Accelerating proximal gradient-type algorithms using damped Anderson acceleration with restarts and Nesterov initialization

Joint Statistical Meetings, August 4, 2026

Nicholas C. Henderson

University of Michigan, Department of Biostatistics

First-order Optimization Methods

- **Large-scale optimization** tasks are ubiquitous in modern machine learning and statistics applications.
- Stable, **first-order methods** such as gradient descent are well-suited to large-scale optimization.
  - Easy to compute
  - Computationally stable
  - Low per-iteration computational costs
  - Scalable to optimization tasks with large numbers of parameters
- Drawbacks:
  - Often slow to converge compared to second-order methods
  - Requires step-size selection

Proximal Gradient Descent

- An **objective function** $\varphi(\boldsymbol{\theta})$ in many applications may be decomposed into the sum of a **smooth** function and a simple, **nonsmooth** function.

$$\varphi(\boldsymbol{\theta}) = \underbrace{g(\boldsymbol{\theta})}_{\text{smooth}} + \underbrace{h(\boldsymbol{\theta})}_{\text{nonsmooth}} \quad ,$$

- **Proximal gradient descent** (PGD):
  - In step k, minimize the sum of h plus a quadratic approximation to g:

$$\boldsymbol{\theta}^{(k+1)} = \text{prox}_{th}\Big(\boldsymbol{\theta}^{(k)} - t\nabla g(\boldsymbol{\theta}^{(k)})\Big)$$

- The "**proximal operator**" is defined as

$$\text{prox}_{th}(\boldsymbol{\theta}) = \arg\min_{\mathbf{z}} \Big\{ th(\mathbf{z}) + \frac{1}{2}||\mathbf{z} - \boldsymbol{\theta}||^2 \Big\}.$$

Proximal Gradient Descent (L1-regularized regression)

- Objective function

$$\varphi(\boldsymbol{\beta}) = \underbrace{\frac{1}{2}||\mathbf{y} - \beta_0 - \mathbf{X}\boldsymbol{\beta}||^2}_{g} + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{h}$$

- Proximal gradient update with step size t

$$\begin{aligned}
\boldsymbol{\beta}^{(k+1)} &= \text{prox}_{th}(\boldsymbol{\beta}^{(k)} - t\nabla g(\boldsymbol{\beta}^{(k)})) \\
&= \arg\min_{\mathbf{z}} \left\{ t\lambda \sum_{j=1}^{p} |z_j| + \frac{1}{2}||\mathbf{z} - \boldsymbol{\beta}^{(k)} - t\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta}^{(k)} - \mathbf{y} - \beta_0)||^2 \right\}
\end{aligned}$$

Proximal Gradient Descent (L1-regularized regression)

- Proximal gradient update with step size t

$$\boldsymbol{\beta}^{(k+1)} = G_t(\boldsymbol{\beta}^{(k)})$$

- Components of the proximal gradient mapping $G(\boldsymbol{\beta})$

$$[G_t(\boldsymbol{\beta})]_j = \begin{cases} \beta_j + \lambda & \text{if } \beta_j < -\lambda \\ 0 & \text{if } -\lambda < \beta_j < \lambda \\ \beta_j - \lambda & \text{if } \beta_j > \lambda \end{cases}$$

Proximal Gradient Descent (Matrix Completion)

- Observed $p \times q$ matrix $\mathbf{A}$ with missing entries.
- Objective function

$$\varphi(\mathbf{Z}) = \frac{1}{2}||P_O(\mathbf{A}) - P_O(\mathbf{Z})||^2 + \lambda \sum_{k=1}^{min(p,q)} \sigma_k(\mathbf{Z}),$$

  where $\sigma_k(\mathbf{Z})$ denote the singular values of $\mathbf{Z}$
- $P_O(\mathbf{A})$ is the matrix that sets the missing indices in $\mathbf{A}$ to 0.

$$[P_O(\mathbf{A})]_{ij} = \begin{cases} A_{ij} & \text{if observed} \\ 0 & \text{otherwise} \end{cases}$$

Proximal Gradient Descent (Matrix Completion)

- Proximal gradient update with step size t

$$\mathbf{Z}^{(k+1)} = G_t(\mathbf{Z}^{(k)})$$

- The proximal gradient mapping is defined as

$$G_t(\mathbf{Z}) = D_{\lambda t}\Big(\mathbf{Z} - tP_O(\mathbf{Z}) + tP_O(\mathbf{A})\Big)$$

where $D_\lambda(\mathbf{B})$ is a "soft-threshold" SVD of $\mathbf{B}$:

$$D_\lambda(\mathbf{B}) = \mathbf{U}_B \mathbf{\Sigma}_{B,\lambda} \mathbf{V}_B^T$$

Nesterov's method/FISTA

- FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) is an **accelerated** proximal gradient method.
- Parameter updates are found by adding a direct, **Nesterov momentum** step.
- Requires essentially **no extra computation** beyond that used to perform the proximal gradient updates.

Nesterov's method/FISTA

- Generates a sequence of iterates $\theta^{(0)}, \theta^{(1)}, ...$
- Iteration k has the following steps:

1. $\boldsymbol{\theta}^{(k)} = G_t(\mathbf{y}_k)$ (Apply proximal gradient update)
2. $a_{k+1} = \left\{1 + \sqrt{1 + 4a_k^2}\right\}/2$
3. $\mathbf{y}_{k+1} = \boldsymbol{\theta}^{(k)} + \left(\frac{a_k - 1}{a_{k+1}}\right)(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)})$.

