

Two Tales, One Resolution for Physics-Informed Inference-time Scaling

Debiasing and Precondition

Yiping Lu
Northwestern | McCORMICK SCHOOL OF
ENGINEERING

Do you trust your theorem?

Big Constant

Theorem If you randomly collect () data, then you can achieve () accuracy with your AI!

Relu network is **optimal** for function regression

PINN is **optimal** for differential equation solving

Diffusion Model is **optimal** for density estimation

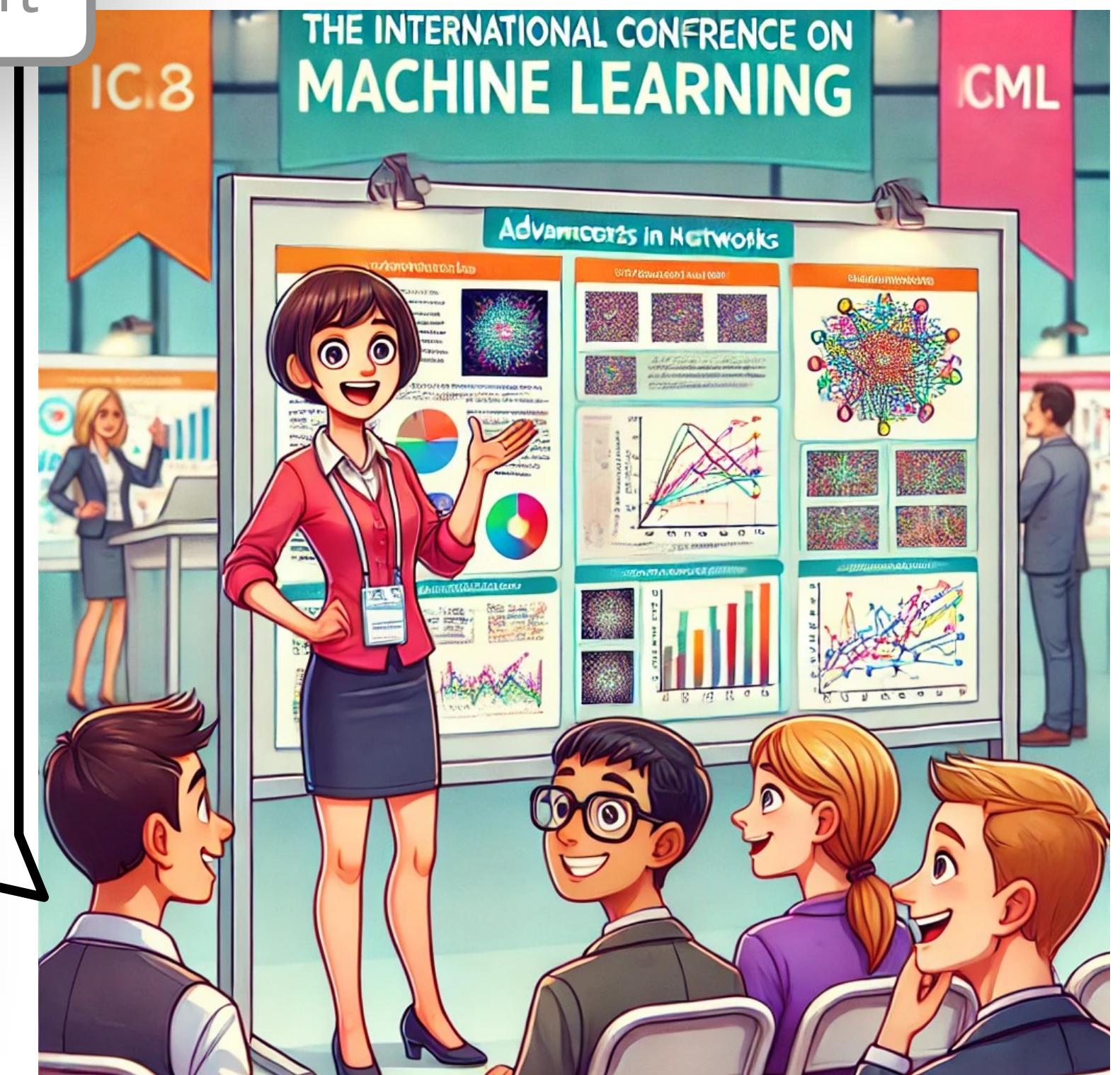
Transformer is **optimal** for in-context learning

...

Global optimization, the structure of target function,
The smoothness of ReLU, lower bound of $p(x)$...



Does Optimal results really gives me guarantee?



Consider How you use ChatGPT...

What is the most efficient route from Times Square to Chelsea Market that minimizes walking?

Option 1: Subway

1. **Walk to a Nearby Subway Entrance:** Head to the Times Sq–42nd Street station.
2. **Board the E Train:** Catch an E train heading downtown (toward World Trade Center).
3. **Ride to 8th Avenue–14th Street Station:** Get off at this station (roughly 4 stops).
4. **Walk to Chelsea Market:** Exit the station and walk east on 14th Street for a few blocks until you reach Chelsea Market at 75 9th Avenue.



double-check
with a map

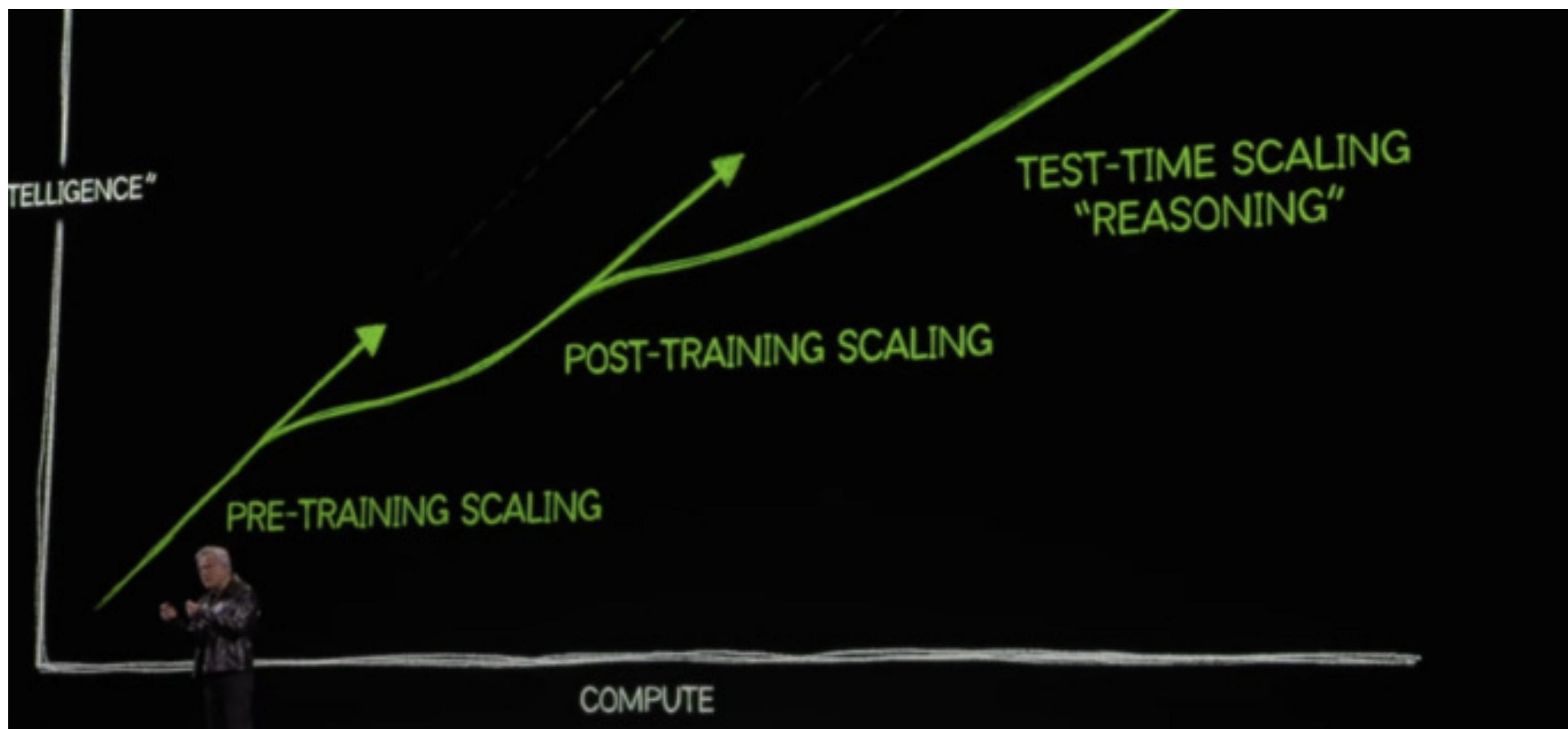
Port Authority

2 stops for A
3 stops for C/E

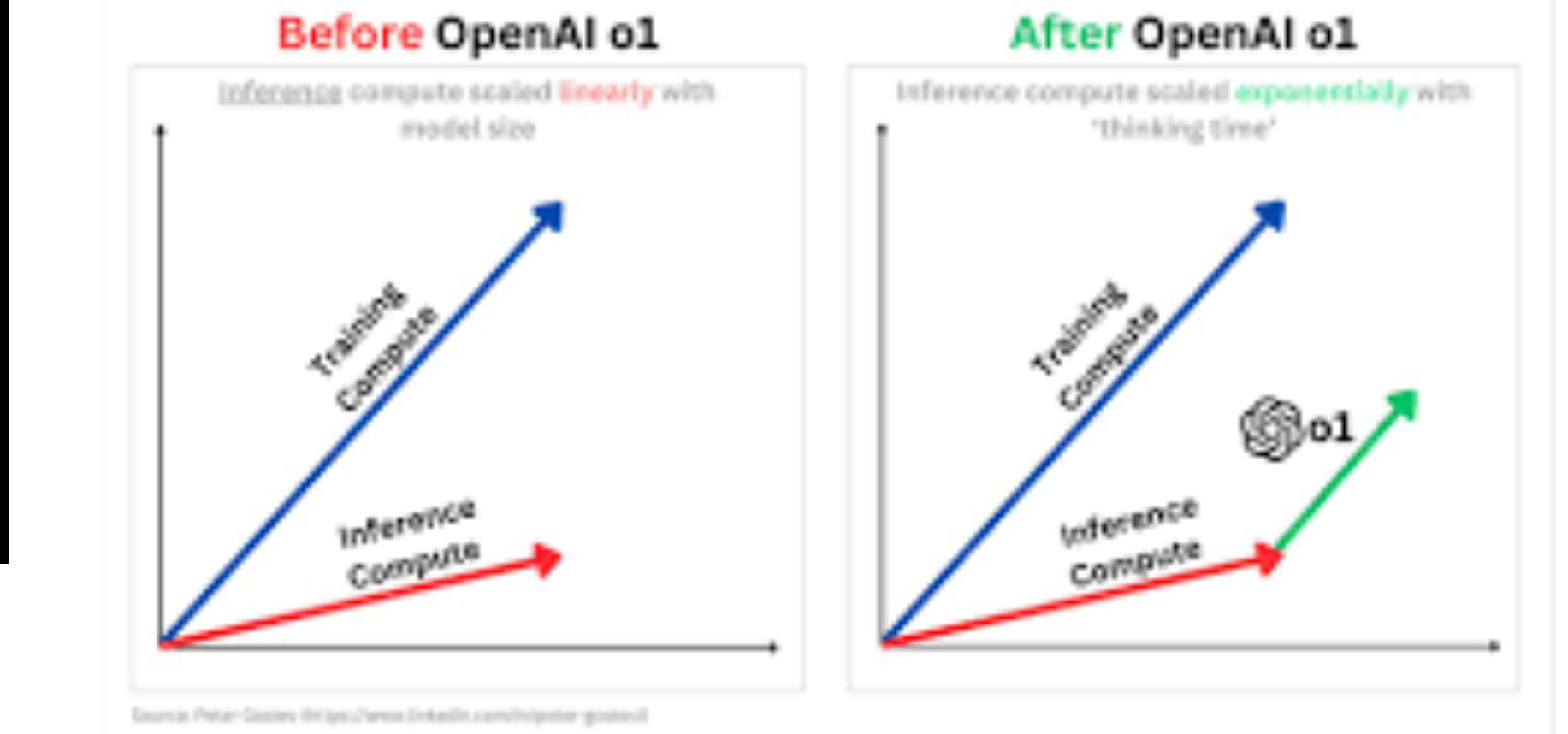
1. **At Times Square:** Enter the Times Square–42nd Street station.
2. **Board the 1 Train:** Hop on a downtown 1 train (the red line).
3. **Ride to 14th Street:** Stay on until you reach the 14th Street station.
4. **Exit Appropriately:** Use the exit that leads toward 9th Avenue—this drop-off point is just a short walk from Chelsea Market (located at 75 9th Ave).



Inference Time Scaling Law



New scaling law: why OpenAI's o1 model matters
OpenAI created a new way to scale - through more compute during generation

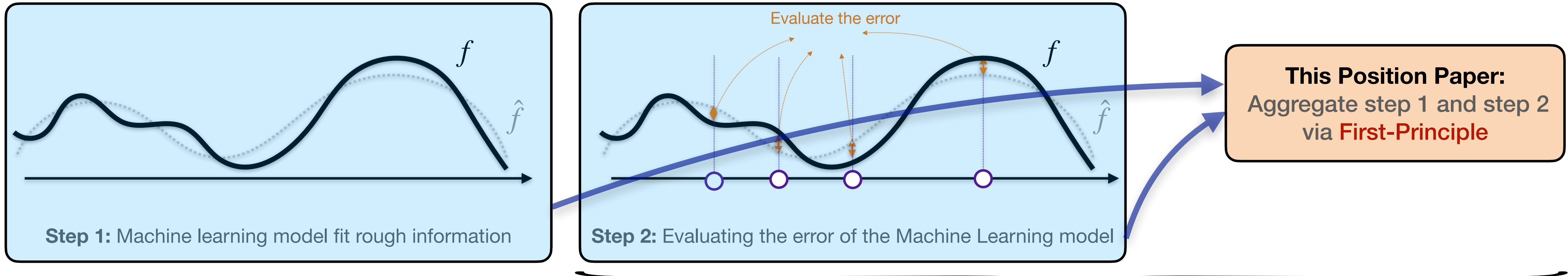


How can we perform Inference-Time Scaling for Scientific Machine Learning?

