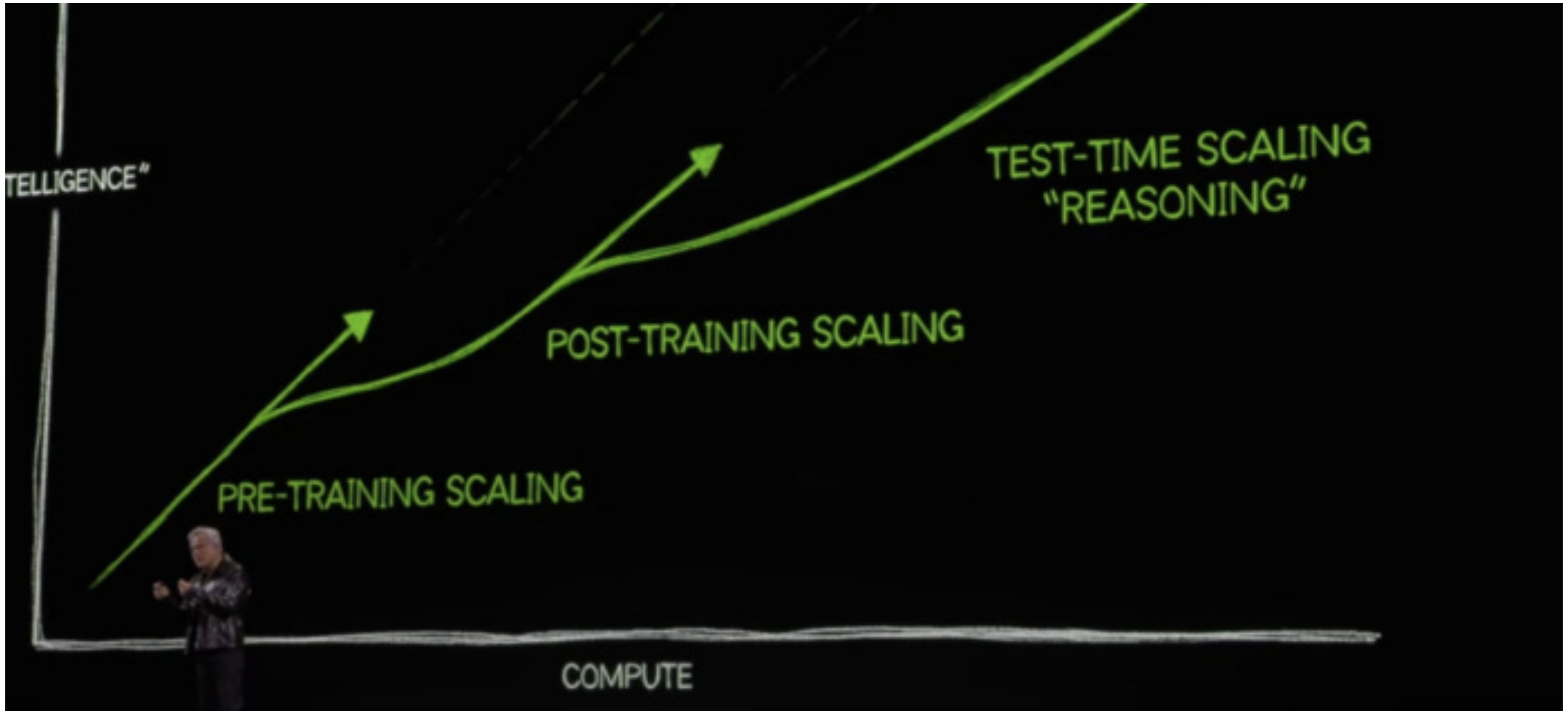
Two Tales, One Resolution for Physics-Informed Inference-time Scaling