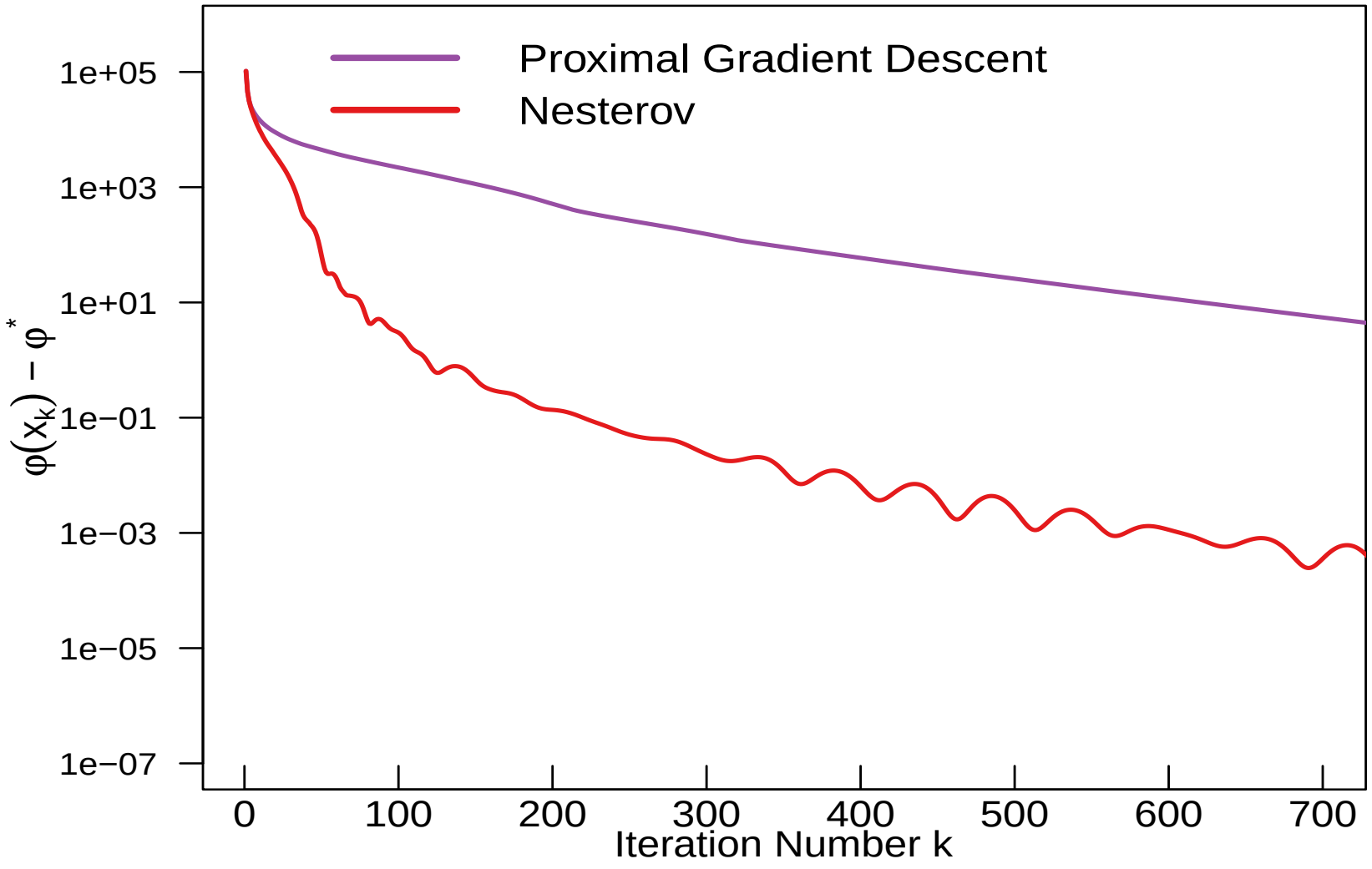
Advantages of Nesterov:

- Advantages of FISTA:
  - Easy to implement
  - No more complex than an implementation of PGD itself
  - Only requires performing a single PG step within each iteration
  - Convergence guarantees
  - Robust, global convergence guarantees
  - Fast initial descent

Algorithm Restarts

- **Early iterations** in Nesterov frequently exhibit fast progress.
- This is especially true for proximal gradient algorithms with **sparse** parameter updates.
- Typically, Nesterov will begin to exhibit **non-monotonicity** and **wave-like oscillations** after running for a while.
- It can help to **"restart"** Nesterov when this occurs (Giselsson and Boyd (2014)).