With trustworthy guarantee

Tale 1: Debiasing Hybrid Scientific Computing and Machine Learning

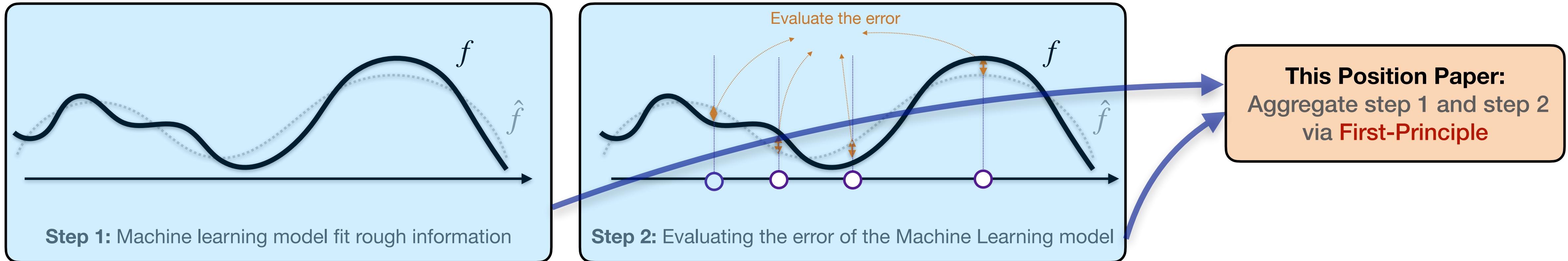
Physics-Informed Inference Time Scaling



Computation only happens at inference time

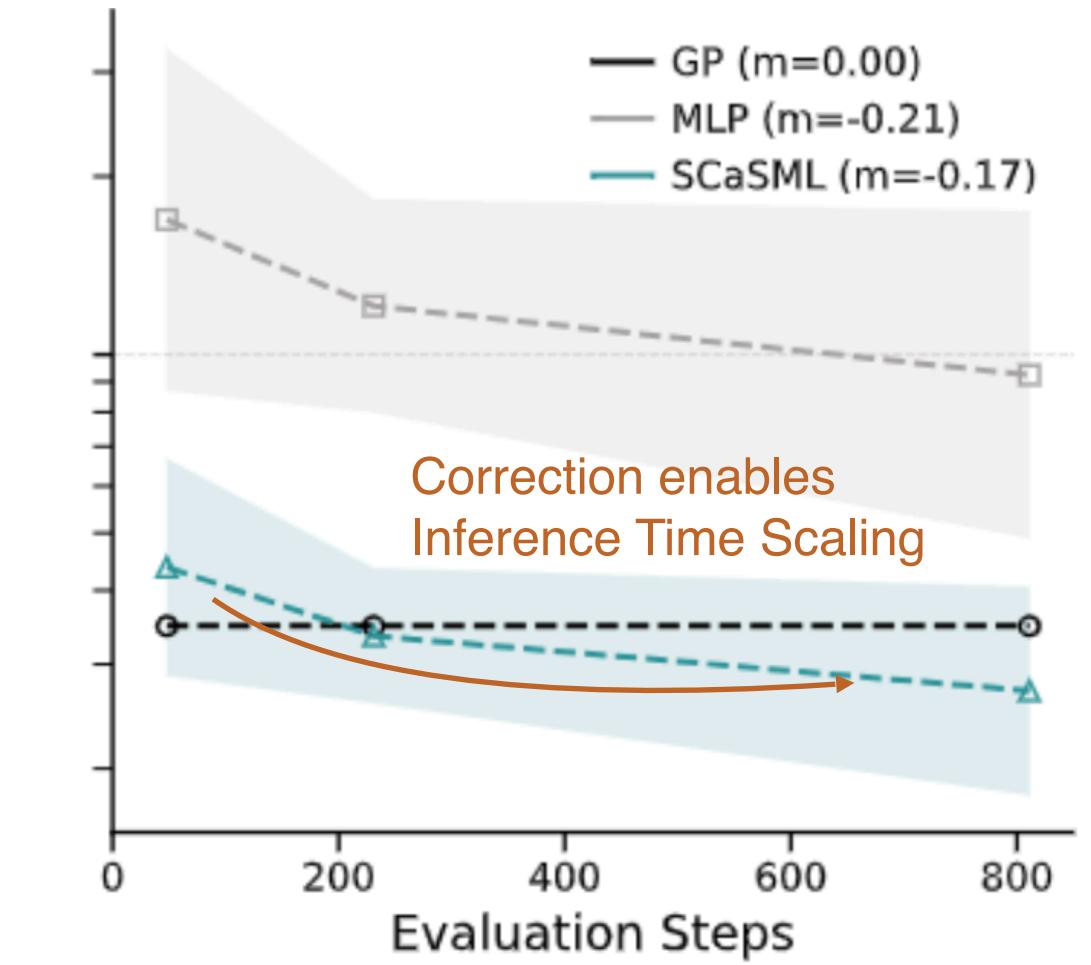
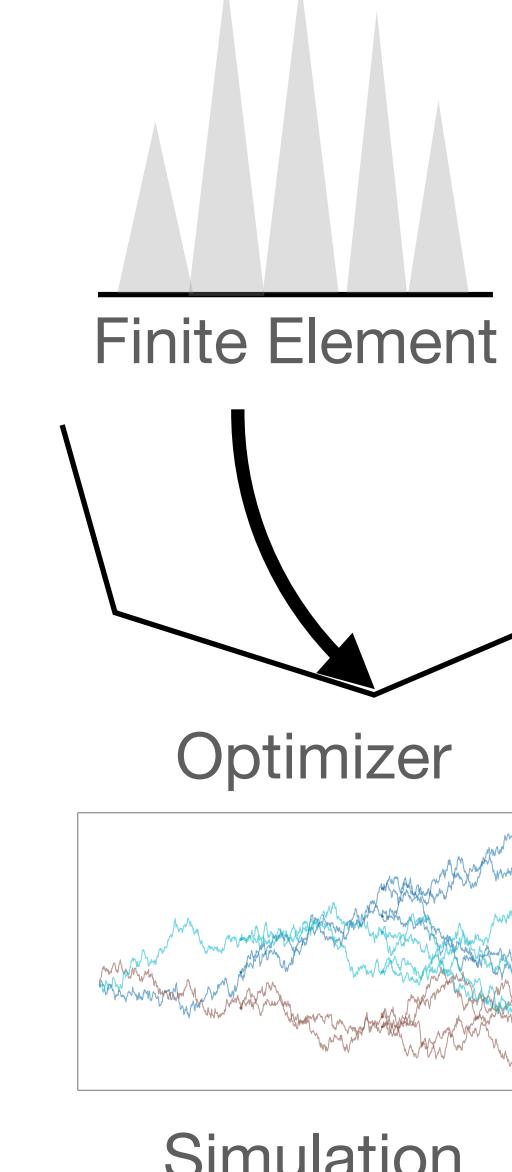
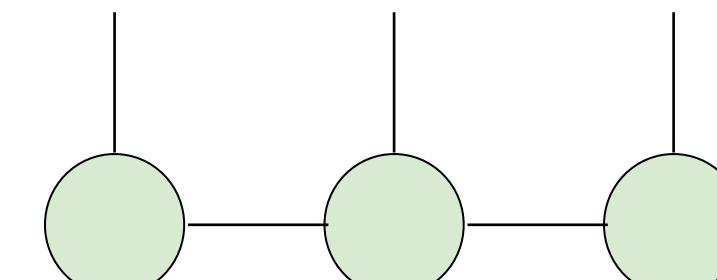
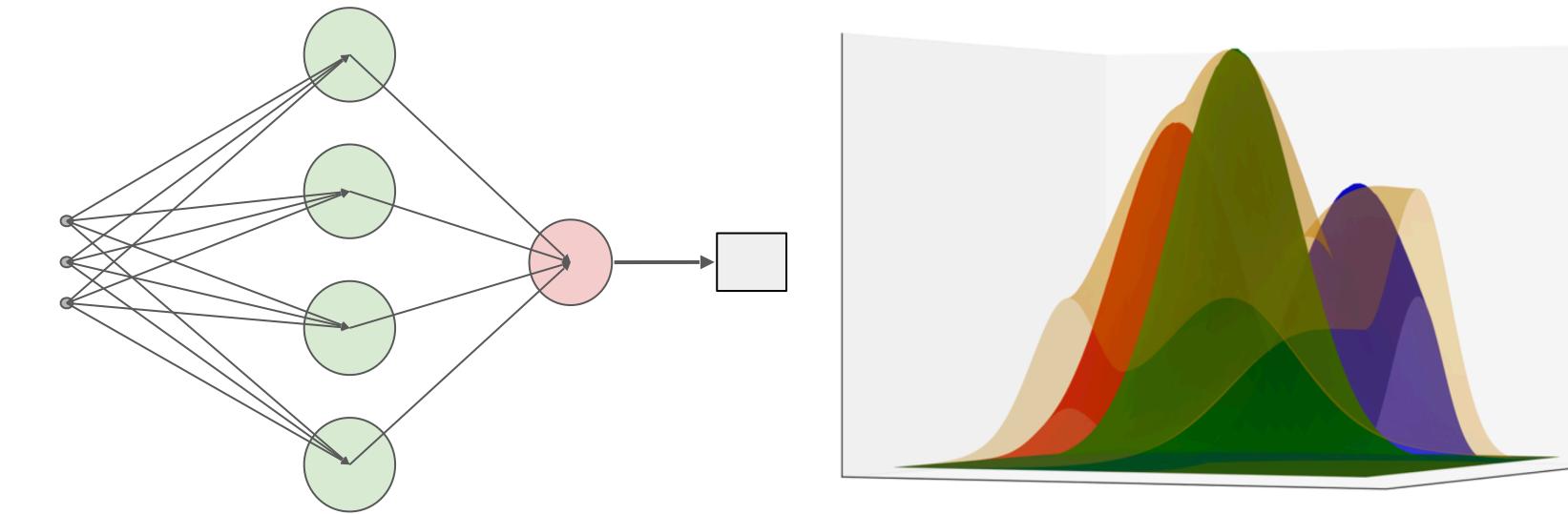
No need for retraining

Physics-Informed Inference Time Scaling



Step 2. Correct with a Trustworthy Solver

Step 1. Train a Surrogate (ML) Model



Our Framework

Start with machine learning

$$\{X_1, \dots, X_n\} \sim \mathbb{P}_\theta \rightarrow \hat{\theta} \rightarrow \Phi(\hat{\theta})$$

Scientific Machine Learning

Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

Function fitting

Example 2

$$\theta = \Delta^{-1}f, \quad X_i = (x_i, f(x_i)) \quad \text{Solving } \Delta u = f$$

Solving PDE

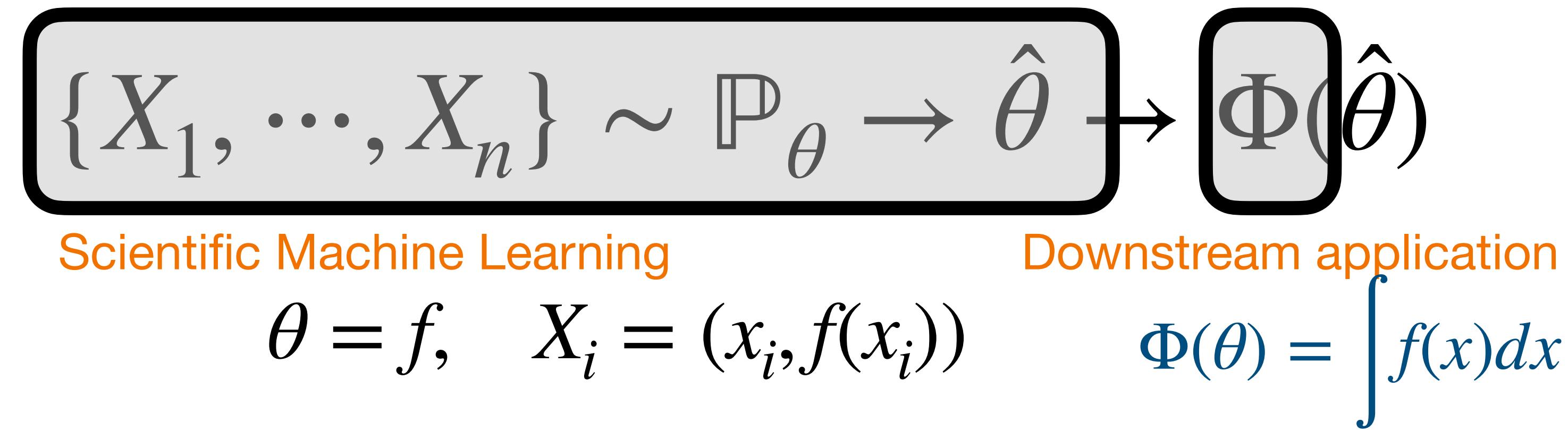
Example 3

$$\theta = A, \quad X_i = (x_i, Ax_i)$$

Estimation \hat{A} via Randomized SVD

Our Framework

Consider the downstream application...



Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i)) \quad \Phi(\theta) = \int f(x) dx$$

Example 2

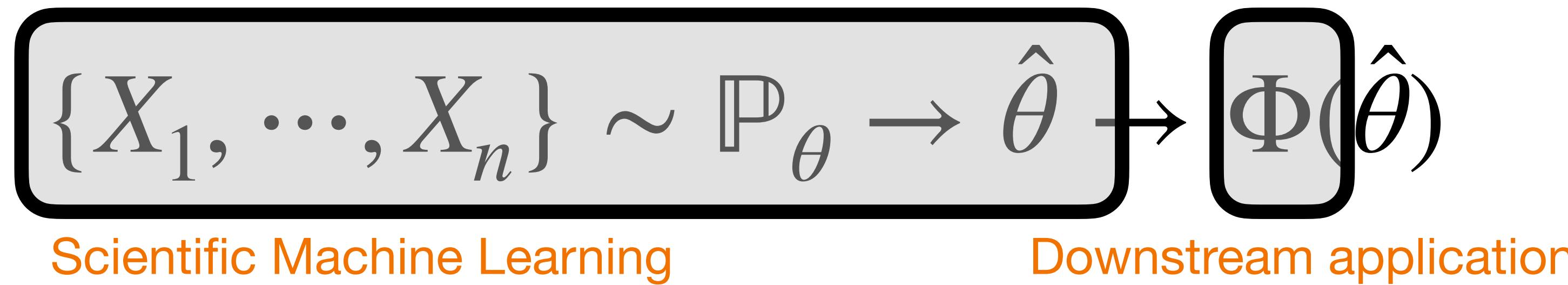
$$\theta = \Delta^{-1}f, \quad X_i = (x_i, f(x_i)) \quad \Phi(\theta) = (\Delta^{-1}f)(x)$$

Example 3

$$\theta = A, \quad X_i = (x_i, Ax_i) \quad \Phi(\theta) = \text{tr}(A)$$

Our Framework

Debiasing the downstream application...



