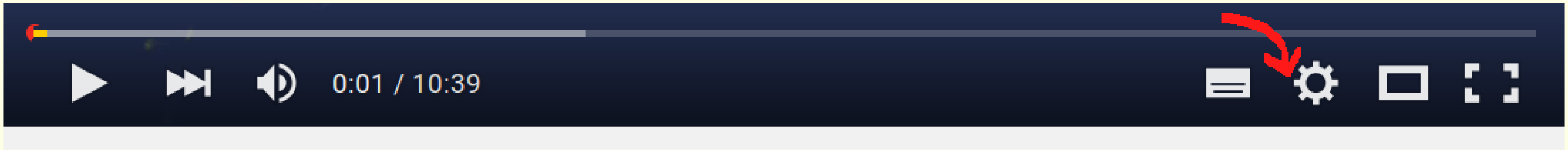
ALGORITHMS FOR TWO-STAGE SP: A PRIMER ON NONSMOOTH OPTIMIZATION

Claudia Sagastizábal

BAS Lecture 9, April 7, 2016, IMPA

SVAN 2016

Set YouTube resolution to 480p



for best viewing

Two-Stage LP with RCR[a], $\Omega = \{\omega^1, \dots, \omega^S\}$

$$\min_{\mathbf{x} \in \mathbf{X}} \mathbf{c}^\top \mathbf{x} + \phi(\mathbf{x}) \qquad \text{for} \qquad \mathbf{X} := \{\mathbf{x} \geq \mathbf{0} : \mathbf{A}\mathbf{x} = \mathbf{b}\},$$

$$\text{where } \phi(x) = \mathbb{E}\left[Q(x, \xi)\right] = \sum_{s=1}^{S} p_s Q(x, \xi^s) \qquad \text{and}$$

$$Q(x, \xi^s) = \begin{cases} \min & q^{s\top} y \\ \text{s.t.} & W^s y = h^s - T^s x \\ & y \geq 0 \end{cases} = \begin{cases} \max & \pi^\top (h^s - T^s x) \\ \text{s.t.} & W^{s\top} \pi \leq q^s \end{cases}$$

$$\partial\phi(x^{\mathbf{k}}) = -\sum_{s=1}^{S} p_s T^{s\top} \arg\max\left\{\pi^\top (h^s - T^s x^{\mathbf{k}}) : \pi \in \Pi(q^s)\right\}$$

[a] next lecture: without Relative Complete Recourse (infeasibility!)

Two-Stage LP with RCR[a], $\Omega = \{\omega^1, \dots, \omega^S\}$

$$\min_{\mathbf{x} \in \mathbf{X}} \mathbf{c}^\top \mathbf{x} + \phi(\mathbf{x}) \qquad \text{for} \qquad \mathbf{X} := \{\mathbf{x} \geq \mathbf{0} : \mathbf{A}\mathbf{x} = \mathbf{b}\},$$

where $\phi(x^{\mathbf{k}}) = \mathbb{E}\left[Q(x^{\mathbf{k}}, \xi)\right] = \sum_{s=1}^{S} p_s Q(x^{\mathbf{k}}, \xi^s)$ and

$$Q(x^{\mathbf{k}}, \xi^s) = \begin{cases} \min & {q^s}^\top y \\ \text{s.t.} & W^s y = h^s - T^s x^{\mathbf{k}} \\ & y \geq 0 \end{cases} = \begin{cases} \max & \pi^\top (h^s - T^s x^{\mathbf{k}}) \\ \text{s.t.} & {W^s}^\top \pi \leq q^s \end{cases}$$

$$\partial\phi(x^{\mathbf{k}}) = -\sum_{s=1}^{S} p_s {T^s}^\top \arg\max \left\{ \pi^\top (h^s - T^s x^{\mathbf{k}}) : \pi \in \Pi(q^s) \right\}$$

---

[a] next lecture: without Relative Complete Recourse (infeasibility!)

Two-Stage LP with RCR, $\Omega = \{\omega^1, \ldots, \omega^S\}$

$$\min_{\mathbf{x}\in\mathbf{X}} \mathbf{c}^\top \mathbf{x} + \phi(\mathbf{x}) \qquad \text{for} \qquad \mathbf{X} := \{\mathbf{x} \geq \mathbf{0} : \mathbf{A}\mathbf{x} = \mathbf{b}\},$$

where $\phi(x^{\mathbf{k}}) = \mathbb{E}\left[Q(x^{\mathbf{k}}, \xi)\right] = \sum_{s=1}^{S} p_s Q(x^{\mathbf{k}}, \xi^s)$ and

$$Q(x^{\mathbf{k}}, \xi^s) = \left\{ \begin{array}{ll} \min & q^{s\,\top} y \\ \text{s.t.} & W^s y = h^s - T^s x^{\mathbf{k}} \\ & y \geq 0 \end{array} \right. = \left\{ \begin{array}{ll} \max & \pi^\top (h^s - T^s x^{\mathbf{k}}) \\ \text{s.t.} & W^{s\,\top} \pi \leq q^s \end{array} \right.$$

Primal solution $y^{s,k}$ Dual solution $\pi^{s,k}$

$$\partial\phi(x^{\mathbf{k}}) = -\sum_{s=1}^{S} p_s T^{s\,\top} \arg\max \left\{ \pi^\top (h^s - T^s x^{\mathbf{k}}) : \pi \in \Pi(q^s) \right\}$$

Evaluating $\phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k\,\top}(h^s - T^s x^k)$

Evaluating $\phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k\,\top}(h^s - T^s x^k)$

gives **for free a subgradient** $\gamma^k = -\sum_{s=1}^{S} p_s T^{s\,\top} \pi^{s,k} \in \partial\phi(x^k)$

Evaluating $\phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k\top}(h^s - T^s x^k)$

gives for free a subgradient $\gamma^k = -\sum_{s=1}^{S} p_s T^{s\top} \pi^{s,k} \in \partial\phi(x^k)$ and

the linearization

$$
\begin{aligned}
&\geq \quad \phi(x^k) + \gamma^{k\top}(x - x^k) \\
&= \quad \sum_{s=1}^{S} p_s \pi^{s,k\top}(h^s - T^s x^k) - \sum_{s=1}^{S} p_s \pi^{s,k\top} T^s (x - x^k) \\
&= \quad \sum_{s=1}^{S} p_s \pi^{s,k\top}(h^s - T^s x)
\end{aligned}
$$

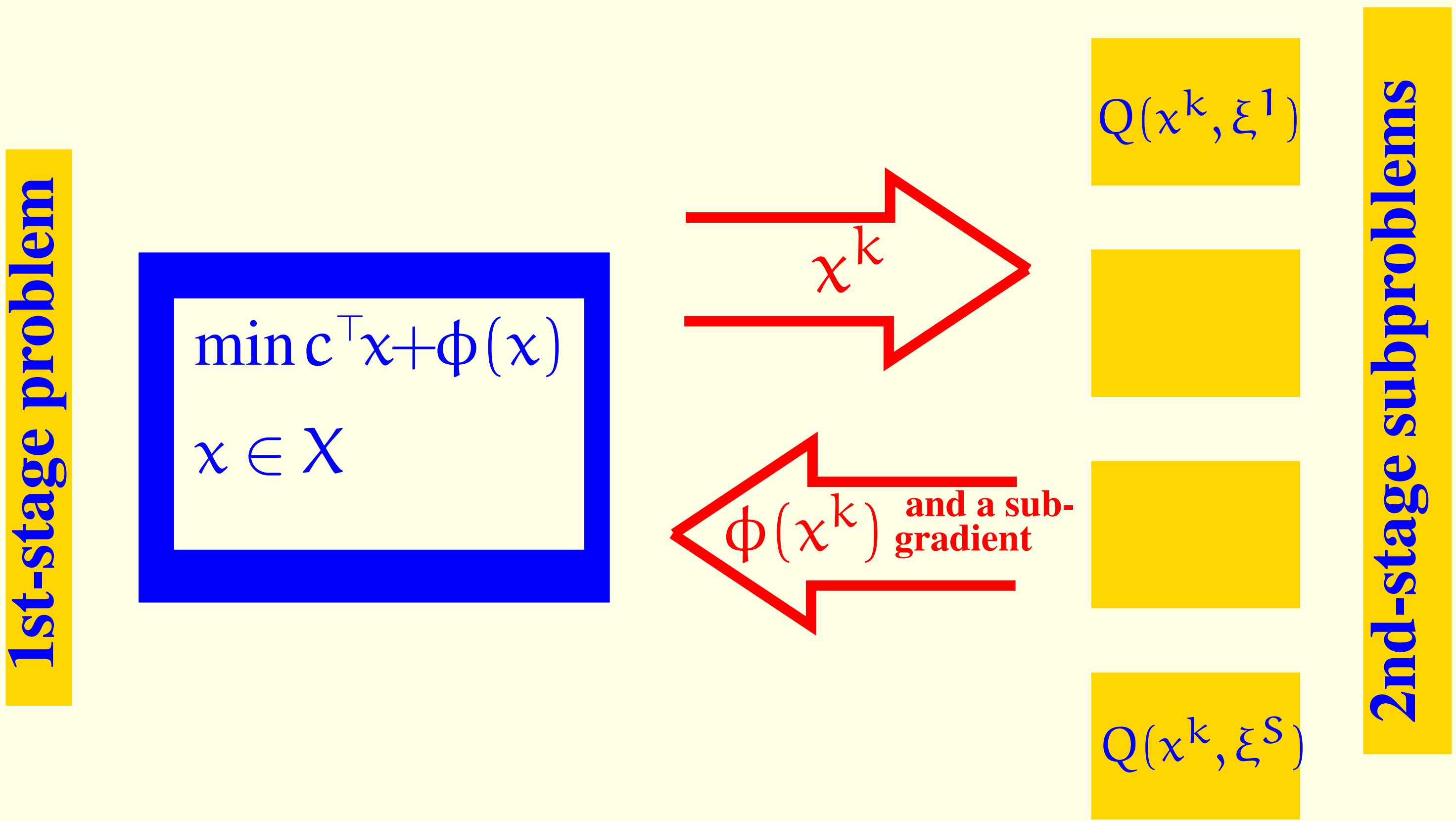
Evaluating $\phi(x^k) = \sum_{s=1}^{S} p_s \pi^{s,k\,\top}(h^s - T^s x^k)$

gives for free a subgradient $\gamma^k = -\sum_{s=1}^{S} p_s T^{s\,\top} \pi^{s,k} \in \partial\phi(x^k)$ and

the linearization

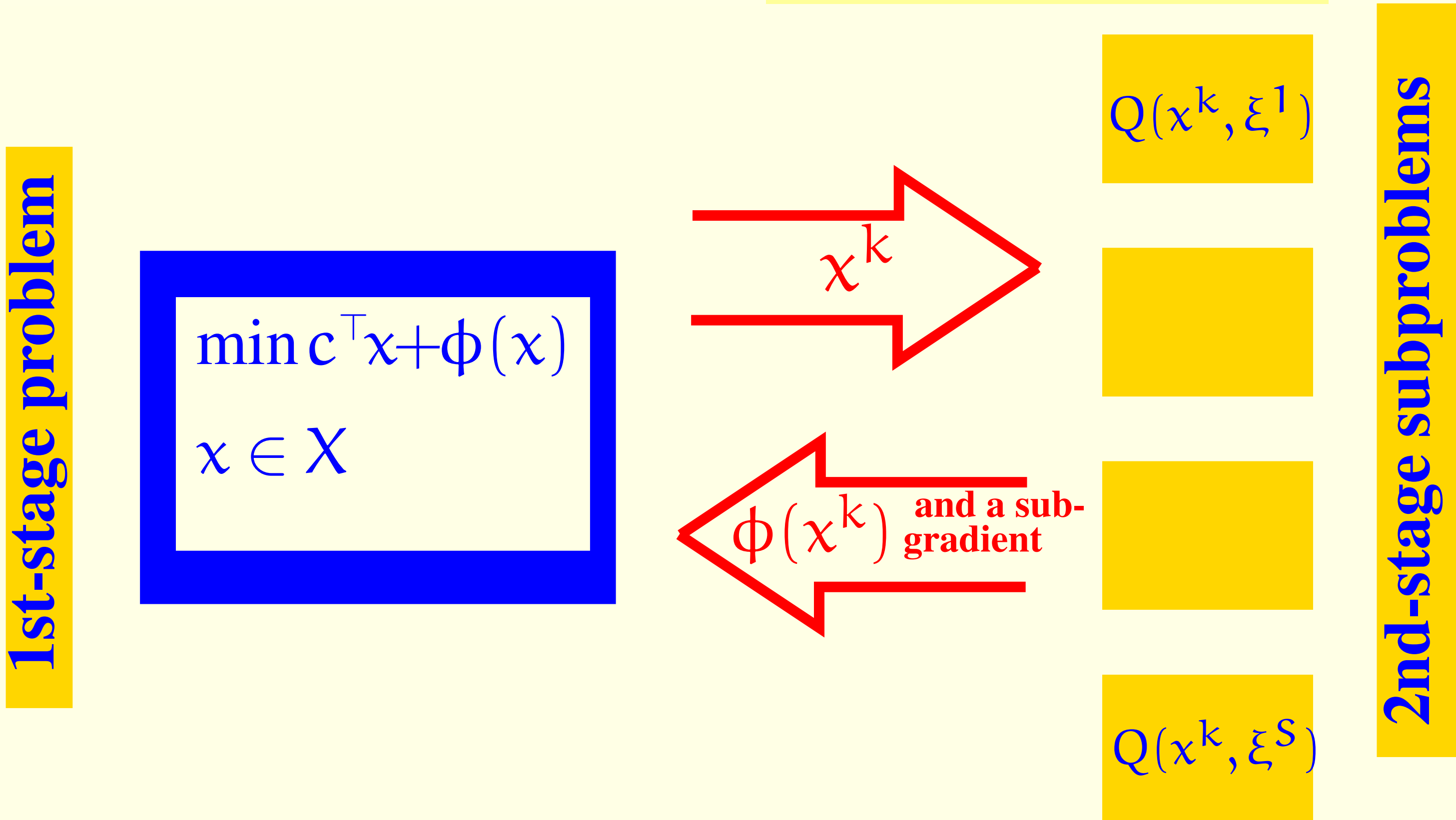
$$\begin{aligned}
\phi(x) &\geq \phi(x^k) + \gamma^{k\,\top}(x - x^k) \\
&= \sum_{s=1}^{S} p_s \pi^{s,k\,\top}(h^s - T^s x^k) - \sum_{s=1}^{S} p_s \pi^{s,k\,\top} T^s (x - x^k) \\
&= \sum_{s=1}^{S} p_s \pi^{s,k\,\top}(h^s - T^s x)
\end{aligned}$$

