Accelerating optimal black-box schemes

Michel Baes,
IFOR / ETH Zürich

Optimization Workshop II:
Numerical methods for Continuous Optimization

IPAM, UCLA, 15 October 2010

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Solving huge convex optimization problems
Black-box method vs interior-point method

$$\min \left\{ \sum_{i=1}^{m} ||A_i x - b_i||_2 : ||x||_2 \le 1 \right\}.$$

$A_i \in \mathbb{R}^{n \times n}$ is sparse, $b_i \in \mathbb{R}^n$,
absolute accuracy $= 0.001$, $m = 10$.

n	10^3 iter	CPU time Opt.	CPU time IPM	n. iter.
100	42	4.4663	0.1103	10
1000	103	49.254	6.7268	11
6000	326	1146.5	730.07	11
6500	387	1490.2	5462.6	12
7000	314	1307.5	30153	11

Solving huge convex optimization problems

Black-box method vs interior-point method

$$\min \left\{ \sum_{i=1}^{m} ||A_i x - b_i||_2 : ||x||_2 \leq 1 \right\}.$$

$A_i \in \mathbb{R}^{n\times n}$ is sparse, $b_i \in \mathbb{R}^n$,
absolute accuracy = 0.001, $m = 10$.

n	10^3 iter	CPU time Opt.	CPU time IPM	n. iter.
100	42	4.4663	0.1103	10
1000	103	49.254	6.7268	11
6000	326	1146.5	730.07	11
6500	387	1490.2	5462.6	12
7000	314	1307.5	30153	11
10000	389	2323.7	*****	-

Solving huge convex optimization problems
Black-box method vs interior-point method

$$\min \left\{ \sum_{i=1}^{m} ||A_i x - b_i||_2 : ||x||_2 \le 1 \right\}.$$

$A_i \in \mathbb{R}^{n \times n}$ is sparse, $b_i \in \mathbb{R}^n$,
absolute accuracy $= 0.001$, $m = 10$.

n	10^3 iter	CPU time Opt.	CPU time IPM	n. iter.
100	42	4.4663	0.1103	10
1000	103	49.254	6.7268	11
6000	326	1146.5	730.07	11
6500	387	1490.2	5462.6	12
7000	314	1307.5	30153	11
10000	389	2323.7	*****	-
50000	1051	33269	*****	-

Solving huge convex optimization problems

Black-box method vs interior-point method

Interior-point methods

Fast convergence:

$\mathcal{O}(m \log(C/\epsilon) + \log m)$ it.

Iteration cost:

$\mathcal{O}(n^3 + mn^2)$ flops.

Black-box methods

Slower convergence:

$\mathcal{O}(\sum_{i=1}^{m} \sigma_1(A_i)/\epsilon)$ it.

Iteration cost:

$\mathcal{O}(mn^2)$ flops.

This simple comparison justifies
the use of black-box methods
for huge-size problems.

I. The estimate sequence algorithm

The Black-Box Model

A tool to construct hard problems

$$\min_{x \in Q} f(x) \quad (\mathcal{P})$$

- ⋆ $Q \subseteq \mathbb{R}^n$ is a convex set;
- ⋆ $f : Q \to \mathbb{R}$ is convex;
- ⋆ desired accuracy on objective's value is ϵ.

The Black-Box Model

- ▶ Initially, we know nothing on $(\mathcal{P})$, but its convexity
- ▶ At every iteration, we can ask a local question on $(\mathcal{P})$
  e.g: given a point x_k, what are $f(x_k)$ and a $g_k \in \partial f(x_k)$?
  (First-Order Oracle)
- ▶ As this information is simple to get, an iteration is cheap

Smoothness plays a decisive role

$$\min_{x \in Q} f(x) \quad (\mathcal{P})$$

- ⋆ $Q \subseteq \mathbb{R}^n$ is a convex set;
- ⋆ $f : Q \to \mathbb{R}$ is convex;
- ⋆ desired accuracy on objective's value is ϵ

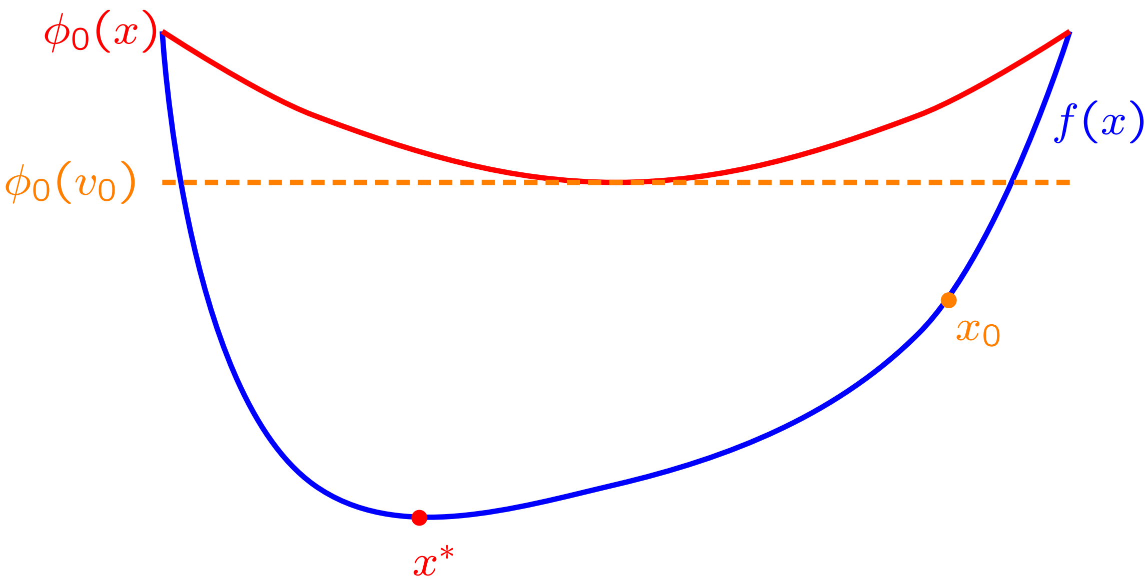
Assume $f \in \mathcal{C}^1(Q, \mathbb{R})$ and has a Lipschitz cont. gradient:

$$||f'(y) - f'(x)||_* \leq L||y - x|| \quad \text{for all } x, y \in Q.$$

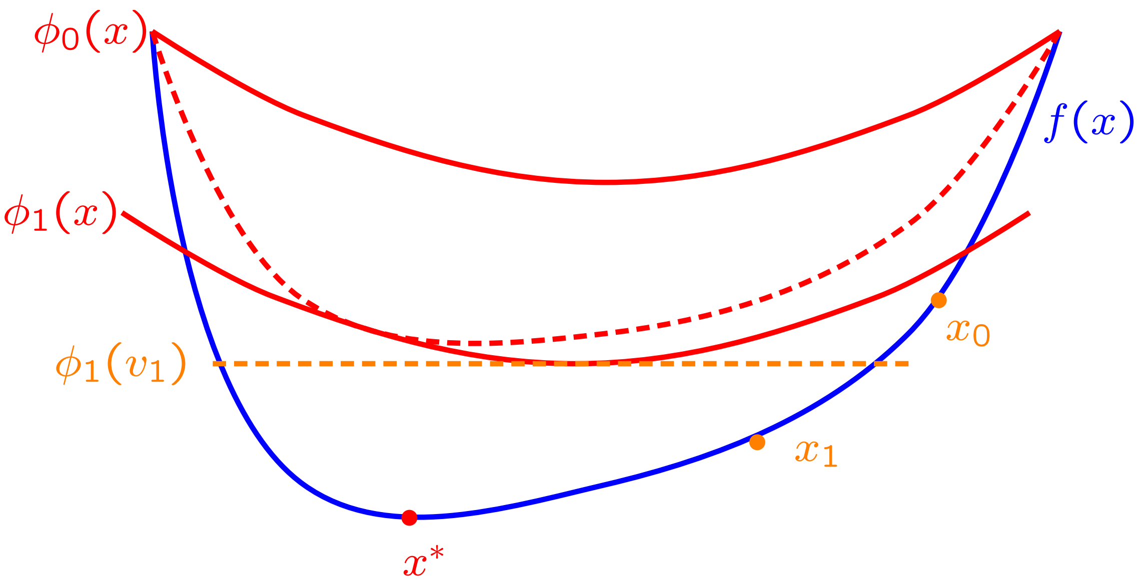
⋆ No first-order oracle method is faster than $\Omega(\min\{n, D_Q\sqrt{L/\epsilon}\})$, where $D_Q := ||x^* - x_0||$.

Optimal methods exist and are constructed using estimate sequences.

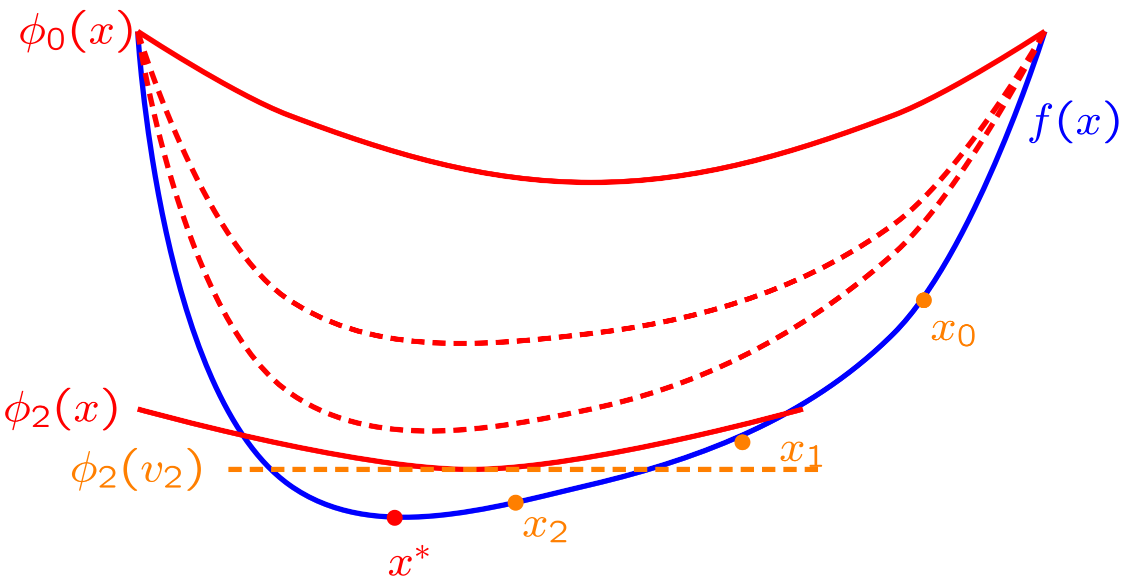
The estimate sequence approach: the general idea



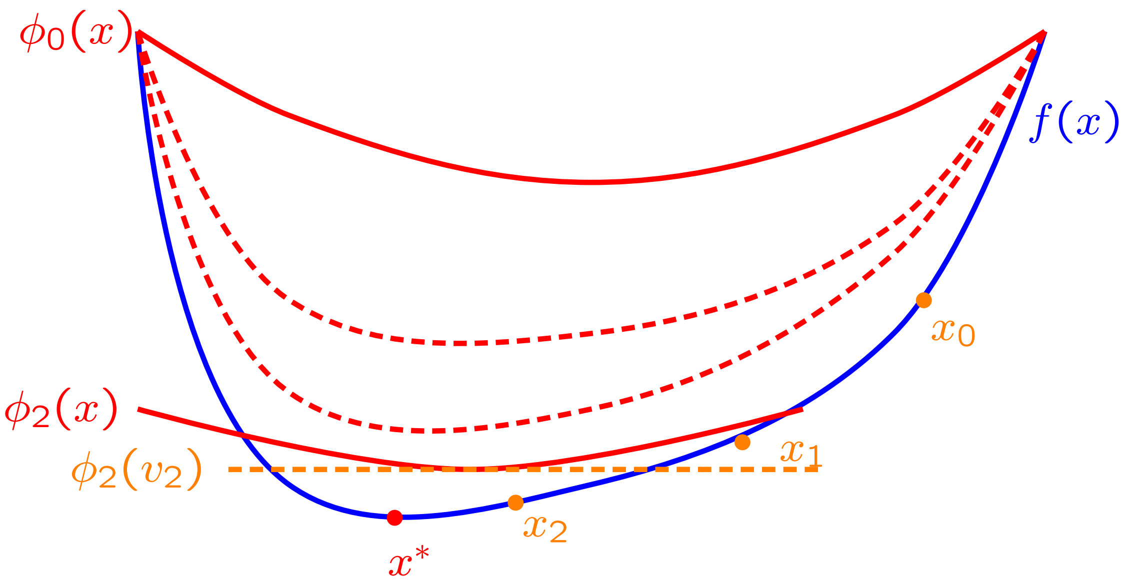
The estimate sequence approach: the general idea



The estimate sequence approach: the general idea



The estimate sequence approach: the general idea



- $\star$ Functions ϕ_k should be simple.
- $\star$ $\phi_k \leq \lambda_k \phi_0 + (1 - \lambda_k) f$.
- $\star$ $\lambda_k \to 0^+$.
- $\star$ $f(x^*) < \min \phi_0$.