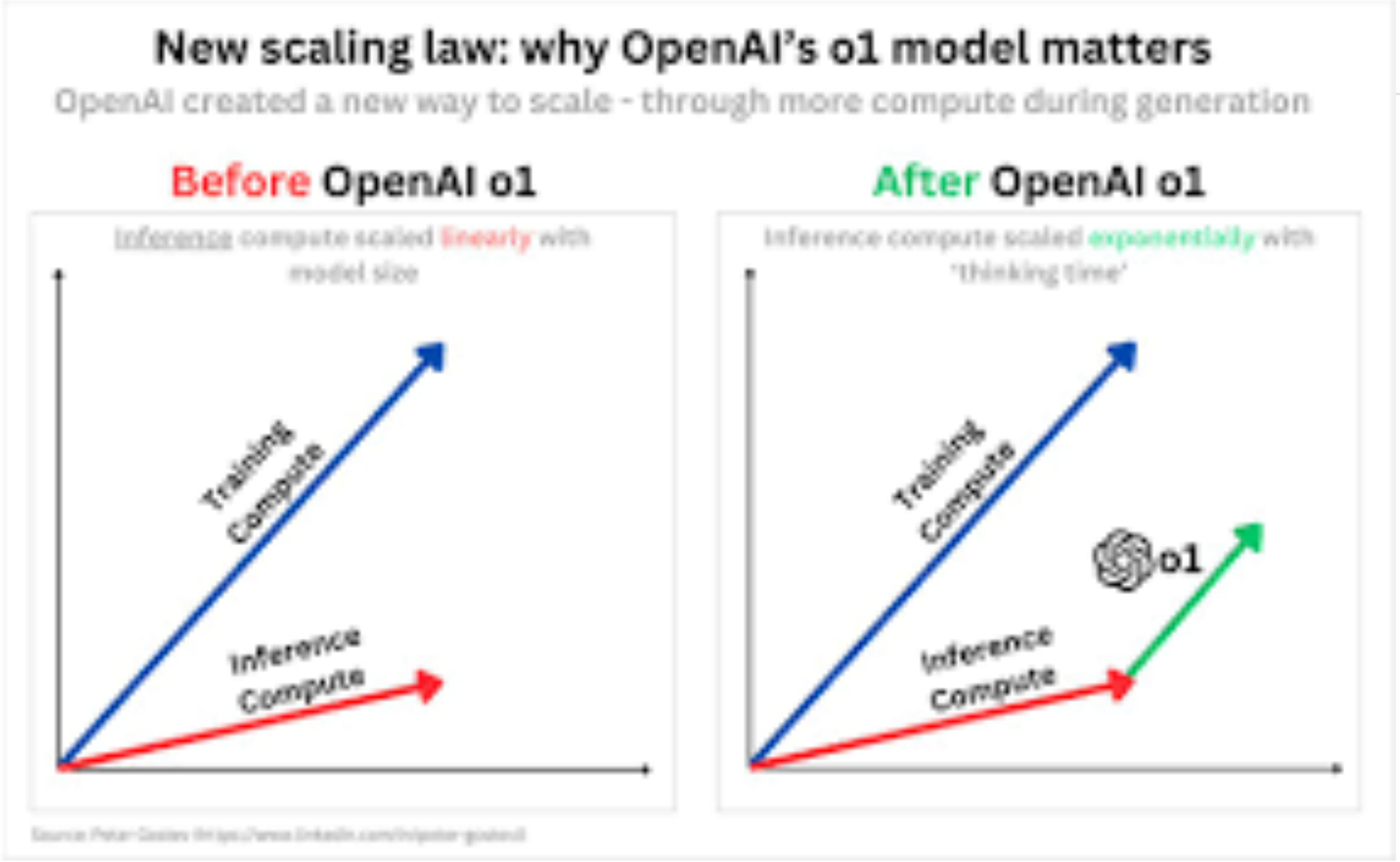
Debiasing and Precondition

Yiping Lu

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Inference Time Scaling Law





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

Tale 1: Debiasing

Hybrid Scientific Computing and Machine Learning

Consider How you use ChatGPT…

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

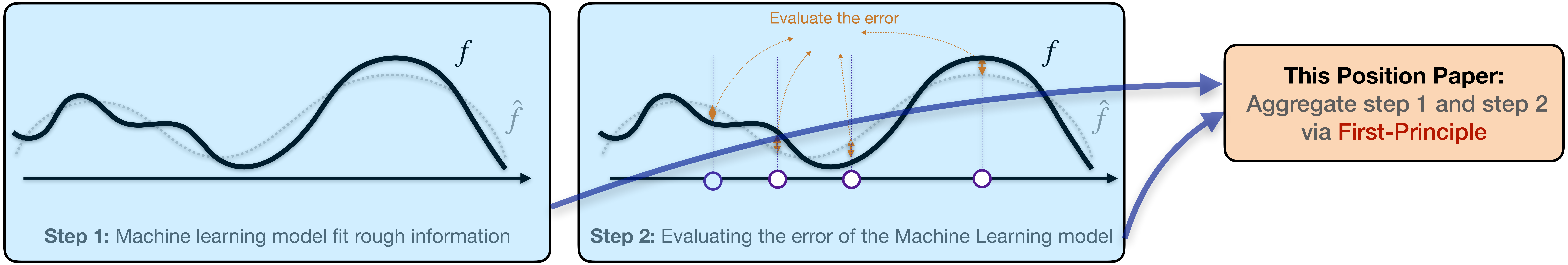
Option 1: Subway

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.

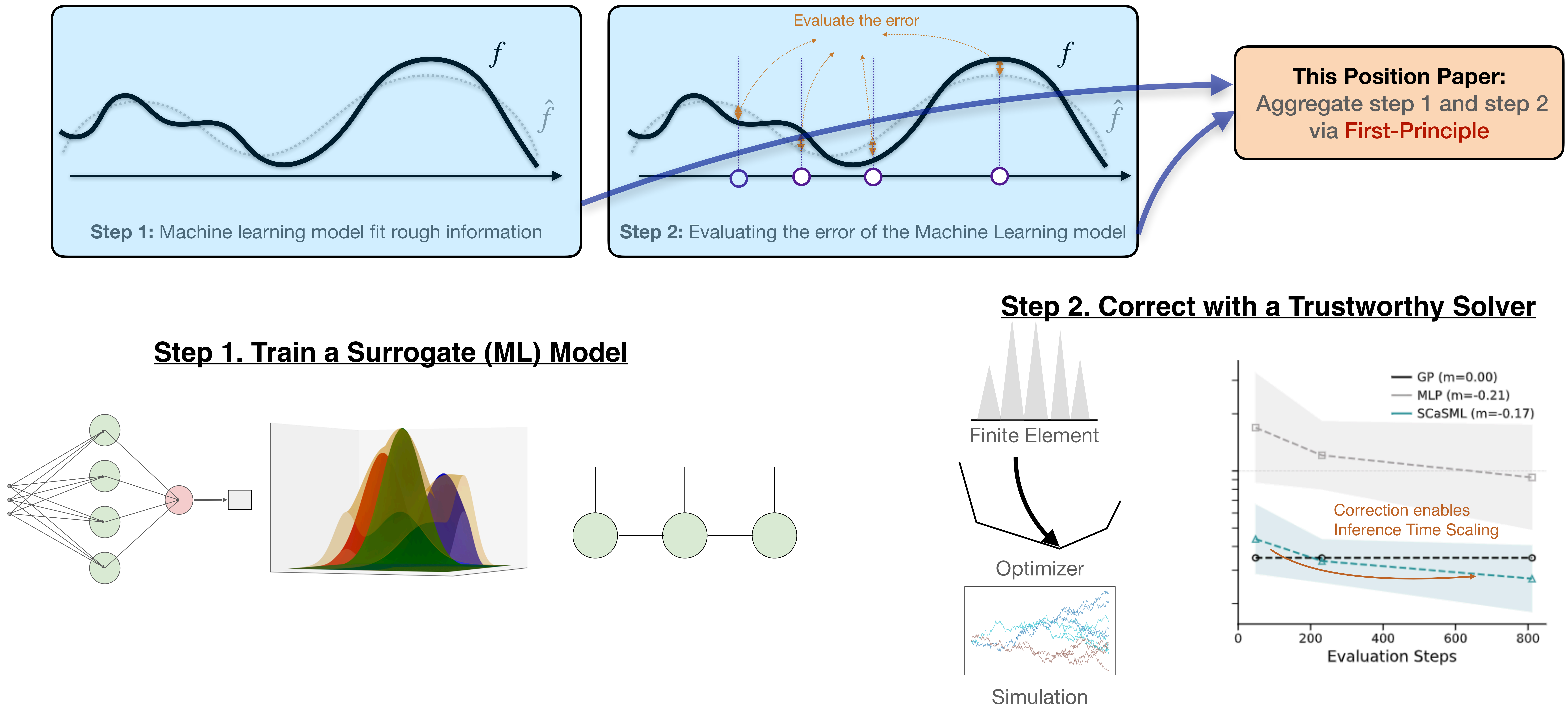
double-check with a map

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 Inference Time Scaling



Physics-Informed Inference Time Scaling



Our Framework

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

Scientific Machine Learning

Downstream application

<u>AIM:</u> Unbiased prediction even with biased machine learning estimator

Our Framework

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

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

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

Scientific Machine Learning

Downstream application

AIM: Unbiased prediction even with biased machine learning estimator

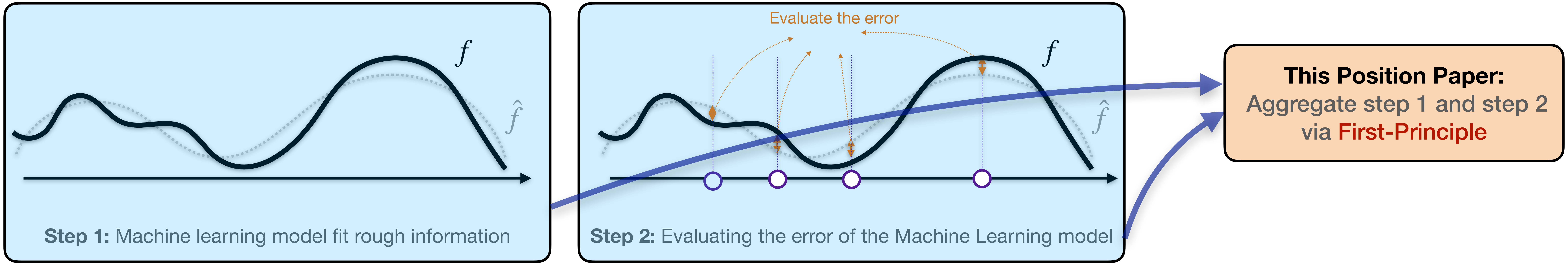
How to estimate $\Phi(\hat{\theta}) - \Phi(\theta)$?