Evaluating ϕ at x^k



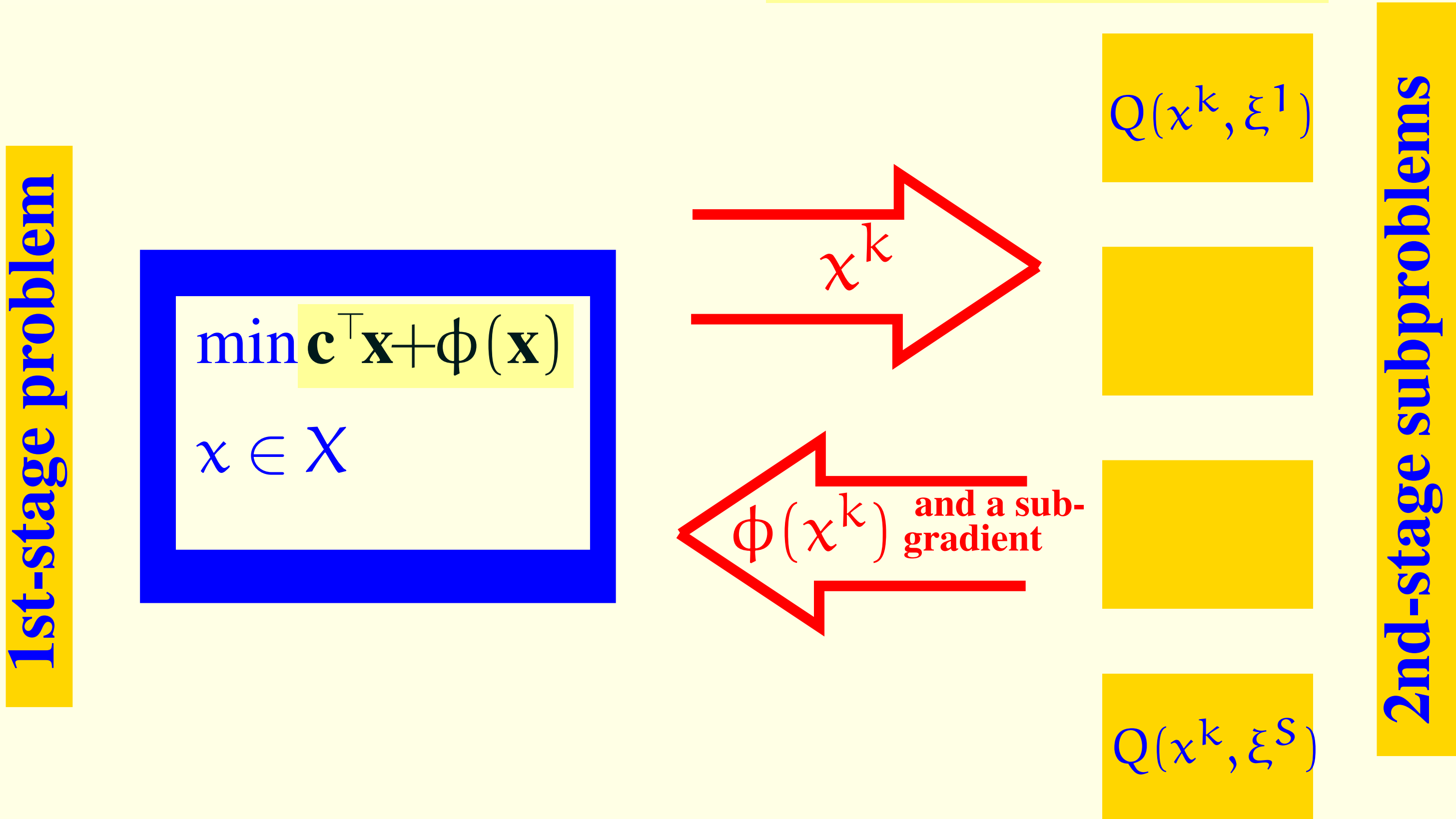
Evaluating ϕ at x^k

gives **the linearization** $\phi(x) \geq \sum_{s=1}^{S} p_s \pi^{s,k\top}(h^s - T^s x)$



Evaluating ϕ at x^k

gives **the linearization** $\phi(x) \geq \sum_{s=1}^{S} p_s \pi^{s,k\top}(h^s - T^s x)$



$\min_{x \in X} f(x)$ convex nonsmooth knowing f(x) and $g(x) \in \partial f(x)$ (one)

Computational NSO: what does it mean?

For the unconstrained[a] problem

$$\boxed{\min f(x)},$$

where f is convex but not differentiable at some points

[a] $X = \mathbb{R}^n$ today

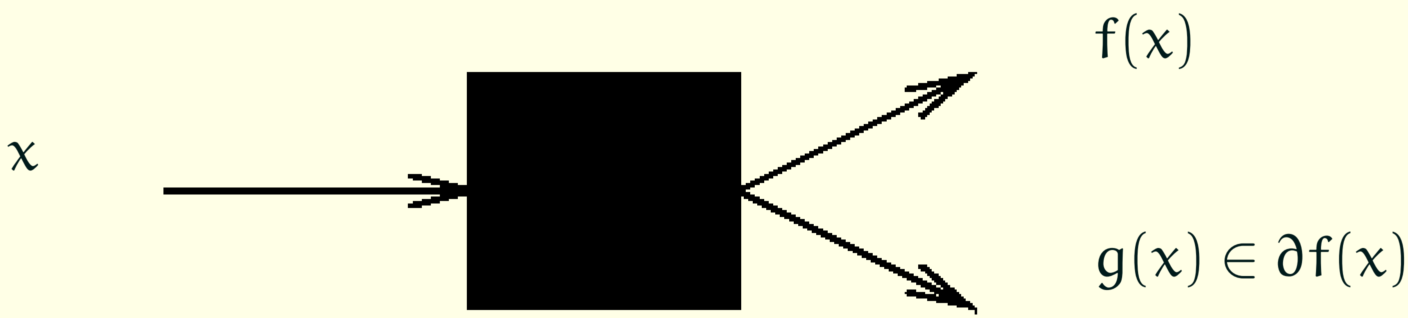
Computational NSO: what does it mean?

For the unconstrained problem

$$\boxed{\min f(x)},$$

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/

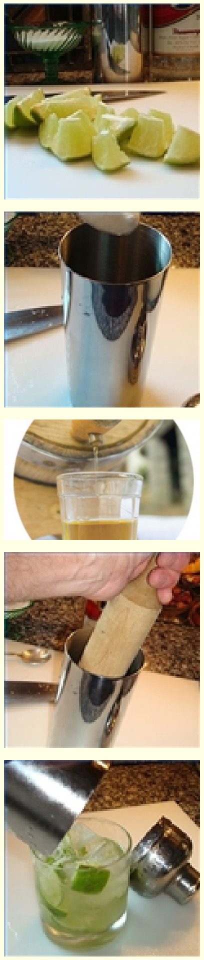
What do we mean by an algorithm?

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?

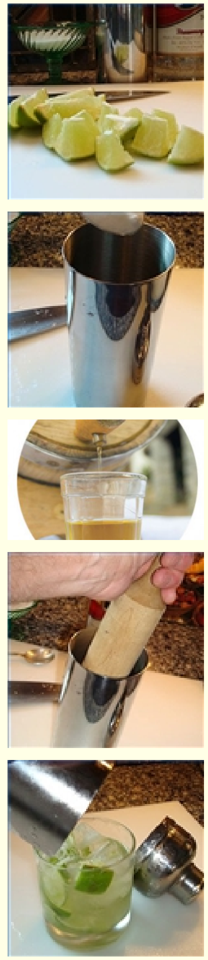
An algorithm

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

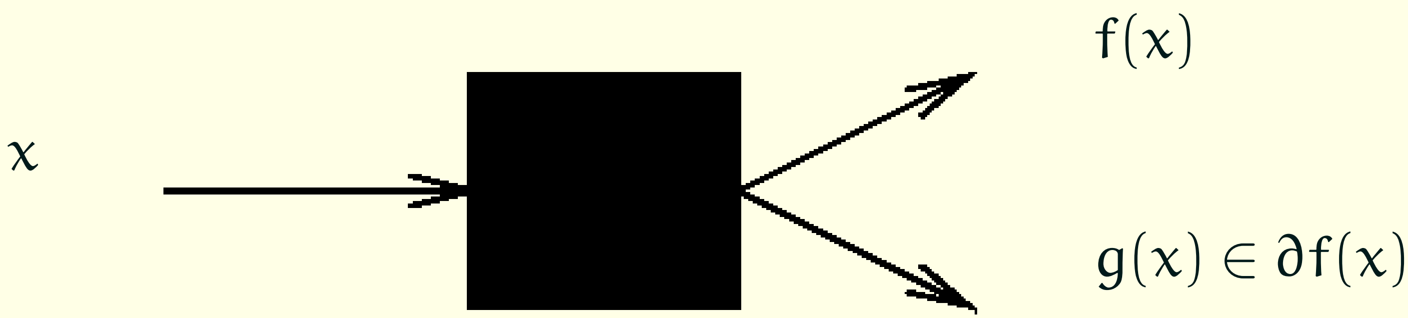
Back to Computational NSO

For the unconstrained problem

$$\min f(x),$$

where f is convex but not differentiable at some points,

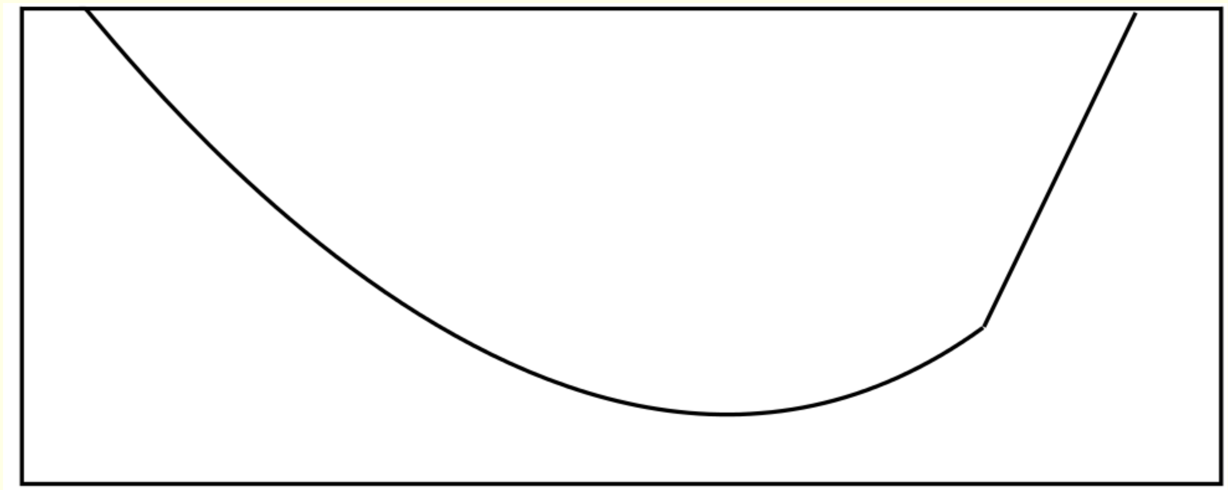
we look for algorithms based on information provided by an oracle or “black box”



endowed with reliable **stopping tests**

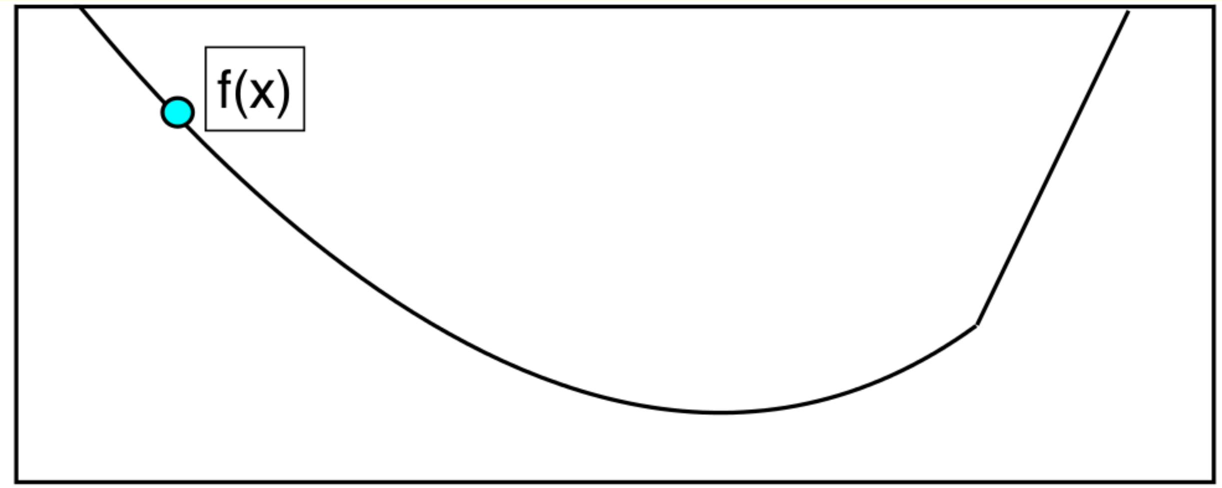
What can be done with the oracle information?