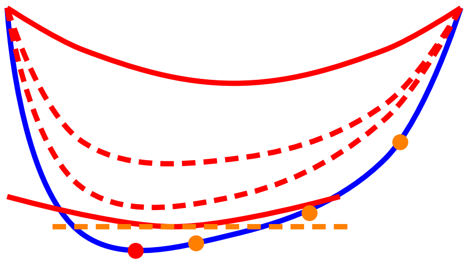
Goal: find x_k such that $f(x_k) \leq \min_x \phi_k(x) = \phi_k(v_k)$

Theorem: $f(x_k) - f(x^*) \leq \lambda_k(\phi_0(x^*) - f(x^*))$

Indeed, $f(x_k) - f(x^*) \leq \phi_k(v_k) - f(x^*)$

$$\leq \phi_k(x^*) - f(x^*) \leq \lambda_k(\phi_0(x^*) - f(x^*))$$ ■

Two issues to solve - 1

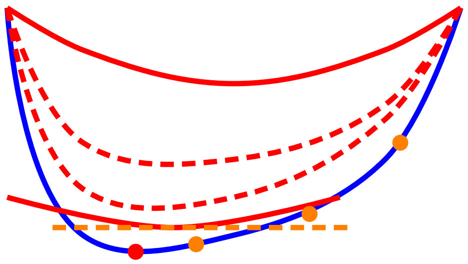


A. Constructing ϕ_k

B. Constructing x_k

such that $\phi_k(v_k) \geq f(x_k)$

Two issues to solve - 1



A. Constructing ϕ_k

B. Constructing x_k

such that $\phi_k(v_k) \geq f(x_k)$

A. Choose a simple ϕ_0 and set $\lambda_0 = 1$

If $\alpha_k \in]0,1[$, $\sum_k \alpha_k = \infty$, $\hat{f}_k(x) \leq f(x)$,

then $\phi_{k+1} := (1-\alpha_k)\phi_k + \alpha_k \hat{f}_k$,

$\lambda_{k+1} = (1-\alpha_k)\lambda_k$ is an estimate sequence

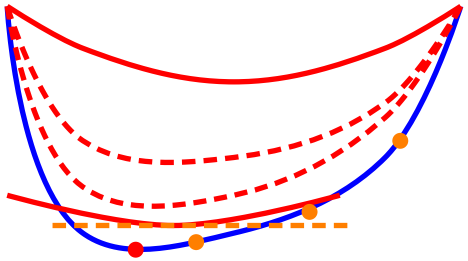
Note 1: When f is convex, we can (and will) take

$$\hat{f}_k(x) := f(y_k) + \langle f'(y_k), x - y_k \rangle$$

Thus, getting v_k is minimizing $\phi_0 +$ affine on Q

Note 2: If $\alpha_k^p/\lambda_{k+1} \geq \beta > 0\, \forall k$, then $\lambda_N \leq (p/N)^p/\beta$

Two issues to solve - 2



A. Constructing ϕ_k

B. Constructing x_k

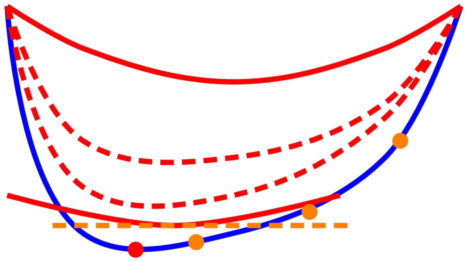
such that $\phi_k(v_k) \geq f(x_k)$

B. An *intermediate step* between $\phi_{k+1}(v_{k+1})$ and $f(x_{k+1})$.

Assume ϕ_0 is L-strongly convex: $\phi_0''(x) \succeq LI$,

thus $\phi_k''(x) \succeq \lambda_k LI$.

Two issues to solve - 2



A. Constructing ϕ_k

B. Constructing x_k

such that $\phi_k(v_k) \geq f(x_k)$

B. An *intermediate step* between $\phi_{k+1}(v_{k+1})$ and $f(x_{k+1})$.

Assume ϕ_0 is L-strongly convex: $\phi_0''(x) \succeq LI$,

thus $\phi_k''(x) \succeq \lambda_k LI$.

$$\begin{aligned}
\phi_{k+1}(x) &= (1-\alpha_k)\phi_k(x) + \alpha_k \hat{f}_k(x) \\
&\geq (1-\alpha_k)\left(\phi_k(v_k) + \langle \phi_k'(v_k), x - v_k\rangle + \lambda_k L||x - v_k||^2/2\right) + \alpha_k \hat{f}_k(x) \\
&\geq (1-\alpha_k)\left(f(x_k) + \lambda_k L||x - v_k||^2/2\right) + \alpha_k \hat{f}_k(x) \\
&\geq (1-\alpha_k)\left(f(y_k) + \langle f'(y_k), x_k - y_k\rangle + \lambda_k L||x - v_k||^2/2\right) \\
&\quad + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle) \\
&\geq f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k\rangle \\
&\quad + \min_{x \in Q} \alpha_k \langle f'(y_k), x - v_k\rangle + \lambda_{k+1} L||x - v_k||^2/2.
\end{aligned}$$

Minimize $\phi_{k+1}(x)$ on x. ■

Is this awful inequality really simpler? YES!

"Intermediate step inequality":

$$\begin{aligned}\phi_{k+1}(v_{k+1}) \;\; \geq \;\; & f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k\rangle \\ & + \min_{x \in Q} \alpha_k \langle f'(y_k), x - v_k\rangle + \lambda_{k+1} L||x - v_k||^2/2 \; ^? \geq^? \; f(x_{k+1})\end{aligned}$$

- Setting $y_k := (1-\alpha_k)x_k + \alpha_k v_k$ kills a term.

Is this awful inequality really simpler? YES!

"Intermediate step inequality":

$$\phi_{k+1}(v_{k+1}) \geq f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k\rangle$$
$$+ \min_{x\in Q} \alpha_k \langle f'(y_k), x - v_k\rangle + \lambda_{k+1} L ||x - v_k||^2/2 \; ^? \geq^? f(x_{k+1})$$

- Setting $y_k := (1-\alpha_k)x_k + \alpha_k v_k$ kills a term.
- The rest looks a lot like the inequality guaranteed by a simple gradient step:

If $y_+ = \pi_Q[f(y - f'(y)/L)]$, then:

$$f(y) + \min_{x\in Q} \langle f'(y), x - y\rangle + L||x-y||^2/2 \geq f(y_+).$$

Is this awful inequality really simpler? YES!

"Intermediate step inequality":

$$\begin{aligned}\phi_{k+1}(v_{k+1}) \;\geq\;& f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k\rangle \\ &+ \min_{x\in Q} \alpha_k \langle f'(y_k), x - v_k\rangle + \lambda_{k+1} L\|x - v_k\|^2/2 \;\overset{?}{\geq}{}^{?}\; f(x_{k+1})\end{aligned}$$

- Setting $y_k := (1-\alpha_k)x_k + \alpha_k v_k$ kills a term.
- The rest looks a lot like the inequality guaranteed by a simple gradient step:

If $y_+ = \pi_Q[f(y - f'(y)/L)]$, then:

$$f(y) + \min_{x\in Q} \langle f'(y), x - y\rangle + L\|x-y\|^2/2 \geq f(y_+).$$

- If $\alpha_k^2/\lambda_{k+1} = 1$, $x_{k+1} := \pi_Q(y_k - f'(y_k)/L)$ works! $(\Rightarrow \; \lambda_k = \mathcal{O}(1/k^2))$ i.e. OPTIMAL SCHEME.

