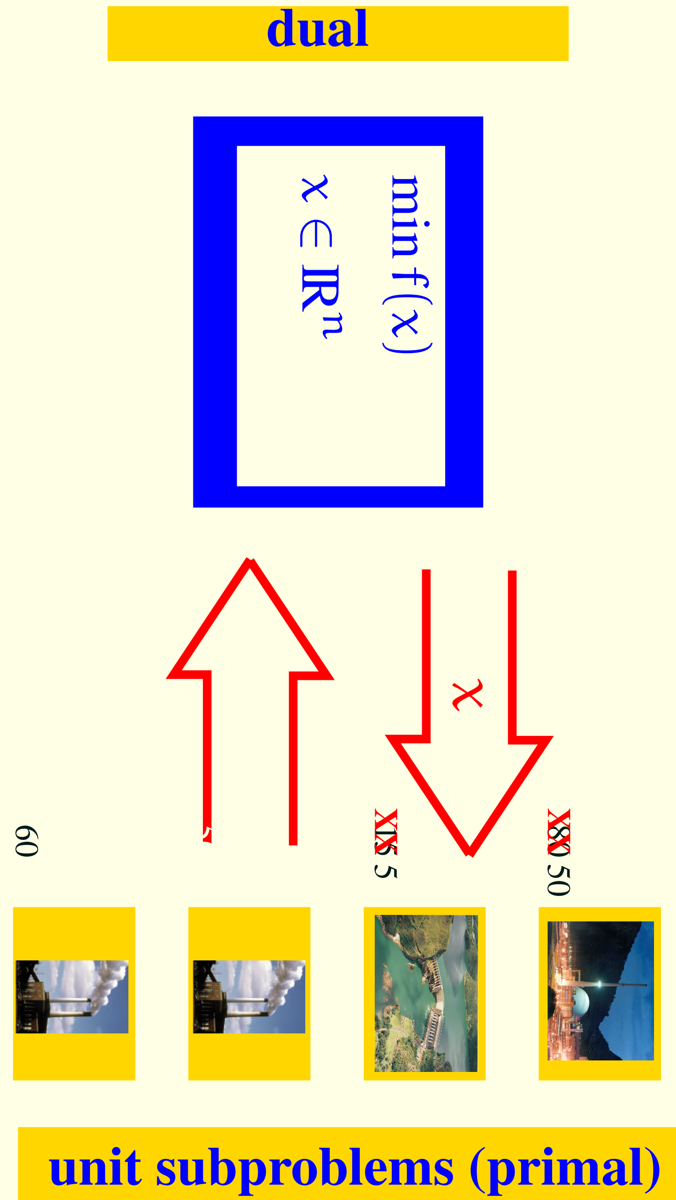
Can we skip/solve approximately

nuclear subproblems,

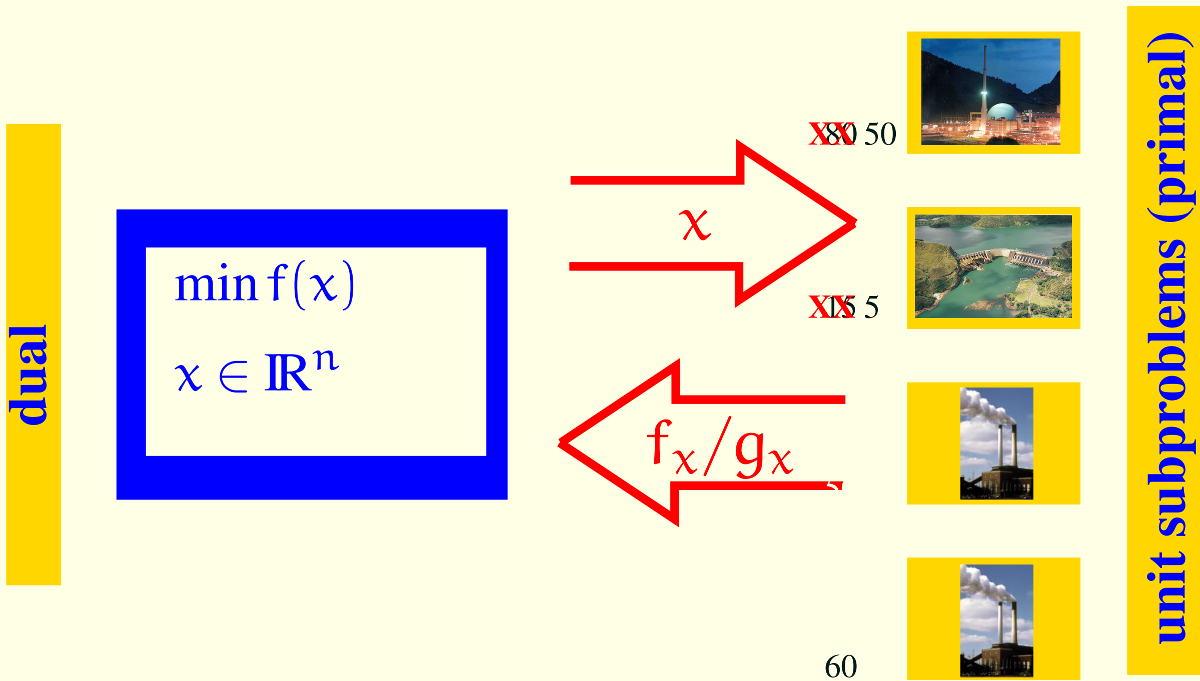
consuming LESS running time without losing accuracy?