An example of a convex nonsmooth function



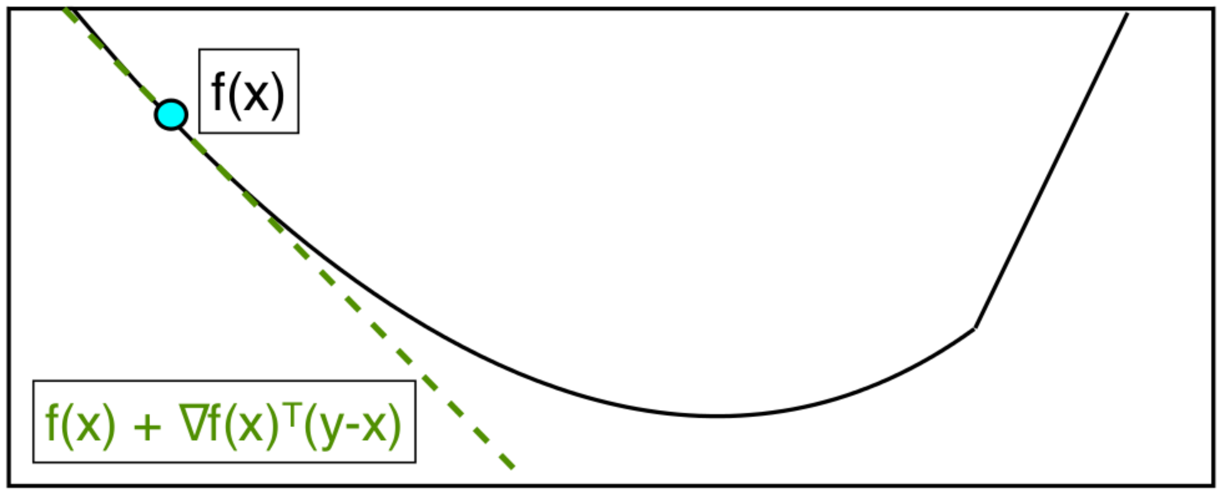
What can be done with the oracle information?

An example of a convex nonsmooth function



What can be done with the oracle information?

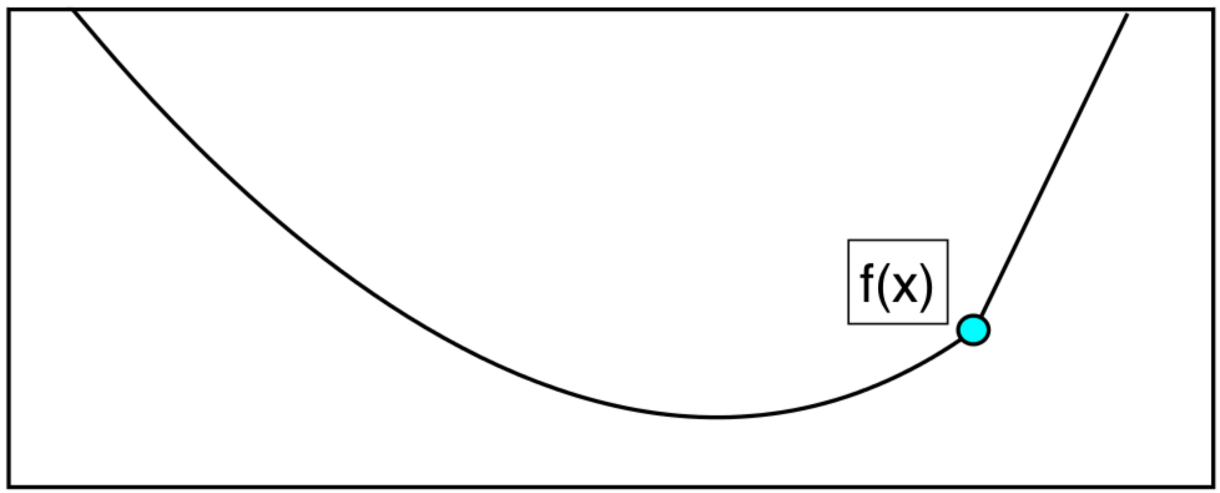
An example of a convex nonsmooth function



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

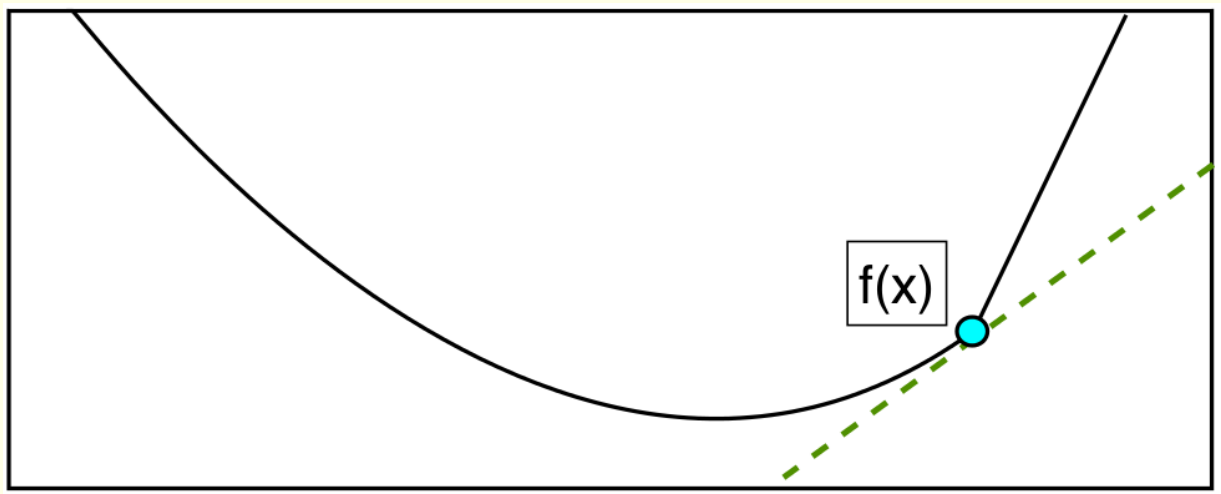
What can be done with the oracle information?

An example of a convex nonsmooth function



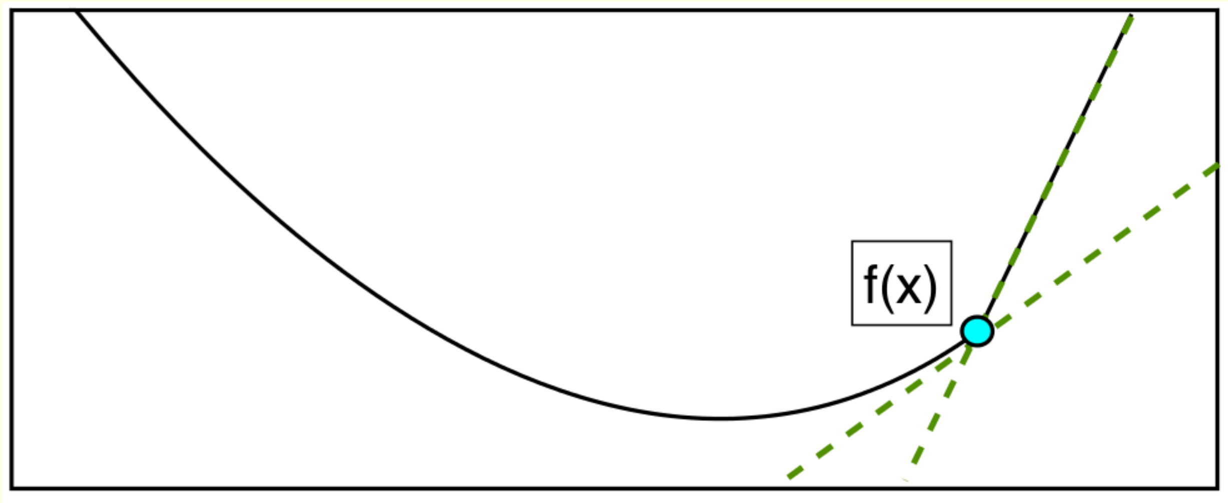
What can be done with the oracle information?

An example of a convex nonsmooth function



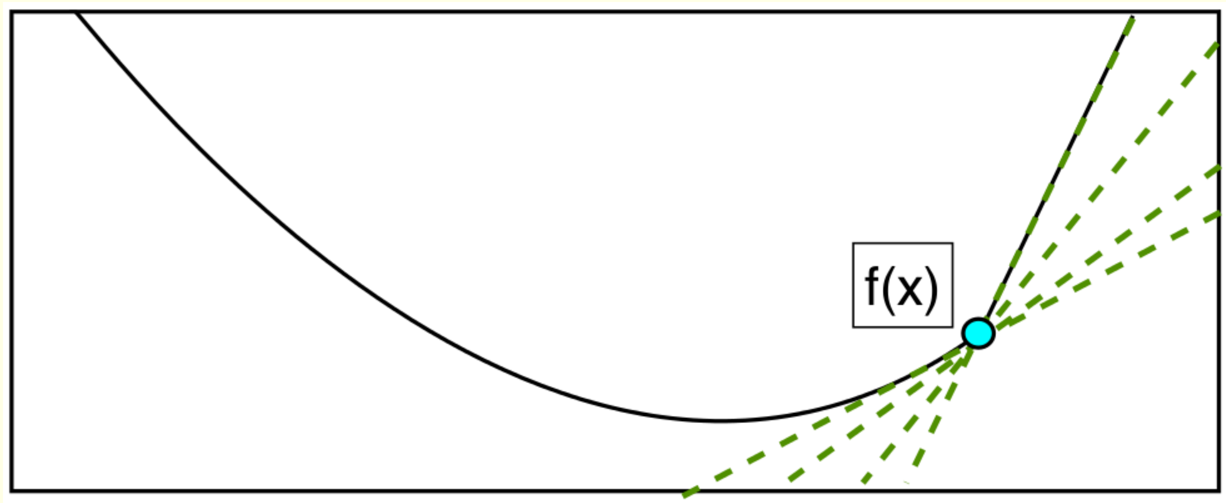
What can be done with the oracle information?

An example of a convex nonsmooth function



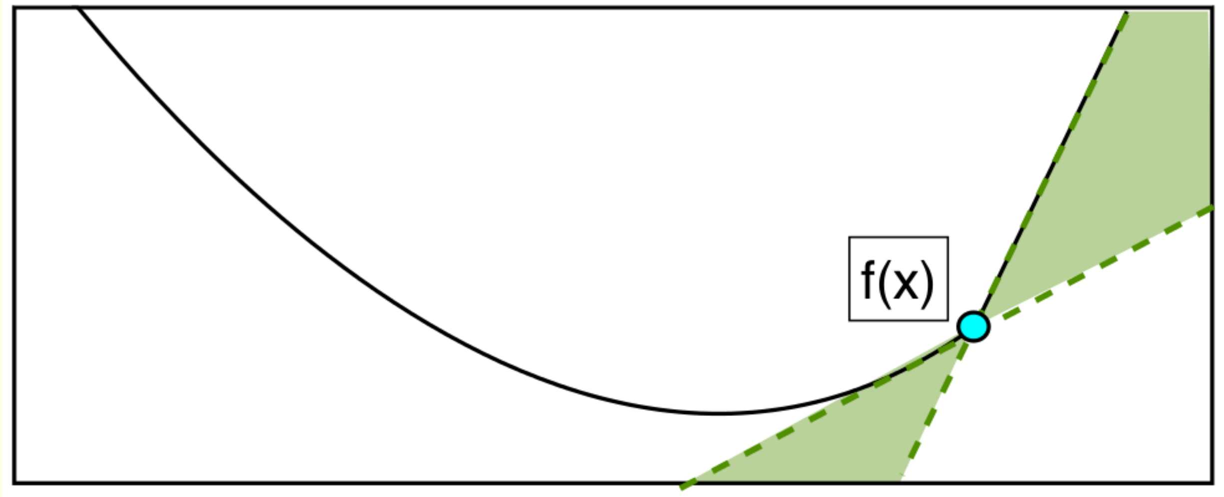
What can be done with the oracle information?

