University of Colorado **Boulder**

High-dimensional DFO: Stochastic Subspace Descent and improvements

Stephen Becker, University of Colorado Boulder

Collaborators: Alireza Doostan, Nuojin Chen, Killian Wood, Drona Khurana (CU)
David Kozak, Luis Tenorio (Colorado School of Mines)

Slides will be posted at https://stephenbeckr.github.io/papers/#talks

ICCOPT, July 2025, USC (Los Angeles)

Outline

Outline and learning objectives of this talk

- Introduce a class of 0th order optimization methods, “SSD”
    - survey our prior analysis results

- Argue that **stepsize** selection is a key issue
    - suggest two ideas to help
    - show some ML **examples** where these methods make sense

- Discuss ongoing directions

Motivation: 0th order methods / derivative-free optimization

$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$ where we do not have access to $\nabla f(\boldsymbol{x})$

Traditional applications: PDE constrained optimization, when the **adjoint state method** or **automatic differentiation** is inapplicable

- e.g., multiphysics codes with complicated adjoints (and hard to code in HPC); memory issues in AD for time-dependent problems, etc.

Motivation: 0th order methods / derivative-free optimization

$\min_{\boldsymbol{x} \in \mathbb{R}^d} \ f(\boldsymbol{x})$ where we do not have access to $\nabla f(\boldsymbol{x})$

Traditional applications: PDE constrained optimization, when the **adjoint state method** or **automatic differentiation** is inapplicable

- e.g., multiphysics codes with complicated adjoints (and hard to code in HPC); memory issues in AD for time-dependent problems, etc.

Machine learning applications

- hyper parameter tuning
- black-box attacks, e.g., adversarial attacks on a model (attacker doesn't have access to source code)

Motivation: 0th order methods / derivative-free optimization

$\min_{\boldsymbol{x}\in\mathbb{R}^d} \; f(\boldsymbol{x})$ where we do not have access to $\nabla f(\boldsymbol{x})$

Traditional applications: PDE constrained optimization, when the **adjoint state method** or **automatic differentiation** is inapplicable

- e.g., multiphysics codes with complicated adjoints (and hard to code in HPC); memory issues in AD for time-dependent problems, etc.

Machine learning applications

- hyper parameter tuning
- black-box attacks, e.g., adversarial attacks on a model (attacker doesn't have access to source code)

Recent theme of my work: exploit **multi-fidelity** models

- in applied math, for PDE simulations, we often have several physics based models (with different approximations), different discretizations, different numerical precisions, reduced-order models, etc

- **machine learning** also has examples of multi-fidelity models

Context

High dimensions,
low accuracy

Low dimensions,
high accuracy

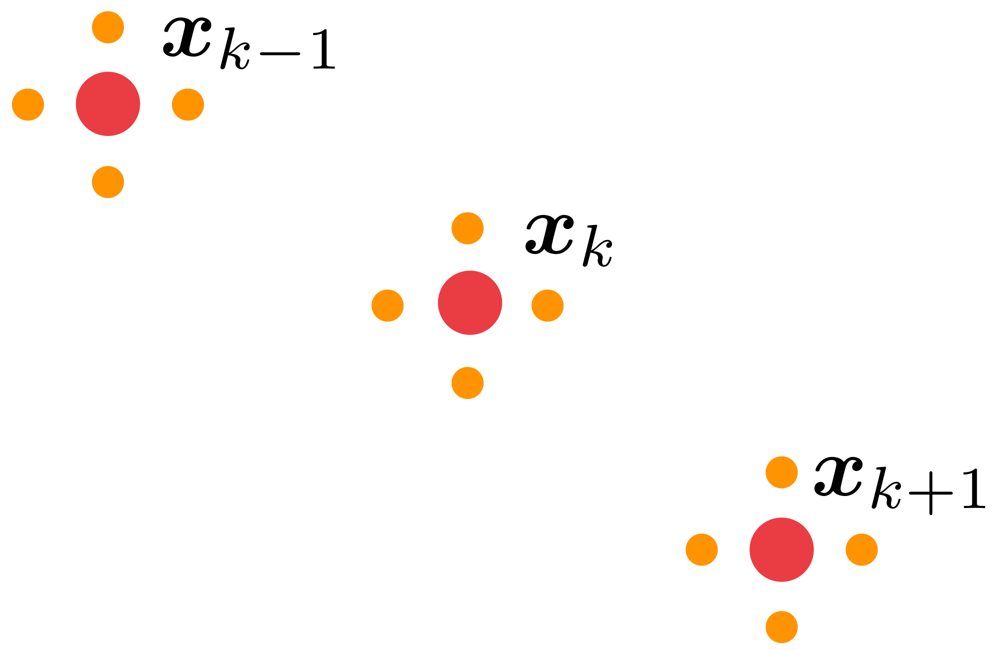
Local methods

"zeroth order": approximate $\nabla f(\boldsymbol{x})$ (e.g., w/ finite differences)

stochastic zeroth order: approximate $\boldsymbol{g}$ such that $\mathbb{E}[\boldsymbol{g}] = \nabla f(\boldsymbol{x})$

today's talk

polling methods: Nelder-Mead, coordinate descent, etc.

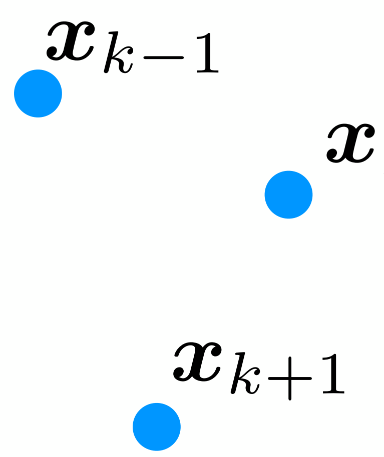


Global / model-based methods

polynomial model, minimize with a trust-region ("DFO-TR")

see our new paper "A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization", N. Cheng et al. ICML '25

Gaussian process model, use acquisition function to tradeoff exploration and exploitation ("Bayesian Optimization")



A guiding principle:

the time spent in the **optimization method** (creating and solving surrogate models)
should equal
the time spent in the **function evaluation** (e.g., solving the PDE, training neural net...)

... hence, the best method to use depends a lot on the problem.

Misc. / heuristics

genetic algorithms, particle swarm, CMA-ES, simulated annealing, etc.

A stochastic zeroth-order method: SSD

Assume we can compute/approximate $\boldsymbol{q}\boldsymbol{q}^\top \nabla f(\boldsymbol{x}) = \left(\lim_{h\to 0} \frac{f(\boldsymbol{x}+h\boldsymbol{q}) - f(\boldsymbol{x})}{h} \right) \boldsymbol{q}$ e.g., finite differences, or *forward-mode* AutoDiff

directional derivative, "two-point" estimator in ZO literature

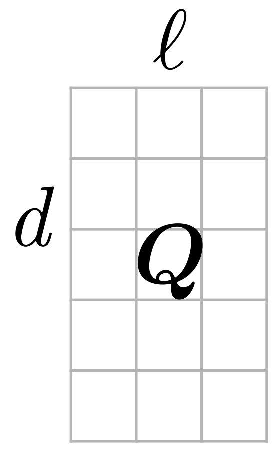
another perspective: $\boldsymbol{q}^\top \nabla f(\boldsymbol{x}) = \frac{d}{dt}\varphi(t)\Big|_{t=0}$ where $\varphi(t) = f(\boldsymbol{x}+t\boldsymbol{q})$

A stochastic zeroth-order method: SSD

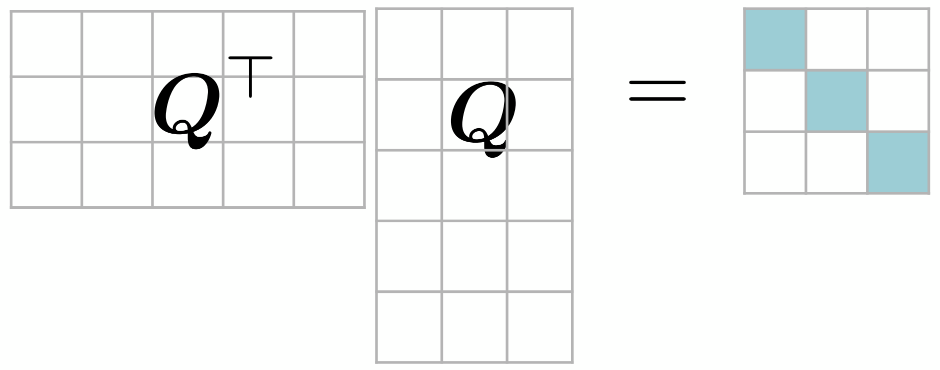
Assume we can compute/approximate $\boldsymbol{q}\boldsymbol{q}^\top \nabla f(\boldsymbol{x}) = \left(\lim_{h\to 0} \frac{f(\boldsymbol{x}+h\boldsymbol{q}) - f(\boldsymbol{x})}{h} \right) \boldsymbol{q}$

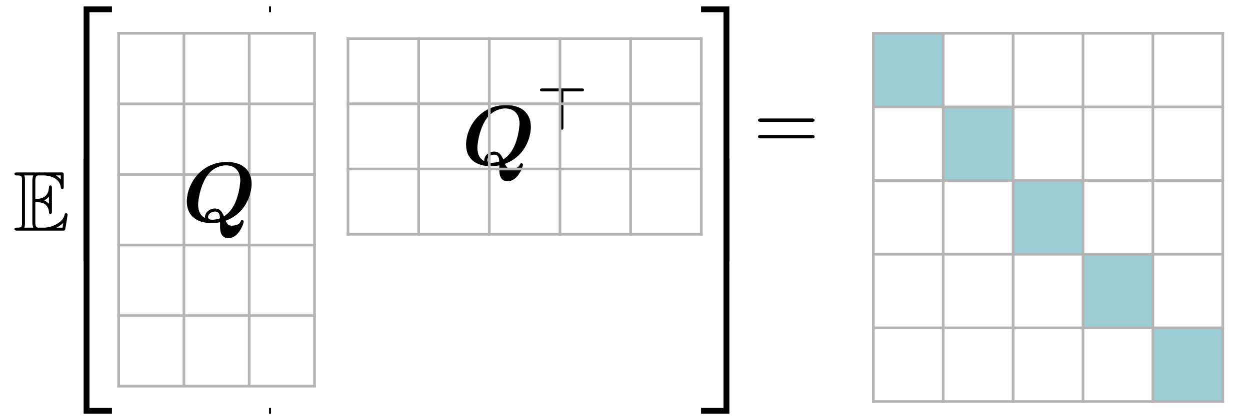
Even better, average a few copies to reduce variance

$$\boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \boldsymbol{q}_i \boldsymbol{q}_i^\top \nabla f(\boldsymbol{x}) \qquad \boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell] \qquad \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_{\ell\times\ell}, \quad \mathbb{E}\left(\frac{d}{\ell}\boldsymbol{Q}\boldsymbol{Q}^\top\right) = \boldsymbol{I}_{d\times d}$$



columns **not** independent!



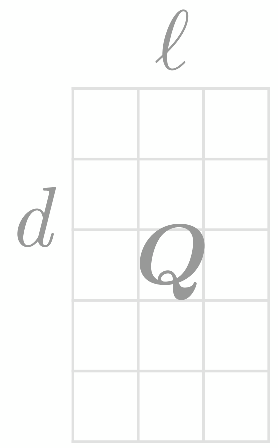


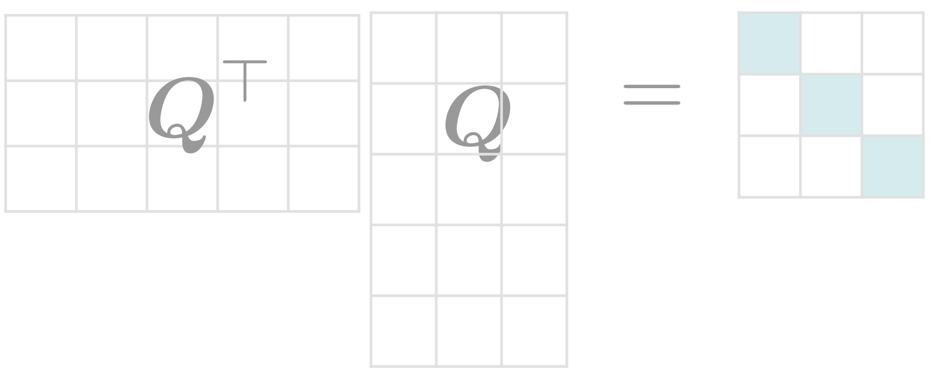
A stochastic zeroth-order method: SSD

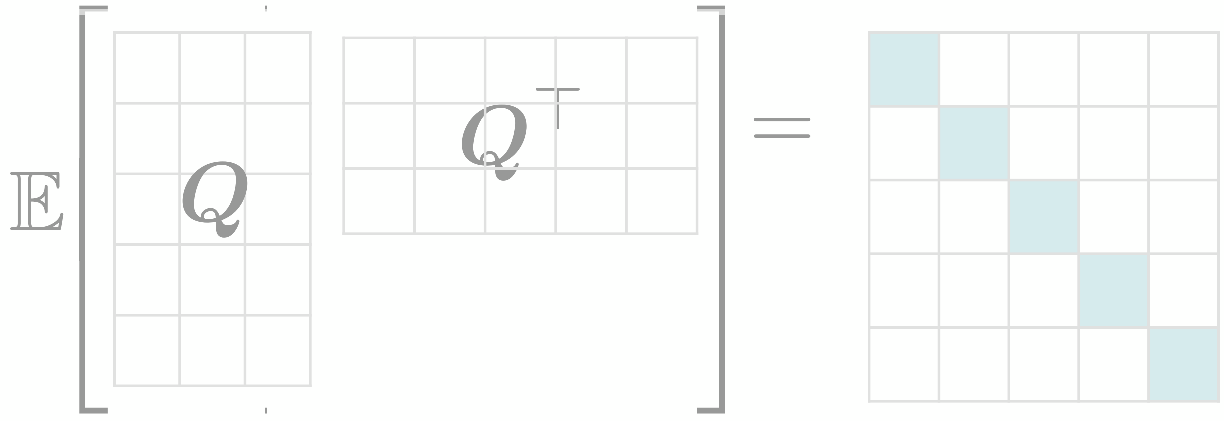
Assume we can compute/approximate $\boldsymbol{q}\boldsymbol{q}^\top \nabla f(\boldsymbol{x}) = \left(\lim_{h\to 0} \frac{f(\boldsymbol{x}+h\boldsymbol{q}) - f(\boldsymbol{x})}{h}\right)\boldsymbol{q}$

Even better, average a few copies to reduce variance

$$\boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \boldsymbol{q}_i \boldsymbol{q}_i^\top \nabla f(\boldsymbol{x}) \qquad \boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell] \qquad \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_{\ell\times\ell}, \quad \mathbb{E}\left(\frac{d}{\ell}\boldsymbol{Q}\boldsymbol{Q}^\top\right) = \boldsymbol{I}_{d\times d}$$







Algorithm: stochastic subspace descent (SSD)

Repeat:

Draw a $\boldsymbol{Q}$

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \frac{d}{\ell} \boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{X})$$

$\mathcal{O}(\ell)$ oracle calls

stepsize

Important: draw new (independent) $\boldsymbol{Q}$ every iteration

It's a type of "stochastic gradient method" (direction is unbiased) but has much stronger guarantees due to its structure

e.g., the direction is **descent direction** with prob. 1 if $\boldsymbol{Q}$ is continuous

A stochastic zeroth-order method: SSD

Assume we can compute/approximate $\boldsymbol{q}\boldsymbol{q}^\top \nabla f(\boldsymbol{x}) = \left(\lim_{h\to 0} \frac{f(\boldsymbol{x}+h\boldsymbol{q}) - f(\boldsymbol{x})}{h} \right) \boldsymbol{q}$

e.g., finite differences, or *forward-mode* AutoDiff

directional derivative, “two-point” estimator in ZO literature

Even better, average a few copies to reduce variance

$$\boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \boldsymbol{q}_i \boldsymbol{q}_i^\top \nabla f(\boldsymbol{x}) \qquad \boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell] \qquad \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_{\ell\times\ell}, \quad \mathbb{E}\left(\frac{d}{\ell} \boldsymbol{Q}\boldsymbol{Q}^\top \right) = \boldsymbol{I}_{d\times d}$$

Algorithm: stochastic subspace descent (SSD)

Repeat:

Draw a $\boldsymbol{Q}$

$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \frac{d}{\ell} \boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{X})$

$\mathcal{O}(\ell)$ oracle calls

stepsize

Equivalent formulation… and why we call it “SSD”

Columns of $\boldsymbol{Q}$ form a basis for $\mathcal{V}$

Algorithm 1 Stochastic Subspace Descent (SSD)

```
Require: $\eta$                                                ▷ Stepsize
Require: $x_0 \in \mathbb{R}^d$                                 ▷ Initial point
for $k = 0, 1, 2, \ldots$ do
    Choose subspace $\mathcal{V}_k$ of dimension $\ell \le d$
    $g_k \leftarrow \mathrm{proj}_{\mathcal{V}_k}(\nabla f(x_k))$          ▷ Project onto subspace
    $x_{k+1} \leftarrow x_k - \eta g_k$
end for
```

Stochastic 0th-order methods

Variants have been investigated for a long time...