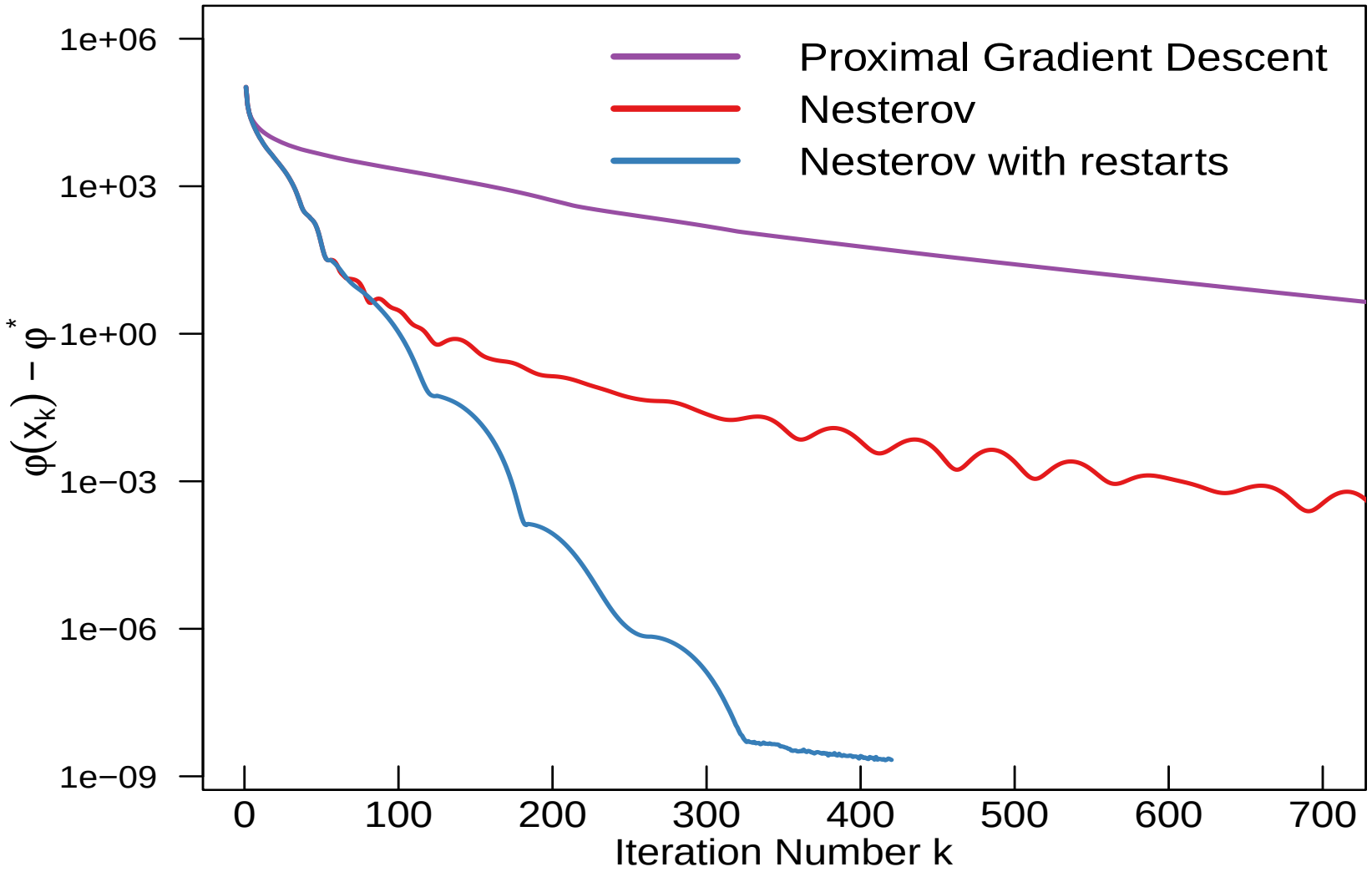
Oscillatory Behavior



Algorithm Restarts

1. $\boldsymbol{\theta}^{(k)} = G_t(\mathbf{y}_k)$ (Apply proximal gradient update)
2. **IF** $\varphi(\boldsymbol{\theta}^{(k)}) > \varphi(\boldsymbol{\theta}^{(k-1)})$
   - Set $\mathbf{y}_{k+1} = \boldsymbol{\theta}^{(k)}$ and go to **next iteration**. (no momentum)
3. **ELSE**
   - $a_{k+1} = \left\{1 + \sqrt{1 + 4a_k^2}\right\}/2$
   - $\mathbf{y}_{k+1} = \boldsymbol{\theta}^{(k)} + \left(\frac{a_k - 1}{a_{k+1}}\right)(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)})$

Nesterov Acceleration with Restarts



Anderson Acceleration

- While adding restarts to Nesterov acceleration does improve convergence speed, it is based on **1-step memory** with a fixed, scalar momentum term.
- Multi-secant methods such as **Anderson acceleration** use higher order memory.
- Uses a **linear combination** of the m_k most recent iterates.
- **Order-**m_k Anderson acceleration

$$\boldsymbol{\theta}^{(k+1)} = \sum_{j=0}^{m_k} \alpha_{jk} G_t(\boldsymbol{\theta}^{(k-j)})$$

Advantages of Anderson Acceleration

- Nesterov acceleration has strong **global convergence** guarantees.
  - $O(1/k^2)$ convergence rate from any starting value.
- However, Anderson acceleration typically has substantially faster **local convergence** once iterates get close to the solution θ^*.

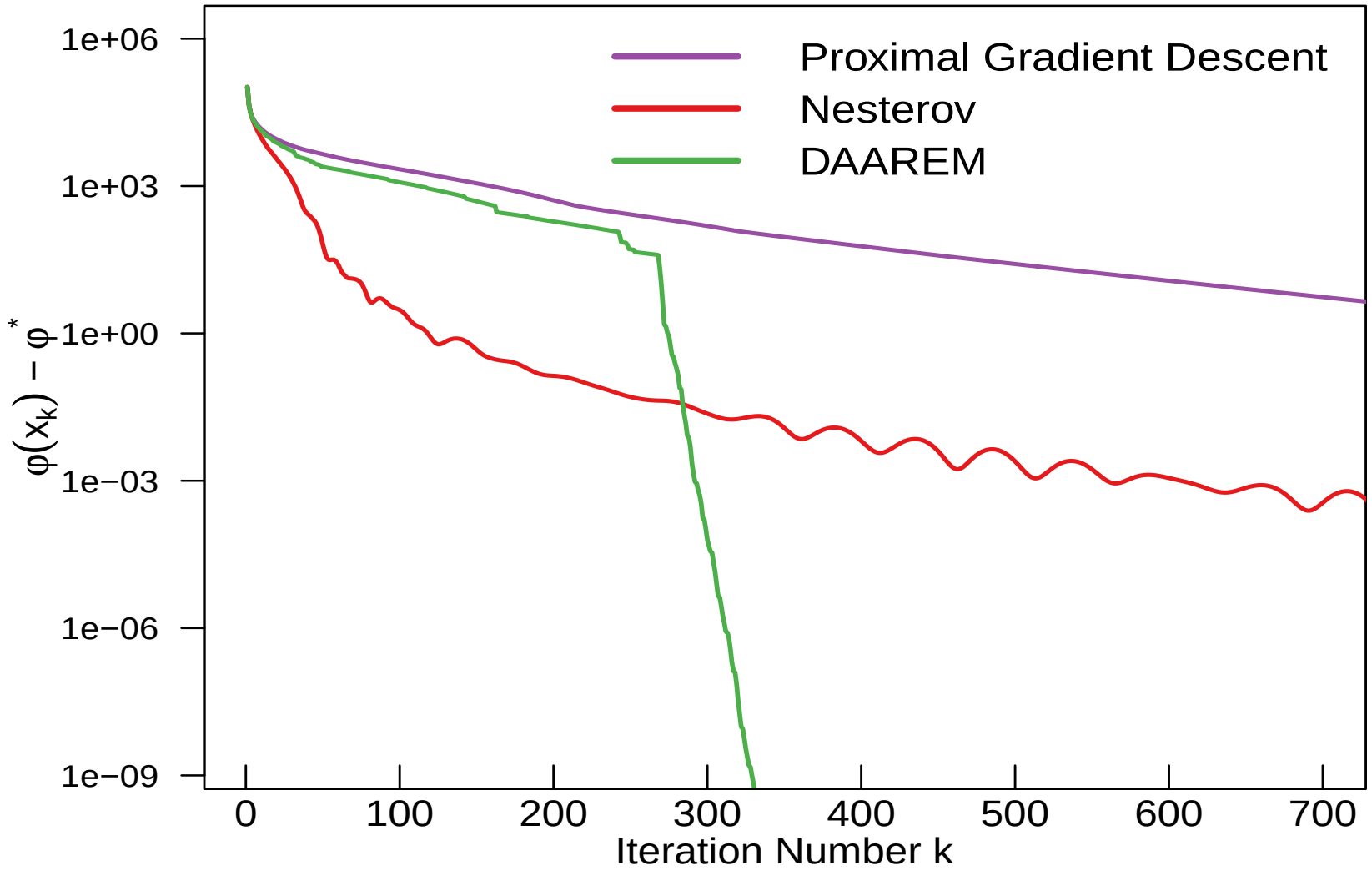
DAAREM (Damped Anderson Acceleration with Restarts and Monotonicity Control)