Can we adapt the oracle response to the solver needs?

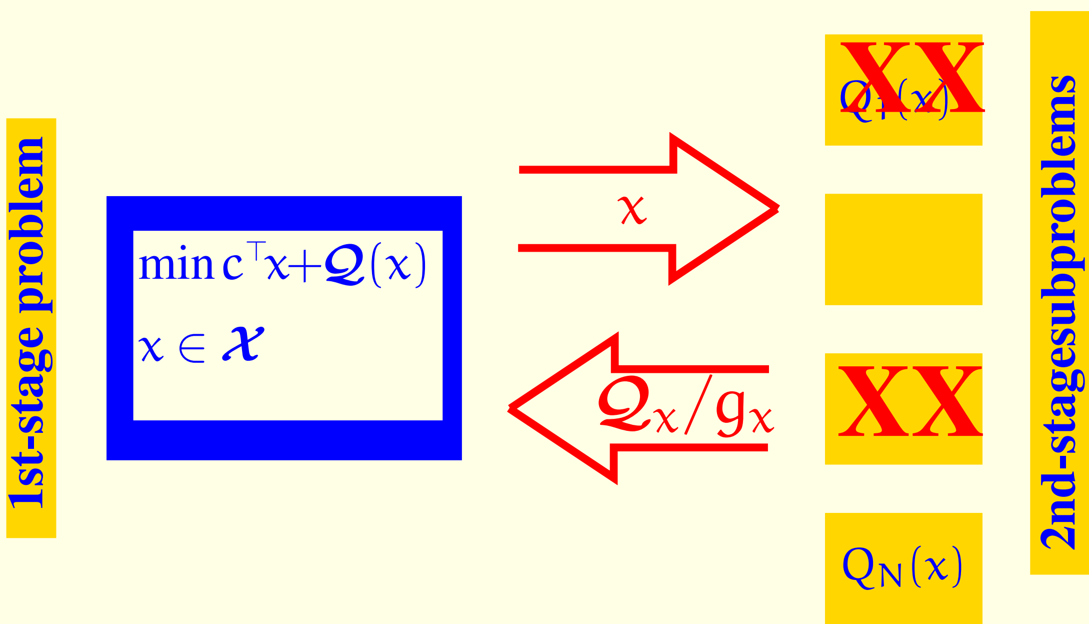




Can we adapt the oracle response to the solver needs?



now the oracle returns INEXACT values



Can we adapt the oracle response to the solver needs? **YES!**

with a NSO method capable of handling

oracles with On-demand Accuracy

Can we adapt the oracle response to the solver needs? **YES!**

with a NSO method capable of handling **oracles with On-demand Accuracy** created over noisy black-boxes

x

$$f_x \approx f(x)$$

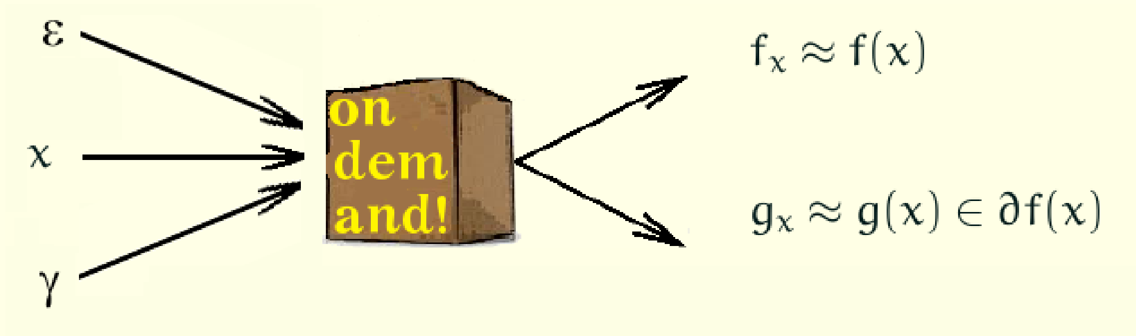
$$g_x \approx g(x) \in \partial f(x)$$

when we have the ability of computing f_x/g_x with more or less accuracy

Oracle with on-demand accuracy

For $f^j(x) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$

we design a noisy black box that gets additional input:



an **error bound ε and a descent target γ** such that

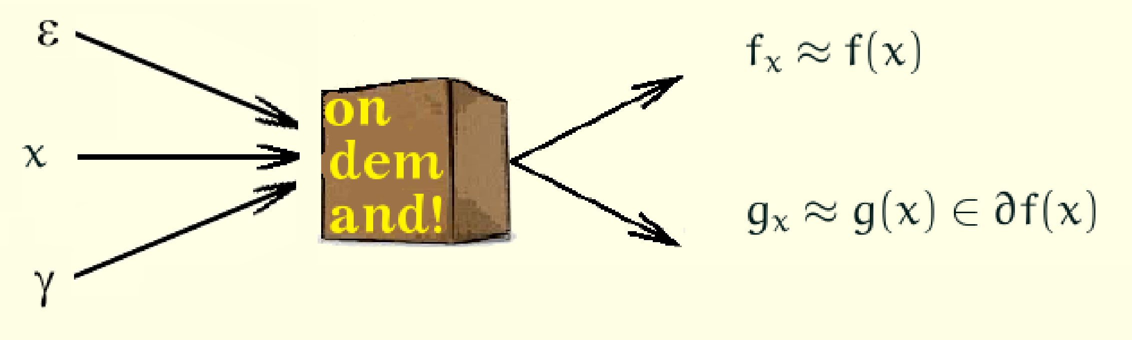
$$\left.\begin{array}{l} f_x = f(x) - \eta(x) \\ g_x \in \partial_{\eta(x)} f(x) \end{array}\right\} \quad \text{for all } x, \text{ with } \eta(x) \geq 0$$

$$\eta(x) \leq \varepsilon \qquad \text{if } x \text{ gave enough descent: } f_x \leq \gamma$$

Oracle with on-demand accuracy

For $f^j(x) := \begin{cases} \max & -\mathcal{C}^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \\ & p^j \in \mathcal{P}^j \end{cases}$

we design a noisy black box that gets additional input:



an **error bound ε and a descent target γ** such that

$$\left.\begin{array}{l} f_x = f(x) - \eta(x) \\ g_x \in \partial_{\eta(x)} f(x) \end{array}\right\} \quad \text{for all } x\text{, with } \eta(x) \geq 0 \ \textbf{unknown}$$

$$\eta(x) \leq \varepsilon \qquad \text{if } x \text{ gave enough descent: } f_x \leq \gamma$$

Classical Bundle Method

0 Choose x^1, set $k = 1$ $\hat{x}^1 = x^1$.

1 Compute $x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$

2 If $\delta_k = f(\hat{x}^k) - \mathbf{M}_k(x^{k+1}) \leq \mathtt{tol}$ STOP

3 Call the oracle at x^{k+1}.

If $f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k$, set $\hat{x}^{k+1} = x^{k+1}$ • (Serious Step)
Otherwise, maintain $\hat{x}^{k+1} = \hat{x}^k$ (Null Step)

4 Define $\mathbf{M}_{k+1}$, t_{k+1}, make $k = k+1$, and loop to 1.

Partly Exact Bundle Method

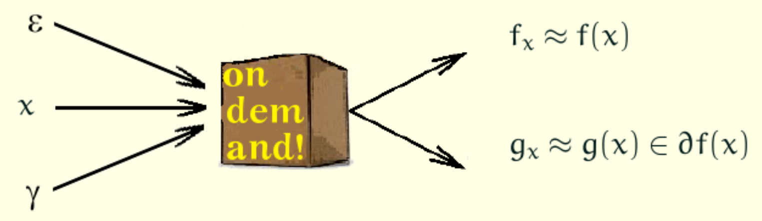
0 Choose x^1, ε_1, set $k=1$ $\hat{x}^1 = x^1$.

1 Compute $x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x-\hat{x}^k|^2$

2 If $\delta_k = f_{\hat{x}^k} - \mathbf{M}_k(x^{k+1})$ **"is too negative"** $t_{k+1} = 10t_k$, go to **1**

Otherwise, if $\delta_k \le \mathtt{tol}$ STOP

3 Call the oracle at x^{k+1} **with $\gamma = f_{\hat{x}^k} - m\delta_k$, decreasing ε_k**



If $f_{x^{k+1}} \le f_{\hat{x}^k} - m\delta_k$, set $\hat{x}^{k+1} = x^{k+1}$ • (Serious Step)
Otherwise, maintain $\hat{x}^{k+1} = \hat{x}^k$ (Null Step)

4 Define $\mathbf{M}_{k+1}$, t_{k+1}, make $k = k+1$, and loop to 1.

Partly Exact Bundle Method

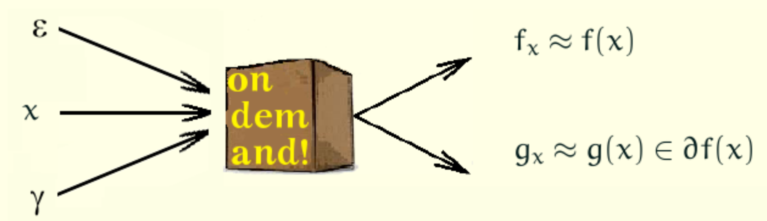
0 Choose x^1, ε_1, set $k = 1$ $\hat{x}^1 = x^1$.

