Regularized Nonlinear Acceleration

Damien Scieur

with **Alexandre d'Aspremont, Francis Bach.**
CNRS, INRIA, Ecole Normale Supérieure, Paris.

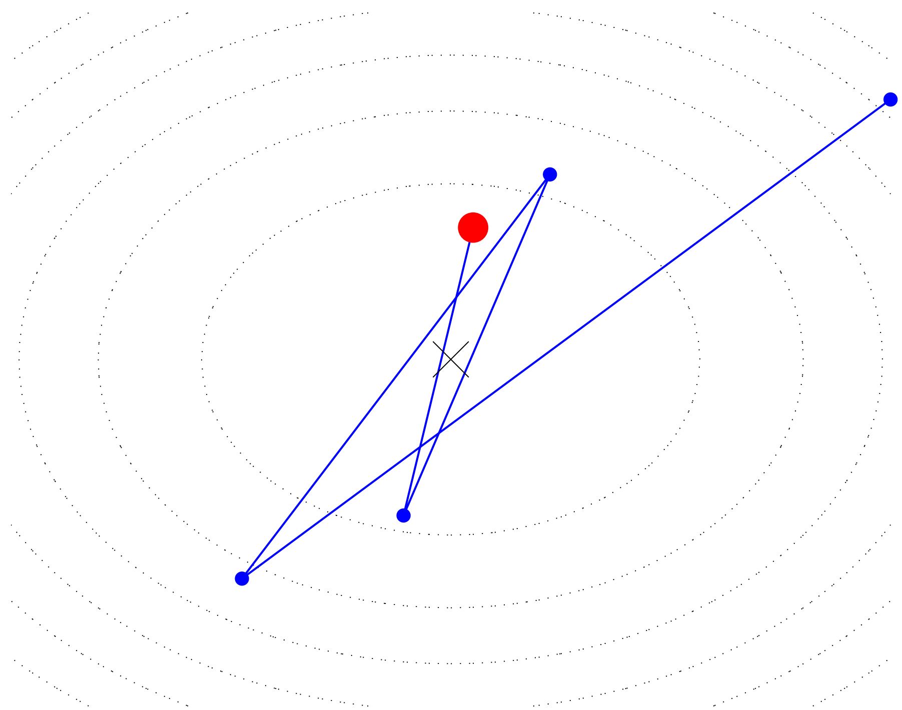
Support from ITN MacSeNet and ERC SIPA.

SpaRTaN
Sparse Representations and Compressed
Sensing Training Network

Introduction

Algorithms produce a **sequence** of iterates. We usually keep the last/best one, or the mean of all iterates.

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



Can we do better?

!!! Spoilers !!!

General structure of the acceleration:

- Run your favorite algorithm
- Extrapolate a new point by solving a (typically) 5×5 linear system
- Enjoy ;-)

Introduction - Aitken's Δ^2

Aitken's Δ^2 [Aitken, 1927]. Given a scalar sequence s_k with fixed-point s^*, following an auto-regressive process

$$s_{k+1} - s_* = a\,(s_k - s_*), \quad \text{for } k = 1, \ldots$$

Step 1: We estimate a from $\{s_{k-1},\, s_k, s_{k+1}\}$ using their differences

$$s_{k+1} - s_k = a\,(s_k - s_{k-1}) \qquad \Rightarrow \qquad a = \frac{s_{k+1} - s_k}{s_k - s_{k-1}}.$$

Step 2: Get **exactly** the limit s^* by injecting the estimate of a

$$s_{k+1} - s^* = \frac{s_{k+1} - s_k}{s_k - s_{k-1}}(s_k - s^*) \quad \Rightarrow \quad s^* = \underbrace{\frac{s_{k-1}s_{k+1} - s_k^2}{s_{k+1} - 2s_k + s_{k-1}}}_{\text{Aitken's } \Delta^2 \text{ formula}}.$$

It needs **only three iterates** (and nothing else).