Under various names, e.g., “random gradient”, “random pursuit”, “directional search”, “random search”

- J. Matyas, *Random* Optimization, Automation & Remote Control, **1965**
- M. Gaviano, *Some general results on convergence of random search algorithms in minimization problems*, Towards Global Optimisation, **1975**.
- F.J. Solis and R. J-B. Wets, *Minimization by random search techniques*, Math. of Operations Research, **1981** *(no rate)*
- B. Polyak **1987**
- Yu. Ermoliev and R.J.-B. Wets, *Numerical techniques for stochastic optimization*, chapter 6, Springer-Verlag, **1988**.

For recent results (2015-2025) on zeroth-order ML, see:

- “Zeroth-order Machine Learning” AAAI tutorial 2024
  - Wotao Yin, Sijia Liu, Pin-Yu Chen
  - *https://sites.google.com/view/zo-tutorial-aaai-2024/*
- Chen et al., *DeepZero: Scaling up Zeroth-Order Optimization for Deep Model Training*, ICLR ’23
- Yu et al., *SubZero: Random Subspace Zeroth-Order Optimization for Memory-Efficient LLM Fine-Tuning*, ’24
- Liu et al., *Sparse MeZO: Less Parameters for Better Performance in Zeroth-Order LLM Fine-Tuning*, ‘24
- Zhang et al., *Revisiting Zeroth-Order Optimization for Memory-Efficient LLM Fine-Tuning: A Benchmark*, ICML ’24

Much recent work on variants, 2011—2020

- D. Leventhal and A.S. Lewis, *Randomized Hessian estimation and directional search*, Optimization (2011)
- Z. Zhang, *Scalable Derivative-Free Optimization Algorithms with Low-Dimensional Subspace Techniques* (thesis in Chinese 2011, preprint with proof sketches in English 2025, arXiv 2501.04536), “NEWUOAs”
- S. U. Stich, C. Muller, and B. Gartner, *Optimization of convex functions with random pursuit*, SIAM J. Opt. (2013)
- Yu. Nesterov, *Random gradient-free minimization of convex functions*, ’11 / Yu. Nesterov and V. Spokoiny, FoCM 2017
- P. Dvurechensky, A. Gasnikov, and A. Tiurin, *Randomized similar triangles method: A unifying framework for accelerated randomized optimization methods (coordinate descent, directional search, derivative-free method)*, arXiv:1707.08486
- P. Dvurechensky, A. Gasnikov, and E. Gorbunov, *An accelerated directional derivative method for smooth stochastic convex optimization*; arXiv:1804.02394
- S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM J. Opt. (2013)
- R. Chen and S. Wild, *Randomized derivative-free optimization of noisy convex functions*, arXiv:1507.03332 (2015).
- K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller, *Structured evolution with compact architectures for scalable policy optimization*, ICML, 2018.
- T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, *Evolution strategies as a scalable alternative to reinforcement learning*, arXiv:1703.03864 (2017).
- J. Duchi, M. Jordan, M. Wainwright, A. Wibisono, *Optimal Rates for Zero-Order Convex Optimization: The Power of Two Function Evaluations*, IEEE Trans Info Theory (2015)
- A. S. Berahas, L. Cao, K. Choromanski, K. Scheinberg, *A Theoretical and Empirical Comparison of Gradient Approximations in Derivative-Free Optimization*, 2019 (published in FoCM ’22)
- F. Hanzely, K. Mishchenko, P. Richtarik, *SEGA: Variance Reduction via Gradient Sketching*, NeurIPS 2018
- *cousin of “direct search” methods, cf.* S. Gratton, C. W. Royer, L. N. Vicente, Z. Zhang, *Direct Search Based on Probabilistic Descent*, SIAM J. Opt. (2015)

Since 2020...

- D. Kozak, **S. B.**, A. Doostan, and L. Tenorio. *A stochastic subspace approach to gradient-free optimization in high dimensions.* COAP, 2021.
- D. Kozak, C. Molinari, L. Rosasco, L. Tenorio, S. Villa. *Zeroth-order optimization with orthogonal random directions.* Math. Program., 2023
- M. Rando, C. Molinari, L. Rosasco, S. Villa. *An optimal structured zeroth-order algorithm for nonsmooth optimization.* NeurIPS 2024.
- M. Rando, C. Molinaro, S. Villa, L. Rosasco. *Stochastic zeroth order descent with structured directions.* COAP, 2024.
- R. Nozawa, P.-L. Poirion, A. Takeda. *Zeroth-Order Random Subspace Algorithm for Non-smooth Convex Optimization*, JOTA, 2025.
- H.Q. Cai, D. McKenzie, W. Yin, Z. Zhang. *Zeroth-Order Regularized Optimization (ZORO): Approximately Sparse Gradients and Adaptive Sampling*, SIOPT, 2022

Stochastic 0th-order methods

Variants have been investigated for a long time...

Under various names, e.g., "random gradient", "random pursuit", "directional search", "random search"

- J. Matyas, *Random* Optimization, Automation & Remote Control, **1965**
- M. Gaviano, *Some general results on convergence of random search algorithms in minimization problems*, Towards Global Optimisation, **1975**.
- F.J. Solis and R. J-B. Wets, *Minimization by random search techniques*, Math. of Operations Research, **1981** *(no rate)*
- B. Polyak **1987**
- Yu. Ermoliev and R.J.-B. Wets, *Numerical techniques for stochastic optimization*, chapter 6, Springer-Verlag, **1988**.

Most literature focuses on $\ell = 1$

How is $\boldsymbol{q}$ typically chosen?

$$\|\boldsymbol{q}\|_2 = 1$$

$$\mathbb{E}[\boldsymbol{q}\boldsymbol{q}^\top] = \frac{1}{d}\boldsymbol{I}_{d\times d}$$

- spherical (or Gaussian, scaled appropriately)

or

- canonical basis vector, $\boldsymbol{q} \sim \text{uniform}[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d]$

(hence SSD reduces to randomized coordinate descent)

Much recent work on variants, 2011—2020

- D. Leventhal and A.S. Lewis, *Randomized Hessian estimation and directional search*, Optimization (2011)
- Z. Zhang, *Scalable Derivative-Free Optimization Algorithms with Low-Dimensional Subspace Techniques* (thesis in Chinese 2011, preprint with proof sketches in English 2025, arXiv 2501.04536), "NEWUOAs"
- S. U. Stich, C. Muller, and B. Gartner, *Optimization of convex functions with random pursuit*, SIAM J. Opt. (2013)
- Yu. Nesterov, *Random gradient-free minimization of convex functions*, '11 / Yu. Nesterov and V. Spokoiny, FoCM 2017
- P. Dvurechensky, A. Gasnikov, and A. Tiurin, *Randomized similar triangles method: A unifying framework for accelerated randomized optimization methods (coordinate descent, directional search, derivative-free method)*, arXiv:1707.08486
- P. Dvurechensky, A. Gasnikov, and E. Gorbunov, *An accelerated directional derivative method for smooth stochastic convex optimization*; arXiv:1804.02394
- S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM J. Opt. (2013)
- R. Chen and S. Wild, *Randomized derivative-free optimization of noisy convex functions*, arXiv:1507.03332 (2015).
- K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller, *Structured evolution with compact architectures for scalable policy optimization*, ICML, 2018.
- T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, *Evolution strategies as a scalable alternative to reinforcement learning*, arXiv:1703.03864 (2017).
- J. Duchi, M. Jordan, M. Wainwright, A. Wibisono, *Optimal Rates for Zero-Order Convex Optimization: The Power of Two Function Evaluations*, IEEE Trans Info Theory (2015)
- A. S. Berahas, L. Cao, K. Choromanski, K. Scheinberg, *A Theoretical and Empirical Comparison of Gradient Approximations in Derivative-Free Optimization*, 2019 (published in FoCM '22)
- F. Hanzely, K. Mishchenko, P. Richtarik, *SEGA: Variance Reduction via Gradient Sketching*, NeurIPS 2018
- *cousin of "direct search" methods, cf.* S. Gratton, C. W. Royer, L. N. Vicente, Z. Zhang, *Direct Search Based on Probabilistic Descent*, SIAM J. Opt. (2015)

Since 2020...

- D. Kozak, **S. B.**, A. Doostan, and L. Tenorio. *A stochastic subspace approach to gradient-free optimization in high dimensions.* COAP, 2021.
- D. Kozak, C. Molinari, L. Rosasco, L. Tenorio, S. Villa. *Zeroth-order optimization with orthogonal random directions.* Math. Program., 2023
- M. Rando, C. Molinari, L. Rosasco, S. Villa. *An optimal structured zeroth-order algorithm for nonsmooth optimization.* NeurIPS 2024.
- M. Rando, C. Molinaro, S. Villa, L. Rosasco. *Stochastic zeroth order descent with structured directions.* COAP, 2024.
- R. Nozawa, P.-L. Poirion, A. Takeda. *Zeroth-Order Random Subspace Algorithm for Non-smooth Convex Optimization*, JOTA, 2025.
- H.Q. Cai, D. McKenzie, W. Yin, Z. Zhang. *Zeroth-Order Regularized Optimization (ZORO): Approximately Sparse Gradients and Adaptive Sampling*, SIOPT, 2022

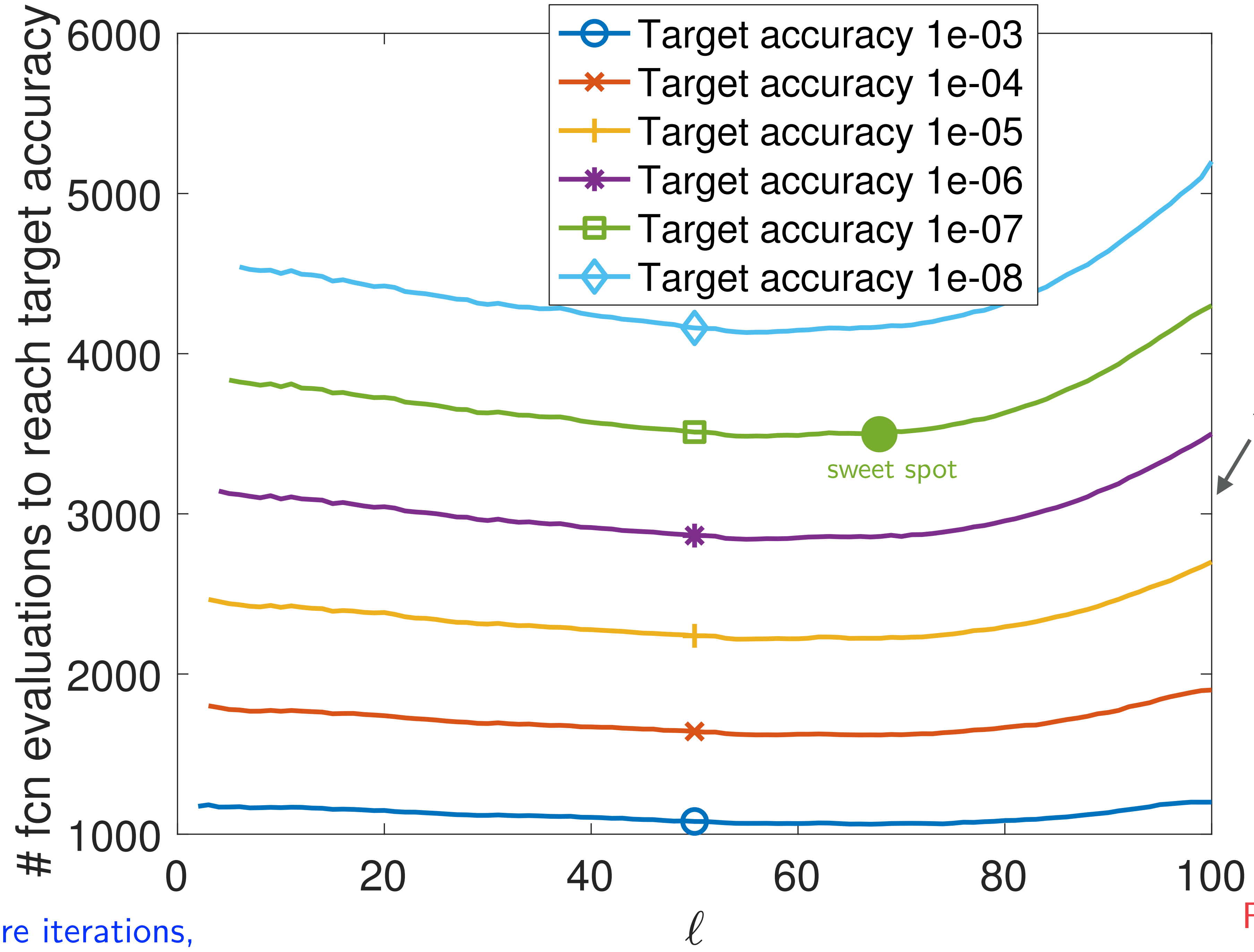
Stochastic 0th-order methods

Variants have been investigated for a long time...

Under various names, e.g., “random gradient”, “random pursuit”, “directiona

- J. Matyas, *Random Op*
- M. Gaviano, *Some ger minimization problems*
- F.J. Solis and R. J-B. Operations Research, 1
- B. Polyak **1987**
- Yu. Ermoliev and R.J.- chapter 6, Springer-Ve

Most work has focused on a single directional derivative, $\ell = 1$



100 dimensional quadratic test problem, using exact linesearch, averaged over 200 experiments

$\ell = d$ is gradient descent, and non-stochastic

More iterations, each one cheap

Fewer iterations, but each one costly

... but the optimal choice may be $1 < \ell < d$

Most literature foc

How is $\boldsymbol{q}$ typically

- spherical (or Ga

or

- canonical basis vector, $\boldsymbol{q} \sim \text{uniform}[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d]$

(hence SSD reduces to randomized coordinate descent)

Much recent work on variants, 2011—2020

- D. Leventhal and A.S. Lewis, *Randomized Hessian estimation and directional search*, Optimization (2011)
- Z. Zhang, *Scalable Derivative-Free Optimization Algorithms with Low-Dimensional Subspace Techniques* (thesis in Chinese 2011, preprint with proof sketches in English 2025, arXiv 2501.04536), “NEWUOAs”
- *with random pursuit*, SIAM J. Opt. (2013)
- Yu. Nesterov and V. Spokoiny, FoCM 2017
- *od: A unifying framework for accelerated ive-free method)*, arXiv:1707.08486
- *tive method for smooth stochastic convex*
- *stochastic programming*, SIAM J. Opt. (2013)
- *nctions*, arXiv:1507.03332 (2015).
- *ured evolution with compact architectures for*
- *scalable alternative to reinforcement learning,*
- *Convex Optimization: The Power of Two*
- *cal Comparison of Gradient Approximations in*
- *Sketching*, NeurIPS 2018
- Zhang, *Direct Search Based on Probabilistic*
- *gradient-free optimization in high dimensions.*
- *with orthogonal random directions*. Math.
- *er algorithm for nonsmooth optimization*. NeurIPS
- M. Rando, C. Molinaro, S. Villa, L. Rosasco. *Stochastic zeroth order descent with structured directions*. COAP, 2024.
- R. Nozawa, P.-L. Poirion, A. Takeda. *Zeroth-Order Random Subspace Algorithm for Non-smooth Convex Optimization*, JOTA, 2025.
- H.Q. Cai, D. McKenzie, W. Yin, Z. Zhang. *Zeroth-Order Regularized Optimization (ZORO): Approximately Sparse Gradients and Adaptive Sampling*, SIOPT, 2022

Our first analysis

Theorem (Kozak, Becker, Tenorio, Doostan '20)
Assume: minimizer attained, gradient Lipschitz, stepsize η_k chosen appropriately.

