What is Sketch-and-Precondition for Eigenvector?

A Iterative Sketching View Point

Yiping Lu

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Ruihan Xu (Uchicago)

WARNING

The first part of the talk <u>**seems**</u>
Irrelevant to second half and title.

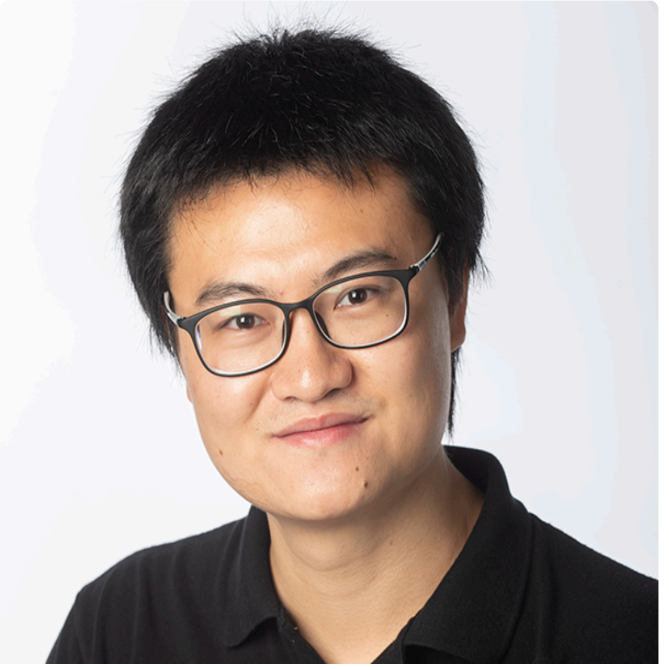
What I'm working on now?

Iterative Refine a ML solution

Numerical linear algebra and beyond NLA…

Lexing Ying (Stanford)

Jose Blanchet (Stanford)



Shihao Yang (Gatech)

Consider How you use ChatGPT…

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

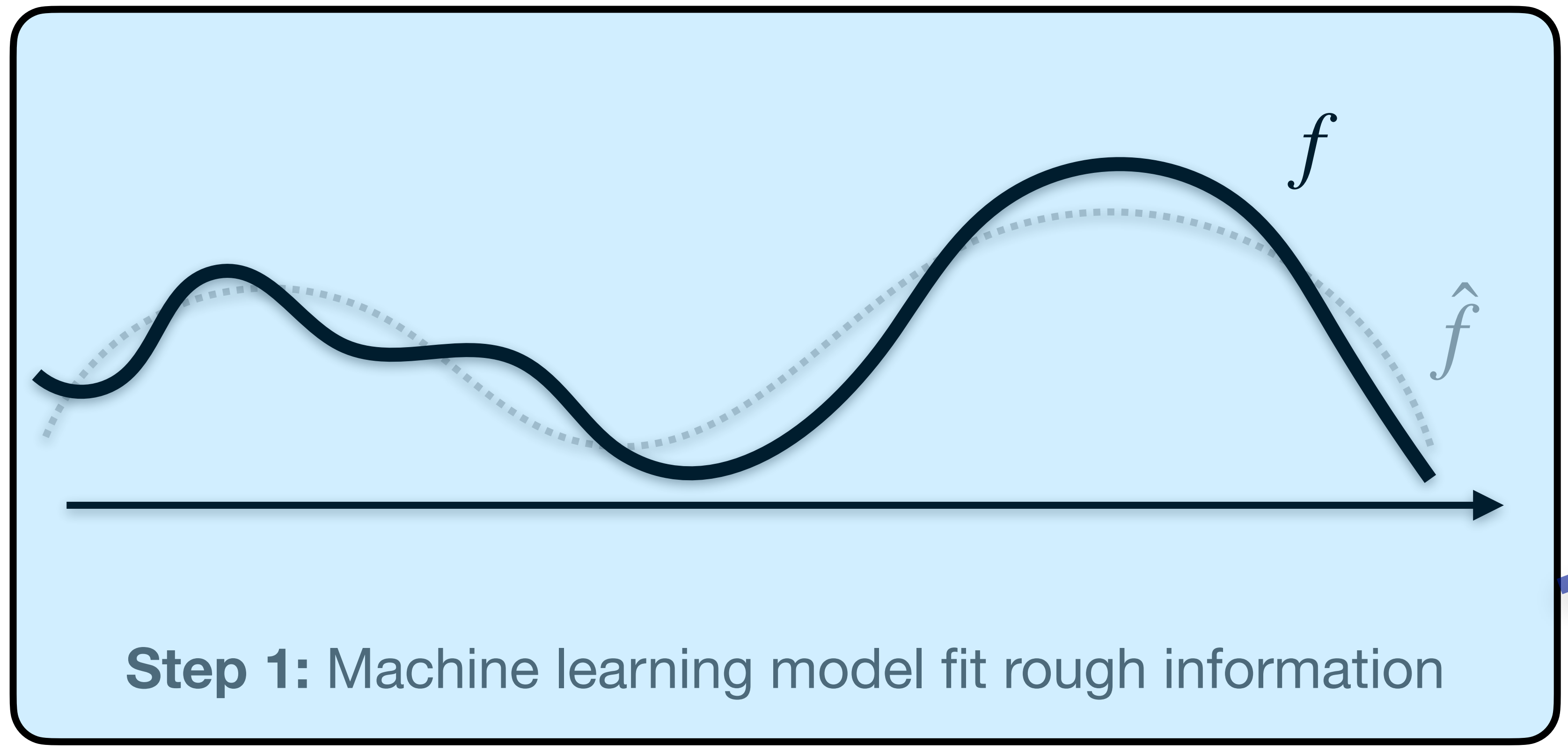
Option 1: Subway

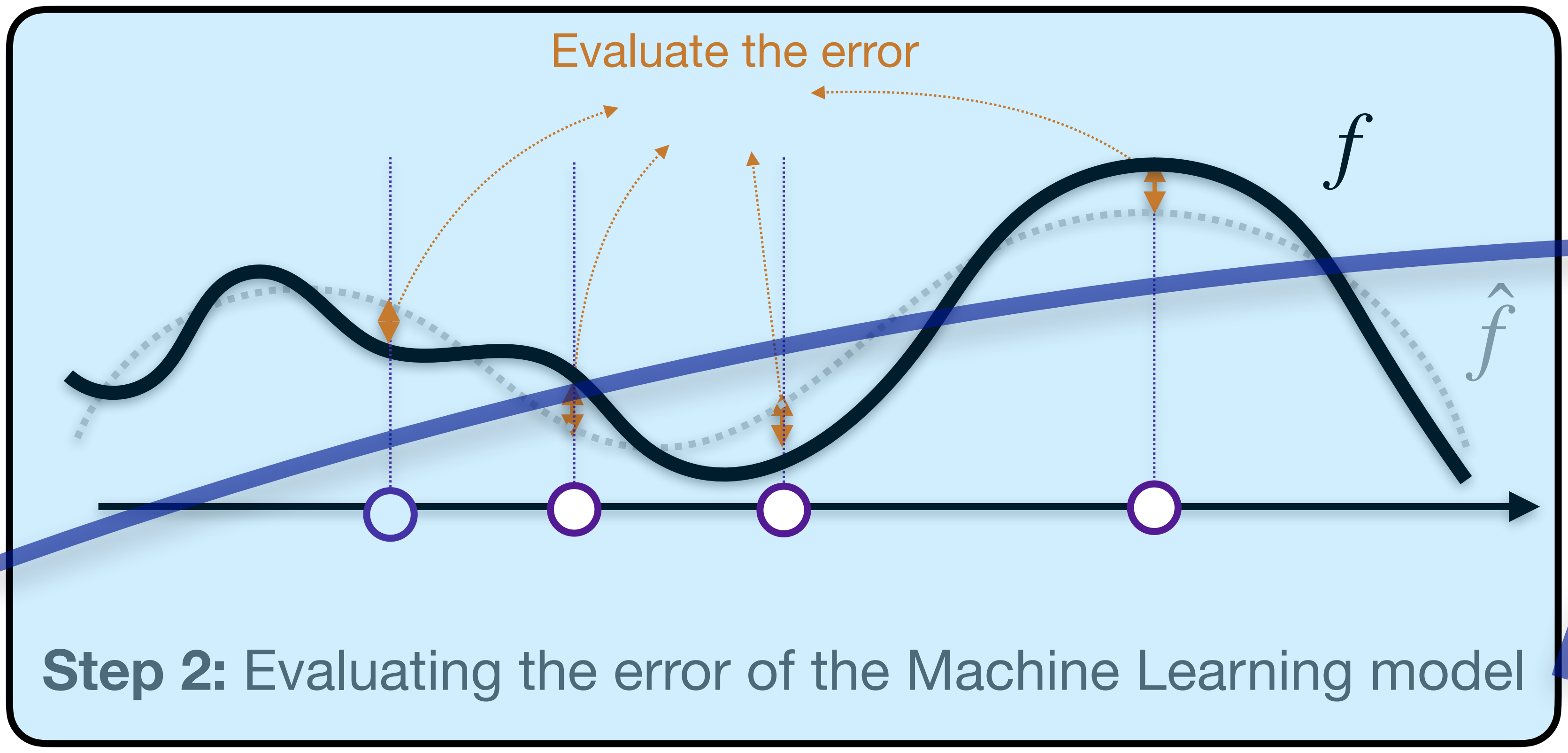
1. **Walk to a Nearby Subway Entrance:** Head to the Times Sq–42nd Street station.
2. **Board the E Train:** Catch an E train heading downtown (toward World Trade Center).
3. **Ride to 8th Avenue–14th Street Station:** Get off at this station (roughly 4 stops).
4. **Walk to Chelsea Market:** Exit the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

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).

Physics-Informed Debiasing the ML Solution





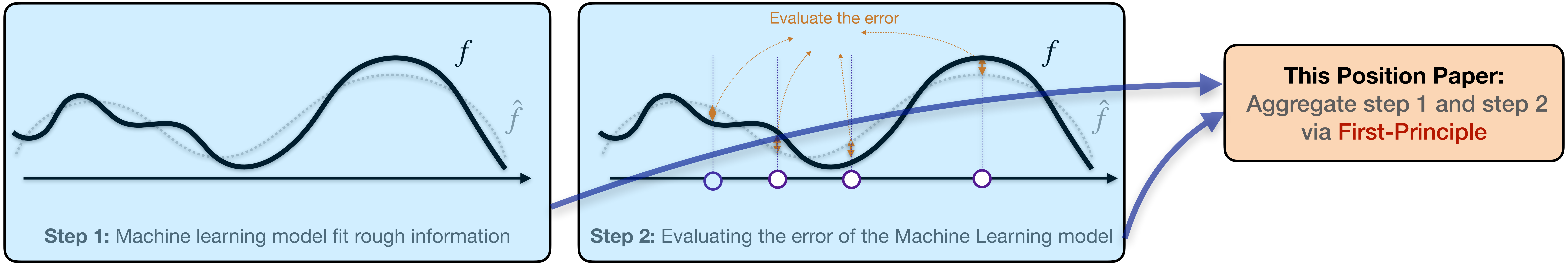
This Position Paper:
Aggregate step 1 and step 2
via First-Principle

In Numerical Linear Algebra:

Numerical Solving $Ax = b$ and get $\hat{x}$

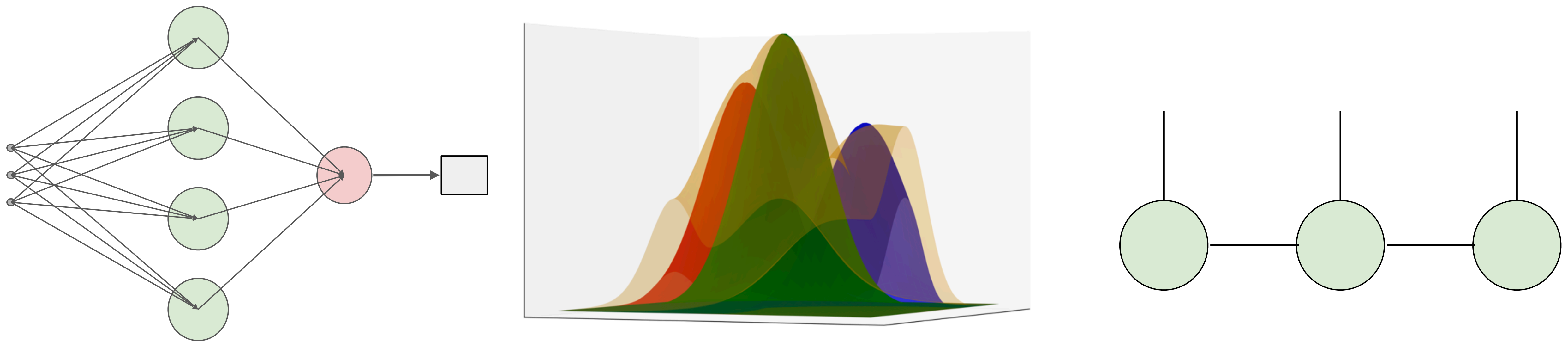
Estimate $x - \hat{x}$ via
Solving $A(x - \hat{x}) = b - A\hat{x}$ and get $\hat{x}$