Extensions suggested

by the Intermediate step inequality

$$\phi_{k+1}(v_{k+1}) \geq f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k \rangle$$
$$+ \min_{x \in Q} \alpha_k \langle f'(y_k), x - v_k \rangle + \lambda_{k+1} L ||x - v_k||^2/2 \ {}^{?}\!\geq^{?} f(x_{k+1})$$

- Replacing $L||x - v_k||^2/2$ by a Bregman distance can improve complexity
- Replacing $L||x - v_k||^2/2$ by $L||x - v_k||^3/6$
(Accelerating cubic regularization, *Nesterov*)
Note: To ensure the second inequality,
f'' must be L-Lipschitz continuous
Complexity: $\mathcal{O}(||x^* - x_0||(L/\epsilon)^{1/3})$
Warning: computing x_{k+1} requires a 2nd order oracle
and subproblems can be hard to solve

Extensions suggested

by the Intermediate step inequality

$$\phi_{k+1}(v_{k+1}) \geq f(y_k) + \langle f'(y_k), (1-\alpha_k)x_k + \alpha_k v_k - y_k\rangle + \min_{x \in Q} \alpha_k \langle f'(y_k), x - v_k\rangle + \lambda_{k+1} L||x - v_k||^2/2 \overset{?}{\geq} f(x_{k+1})$$

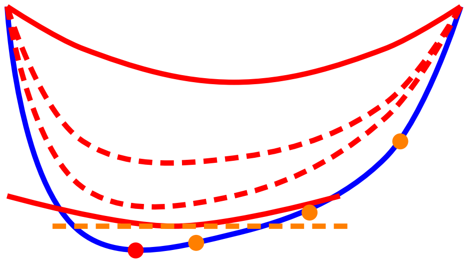
- Replacing $L||x - v_k||^2/2$ by a Bregman distance can improve complexity
- Replacing $L||x - v_k||^2/2$ by $L||x - v_k||^m/m$!

  Note: To ensure the second inequality, $f^{(m)}$ must be L-Lipschitz continuous

  Complexity: $\mathcal{O}(||x^* - x_0||(L/\epsilon)^{1/m})$

  Warning: computing x_{k+1} requires an $m-1$ order oracle and subproblems can be hard to solve

The estimate sequence algorithm



Permanent task:
ensuring $\phi_k(v_k) \geq f(x_k)$

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q}\phi_{k+1}(x)$.
End ∎

Potential drawbacks

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q}\phi_{k+1}(x)$.
End ■

1.- A priori, no other stopping criterion than
the iteration counter: this algorithm is primal-only.
2.- Sophisticated method, whose extensions
are not easy to develop.
3.- In spite of its optimality, not frequently used in practice
before 2005 (except for smoothing techniques)

II. Extending
the estimate sequence algorithm

Our toy problem:

minimizing matrix p-norms

Given a matrix $B \in \mathbb{S}^n$ and $A_1, \ldots, A_m \in \mathbb{S}^n$, $(m < n)$ represent B as $\sum_{i=1}^m x_i A_i$ "at best".

Our toy problem:
minimizing matrix p-norms

Given a matrix $B \in \mathbb{S}^n$ and $A_1, \ldots, A_m \in \mathbb{S}^n$, $(m < n)$ represent B as $\sum_{i=1}^m x_i A_i$ "at best".

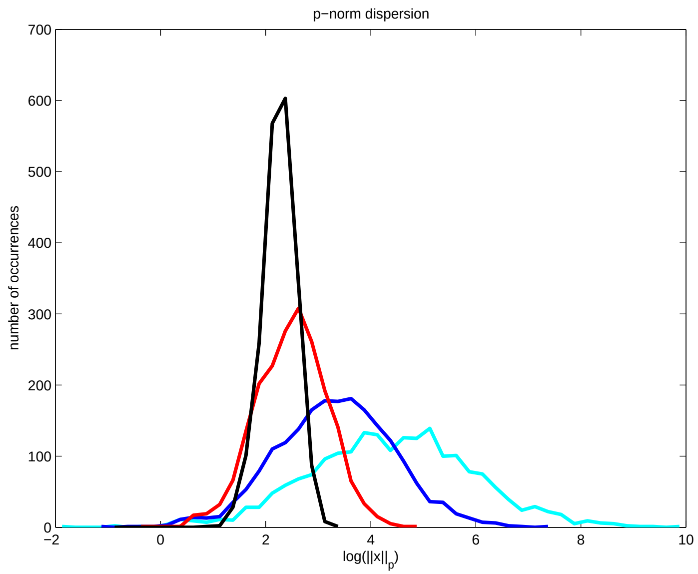
Our criterion: $\min ||B - \sum_{i=1}^m x_i A_i||_p^2/2$, with $p > 2$

Our toy problem: minimizing matrix p-norms

Given a matrix $B \in \mathbb{S}^n$ and $A_1, \ldots, A_m \in \mathbb{S}^n$, $(m < n)$
represent B as $\sum_{i=1}^m x_i A_i$ "at best".
Our criterion: $\min ||B - \sum_{i=1}^m x_i A_i||_p^2/2$, with $p > 2$

Why the p-norm?

When matrices are large, eigenvalues p-norms are more dispersed than with the 2-norm.



We can use estimate sequences on our toy problem

$$\min ||B - \textstyle\sum_{i=1}^{m} x_i A_i||_p^2/2$$

Smoothness: due to the $(p-1)$-regularity of $X \mapsto ||X||_p^2/2$.

Explicit computation of the gradient,
albeit in $\mathcal{O}(n^3 + mn^2)$, due to a necessary eigendecomposition of $B - \sum_{i=1}^{m} x_i A_i$.

Strong dual: with $1/p + 1/q = 1$,

$$\max\{||S||_q^2/2 - \langle S, B\rangle : \langle S, A_i\rangle = 0,\ 1 \le i \le m\}.$$

We can use estimate sequences on our toy problem

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q}\phi_{k+1}(x)$.
End ∎

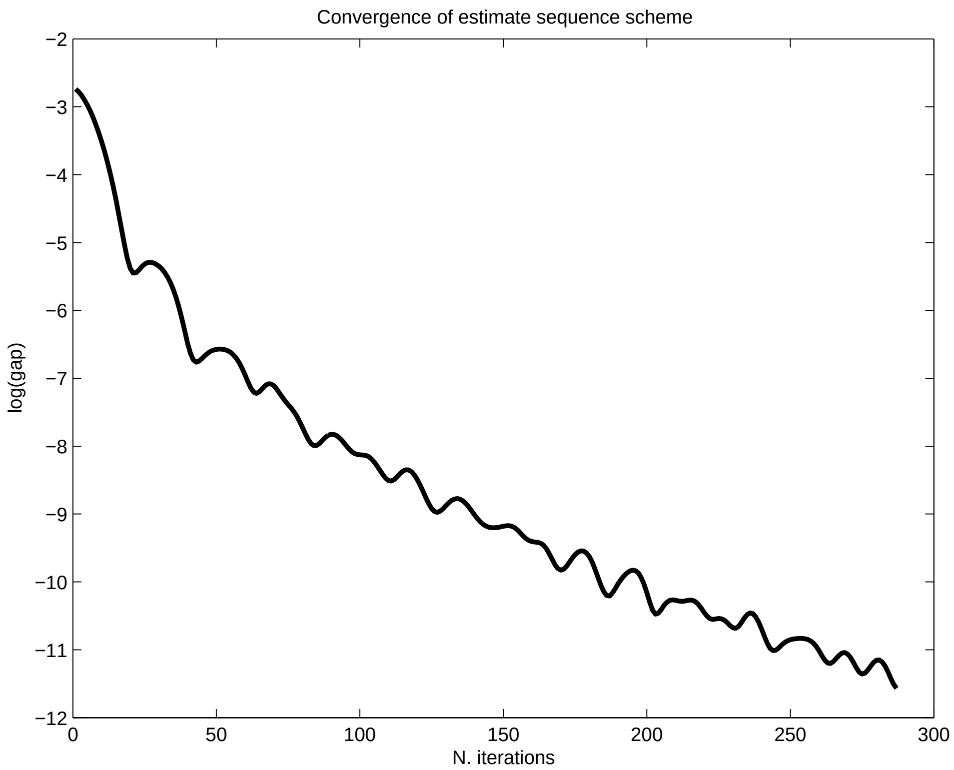
Best prox-function: if we take $d(x) := ||x - x_0||_\gamma^2/2$,
the optimal γ is close to $1/(\log(n) - 2)$ for n large.
We can evaluate L in $\mathcal{O}(n^3)$ once forever.