Introduction - Extrapolation of scalar sequences

Convergence acceleration. Consider

$$s_k = \sum_{i=0}^{k} \frac{(-1)^i}{(2i+1)} \quad \xrightarrow{k\to\infty} \quad \frac{\pi}{4} = 0.785398\ldots$$

This sequence is not auto-regressive. However, we have

k	$\sum_{i=0}^{k} \frac{(-1)^i}{(2i+1)}$	Aitken's Δ^2
0	1.0000	–
1	0.66667	–
2	0.86667	**0.7**9167
3	**0.7**2381	**0.78**333
4	0.83492	**0.78**631
5	**0.7**4401	**0.78**492
6	0.82093	**0.785**68
7	**0.7**5427	**0.785**22
8	0.81309	**0.785**52
9	**0.7**6046	**0.785**31

Many extensions to vector case (cf. survey by [Brezinski, 1977]).
We will focus on **Minimal Polynomial Extrapolation** [Sidi et al., 1986].

Outline

- Introduction
- **Minimal Polynomial Extrapolation**
- Regularized MPE
- Numerical results

Gradient method for quadratic functions

Suppose we solve the quadratic form

$$\text{minimize} \ \frac{1}{2}(x - x^*)^T M (x - x^*)$$

in $x \in \mathbb{R}^d$ using the **gradient method**. The iterates satisfy

$$x_{k+1} := x_k - \alpha M(x_k - x^*) \quad \text{for } k = 1, \ldots$$

for some $\alpha > 0$. Removing x^* on both sides, we get

$$x_{k+1} - x^* = \underbrace{(I - \alpha M)}_{=A}(x_k - x^*)$$

which means $x_{k+1} - x^*$ follows a **vector autoregressive process**.

As in Aitken's Δ^2, we can **compute x^* exactly from the iterates x_k!**

Minimal polynomial extrapolation

Given iterates $x_k \in \mathbb{R}^d$ satisfying

$$x_{k+1} - x^* = A(x_k - x^*) = A^{k+1}(x_0 - x^*) \qquad k = 0, \ldots, N-1$$

Averaging with coefficients c_i (with unitary sum) yields

$$\sum_{i=0}^{N} c_i x_i - x^* = \underbrace{\sum_{i=0}^{N} c_i A^i}_{=p(A)} (x_0 - x^*)$$

If $p(A)$ is chosen to be the **characteristic polynomial of A** (so $p(A) = 0$),

$$\begin{aligned} \sum_{i=0}^{N} c_i x_i - x^* &= p(A)(x_0 - x^*) \\ &= 0 \end{aligned}$$

which means $x^* = \sum_{i=0}^{N} c_i x_i$.

Minimal polynomial extrapolation

Intuition

If $p(A)(x_0 - x^*) = 0$ then our estimate statisfy

$$\sum_{i=0}^{N} c_i x_i = x^*$$

So, if the residual $\|p(A)(x_0 - x^*)\|_2 \approx 0$ then

$$\sum_{i=0}^{N} c_i x_i \approx x^*$$

But . . .

- We typically **do not observe** A and of course x^*.
- How to extract c using **only the iterates** x_k?

How to get $p(x)$?

<u>Goal:</u> Find $p(x)$ such that $\|p(A)(x_0 - x^*)\|_2^2$ is **as small as possible**.

The iterate differences satisfy

$$
\begin{aligned}
x_{k+1} - x_k &= (x_{k+1} - x^*) - (x_k - x^*) \\
&= A(x_k - x^*) - (x_k - x^*) = (A - I)(x_k - x^*)
\end{aligned}
$$

hence, the averaging leads to

$$
\begin{aligned}
\sum_{i=0}^{N} c_i(x_{i+1} - x_i) &= (A - I)\sum_{i=0}^{N} c_i(x_i - x^*) \\
&= (A - I)p(A)(x_0 - x^*)
\end{aligned}
$$

We find c by minimizing $\left\|\sum_{i=0}^{N} c_i(x_{i+1} - x_i)\right\|_2$ $\approx \|p(A)(x_0 - x^*)\|_2$.

Approximate Minimal Polynomial Extrapolation

Approximate MPE.

- **Step 1:** Set $U \in \mathbb{R}^{d\times N+1}$, with $U_i = x_{i+1} - x_i$, then solve the **small linear system** in the variable $c \in \mathbb{R}^{N+1}$

$$c^* \triangleq \operatorname*{argmin}_{\mathbf{1}^T c = 1} \|Uc\|_2 \;=\; \frac{(U^TU)^{-1}\mathbf{1}}{\mathbf{1}^T(U^TU)^{-1}\mathbf{1}}$$

  Step 2: Estimate the solution

$$x^* \approx \sum_{i=0}^{N} c_i x_i.$$

- A.k.a. Eddy-Mešina method [Mešina, 1977, Eddy, 1979] or Reduced Rank Extrapolation with arbitrary k (see [Smith et al., 1987]).
- Achieves an **optimal rate** of convergence when applied to the gradient method on quadratic functions.