1. If f is **convex**,

$$\mathbb{E}f(\boldsymbol{x}_k) - f^\star \leq 2\frac{d}{\ell}\frac{L}{k}R^2 = \mathcal{O}(k^{-1})$$

2. If f is **not necessarily convex** but satisfies the **Polyak-Lojasiewicz** inequality,

$$\mathbb{E}f(\boldsymbol{x}_k) - f^\star \leq \rho^k(f(\boldsymbol{x}_0) - f^\star) = \mathcal{O}(\rho^k) \quad \text{and} \quad f(\boldsymbol{x}_k) \xrightarrow{\text{a.s.}} f^\star$$

3. If f is **strongly convex**, statements of 2 above hold, and also

$$\boldsymbol{x}_k \xrightarrow{\text{a.s.}} \operatorname{argmin}_{\boldsymbol{x}} f(\boldsymbol{x})$$

4. If f is **not convex** (nor PL),

$$\min_{k' \in \{0,\ldots,k\}} \mathbb{E}\|\nabla f(\boldsymbol{x}_{k'})\|^2 \leq \frac{d}{\ell}\frac{2L(f(\boldsymbol{x}_0) - f^\star)}{k+1}$$

Generic SSD

$$\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}_{\ell\times\ell}, \quad \mathbb{E}\left(\frac{d}{\ell}\boldsymbol{Q}\boldsymbol{Q}^\top\right) = \boldsymbol{I}_{d\times d}$$

David Kozak

$f^\star \overset{\text{def}}{=} \min_{\boldsymbol{x}} f(\boldsymbol{x})$

$\rho = 1 - \frac{\mu}{L}\frac{\ell}{d}$

d = ambient dimension

ℓ = # directional derivs

$\frac{d}{\ell} = 1$ is gradient descent

μ is PL constant

L is gradient Lipschitz constant

$\eta = \frac{\ell}{d}\frac{1}{L}$ is stepsize

Our first analysis

Our analysis is comparable to analysis of similar algorithms

Assume f obtains its minimum and ∇f is L-Lipschitz continuous.

Theorem 1 (Kozak, Becker, Tenorio, Doostan '19, Thm. 2.4). *The SSD algorithm with stepsize* $\eta = \frac{1}{L}\frac{\ell}{d}$ *gives*

$$\mathbb{E}\, f(x_k) - f^\star \le \boxed{2\frac{d}{\ell}\frac{L}{k}R^2}$$

where

$$R = \sup_{x | f(x) \le f(x_0)} \inf_{x^\star \in \operatorname{argmin} f} \|x - x^*\|$$

(e.g., f is coercive $\implies R < \infty$).

SSD

$1 \le \ell \le d$

Theorem 2 (Nesterov, Spokoiny '17, Thm. 8). *Take stepsize* $\eta = \frac{1}{4(d+4)L}$*, then the random gradient method with a Gaussian direction converges as*

$$\frac{1}{k}\sum_{i=0}^{k-1} \mathbb{E}\, f(x_i) - f^\star \le \boxed{\frac{4(d+4)L}{k}\|x_0 - x^\star\|^2}$$

where $x^\star$ is any optimal solution.

Gaussian

$\ell = 1$

convex, not necessarily strongly convex, scenario

Our first analysis

Our analysis is comparable to analysis of similar algorithms

Assume f obtains its minimum, ∇f is L-Lipschitz continuous, and f is μ PL or strongly convex.

Theorem 3 (Kozak, Becker, Tenorio, Doostan '19, Cor. 2.3). *The SSD algorithm with stepsize* $\eta = \frac{1}{L}\frac{\ell}{d}$ *gives*

$$\mathbb{E}\, f(x_k) - f^\star \leq \rho^k \left(f(x_0) - f^\star\right) \quad \textit{with} \quad \rho = \boxed{1 - \frac{\mu}{L}\frac{\ell}{d}}.$$

SSD

$1 \leq \ell \leq d$

Theorem 4 (Nesterov, Spokoiny '17, Thm. 8). *Take stepsize* $\eta = \frac{1}{4(d+4)L}$, *then the random gradient method with a Gaussian direction converges as*

$$\mathbb{E}\, f(x_k) - f^\star \leq \frac{L}{2}\rho^k \|x_0 - x^*\|^2 \quad \textit{with} \quad \rho = \boxed{1 - \frac{\mu}{L}\frac{1}{8(d+4)}}$$

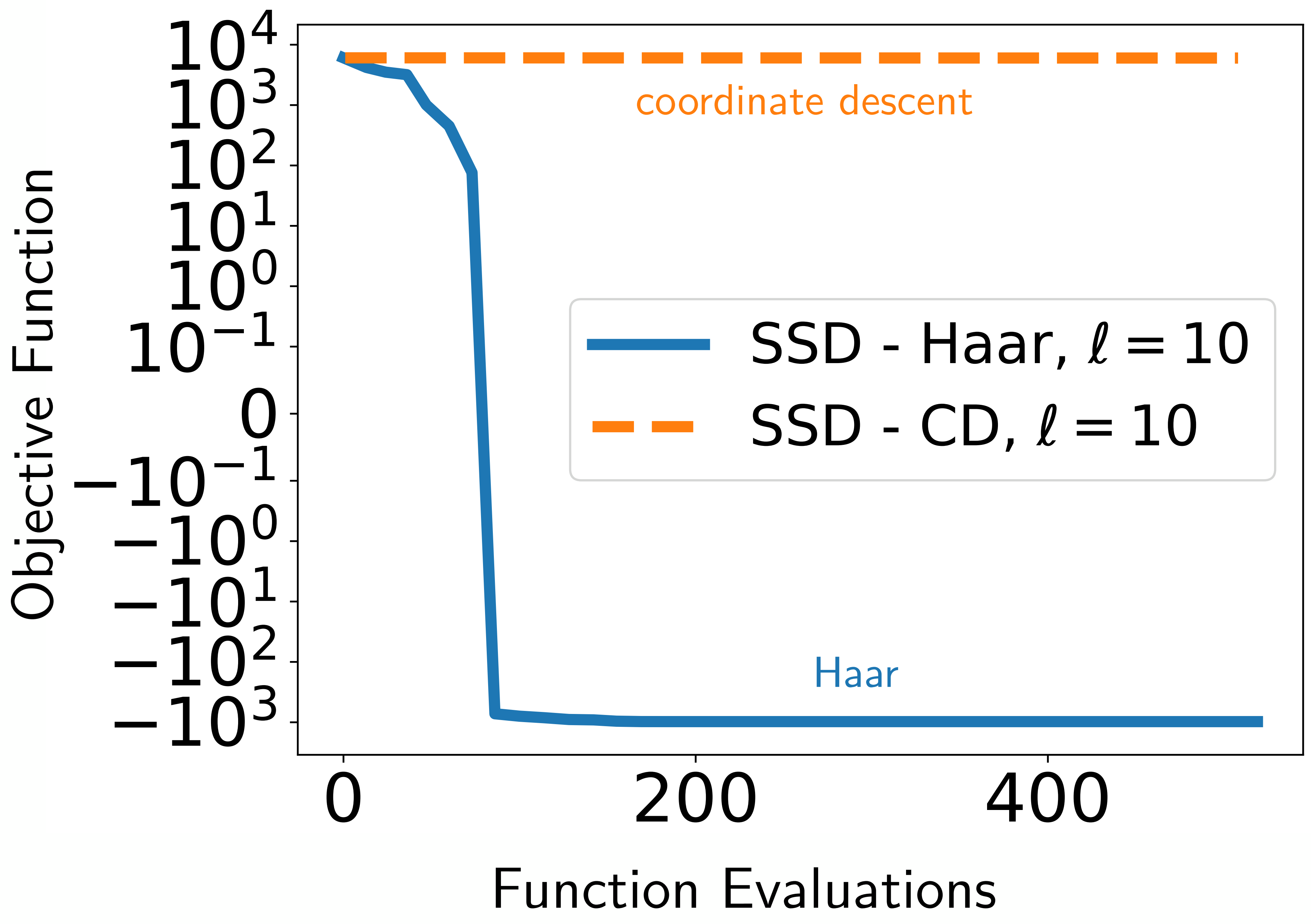
where $x^\star$ *is any optimal solution.*

Gaussian

$\ell = 1$

strongly convex or **PL scenario**

… but that theory doesn't capture the full story



Observation: sometimes SSD (with Haar) drastically outperforms randomized coordinate descent (CD)

Both of them are valid instantiations of SSD

So, the details of $\boldsymbol{Q}$ matter!

Algorithm: "Haar" SSD

Draw the random matrix $\boldsymbol{Q}$ as follows:

$\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_\ell\}$ a basis for $\operatorname{span}\{\tilde{\boldsymbol{q}}_1, \ldots, \tilde{\boldsymbol{q}}_\ell\}$ $\quad \tilde{\boldsymbol{q}}_i \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

(or any uniformly random subspace)

Equivalently, draw from the Haar distribution

Satisfies

$$\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_{\ell\times\ell}, \quad \mathbb{E}\left(\frac{d}{\ell}\boldsymbol{Q}\boldsymbol{Q}^\top\right) = \boldsymbol{I}_{d\times d}$$

but goes even further.

… but that theory doesn't capture the full story

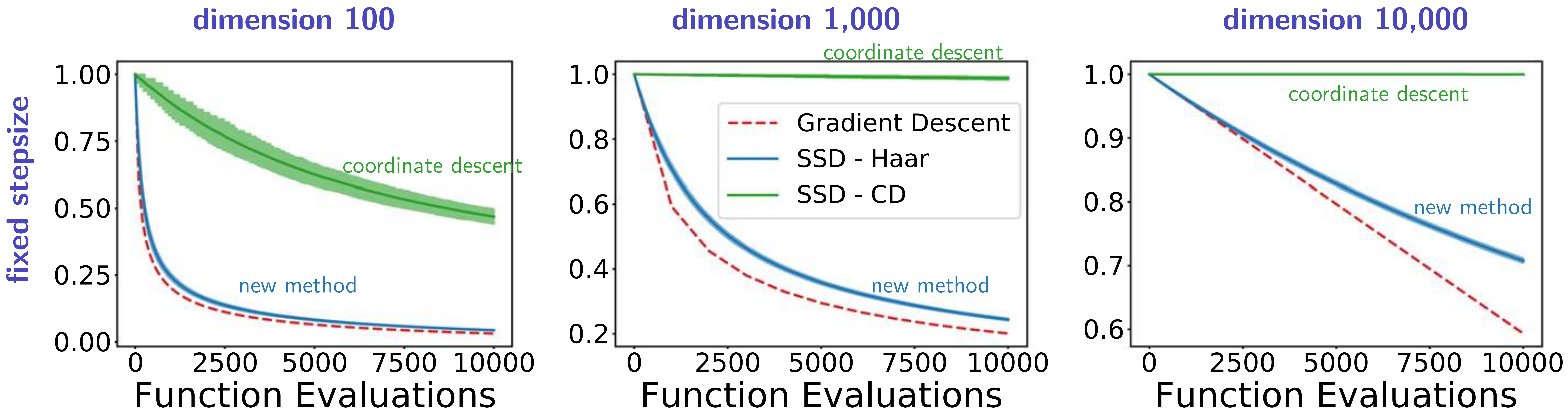
SSD drastically outperforms randomized coordinate descent (CD)

We can force it to happen by making a problem with low "intrinsic" dimension, e.g., Nesterov's "worst function in the world"

$$f_{\lambda,r}(\boldsymbol{x}) = \lambda((x_1^2 + \sum_{i=1}^{r-1}(x_i - x_{i+1})^2 + x_r^2)/2 - x_1)/4,$$

This has **intrinsic dimension** of r

$r = 20,\ \ell = 3$



Improved theory (specialized to SSD-Haar)

Recall...
$d =$ ambient dimension
$\ell = \#$ directional derivs

Tighter analysis using concentration-of-measure:

Lemma 2 (Johnson-Lindenstrauss style embedding, from Kozak, Becker, Tenorio '19, Lemma 1)**.** $\forall \epsilon \in (0,1)$, *if* $\ell \gtrsim \epsilon^{-2}$, $Q \sim \mathrm{Haar}(d \times \ell)$, *then* $\forall 0 \neq g \in \mathbb{R}^d$,

$$1 - \epsilon \leq \frac{d}{\ell} \frac{\|Q^T g\|^2}{\|g\|^2} \leq 1 + \epsilon \quad \textit{w/ prob. } \delta \geq 0.8$$

Note: *coordinate descent* style projections do **not** have similar nice embedding properties

Improved theory (specialized to SSD-Haar)

Recall...
$d =$ ambient dimension
$\ell = \#$ directional derivs

Tighter analysis using concentration-of-measure:

Lemma 2 (Johnson-Lindenstrauss style embedding, from Kozak, Becker, Tenorio '19, Lemma 1)**.** *$\forall \epsilon \in (0,1)$, if $\ell \gtrsim \epsilon^{-2}$, $Q \sim \mathrm{Haar}(d \times \ell)$, then $\forall 0 \neq g \in \mathbb{R}^d$,*

$$1 - \epsilon \leq \frac{d}{\ell}\frac{\|Q^T g\|^2}{\|g\|^2} \leq 1 + \epsilon \ \ \textit{w/ prob. } \delta \geq 0.8$$

... and putting it all together

Theorem 3 (Kozak, Becker, Tenorio '19, Thm. 1)**.** *If f is strongly convex and ∇f is Lipschitz continuous, then for an appropriate stepsize η_k, the sequence (x_k) generated by SSD (with $Q \sim$ Haar), for $k > 100$, satisfies*

$$f(x_k) - f^\star \leq (1 + (1-\epsilon)\rho)^{k/2}\,(f(x_0) - f^\star) \quad \textit{with probability } \geq 0.998,$$

where $\rho < 1$ depends on ℓ, d and the Lipschitz and strong convexity parameters.

due to possibility of failure of JL

error in JL embedding

Extension: Variance Reduction

control variate

```
**Algorithm** SVRG-style Variance Reduced SSD method, "VRSSD"
for $k = 1, 2, \ldots$ do                                   ▷ $k$ is the "epoch"
    $\bar{z} \leftarrow \nabla f(x_k)$                      ▷ Expensive, but not done often
    $w_0 \leftarrow x_k$
    for $t = 1, 2, \ldots, T$ do                            ▷ Typically $T = \mathcal{O}(d)$
        Draw $Q \sim \text{Haar}(d \times \ell)$
        $w_{t+1} \leftarrow w_t - \eta \left( \underset{\text{regular SSD term}}{\tfrac{d}{\ell} QQ^T \nabla f(w_t)} - \alpha_k \underbrace{\left( \tfrac{d}{\ell} QQ^T - I \right)}_{\text{orthogonal projection}} \bar{z} \right)$    ▷ $\alpha_k$ to be estimated
    $x_{k+1} \leftarrow w_T$
```

only use control variate in orthogonal subspace

(since we know gradient in main subspace)

Theorem 4 (Kozak, Becker, Tenorio, Doostan 2019; Thm. 2.7)**.** *If f is strongly convex and ∇f is Lipschitz continuous, then for an appropriate stepsize η_k, the sequence (x_k) generated by* <u>*VRSSD*</u> *converges almost surely to the (unique) minimizer of f and at a linear rate (the rate depends on η_k and α_k).*

⚠ We do not require the ERM structure!