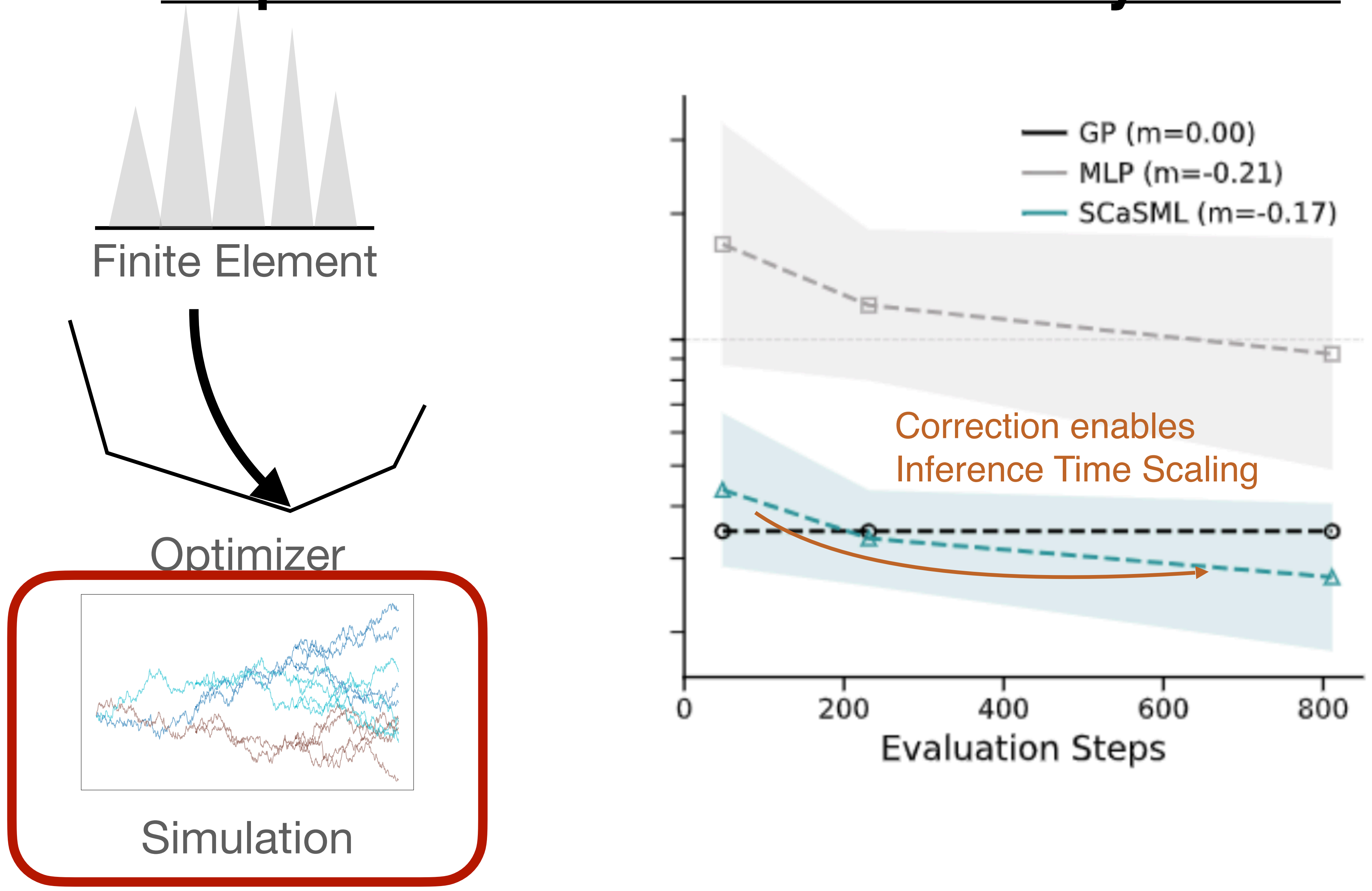
Physics-Informed Debiasing the ML Solution



Step 2. Correct with a Trustworthy Solver

Step 1. Train a Surrogate (ML) Model





Our Framework

Step 1: Sceintific Computing as Machine Learning

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Example 1

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

Function fitting

Example 2

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

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

Step 2: Consider a Downstream Application

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning | Downstream application

Example 1 $\theta = f, \quad X_i = (x_i, f(x_i))$ $\Phi(\theta) = \int f(x)dx$

Example 2 $\theta = \Delta^{-1}f, \quad X_i = (x_i, f(x_i))$ $\Phi(\theta) = (\Delta^{-1}f)(x)$

Example 3 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\Phi(\theta) = \mathrm{tr}(A), \mathrm{eigs}(A)$

Our Framework

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

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

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

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

A Numerical Linear Algebra Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\theta = A, \underbrace{\quad X_i}_{} = (x_i, Ax_i)$$

(Randomized) Subspace methods

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

Randomized SVD: $\hat{\theta} = \hat{A}$

A Numerical Linear Algebra Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\underbrace{\theta = A, \quad X_i}_{} = (x_i, Ax_i)$$

(Randomized) Subspace methods

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

Application in graph theory, quantum ...

Randomized SVD: $\hat{\theta} = \hat{A}$ $\longrightarrow$ $\Phi(\hat{\theta}) = \mathrm{tr}(\hat{A})$

A Numerical Linear Algebra Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\underbrace{\theta = A, \quad X_i}_{\text{(Randomized) Subspace methods}} = (x_i, Ax_i)$$

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

$$\|$$

Randomized SVD: $\hat{\theta} = \hat{A}$ $\longrightarrow$ $\Phi(\hat{\theta}) = \text{tr}(\hat{A})$

$$+$$

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

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

Lin 17 Numerische Mathematik and Mewyer-Musco-Musco-Woodruff 20

More Examples…

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning — Downstream application

Example 1 $\theta = f, \quad X_i = (x_i, f(x_i))$ $\quad \Phi(\theta) = \int f^q(x)dx$

Example 2 $\theta = \Delta^{-1} f, \quad X_i = (x_i, f(x_i))$ $\quad \Phi(\theta) = \theta(x)$

Example 3 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\quad \Phi(\theta) = \mathrm{tr}(A)$

Estimation $\hat{A}$ via Randomized SVD — Estimate $\mathrm{tr}(A - \hat{A})$ via Hutchinson's estimator

Example 4

Siegel J W, Xu J. Sharp bounds on the approximation rates, metric entropy, and n-widths of shallow neural networks. Foundations of Computational Mathematics, 2024, 24(2): 481-537.

The 101 Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\theta = f, \underbrace{\quad X_i}\ = (x_i, f(x_i))$$

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

Machine Learning: $\hat{\theta} = \hat{f}$

The 101 Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\theta = f, \quad \underbrace{X_i = (x_i, f(x_i))}$$

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

Machine Learning: $\hat{\theta} = \hat{f}$ $\longrightarrow$ $\Phi(\hat{\theta}) = \int \hat{f}(x)dx$

The 101 Example

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

Example

$$\theta = f, \quad \underbrace{X_i = (x_i, f(x_i))}$$

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

$$=$$

Machine Learning: $\hat{\theta} = \hat{f}$ $\longrightarrow$

$$\Phi(\hat{\theta}) = \int \hat{f}(x)dx$$

$$+$$

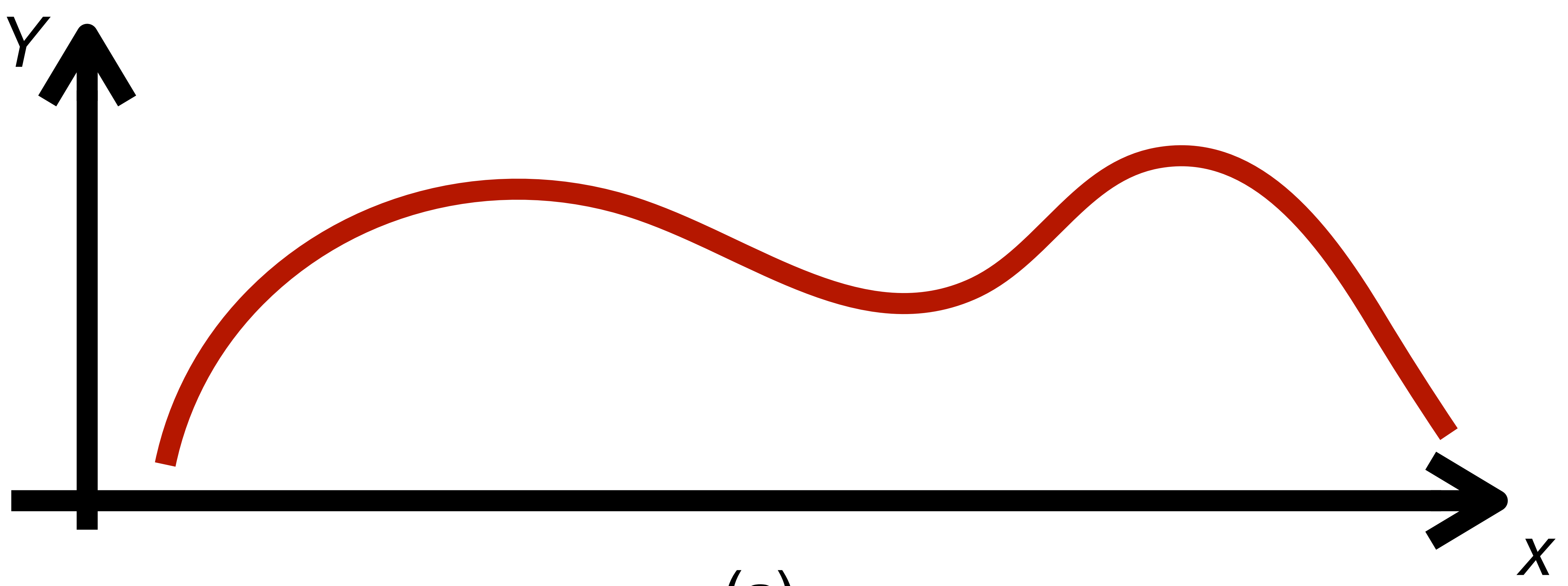
$$\Phi(\theta) - \Phi(\hat{\theta}) = \underbrace{\int (f(x) - \hat{f}(x))dx}$$

Using Monte Carlo Methods to approximate

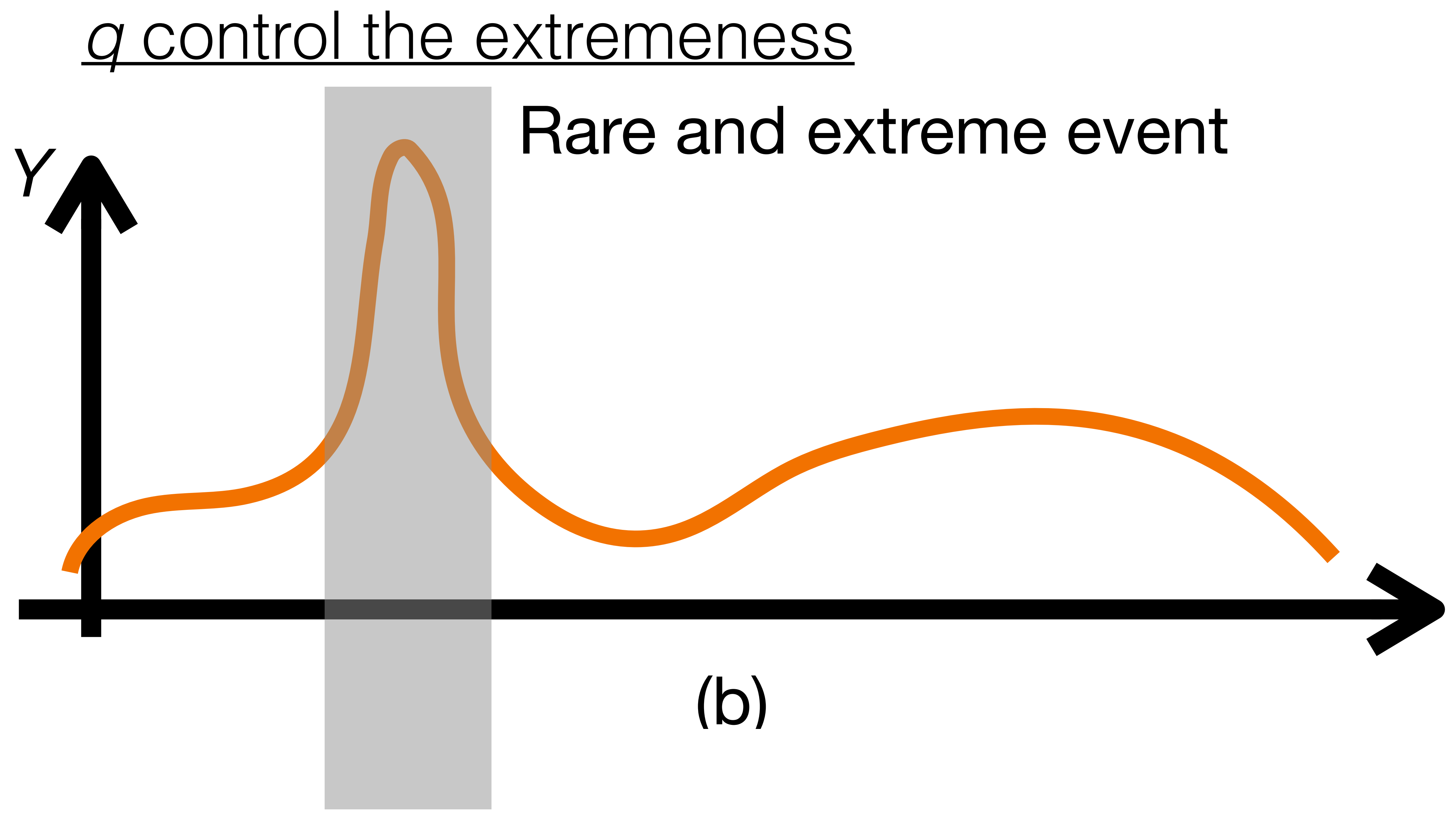
Optimal Algorithm!

- Jose Blanchet, Haoxuan Chen, Yiping Lu, Lexing Ying. When can Regression-Adjusted Control Variates Help? Rare Events, Sobolev Embedding and Minimax Optimality Neurips 2023

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



(a)



(b)

PDE Solver

The PDE Example

Let's consider $\Delta u = f$

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \Phi(\hat{\theta})$$

Scientific Machine Learning

Downstream application

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

$$\Phi(\theta) = u(x), \text{ or } \int (u(x))dx$$

FEM/PINN/DGM/Tensor/Sparse Grid/…:

$$\hat{\theta} = \hat{u}$$

The PDE Example

Let's consider $\Delta u = f$

$$\boxed{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}} \to \boxed{\Phi}(\hat{\theta})$$

Scientific Machine Learning

Downstream application

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

$\Phi(\theta) = u(x)$, or $\int (u(x))dx$

What is $\Phi(\theta) - \Phi(\hat{\theta}) = u(x) - \hat{u}(x)$?

FEM/PINN/DGM/Tensor/Sparse Grid/…:

$$\hat{\theta} = \hat{u} \longrightarrow \Phi(\hat{\theta}) = \hat{u}(x)$$

The PDE Example

Let's consider $\Delta u = f$

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

Scientific Machine Learning

Downstream application

$\boxed{\Delta u = f}$

$$\theta = u, \quad \underbrace{X_i = (x_i, f(x_i))}$$

$\Phi(\theta) = u(x)$, or $\int (u(x))dx$

What is $\Phi(\theta) - \Phi(\hat{\theta}) = u(x) - \hat{u}(x)$?