AMPE for Optimization

Optimization algorithms.

- For gradient descent,

$$x_{k+1} := x_k - \frac{1}{L}\nabla f(x_k)$$

which means

$$x_{k+1} - x^* := A(x_k - x^*) + O(\|x_k - x^*\|_2^2)$$

where

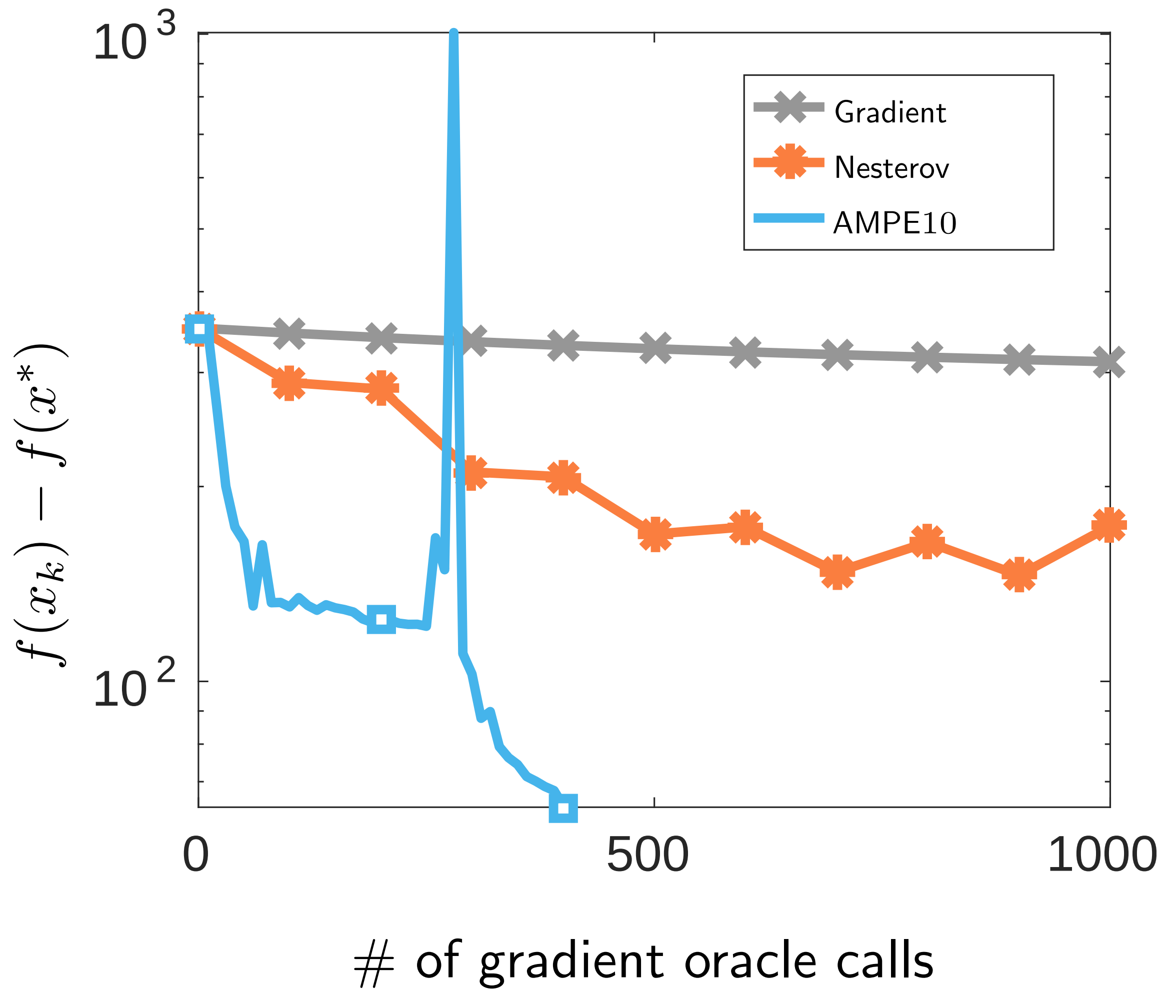
$$A = I - \frac{1}{L}\nabla^2 f(x^*).$$

so, at the first order, x_k follows an **auto-regressive process.**

- What is the impact of perturbations on the coefficients?