$f_c(x) \approx f(x)$

generic (non-algorithmic) control variates

control variate, coarse approximation, cheap to evaluate

```
**Algorithm** Proposed coarse-model variance reduced SSD/Random-Gradient
for $k = 1, 2, \ldots$ do
    $\bar{z} \leftarrow \nabla f_c(x_k)$                    ▷ Full coarse-grid gradient
    Draw $Q \sim \text{Haar}(d \times \ell)$
    $x_{k+1} \leftarrow x_k - \eta_k \left( \tfrac{d}{\ell} QQ^T \nabla f(x_k) + \alpha_k \left( \tfrac{d}{\ell} QQ^T \bar{z} - \bar{z} \right) \right)$
```

Key idea: easy to do **orthogonal projection**

from the literature:

```
**Algorithm** SAGA (Defazio, Bach, Lacoste-Julien '14) for solving the ERM model
$\forall i = 1, \ldots, N,\ x^{(i)} \overset{\text{def}}{=} x_0$; store $\{\nabla f_i(x^{(i)})\}_{i=1}^N$ in table
for $k = 1, 2, \ldots$ do
    Draw $j \sim \text{Uniform}([1, \ldots, N])$
    $\bar{z} \leftarrow \frac{1}{N} \sum_{i=1}^N \nabla f_i(x^{(i)})$     ▷ From table
    $x_{k+1} \leftarrow x_k - \eta \left( \nabla f_j(x_k) - \nabla f_j(x^{(j)}) + \bar{z} \right)$
    Re-define $x^{(j)} \leftarrow x_k$ and update table with $\nabla f_j(x^{(j)})$
```

our variant:

```
**Algorithm** SAGA-style Variance Reduced SSD method
Pre-compute $\bar{z} \leftarrow \nabla f(x_0)$
for $k = 1, 2, \ldots$ do
    Draw $Q \sim \text{Haar}(p \times r)$
    $x_{k+1} \leftarrow x_k - \eta \left( \tfrac{d}{\ell} QQ^T \nabla f(x_k) - \tfrac{d}{\ell} QQ^T \bar{z} + \bar{z} \right)$
    $\bar{z} \leftarrow \bar{z} + QQ^T (\nabla f(x_k) - \bar{z})$    ▷ Update of $\bar{z}$ is low-memory, unlike original SAGA
```

key: update control variate in the subspace

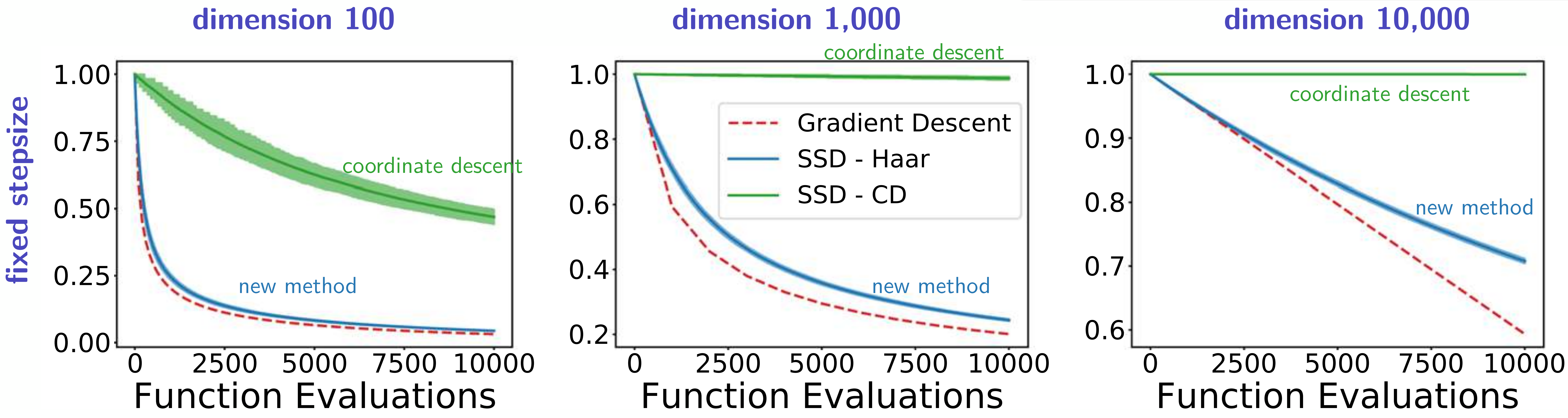
Extension: stepsize selection

$r = 20, \ \ell = 3$

Nesterov's "worst function in the world"

$$f_{\lambda,r}(\boldsymbol{x}) = \lambda((x_1^2 + \sum_{i=1}^{r-1}(x_i - x_{i+1})^2 + x_r^2)/2 - x_1)/4,$$

This has **intrinsic dimension** of r



Recall previous example

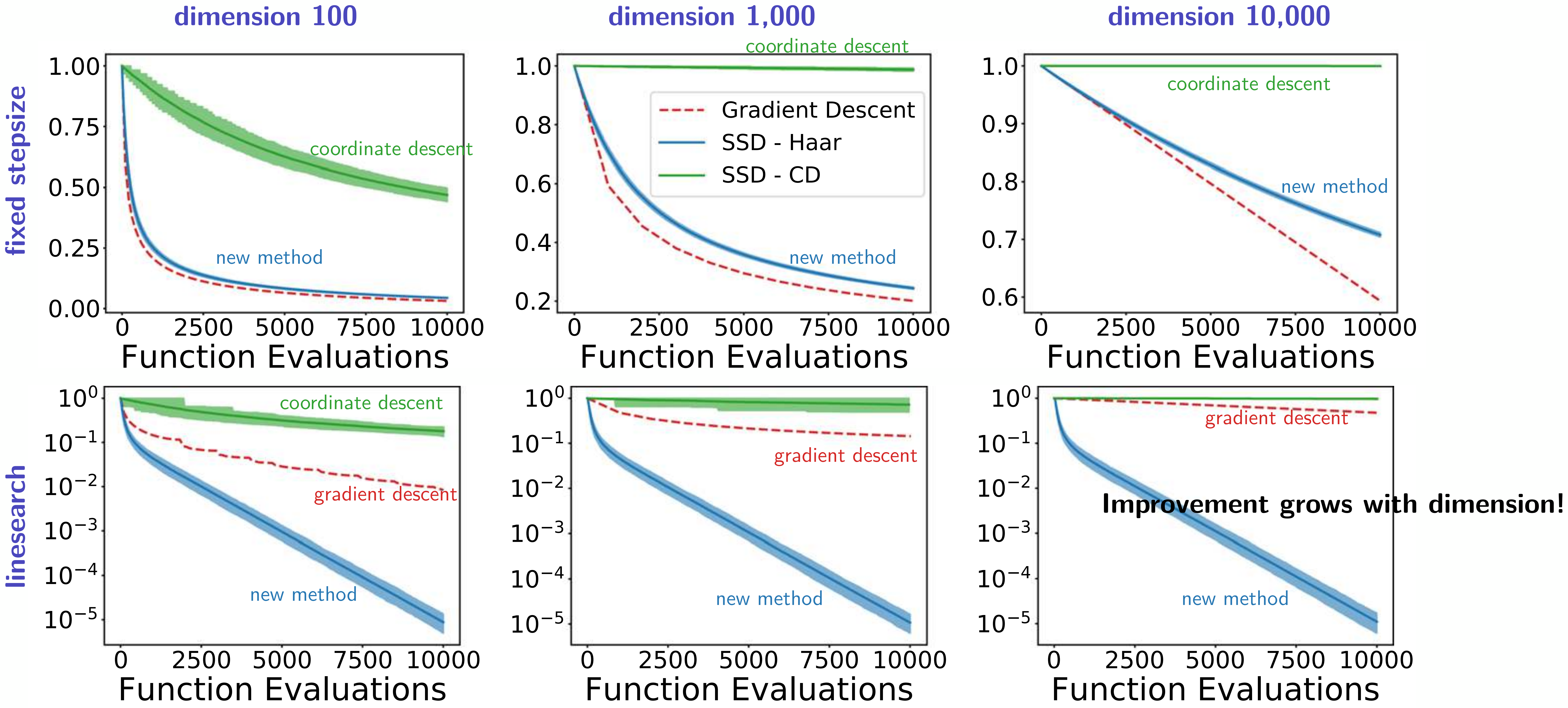
Extension: stepsize selection

$r = 20,\ \ell = 3$

Nesterov's "worst function in the world"

$$f_{\lambda,r}(\boldsymbol{x}) = \lambda((x_1^2 + \sum_{i=1}^{r-1}(x_i - x_{i+1})^2 + x_r^2)/2 - x_1)/4,$$

This has **intrinsic dimension** of r



Improvement grows with dimension!

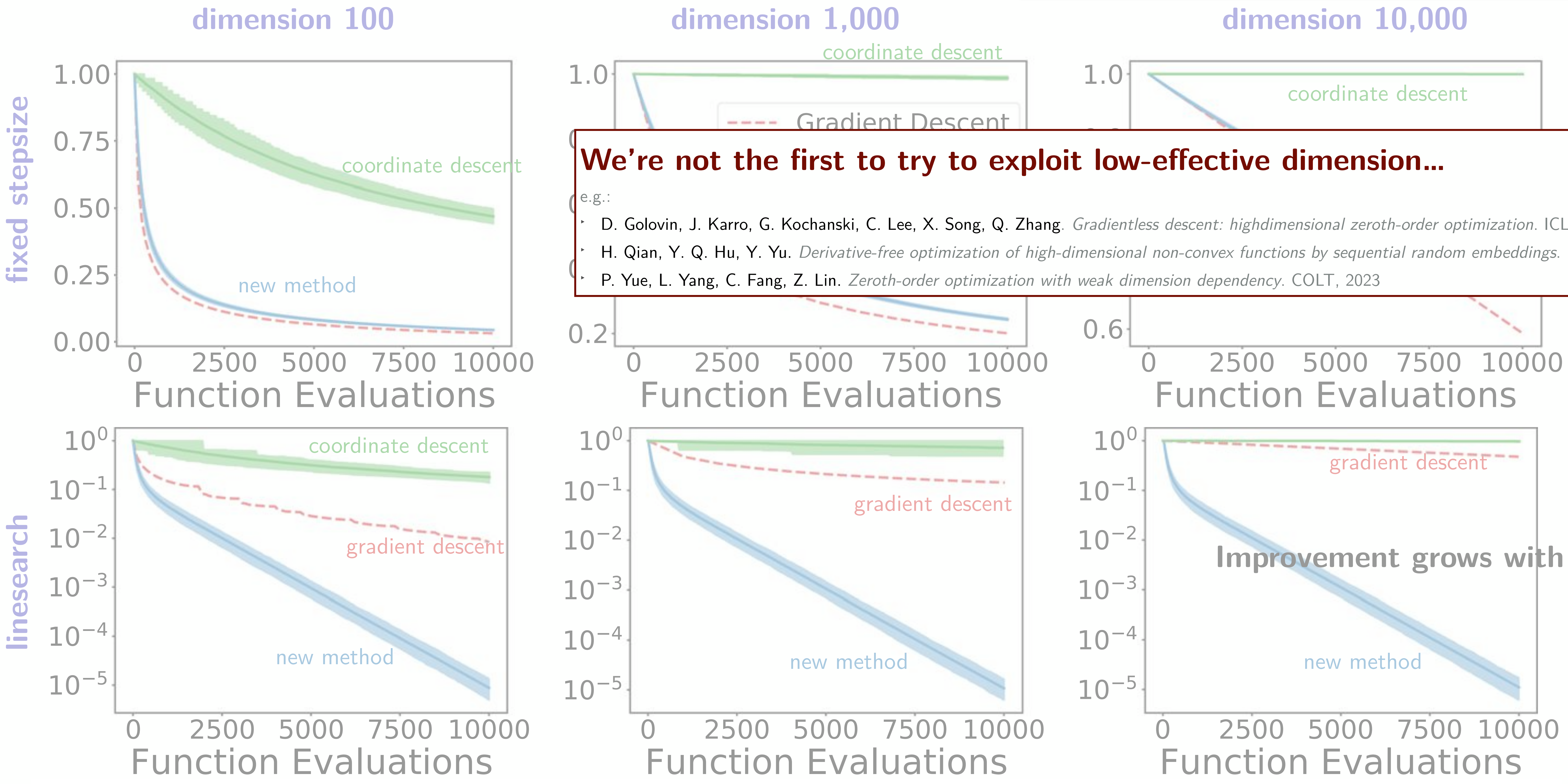
Extension: stepsize selection

$r = 20, \ \ell = 3$

Nesterov's "worst function in the world"

$$f_{\lambda,r}(\boldsymbol{x}) = \lambda((x_1^2 + \sum_{i=1}^{r-1}(x_i - x_{i+1})^2 + x_r^2)/2 - x_1)/4,$$

This has **intrinsic dimension** of r



We're not the first to try to exploit low-effective dimension...

e.g.:

- D. Golovin, J. Karro, G. Kochanski, C. Lee, X. Song, Q. Zhang. *Gradientless descent: highdimensional zeroth-order optimization.* ICLR, 2020
- H. Qian, Y. Q. Hu, Y. Yu. *Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings.* IJCAI, 2016
- P. Yue, L. Yang, C. Fang, Z. Lin. *Zeroth-order optimization with weak dimension dependency.* COLT, 2023

Improvement grows with dimension!

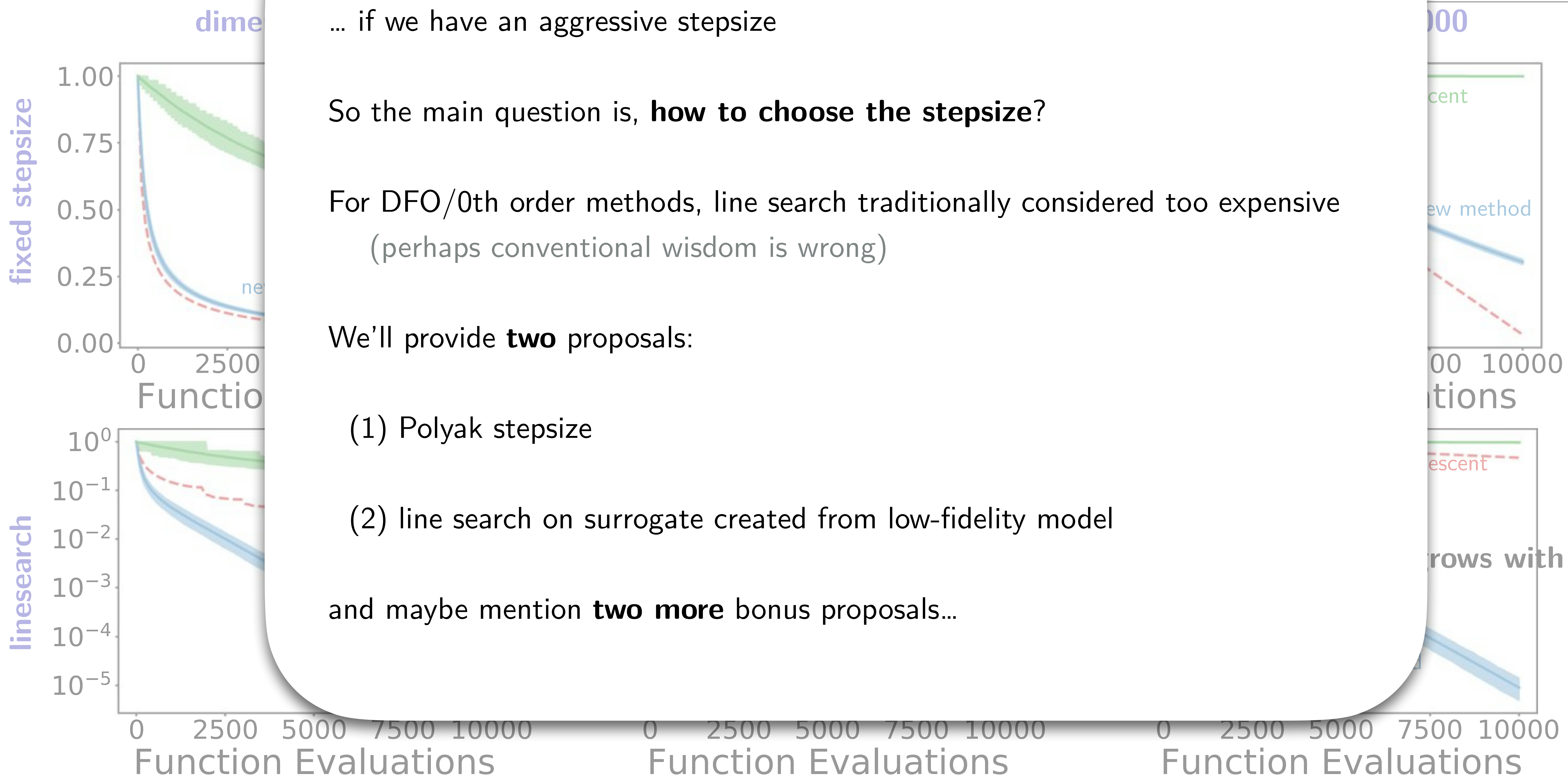
Extension: stepsize selection

$r = 20, \; \ell = 3$

Nesterov's "worst function in the world"

$r-1$

$_{i+1})^2 + x_r^2)/2 - x_1)/4,$



Working hypothesis: SSD-Haar nicely exploits low-dimensional structure...
... if we have an aggressive stepsize

So the main question is, **how to choose the stepsize**?

For DFO/0th order methods, line search traditionally considered too expensive
(perhaps conventional wisdom is wrong)

We'll provide **two** proposals:

(1) Polyak stepsize

(2) line search on surrogate created from low-fidelity model

and maybe mention **two more** bonus proposals...