Physics-Informed! (Structure of Φ)

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

Variance Reduction

Debiasing a Machine Learning Solution



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

Scientific Machine Learning

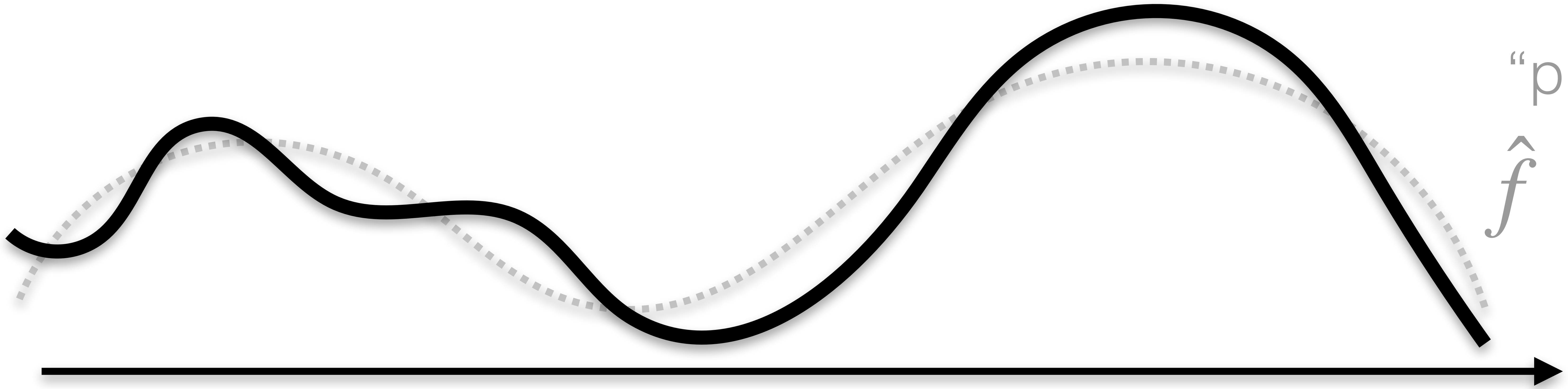
Downstream application

Example 1

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

Temperature, overall velocity…

Debiasing a Machine Learning Solution



$$\underbrace{\{X_1, \cdots, X_n\} \sim \mathbb{P}_\theta \to \hat{\theta}}_{\text{Scientific Machine Learning}} \to \underbrace{\Phi}_{\text{Downstream application}}(\hat{\theta})$$

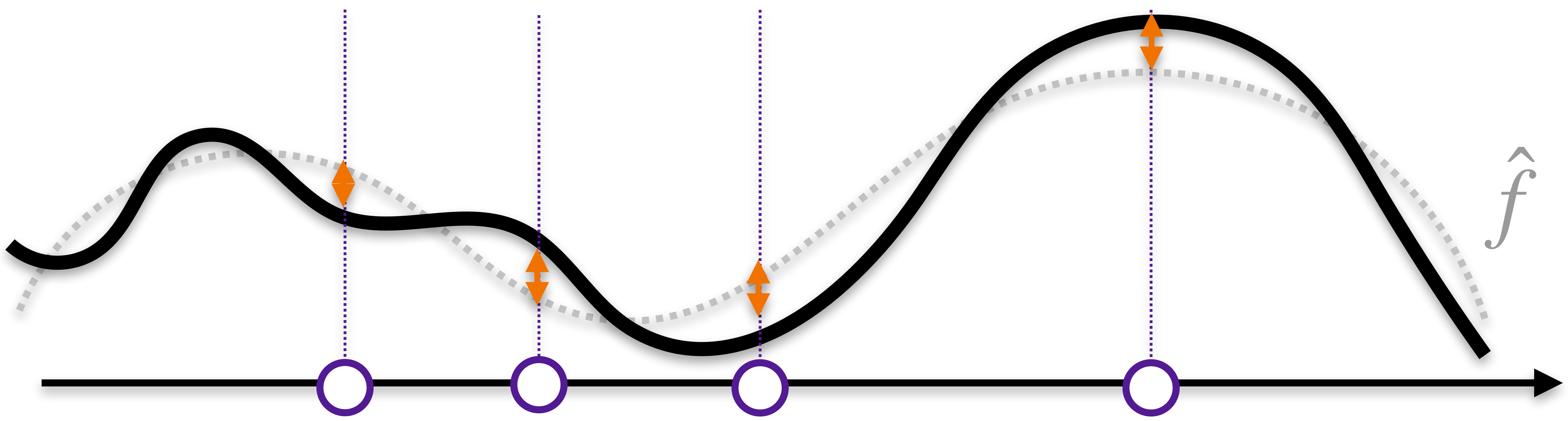
Scientific Machine Learning

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 Approah

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

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

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

Scientific Machine Learning

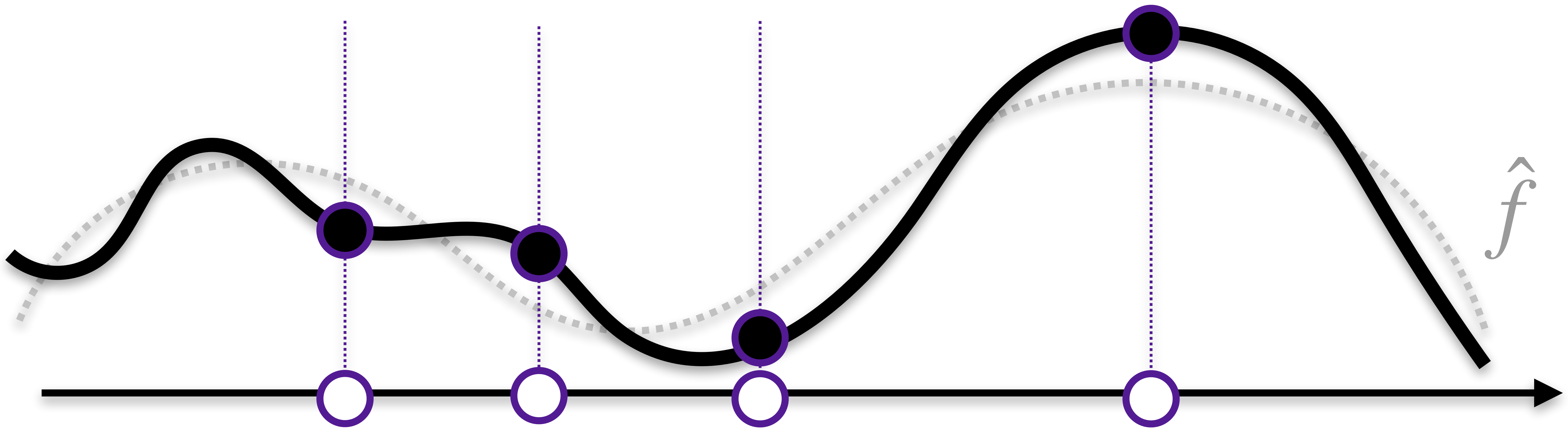
Downstream application

Example 1

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

Temperature, overall velocity…

Debiasing a Machine Learning Solution



Monte Carlo?