We can use estimate sequences on our toy problem

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x-y_k\rangle + Ld(x,y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x-y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q}\phi_{k+1}(x)$.
End ■

Best prox-function: if we take $d(x) := ||x-x_0||_\gamma^2/2$,
the optimal γ is close to $1/(\log(n)-2)$ for n large.
We can evaluate L in $\mathcal{O}(n^3)$ once forever.

Analytic solution of subproblem, which costs $\mathcal{O}(m)$.



Here, $n = 200$, $m = 20$, $p = 16$

CPU time for $\epsilon = 0.0001$**:** 126.39s

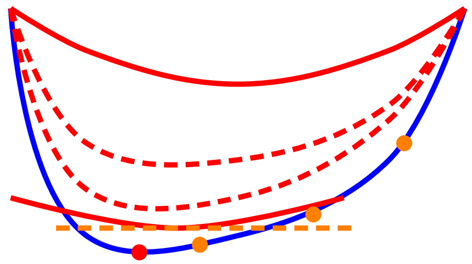
Three ways to accelerate the algorithm

1: Approximations

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q}\phi_{k+1}(x)$.
End ■

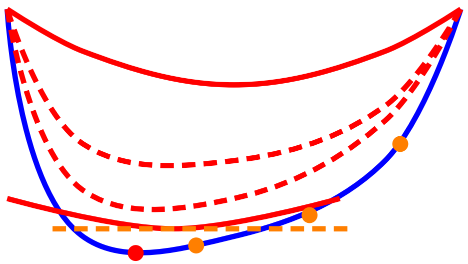
In our toy problem, computing $f'(y_k)$ is very expensive. What about using a suitable approximation of $f'(y_k)$?

A suitable approximation of the gradient



Permanent task:
ensuring $\phi_k(v_k) \geq f(x_k)$

A suitable approximation of the gradient



New permanent task:
ensuring $\phi_k(v_k) \geq f(x_k) - \epsilon$

Instead of using $f'(y_k)$, we use $g \in \mathbb{R}^n$ such that:

$$f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g \quad \forall x \in \text{dom} f$$

and $f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g ||x - y_k|| \quad \forall x \in \text{dom} f$

Note: The second inequality implies $||g - f'(y_k)||_* \leq \epsilon_g$

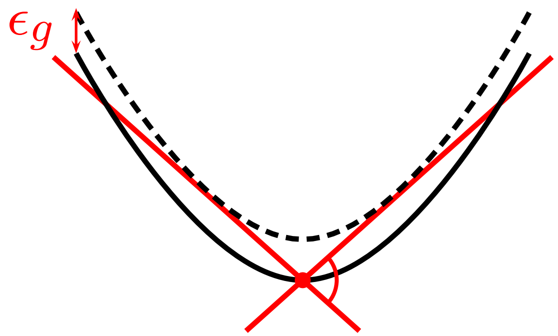
We write $g \in \partial_{\epsilon_g} f(y_k)$.

A suitable approximation of the subgradient

Instead of using $f'(y_k)$, we use $g \in \mathbb{R}^n$ such that:

$$f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g \quad \forall x \in \mathrm{dom} f$$

and $$f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g ||x - y_k|| \quad \forall x \in \mathrm{dom} f$$



Note: The second inequality implies $||g - f'(y_k)||_* \leq \epsilon_g$

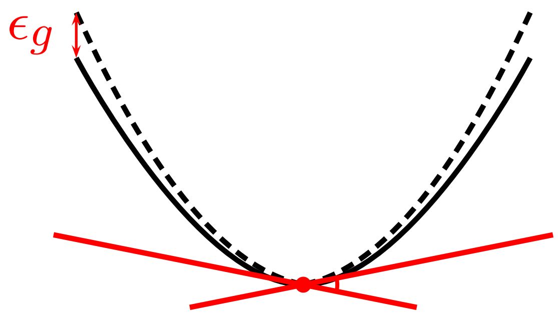
We write $g \in \partial_{\epsilon_g} f(y_k)$.

A suitable approximation of the subgradient

Instead of using $f'(y_k)$, we use $g \in \mathbb{R}^n$ such that:

$$f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g \quad \forall x \in \text{dom} f$$

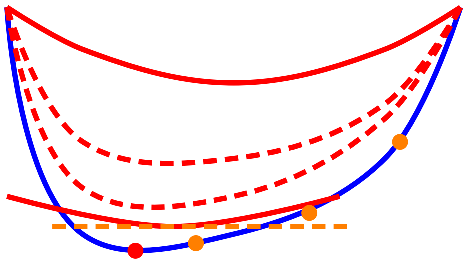
and $f(x) \geq f(y_k) + \langle g, x - y_k \rangle - \epsilon_g ||x - y_k|| \quad \forall x \in \text{dom} f$



Note: The second inequality implies $||g - f'(y_k)||_* \leq \epsilon_g$

We write $g \in \partial_{\epsilon_g} f(y_k)$.

A suitable approximation of the subgradient



We assume some error has already been done: $\phi_k(v_k) \geq f(x_k) - \epsilon$

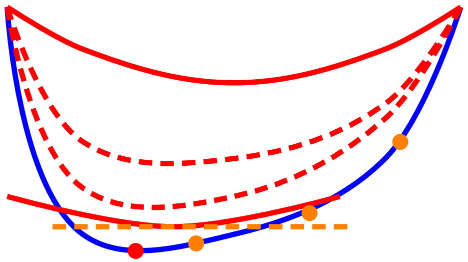
Suppose that Q is bounded:

$D_Q := \sup\{||y - x|| : y, x \in Q\} < \infty$

If $g_k \in \partial_{\epsilon_{g,k}} f(y_k)$, with $\epsilon_{g,k} \leq \alpha_k \epsilon / (D_Q + 1)$,

then $\phi_{k+1}(v_{k+1}) \geq f(\hat{x}_{k+1}) - \epsilon$: no error propagation

A suitable approximation of the subgradient



We assume some error has already been done: $\phi_k(v_k) \geq f(x_k) - \epsilon$

Suppose that Q is bounded:
$D_Q :=$ sup$\{||y - x|| : y, x \in Q\} < \infty$
If $g_k \in \partial_{\epsilon_{g,k}} f(y_k)$, with $\epsilon_{g,k} \leq \alpha_k \epsilon/(D_Q + 1)$,
then $\phi_{k+1}(v_{k+1}) \geq f(\hat{x}_{k+1}) - \epsilon$: no error propagation

Note 1: Construction of the estimate sequence
with $\hat{f}_k(x) := f(y_k) + \langle g_k, x - y_k \rangle - \epsilon_{g,k}$

Note 2: In our case, $\epsilon_{g,k} = \mathcal{O}(\epsilon/||B||_q)$ is enough.