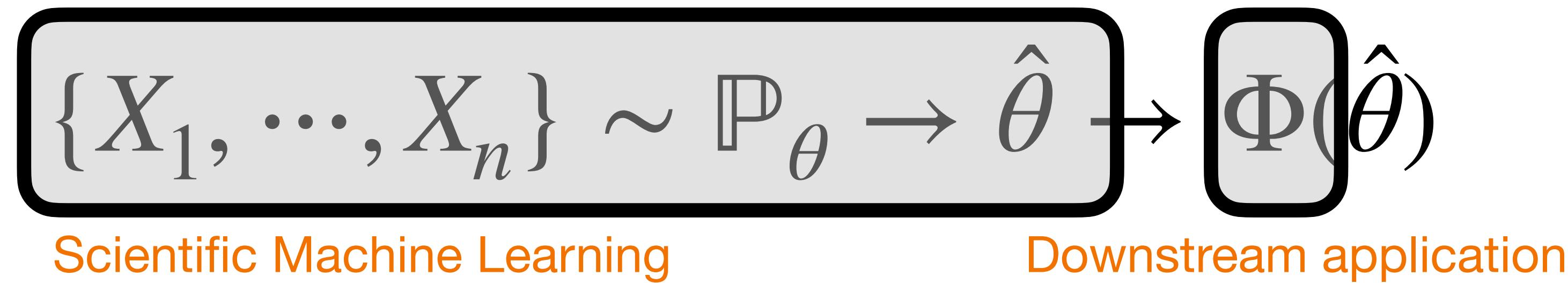
AIM: Unbiased prediction even with biased machine learning estimator

AIM: Compute $\Phi(\hat{\theta}) - \Phi(\theta)$ during Inference time



Using (stochastic) simulation to calibrate the (scientific) machine learning output !

Our Framework



AIM: Unbiased prediction even with biased machine learning estimator

How to estimate $\Phi(\hat{\theta}) - \Phi(\theta)$?

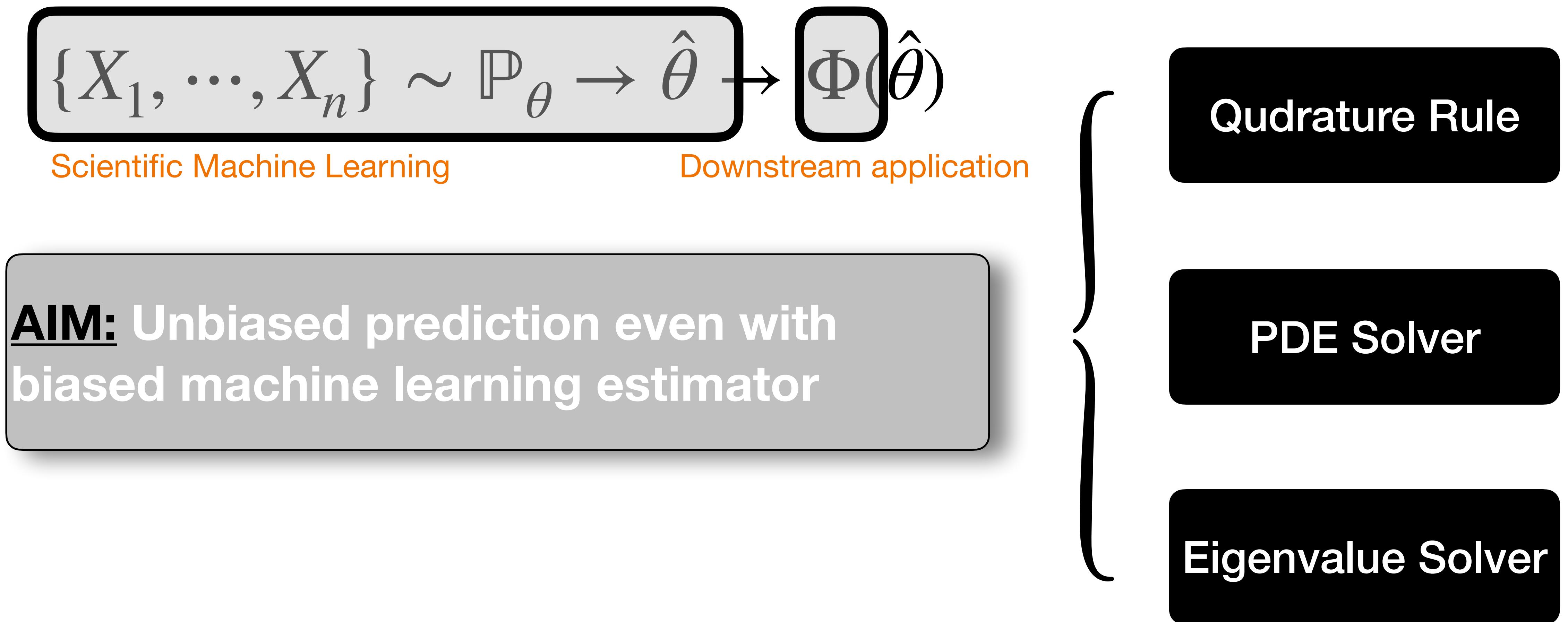


Physics-Informed! (Structure of Φ)

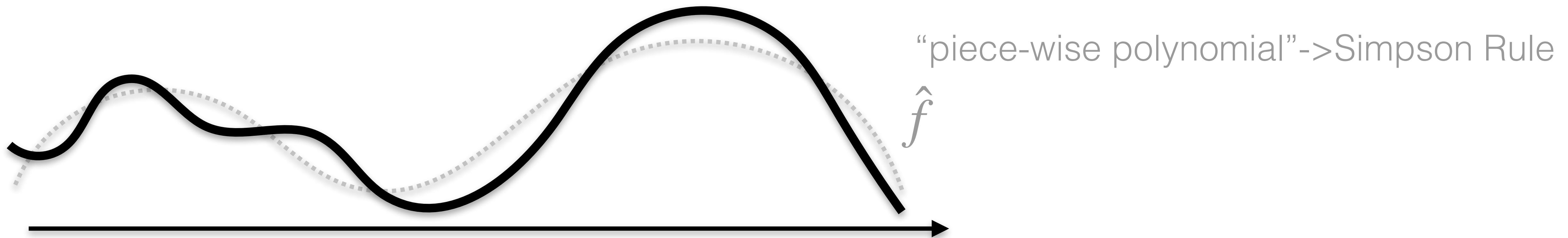
Why it is easier than directly estimate $\Phi(\theta)$?

Variance Reduction

Our Framework



Debiasing a Machine Learning Solution



$$\{X_1, \dots, X_n\} \sim \mathbb{P}_\theta \rightarrow \hat{\theta} \rightarrow \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

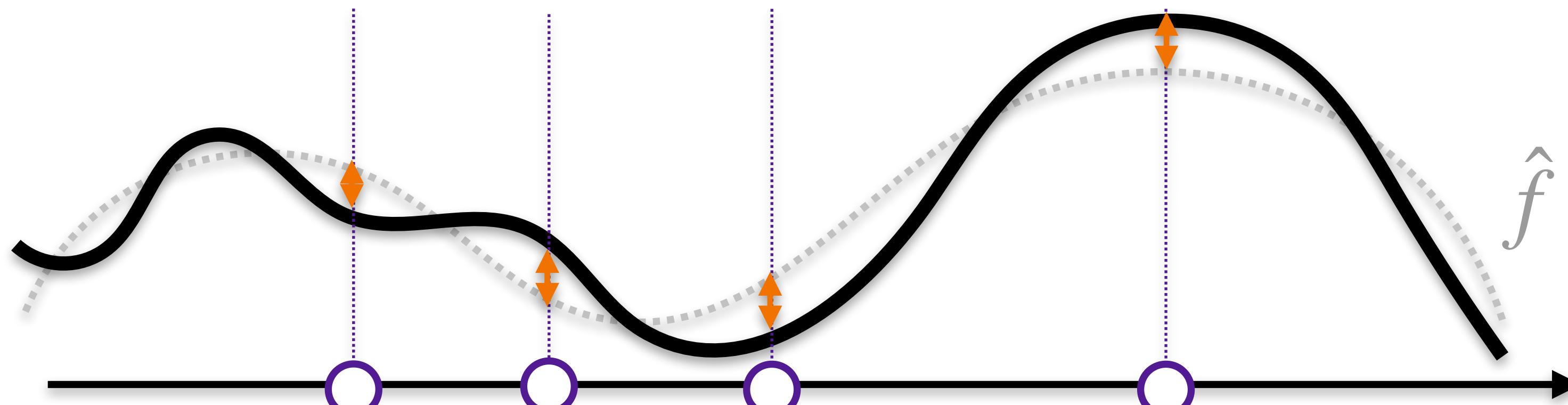
Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \int f^q(x) dx$$

Temperature, overall velocity...

Debiasing a Machine Learning Solution



Our Approach

$$\text{Estimate } \mathbb{E}_P f \approx \mathbb{E}_P \hat{f} + \mathbb{E}_{\hat{P}} f - \hat{f}$$

An estimate to $\Phi(\hat{\theta}) - \Phi(\theta)$



Scientific Machine Learning

Downstream application

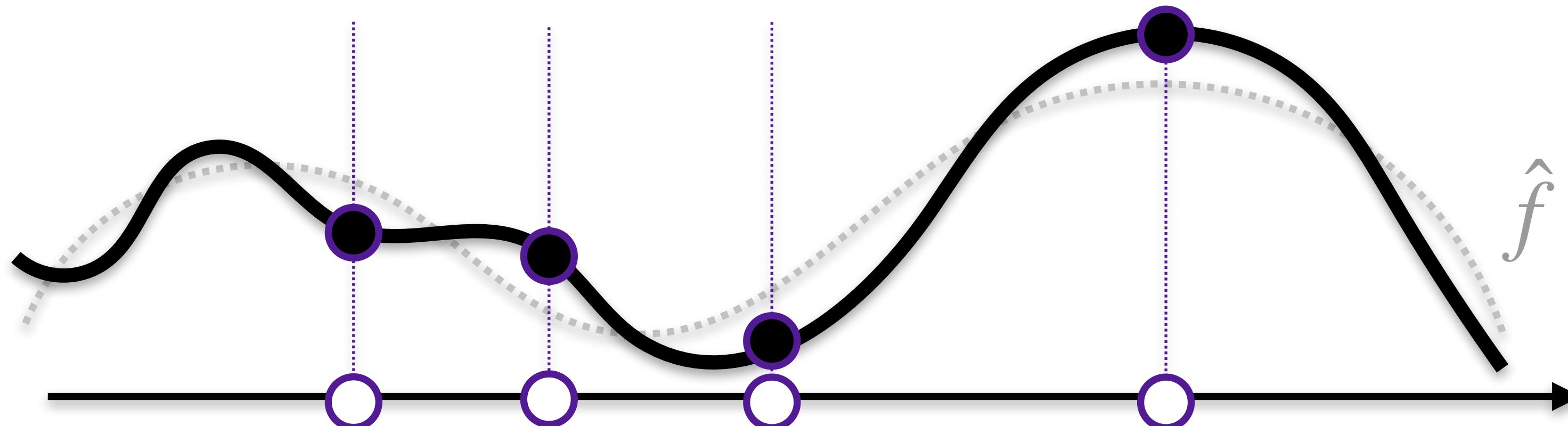
Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \int f^q(x) dx$$

Temperature, overall velocity...

Debiasing a Machine Learning Solution



Monte Carlo?

Estimate $\mathbb{E}_P f \approx \mathbb{E}_{\hat{P}} \hat{f}$



Scientific Machine Learning

Downstream application

Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \int f^q(x) dx$$

Temperature, overall velocity...

Debiasing a Machine Learning Solution

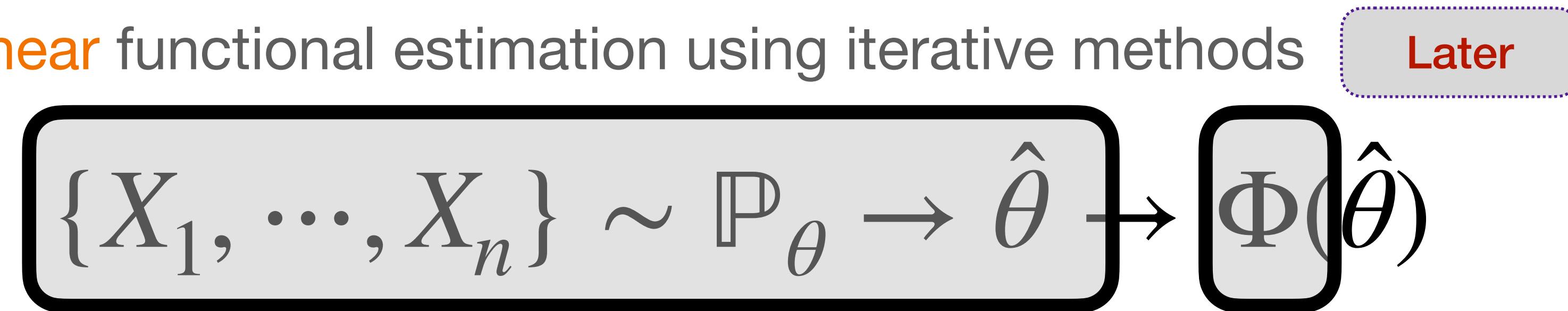


Regression-adjusted Control Variates

Doubly Robust Estimator

...

- Investigated the **optimality** of the SCaSML Framework
 - Jose Blanchet, Haoxuan Chen, Yiping Lu, Lexing Ying. When can Regression-Adjusted Control Variates Help? Rare Events, Sobolev Embedding and Minimax Optimality Neurips 2023
- Extend to **nonlinear** functional estimation using iterative methods Later



Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \int f^q(x) dx$$

Temperature, overall velocity...