An example of a convex nonsmooth function



What can be done with the oracle information?

An example of a convex nonsmooth function

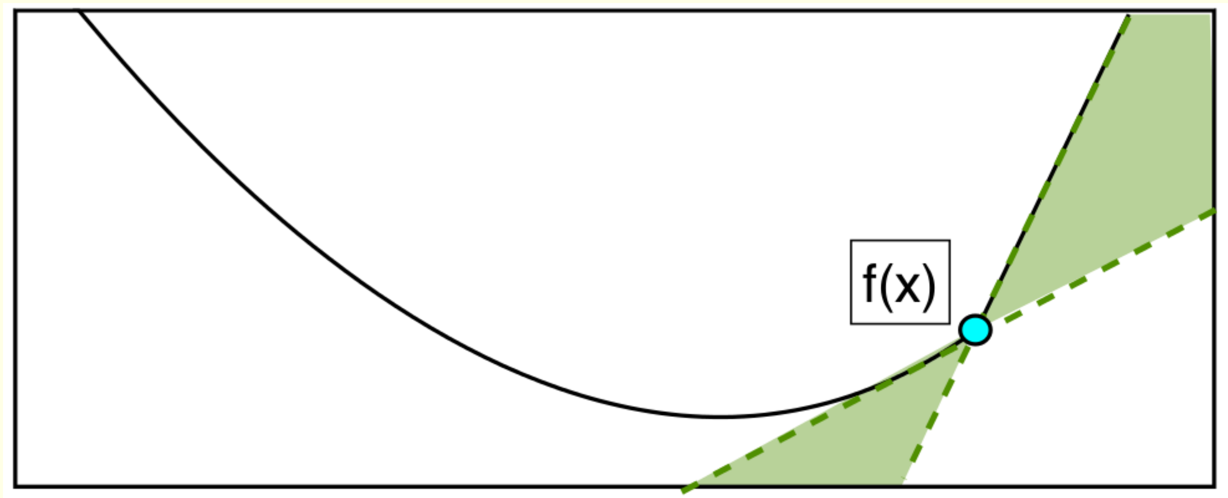


$$\partial f(x) \quad = \quad \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y\}$$

f $\qquad$ x

What can be done with the oracle information?

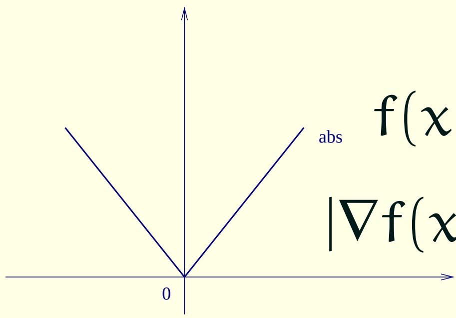
An example of a convex nonsmooth function



$$\begin{aligned}\partial f(x) &= \{g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y\} \\ &= \{\text{slopes of linearizations supporting } f, \text{ tangent at } x\}\end{aligned}$$

Why special NSO methods?

Smooth optimization methods **do not work**



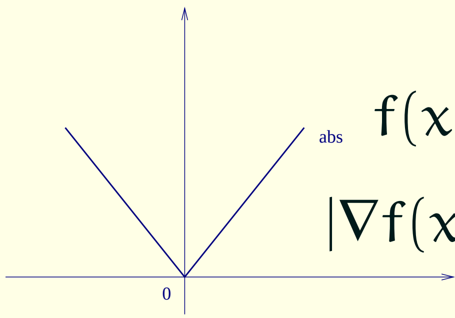
$$f(x) \quad = |x|$$

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

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

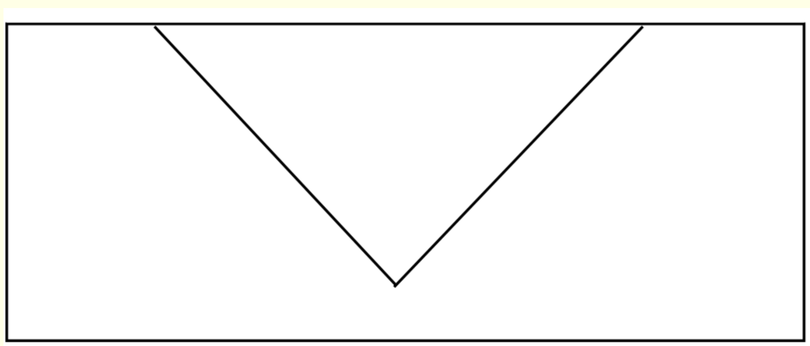
Why special NSO methods?

Smooth optimization methods **do not work**



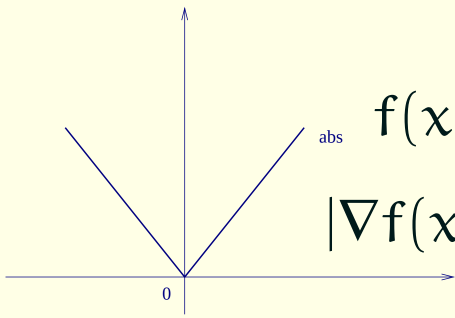
$$
\begin{aligned}
f(x) &= |x| \\
|\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{aligned}
$$

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



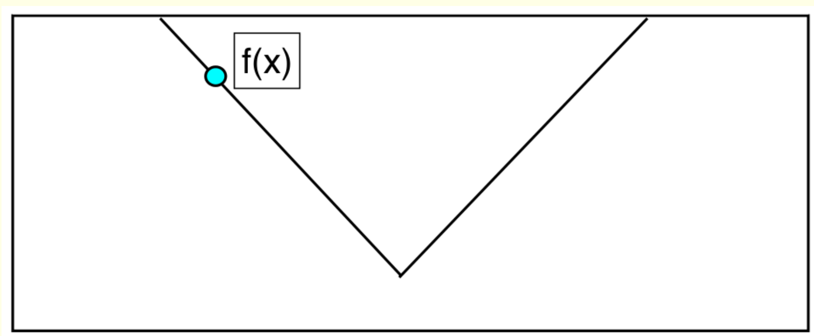
Why special NSO methods?

Smooth optimization methods **do not work**



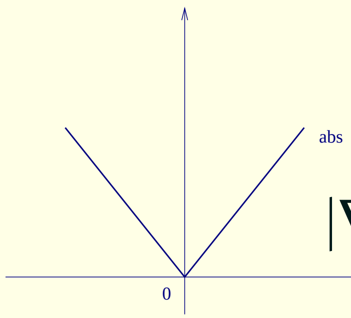
$$
\begin{aligned}
f(x) &= |x| \\
|\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{aligned}
$$

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



Why special NSO methods?

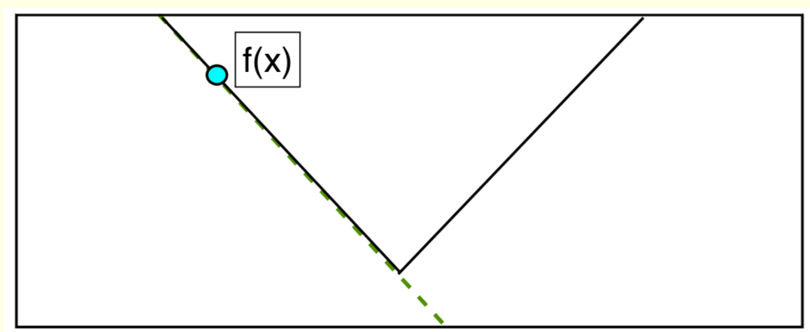
Smooth optimization methods **do not work**



$$f(x) \quad = |x|$$

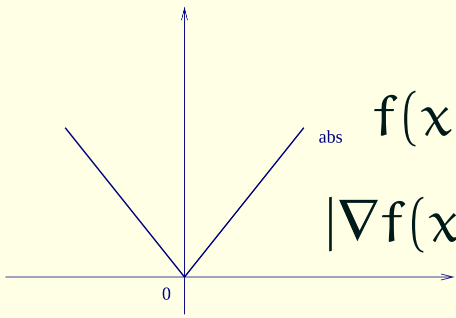
$$|\nabla f(x^k)| \quad = 1, \ \forall x \neq 0 \qquad \partial f(0) = [-1, 1]$$

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



Why special NSO methods?

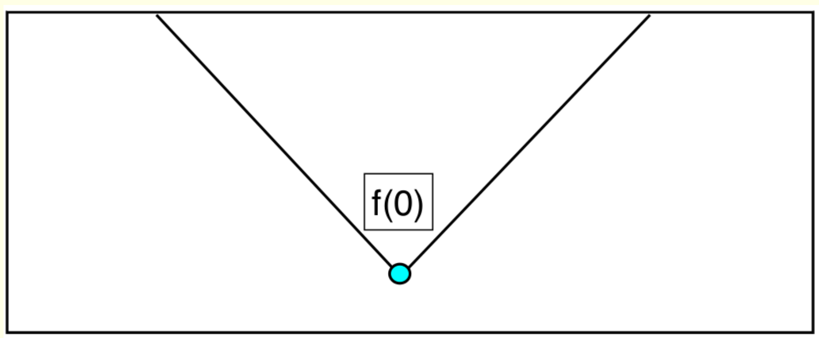
Smooth optimization methods **do not work**



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

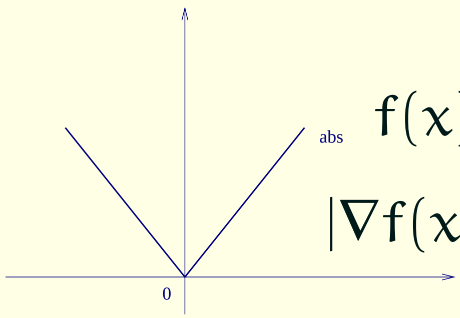
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \qquad \partial f(0) = [-1, 1]$$

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



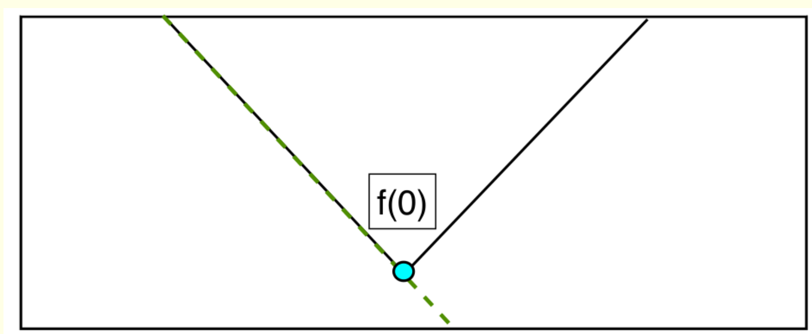
Why special NSO methods?

Smooth optimization methods **do not work**



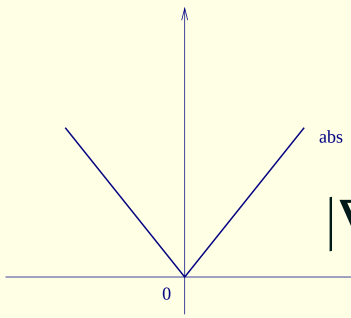
$$
\begin{aligned}
f(x) &= |x| \\
|\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{aligned}
$$

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



Why special NSO methods?