Three ways to accelerate the algorithm

1: Approximations

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x \in Q} \langle f'(y_k), x - y_k \rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k \rangle)$;
Set $v_{k+1} := \arg\min_{x \in Q} \phi_{k+1}(x)$.
End ■

To guarantee

$$\phi_k(v_k) \geq f(x_k) - \epsilon \Rightarrow \phi_{k+1}(v_{k+1}) \geq f(x_{k+1}) - \epsilon,$$

we can compute x_{k+1} with an absolute accuracy of $\alpha_k \epsilon$.
Note: Also valid for cubic and m-regularization

Three ways to accelerate the algorithm 1: Approximations

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x \in Q} \langle f'(y_k), x - y_k \rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k \rangle)$;
Set $v_{k+1} :=$ $\arg\min_{x \in Q} \phi_{k+1}(x)$.
End ■

To guarantee

$$\phi_k(v_k) \geq f(x_k) - \epsilon \Rightarrow \phi_{k+1}(v_{k+1}) \geq f(x_{k+1}) - \epsilon,$$

we can compute v_{k+1} with an absolute accuracy of

$$\min\left\{1, \left(\frac{\alpha_k}{1-\alpha_k}\right)^2 \left(\frac{\epsilon}{1 + D_Q L\sqrt{2\lambda_k/\sigma}}\right)^2\right\}.$$

Three ways to accelerate the algorithm 1: Approximations

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} :=$ $\arg\min_{x\in Q}\phi_{k+1}(x)$.
End ∎

- For cubic regularization, and m-regularization, we can compute v_{k+1} with an absolute accuracy of:

$$\min\left\{1, \left(\frac{\alpha_k}{1-\alpha_k}\right)^m \left(\frac{\epsilon}{1 + D_Q(L/\sigma)\sqrt[m]{m(\lambda_k\sigma)^{m-1}}}\right)^m\right\}.$$

Illustration 1: cheap approximate gradients for our toy problem

$$\min ||B - \sum_{i=1}^{m} x_i A_i||_p^2/2 = \min ||B - \mathcal{A}(x)||_p^2/2$$

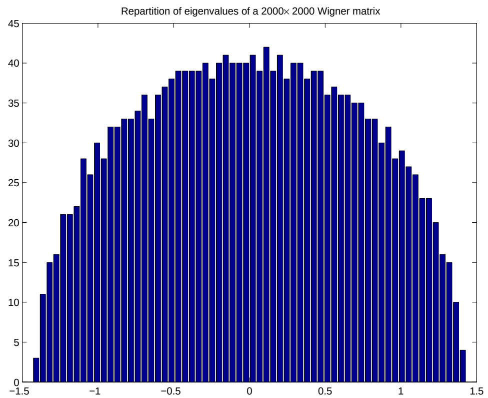
For computing the gradient exactly, we need a full eigenvalue decomposition of $B - \mathcal{A}(x)$.

What happens if we retain only the ones with largest magnitude?

Illustration 1: cheap approximate gradients for our toy problem

$$\min ||B - \textstyle\sum_{i=1}^{m} x_i A_i||_p^2/2 = \min ||B - \mathcal{A}(x)||_p^2/2$$

Assumption:
$\{A_i\}, B$ are
normalized Wigner:
(iid coeff. $\sim N(0, \sigma^2/n)$).
Then $B - \mathcal{A}(x)$ is Wigner.

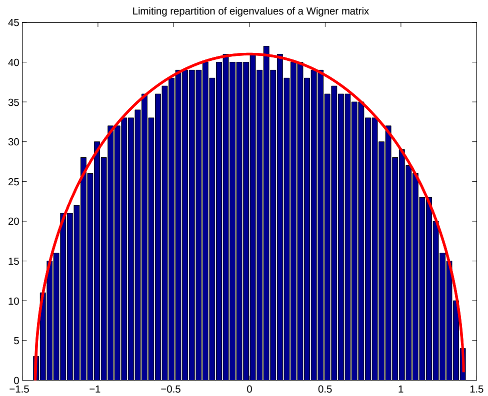


Fact: Eigenvalue distribution of large Wigner matrices is very regular.

Illustration 1: cheap approximate gradients for our toy problem

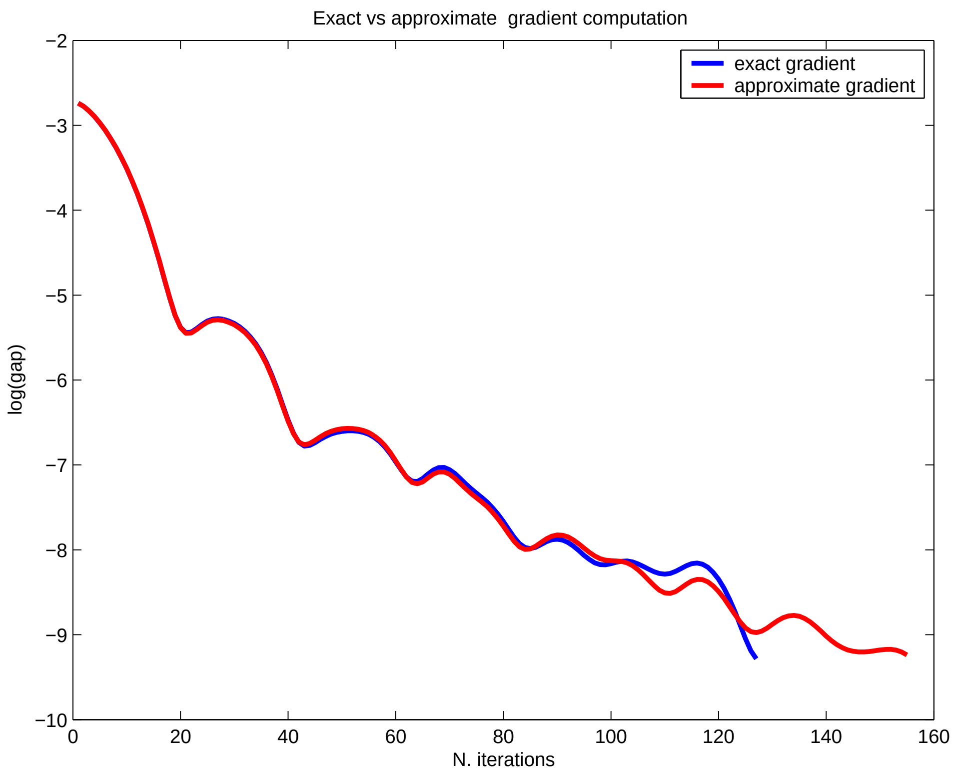
$$\min ||B - \sum_{i=1}^{m} x_i A_i||_p^2/2 = \min ||B - \mathcal{A}(x)||_p^2/2$$

Assumption:
$\{A_i\}, B$ are
normalized Wigner
(iid coeff. $\sim N(0, \sigma^2/n)$).
Then $B - \mathcal{A}(x)$ is Wigner.