$-$

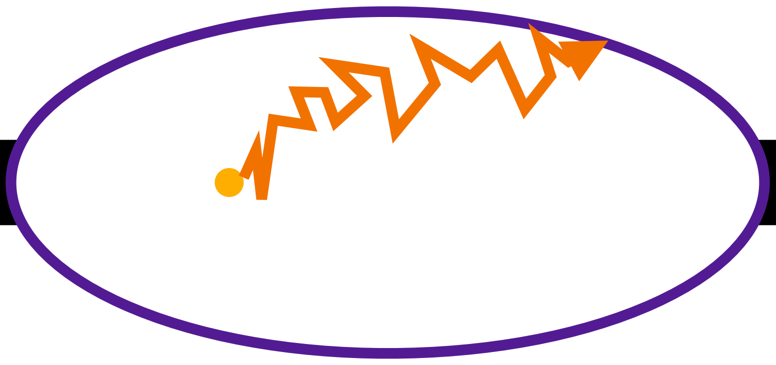
$\boxed{\Delta \hat{u} = \hat{f}}$

$=$

FEM/PINN/DGM/Tensor/Sparse Grid/…:

$$\hat{\theta} = \hat{u} \longrightarrow \Phi(\hat{\theta}) = \hat{u}(x)$$

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



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

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

NN

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

$g(x,t)$ 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

NN

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

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

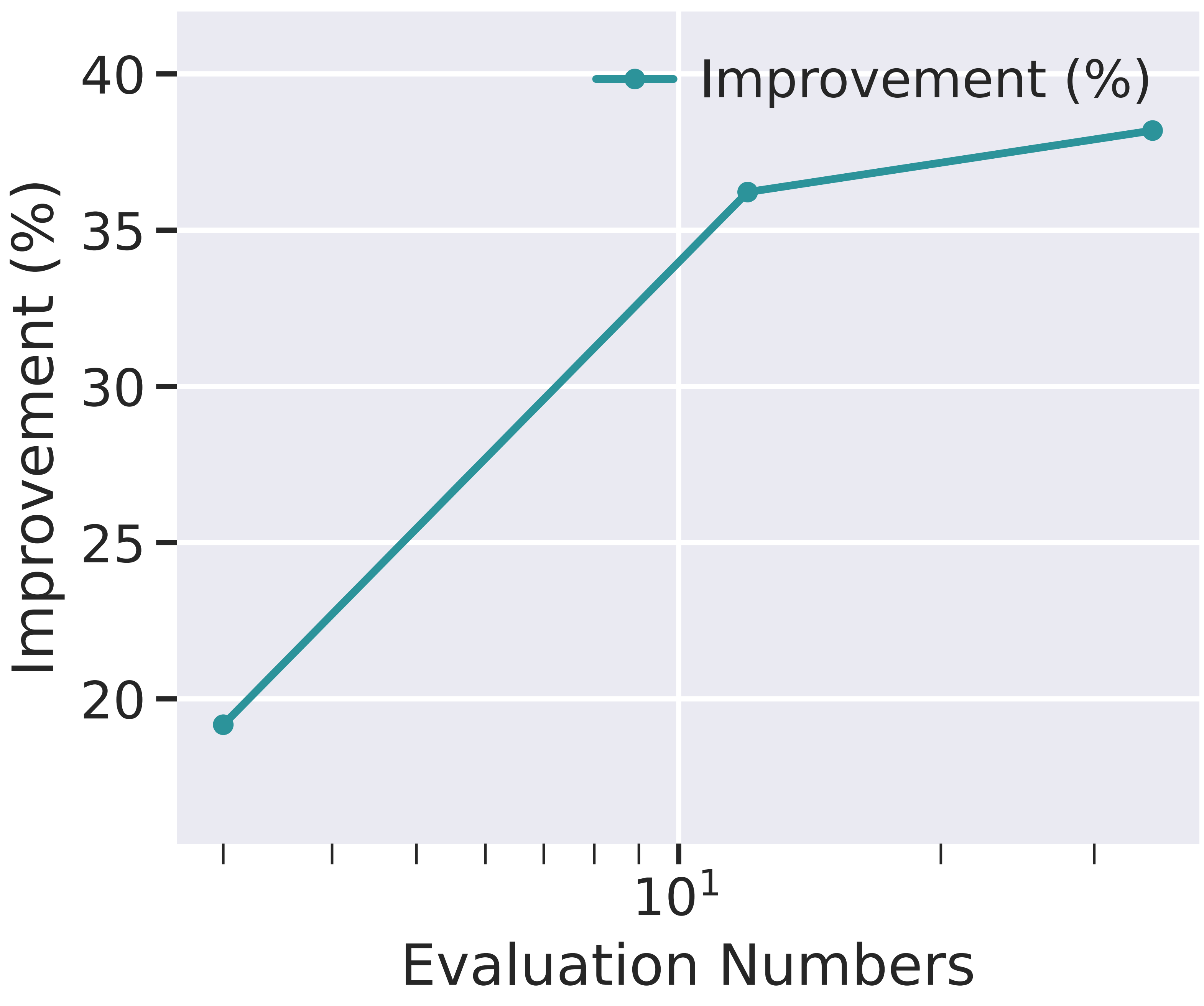
Subtract two equations

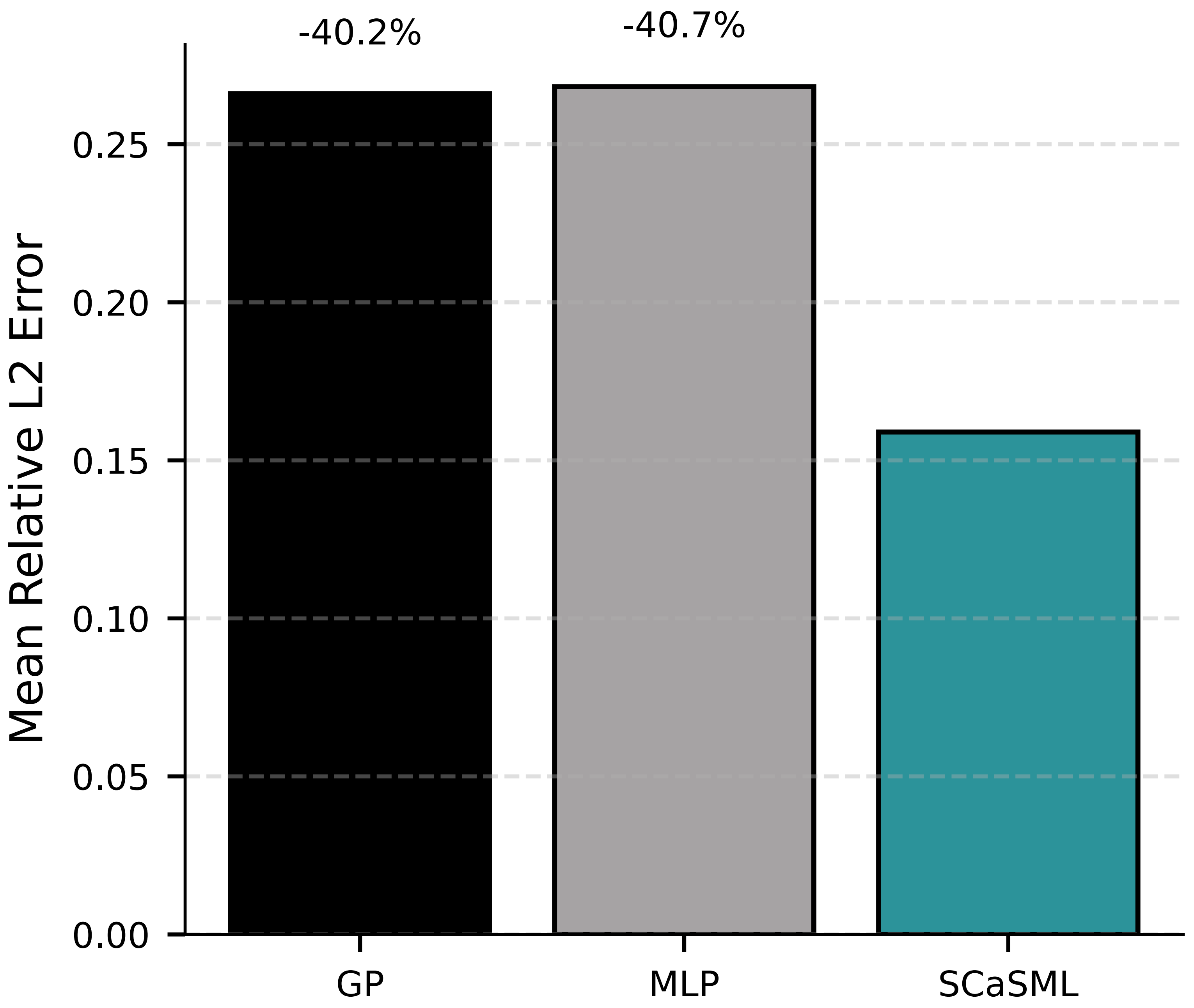
Keeps the linear structure

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

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 $g(x) = \frac{\exp(T + \sum_i x_i}{1 + \exp(T + \sum_i x_i)}$

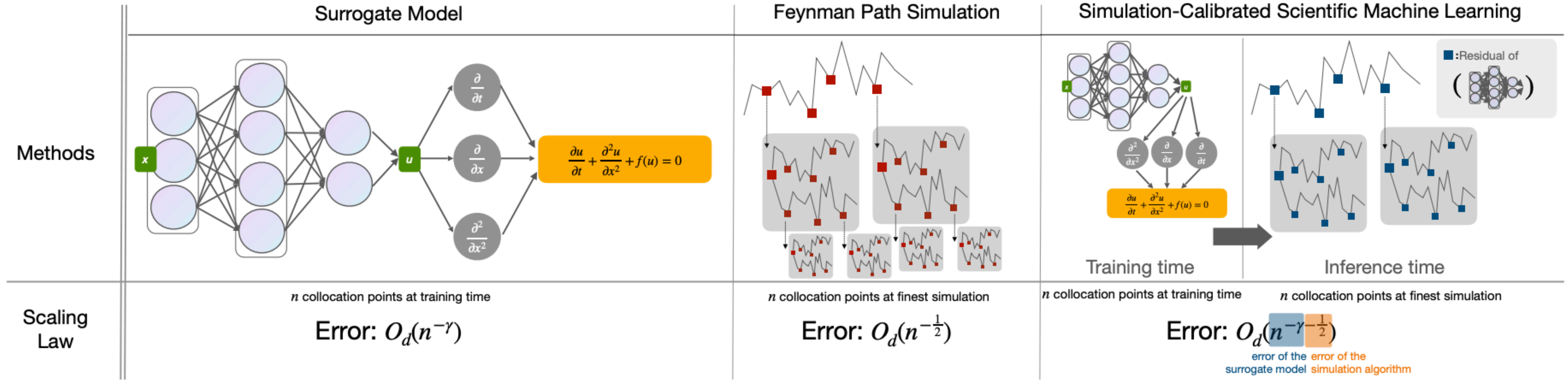




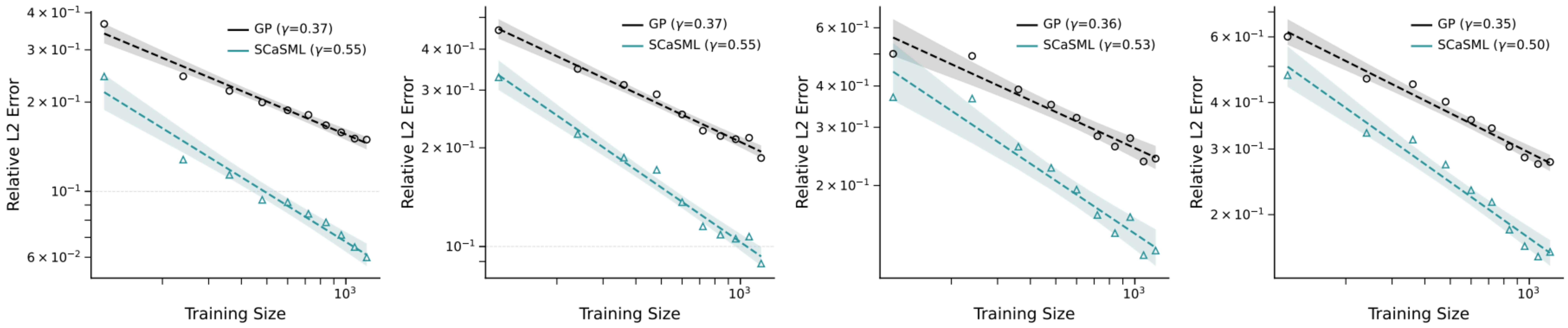
Method	Convergence Rate
PINN	$O(n^{-s/d})$
MLP	$O(n^{-1/2})$
ScaSML	$O(n^{-1/2-s/d})$

Better Scaling Law

a)



b)



(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

SCaSML
Simulation-Calibrated
Scientific Machine Learning

Arxiv

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

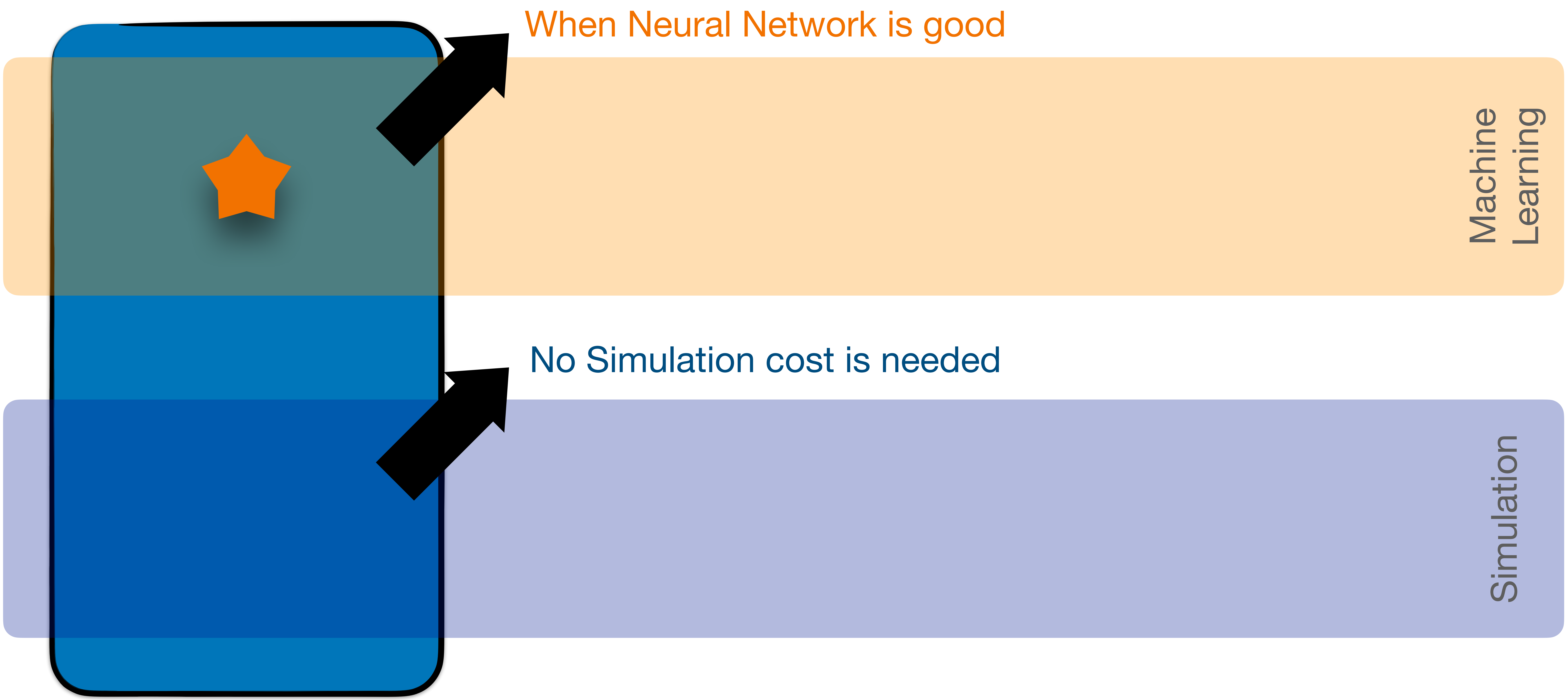
Zexi Fan[1], Yan Sun [2], Shihao Yang[3], Yiping Lu*[4]