Stepsize selection: Polyak stepsize

Joint project with Killian Wood,
Drona Khurana

Polyak stepsize for gradient descent (1983)

Recently revisited a lot in literature

$$\eta_k^{\text{Polyak}} = \frac{f(\boldsymbol{x}_k) - f^\star}{\|\nabla f(\boldsymbol{x}_k)\|^2} \qquad f^\star = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k^{\text{Polyak}} \nabla f(\boldsymbol{x}_k)$$

well, to be precise, actually

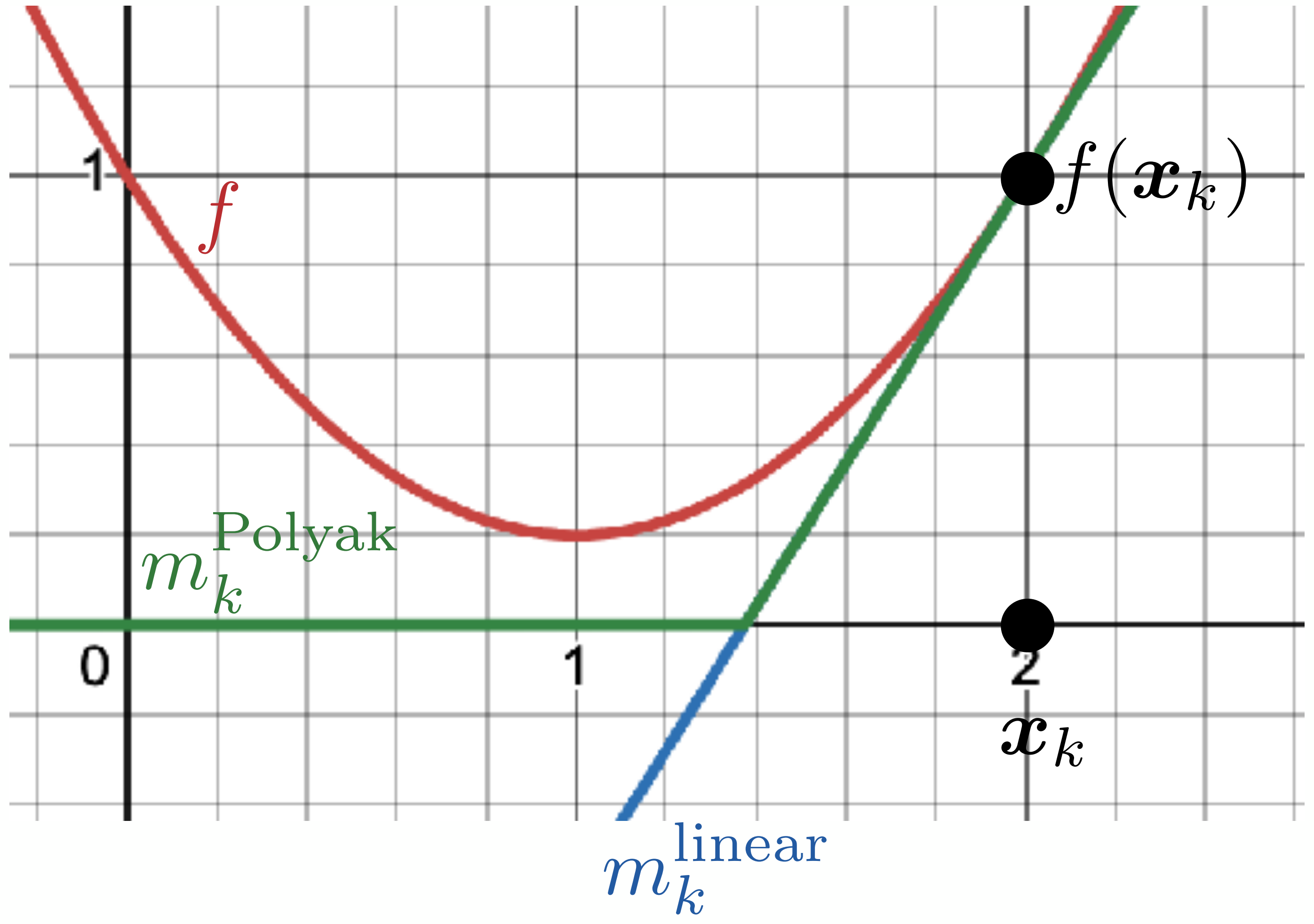
$$\eta_k^{\text{Polyak}} = \min\left\{ \frac{f(\boldsymbol{x}_k) - f^\star}{\|\nabla f(\boldsymbol{x}_k)\|^2}, \eta_{\max} \right\}$$

cf. John Duchi's semi-plenary July 22

Derivation from a *model-based view point*: $\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boldsymbol{y}} \; m_k(\boldsymbol{y}) + \frac{1}{2\eta_{\max}} \|\boldsymbol{y} - \boldsymbol{x}_k\|^2$

Asi & Duchi '19; Loyizou et al. '21; Berrada et al. '19; Davis & Drusvyatskiy '19; Schaipp et al. '23

Model	**Algorithm**
$m_k(\boldsymbol{y}) = f(\boldsymbol{y})$	Proximal Point
$m_k(\boldsymbol{y}) = f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle$	Gradient descent, fixed stepsize
$m_k(\boldsymbol{y}) = \max\{ f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle, f^\star \}$	Gradient descent, Polyak stepsize



Stepsize selection: Polyak stepsize

Polyak stepsize for gradient descent (1983)

Recently revisited a lot in literature

$$\eta_k^{\text{Polyak}} = \frac{f(\boldsymbol{x}_k) - f^\star}{\|\nabla f(\boldsymbol{x}_k)\|^2} \qquad f^\star = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k^{\text{Polyak}} \nabla f(\boldsymbol{x}_k)$$

Our extension to SSD case:

$$\eta_k^{\text{Polyak-SSD}} = \frac{f(\boldsymbol{x}_k) - f_k^\star}{\|\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k)\|^2} \qquad f_k^\star = \min_{\boldsymbol{x} \in \{\boldsymbol{x}_k\} + \text{col}\boldsymbol{Q}} f(\boldsymbol{x})$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k^{\text{Polyak-SSD}} \boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k)$$

Derivation from a *model-based view point*: $\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boldsymbol{y}} \; m_k(\boldsymbol{y}) + \frac{1}{2\eta_{\max}} \|\boldsymbol{y} - \boldsymbol{x}_k\|^2$

keep the same model... $m_k(\boldsymbol{y}) = \max\{f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle, f_k^\star\}$

... but change the context: $\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boxed{\boldsymbol{y} \in \{\boldsymbol{x}_k\} + \text{col}\boldsymbol{Q}}} \; m_k(\boldsymbol{y}) + \frac{1}{2\eta} \|\boldsymbol{y} - \boldsymbol{x}_k\|^2$

Stepsize selection: Polyak stepsize

Polyak stepsize for gradient descent (1983)

Recently revisited a lot in literature

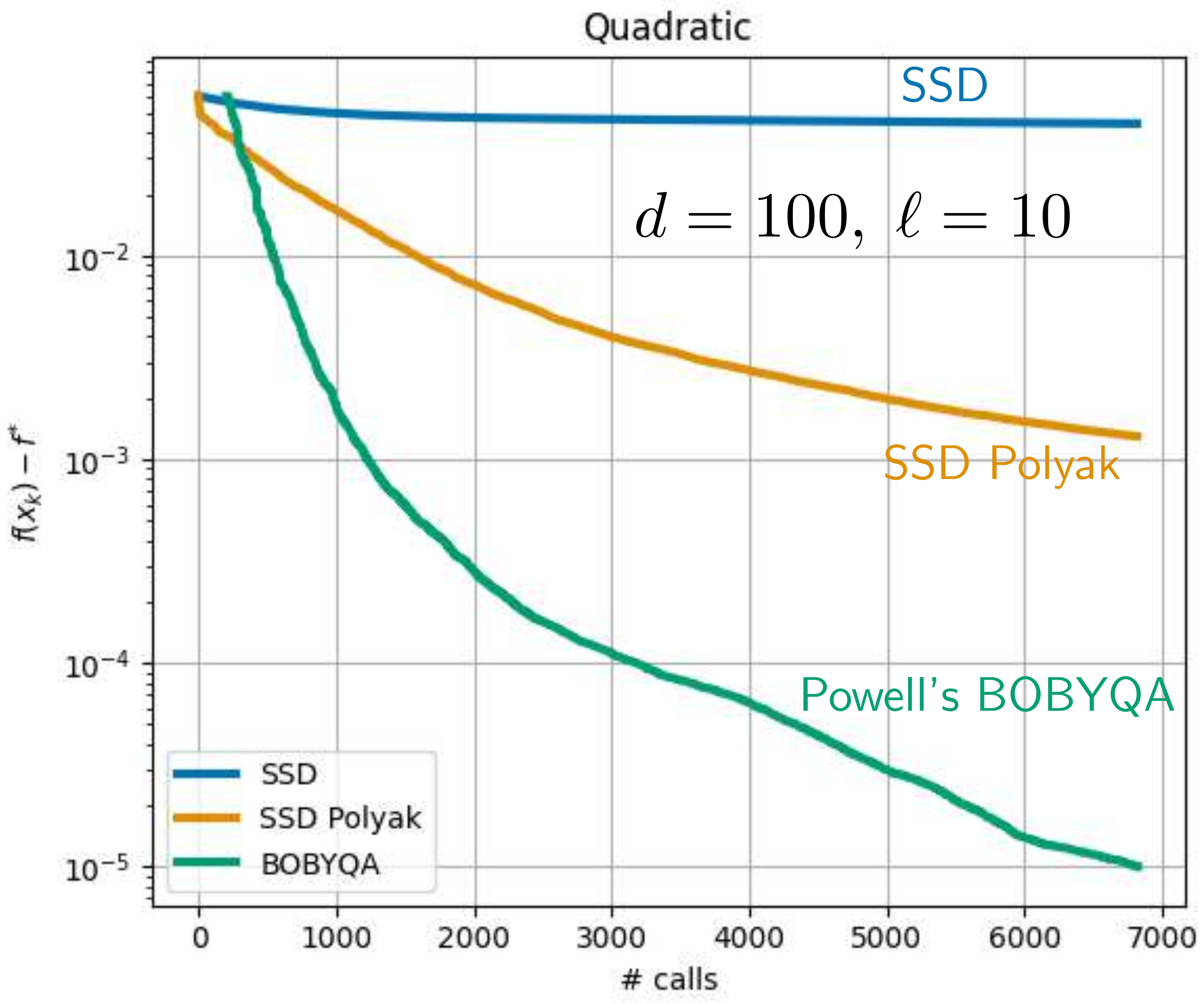
$$\eta_k^{\text{Polyak}} = \frac{f(\boldsymbol{x}_k) - f^\star}{\|\nabla f(\boldsymbol{x}_k)\|^2} \qquad f^\star = \min_{\boldsymbol{x}} f(\boldsymbol{x})$$

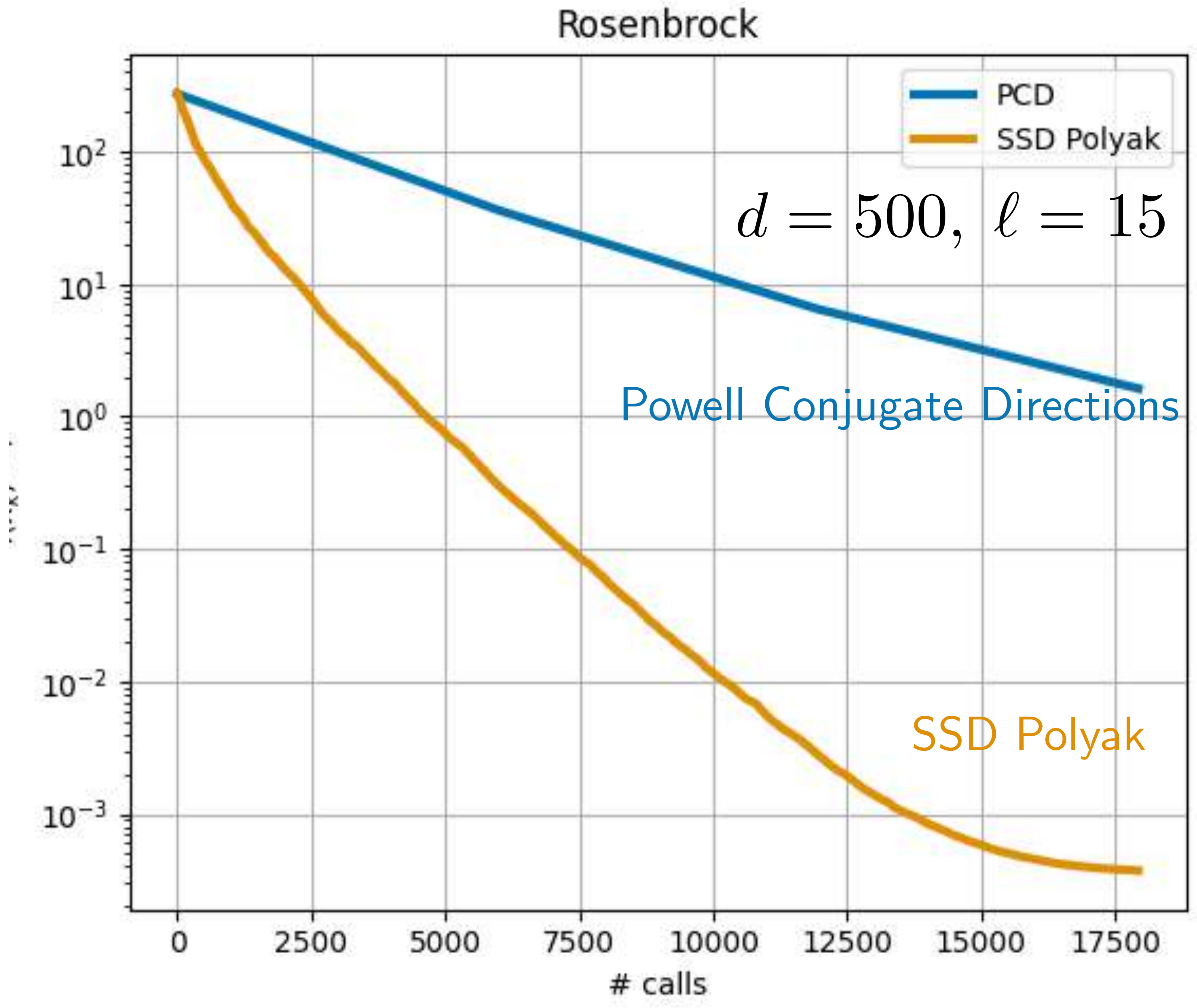
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k^{\text{Polyak}} \nabla f(\boldsymbol{x}_k)$$

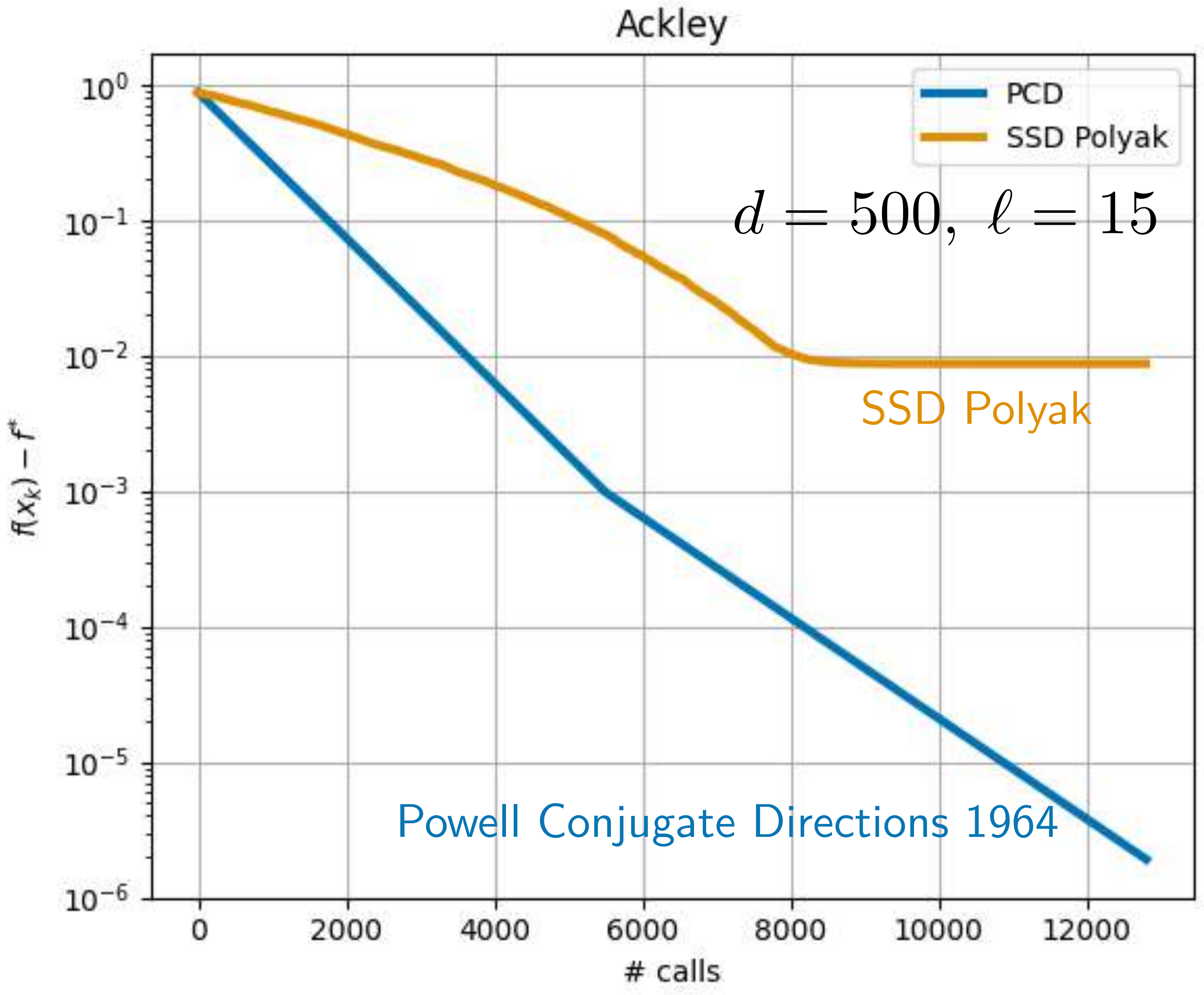
Our extension to SSD case:

$$\eta_k^{\text{Polyak-SSD}} = \frac{f(\boldsymbol{x}_k) - f_k^\star}{\|\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k)\|^2} \qquad f_k^\star = \min_{\boldsymbol{x} \in \{\boldsymbol{x}_k\} + \text{col}\boldsymbol{Q}} f(\boldsymbol{x})$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k^{\text{Polyak-SSD}} \boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k)$$