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

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

Scientific Machine Learning

Downstream application

Example 1

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

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**

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

Scientific Machine Learning — Downstream application

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

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

Class we consider $\left\{ f : \int \|\nabla^s f\|^p \leq 1 \right\}$

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

Scientific Machine Learning

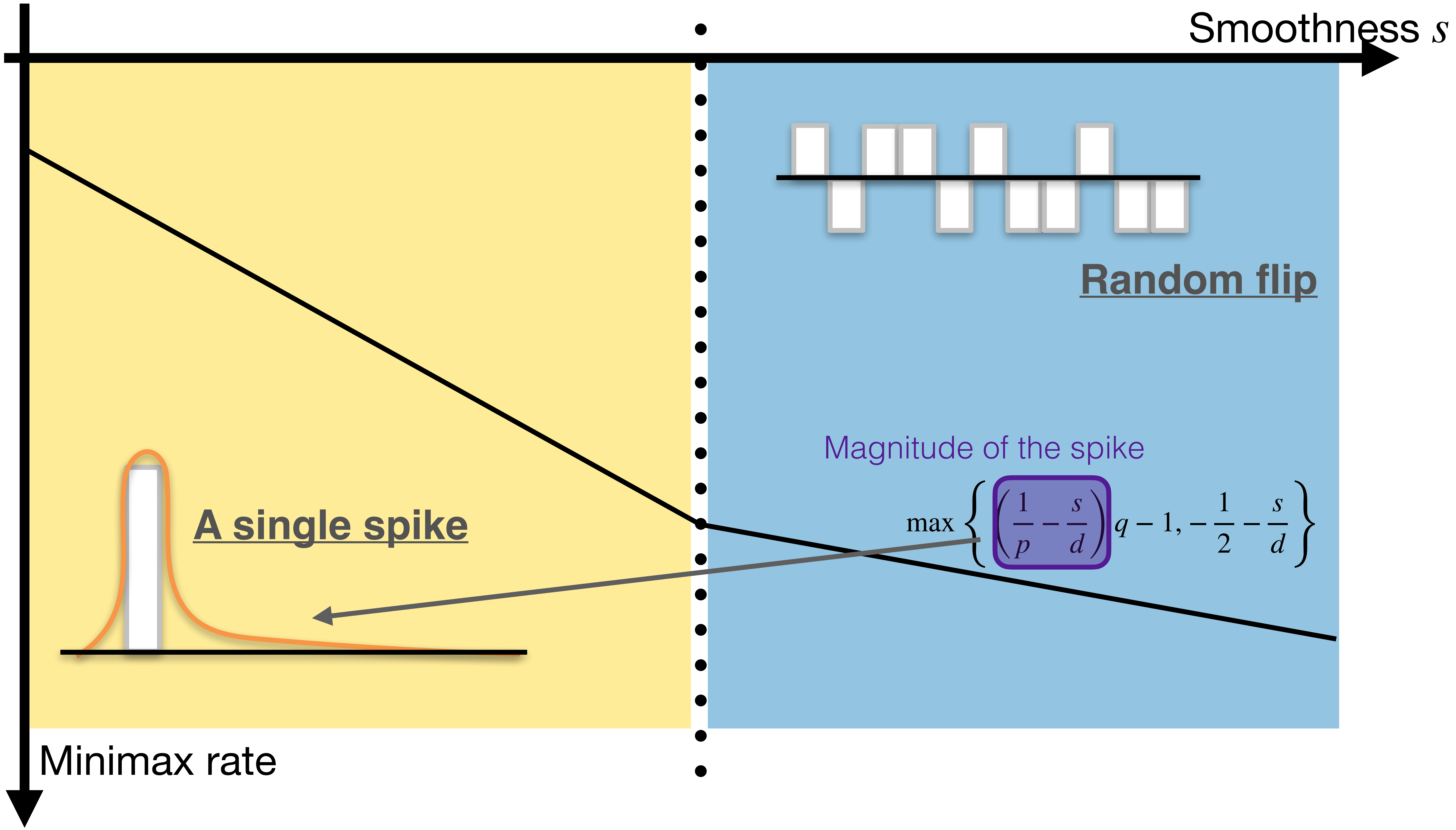
Downstream application

Example 1

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

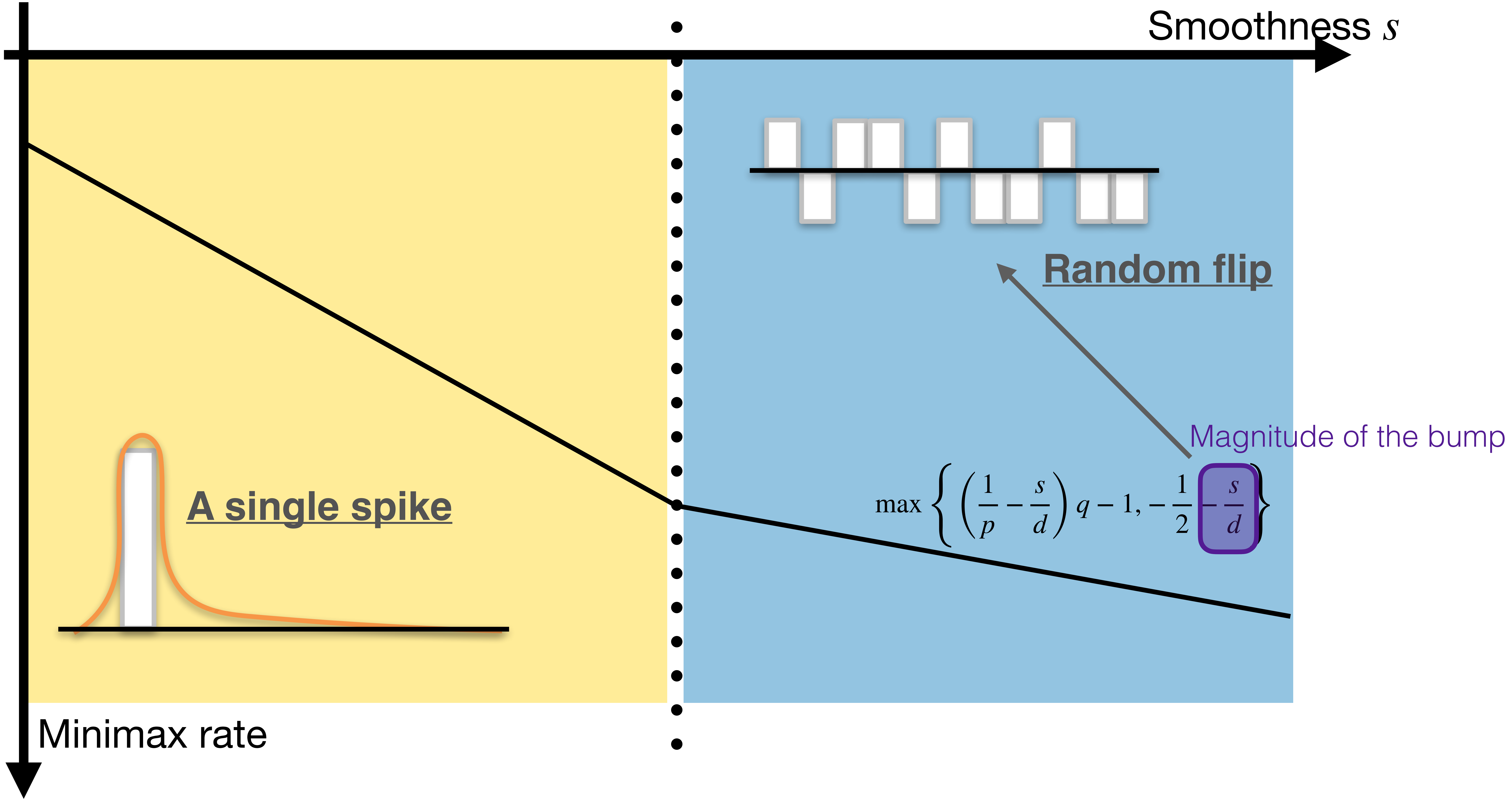
Temperature, overall velocity…

Lower Bound

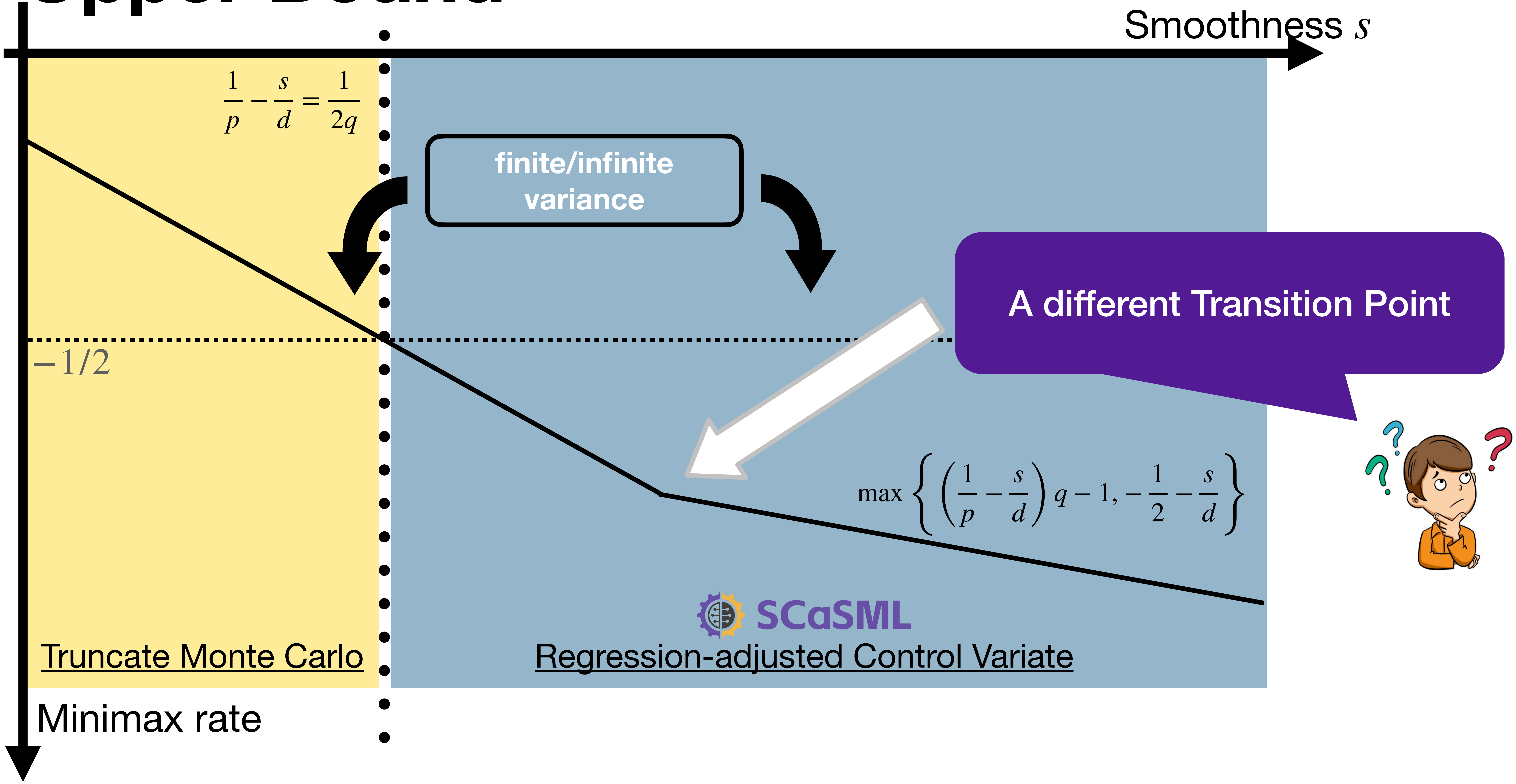


Smoothness s
Random flip
Magnitude of the spike
$\max\left\{\left(\frac{1}{p}-\frac{s}{d}\right)q-1, -\frac{1}{2}-\frac{s}{d}\right\}$
A single spike
Minimax rate

Lower Bound


Smoothness s
Random flip
Magnitude of the bump