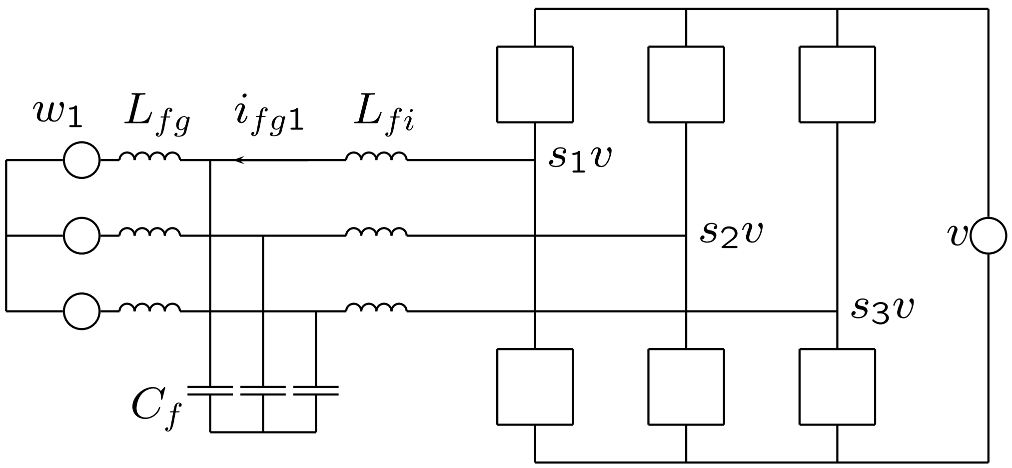
Using a recent result of Erdos *et al.*, we have bounds on the number of eigenvalues for approxim. gradient

e.g.: if $p = 16$, $n \geq 1000$, $\epsilon = 0.0001$, $||x||_1 \leq 1$, 10% are enough.



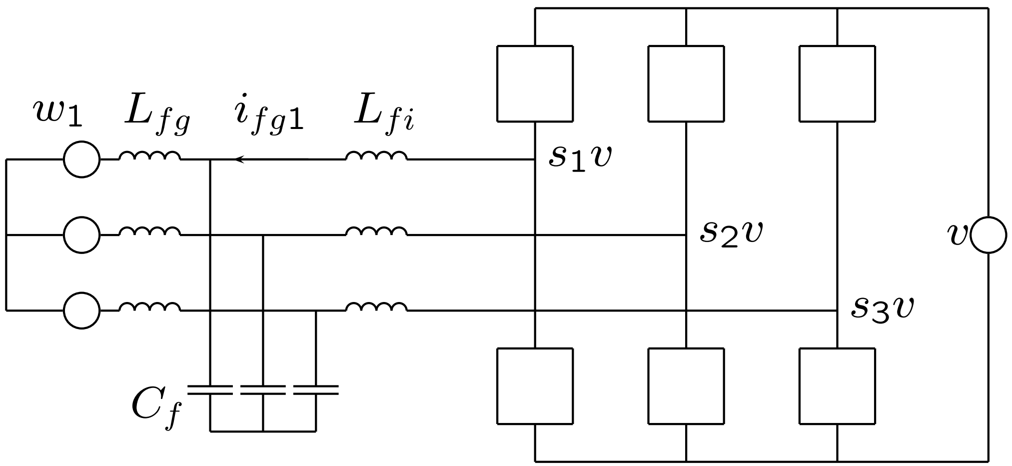
Here, $n = 200$, $m = 20$, $p = 16$,
25% of eigenvalues retained as $||x_k||_1 \leq 1$
CPU time for $\epsilon = 0.0001$**:** 64.4060s
(vs. 126.39s for the exact gradient version)

Illustration 2: Solving an MPC fast and inexpensively



AC/DC power converter *(Richter, Mariethoz, Morari, 2010)*

Illustration 2: Solving an MPC fast and inexpensively



$$\begin{aligned} \text{min} \quad & \textstyle\sum_{k=0}^{N-1} \|u_k - u_{ss}\|^2 + \sum_{k=0}^{N-1} \|x_k - x_{ss}\|^2 \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + B_w w \\ & u_k \in \mathbb{U}(v, \phi - k\omega\Delta T) \\ & x_0 = \tilde{x} \\ & \text{A stability criterion} \end{aligned}$$

(Richter, Mariethoz, Morari, 2010)

Illustration 2: Solving an MPC fast and inexpensively

$$\begin{array}{rl} \min & \sum_{k=0}^{N-1} ||u_k - u_{ss}||^2 + \sum_{k=0}^{N-1} ||x_k - x_{ss}||^2 \\ \text{s.t.} & x_{k+1} = Ax_k + Bu_k + B_w w \\ & u_k \in \mathbb{U}(v, \phi - k\omega\Delta T) \\ & x_0 = \tilde{x} \\ & \text{A stability criterion} \end{array}$$

- Here, $\mathbb{U}(v, \phi - k\omega\Delta T)$ is a regular hexagon of radius v, tilted with angle $\phi - k\omega\Delta T$
- An accuracy of $\epsilon = 0.02$ is enough
- We can only afford cheap processors, we need to be fast
- Stability criterion 1: add $\langle Qx_N, x_N \rangle$ to objective, for a well-chosen Q_N.
  Estimate sequences work very efficiently
  (Richter, Mariethoz, Morari, 2010)

Illustration 2: Solving an MPC fast and inexpensively

$$
\begin{array}{rl}
\text{min} & \sum_{k=0}^{N-1} ||u_k - u_{ss}||^2 + \sum_{k=0}^{N-1} ||x_k - x_{ss}||^2 \\
\text{s.t.} & x_{k+1} = Ax_k + Bu_k + B_w w \\
& u_k \in \mathbb{U}(v, \phi - k\omega\Delta T) \\
& x_0 = \tilde{x} \\
& \text{A stability criterion}
\end{array}
$$

- Observation: if, due to circuit imperfections, hexagons $\mathbb{U}_k := \mathbb{U}(v, \phi - k\omega\Delta T)$ are not regular, estimate sequences still work if projections on imperfect hexagons are within $\mathcal{O}(\epsilon^3/N)$ of projections on $\mathbb{U}_k$.