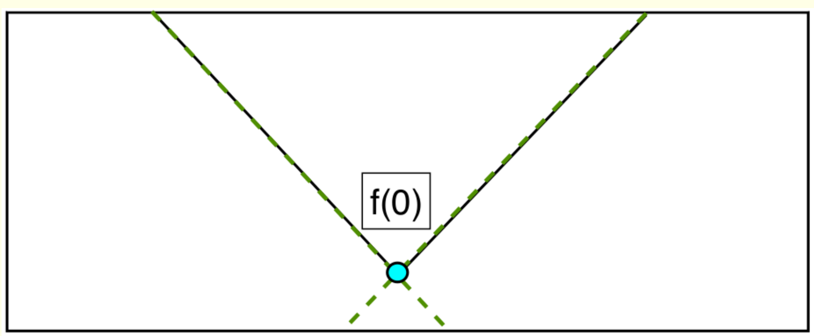
Smooth optimization methods **do not work**



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

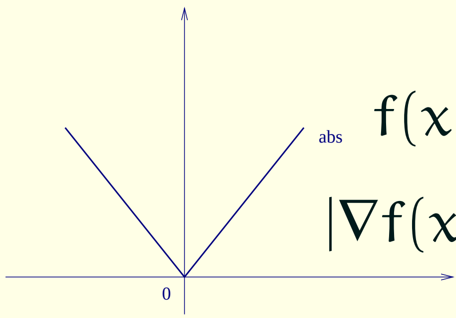
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

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



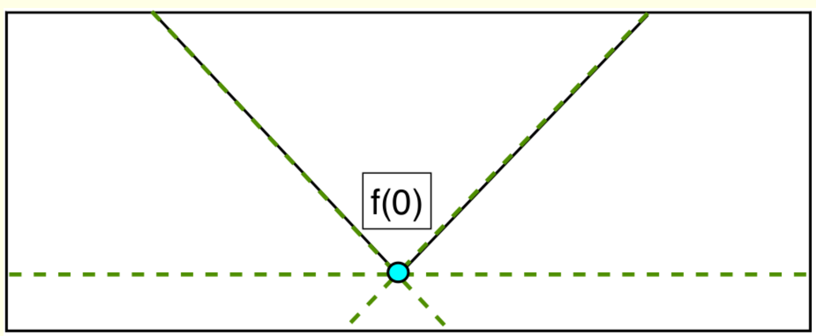
Why special NSO methods?

Smooth optimization methods **do not work**



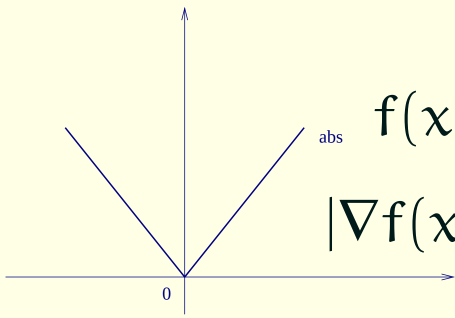
$$
\begin{aligned}
f(x) &= |x| \\
|\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{aligned}
$$

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



Why special NSO methods?

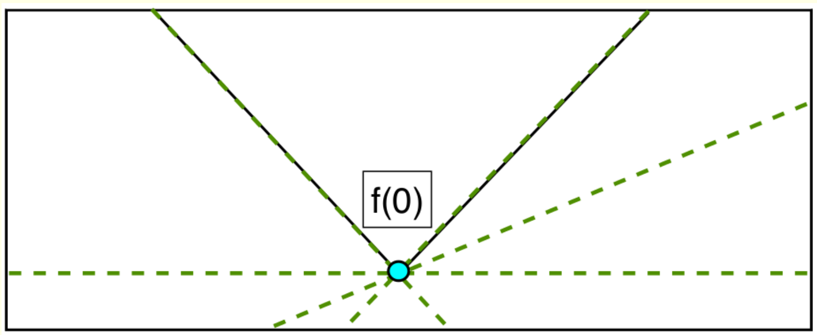
Smooth optimization methods **do not work**



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

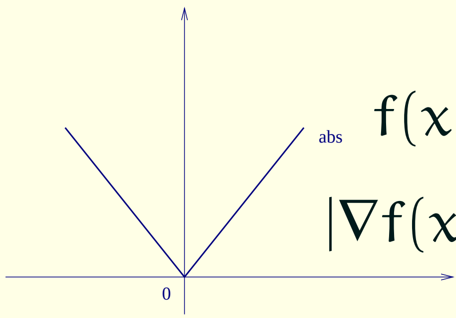
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

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



Why special NSO methods?

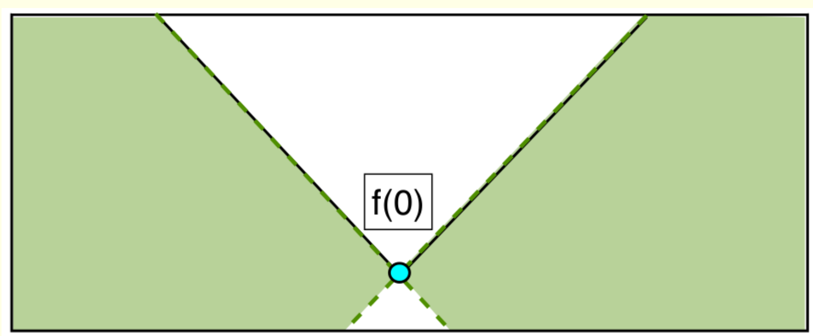
Smooth optimization methods **do not work**



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

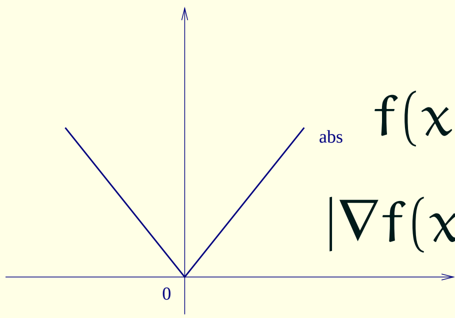
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

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**



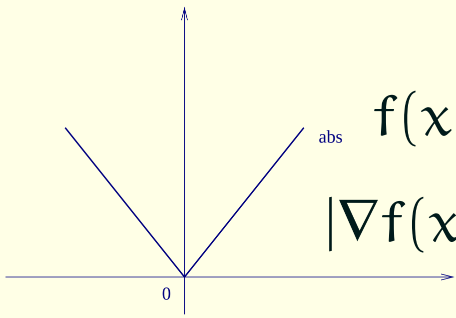
$$\begin{aligned} f(x) &= |x| \\ |\nabla f(x^k)| &= 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1] \end{aligned}$$

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

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

Why special NSO methods?

Smooth optimization methods **do not work**



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

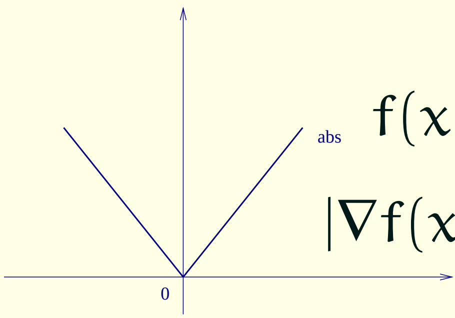
Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$

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$)$

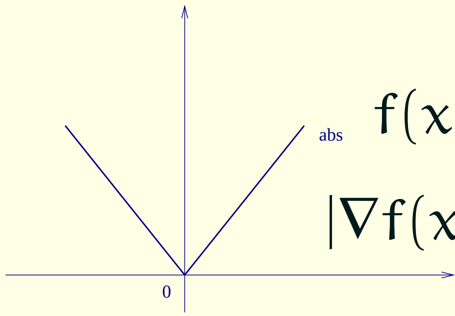
Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

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

Why special NSO methods?

Smooth optimization methods **do not work**



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

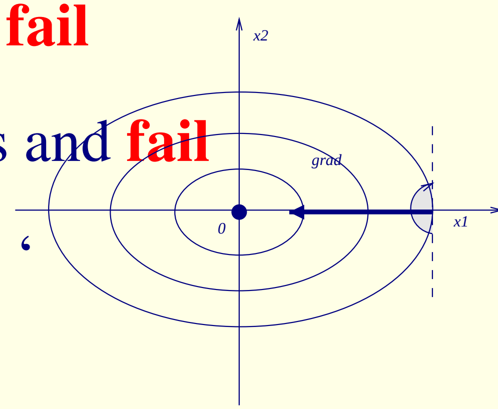
$$|\nabla f(x^k)| = 1,\ \forall x \neq 0 \quad \partial f(0) = [-1, 1]$$

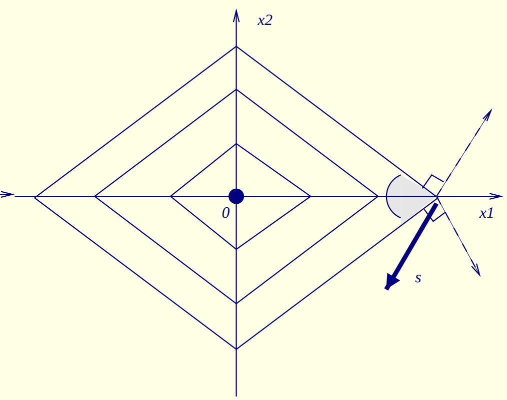
Smooth stopping test **fails**: $|\nabla f(x^k)| \leq$TOL $\qquad (\leftrightarrow |g(x^k)| \leq$TOL$)$

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**

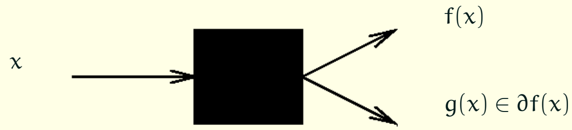
$-g(x^k)$ may **not** provide descent ‘





How is the oracle information used?

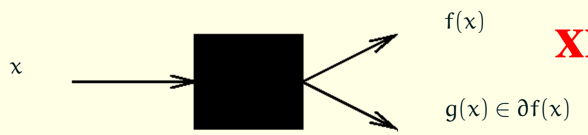
We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

How is the oracle information used?

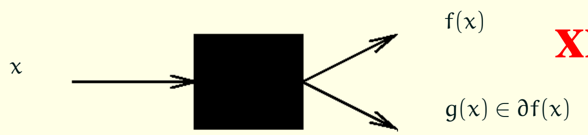
We look for algorithms based on information provided by an oracle



~~endowed with reliable stopping tests~~

How is the oracle information used?

We look for algorithms based on information provided by an oracle

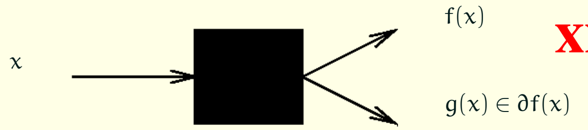


~~endowed with reliable stopping tests~~

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 $k = 1$.

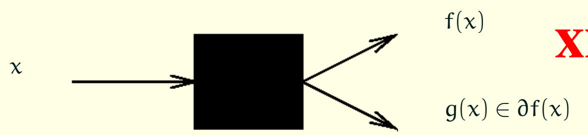
1 Call the oracle at x^k.

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

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

Is this a good "recipe"?


CORAL SEA COURT
LOWER ENTRANCE
USE TOP ENTRANCE

Subgradient Methods

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

1 Call the oracle at x^k.

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

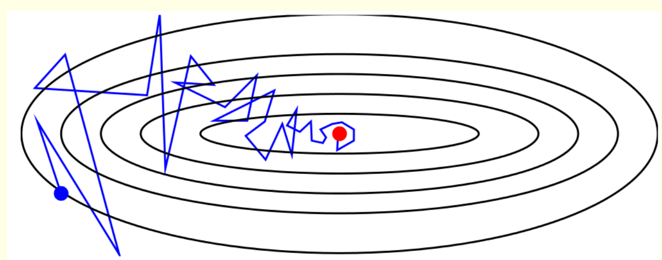
3 Make $k = k + 1$ and loop to 1.

SG methods are

the algorithmic version

of this road sign

not a good recipe



Non-monotone!

Subgradient Methods: why a "not-good" recipe

$$\begin{cases} \text{Non-monotone functional values, but converges} \\ \text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty \\ \end{cases}$$

Subgradient Methods: why a "not-good" recipe

$$\left\{\begin{array}{l}\text{Non-monotone functional values, but converges}\\ \text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty\\ \\ \end{array}\right.$$

Constrained case dealt with by projecting onto X: reasonable for simple X only

Subgradient Methods: why a “not-good” recipe

$$\begin{cases} \text{Non-monotone functional values, but converges} \\ \text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty \\ \text{Lacks a stopping test} \end{cases}$$

Subgradient Methods: why a “not-good” recipe

$$\begin{cases} \text{Non-monotone functional values, but converges} \\ \text{because distance to solution set decreases for } \sum t_k = +\infty, \sum t_k^2 < +\infty \\ \text{Lacks a stopping test} \end{cases}$$

…does not use all available information

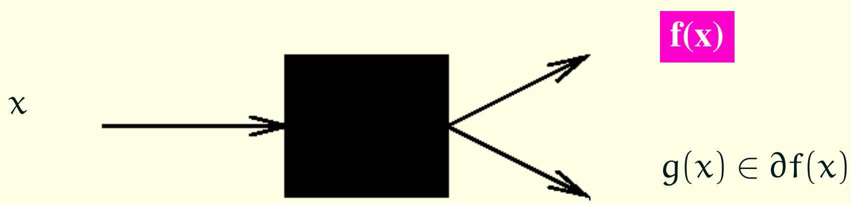
Subgradient Methods: why a "not-good" recipe

Non-monotone functional values, but converges

because distance to solution set decreases for $\sum t_k = +\infty, \sum t_k^2 < +\infty$

Lacks a stopping test

…does not use all **available** information



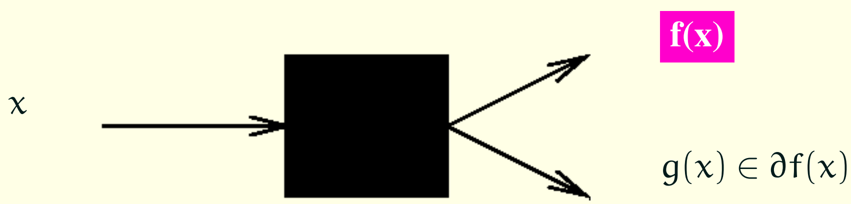
Subgradient Methods: why a "not-good" recipe

Non-monotone functional values, but converges

because distance to solution set decreases for $\sum t_k = +\infty, \sum t_k^2 < +\infty$

Lacks a stopping test

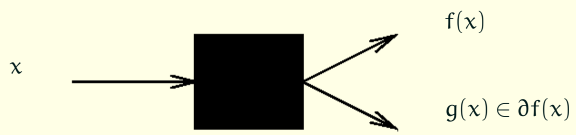
…does not use all **available** information



SG methods are like caipirinha without cachaça

How is the oracle information used?