[1] **Peking University** [2] **Visa Inc.** [3] **Georgia Institute of Technology** [4] **Northwestern University**

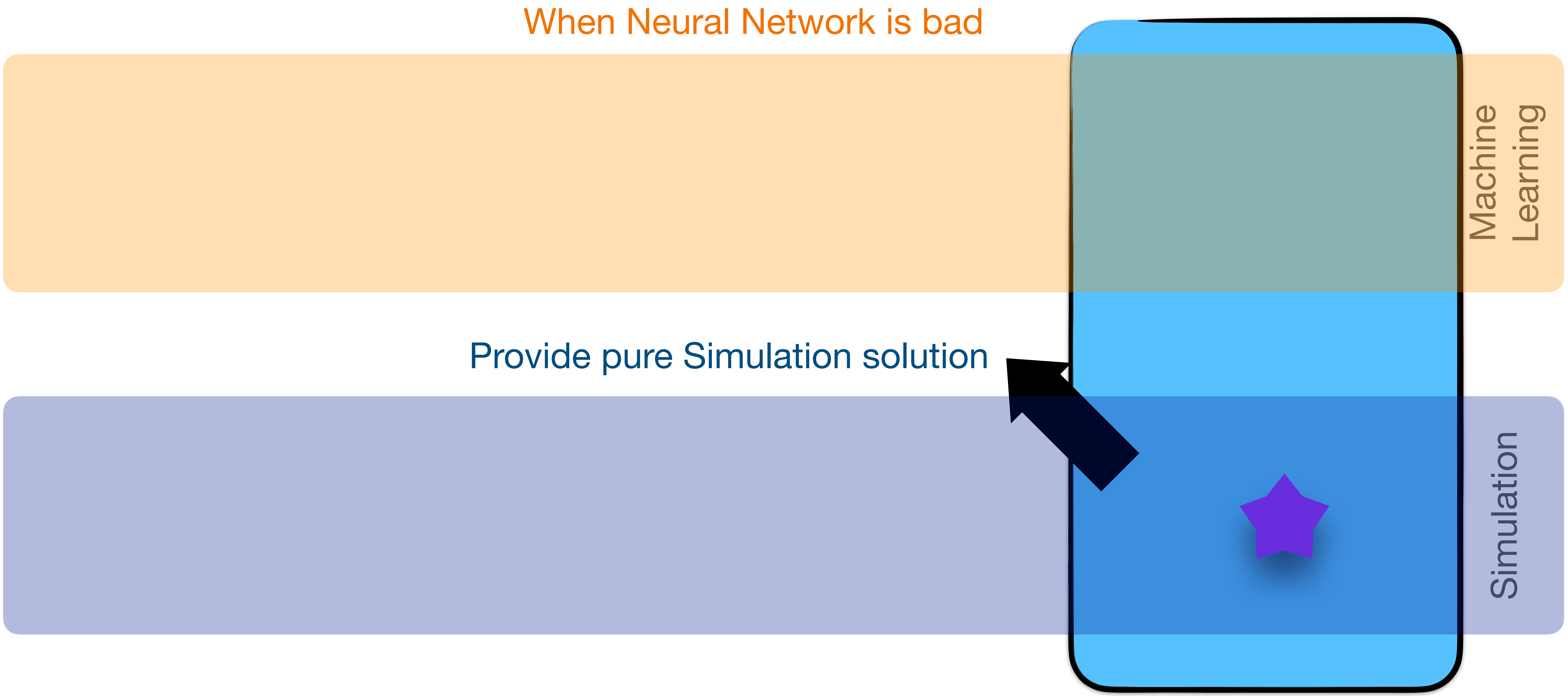
fanzexi_francis@stu.pku.edu.cn,yansun414@gmail.com,
shihao.yang@isye.gatech.edu,yiping.lu@northwestern.edu

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

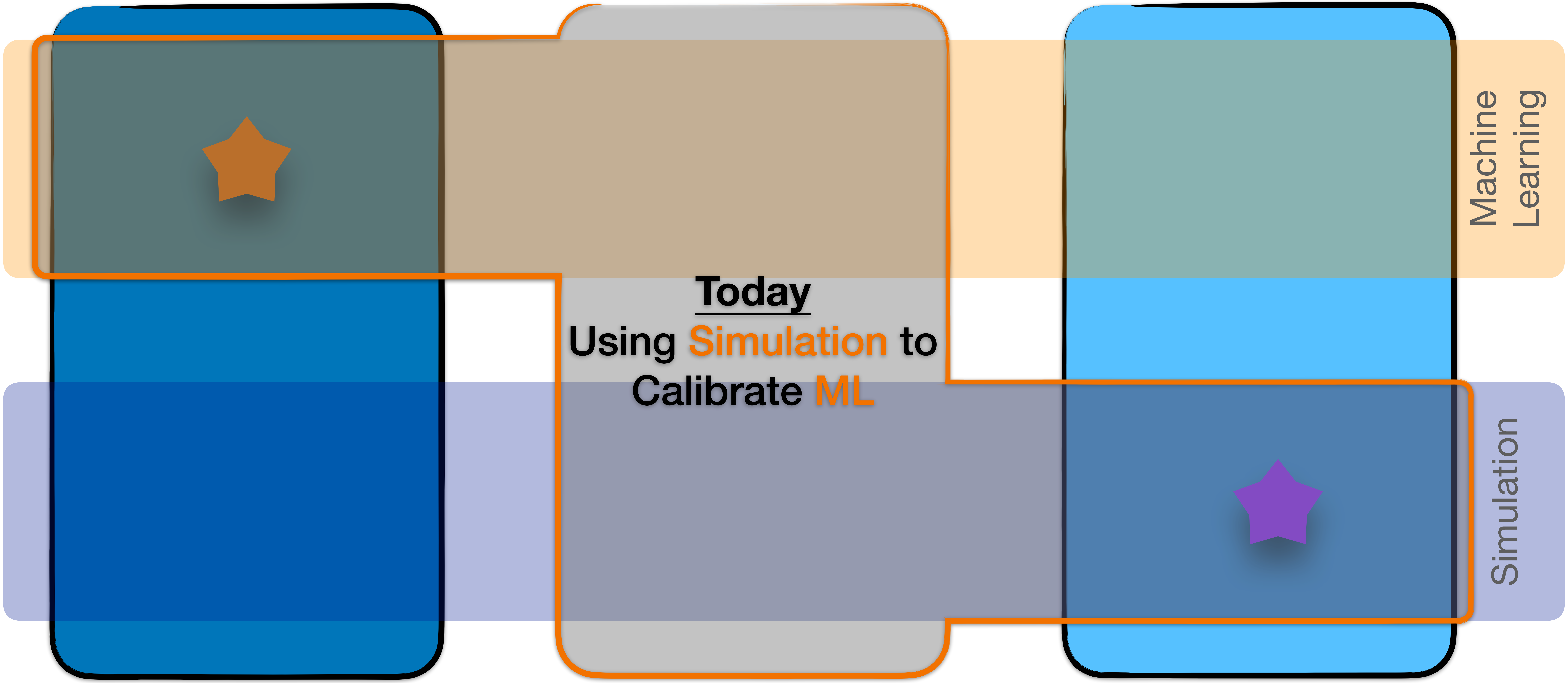
Our Aim Today : A Marriage



Our Aim Today : A Marriage



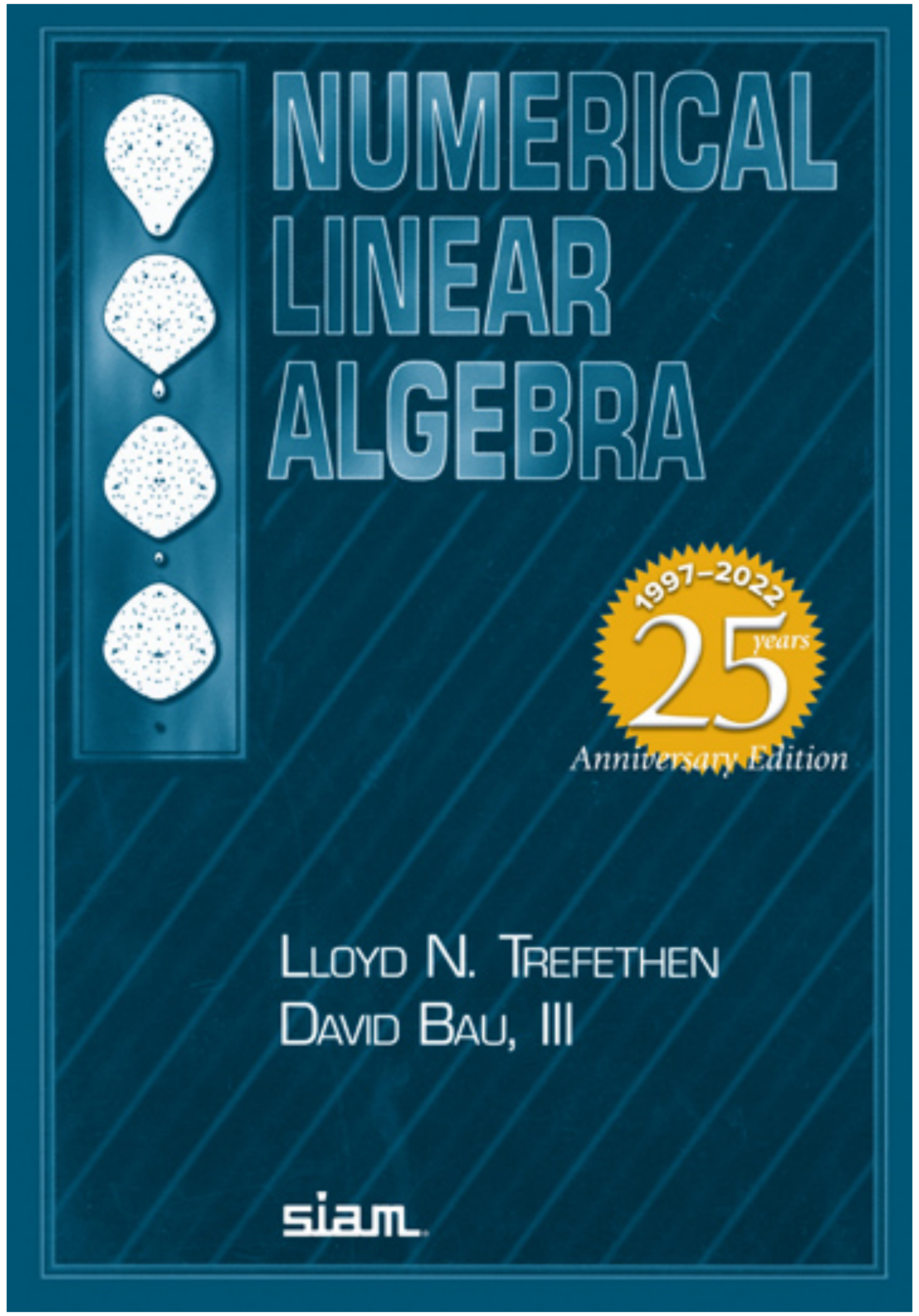
Our AIM Today: A Marriage



But your talk is on sketch-and-precondition?

Surprising(?) implementation of Pre-condition via 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 $\boxed{A}x = b$ is equivalent to solving $\boxed{B^{-1}A}x = B^{-1}b$

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 $\boxed{A(x - x_1) = b - Ax_1}$
  - Using the approximate solver to approximate $x - x_1$ via $Bx_2 = b - Ax_1$

Easy to solve for $b - Ax_1$ is small

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 $Bx_{i+1} = b - A\sum_{i=1}^{t} x_i$

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

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 $A(\hat{u}) \approx b$
  - Via machine learning/sketching/finite element….

Unrealiable approximate solver as preconditioner

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

Connection with control variate, doubly robust estimator, Multifidelity Monte Carlo

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

$$\underbrace{\boxed{\{X_1, \cdots, X_n\}}}_{X_i = (x, Ax)} \sim \underbrace{\mathbb{P}_\theta \to \theta}_{A} \to \Phi(\theta)$$

$$\Phi(A) = \begin{cases} A^{-1}b \\ \text{Eigenvalue of } A \end{cases}$$

"Randomized Numerical Linear Algebra"/Sketching

"Sketch-and-Solve"

> *It seems easier to train a bi-directional LSTM with attention than to compute the SVD of a large matrix. –Chris Re*
>
> NeurIPS 2017 Test-of-Time Award, Rahimi and Recht (Rahimi and Recht, 2017).