1 Compute $x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$

2 If $\delta_k = f_{\hat{x}^k} - \mathbf{M}_k(x^{k+1})$ **"is too negative"** $t_{k+1} = 10t_k$, go to **1**

Otherwise, if $\delta_k \leq \mathrm{tol}$ STOP

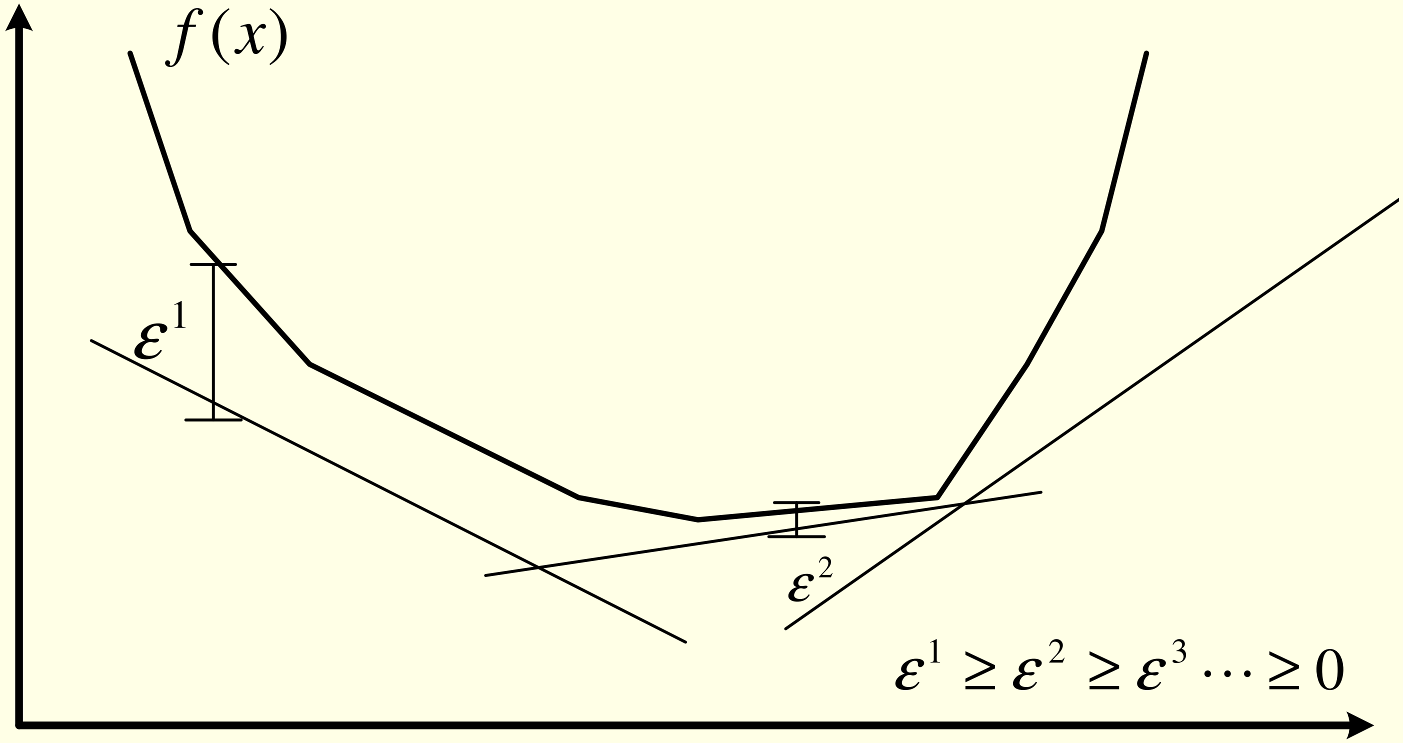
3 Call the oracle at x^{k+1} **with $\gamma = f_{\hat{x}^k} - m\delta_k$, decreasing ε_k**



If $f_{x^{k+1}} \leq f_{\hat{x}^k} - m\delta_k$, set $\hat{x}^{k+1} = x^{k+1}$ • (Serious Step)
Otherwise, maintain $\hat{x}^{k+1} = \hat{x}^k$ (Null Step)

4 Define $\mathbf{M}_{k+1}$, t_{k+1}, make $k = k + 1$, and loop to 1.

as $\varepsilon_k \to 0$, $f_{\hat{x}^k} \to f(\hat{x}^k)$, the method finds exact solutions!