- Adaptation of Anderson acceleration designed specifically for EM/MM algorithms.

$$\boldsymbol{\theta}^{(k+1)} = \sum_{j=0}^{m_k} \alpha_{jk}(\lambda) G_t(\boldsymbol{\theta}^{(k-j)})$$

- $\alpha_{jk}(\lambda)$: L2-regularized coefficient estimates
- DAAREM adds systematic **algorithm restarts**.
- DAAREM monitors if iterates substantially worsen the objective.
  - If an iterate worsens the objective function, just take an EM or MM step instead.
  - Can follow the same strategy for PGD, if step size is chosen appropriately.

DAAREM convergence



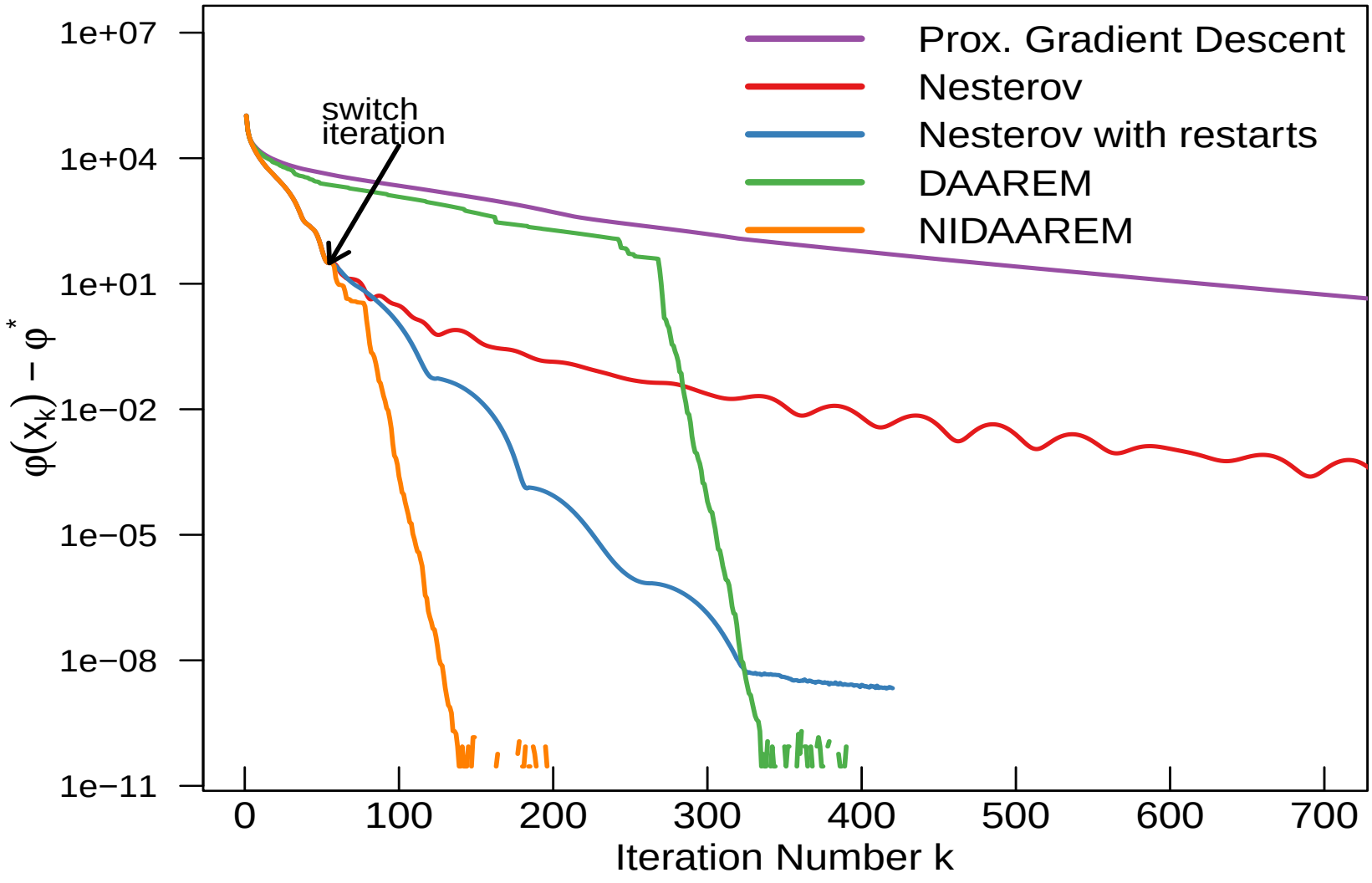
Combining Adaptive Nesterov Restarts with DAAREM