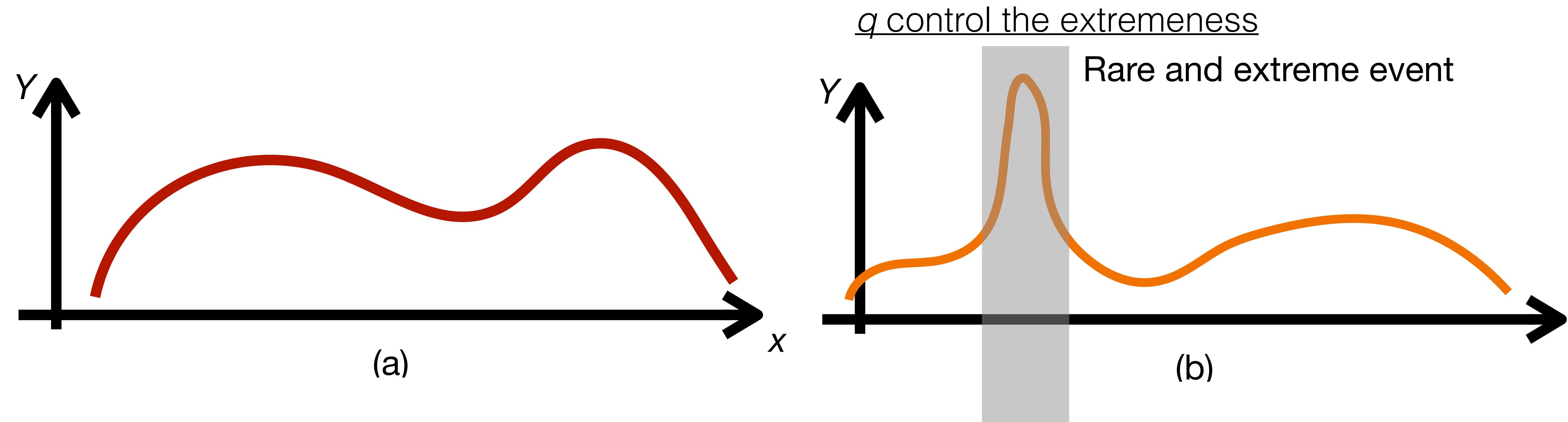
| Take Home Message on the Theory



SCaML

- a) Statistical optimal regression is the optimal control variate
- b) It helps only if there isn't a hard to simulate (infinite variance)

Rare and extreme event



When can Regression-Adjusted Control Variates Help? Rare Events, Sobolev Embedding and Minimax Optimality

Jose Blanchet

Department of MS&E and ICME
Stanford University
Stanford, CA 94305
jose.blanchet@stanford.edu

Haoxuan Chen

ICME
Stanford University
Stanford, CA 94305
haoxuanc@stanford.edu

Yiping Lu

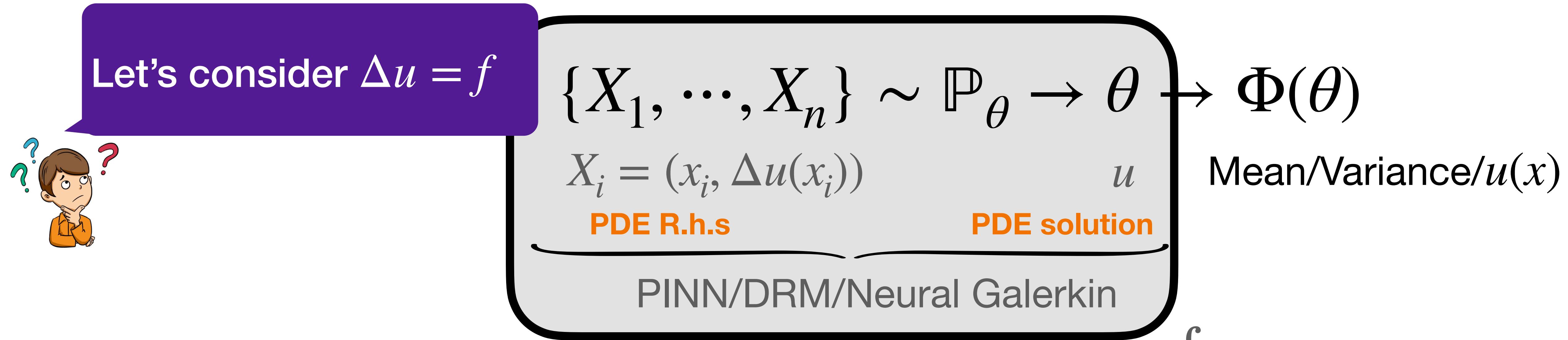
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012
yiping.lu@nyu.edu

Lexing Ying

Department of Mathematics and ICME
Stanford University
Stanford, CA 94305
lexing@stanford.edu

PDE Solver

High Dimensional PDE-Solving

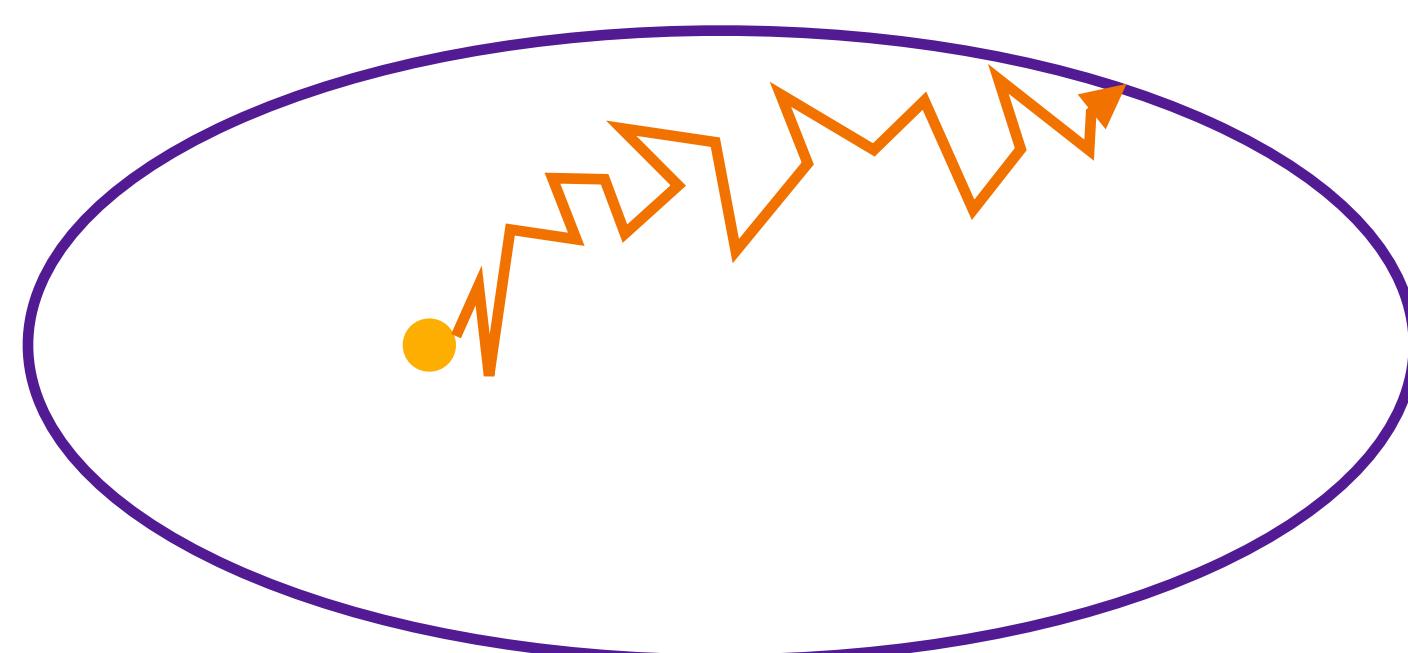


$$\Delta u = f$$

$$\Delta \hat{u} = \hat{f}$$

$$\Delta(u - \hat{u}) = f - \hat{f}$$

$$(u - \hat{u})(x) = \mathbb{E} \int (f - \hat{f})(X_t) dt$$



Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x, t) + \boxed{\Delta U(x, t)} + f(U(x, t)) = 0$$

Keeps the structure to enable brownian motion simulation



Can you do simulation
for nonlinear equation?



Δ is linear!

Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x, t) + \boxed{\Delta U(x, t)} + f(U(x, t)) = 0$$

Keeps the structure to enable brownian motion simulation

$$\frac{\partial \hat{U}}{\partial t}(x, t) + \boxed{\Delta \hat{U}(x, t)} + f(\hat{U}(x, t)) = g(x, t)$$

$g(x, t)$ is the error made by NN

Works for Semi-linear PDE

$$\frac{\partial U}{\partial t}(x, t) + \boxed{\Delta U(x, t)} + f(U(x, t)) = 0$$

Keeps the structure to enable brownian motion simulation

$$\frac{\partial \hat{U}}{\partial t}(x, t) + \boxed{\Delta \hat{U}(x, t)} + f(\hat{U}(x, t)) = g(x, t)$$

NN

g(x, t) is the error made by NN

Subtract two equations

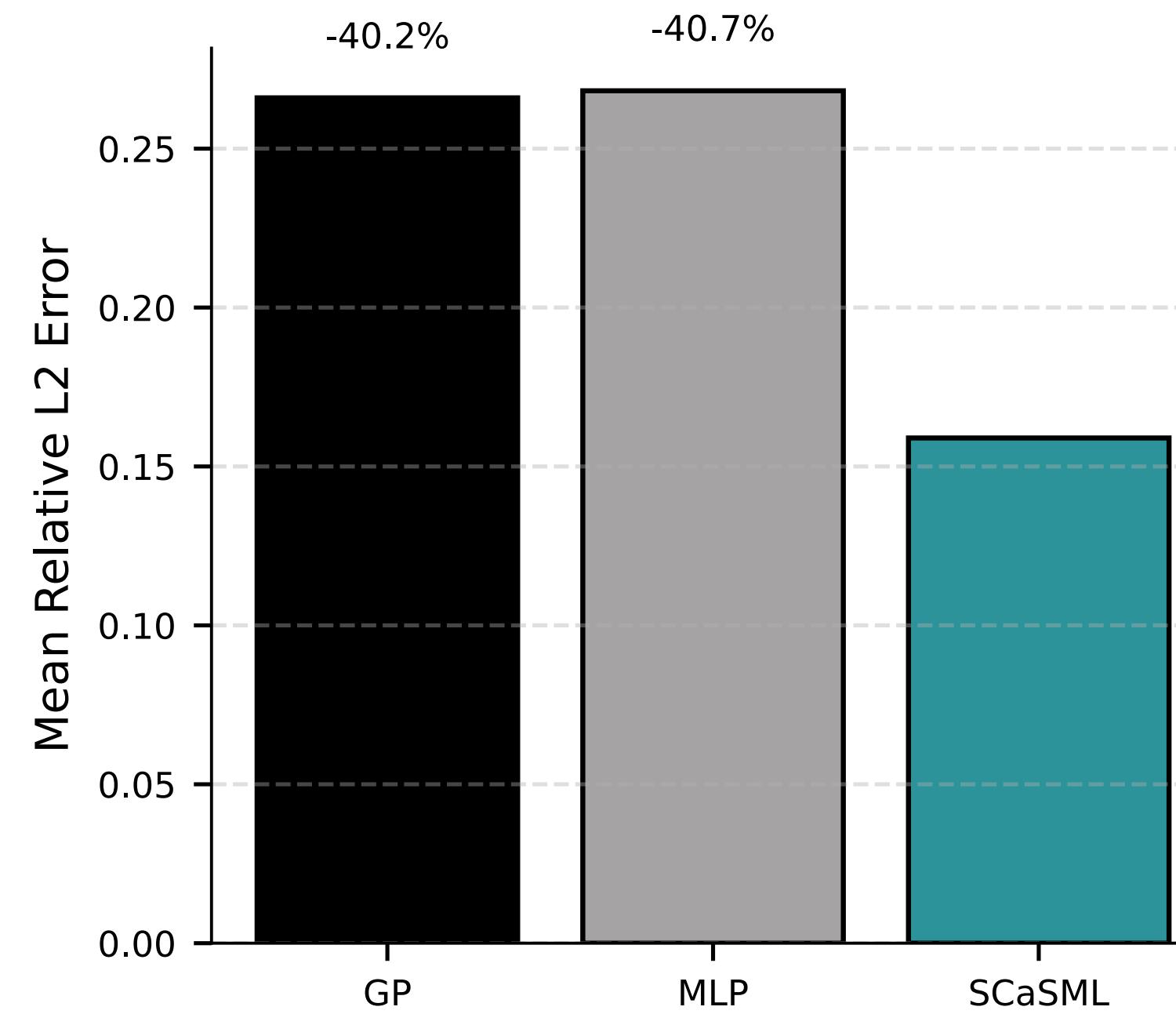
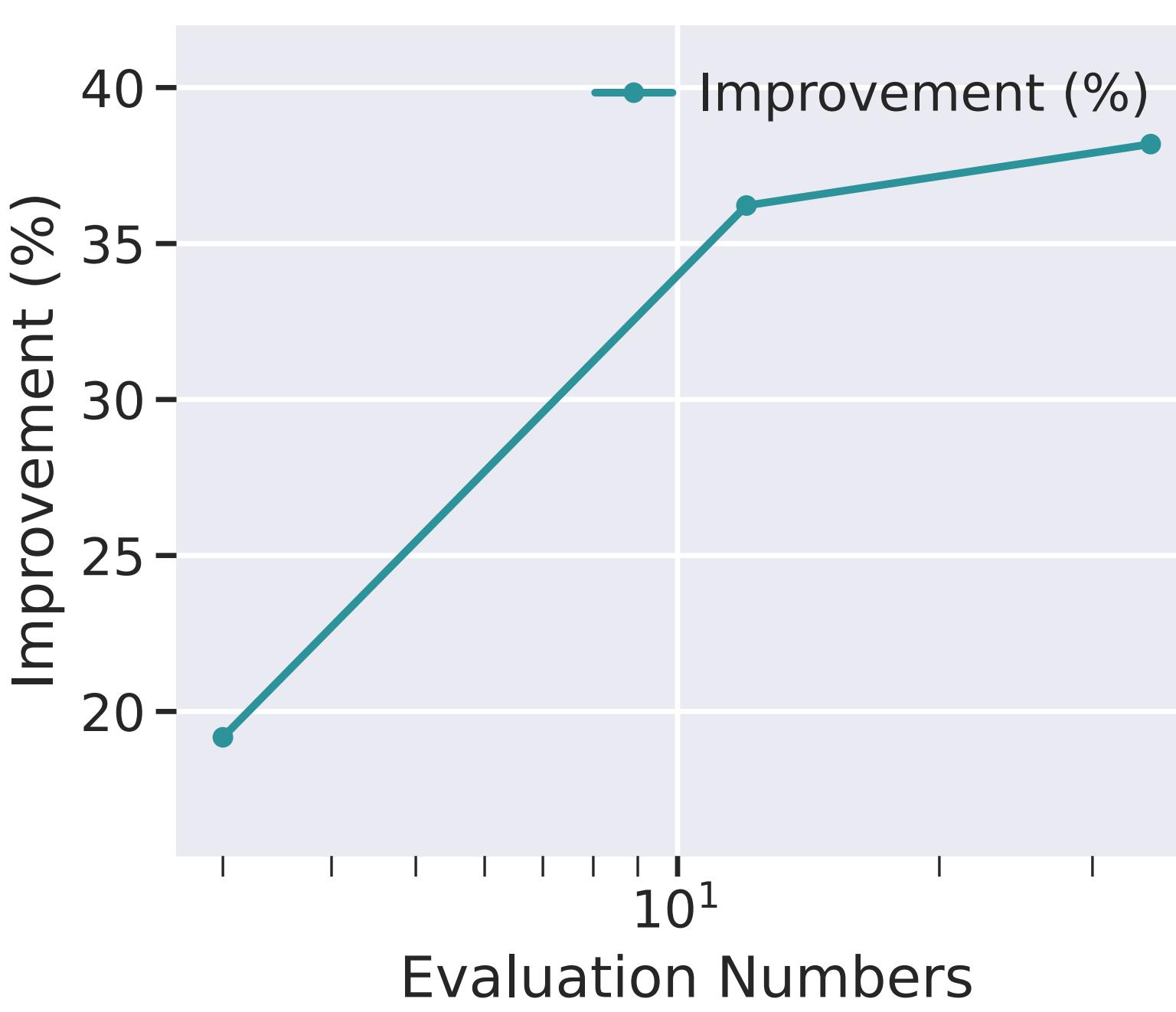
$$\frac{\partial(U - \hat{U})}{\partial t}(x, t) + \boxed{\Delta(U - \hat{U})(x, t)} + \underbrace{f(t, \hat{U}(x, t) + U(x, t) - \hat{U}(x, t)) - f(t, \hat{U}(x, t))}_{G(t, (U - \hat{U})(x, t))} = g(x, t).$$