$\max\left\{\left(\frac{1}{p}-\frac{s}{d}\right)q-1,-\frac{1}{2}-\frac{s}{d}\right\}$
A single spike
Minimax rate

Upper Bound


Smoothness s
$\frac{1}{p} - \frac{s}{d} = \frac{1}{2q}$
finite/infinite variance
A different Transition Point
$-1/2$
$\max\left\{\left(\frac{1}{p} - \frac{s}{d}\right)q - 1, -\frac{1}{2} - \frac{s}{d}\right\}$
SCaSML
Truncate Monte Carlo
Regression-adjusted Control Variate
Minimax rate

Why?

SCaSML estimate of $\mathbb{E}_P f^q, f \in W^{s,p}$

Step 1 Using half of the data to estimate $\hat{f}$

Step 2 $\mathbb{E}_P f^q = \mathbb{E}_P(\hat{f}^q) + \mathbb{E}_P(f^q - \hat{f}^q)$

Low order term

$f^{q-1}(f - \hat{f}) + (f - \hat{f})^q$

“influnce function” (gradient) **Error propagation**

How does step2 variance depend on estimation error?

Why?

SCaSML estimate of $\mathbb{E}_P f^q, f \in W^{s,p}$

Step 1 Using half of the data to estimate $\hat{f}$

Step 2 $\mathbb{E}_P f^q = \mathbb{E}_P(\hat{f}^q) + \mathbb{E}_P(f^q - \hat{f}^q)$

Low order term

$f^{q-1}(f - \hat{f}) + (f - \hat{f})^q$

"influnce function" (gradient)

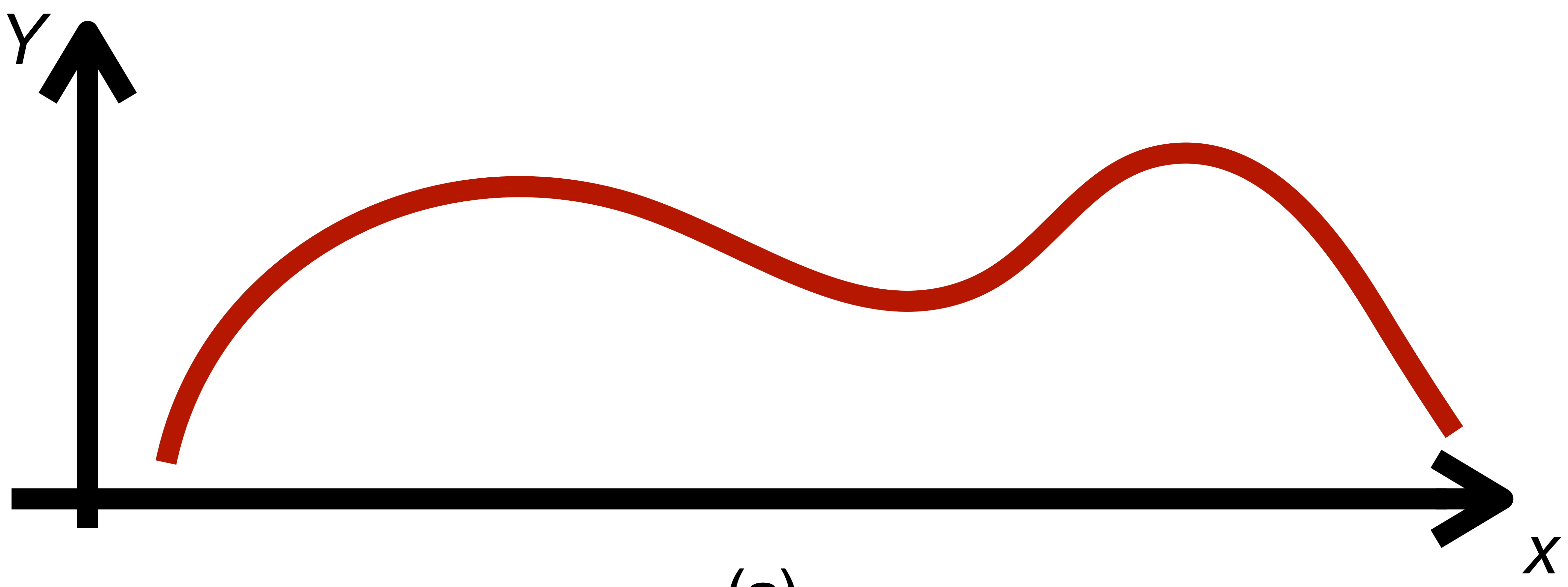
Error p

Embed f^{q-1} and $f - \hat{f}$ into "dual" space

How to select the Sobolev emebedding?