$f(x)$
ε^1
ε^2
$\varepsilon^1 \geq \varepsilon^2 \geq \varepsilon^3 \cdots \geq 0$

Oracle with on-demand accuracy: versatility

$$\left.\begin{aligned} f_x &= f(x) - \eta(x) \\ g_x &\in \partial_{\eta(x)} f(x) \end{aligned}\right\} \quad \text{for all } x, \text{ with } \eta(x) \geq 0$$

$$\eta(x) \leq \varepsilon \qquad \text{if } x \text{ gave enough descent: } f_x \leq \gamma$$

We control both ε and γ, which can vary with x:

newline – $\varepsilon_x = 0$ and $\gamma_x = +\infty$ is an exact oracle.

newline – $\varepsilon_x \to 0$ along the iterative process and $\gamma_x = +\infty$ is an asymptotically exact oracle

newline – $\varepsilon_x = 0$ with finite γ_x gives a partly inexact oracle

newline – $\varepsilon_x > 0$ unknown, but bounded, with $\gamma_x = +\infty$ is an inexact oracle

Theoretical Results

Convex proximal bundle methods in depth: a unified analysis for inexact oracles

W. de Oliveira, C. Sagastizábal, C. Lemaréchal

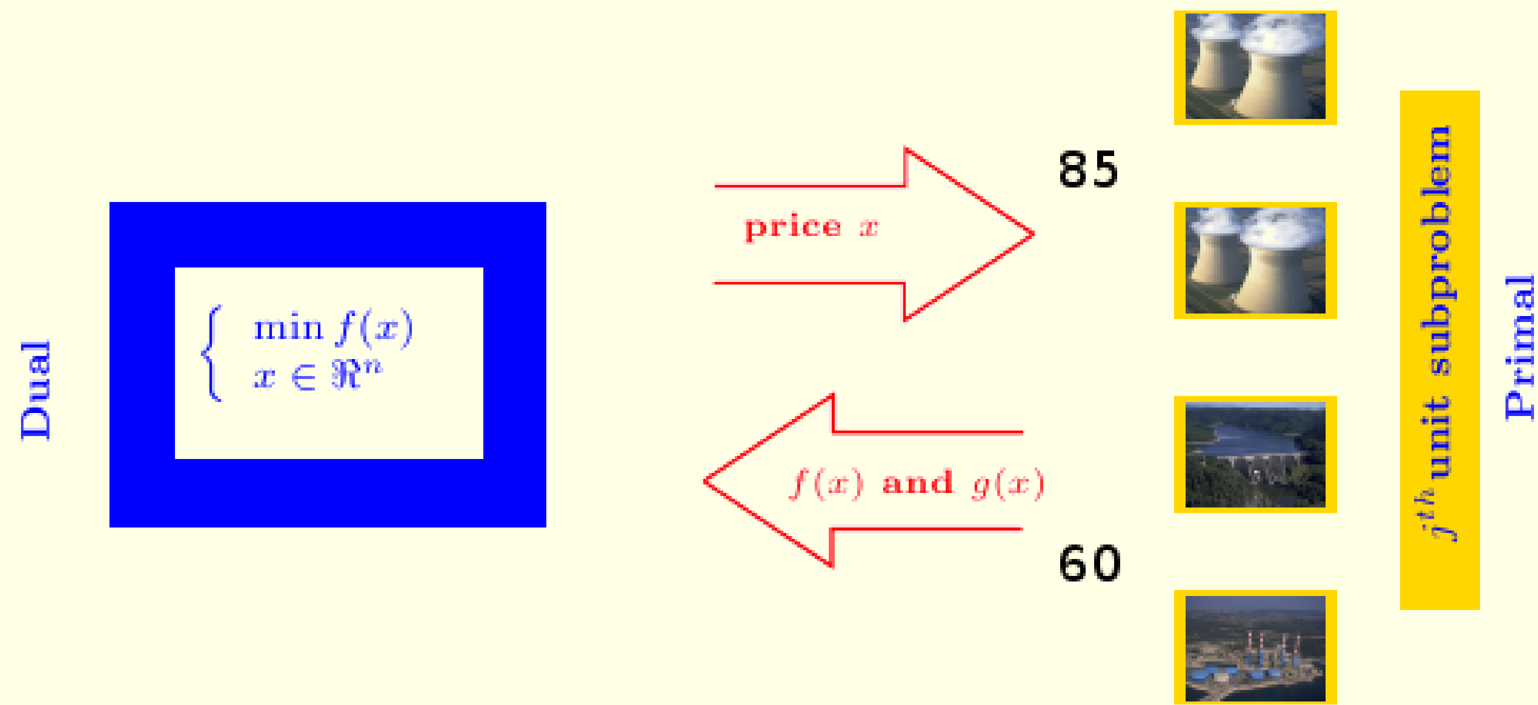
MathProg 148, pp 241-277, 2014

General and versatile convergence theory for inexact oracles, including

- asymptotically exact ones (driving ε to 0).
- inexact oracles (convergence within accuracy bound)
- lower an dupper oracles
- previous exact bundle variants
- new ones