Keeps the linear structure

Inference-Time Scaling

$$\frac{\partial}{\partial t} u + \left[\sigma^2 u - \frac{1}{d} - \frac{\bar{\sigma}^2}{2} \right] (\nabla \cdot u) + \frac{\bar{\sigma}^2}{2} \Delta u = 0$$

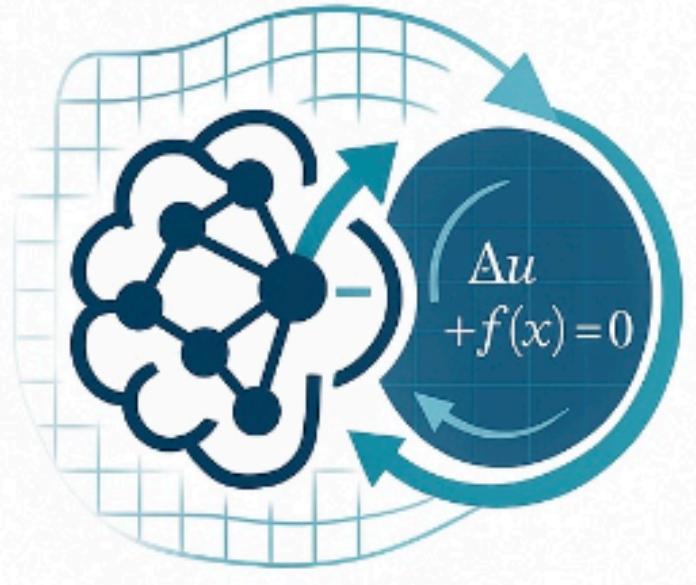
have closed-form solution $g(x) = \frac{\exp(T + \sum_i x_i)}{1 + \exp(T + \sum_i x_i)}$



Method	Convergence Rate
PINN	$O(n^{-s/d})$
MLP	$O(n^{-1/4})$
SCaSML	$O(n^{-1/4-s/d})$

Numerical Results

		Time (s)			Relative L^2 Error			L^∞ Error			L^1 Error		
		SR	MLP	SCaSML	SR	MLP	SCaSML	SR	MLP	SCaSML	SR	MLP	SCaSML
LCD	10d	2.64	11.24	23.75	5.24E-02	2.27E-01	2.73E-02	2.50E-01	9.06E-01	1.61E-01	3.43E-02	1.67E-01	1.78E-02
	20d	1.14	7.35	17.59	9.09E-02	2.35E-01	4.73E-02	4.52E-01	1.35E+00	3.28E-01	9.47E-02	2.37E-01	4.52E-02
	30d	1.39	7.52	25.33	2.30E-01	2.38E-01	1.84E-01	4.73E+00	1.59E+00	1.49E+00	1.75E-01	2.84E-01	1.91E-01
	60d	1.13	7.76	35.58	3.07E-01	2.39E-01	1.32E-01	3.23E+00	2.05E+00	1.55E+00	5.24E-01	4.07E-01	2.06E-01
VB-PINN	20d	1.15	7.05	13.82	1.17E-02	8.36E-02	3.97E-03	3.16E-02	2.96E-01	2.16E-02	5.37E-03	3.39E-02	1.29E-03
	40d	1.18	7.49	16.48	3.99E-02	1.04E-01	2.85E-02	8.16E-02	3.57E-01	7.16E-02	1.97E-02	4.36E-02	1.21E-02
	60d	1.19	7.57	19.83	3.97E-02	1.17E-01	2.90E-02	8.10E-02	3.93E-01	7.10E-02	1.95E-02	4.82E-02	1.24E-02
	80d	1.32	7.48	21.99	6.78E-02	1.19E-01	5.68E-02	1.89E-01	3.35E-01	1.79E-01	3.24E-02	4.73E-02	2.49E-02
VB-GP	20d	1.97	10.66	65.46	1.47E-01	8.32E-02	5.52E-02	3.54E-01	2.22E-01	2.54E-01	7.01E-02	3.50E-02	1.91E-02
	40d	1.68	10.14	49.38	1.81E-01	1.05E-01	7.95E-02	4.01E-01	3.47E-01	3.01E-01	9.19E-02	4.25E-02	3.43E-02
	60d	1.01	7.25	35.14	2.40E-01	2.57E-01	1.28E-01	3.84E-01	9.50E-01	7.10E-02	1.27E-01	9.99E-02	6.11E-02
	80d	1.00	7.00	38.26	2.66E-01	3.02E-01	1.52E-01	3.62E-01	1.91E+00	2.62E-01	1.45E-01	1.09E-01	7.59E-02
LQG	100d	1.54	8.67	26.95	7.96E-02	5.63E+00	5.51E-02	7.78E-01	1.26E+01	6.78E-01	1.40E-01	1.21E+01	8.68E-02
	120d	1.25	8.17	27.46	9.37E-02	5.50E+00	6.64E-02	9.02E-01	1.27E+01	8.02E-01	1.73E-01	1.22E+01	1.05E-01
	140d	1.80	8.27	29.72	9.79E-02	5.37E+00	6.78E-02	1.00E+00	1.27E+01	9.00E-01	1.91E-01	1.23E+01	1.11E-01
	160d	1.74	9.07	32.08	1.11E-01	5.27E+00	9.92E-02	1.38E+00	1.28E+01	1.28E+00	2.15E-01	1.23E+01	1.79E-01
DR	100d	1.62	7.75	60.86	9.52E-03	8.99E-02	8.87E-03	7.51E-02	6.37E-01	6.51E-02	1.13E-02	9.74E-02	1.11E-02
	120d	1.26	7.28	65.66	1.11E-02	9.13E-02	9.90E-03	7.10E-02	5.74E-01	6.10E-02	1.40E-02	9.97E-02	1.23E-02
	140d	2.38	7.82	76.90	3.17E-02	8.97E-02	2.94E-02	1.79E-01	8.56E-01	1.69E-01	3.96E-02	9.77E-02	3.67E-02
	160d	1.75	7.42	82.40	3.46E-02	9.00E-02	3.23E-02	2.08E-01	8.02E-01	1.98E-01	4.32E-02	9.75E-02	4.02E-02



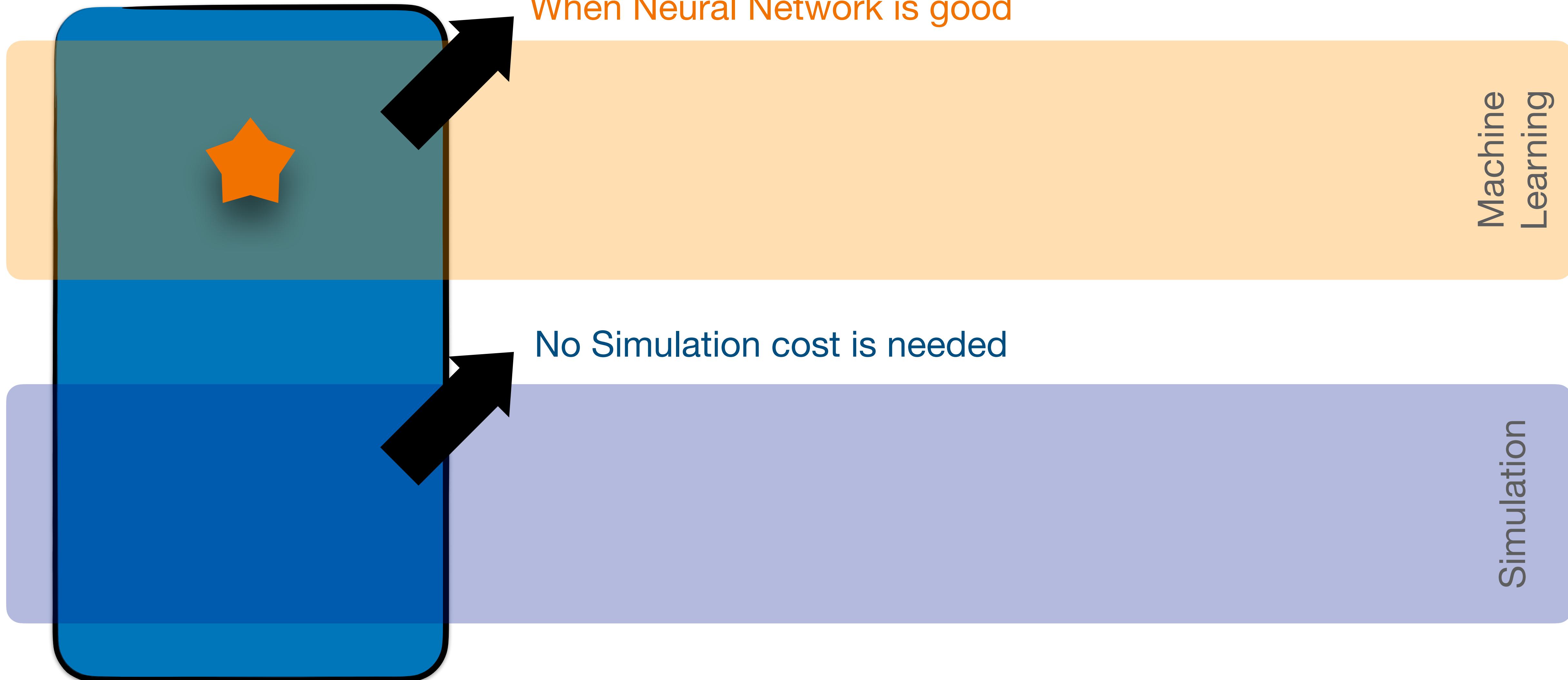
Physics-Informed Inference Time Scaling via Simulation-Calibrated Scientific Machine Learning

Zexi Fan¹, Yan Sun², Shihao Yang³, Yiping Lu^{*4}

¹ Peking University ² Visa Inc. ³ Georgia Institute of Technology ⁴ Northwestern University
fanzexi_francis@stu.pku.edu.cn, yansun414@gmail.com,
shihao.yang@isye.gatech.edu, yiping.lu@northwestern.edu

https://2prime.github.io/files/scasml_techreport.pdf

Our Aim Today : A Marriage



Our Aim Today : A Marriage

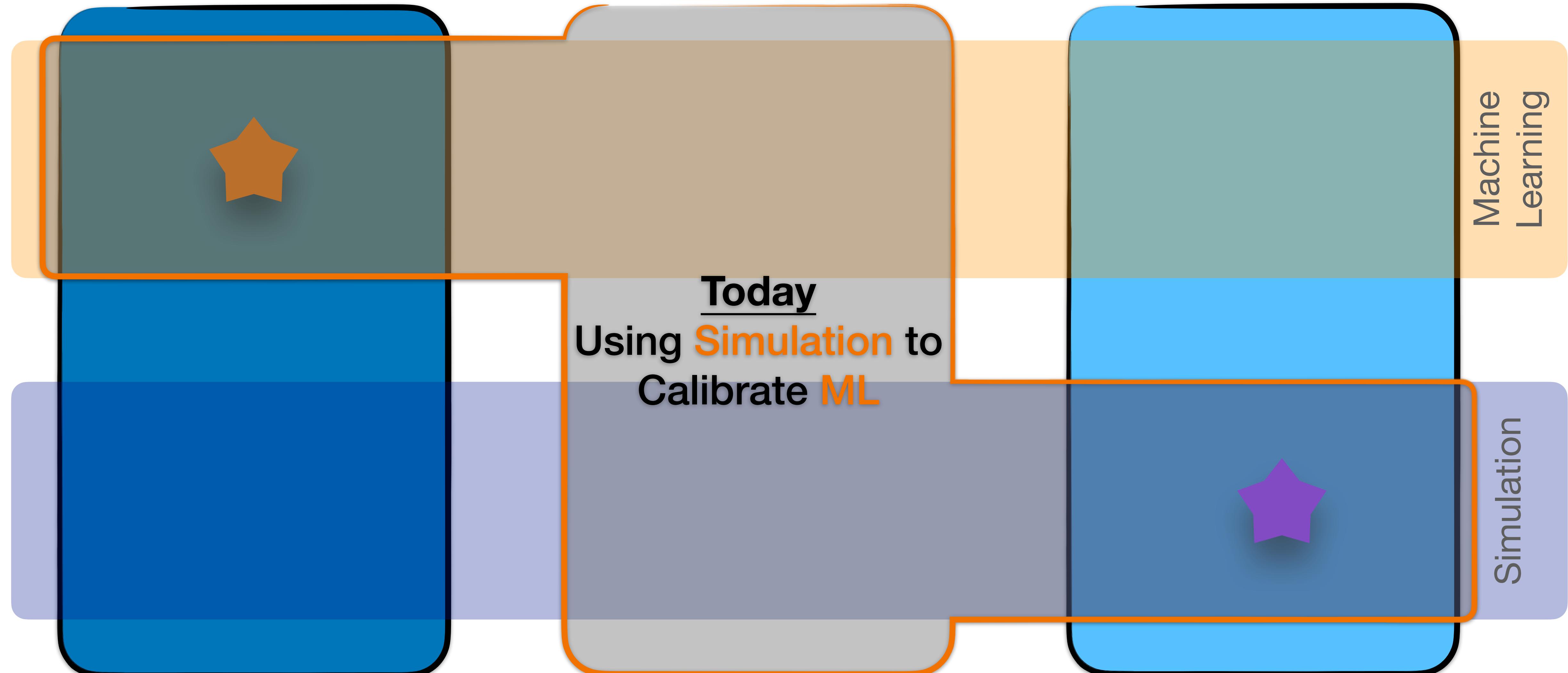
When Neural Network is bad

Provide pure Simulation solution

Machine
Learning

Simulation

Our Aim Today : A Marriage



More Examples...