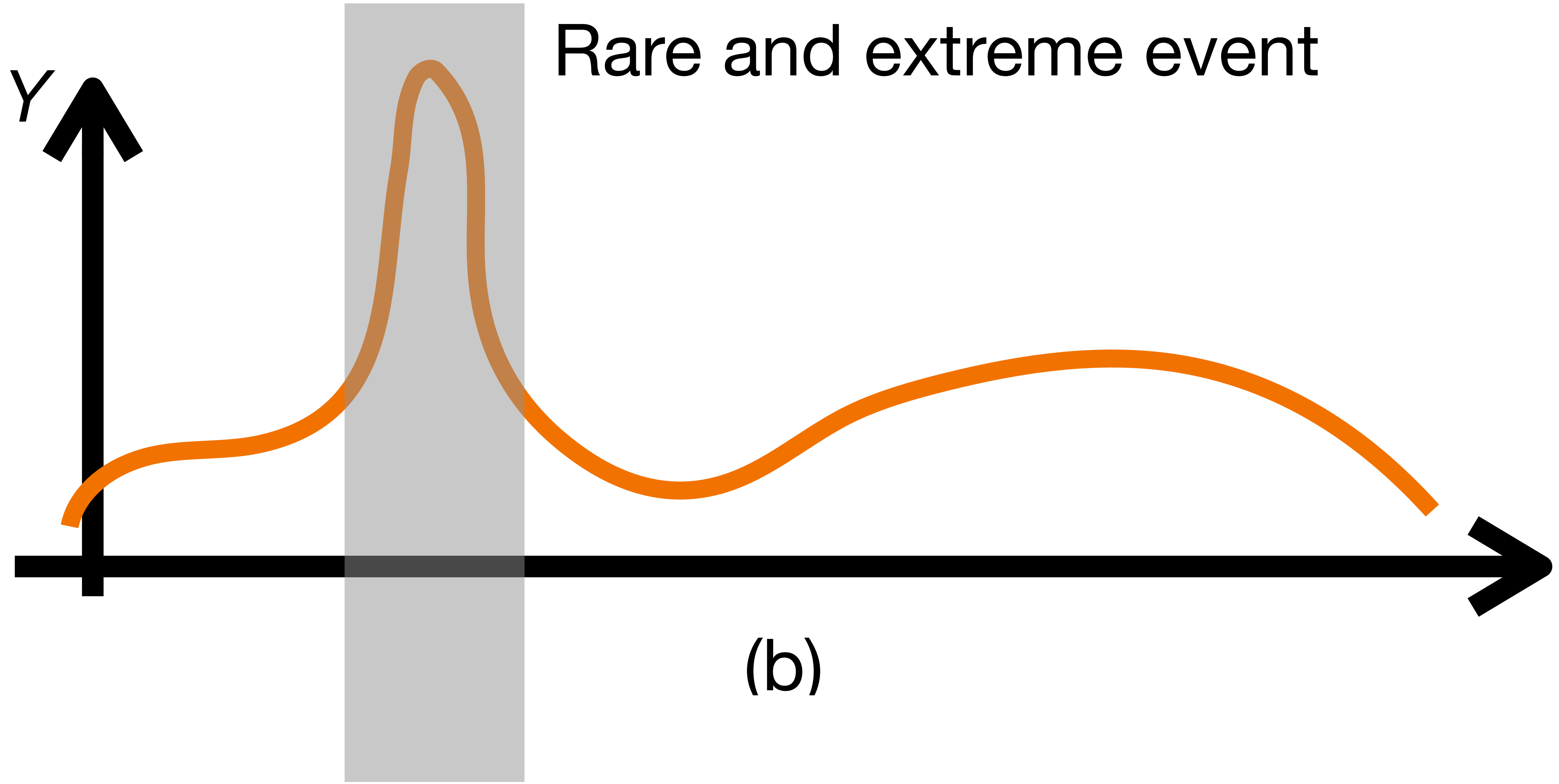
Take Home Message

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)

<u>*q* control the extremeness</u>



(b)

SCaSML

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

Scientific Machine Learning   Downstream application

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

Example 2 $\theta = A, \quad X_i = (x_i, Ax_i)$ $\Phi(\theta) = \text{tr}(A)$

Huch++

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

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

What if Φ is nonlinear?

Iterative Solver!

This Talk: Debiasing

A new way for hybrid scientific computing and machine learning

- Eigenvalue decomposition
  - Preconditioned (randomized) computation of Eigenvalue Problem via Debiasing
- PDE-Solver
  - Inference time scaling for ML-based PDE solver

High Dimensional PDE-Solving

Let's consider $\Delta u = f$

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

$X_i = (x_i, \Delta u(x_i))$ — PDE R.h.s

u — PDE solution

PINN/DRM/Neural Galerkin

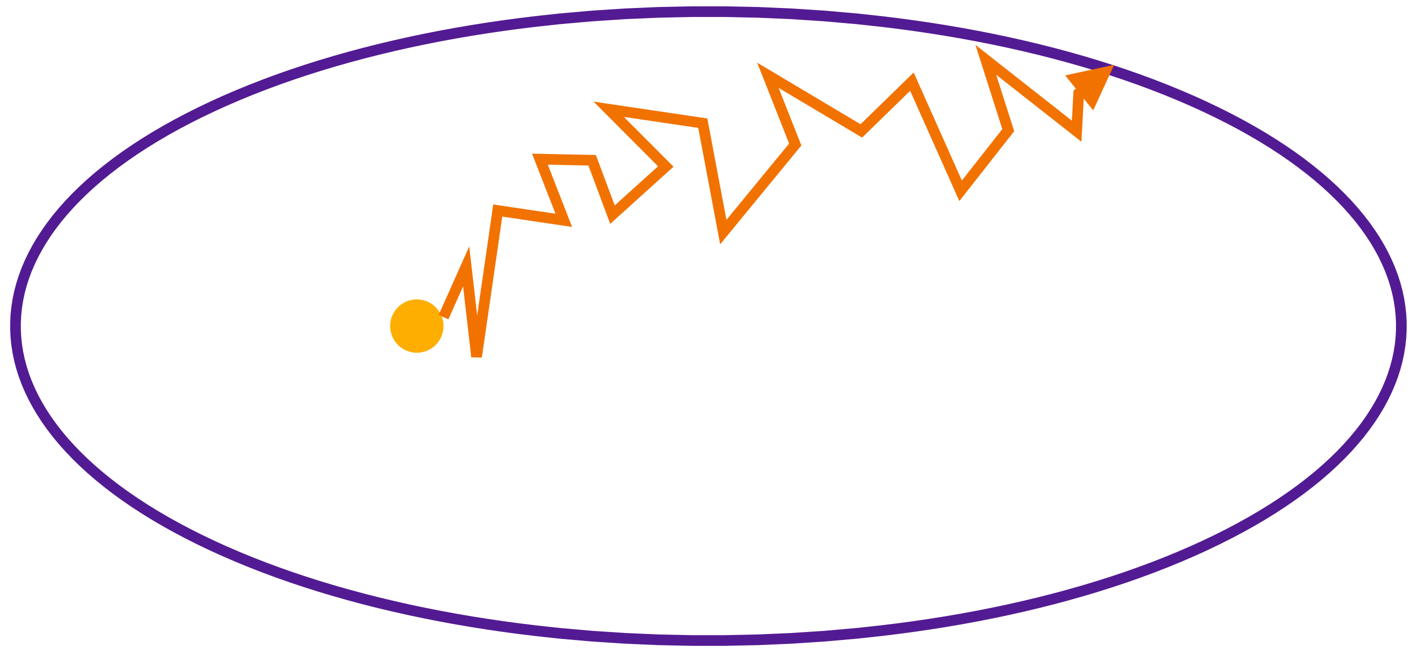
Mean/Variance/$u(x)$

$$\Delta u = f$$

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

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

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



Works for Semi-linear PDE

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

Keeps the structure to enable brownian motion simulation

Can you do simulation for nonlinear equation?

Δ is linear!