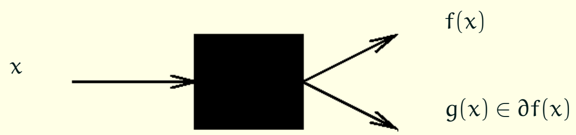
We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

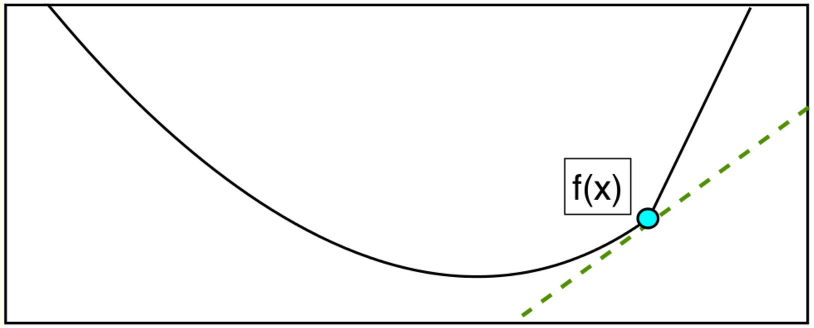
How is the oracle information used?

We look for algorithms based on information provided by an oracle



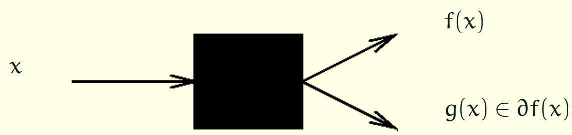
endowed with reliable stopping tests

Black box information defines linearizations



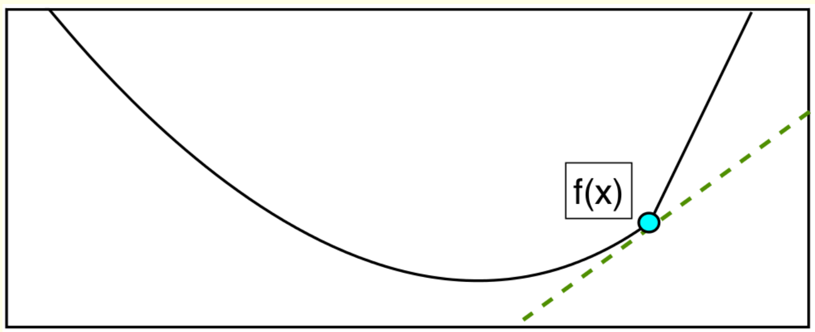
How is the oracle information used?

We look for algorithms based on information provided by an oracle



endowed with reliable stopping tests

Black box information defines linearizations

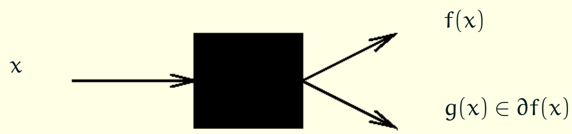


that put together create a **model M** of the function f.

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

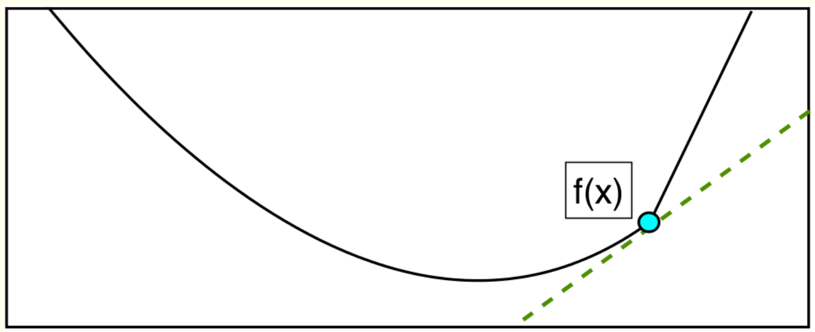
How is the oracle information used?

We look for algorithms based on information provided by an oracle

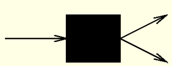


endowed with reliable stopping tests

Black box information defines linearizations

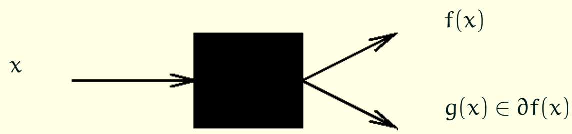


that put together create a **model M** of the function f.

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

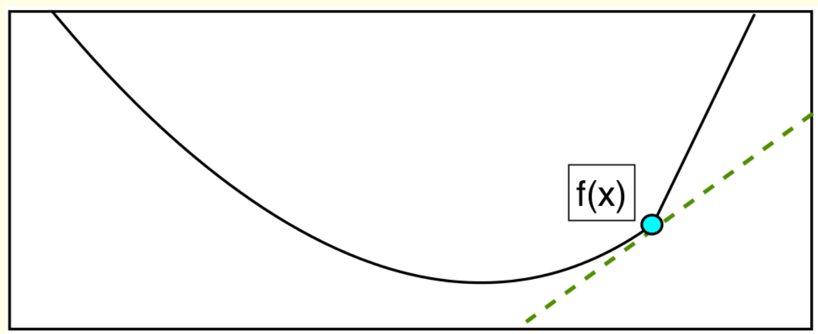
How is the oracle information used?

We look for algorithms based on information provided by an oracle



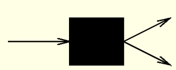
endowed with reliable stopping tests

Black box information defines linearizations



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

x^i

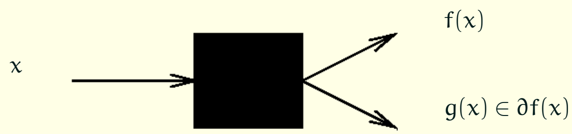


$$f^i = f(x^i)$$
$$g^i = g(x^i)$$

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

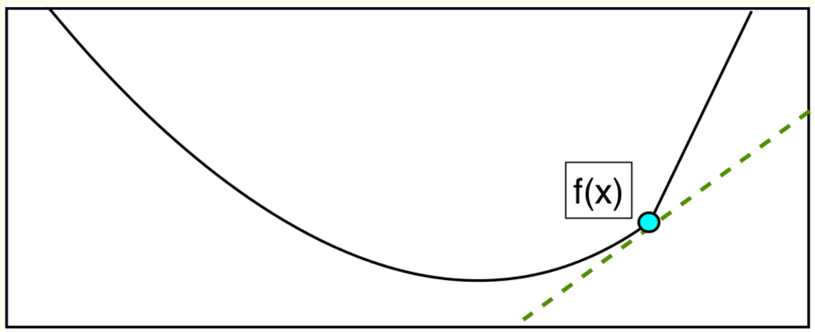
How is the oracle information used?

We look for algorithms based on information provided by an oracle



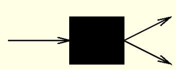
endowed with reliable stopping tests

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:

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

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

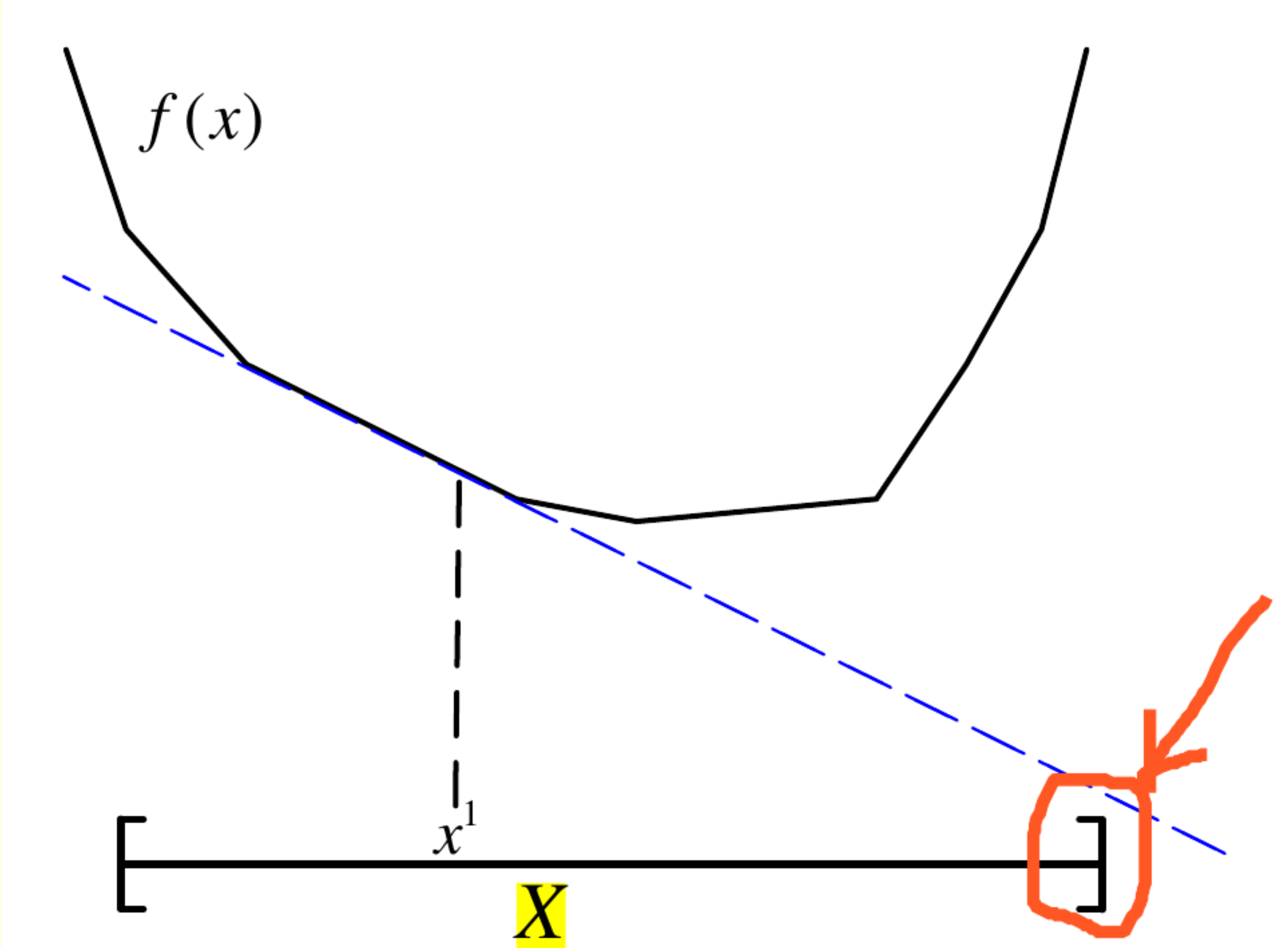
Improve the model at each iteration:

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

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

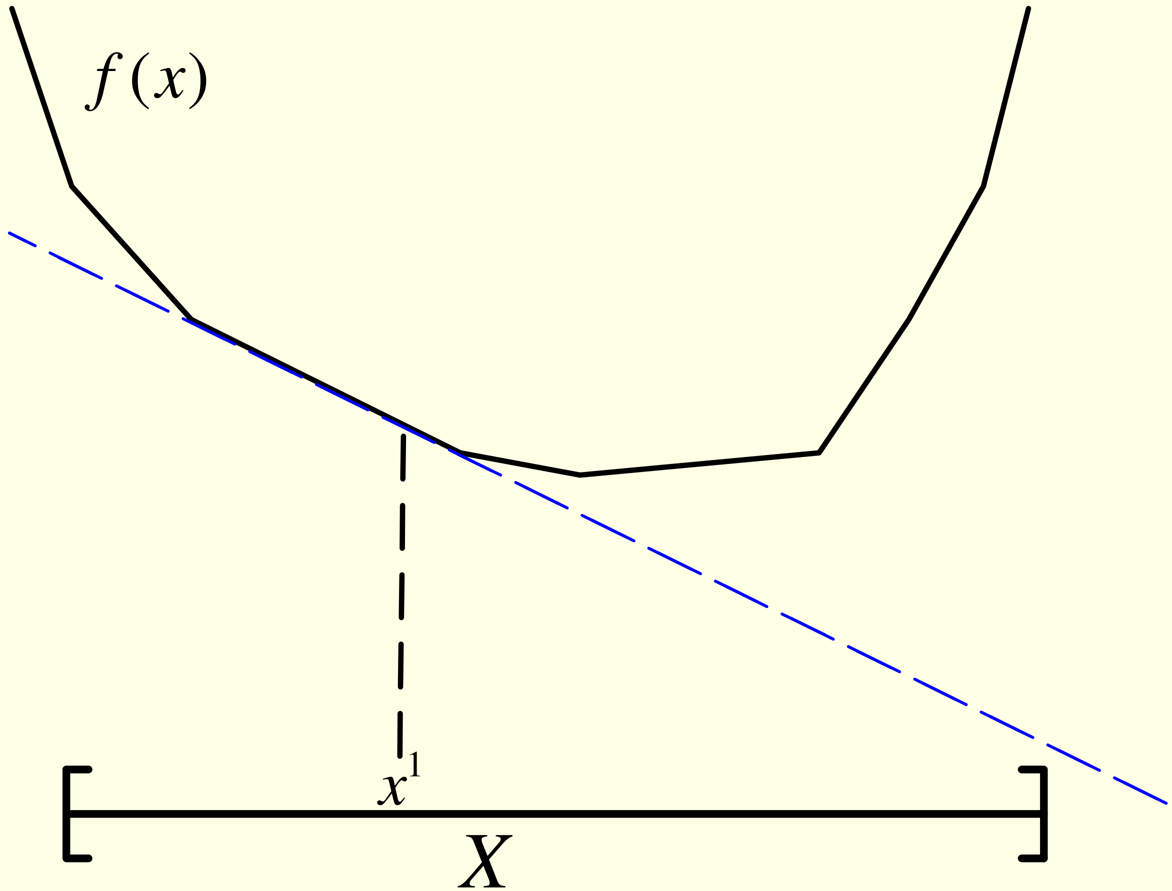
$x^{k+1} \in \arg\min \mathbf{M}_k(x)$ **iteratively**