Randomized NLA as Machine Learning

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

$$\underbrace{\boxed{\{X_1, \cdots, X_n\}}}_{X_i = (x, Ax)} \sim \underbrace{\mathbb{P}_\theta}_{A} \to \theta \to \Phi(\theta)$$

Project to (Randomized) Subspace methods

$$\Phi(A) = \begin{cases} A^{-1}b \\ \boxed{\text{Eigenvalue of } A} \end{cases}$$

Esimtation of A: $\hat{A} = QQ^\top A$ $\longrightarrow$ $\Phi(\hat{\theta}) = \mathrm{svd}(\hat{A})$

“Randomized SVD”

Randomized NLA as Machine Learning

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

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

$X_i = (x, Ax)$ $\qquad$ A

Project to (Randomized) Subspace methods

$$\Phi(A) = \begin{cases} A^{-1}b \\ \text{Eigenvalue of } A \end{cases}$$

What is $\Phi(\theta) - \Phi(\hat{\theta})$?

<u>**Esimtation of A:**</u> $\hat{A} = QQ^\top A$ $\longrightarrow$ $\Phi(\hat{\theta}) = \text{svd}(\hat{A})$

"Randomized SVD"

Randomized NLA as Machine Learning

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

$$\underbrace{\boxed{\{X_1, \cdots, X_n\}} \sim \mathbb{P}_\theta \to \theta}_{X_i = (x, Ax) \qquad\qquad A} \to \Phi(\theta)$$

$$\Phi(A) = \begin{cases} A^{-1}b \\ \text{Eigenvalue of } A \end{cases}$$

Structure here: Φ is the solution of a fixed point equation

$$\Phi(\hat{\theta}) - \Phi(\theta) - \boxed{\nabla\Phi(\hat{\theta})(\hat{\theta} - \theta)} = O(\epsilon^2)$$

"A Newton Step"

Radomized estimation

Exact estimation

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

Relationship with Inverse Power Methods

(Approximate) Inverse Power Method	Our Method
$X_{n+1} = (\lambda I - A)^{\dagger} X_n$	$X_{n+1} = \underbrace{\boxed{(\lambda I - \hat{A})^{\dagger}}}_{\nabla \Phi(\hat{\theta})} \underbrace{\boxed{(A - \hat{A}) X_n}}_{(\theta - \hat{\theta})}$

Relationship with Inverse Power Methods

(Approximate) Inverse Power Method	Our Method
$X_{n+1} = (\lambda I - A)^{\dagger} \boxed{X_n}$	$X_{n+1} = (\lambda I - \hat{A})^{\dagger} \underbrace{\boxed{(A - \hat{A})}} X_n$

Replace with an approximate solver $\hat{A}$ changes the fixed point

Ture eigenvector is the fix point for every approximate solver $\hat{A}$

<u>Take Hoem Message 1</u>:
Power the Residual but not Power the vector

Relationship with Inverse Power Methods

(Approximate) Inverse Power Method	Our Method
$X_{n+1} = (\lambda I - A)^{\dagger} \boxed{X_n}$	$X_{n+1} = (\lambda I - \hat{A})^{\dagger} \underbrace{\boxed{(A - \hat{A})}} X_n$
Replace with an approximate solver $\hat{A}$ changes the fixed point	Ture eigenvector is the fix point for every approximate solver $\hat{A}$

How do you select the preconditioner $\hat{A}$?

Nyström approximation $\hat{A} = U \Lambda U^{\top}$
Using Woodbury to compute $(I - \hat{A})^{-1}$

Why better than Directly DMD

"Sketch-and-Solve" VS "Sketch-and-Precondition"

	Sketch-and-Solve	**Sketch-and-Precondition**
Least Square	😭	Sketch-and-precondition, Sketch-and-project, Iterataive Sketching, ….
Low rank Approx	Idea 1: plug in a SVD Solver: Random SVD Idea 2: plug in a inverse power method	 ***<u>Our Work!</u>***
	Use sketched matrix $\hat{A}$ as an approximation to A	Use sketched matrix $\hat{A}$ as an precondition to the probelm

Sorry… but I can't see the relationship….

Ruihan Xu, **Yiping Lu**. What is a Sketch-and-Precondition Derivation for Low-Rank Approximation? Power Error or Power Estimation?

Why better than Directly DMD

"Sketch-and-Solve" VS "Sketch-and-Precondition"

	Sketch-and-Solve	Sketch-and-Precondition
Least Square	😭	Sketch-and-precondition, Sketch-and-project, Iterataive Sketching, ….
Low rank Approx	Idea 1: plug in a SVD Solver: Random SVD Idea 2: plug in a inverse power method	***Our Work!***

Use sketched matrix $\hat{A}$ as an approximation to A

Use sketched matrix $\hat{A}$ as an precondition to the probelm

We only sketch the Hessian

Idea: using (approximate) Newton method to solve the Lagrange from

$$\min_u u^\top A u - \lambda(x^\top x - 1)$$

Thus Our convergence is linear-quadratic

Contraction coefficient improves when sketching quality increases

Ruihan Xu, **Yiping Lu**. What is a Sketch-and-Precondition Derivation for Low-Rank Approximation? Power Error or Power Estimation?

Theortical Garuntee

Computing top-1 eigenvector

Theorem 2.2: Convergence Rate of EPSI

For estimate $\hat{A}$ of semi positive definite matrix A with $\boxed{A - 3\eta I \preceq \hat{A} \preceq A - \eta I}$, and furthermore shares a same column space with A. Let $A = V \Lambda V^\top = \lambda_* v_* v_*^\top + V_2 \Lambda_2 V_2^\top$ be its eigen-decomposition, where $\lambda_* = \lambda_1$ is its max eigenvalue and $\Lambda_2 = diag(\lambda_2, \lambda_3, \cdots, \lambda_n)$ satisfying $\lambda_2 > \lambda_3 > \cdots > \lambda_n$. Then EPSI yields a (normalized) series $\{u_k\}$ which converges to u_* in a linear-quadratic behavior. Suppose that u_k satisfies $\frac{\|V_2^\top u_k\|}{\|u_k\|} \le \epsilon$ with $\frac{\lambda_1}{\eta}\epsilon < 1$, then the convergence of u_{k+1} is guaranteed by

Better embedding convergence faster

$$\|V_2^\top u_{k+1}\| \le \underbrace{\frac{3\lambda_1}{\lambda_1 - \lambda_2} \frac{\boxed{\eta}}{\lambda_1} \frac{1}{1 - \frac{\lambda_1}{\eta}\epsilon} \|V_2^\top u_k\|}_{\text{Linear convergence}} + \underbrace{\frac{2\lambda_1}{\eta} \|V_2^\top u_k\|^2}_{\text{Qudratic convergence}}$$

Computing top-k eigenvctors

Algorithm 2 Lazy-EPSI for Computing the First k Singular Vectors

Require: A, the input matrix; $k \in \mathbb{Z}^+$, the number of components; $q_{\max} \in \mathbb{Z}^+$, the maximum number of iterations.

for $q = 1$ to $q_{\max}$ **do**
- $U \leftarrow [\,]$ ▷ Initialize U as an empty matrix.
- **EPSI Iteration: Update first k eigenspace estimation U**
- **for** $i = 1$ to k **do**
  - $\hat{\lambda}_i \leftarrow \frac{(u_q^i)^\top A u_q^i}{(u_q^i)^\top u_q^i}$ ▷ Compute rayleigh quotient estimation $\hat{\lambda}_i$ for the ith eigenvector.
  - $\hat{u}_{q+1}^i \leftarrow ((I - UU^\top)\hat{A}(I - UU^\top) - \hat{\lambda}_i I)^{-1}((I - UU^\top)\hat{A}(I - UU^\top) - A)u_q^i$ ▷ Update $\hat{u}_{q+1}^i$.
  - $U \leftarrow orth([U, \hat{u}_{q+1}^i])$ ▷ Append gram-schmidt orthogonalized $\hat{u}_{q+1}^i$ to U.
- **end for**
- **Orthogonalization step: Use Estimated Eigenvector as Rangefinder**
- $\Pi \leftarrow UU^\dagger$ ▷ Compute the projection matrix Π.
- $A_U \leftarrow \Pi A \Pi$ ▷ Compute the projected matrix A_U.
- $[u_{q+1}^1, u_{q+1}^2, \ldots, u_{q+1}^k] \leftarrow \mathrm{SVD}(A_U)$ ▷ Perform SVD to update u_{q+1}^i.

end for

return U ▷ Return the updated matrix U.

Using EPSI to update each eigenvectors

Using new basis as embedding as RandSVD

Theortical Garuntee

Computing top-k eigenvector

Projection makes the error propagation quadratic

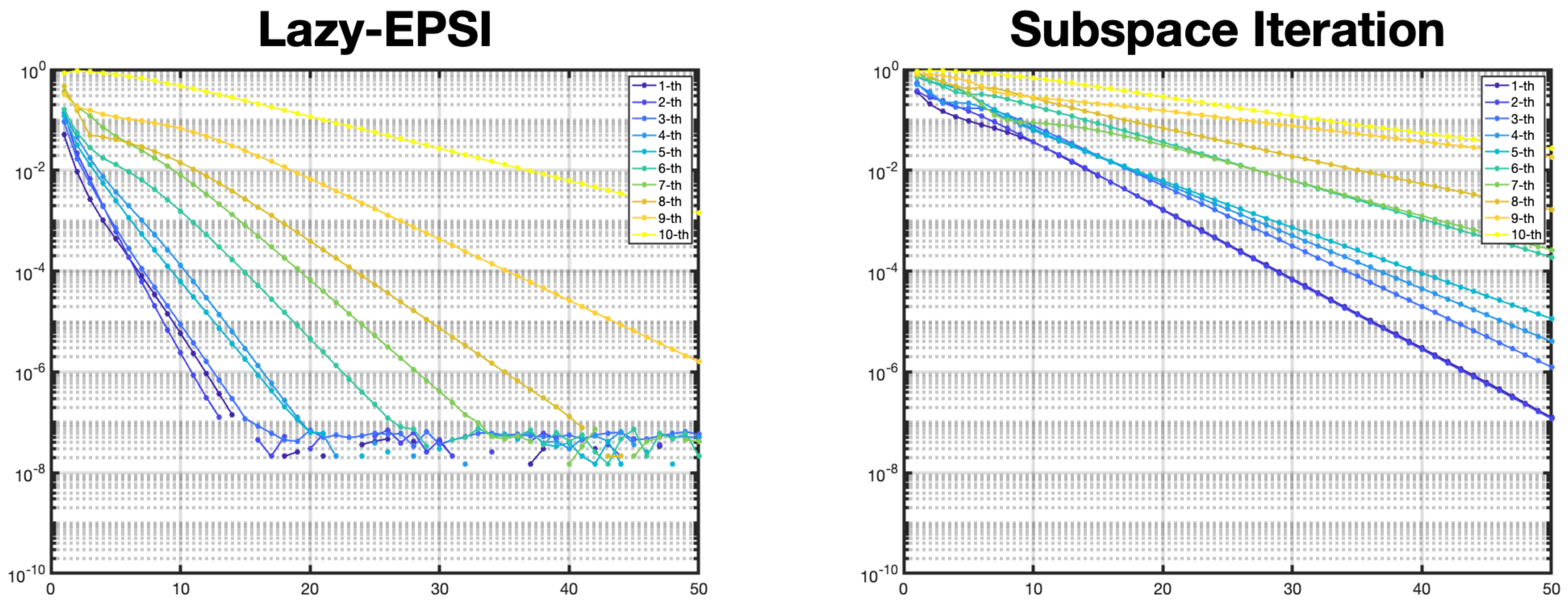
Lemma 2.3: Local Convergence Rate for Lazy-EPSI

Suppose that PSD matrix $A \in \mathbb{R}^{n \times n}$ has exact eigendecomposition $A = V\Lambda V^\top = V_1\Lambda_1 V_1^\top + V_2\Lambda_2 V_2^\top$, where V_1 has size $n \times k$ and the diagonal of Λ_1, Λ_2 is in descending order. The approximation $\hat{A}$ of A satisfies $A - 3\eta I \preceq \hat{A} \preceq A - \eta I$. Suppose that U, which is the $i-1$ eigenspace estimation, satisfies $\|V_{i:n}^\top U\| \leq \epsilon$ for small constant $\epsilon < \frac{\eta}{26(\lambda_1 - \eta)}$. Then $\hat{u}_{q+1}^i = ((I - UU^\top)\hat{A}(I - UU^\top) - \lambda_i I)^{-1}((I - UU^\top)\hat{A}(I - UU^\top) - A)u_q^i$ satisfies:

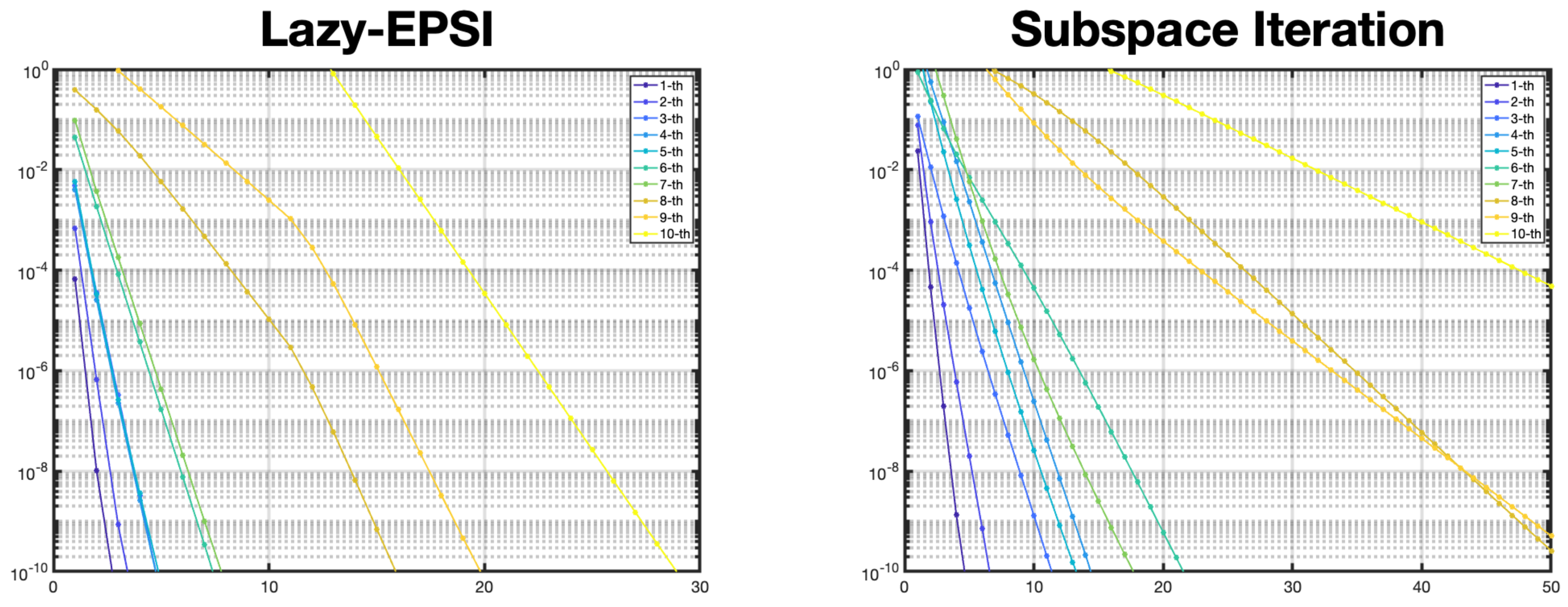
$$\frac{\|V_2^\top u_{q+1}^i\|}{\|u_{q+1}^i\|} \leq \frac{1}{1-\epsilon_0}\Big(\underbrace{\frac{3.5\lambda_k}{\lambda_i - \lambda_{k+1}}\frac{\eta}{\lambda_k}\frac{\|V_2^\top u_q^i\|}{\|u_q^i\|}}_{\text{linear convergence}} + \boxed{\underbrace{\frac{c_1(\lambda_i - \lambda_n)}{(\lambda_i - \lambda_{k+1})^2}\frac{\|(\lambda_i I - \Lambda_1)V_1^\top u_q^i\|}{\|u_q^i\|}}_{\text{error caused by imperfect projection}}}\Big)$$

where $\epsilon_0 = max\{\frac{4(\lambda_i - \lambda_{k+1})}{\eta}\|V_2^\top u_q^i\|, \frac{68\lambda_1}{\eta}\|V_2^\top U_q\|^2\}$ with $U_q = [u_q^1, u_q^2, \cdots, u_q^k]$, and c_1 is a small constant that depends on $\sqrt{k}$.