Scientific Machine Learning

Downstream application

Example 1

$$\theta = f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \int f^q(x) dx$$

Example 2

$$\theta = \Delta^{-1}f, \quad X_i = (x_i, f(x_i))$$

$$\Phi(\theta) = \theta(x)$$

Example 3

$$\theta = A, \quad X_i = (x_i, Ax_i)$$

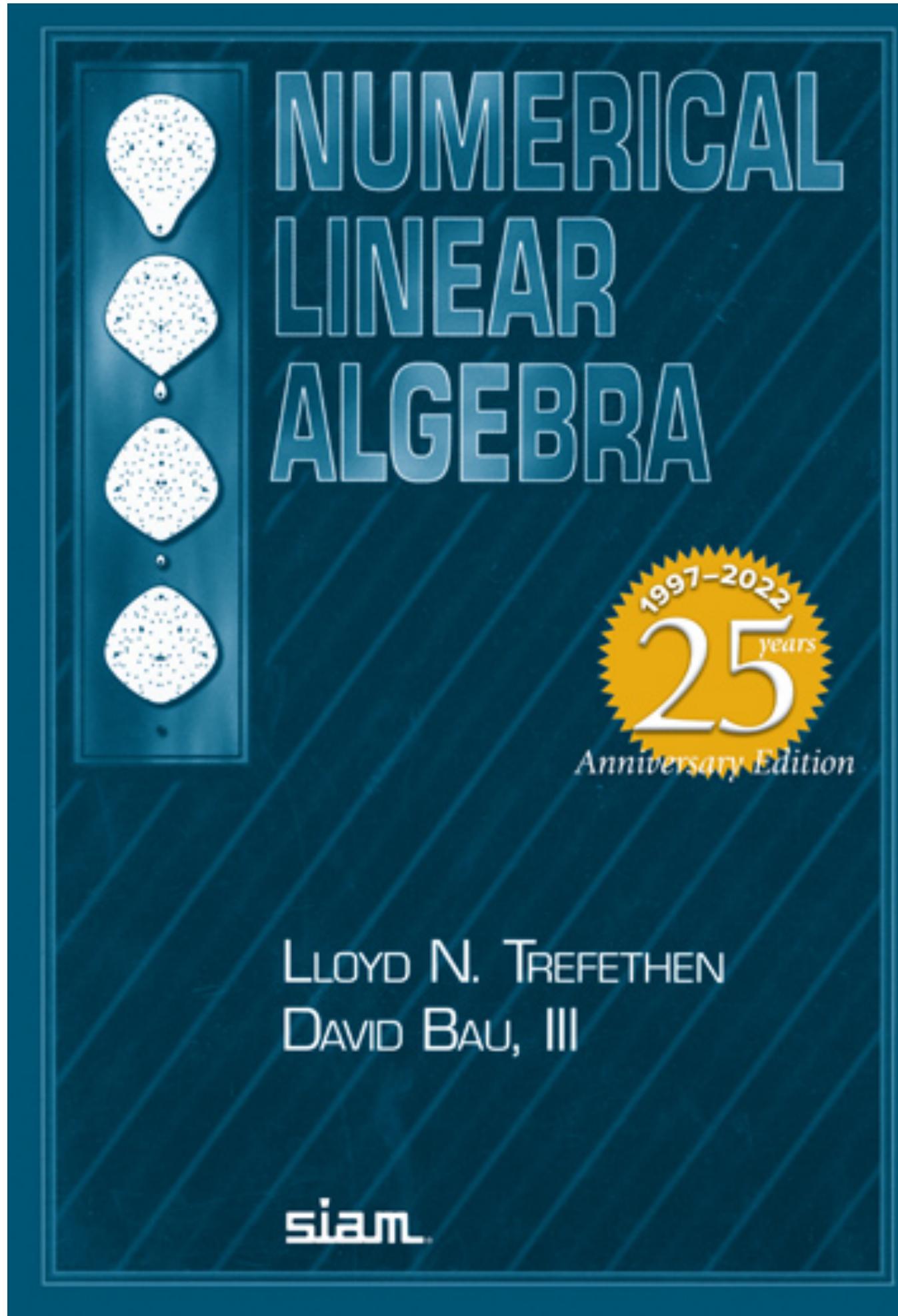
Estimation \hat{A} via Randomized SVD

$$\Phi(\theta) = \text{tr}(A)$$

Estimate $\text{tr}(A - \hat{A})$ via Hutchinson's estimator

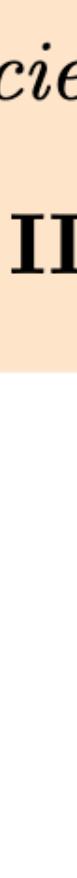
Tale 2: Pre-condition with a surprising connection with debiasing

Tale 2: Preconditioning



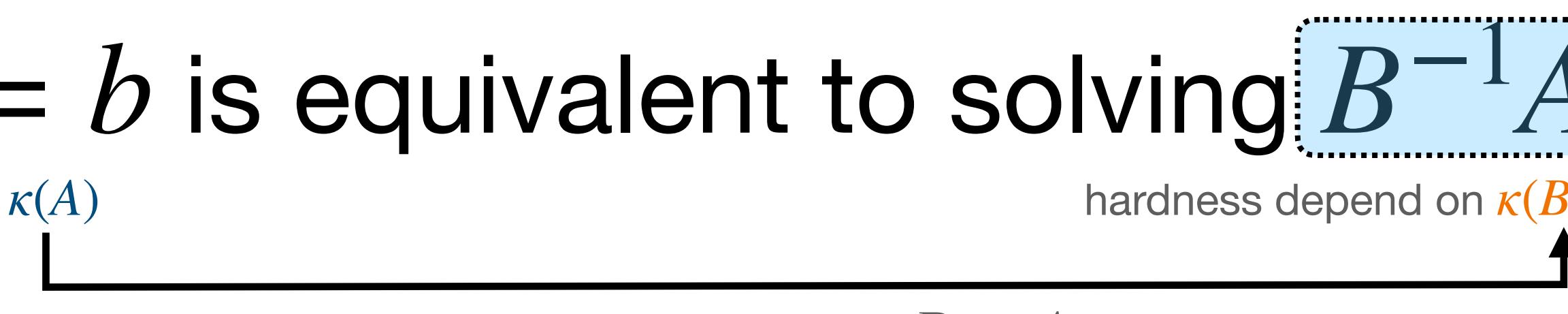
"In ending this book with the subject of preconditioners, we find ourselves at the philosophical center of the scientific computing of the future."

— L. N. Trefethen and D. Bau III, Numerical Linear Algebra [TB22]



Nothing will be more central to computational science in the next century than **the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly**.

What is precondition

- Solving $Ax = b$ is equivalent to solving $B^{-1}Ax = B^{-1}b$
hardness depend on $\kappa(A)$ hardness depend on $\kappa(B^{-1}A)$


Become easier when $B \approx A$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
 - Using an approximate solver $Bx_1 = b$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
 - Using an approximate solver $Bx_1 = b$
 - $x - x_1$ satisfies the equation $A(x - x_1) = b - Ax_1$
 - Using the approximate solver to approximate $x - x_1$ via $Bx_2 = b - Ax_1$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$
 - Using an approximate solver $Bx_1 = b$

Iterative Refinement Algorithm

- $x - \sum_{i=1}^t x_i$ satisfies the equation $A(x - \sum_{i=1}^t x_i) = b - A \sum_{i=1}^t x_i$
- Using the approximate solver to approximate $x - \sum_{i=1}^t x_i$ via $Bx_{i+1} = b - A \sum_{i=1}^t x_i$

A New Way to Implement Precondition

- Debiasing is a way of solving $Ax = b$

- Using an approximate solver $Bx_1 = b$

Iterative Refinement Algorithm

. $x - \sum_{i=1}^t x_i$ satisfies the equation $A(x - \sum_{i=1}^t x_i) = b - A \sum_{i=1}^t x_i$

. Using the approximate solver to approximate $x - \sum_{i=1}^t x_i$ via $Bx_{i+1} = b - A \sum_{i=1}^t x_i$

$$x_{i+1} = (I - B^{-1}A)x_i + B^{-1}b$$

Preconditioned Jacobi Iteration

This Talk: A New Way to Implement Precondition Via Debiasing

- **Step 1:** Aim to solve (potentially nonlinear) equation $A(u) = b$

use Machine Learning

- **Step 2:** Build an approximate solver $A(\hat{u}) \approx b$

Unreliable approximate
solver as preconditioner

- Via machine learning/sketching/finite element....

- **Step 3:** Solve $u - \hat{u}$



AIM: Debiasing a Learned Solution = Using Learned Solution as preconditioner!

Randomized NLA as Machine Learning

AIM: using matrix-vector multiplication to compute eigenvalue/least square problem

$$\underbrace{\{X_1, \dots, X_n\} \sim P_\theta \rightarrow \theta \rightarrow \Phi(\theta)}_{A} \quad \Phi(A) = \begin{cases} A^{-1}b \\ \text{Eigenvalue of } A \end{cases}$$

$X_i = (x, Ax)$

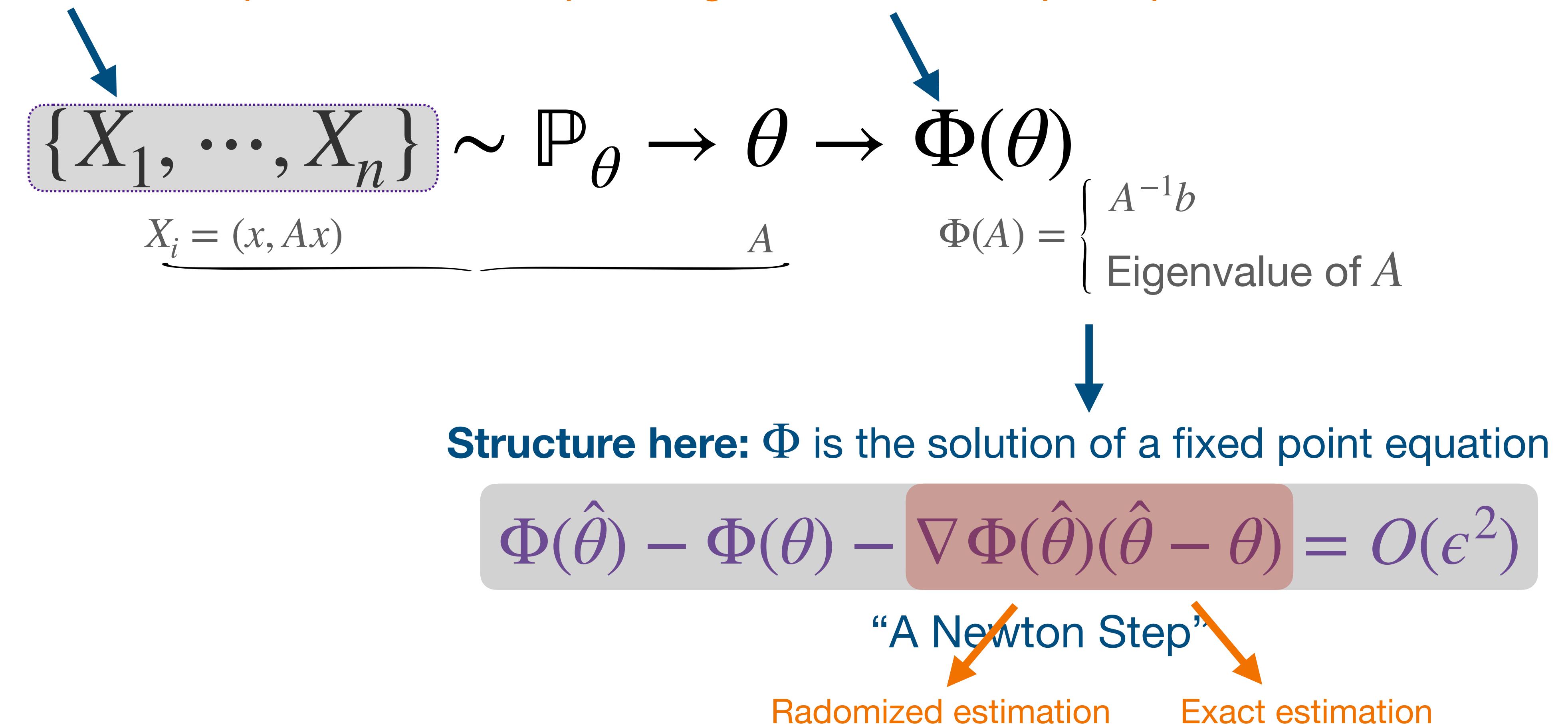
“Randomized Numerical Linear Algebra”/Sketching



“Sketch-and-Solve”

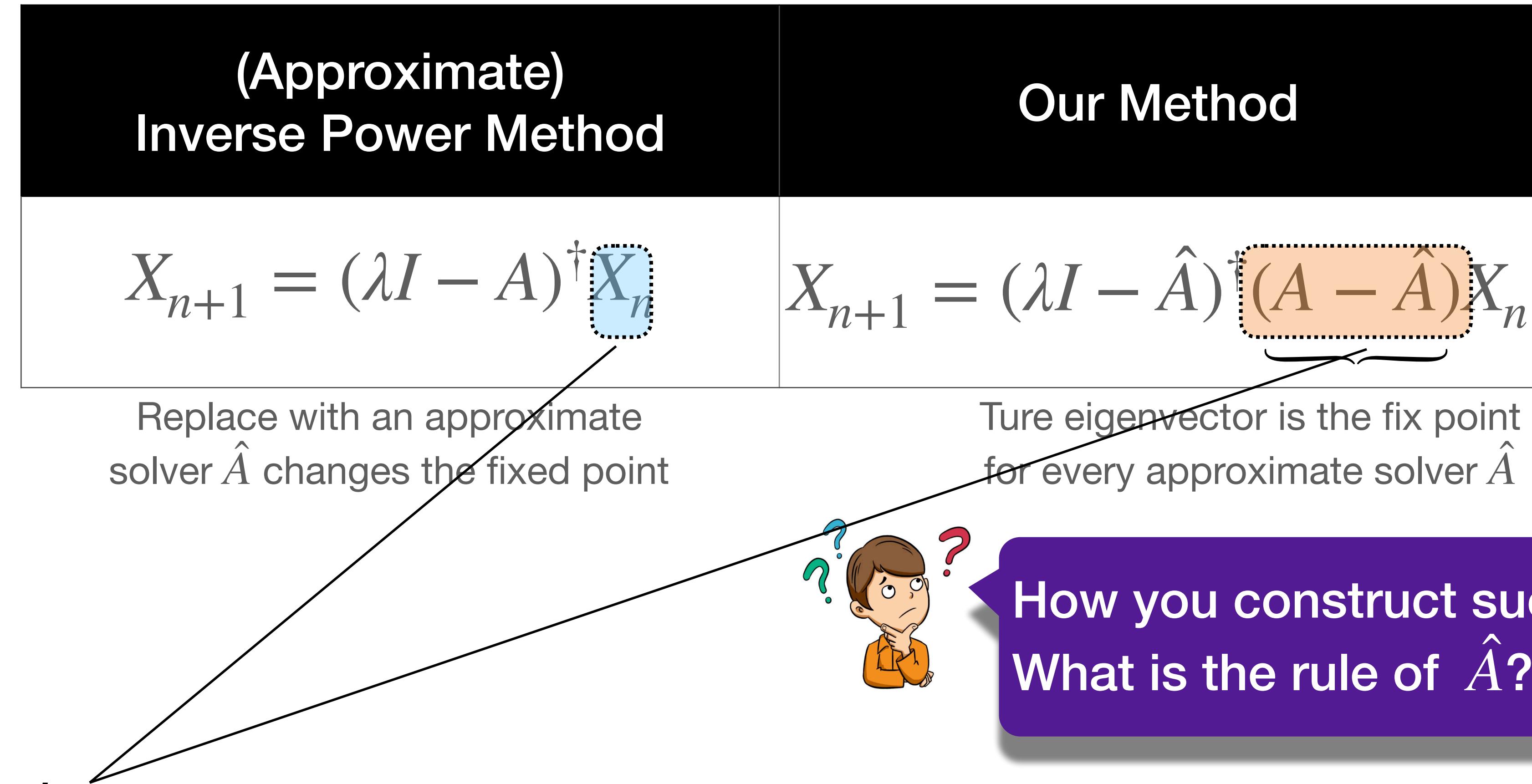
Randomized NLA as Machine Learning

AIM: using matrix-vector multiplication to compute eigenvalue/least square problem