Numerical Results

Solving logistic regression



Morality: AMPE does **not** work on non-linear functions (in general).

AMPE stability?

Proposition

AMPE solution. *Let U the matrix of differences. The solution of AMPE is*

$$c^* \triangleq \underset{\mathbf{1}^T c = 1}{\operatorname{argmin}} \|Uc\|_2 = \frac{(U^T U)^{-1}\mathbf{1}}{\mathbf{1}^T (U^T U)^{-1}\mathbf{1}}$$

If $U^T U$ is perturbed by a matrix P, the perturbation is bounded by

$$\|(\mathbf{U}^{\mathbf{T}}\mathbf{U})^{-\mathbf{1}}\|_{\mathbf{2}} \|P\|_2 \|c^*\|_2$$

Unstable. U is close to a **Krylov matrix**, so its condition number typically grows exponentially fast with N [Tyrtyshnikov, 1994].

Regularized Minimal Polynomial Extrapolation

RMPE algorithm. Using the last k iterates and **Tikhonov regularization**.

Input: Sequence $\{x_{N-k}, x_{N-k+1}, ..., x_N\}$, parameter $\lambda > 0$

1: Form $U = [x_{N-k+1} - x_{N-k}, ..., x_N - x_{N-1}]$ $\quad O(dk)$

2: Compute $U^T U$ $\quad O(dk^2)$

3: Solve the linear system $(U^T U + \lambda I)z = \mathbf{1}$ $\quad O(k^3)$

4: Set $c = z/(z^T \mathbf{1})$

Output: Return $\sum_{i=1}^{N} c_i x_i$, approximating the optimum x^*

Algorithmic complexity. In practice, $d \gg k$ (k is typically 5 in the experiments that follow), so the complexity is $O(dk^2)$ and RMPE scales **linearly** with the dimension.

Matlab complexity. More or less 5 lines of code!

Regularized Minimal Polynomial Extrapolation

Proposition [Scieur, d'Aspremont, and Bach, 2016]

Asymptotic acceleration *Using the gradient method with stepsize in* $]0, \frac{2}{L}[$ *on a* L*-smooth,* μ*-strongly convex function* f *with Lipschitz-continuous Hessian. If* $\lambda = O(\|P\|_2)$*, then for* $\|x_0 - x^*\|$ *small enough*

$$\left\| \sum_{i=1}^{k} c_i x_{N-k+i} - x^* \right\|_2 \leq O\left(\left(1 - \sqrt{\mu/L}\right)^k \|x_0 - x^*\|_2 \right)$$

We (asymptotically) recover the accelerated rate in [Nesterov, 1983].
Nonasymptotic bounds hold as well.

Outline

- Introduction
- Minimal Polynomial Extrapolation
- Regularized MPE
- **Numerical results**