Works for Semi-linear PDE

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

Keeps the structure to enable brownian motion simulation

NN

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

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

Works for Semi-linear PDE

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

Keeps the structure to enable brownian motion simulation

NN

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

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

Subtract two equations

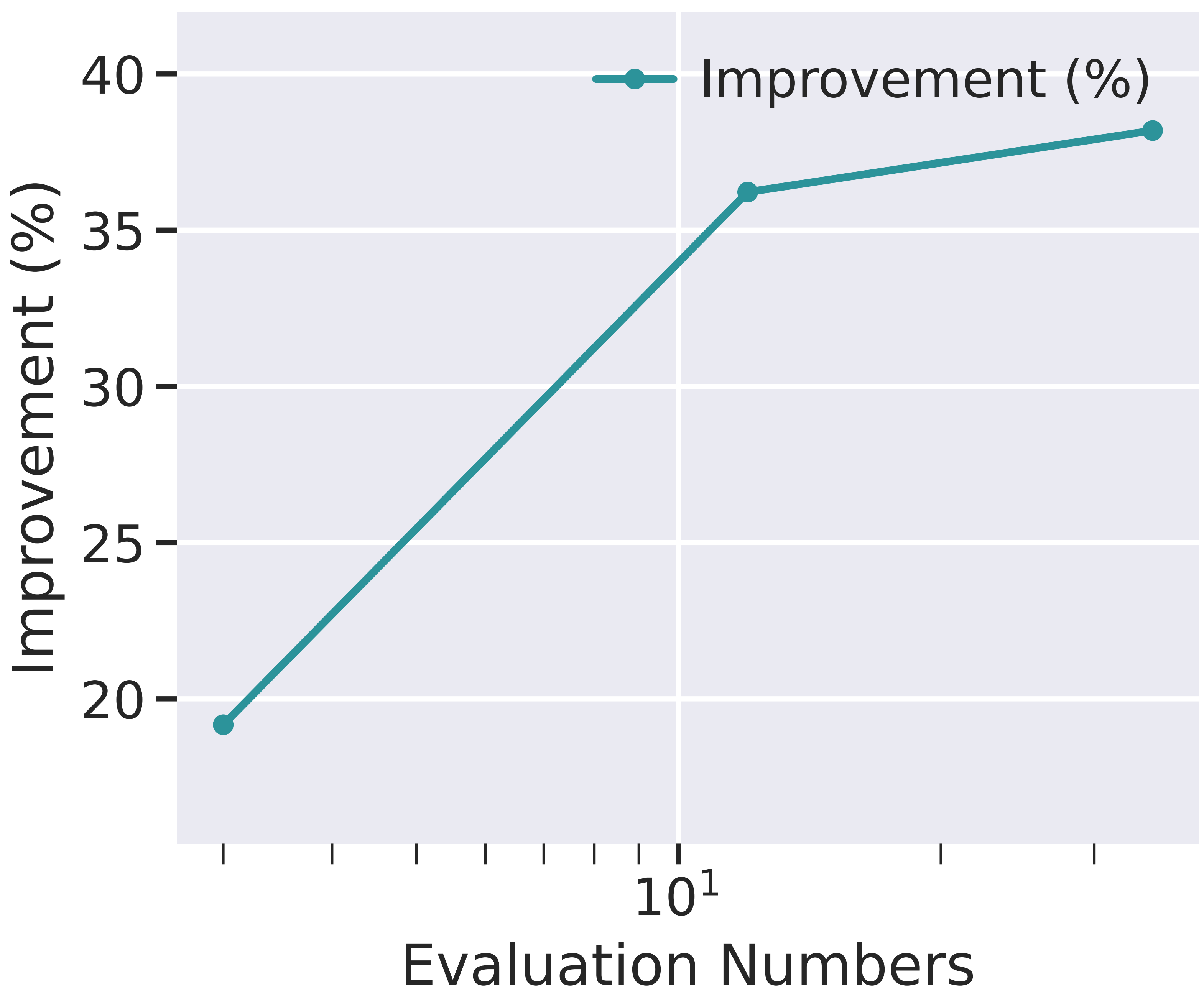
Keeps the linear structure

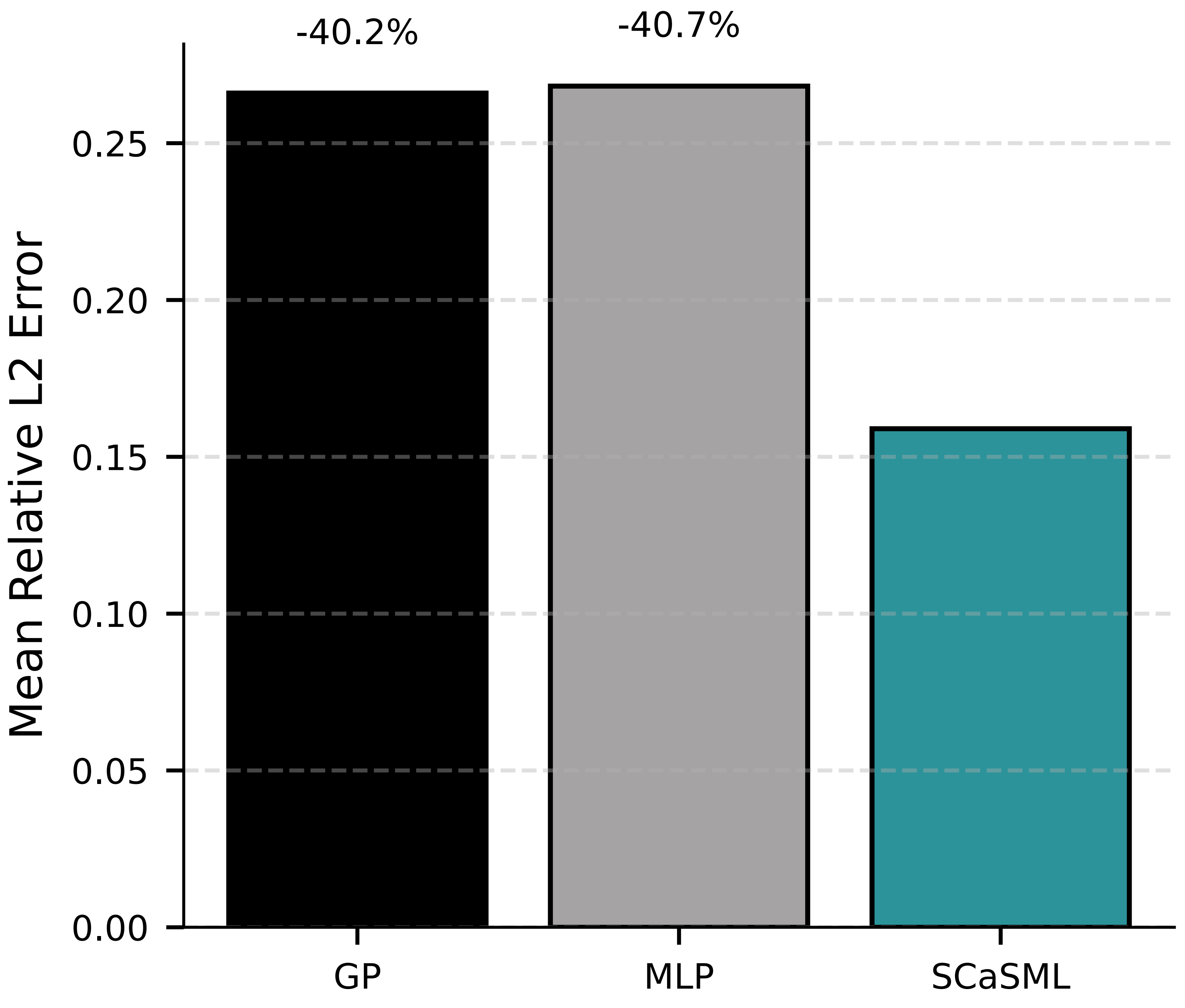
$$\frac{\partial (U-\hat{U})}{\partial t}(x,t) + \Delta(U-\hat{U})(x,t)) + \underbrace{f(t,\hat{U}(x,t) + U(x,t) - \hat{U}(x,t)) - f(t,\hat{U}(x,t))}_{G\left(t,(U-\hat{U})(x,t)\right)} = g(x,t)\,.$$