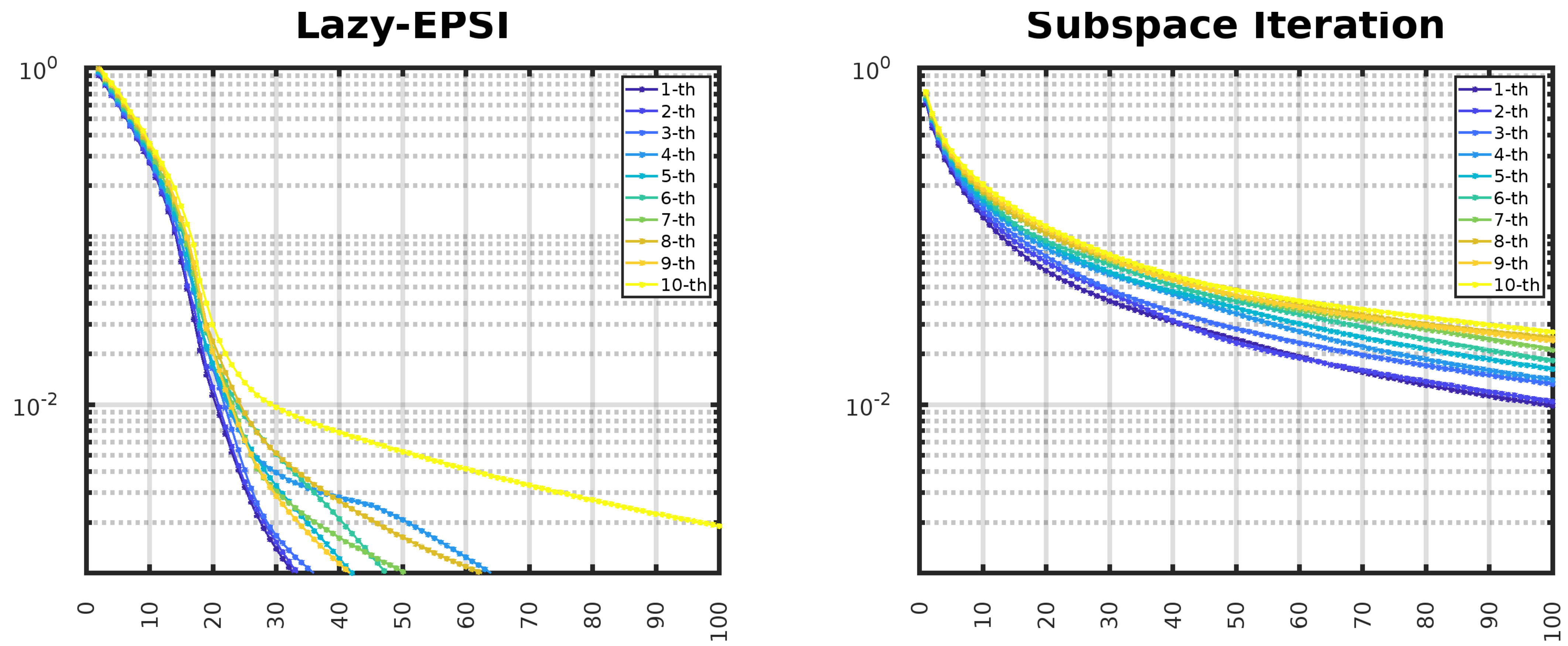
Eigenvalue Computation



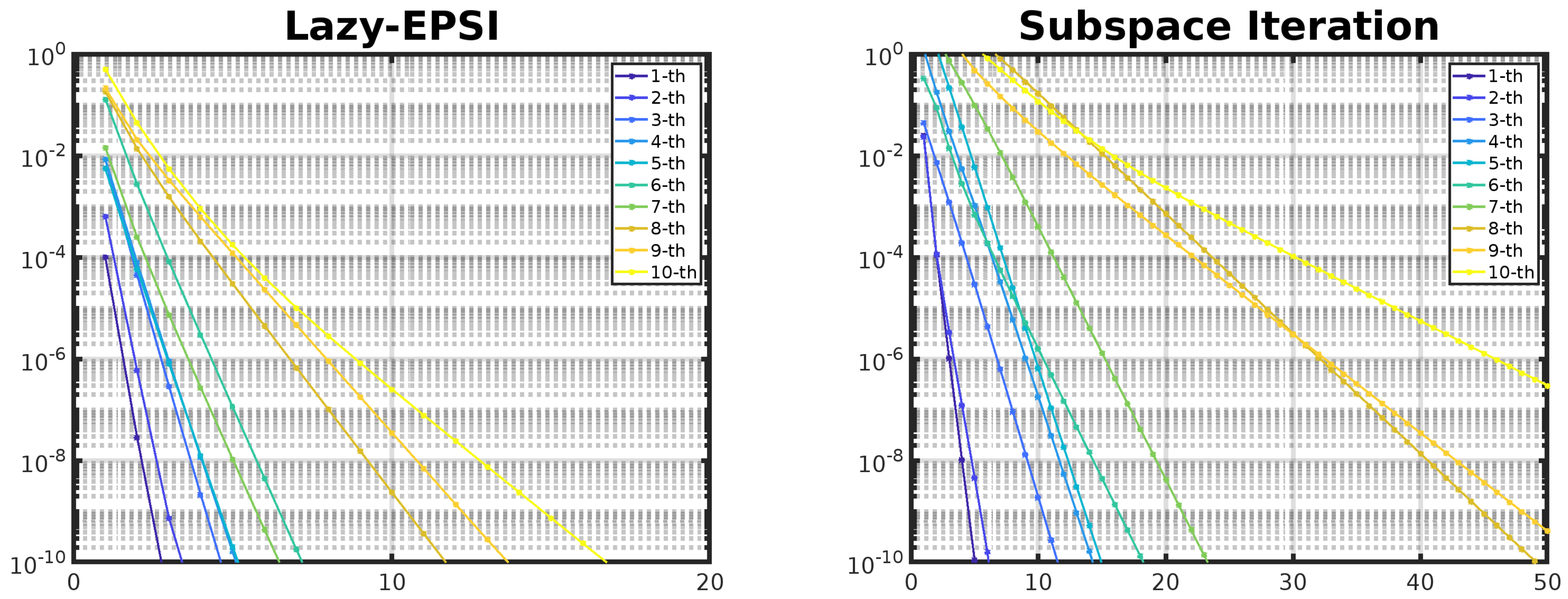
Random ill-conditioned matrix



Amazon (SNAP)

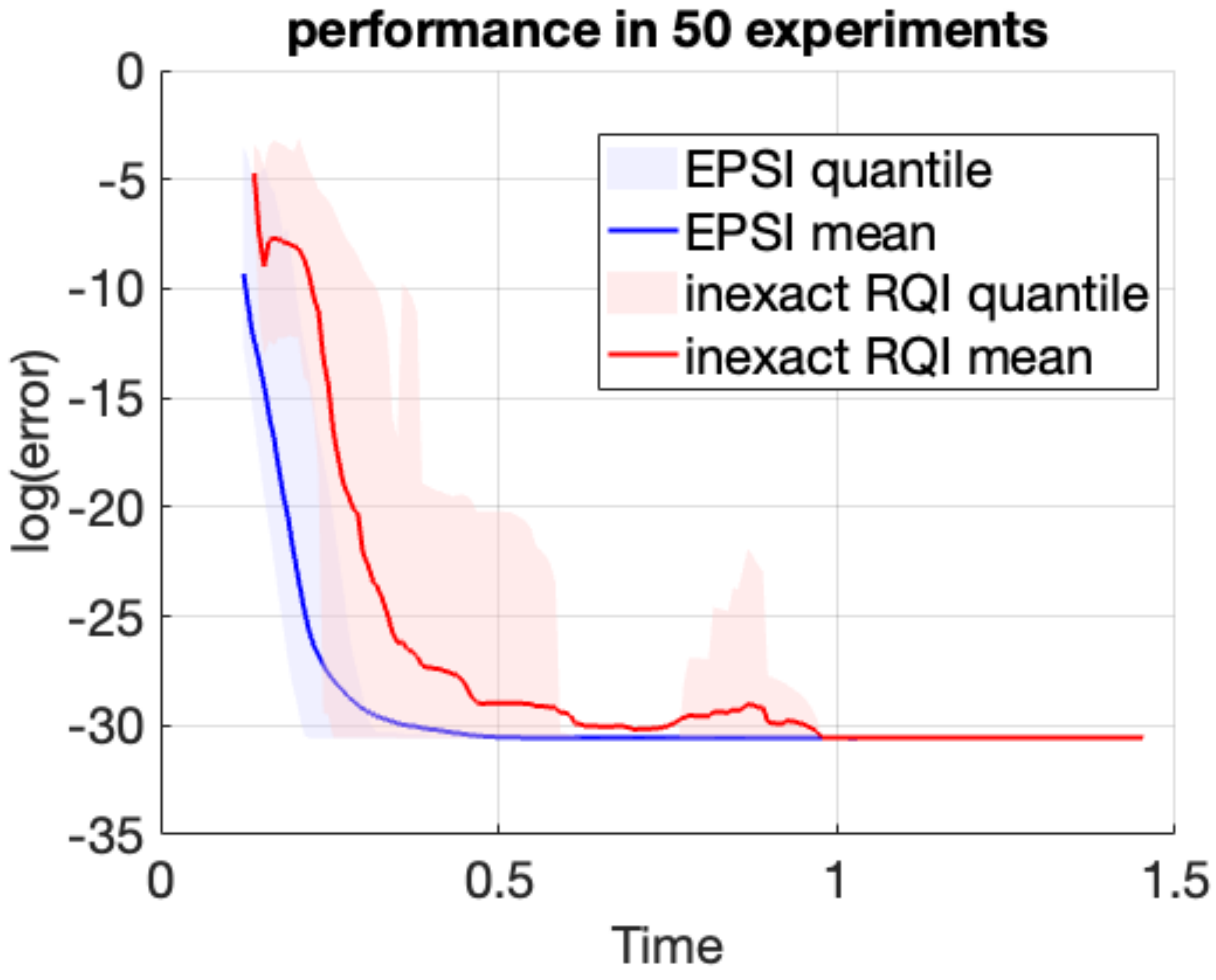


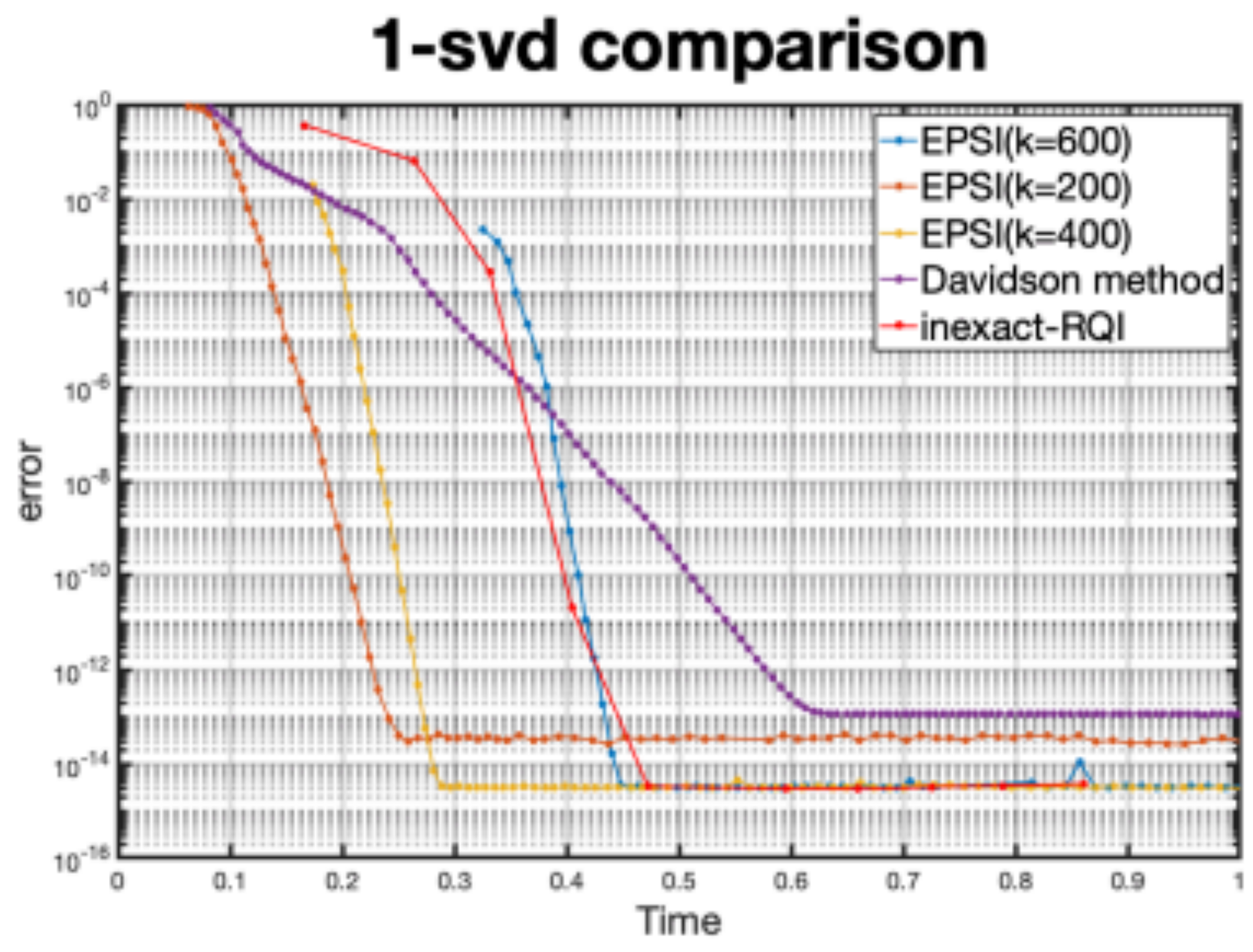
PDE (Laplacian)