(In)exact Sub-sample Newton Method/Sketch-and-Precondition

Relationship with Inverse Power Methods



Take Home Message 1:

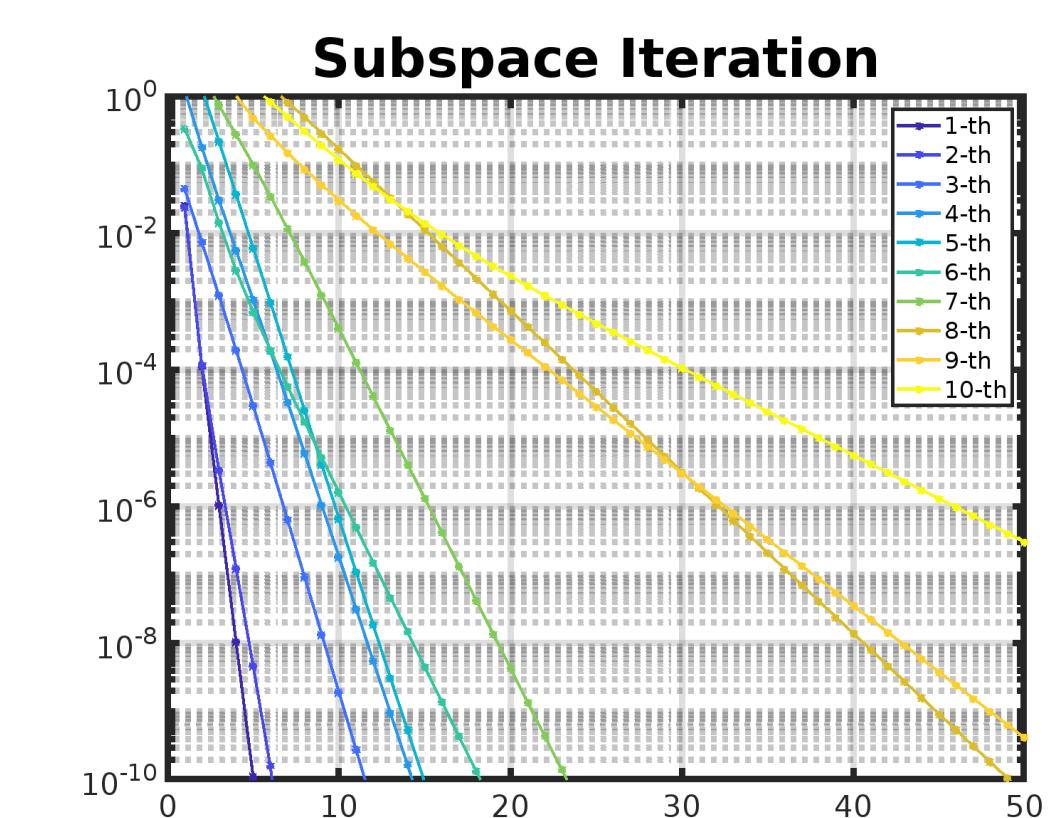
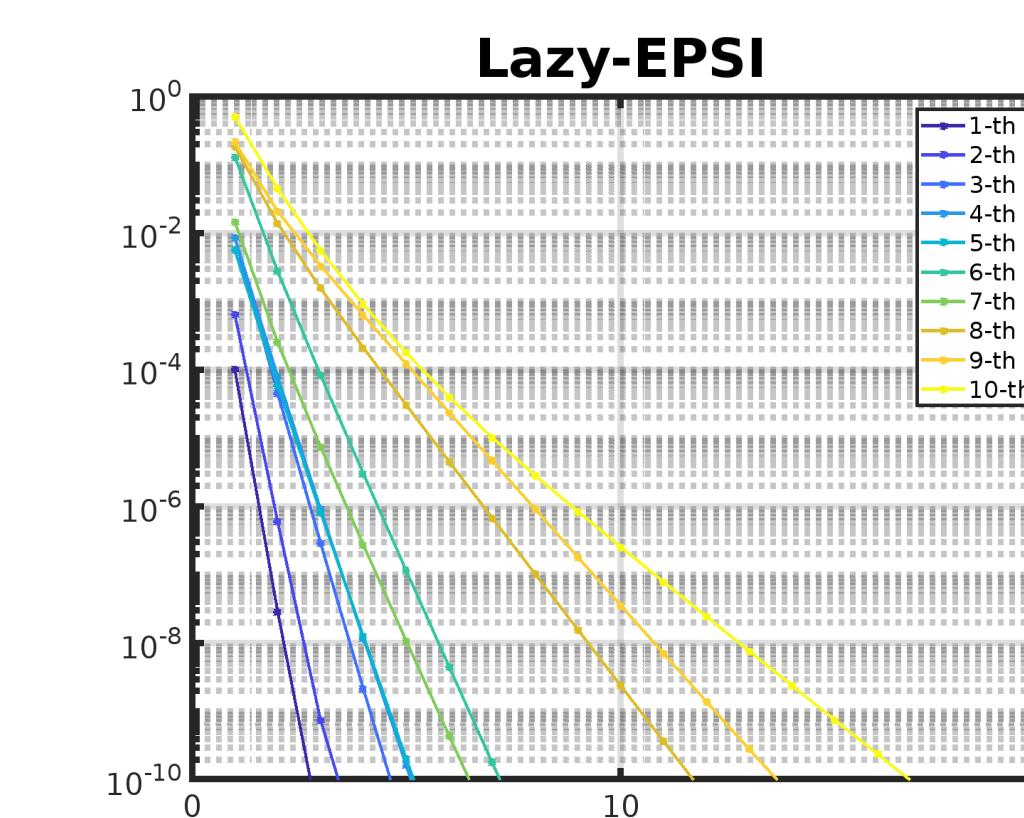
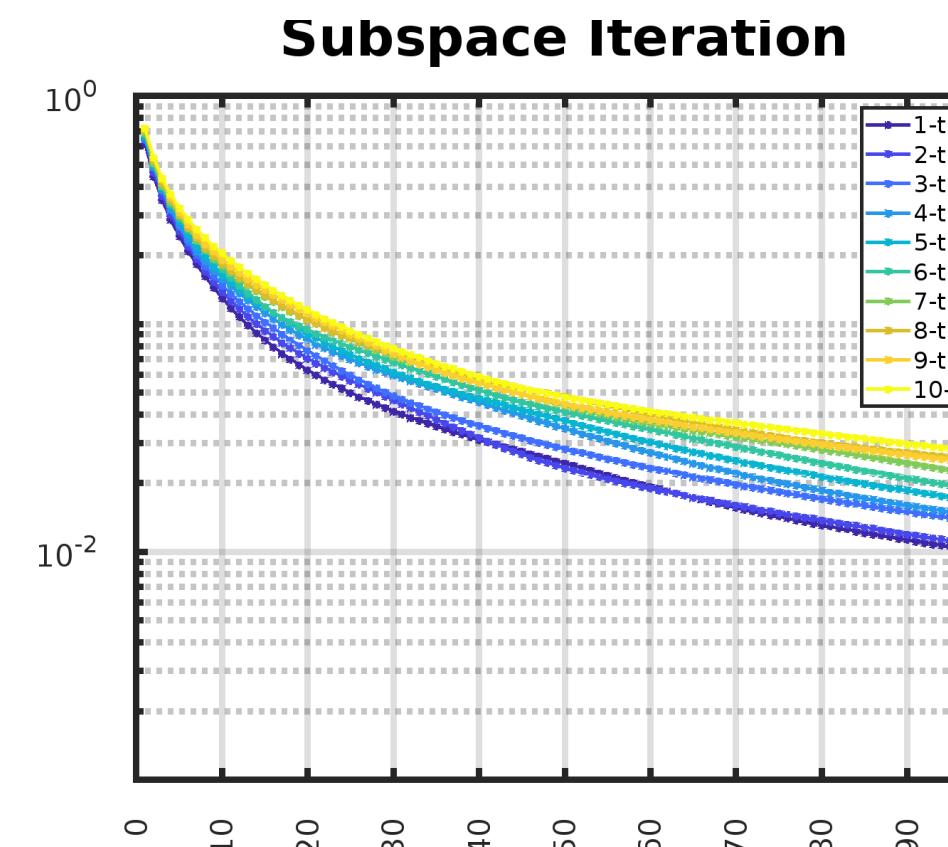
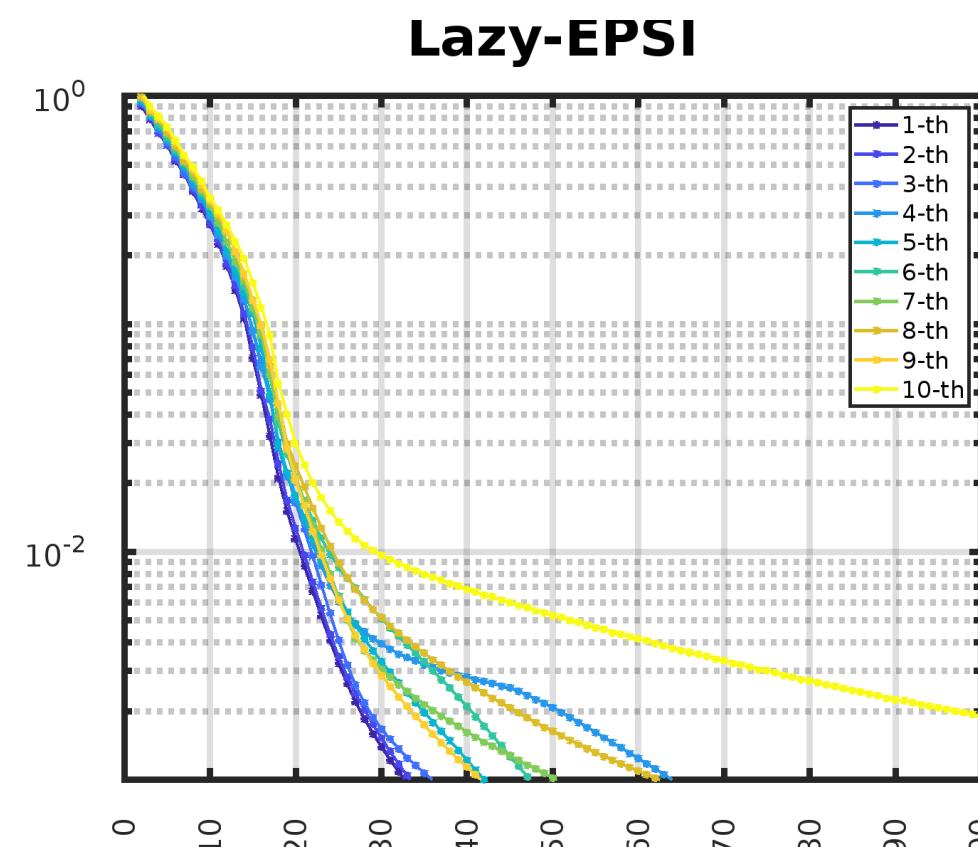
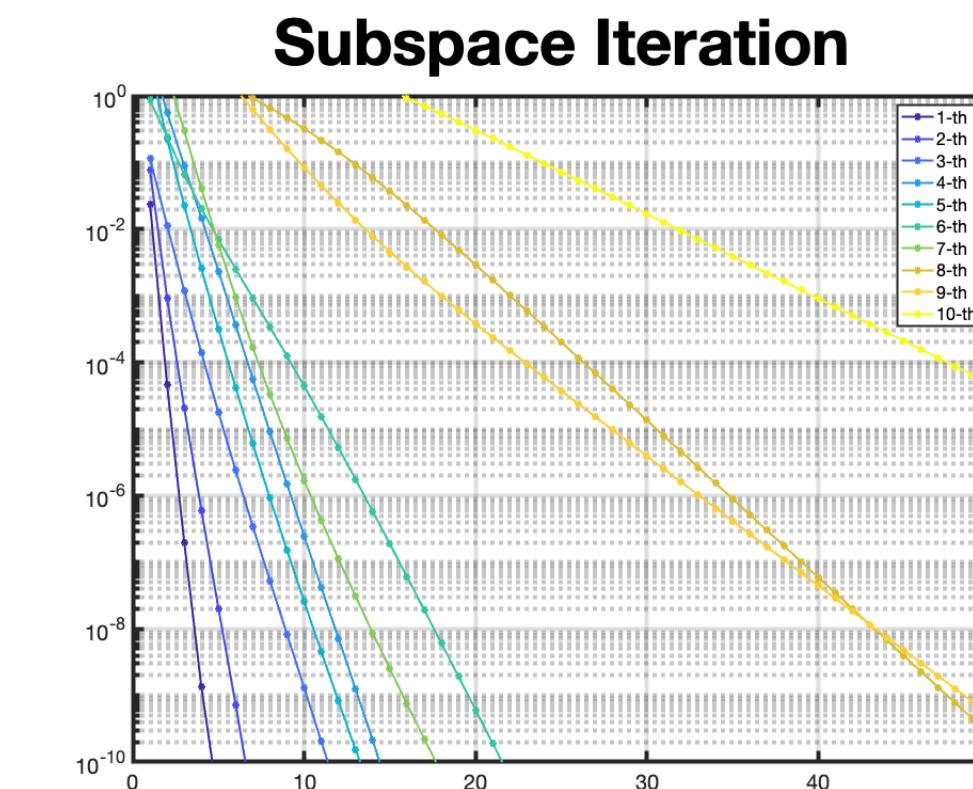
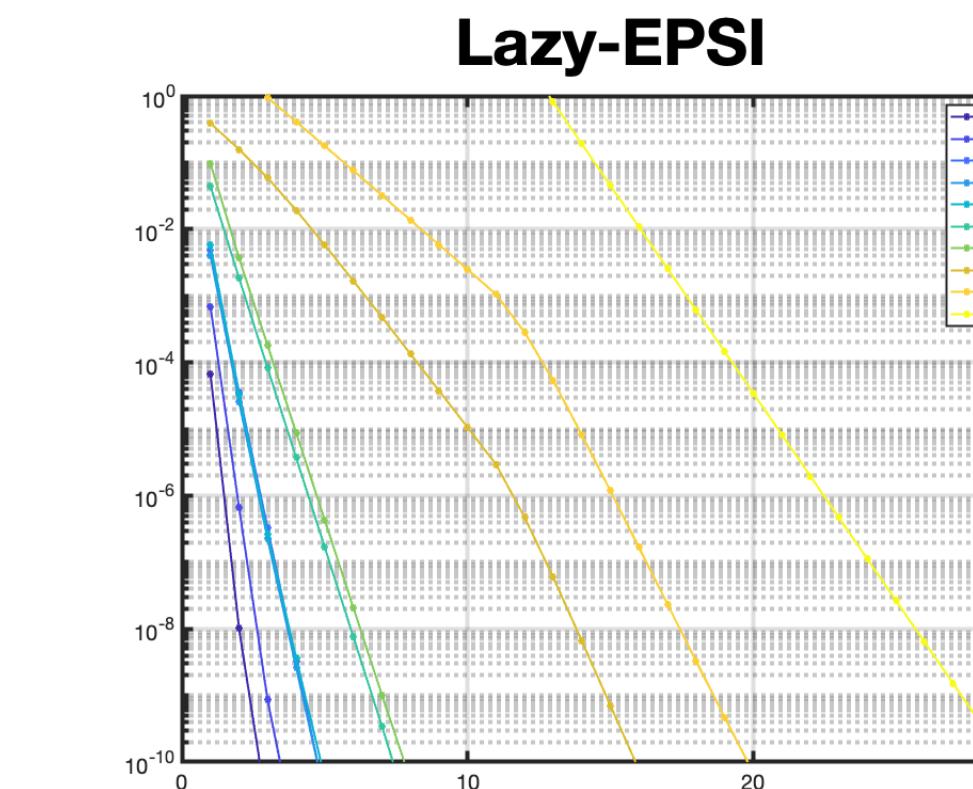
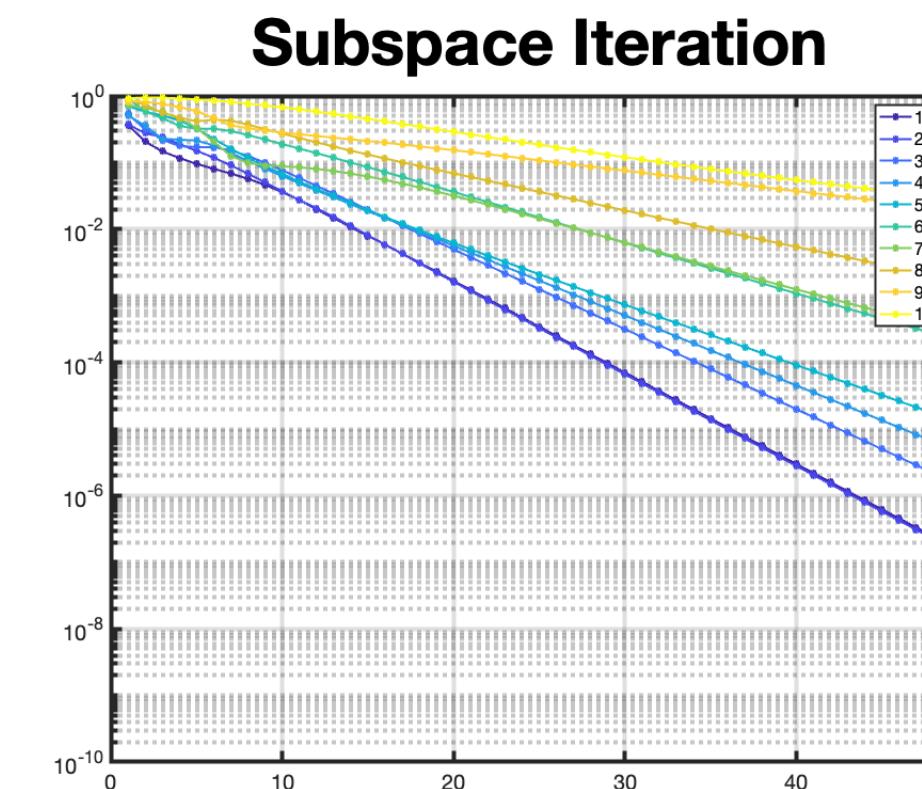
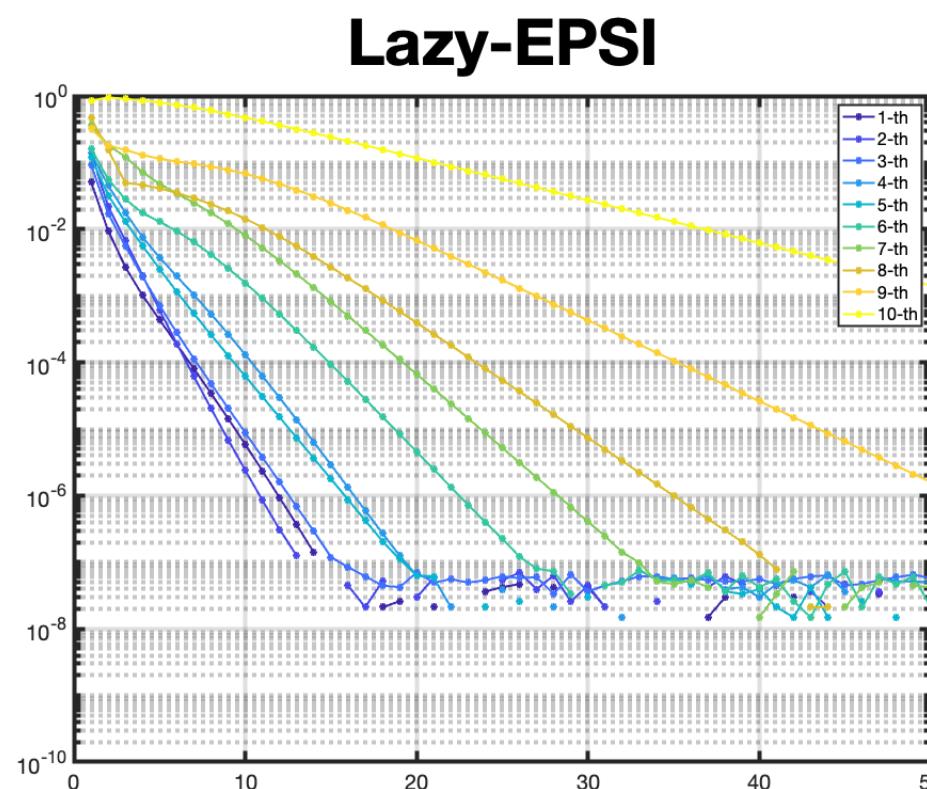
Power the Residual but not Power the vector