Experimental scheme

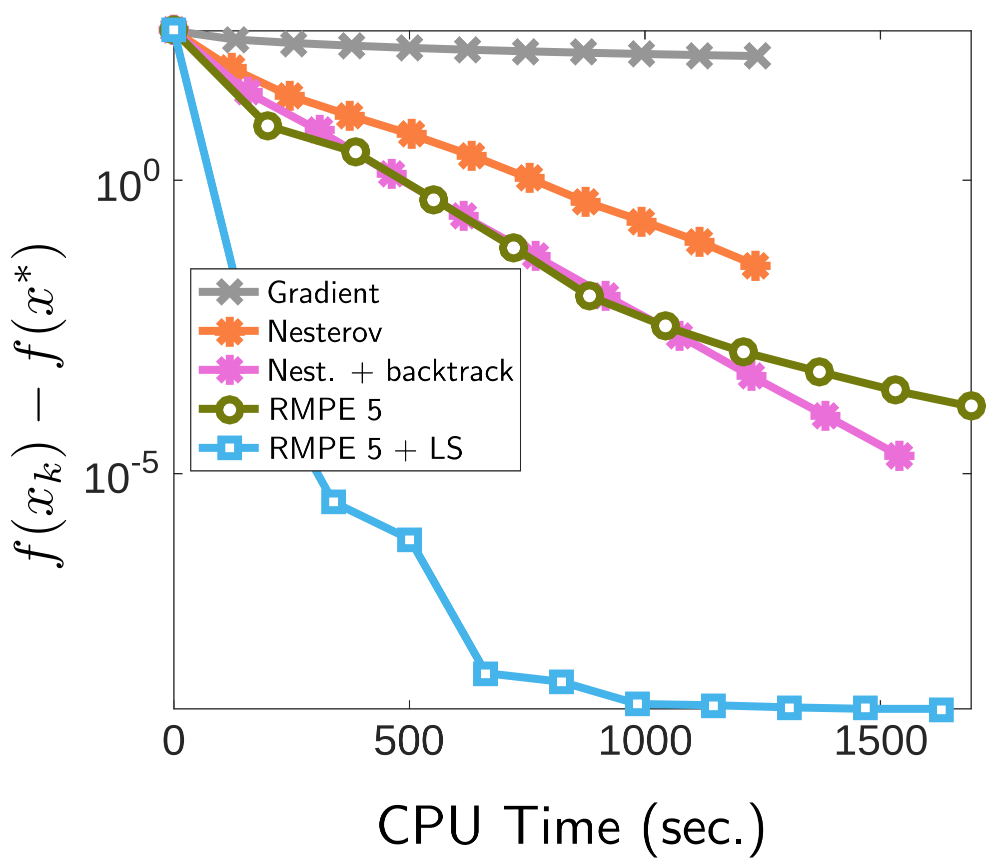
We use RMPE with restart:

- Compute k steps of gradient method (or anything else) from x_0
- Extrapolate the sequence (λ is found by grid search)
- $(Optional)$ Perform a line-search on the stepsize
- Set $x_0 =$ extrapolation, and repeat

We compare RMPE with classical optimization benchmark on various problems.

Logistic regression.

$$f(\omega) = \sum_{i=1}^{m} \log\left(1 + \exp(-y_i X_i^T \omega)\right) + \frac{\tau}{2}\|\omega\|^2$$