Works fine in practice, our analysis is ongoing

Stepsize selection: bifidelity surrogate

Joint project with Nuojin (Noki) Cheng (Google)

Classic exact linesearch

$$g_k = QQ^\top \nabla f(x_k)$$

$$\eta^\star = \operatorname{argmin} \varphi(\eta) \qquad \varphi(\eta) \stackrel{\text{def}}{=} f(x_k - \eta g_k)$$

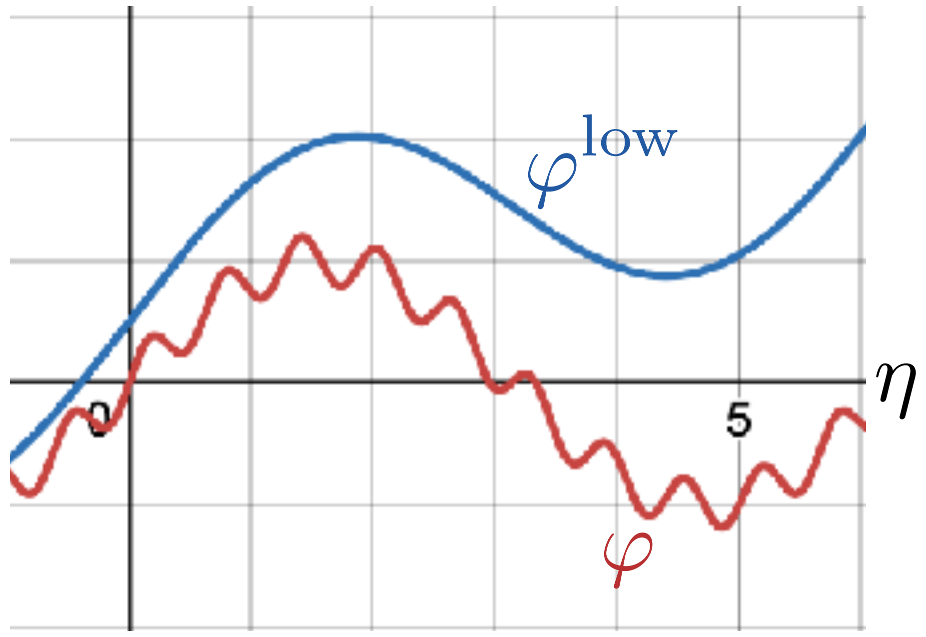
$$x_{k+1} = x_k - \eta^\star g_k$$

ideally would do line search on this but too expensive

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Expensive $\varphi(\eta) \stackrel{\text{def}}{=} f(x_k - \eta g_k)$

Inaccurate $\varphi^{\text{low}}(\eta) \stackrel{\text{def}}{=} f^{\text{low}}(x_k - \eta g_k)$



Stepsize selection: bifidelity surrogate

Joint project with Nuojin (Noki) Cheng (Google)

Classic exact linesearch

$$g_k = QQ^\top \nabla f(x_k)$$

$$\eta^\star = \operatorname{argmin} \varphi(\eta) \qquad \varphi(\eta) \overset{\text{def}}{=} f(x_k - \eta g_k)$$

$$x_{k+1} = x_k - \eta^\star g_k$$

ideally would do line search on this but too expensive

Premise: suppose we have a cheap, inaccurate approximation f^{low}

data (function evaluations)

surrogate model

Expensive $\varphi(\eta) \overset{\text{def}}{=} f(x_k - \eta g_k)$ $\{\varphi(0), \varphi(\eta_{\max})\}$

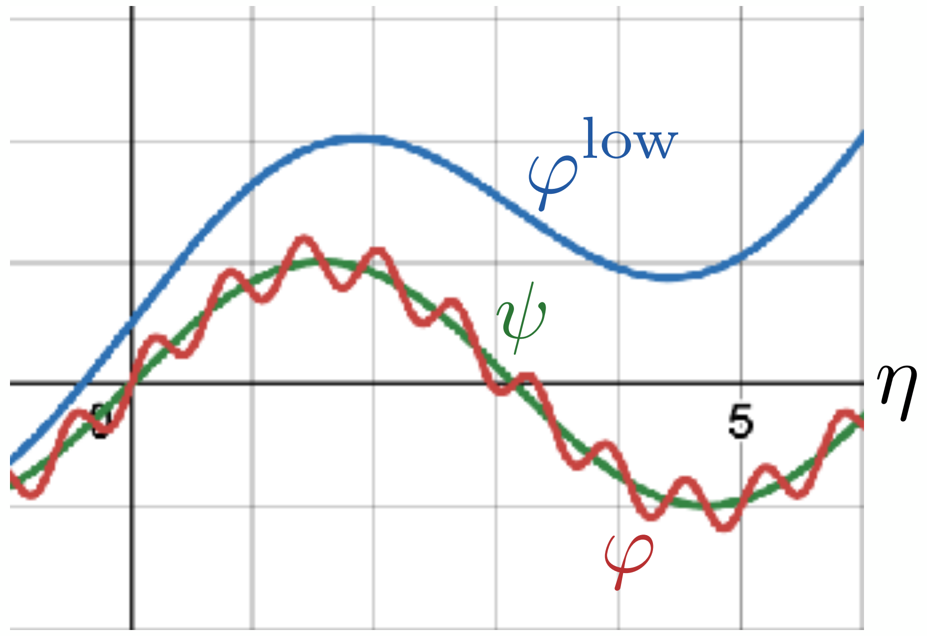
Inaccurate $\varphi^{\text{low}}(\eta) \overset{\text{def}}{=} f^{\text{low}}(x_k - \eta g_k)$ $\{\varphi^{\text{low}}(\eta_i)\}_{i=1}^{20}$

$\}$ co-kriging (1D) ⇨ $\psi(\eta)$ ⇨ $\eta^\star = \operatorname{argmin} \psi(\eta)$

traditional line search on surrogate model

(computationally "free")

e.g., calibrate low-fidelity model



Convergence analysis in our preprint "Stochastic Subspace Descent Accelerated via Bi-fidelity Line Search" arxiv.org/abs/2505.00162, Nuojin Chen, Alireza Doostan, Stephen Becker

ML bifidelity example 1

Context:
- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: black-box adversarial attack Carlini & Wagner '17, black-box extension Chen et al. '17

For a given sample, find a small perturbation such that the machine learning algorithm misclassifies it

switching to ML notation!

particular training example
(features and true label)

$$\min_{\boldsymbol{\epsilon}} -f_{\text{cross-entropy}}(g(\boldsymbol{x}^\dagger + \boldsymbol{\epsilon}), y^\dagger) + \tau\|\boldsymbol{\epsilon}\|^2$$

encourages small perturbation

model output: vector with probability of different classes

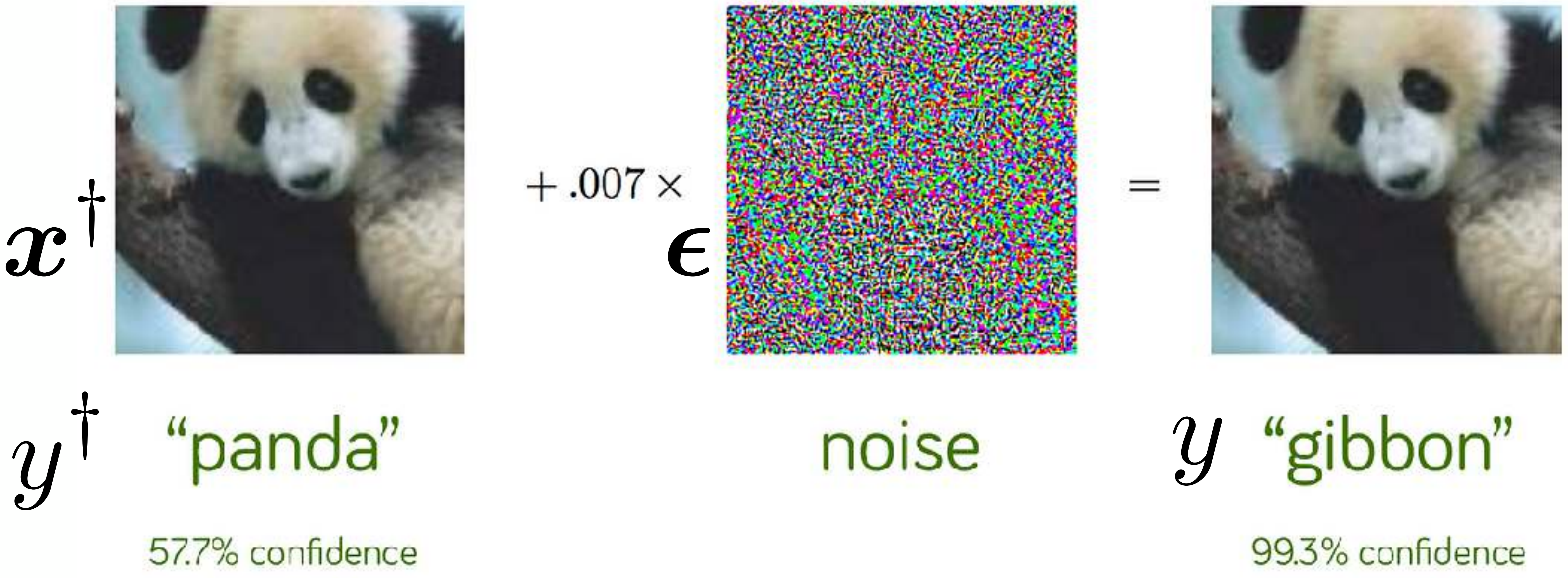


Image source: Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015

ML bifidelity example 1

Context:
- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: black-box adversarial attack Carlini & Wagner '17, black-box extension Chen et al. '17

For a given sample, find a small perturbation such that the machine learning algorithm misclassifies it

particular training example (features and true label)

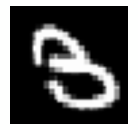
$$\min_{\boldsymbol{\epsilon}} -f_{\text{cross-entropy}}(g(\boldsymbol{x}^\dagger + \boldsymbol{\epsilon}), y^\dagger) + \tau\|\boldsymbol{\epsilon}\|^2$$

encourages small perturbation

model output: vector with probability of different classes

switching to ML notation!

MNIST is 28 x 28 images so $d = 784$



Train two models on MNIST data: (60k training, 10k test)

f is output of **large model**, trained conventionally
convolution (32 filters) -> convolution (64 filters) ->
max-pooling/flatten, fully connected (1024 neurons)
-> 10 class output. ReLU activation, 5x5 kernels

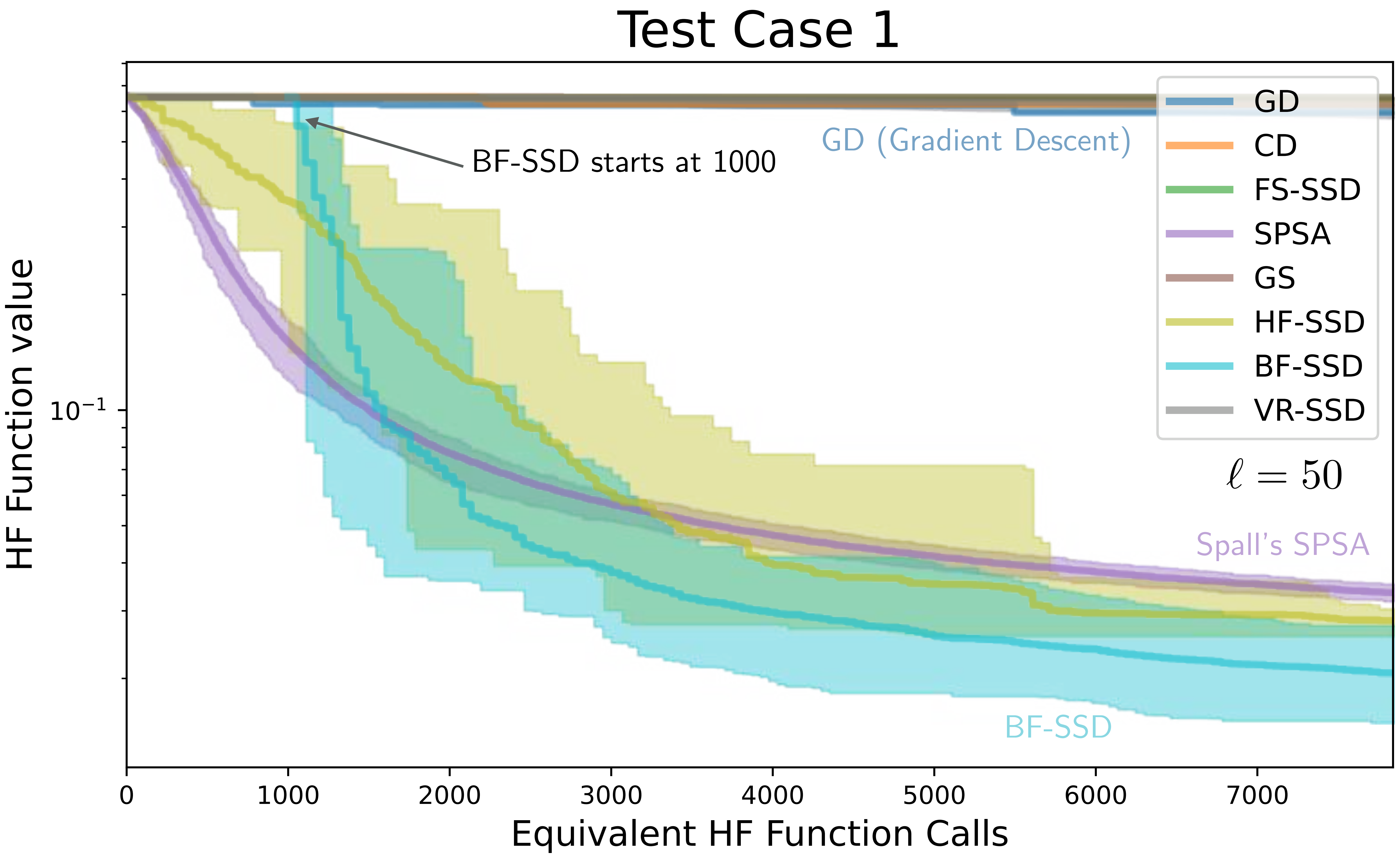
f^{low} is output of **small model**
trained not on MNIST but on output of large model
(knowledge distillation), 1000 samples
convolution (2 filters) -> max-pooling/flatten, fully connected (16 neurons)
-> 10 class output. ReLU activation, 2x3 kernels

119x larger

	Large model	Small model
# parameters	3,274,634	27,562
Test Accuracy	99.02%	82.21%