Look back to literature

“Sketch-and-Solve” VS “Sketch-and-Precondition”

	Overparameterized Least Square	Low Rank Approximation	
Sketch-and-Solve	[Sar06, DMMS11]	[LWM ⁺ 07, HMST11]	approximate in geometry-preserving, lower-dimensional space.
Randomized Initialization		[Gu15, MM15]	Initialize an (iterative) algorithm at a random point to make a favorable trajectory
Sketch-and-Precondition	[RT08, AMT10] [MSM14, DR24]	<u>This work</u>	boosting solver convergence while preserving the original problem's accuracy

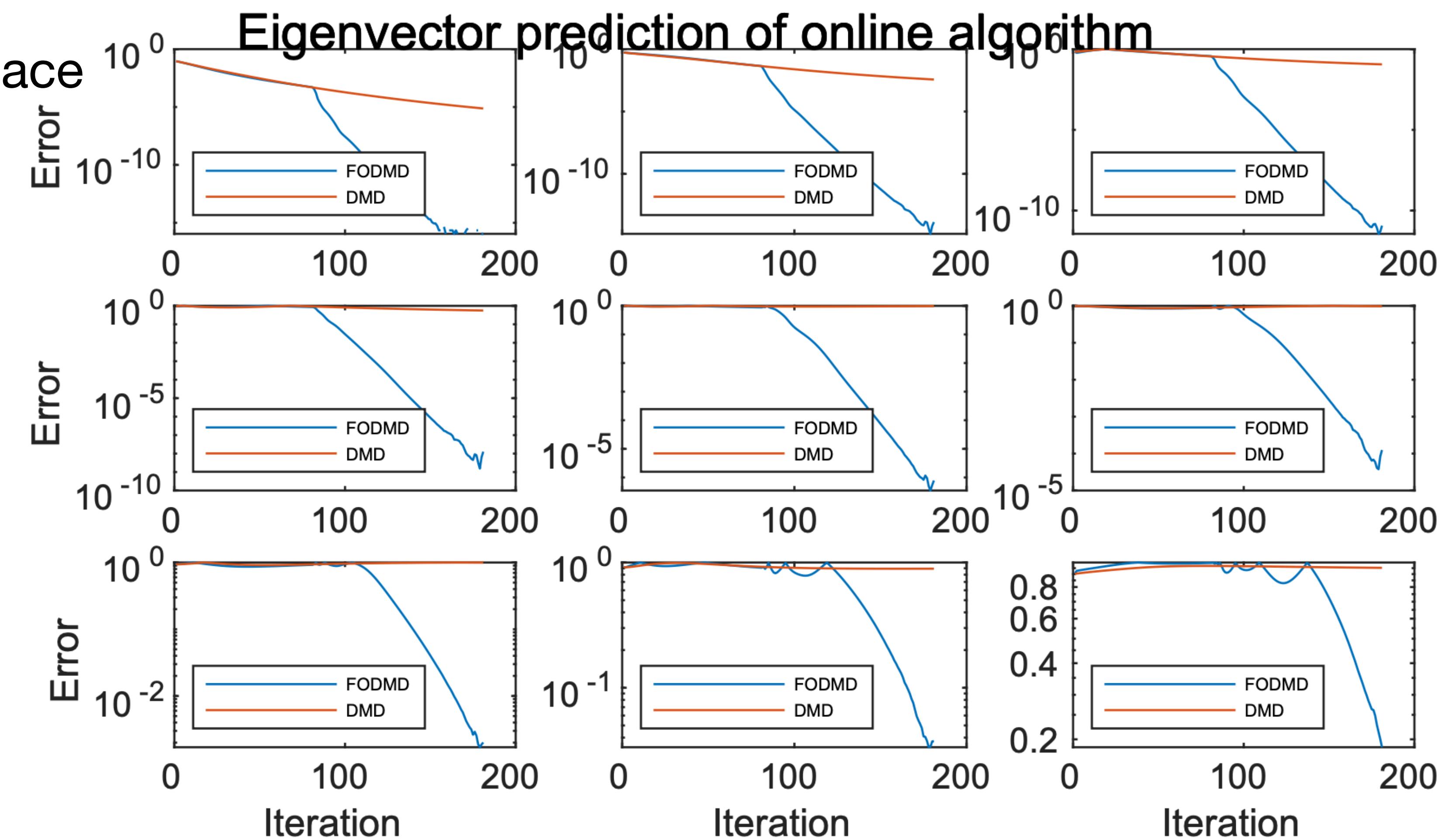
Eigenvalue Computation



Another Supersing Fact...

Iteration lies in the Krylov Subspace

- enable dynamic mode decomposition
- Online fast update
- Much better than DMD



What is a Sketch-and-Precondition Derivation for Low-Rank Approximation? Inverse Power Error or Inverse Power Estimation?

Ruihan Xu *

Yiping Lu †

Abstract

Randomized sketching accelerates large-scale numerical linear algebra by reducing computational complexity. While the traditional sketch-and-solve approach reduces the problem size directly through sketching, the sketch-and-precondition method leverages sketching to construct a computational friendly preconditioner. This preconditioner improves the convergence speed of iterative solvers applied to the original problem, maintaining accuracy in the full space. Furthermore, the convergence rate of the solver improves at least linearly with the sketch size. Despite its potential, developing a sketch-and-precondition framework for randomized algorithms in low-rank matrix approximation remains an open challenge. We introduce the *Error-Powered Sketched Inverse Iteration* (EPSI) Method via run sketched Newton iteration for the Lagrange form as a sketch-and-precondition variant for randomized low-rank approximation. Our method achieves theoretical guarantees, including a convergence rate that improves at least linearly with the sketch size.

One more thing...

Iterative
debiasing

Easier for numerical stability computation
Algorithms can do **online** computation



Newton
Methods

Easier for convergence analysis



Randomized Iterative Solver as Iterative Refinement A Simple Fix Towards Backward Stability

Ruihan Xu
University of Chicago

Yiping Lu
Northwestern University

Abstract

Iterative sketching and sketch-and-precondition are well-established randomized algorithms for solving large-scale over-determined linear least-squares problems. In this paper, we introduce a new perspective that interprets Iterative Sketching and Sketching-and-Precondition as forms of Iterative Refinement. We also examine the numerical stability of two distinct refinement strategies: iterative refinement and recursive refinement, which progressively improve

tive tools for developing approximate matrix factorizations. These methods are remarkable for their simplicity and efficiency, often producing surprisingly accurate results.

In this paper, we consider randomized algorithms to solve the overdetermined linear least-squares problem

$$x = \arg \min_{y \in \mathbb{R}^n} \|b - Ay\| \quad (A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m) \quad (1)$$

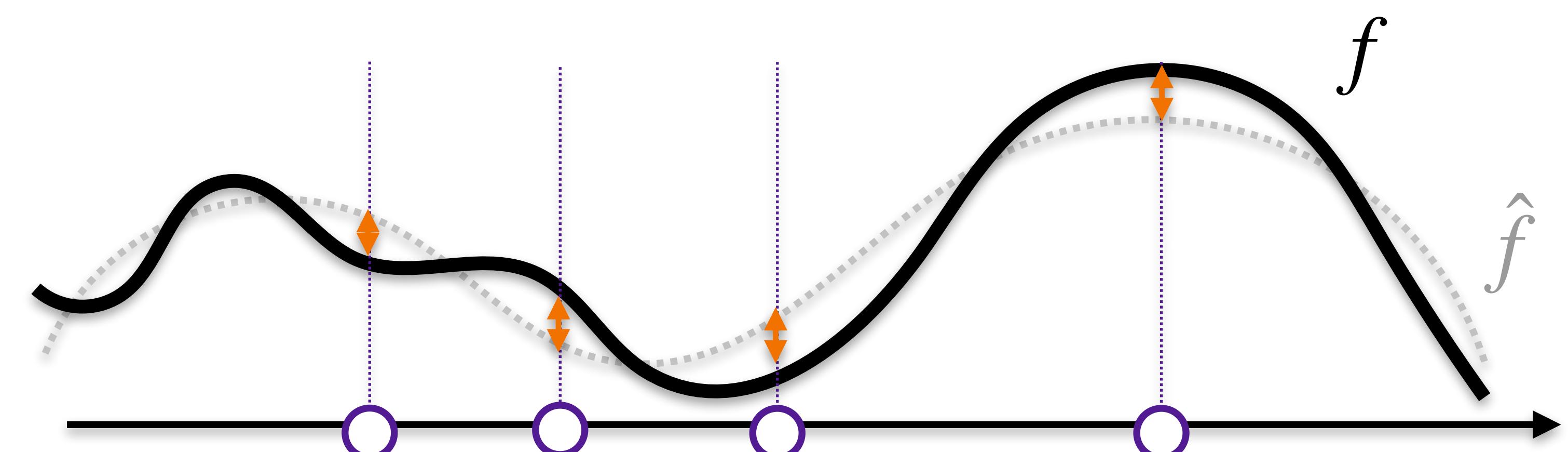
I What is SCaSML about?



$$\{X_1, \dots, X_n\} \sim \mathbb{P}_\theta \rightarrow \theta \rightarrow \Phi(\theta)$$

Step 1: Using Machine Learning to fit the rough function/environment

Step 2: Using validation dataset to know how much mistake machine learning algorithm has made



Step 3: Using Simulation algorithm to estimate $\Phi(\theta) - \Phi(\hat{\theta})$



Examples Later!

Using ML surrogate during inference time to improve ML solution