Application in Energy I

Mid-term planning for power generation

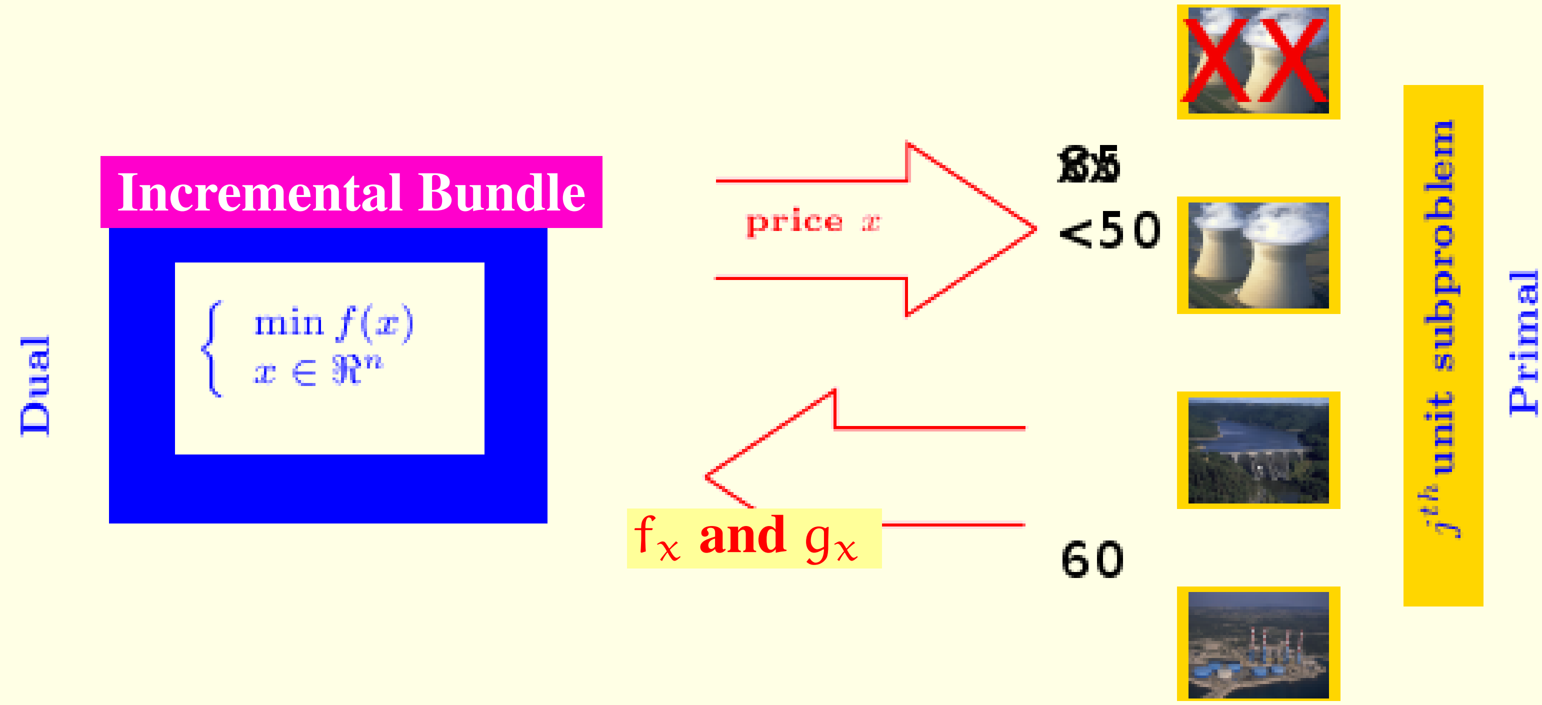


Scenario tree with 50,000 nodes

Nuclear LPs with 100,000 variables and 300,000 constraints

Application in Energy I

Mid-term planning for power generation

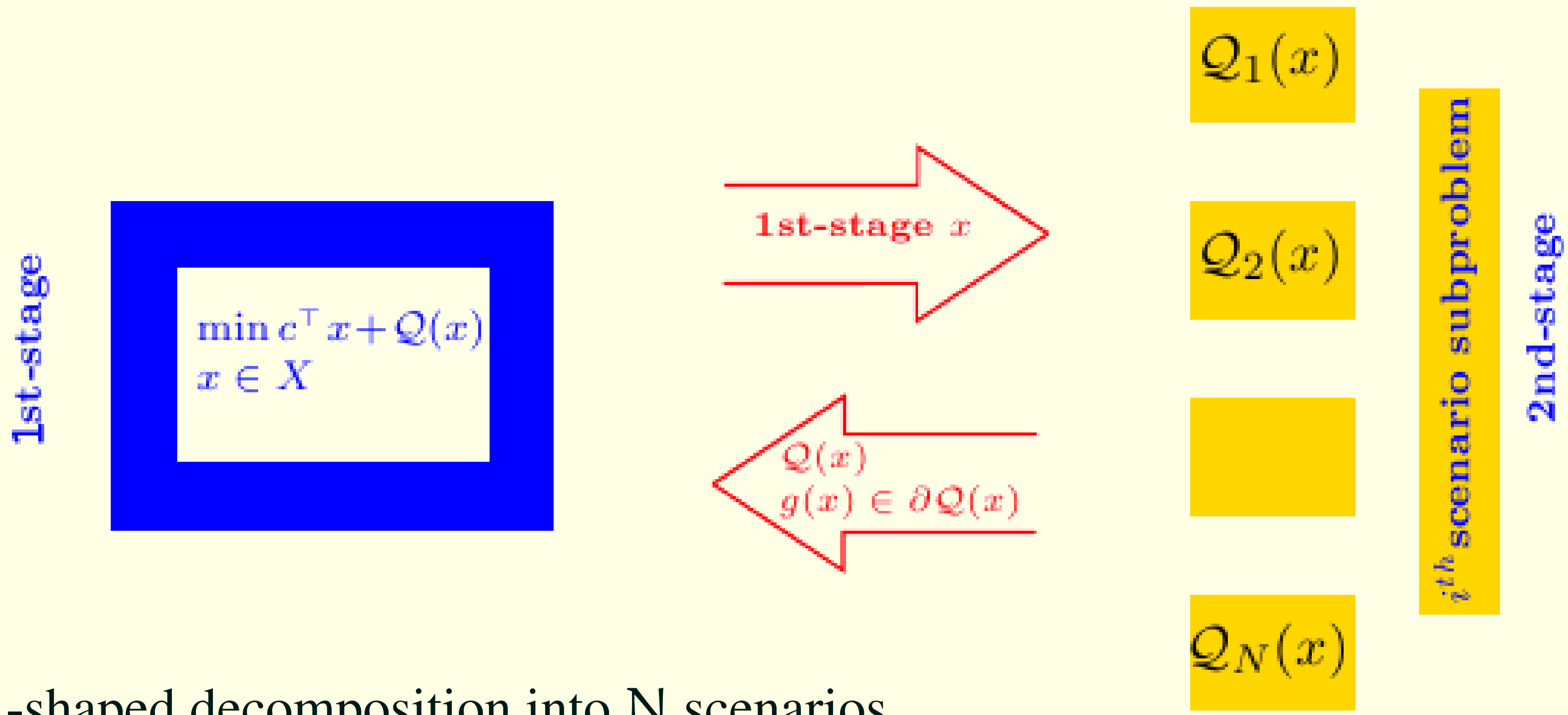


Skips Nuclear LPs (alternating) $\equiv$ noisy black box

25% less CPU time than exact bundle, same accuracy

Application in Energy II

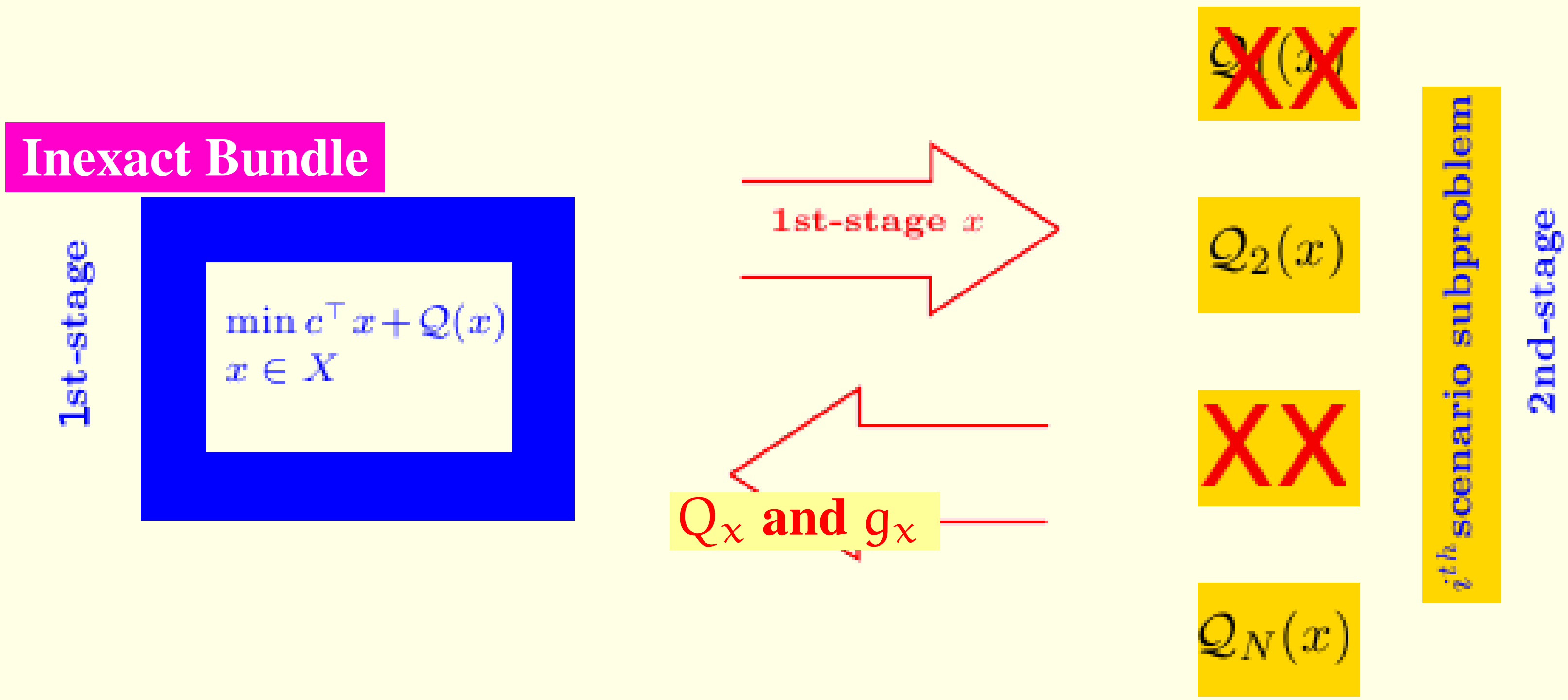
2-stage stochastic linear programs



L-shaped decomposition into N scenarios

Application in Energy II

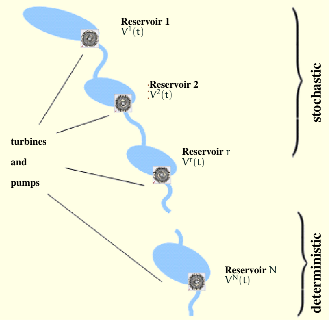
2-stage stochastic linear programs



Skips 80% LPs solution ≡ noisy black box

4 times faster than L-shaped, same accuracy

Applications in Energy III



Maximize revenue of hydro producers keeping reservoir levels between min-zones with 90% confidence (numerical integration in dimension 192!)

Comparison with previous values obtained by Wim van Ackooij, from R&D at EDF on several instances from Val d'Isère (Alpes), using a method by A. Prékopa.

Huge reduction in CPU times: **drops from almost 3h to 3 minutes**