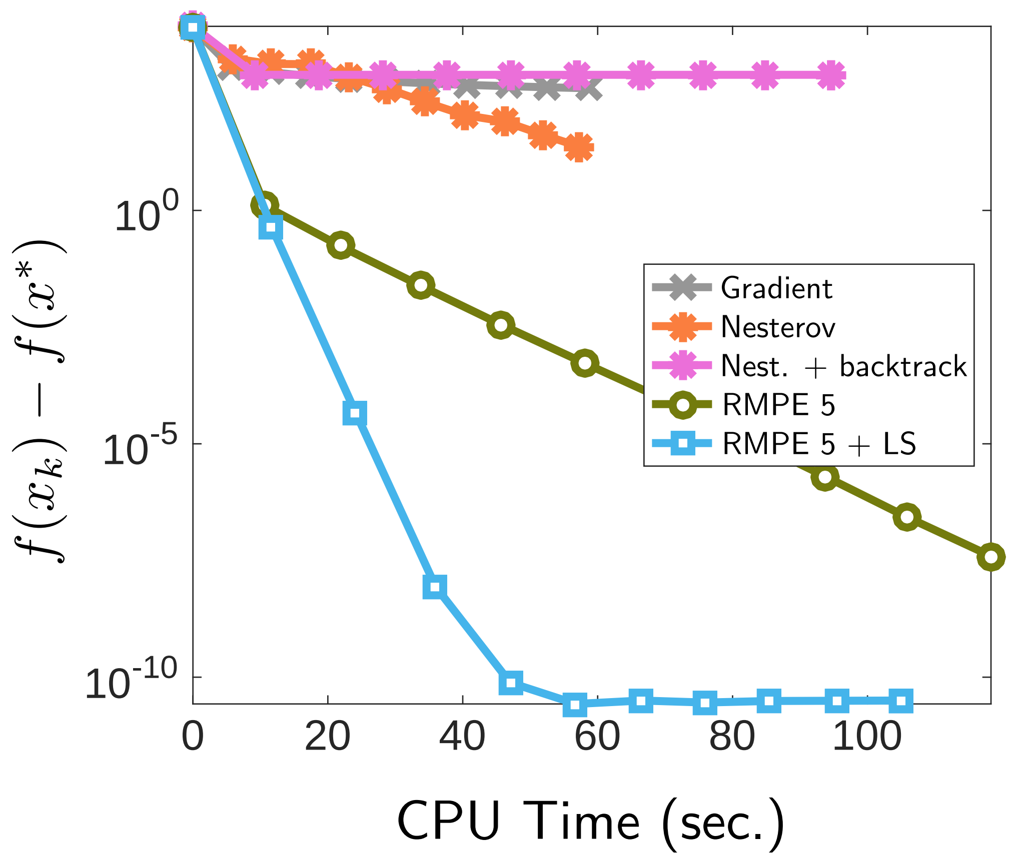


Madelon Dataset
Features: 500
Data points: 2000
Condition number: 1.2×10^9

Numerical Results

Logistic regression.

$$f(\omega) = \sum_{i=1}^{m} \log\left(1 + \exp(-y_i X_i^T \omega)\right) + \frac{\tau}{2}\|\omega\|^2$$

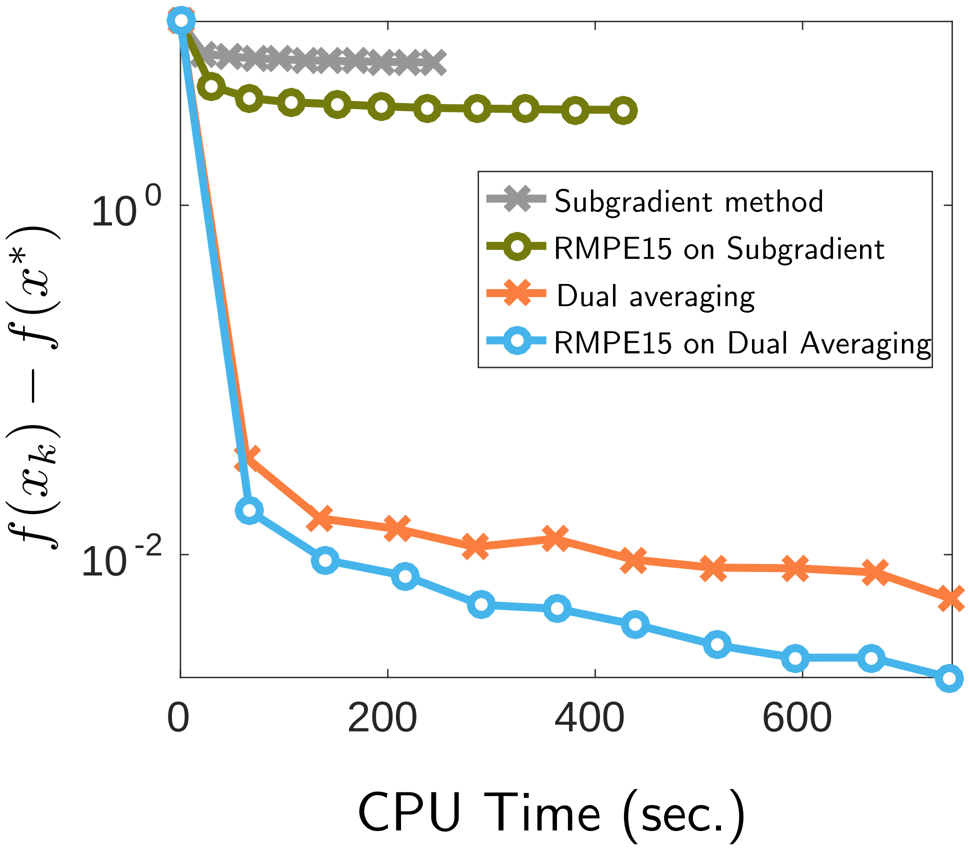


Sido0 Dataset
Features: 4932
Data points: 12678
Condition number: 1.5×10^5

Numerical Results

Dual maximum cut.

$$\min_{z} \; -\mathbf{1}^T z + \lambda_{\max}\big(\text{Laplacian}(G) + \mathbf{diag}(z)\big)$$