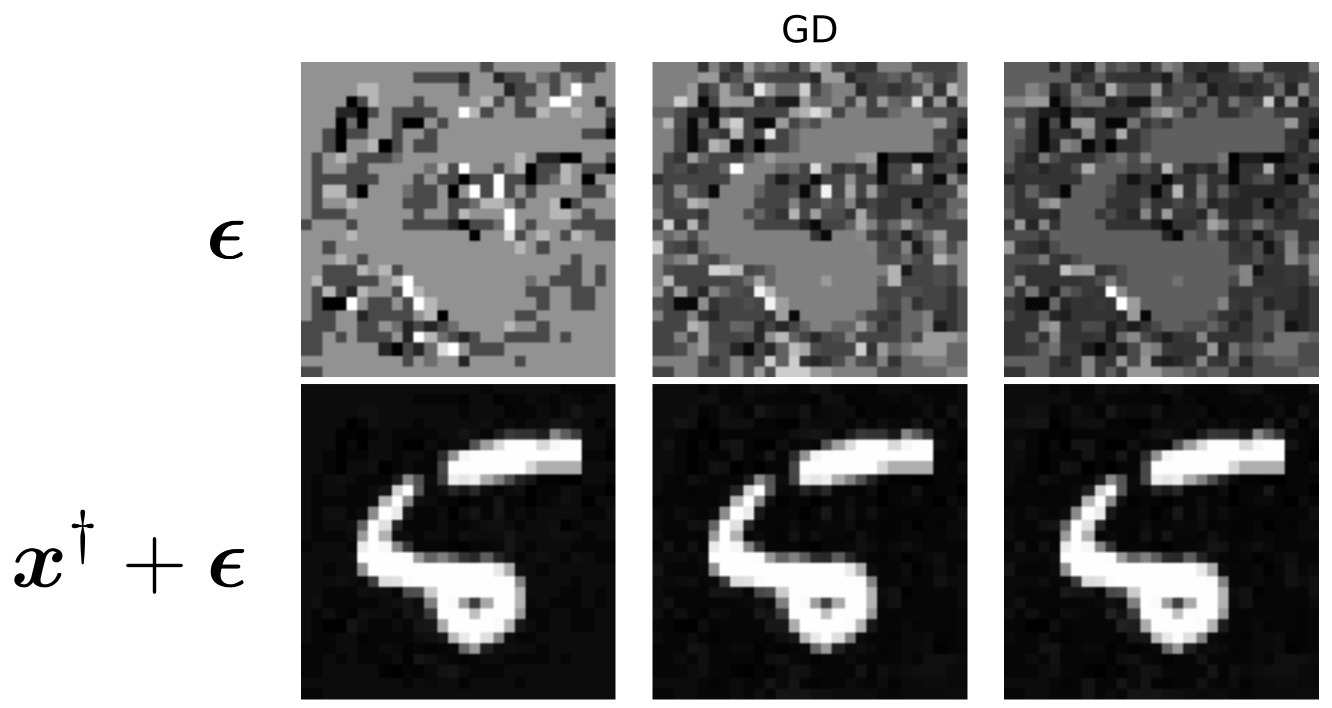
In some scenarios, small model is not just cheap but "free"

ML bifidelity example 1: Results

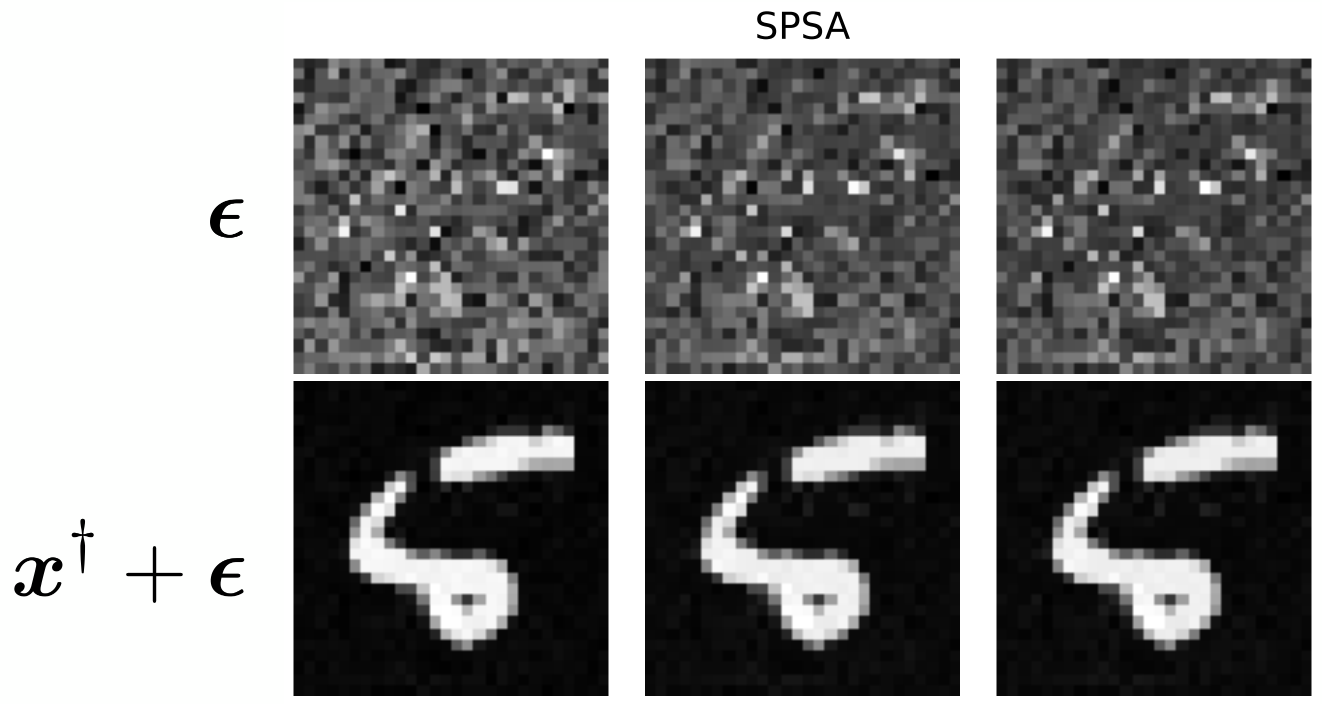


HF-SSD uses "perfect" linesearch

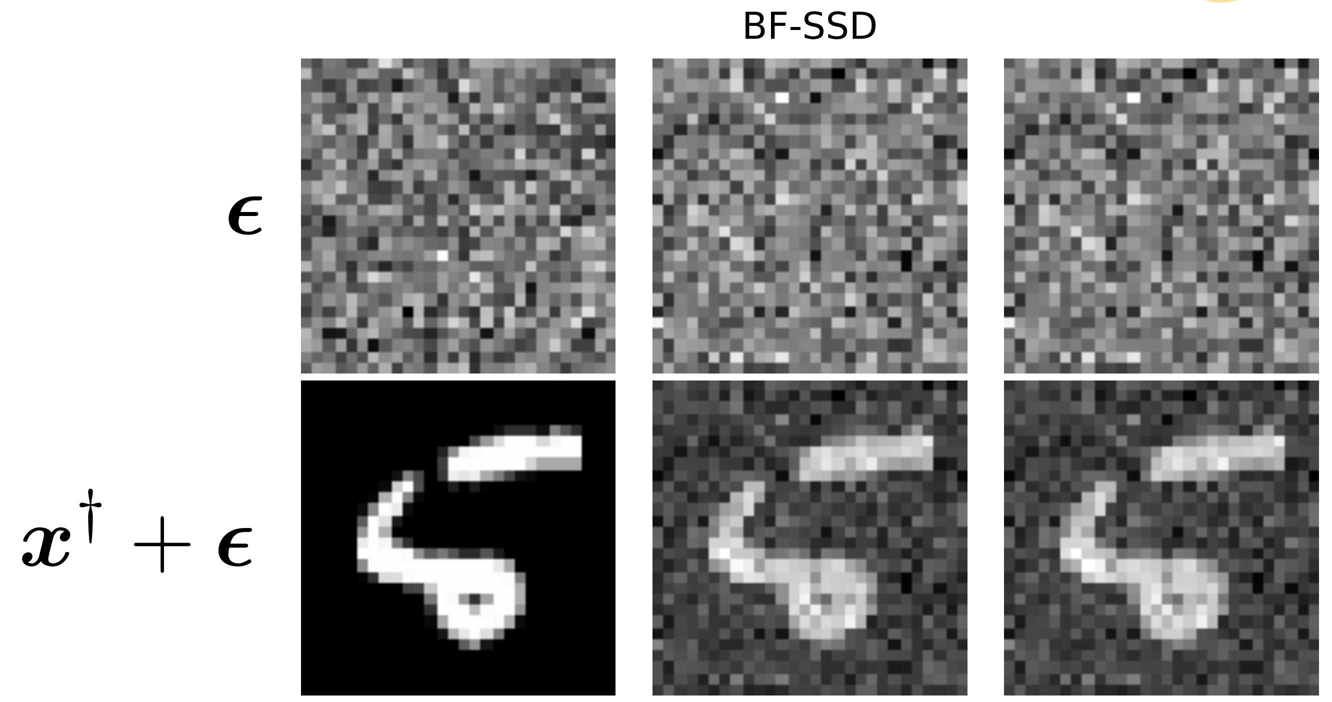
HF fun calls: 2000 5000 7000



(a) GD (Label=5, Predict=5)



(b) SPSA (Label=5, Predict=6)



(d) BF-SSD (Label=5, Predict=6)

ML bifidelity example 2

Context:

- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: soft prompting black-box LLM

We want to fine-tune a LLM like BERT or GPT

Instead of modifying network, lightweight alternative is to learn embeddings that are prepended to input sequence

ML bifidelity example 2

Context:
- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: soft prompting black-box LLM

We want to fine-tune a LLM like BERT or GPT

Instead of modifying network, lightweight alternative is to learn embeddings that are prepended to input sequence

Task: binary sentiment analysis (*classify a movie review as positive or negative*)

$$\text{Pretrained: } \begin{cases} f_{\text{token}} : \texttt{str} \to \mathbb{R}^{L_t \times d} & \text{tokenizer converts strings of any length to an embedding} \\ f_c : \mathbb{R}^{L_t \times d} \to [0,1] & \text{classifier (we use small } \underbrace{\texttt{DistilBERT}}_{\text{transformer}}\text{, small version of } \texttt{BERT}) \\ (\boldsymbol{z}, y) \in \texttt{str} \times \{0,1\} & \text{data from } \texttt{aclIMDB} \text{ database} \end{cases}$$

$d = 784$

ML bifidelity example 2

Context:
- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: soft prompting black-box LLM

We want to fine-tune a LLM like BERT or GPT

Instead of modifying network, lightweight alternative is to learn embeddings that are prepended to input sequence

Task: binary sentiment analysis (*classify a movie review as positive or negative*)

$$\text{Pretrained:}\begin{cases} f_{\text{token}} : \texttt{str} \to \mathbb{R}^{L_t \times d} & \text{tokenizer converts strings of any length to an embedding} \\ f_c : \mathbb{R}^{L_t \times d} \to [0,1] & \text{classifier (we use small } \texttt{DistilBERT}\text{, small version of } \texttt{BERT}\text{)} \\ (\boldsymbol{z}, y) \in \texttt{str} \times \{0,1\} & \text{data from } \texttt{aclIMDB} \text{ database} \end{cases}$$

transformer

$d = 784$

cross-entropy loss

$$\boxed{\boldsymbol{x}^\star \in \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \mathbb{E}_{(\boldsymbol{z},y)} \left[\mathrm{CE}(f_c(\mathrm{cat}[\boldsymbol{x}, f_{\text{token}}(\boldsymbol{z})]), y) \right]}$$

risk, replaced by empirical risk for training

$$f(\boldsymbol{x}) = \frac{1}{10} \sum_{i=1}^{10} \mathrm{CE}(f_c(\mathrm{cat}[\boldsymbol{x}, f_{\text{token}}(\boldsymbol{z}_i)]), y_i)$$

ML bifidelity example 2

Context:
- black-box model
- high-dimensional, low accuracy

Premise: suppose we have a cheap, inaccurate approximation f^{low}

Example: soft prompting black-box LLM

We want to fine-tune a LLM like BERT or GPT

Instead of modifying network, lightweight alternative is to learn embeddings that are prepended to input sequence

Task: binary sentiment analysis (*classify a movie review as positive or negative*)

Pretrained:

$f_{\text{token}} : \texttt{str} \to \mathbb{R}^{L_t \times d}$	tokenizer converts strings of any length to an embedding
$f_c : \mathbb{R}^{L_t \times d} \to [0,1]$	classifier (we use small `DistilBERT`, small version of `BERT`) transformer
$(\boldsymbol{z}, y) \in \texttt{str} \times \{0,1\}$	data from `aclIMDB` database

$d = 784$

cross-entropy loss

$$\boxed{\boldsymbol{x}^\star \in \underset{\boldsymbol{x}\in\mathbb{R}^d}{\operatorname{argmin}} \ \mathbb{E}_{(\boldsymbol{z},y)} \left[\mathrm{CE}(f_c(\mathrm{cat}[\boldsymbol{x}, f_{\text{token}}(\boldsymbol{z})]), y)\right]}$$

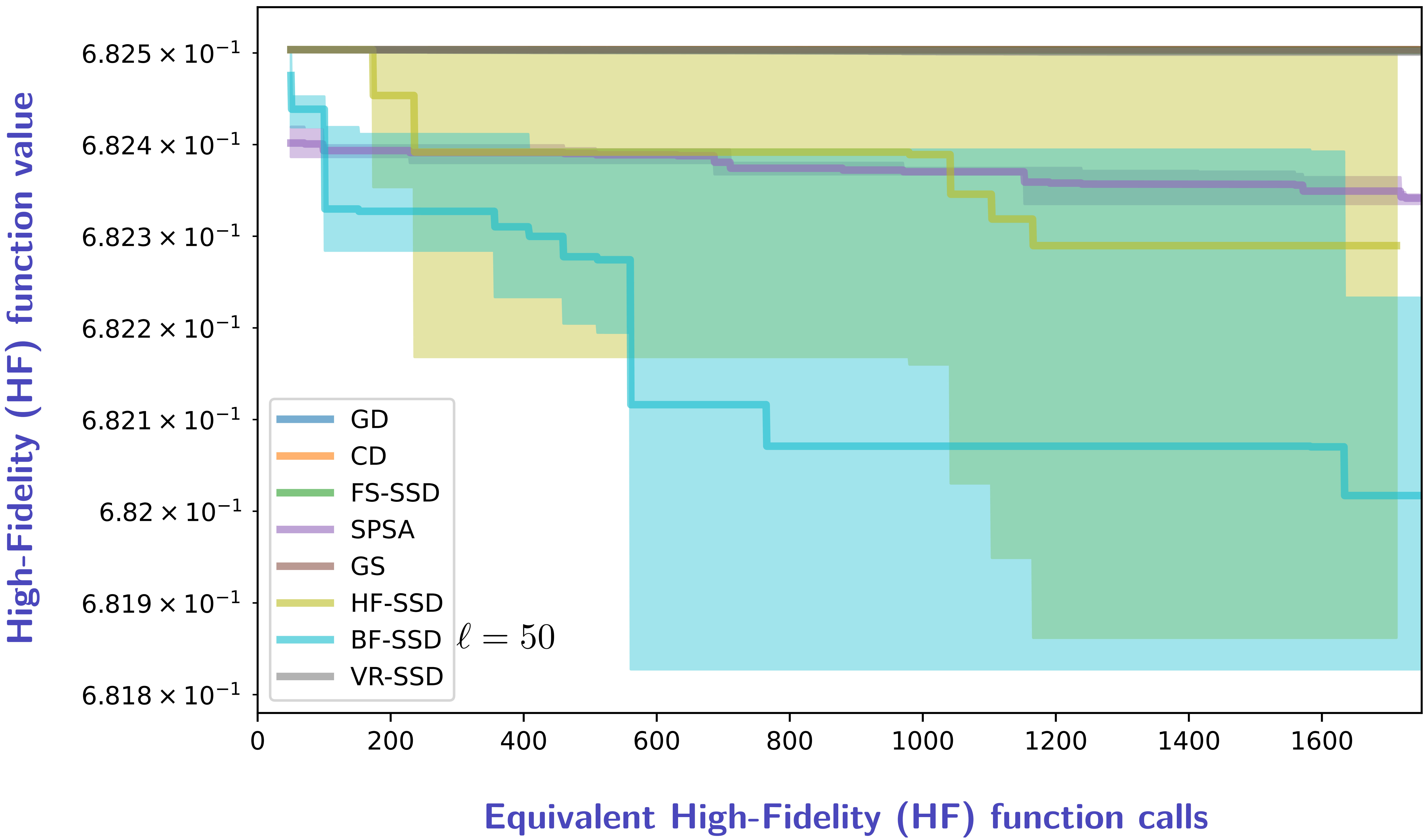
risk, replaced by empirical risk for training

$$f(\boldsymbol{x}) = \frac{1}{10}\sum_{i=1}^{10} \mathrm{CE}(f_c(\mathrm{cat}[\boldsymbol{x}, f_{\text{token}}(\boldsymbol{z}_i)]), y_i)$$

f uses a sample size of 10 High-Fidelity

f^{low} uses a sample size of 2 Low-Fidelity

ML bifidelity example 2: Results



Spall's SPSA

HF-SSD uses
"perfect" linesearch

BF-SSD

Bonus ideas

3. Just do a line search!

e.g.: C. Paquette, K. Scheinberg. *A Stochastic Line Search Method with Expected Complexity Analysis.* SIOPT, 2020.