Cutting-plane methods

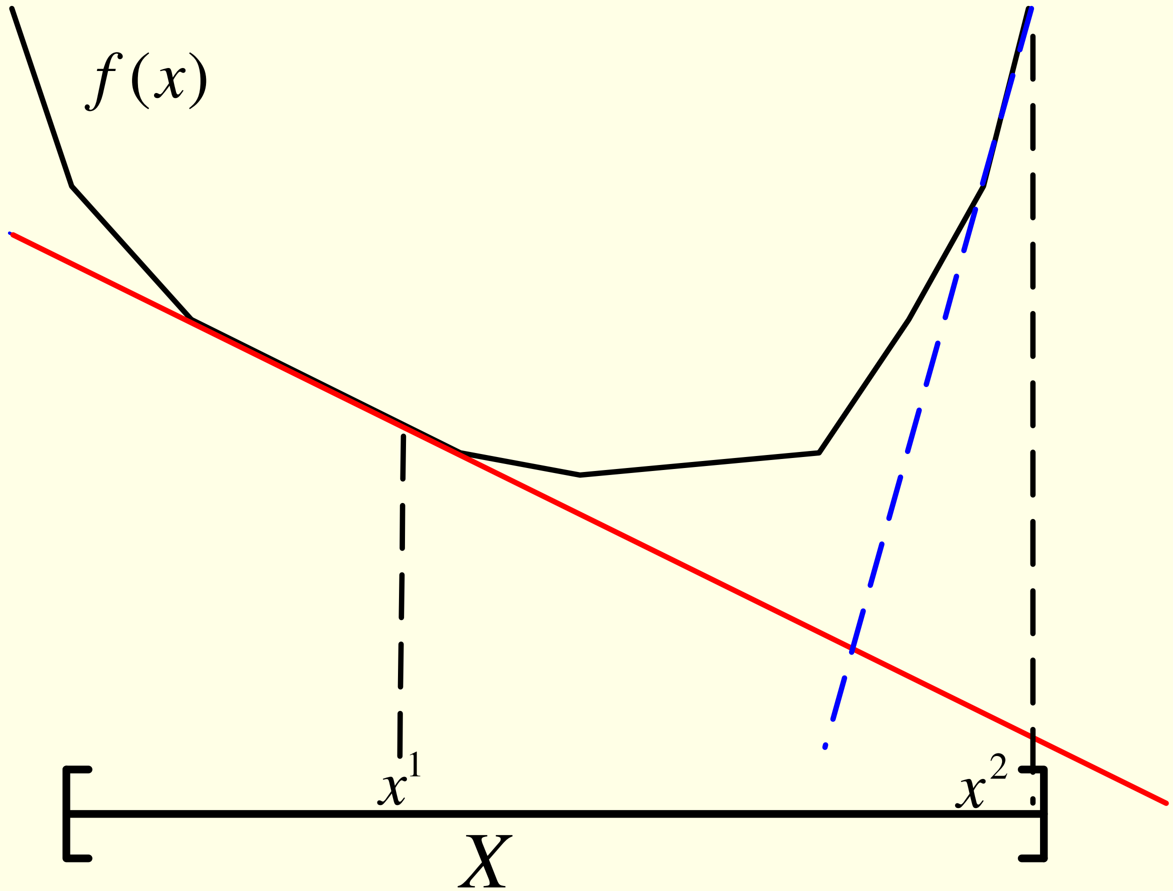


Artificial bounding at least for the first iterations

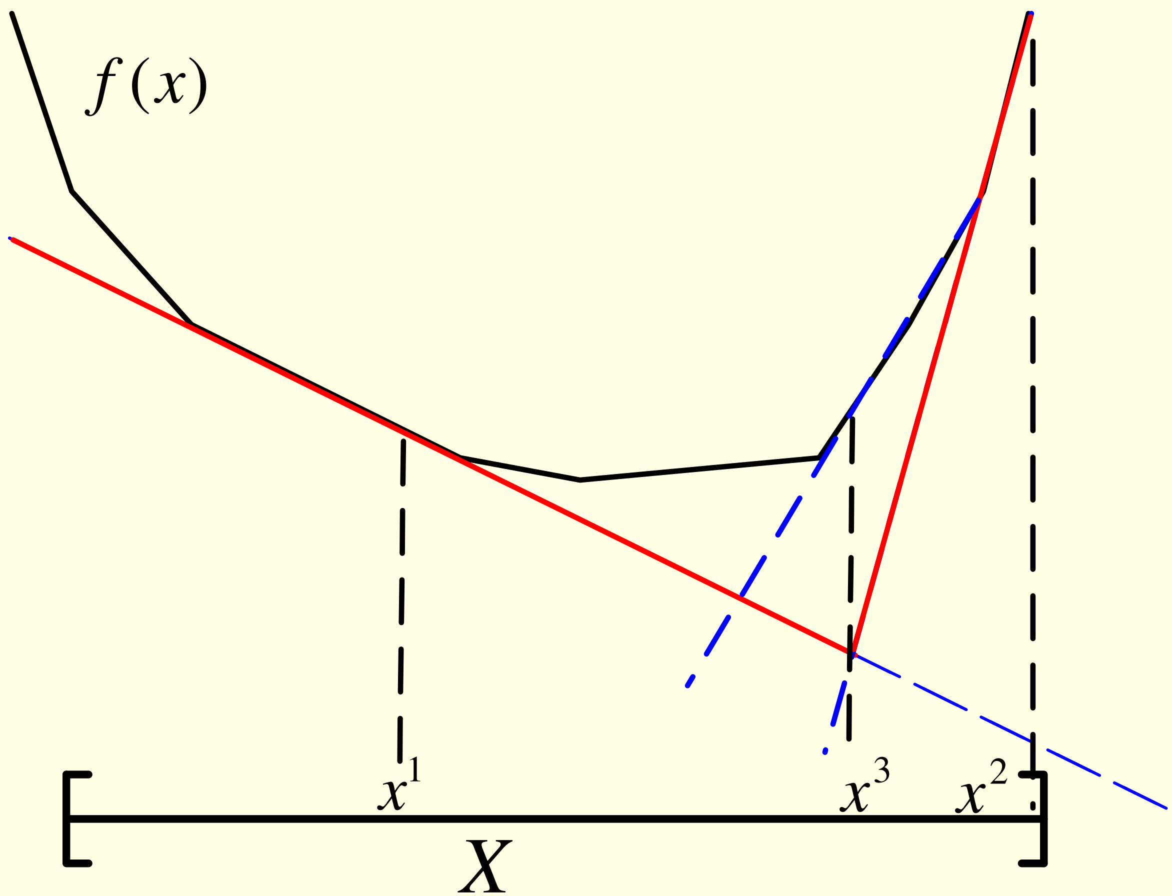
Cutting-plane methods



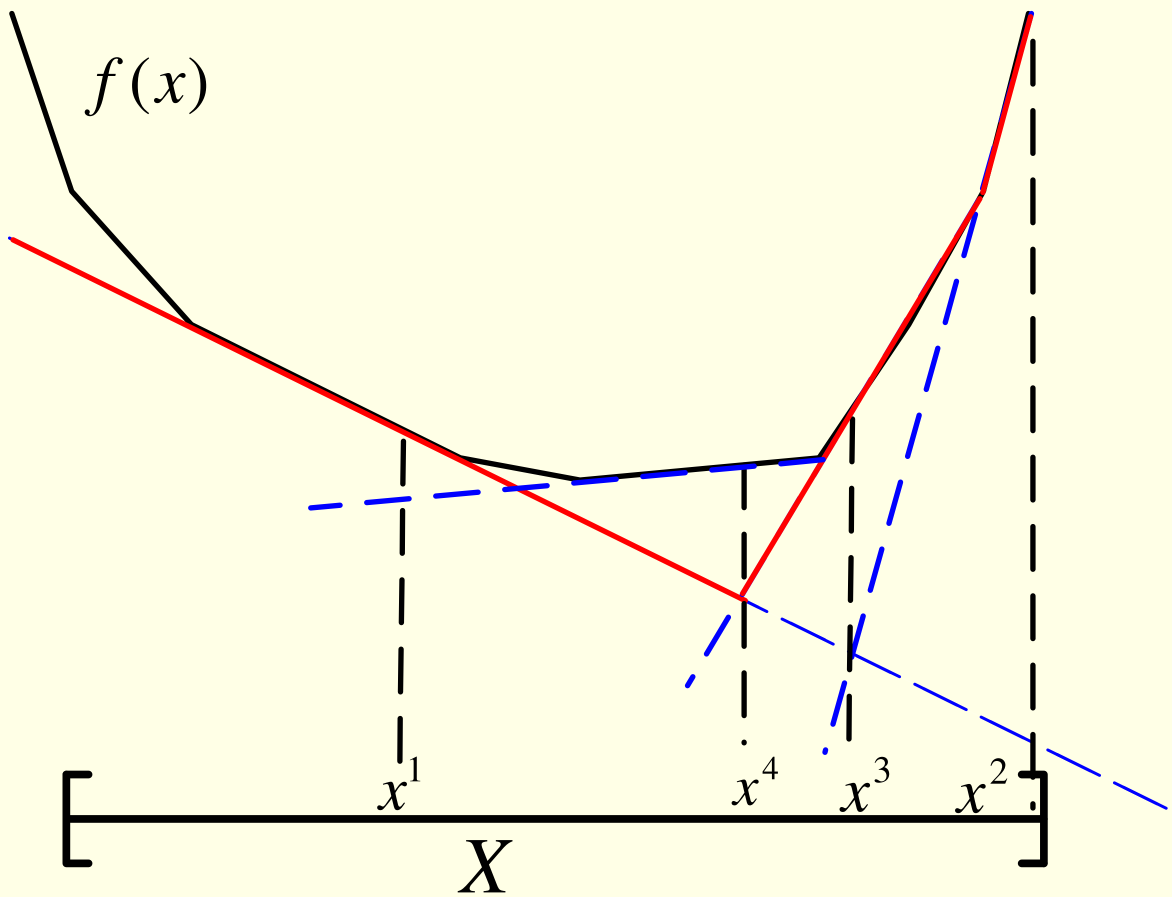
Cutting-plane methods



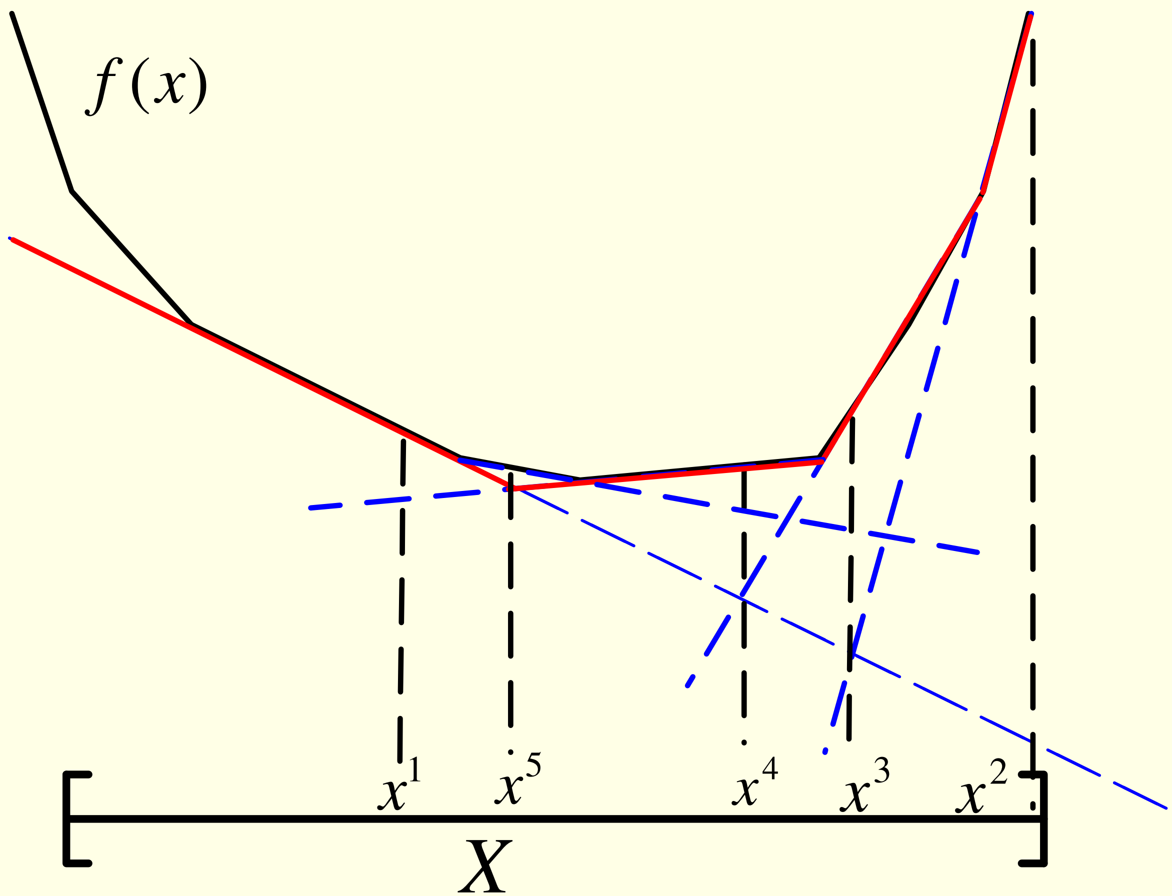
Cutting-plane methods



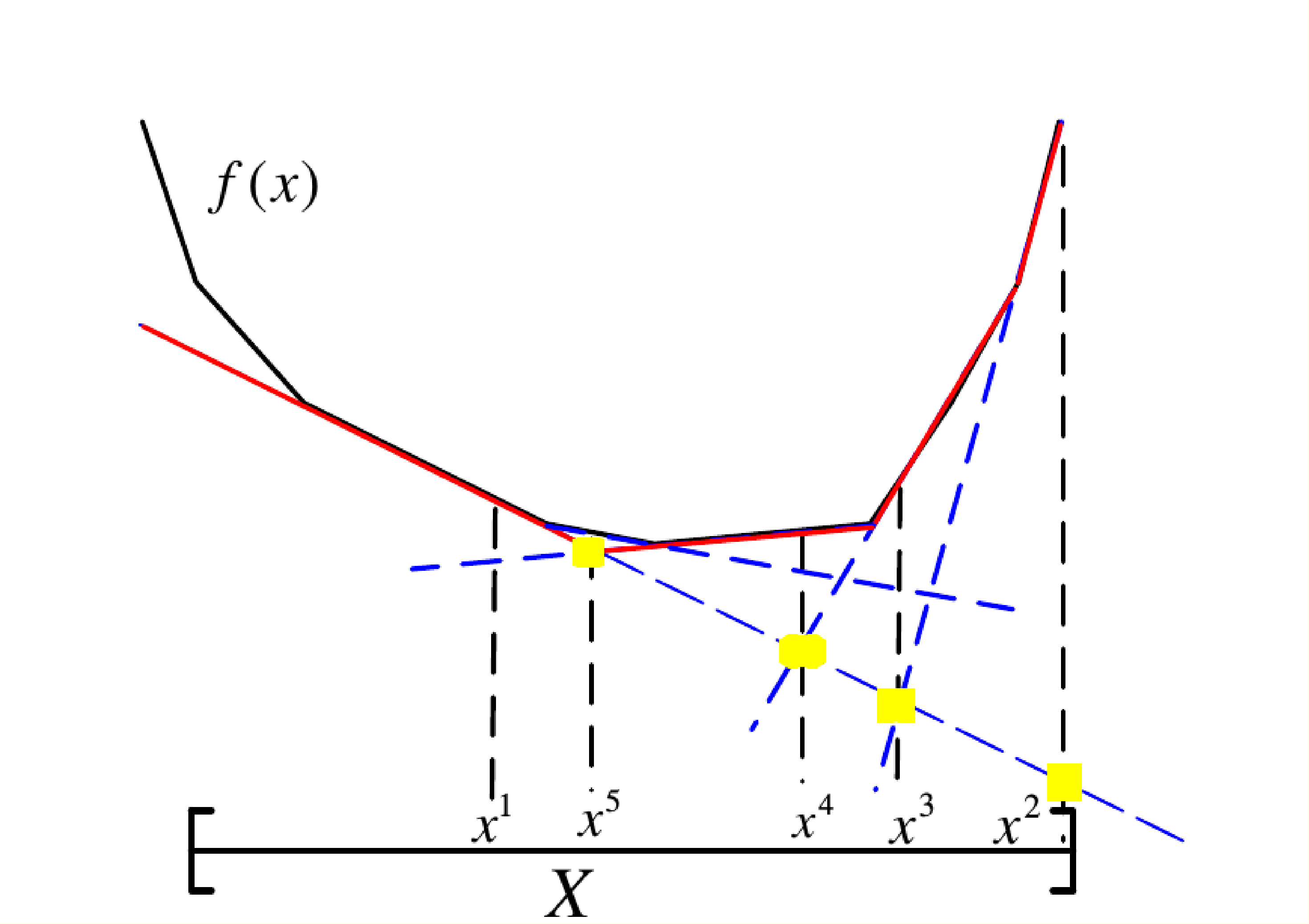
Cutting-plane methods