- Start by running Nesterov acceleration.
- At each iteration, check whether or not the following **switching criterion** is satisfied

$$\varphi(\boldsymbol{\theta}^{(k)}) > \varphi(\boldsymbol{\theta}^{(k-1)})$$

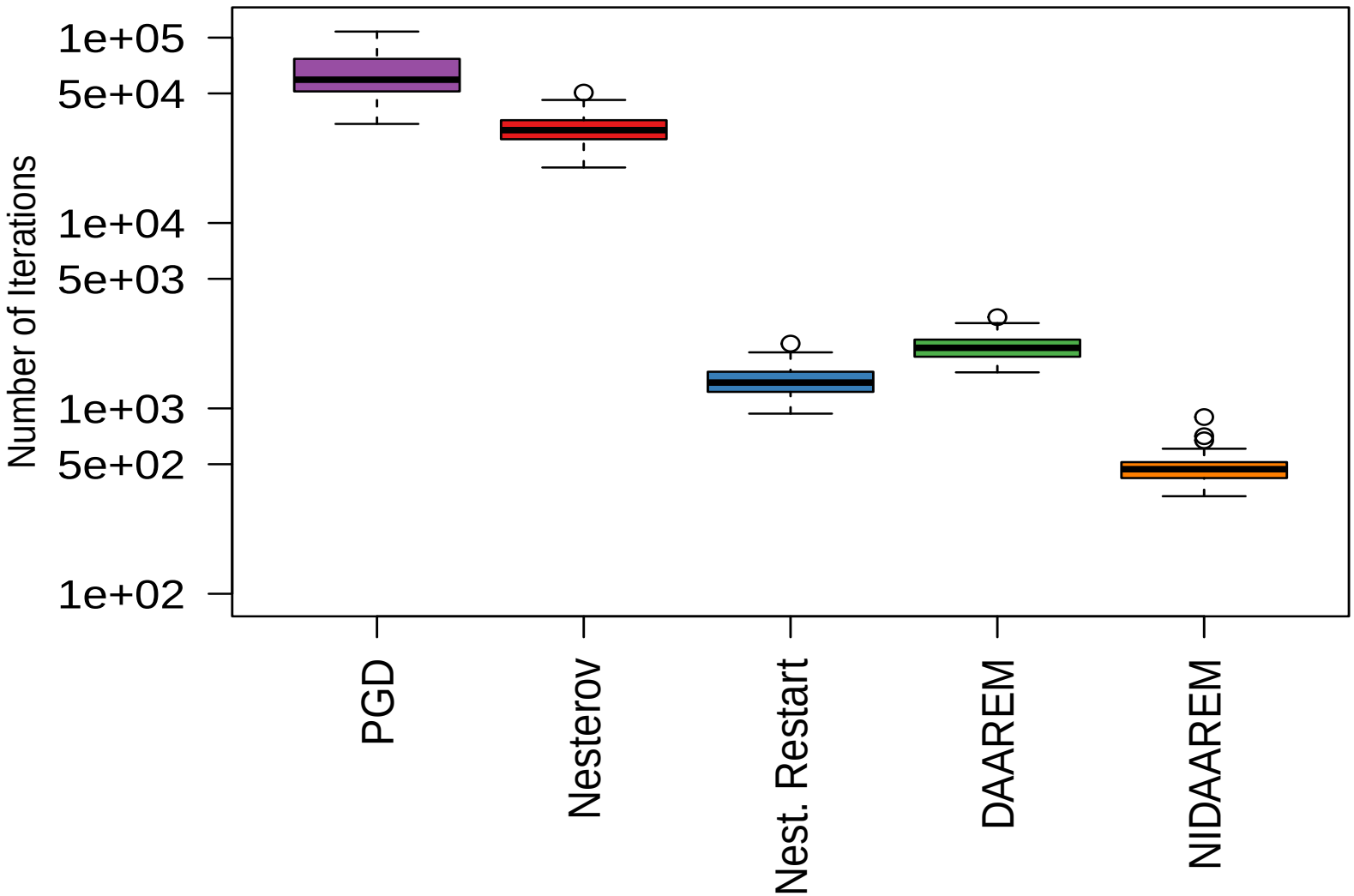
- **IF** switching criterion is satisfied
  - Run DAAREM until convergence
- **ELSE** if switching criterion is **not** satisfied
  - Continue to run Nesterov acceleration

Nesterov Initialized DAAREM (NIDAAREM)