Probably works great, but we haven't analyzed in our context

Bonus ideas

3. Just do a line search!

e.g.: C. Paquette, K. Scheinberg. *A Stochastic Line Search Method with Expected Complexity Analysis*. SIOPT, 2020.

Probably works great, but we haven't analyzed in our context

4. Barzilai-Borwein stepsize

has been used before in SGD methods, e.g., C. Tan, S. Ma, Y.-H. Dai, Y. Qian. *Barzilai-Borwein Step Size for Stochastic Gradient Descent*. NIPS, 2016.

Popular scalar approximation to attempt to solve secant equation $\boldsymbol{B}\boldsymbol{s} = \boldsymbol{y}$ where

$$\boldsymbol{s} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$$

$$\boldsymbol{y} = \nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1})$$

$$\eta_{\text{BB1}} = \frac{\|\boldsymbol{s}\|^2}{\boldsymbol{s}^\top \boldsymbol{y}}$$

$$\eta_{\text{BB2}} = \frac{\boldsymbol{s}^\top \boldsymbol{y}}{\|\boldsymbol{y}\|^2}$$

In a DFO setting, the BB1 stepsize is reasonable: we have $\boldsymbol{s}$ explicitly, and $\boldsymbol{s}^\top \boldsymbol{y}$ can be computed with two directional derivatives

Downside: even in deterministic full-gradient settings, convergence isn't guaranteed

What's next? Biased sampling

Suppose we have a crude estimate, $\boldsymbol{g}$, of the gradient

e.g., from past iterations, or low-fidelity model, etc.

try quasi-Newton ideas?

How to exploit this?

we already saw a few ways — here is one more:

$$\boldsymbol{v} = \frac{1}{\ell}\boldsymbol{\Sigma}^{-1}\boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k) \text{ and } \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \boldsymbol{v}$$

where each column $\boldsymbol{q}$ of $\boldsymbol{Q}$ is iid $\boldsymbol{q} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$

and $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} + \boldsymbol{g}\boldsymbol{g}^\top / \|\boldsymbol{g}\|^2$

adjust based on our *confidence* in gradient estimate

Then still unbiased, $\mathbb{E}[\boldsymbol{v}] = \nabla f(\boldsymbol{x}_k)$

but hopefully much lower variance.

inspired by Boullé and Townsend '22

What’s next? Biased sampling

Suppose we have a crude estimate, $\boldsymbol{g}$, of the gradient

e.g., from past iterations, or low-fidelity model, etc.

try quasi-Newton ideas?

How to exploit this?

we already saw a few ways — here is one more:

$$\boldsymbol{v} = \frac{1}{\ell}\boldsymbol{\Sigma}^{-1}\boldsymbol{Q}\boldsymbol{Q}^\top \nabla f(\boldsymbol{x}_k) \quad \text{and} \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \boldsymbol{v}$$

where each column $\boldsymbol{q}$ of $\boldsymbol{Q}$ is iid $\boldsymbol{q} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$

and $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} + \boldsymbol{g}\boldsymbol{g}^\top / \|\boldsymbol{g}\|^2$

adjust based on our *confidence* in gradient estimate

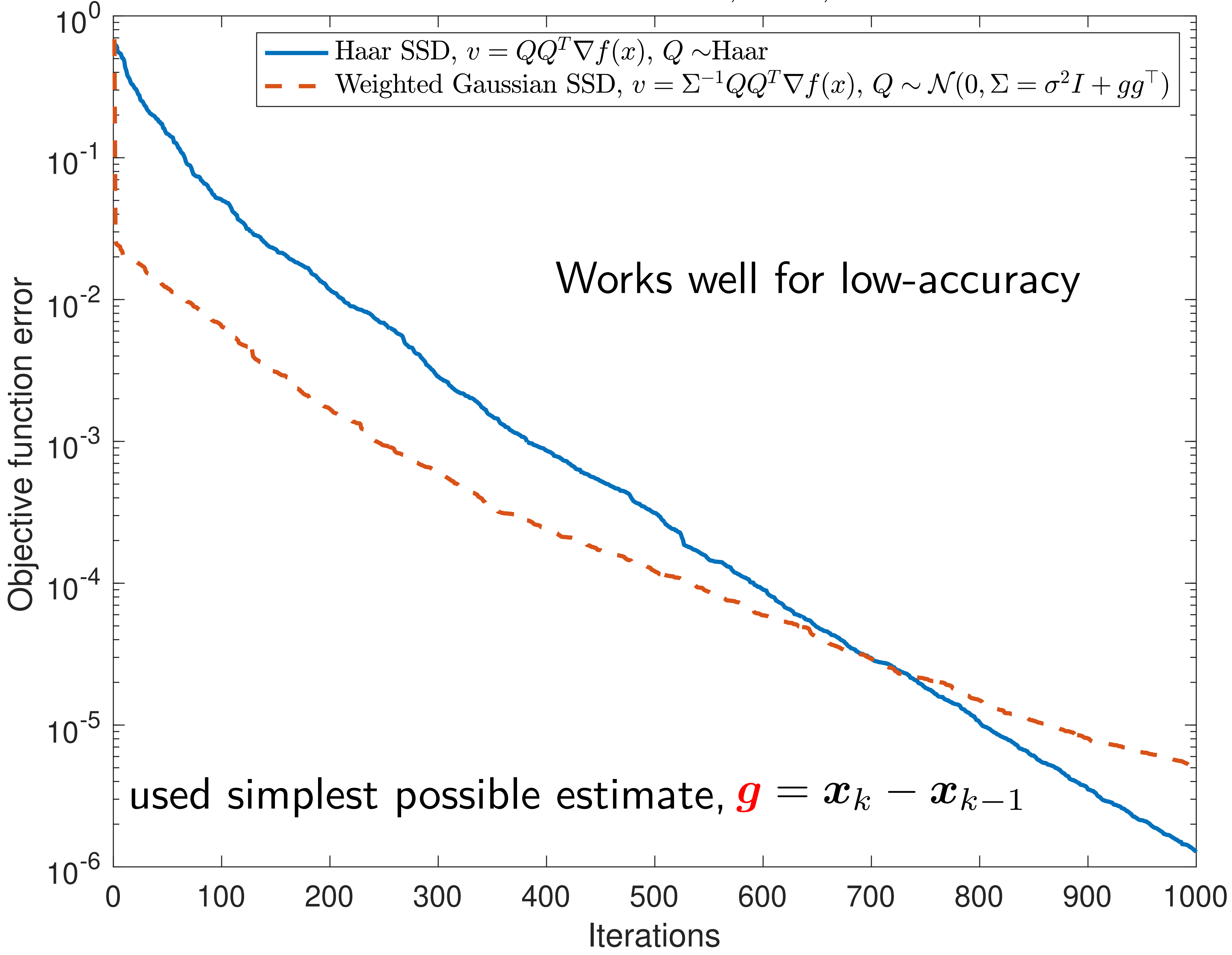
Then still unbiased, $\mathbb{E}[\boldsymbol{v}] = \nabla f(\boldsymbol{x}_k)$

but hopefully much lower variance.

inspired by Boullé and Townsend ‘22

Preliminary example on a quadratic function

SSD with exact linesearch, $\ell = 3$, $d = 100$



Works well for low-accuracy

used simplest possible estimate, $\boldsymbol{g} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$

works poorly when ℓ is large,

since then Gaussian is inferior to Haar

What's next? Understand dependence on Hessian eigenvalues

Restrict to convex quadratic functions for simplicity, $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top H\boldsymbol{x} + \boldsymbol{c}^\top\boldsymbol{x}$

The **min** and **max** eigenvalue of H control worst-case behavior (if using fixed stepsize)

... but the **interior** eigenvalues play a role with *average-case analysis* and/or *exact line search*

What's next? Understand dependence on Hessian eigenvalues

Restrict to convex quadratic functions for simplicity, $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top H \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}$

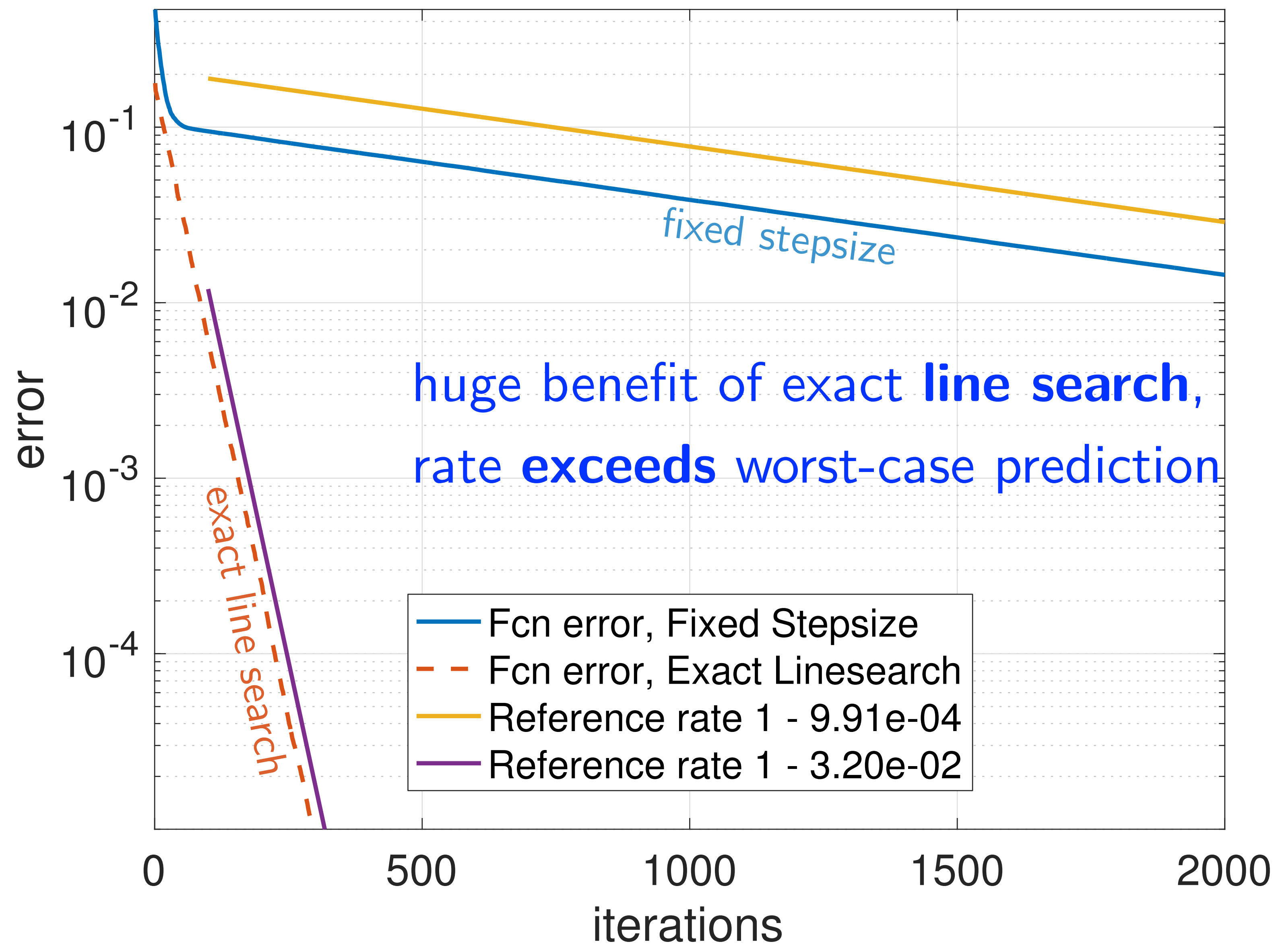
The **min** and **max** eigenvalue of H control worst-case behavior (if using fixed stepsize)

$d = 100, \ell = 5$

... but the **interior** eigenvalues play a role with *average-case analysis* and/or *exact line search*

Case A: 1 big eigenvalue, the rest small

(Small "numerical rank") $\kappa = \lambda_{\max}/\lambda_{\min} = 100$

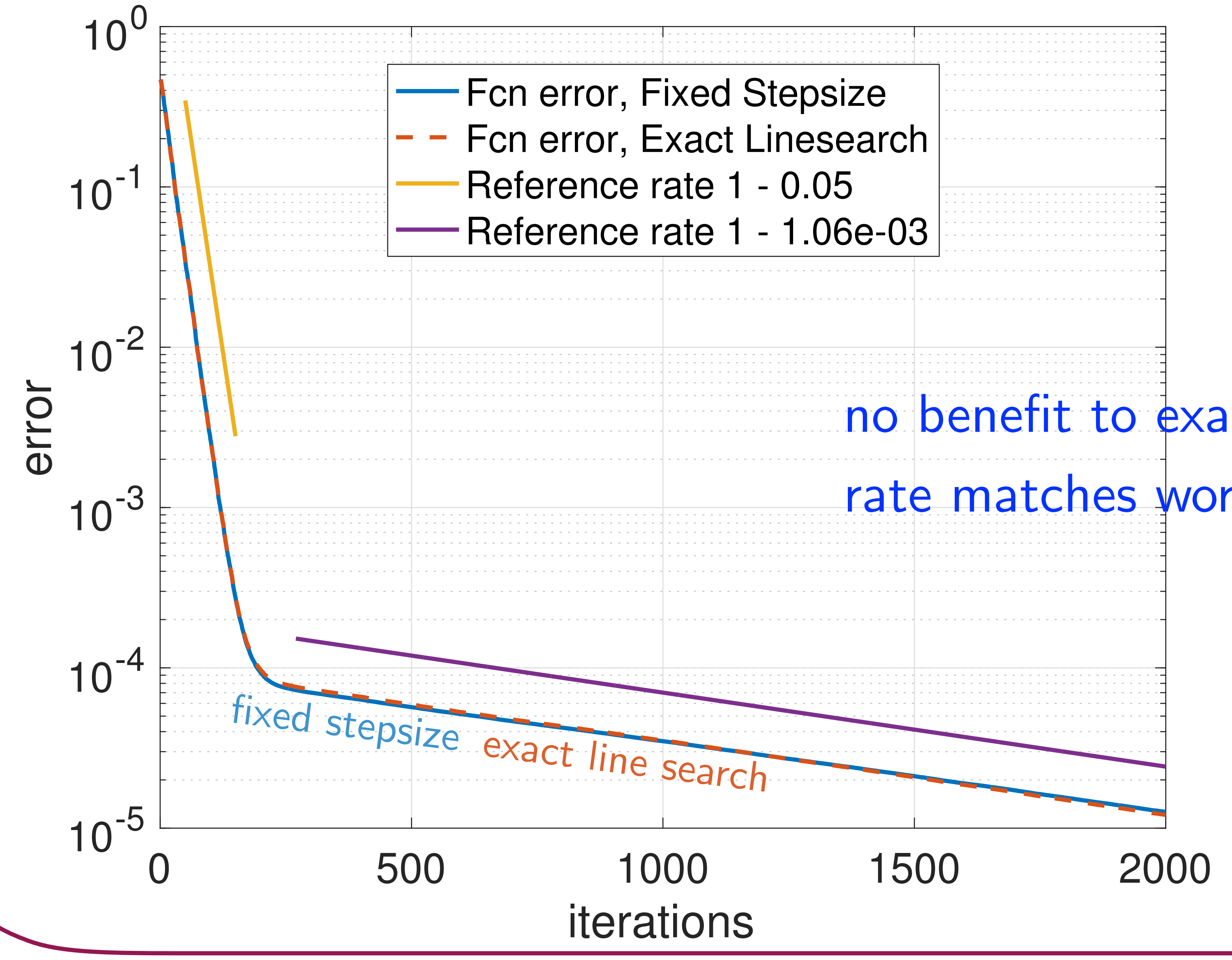


huge benefit of exact **line search**,

rate **exceeds** worst-case prediction

Case B: 1 small eigenvalue, the rest large

(Full "numerical rank") $\kappa = \lambda_{\max}/\lambda_{\min} = 100$



no benefit to exact line search,

rate matches worst-case prediction

Conclusion

- 0th order methods have their roles
- stepsize selection (and/or line search) is important
- multi fidelity is useful

Thanks for your attention

Slides will be posted at https://stephenbeckr.github.io/papers/#talks