Cutting-plane methods

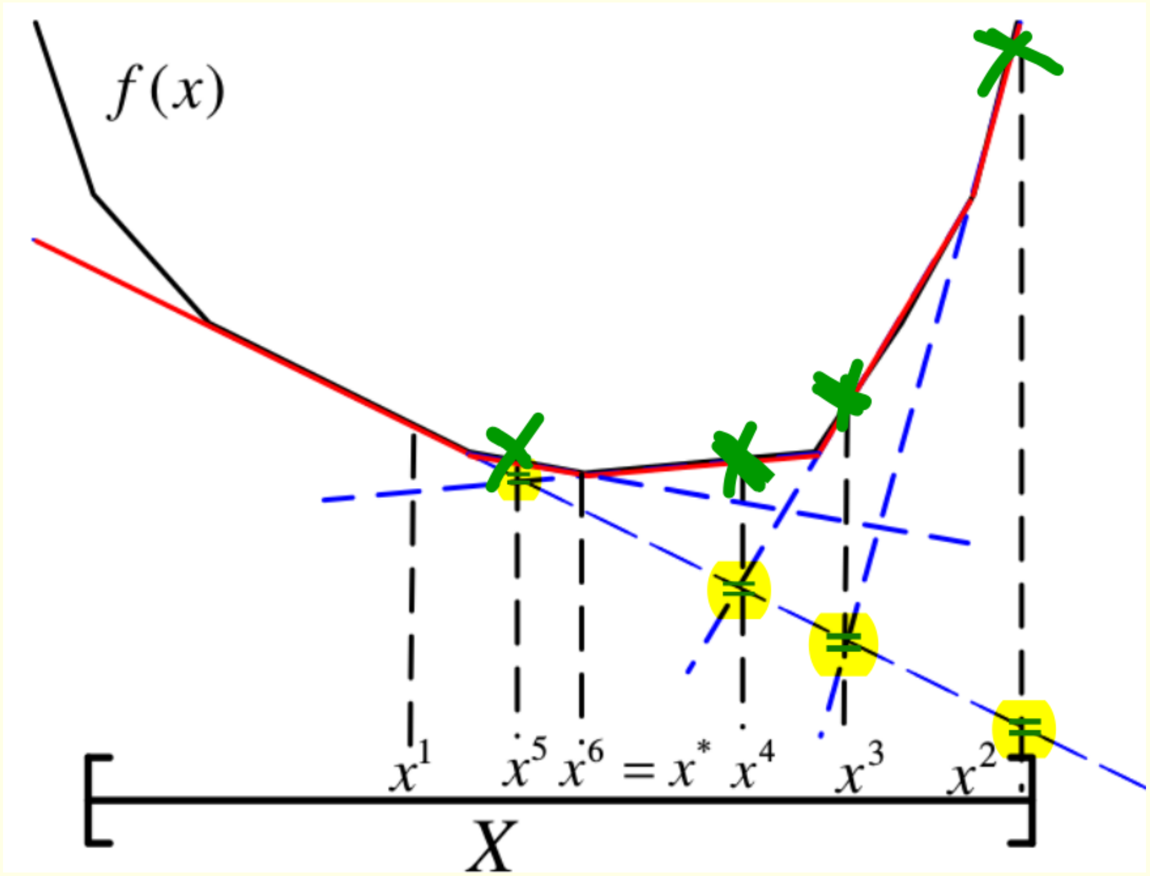


Cutting-plane methods



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

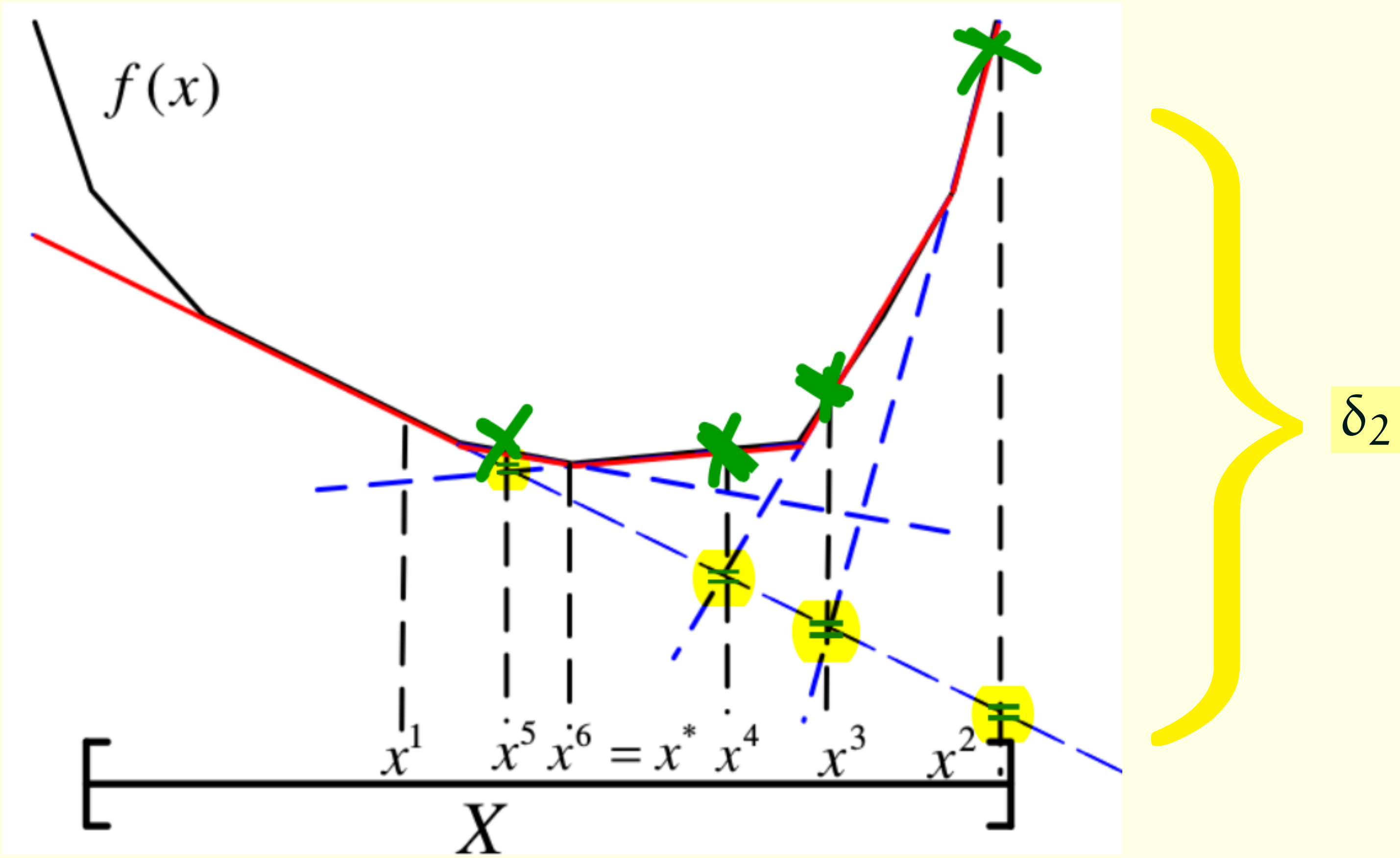
Cutting-plane methods



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

Cutting-plane Methods

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

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