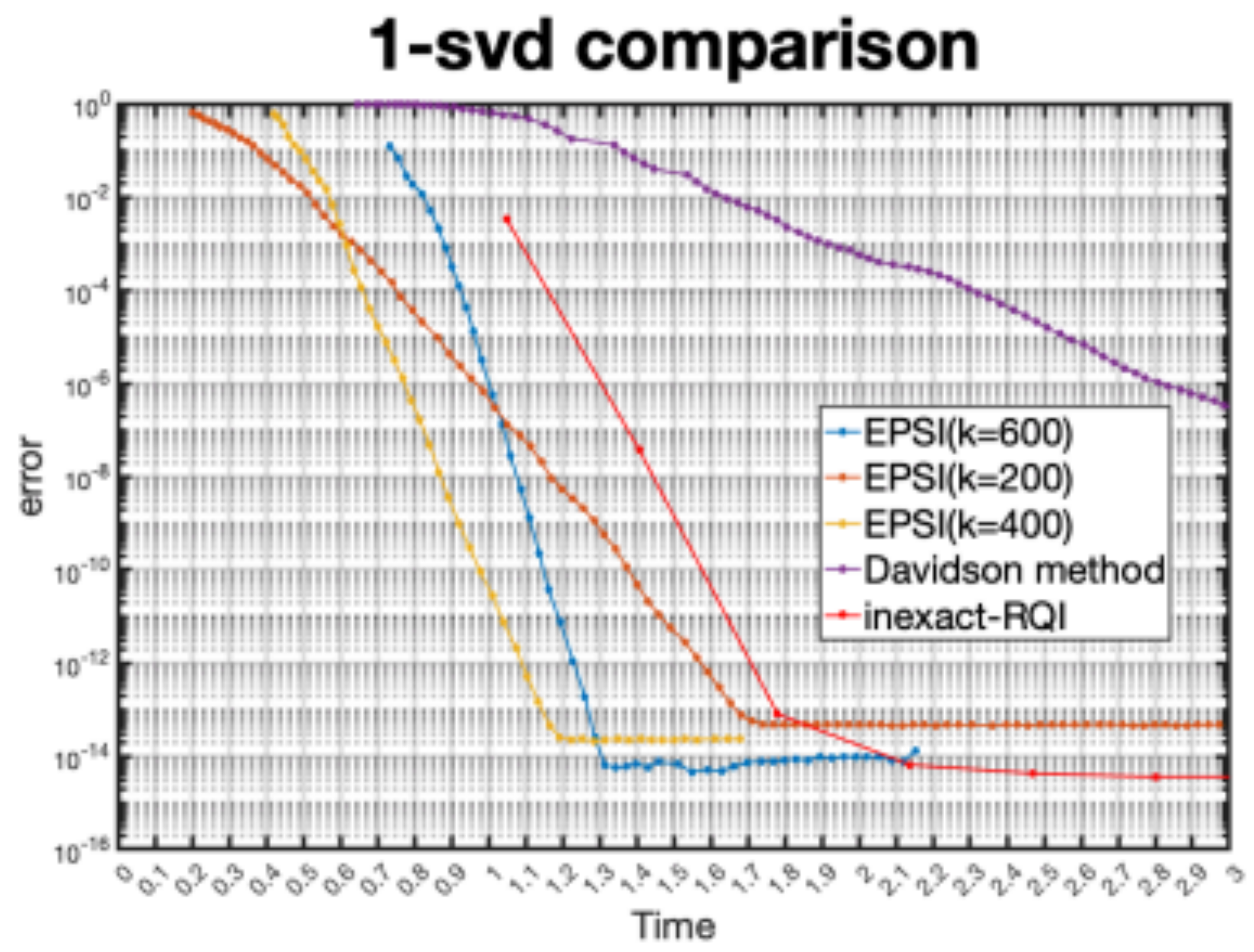
Web Stanford (SNAP)

Runing Time





(b) $n = 2000, \kappa = 10^{-6}$



(c) $n = 4000, \kappa = 10^{6}$

arXiv

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

Ruihan Xu *   Yiping Lu †

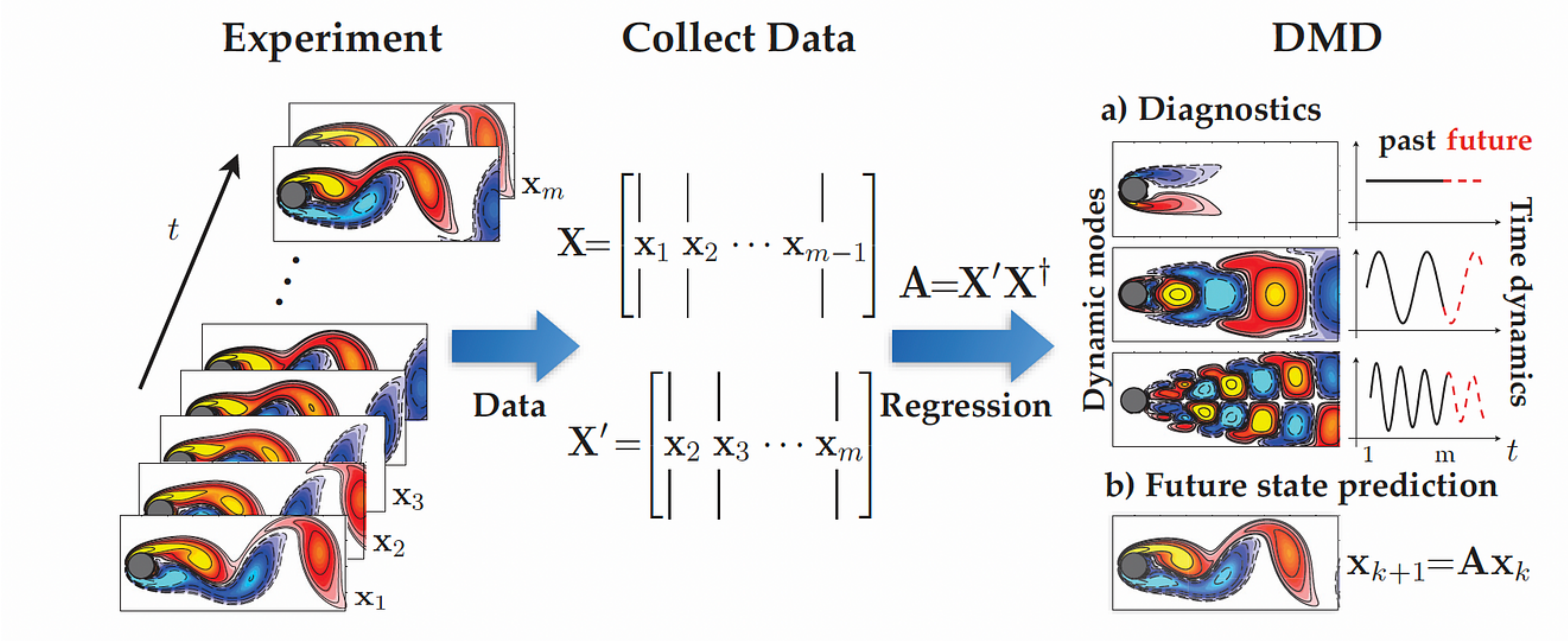
Abstract

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.

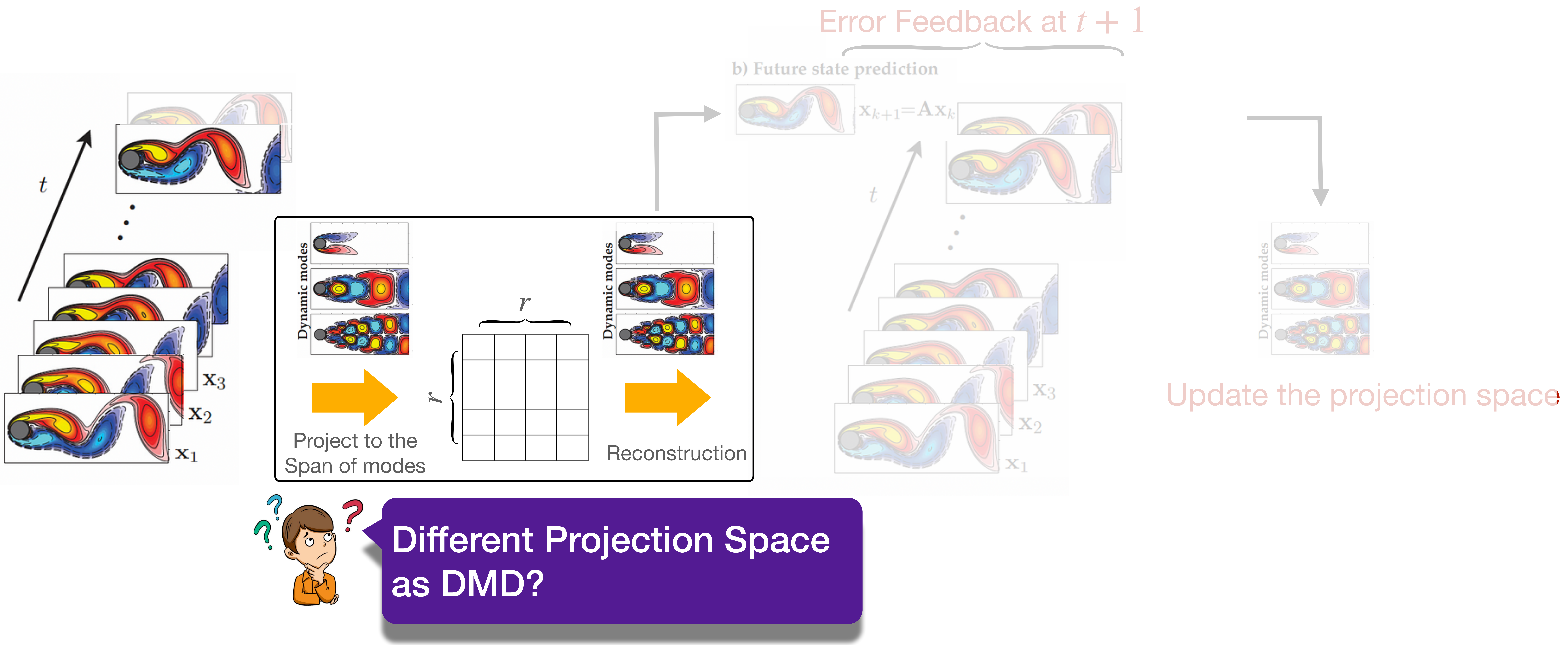
Another Supersing Fact…

Iteration lies in the Krylov Subspace

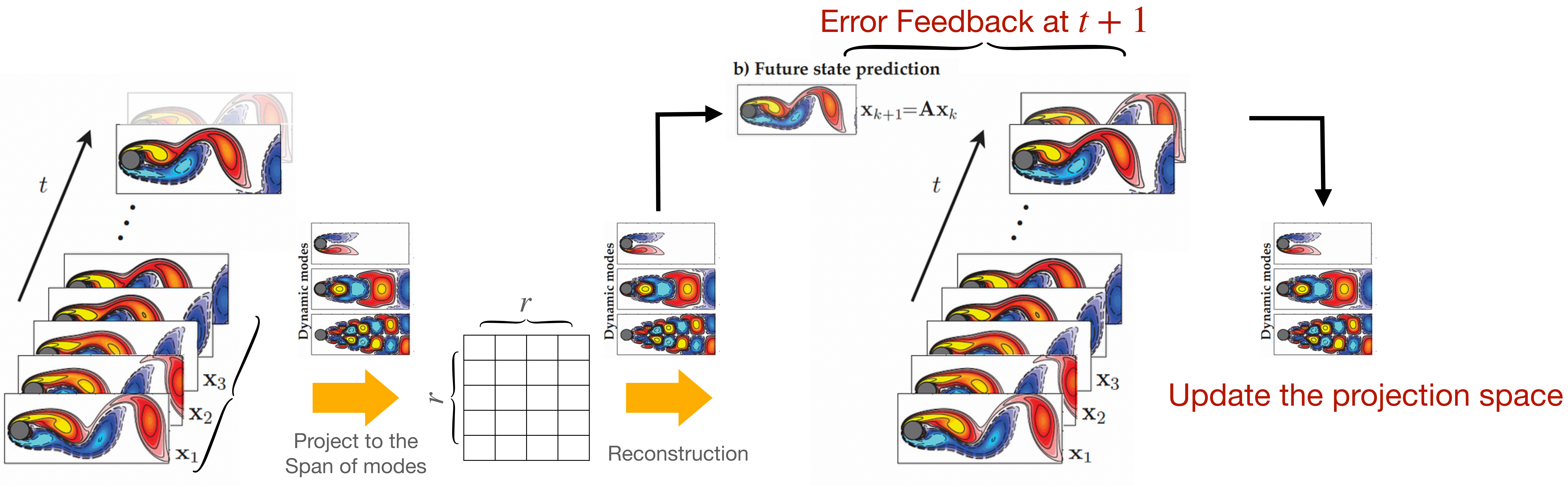
- enable dynamic mode decomposition
- Online fast update