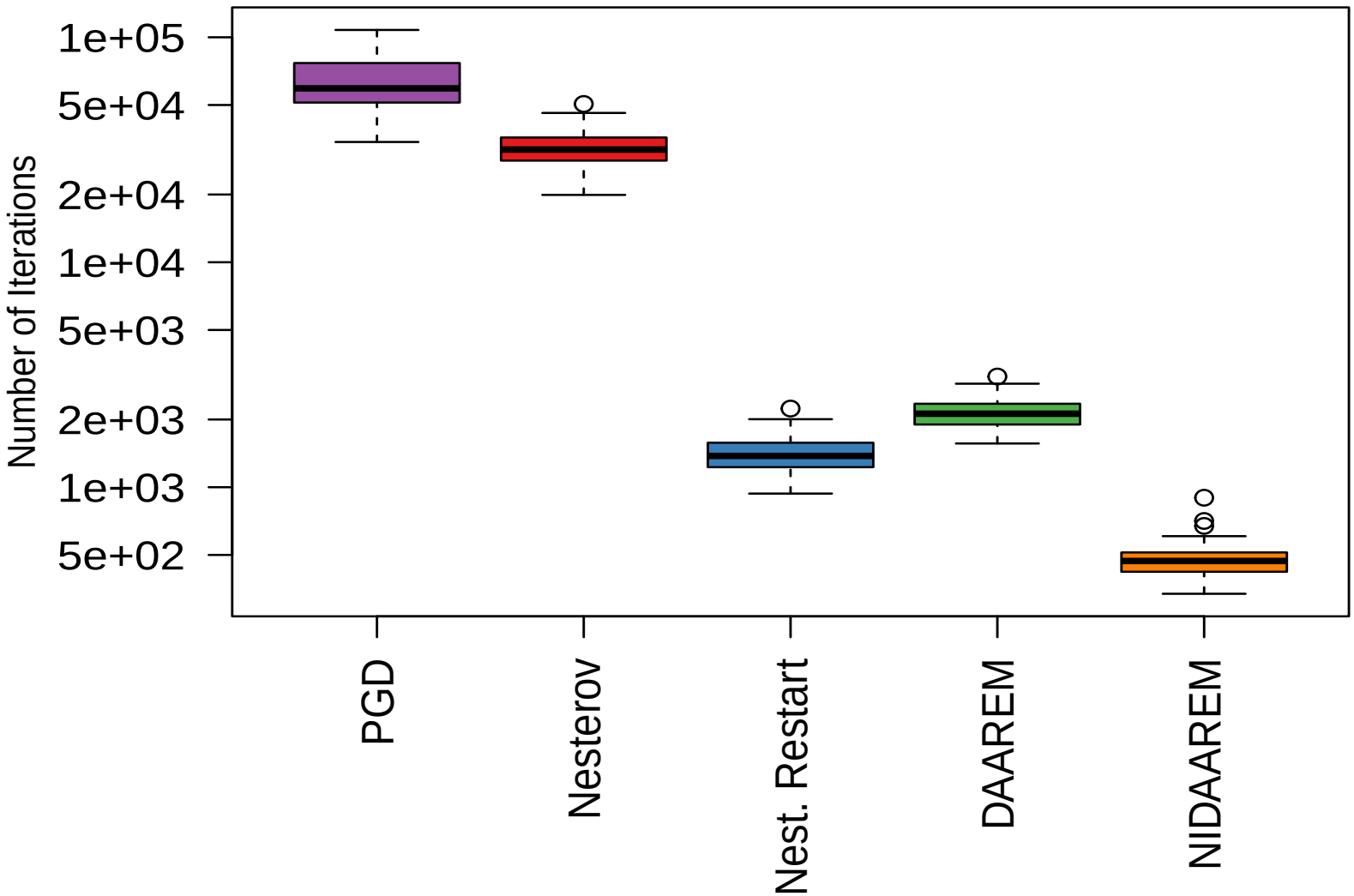
Simulations (L1 Penalized Regression)

- $n = 2,000$, $p = 500$
- Note that y-axis (number of PGD iterations) is on **log scale**



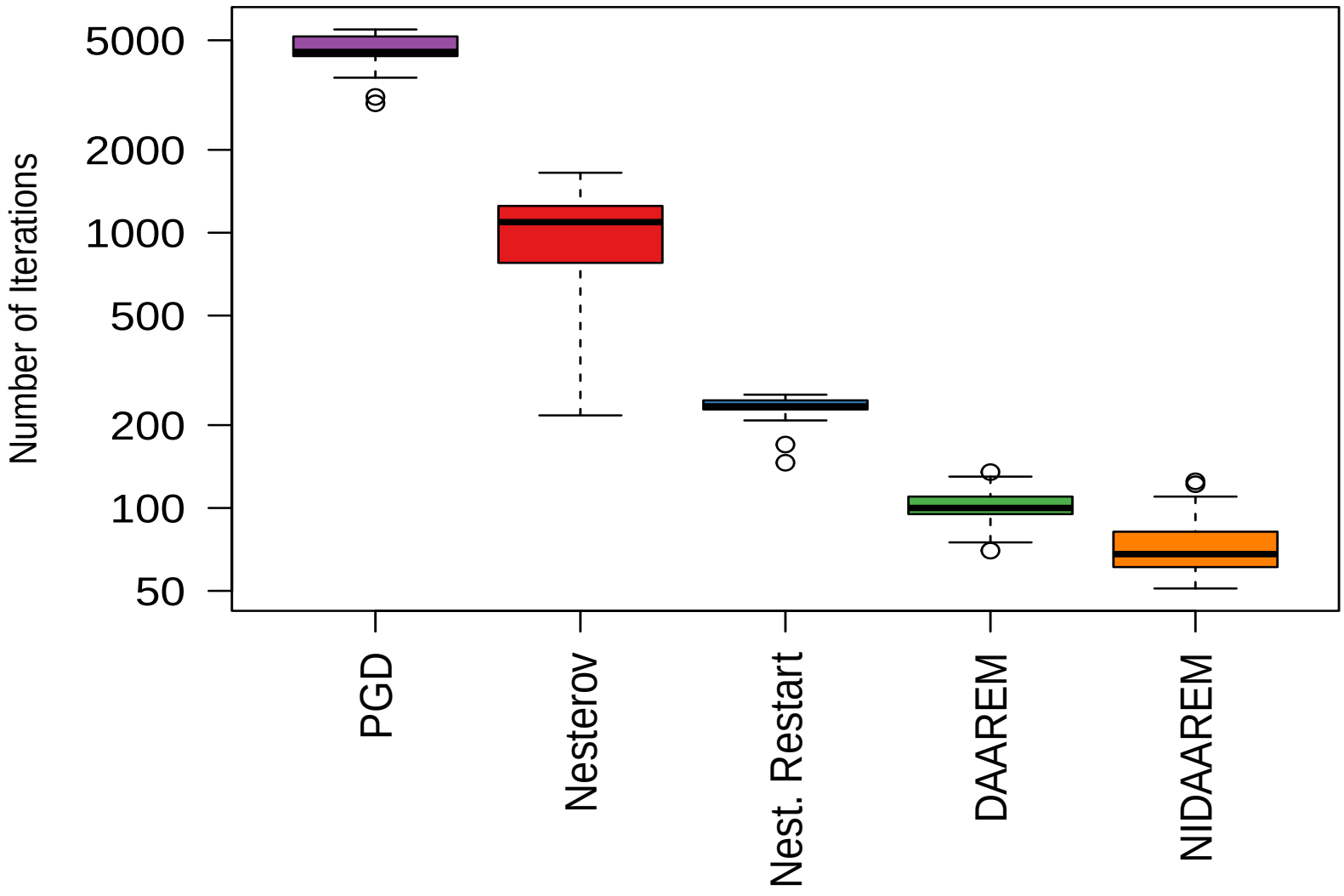
Simulations (L1 Penalized Regression)