Closing remarks

- Thanks to Welington de Oliveira and Marc Schmidt for some of the images.
- Credits to some co-authors: Welington de Oliveira, Claude Lemaréchal, Wim van Ackooij
- **Warning:** This tutorial does not intend to encourage drinking caipirinha.

Closing remarks

- Thanks to Welington de Oliveira and Marc Schmidt for some of the images.
- Credits to some co-authors: Welington de Oliveira, Claude Lemaréchal, Wim van Ackooij
- **Warning:** This tutorial does not intend to encourage drinking caipirinha.

  It is rather meant to facilitate the use of modern (on-demand accuracy) bundle methods.

Any doubts or questions, just e-mail me

To learn more

(exact) Bundle books

J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C. Sagastizábal, Numerical Optimization: Theoretical and Practical Aspects, Springer, 2nd ed., 2006.

J.B. Hiriart-Urruty and C. Lemaréchal, Convex Analysis and Minimization Algorithms II, no. 306 in Grund. der math. Wissenschaften, Springer, 2nd ed., 1996.

Inexact Bundle theory

M. Hintermüller, A proximal bundle method based on approximate subgradients, COAp, 20 (2001), pp. 245–266.

K.C. Kiwiel, A proximal bundle method with approximate subgradient linearizations, SiOpt, 16 (2006), pp. 1007–1023.

W. de Oliveira, C. Sagastizábal, and C. Lemaréchal, Convex proximal bundle methods in depth: a unified analysis for inexact oracles, MathProg, 148 (2014), pp. 241–277.

Inexact Bundle variants with applications

G. Emiel and C. Sagastizábal, Incremental-like bundle methods with application to energy planning, COAp, 46 (2010), pp. 305–332.

W. de Oliveira, C. Sagastizábal, and S. Scheimberg, Inexact bundle methods for two-stage stochastic programming, SiOpt, 21 (2011), pp. 517–544.

W. van Ackooij and C. Sagastizábal, Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems, SiOpt, 24 (2014), pp. 733–765.

W. de Oliveira and C. Sagastizábal, Level bundle methods for oracles with on-demand accuracy, OMS 29 (2014), pp. 1180 –1209

W. de Oliveira and C. Sagastizábal, Bundle methods in the xxi century: A birds'-eye view, Pesquisa Operacional, 34 (2014), pp. 647 – 670.

W. de Oliveira and M. Solodov, A doubly stabilized bundle method for nonsmooth convex optimization, accepted in MathProg, 2015.

and my web-page: `http://www.impa.br/~sagastiz`

Save the date!