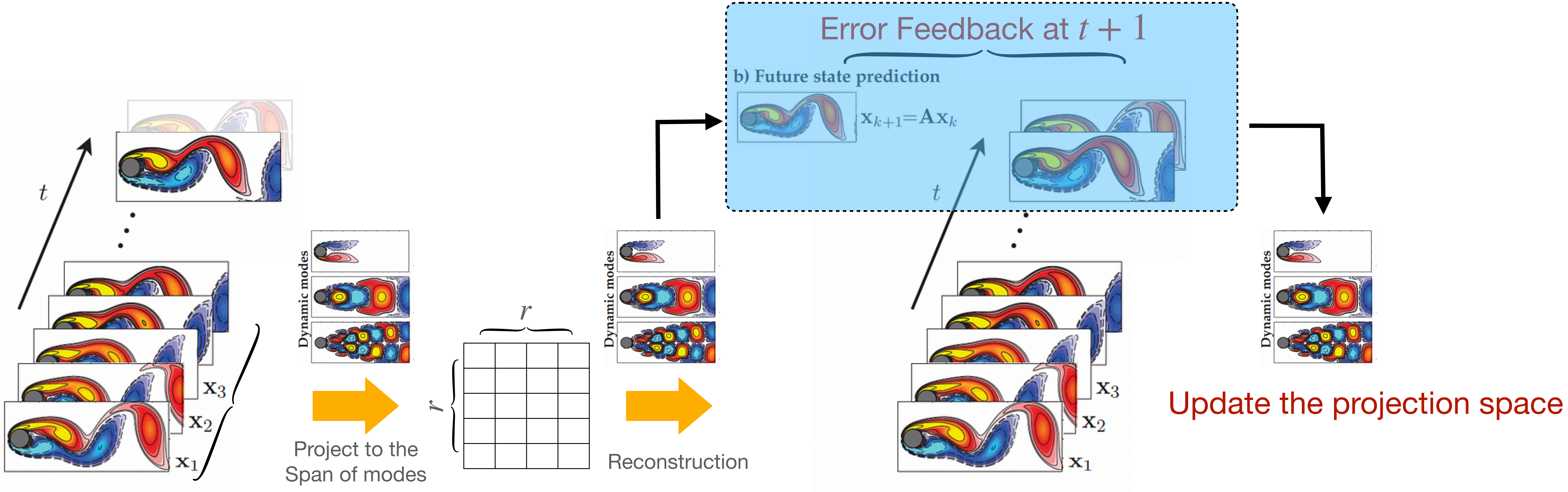
DMD with First-Order Feedback



DMD with First-Order Feedback



DMD with First-Order Feedback

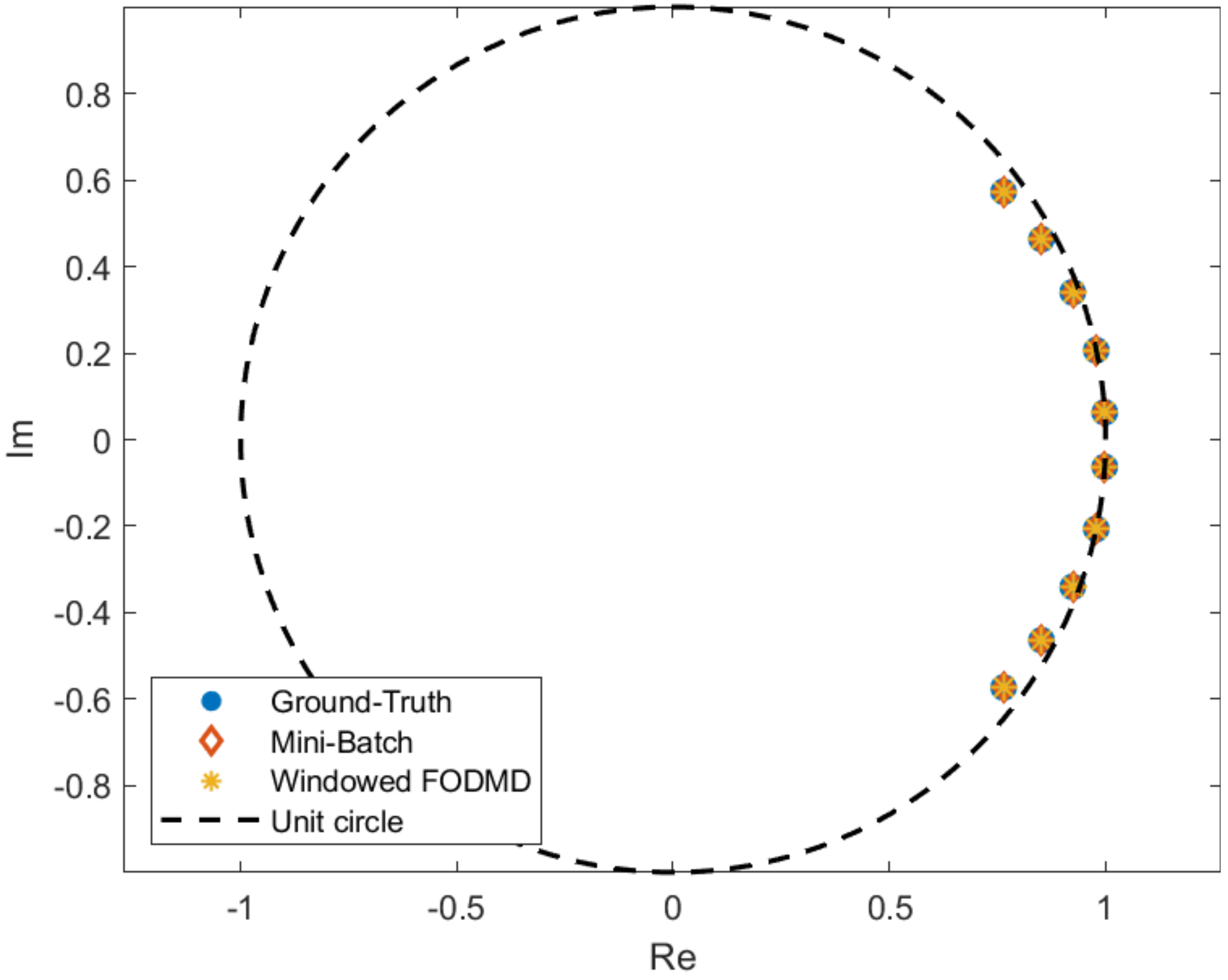


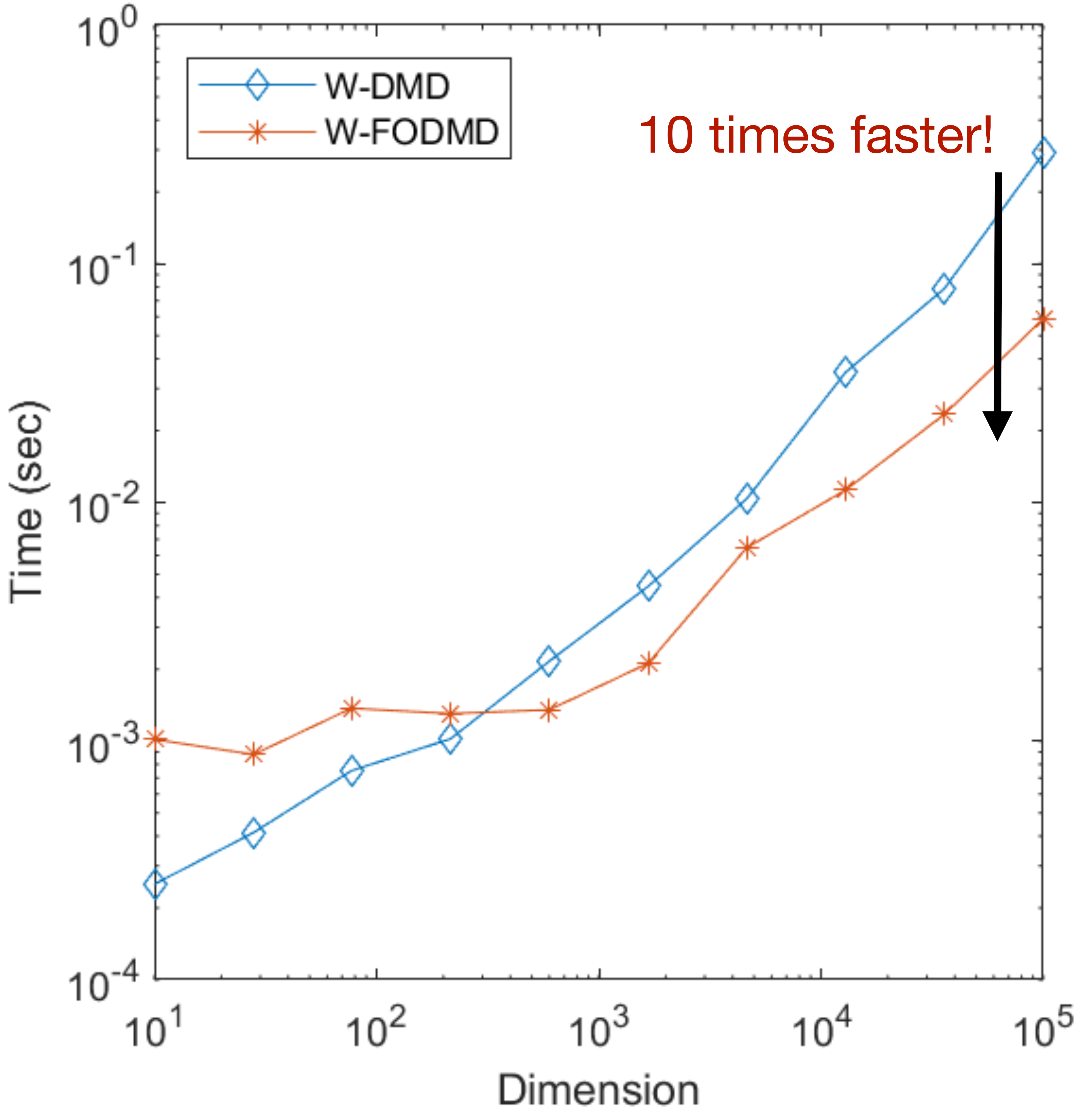
No matrix inverse, No SVD computation

Only a $n \times r$ QR decomposition

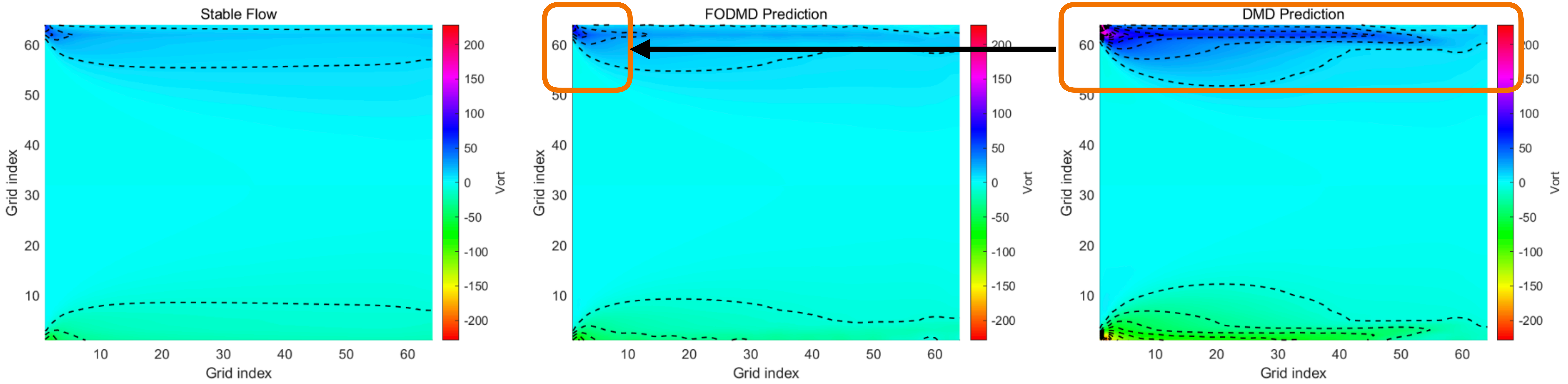
(Everything has a closed-form solution)

Faster than Recomputation!





Prediction of Tube Flow



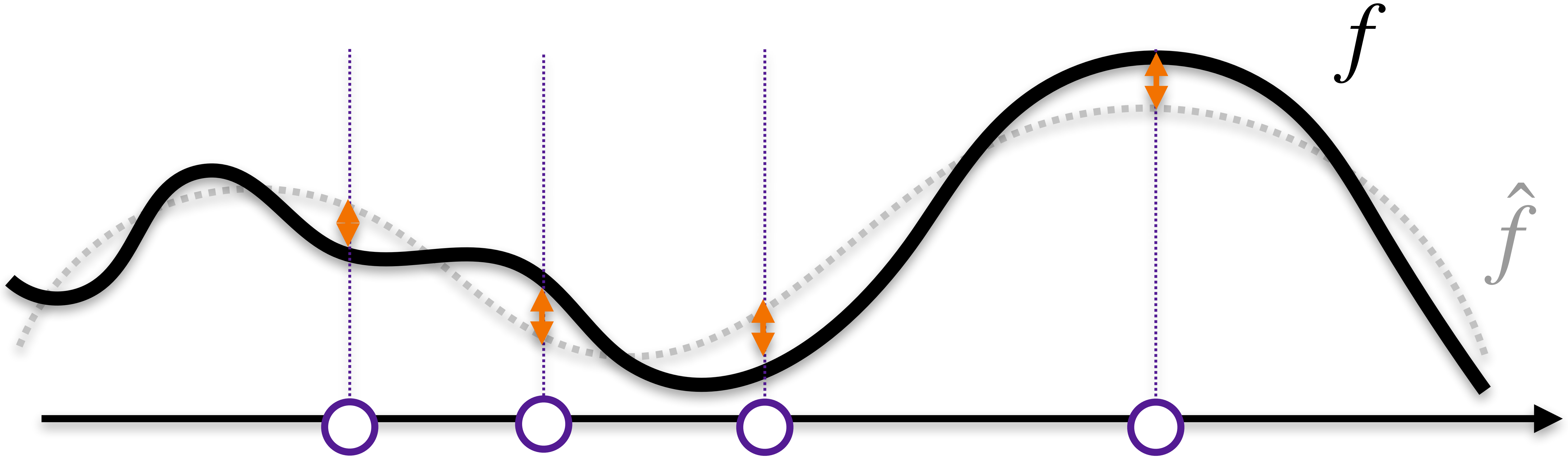
What is SCaSML about?

SCaSML

$$\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \theta \to \Phi(\theta)$$

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 $\Phi(\theta) - \Phi(\hat{\theta})$

Using ML surrogate during inference time to improve ML solution