Illustration 2: Solving an MPC fast and inexpensively

$$
\begin{array}{ll}
\text{min} & \sum_{k=0}^{N-1} ||u_k - u_{ss}||^2 + \sum_{k=0}^{N-1} ||x_k - x_{ss}||^2 \\
\text{s.t.} & x_{k+1} = Ax_k + Bu_k + B_w w \\
& u_k \in \mathbb{U}(v, \phi - k\omega\Delta T) \\
& x_0 = \tilde{x} \\
& \text{A stability criterion}
\end{array}
$$

- Stability criterion 2: add as constraint $l \leq x_N \leq u$
- We can afford approximate projections (accuracy $\mathcal{O}(\epsilon^3)$)
- In practice, requires about 50 times
  the projection cost on $\mathbb{U}_k$
  (These results are only preliminary)

Three ways to accelerate the algorithm

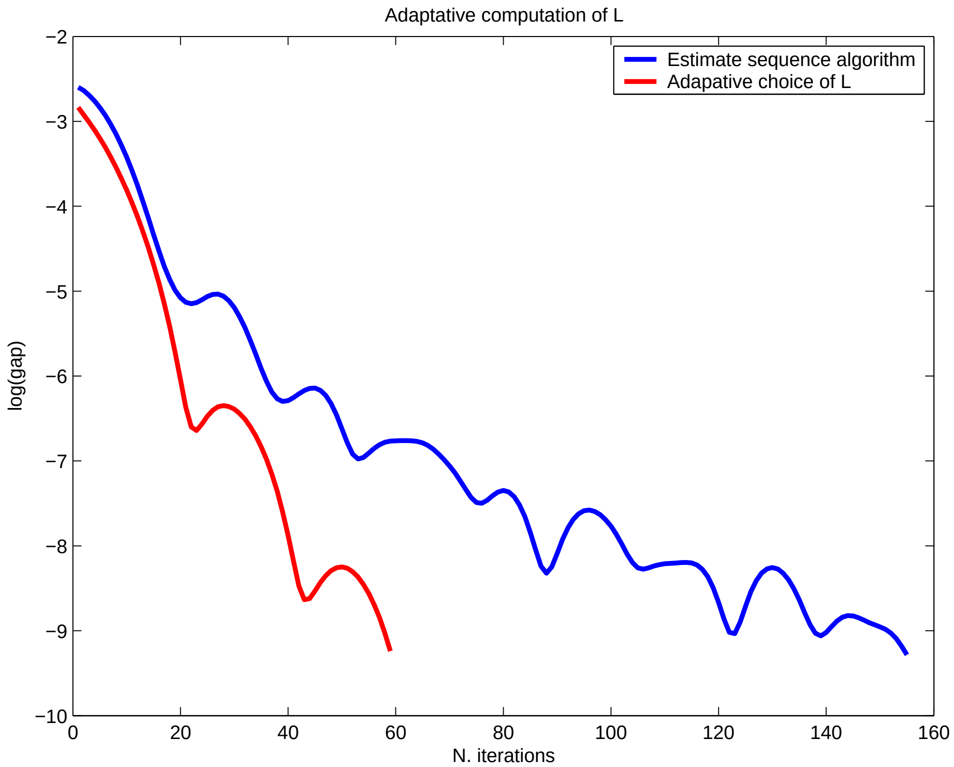
2: Reevaluation of L

Algorithm 1 *Set* $\phi_0 := f(x_0) + Ld(x)$, $v_0 := x_0$.
For $k \geq 0$,
Find α_k *such that* $\alpha_k^2 = (1-\alpha_k)\lambda_k$; *set* $\lambda_{k+1} := (1-\alpha_k)\lambda_k$;
Set $y_k := \alpha_k v_k + (1-\alpha_k)x_k$;
Set $x_{k+1} := \arg\min_{x\in Q}\langle f'(y_k), x - y_k\rangle + Ld(x, y_k)$;
Set $\phi_{k+1}(x) := (1-\alpha_k)\phi_k(x) + \alpha_k(f(y_k) + \langle f'(y_k), x - y_k\rangle)$;
Set $v_{k+1} := \arg\min_{x\in Q} \phi_{k+1}(x)$.
End ∎

Smoothness makes $\phi_k(v_k) \geq f(x_k)$ possible. In fact,

- only $f(y_k) + \langle f'(y_k), x_{k+1} - y_k\rangle + L||x_{k+1} - y_k||^2/2 \geq f(x_{k+1})$ is enough.
- The smaller L, the bigger step-sizes, the faster convergence.

Allow a decrease of L below its actual value *(Nesterov)*.



Here, $n = 200$, $m = 20$, $p = 16$

CPU time for $\epsilon = 0.0001$**: 52.4850s**

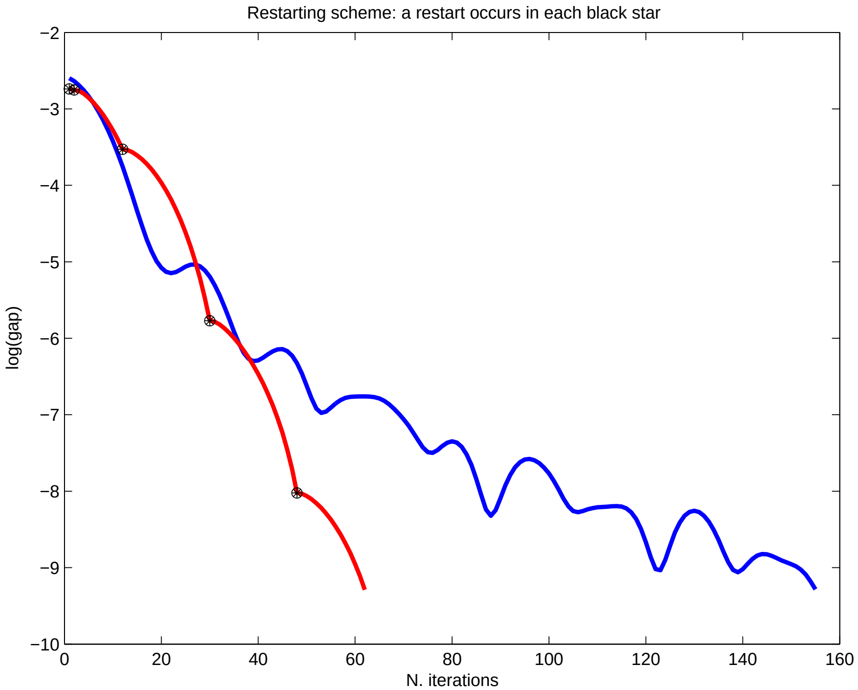
(vs. 126.39s for the exact gradient version)

Three ways to accelerate the algorithm 3: Restarting the estimate sequence

A potential drawback of the estimate sequence algorithm is that it carries early information even in the last stages of the computations.

Also, this algorithm converges usually (much) faster than what the theory predicts when the starting point is close from the optimum.

We solve the problem for a crude accuracy, and restart the algorithm from the point found for a higher accuracy.



Here, $n = 200$, $m = 20$, $p = 16$. Restart at accuracies Le^{-2}, Le^{-4},...

CPU time for $\epsilon = 0.0001$**:** 46.9608s

(vs. 126.39s for the exact gradient version).

Note: Consider restarting at $\epsilon_0, \gamma\epsilon_0, \gamma^2\epsilon_0, \ldots$.

If there exists $\Gamma > 0$ such that $f(x) - f^* \geq \Gamma\phi_0(x) - f(x_0)$,

$\gamma^* = \exp(-2)$, and we need $\mathcal{O}(\sqrt{1/\Gamma}\log(\epsilon_1/\epsilon))$ it.

Mixed strategies

- Restarting (R) is the fastest on moderate size problems Combined with reevaluating L (L) yields the smallest number of iterations.
- For large problems, restarting + approximate subgradients (A) is the fastest (due also to very crude initial gradient approximation)

Strategies	m	n	ϵ	n. iter	CPU (s)
Standard	10	100	1e-5	150	3.95
R				120	3.36
Standard	10	100	1e-7	830	21.95
R				200	5.35
RL				130	8.20
Standard	10	1000	1e-5	70	1603
RA				60	947

Thank you!