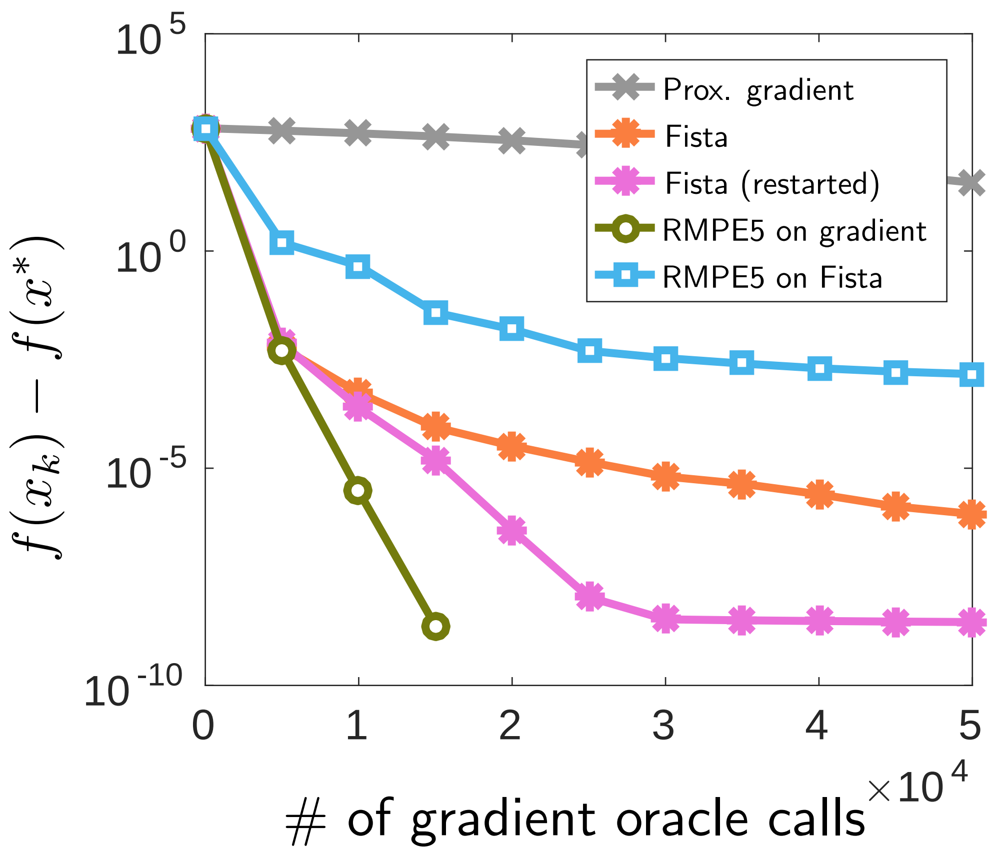


Random graph
Nodes: 200
Edges: 2000

Numerical Results

Dual SVM.

$$\min_z \frac{1}{2}\|X\,\mathbf{diag}(y)z\|_2^2 - \mathbf{1}^T z \qquad \text{s.t.} \quad 0 \leq z \leq 1$$



Artificial Dataset
Features: 200
Data points: 1000

Conclusion

Postprocessing works.

- Simple **postprocessing** step.
- Negligible additional computation cost.
- Significant convergence speedup over optimal methods.

Work in progress. . .

- Extrapolating accelerated methods.
- Constrained problems.
- Non–strongly (and non-smooth) convex functions.
- . . .

*

References

Alexander Craig Aitken. On Bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305, 1927.

C Brezinski. Accélération de la convergence en analyse numérique. *Lecture notes in mathematics*, (584), 1977.

RP Eddy. Extrapolating to the limit of a vector sequence. *Information linkage between applied mathematics and industry*, pages 387–396, 1979.

M Mešina. Convergence acceleration for the iterative solution of the equations x= ax+ f. *Computer Methods in Applied Mechanics and Engineering*, 10(2):165–173, 1977.

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

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. *arXiv preprint arXiv:1606.04133*, 2016.

Avram Sidi, William F Ford, and David A Smith. Acceleration of convergence of vector sequences. *SIAM Journal on Numerical Analysis*, 23 (1):178–196, 1986.

David A Smith, William F Ford, and Avram Sidi. Extrapolation methods for vector sequences. *SIAM review*, 29(2):199–233, 1987.

Evgenij E Tyrtyshnikov. How bad are Hankel matrices? *Numerische Mathematik*, 67(2):261–269, 1994.