- $n = 100$, $p = 10,000$
- Note that y-axis (number of PGD iterations) is on **log scale**



Simulations (Penalized Logistic Regression)

- $n = 2,600$ and $p = 500$ (Madelon Data)



Subsetted NIDAAREM (SNIDAAREM)

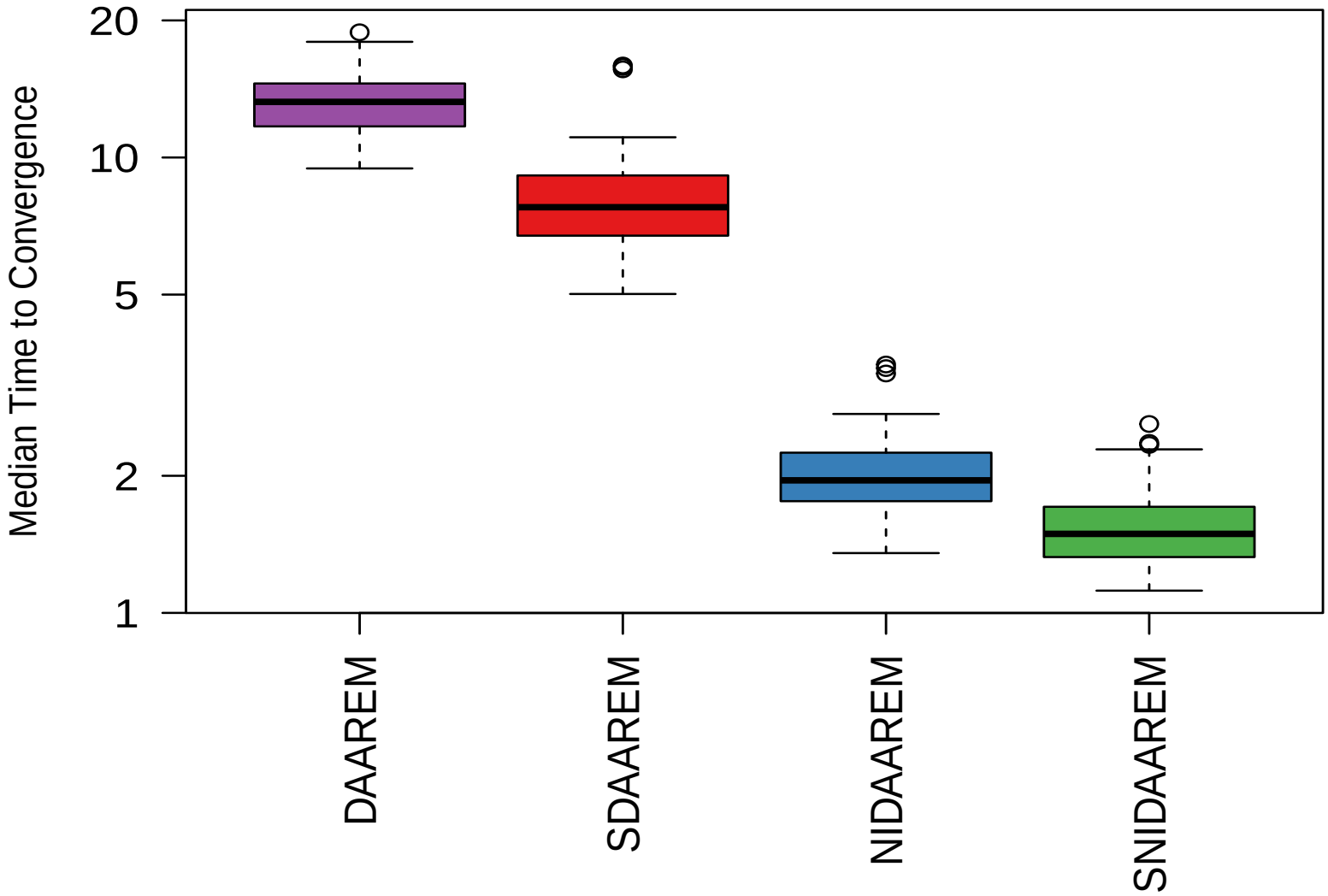
- **Sparse** proximal gradient mappings: many elements of the parameter vector are mapped directly to zero.
- With sparse PGD updates, many components of the parameter vector $\boldsymbol{\theta}^{(k)}$ are **repeatedly mapped to zero**.
- One can use this to create more **efficient implementations** of DAAREM and other variations of Anderson acceleration.
- Strategy: eliminate **many of the rows** used in the least-squares problem for finding the Anderson acceleration coefficients.
- Coefficients based on **subsetting**: subsetted NIDAAREM (SNIDAAREM).

Subsetted NIDAAREM (SNIDAAREM)

- The gains in computational time from subsetting are **often modest**.
- The gains from solving a smaller least-squares problem is relatively small compared to the computation required to perform the PGD update.
- Nevertheless, using SNIDAAREM can at least provide **noticeable gains** in very sparse problems.

Simulation (L1 Penalized Regression)

- $n = 2,000$, $p = 500$
- Note that y-axis (median timing) is on **log scale**



Beyond Proximal Gradient Descent

- Nesterov's method is **strongly connected** to gradient descent/proximal gradient descent
- Nevertheless, it can be directly used for more general iterative procedures (e.g., EM or MM algorithms).
- For the fixed-point mapping $\boldsymbol{\theta}^{(k)} = G(\boldsymbol{\theta}^{(k-1)})$, just apply momentum

$$\begin{aligned}
\boldsymbol{\theta}^{(k)} &= G(\mathbf{y}_k) \\
\mathbf{y}_{k+1} &= \boldsymbol{\theta}^{(k)} + \left(\frac{a_k - 1}{a_{k+1}}\right)(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)})
\end{aligned}$$

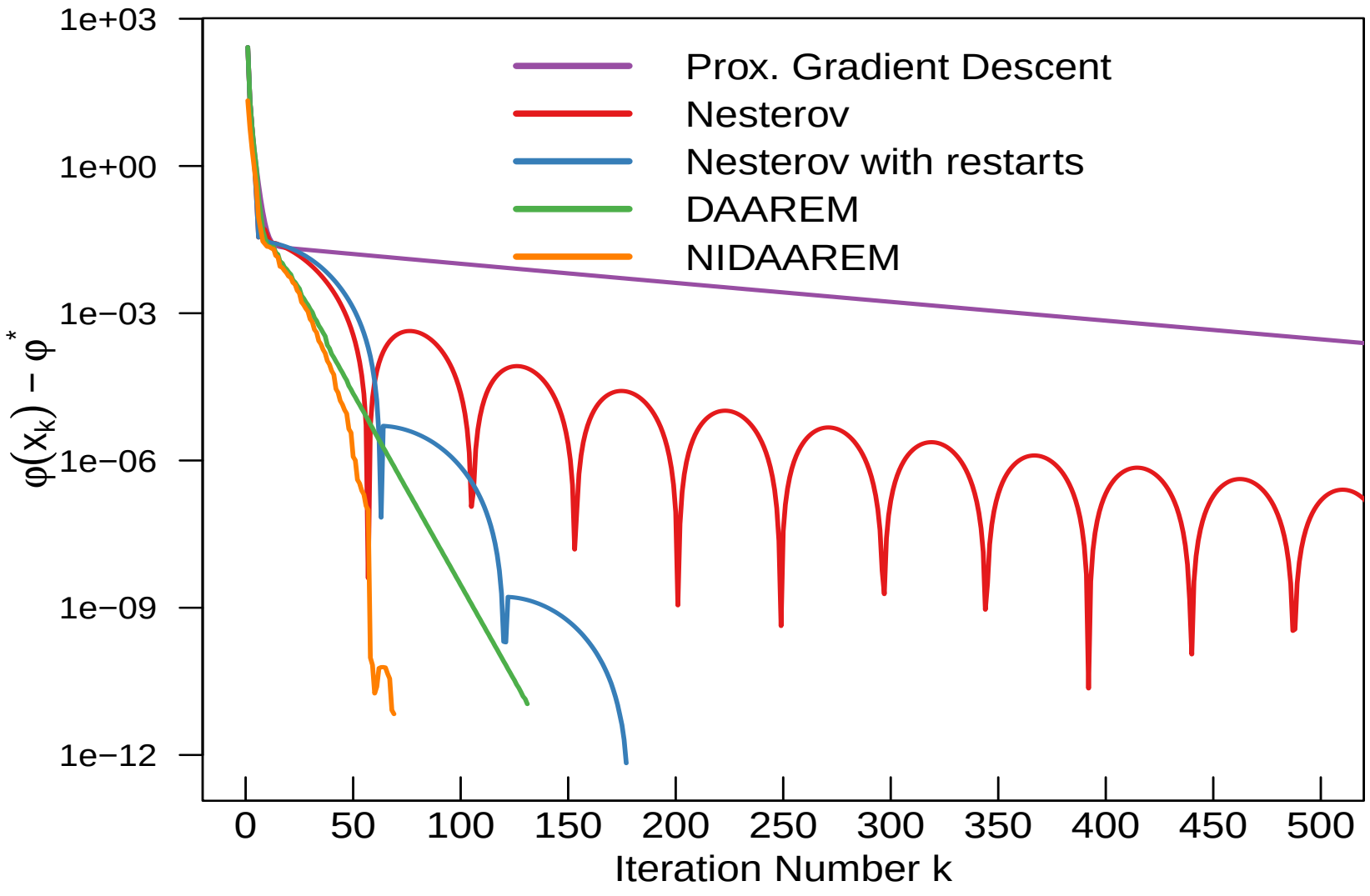
Beyond Proximal Gradient Descent

- Many fixed-point mappings have features where using Nesterov-style momentum is still effective (e.g., co-coercive operators (Tran-Dinh, 2024)).
- If the fixed-point mapping corresponds to an EM/MM algorithm, can always **"fall back"** on EM steps if Nesterov leads you in a poor direction.

Nesterov in Accelerating EM (Poisson Mixtures)

Metric	EM	SQUAREM	DAAREM	pEM	Quasi-Newton	Nesterov
fpevals	2238 ± 273	77 ± 19.1	$\mathbf{53 \pm 14.3}$	195 ± 101	131 ± 144	166 ± 23.5
elapsed	32.8 ± 6.34	$\mathbf{2.82 \pm 4.62}$	5.4 ± 4.28	5.49 ± 4.96	5.01 ± 5.82	3.82 ± 3.45
# failures	0	0	0	0	1	0

NIDAAREM in Accelerating EM (Poisson Mixtures)



Conclusions

- **NIDAAREM** is a **hybrid procedure** for accelerating proximal gradient descent.
- Nesterov acceleration: often has **fast early descent** with subsequent oscillatory behavior and slower convergence.
- DAAREM and other variations of Anderson acceleration often exhibit a **longer "plateau" phase** before delivering **fast local convergence**.
- We assumed PGD used a **fixed steplength**. Extensions of NIDAAREM to **adaptive** steplength choices would be interesting.
- Further evaluations of NIDAAREM for accelerating **EM and MM algorithms** would be worthwhile.

Resources

- ArXiv paper describing **NIDAAREM** in more detail:
  - **"Accelerating Proximal Gradient-type Algorithms using Damped Anderson Acceleration with Restarts and Nesterov Initialization"**. https://arxiv.org/abs/2508.12177
- **nidaarem** R package:
  https://github.com/nchenderson/nidaarem

References

- Tran-Dinh, Q. "From Halpern's fixed-point iterations to Nesterov's accelerated interpretations for root-finding problems". *Computational Optimization and Applications* **87**, (2024): 181-218.
- Giselsson, P. and Boyd, S. "Monotonicity and restart in fast gradient methods." In *53rd IEEE Conference on Decision and Control*, pp. 5058-5063. IEEE, 2014.