Inference-Time Scaling

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

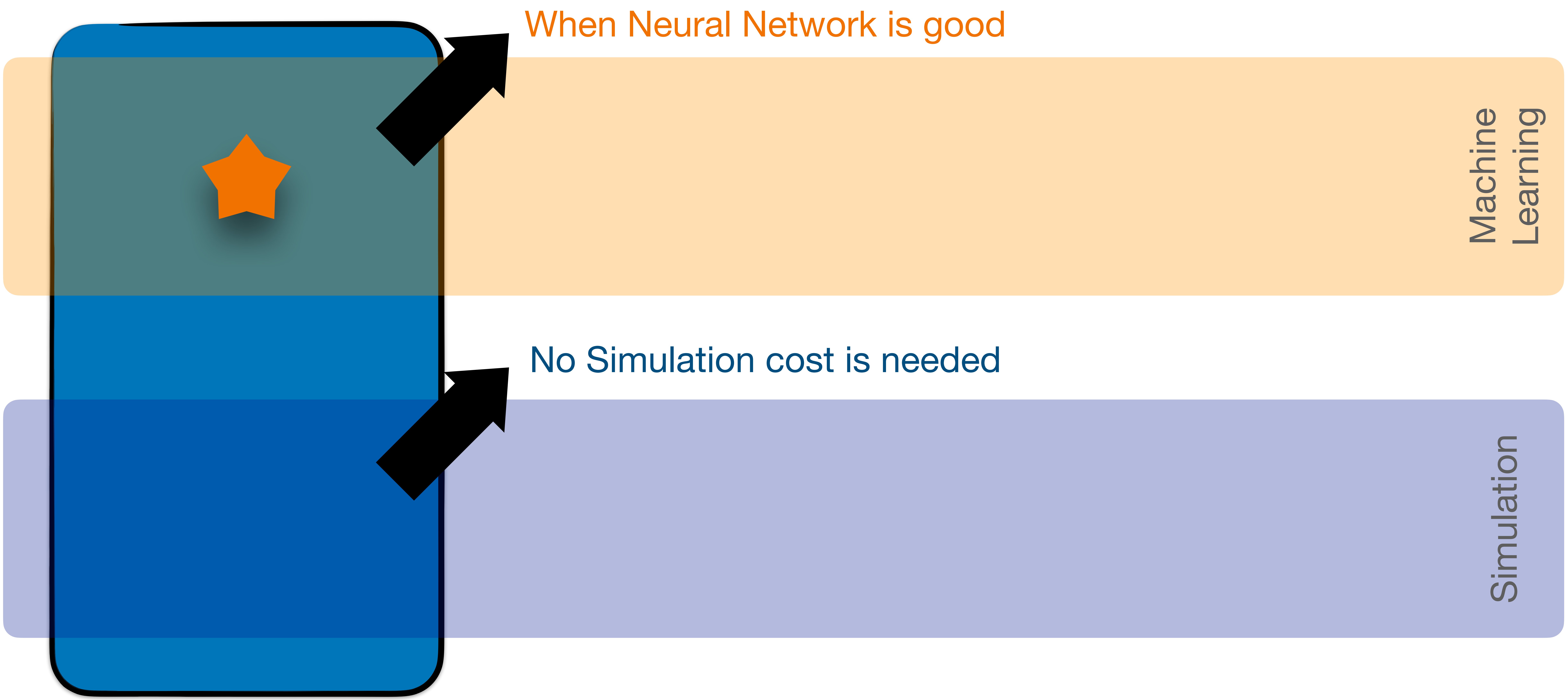
have closed-form solution $g(x) = \frac{\exp(T + \sum_i x_i}{1 + \exp(T + \sum_i x_i)}$



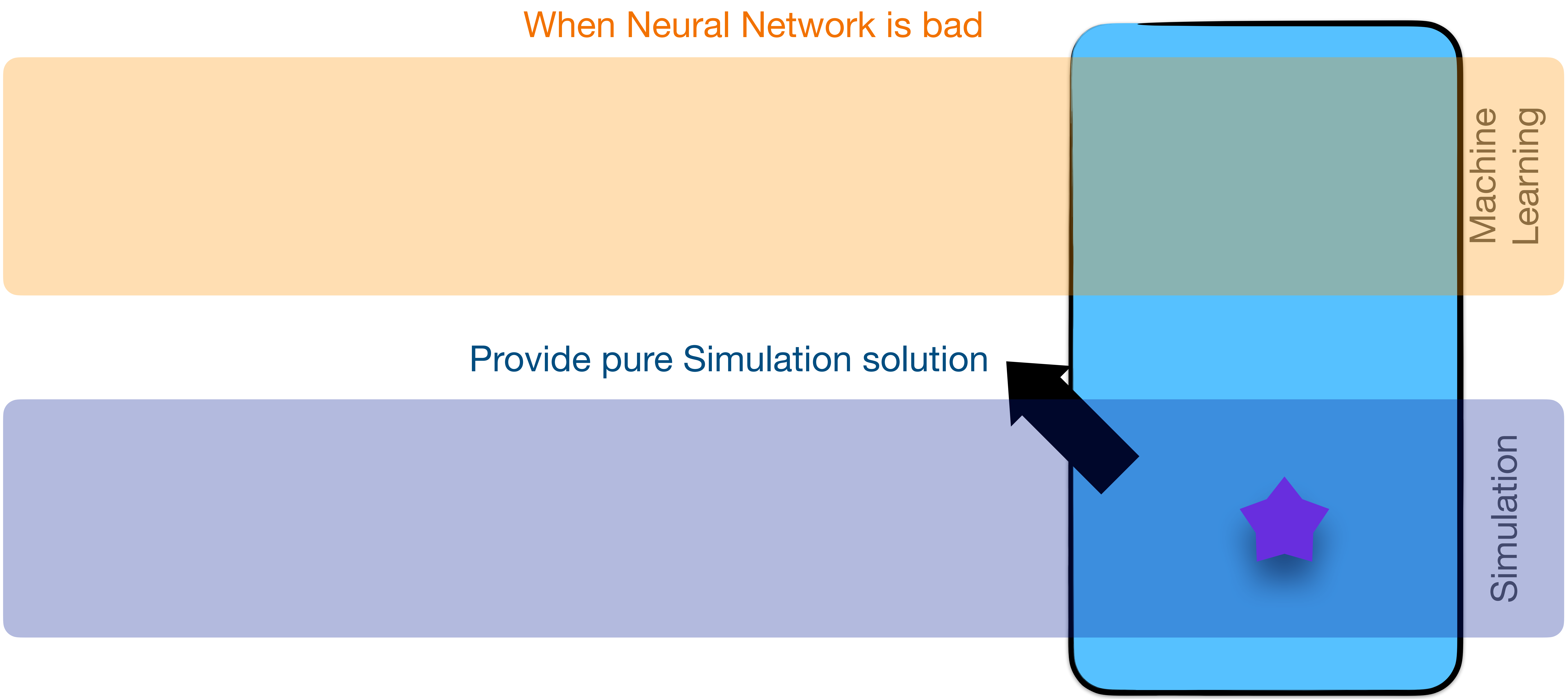


Method	Convergence Rate
PINN	$O(n^{-s/d})$
MLP	$O(n^{-1/4})$
ScaSML	$O(n^{-1/4-s/d})$

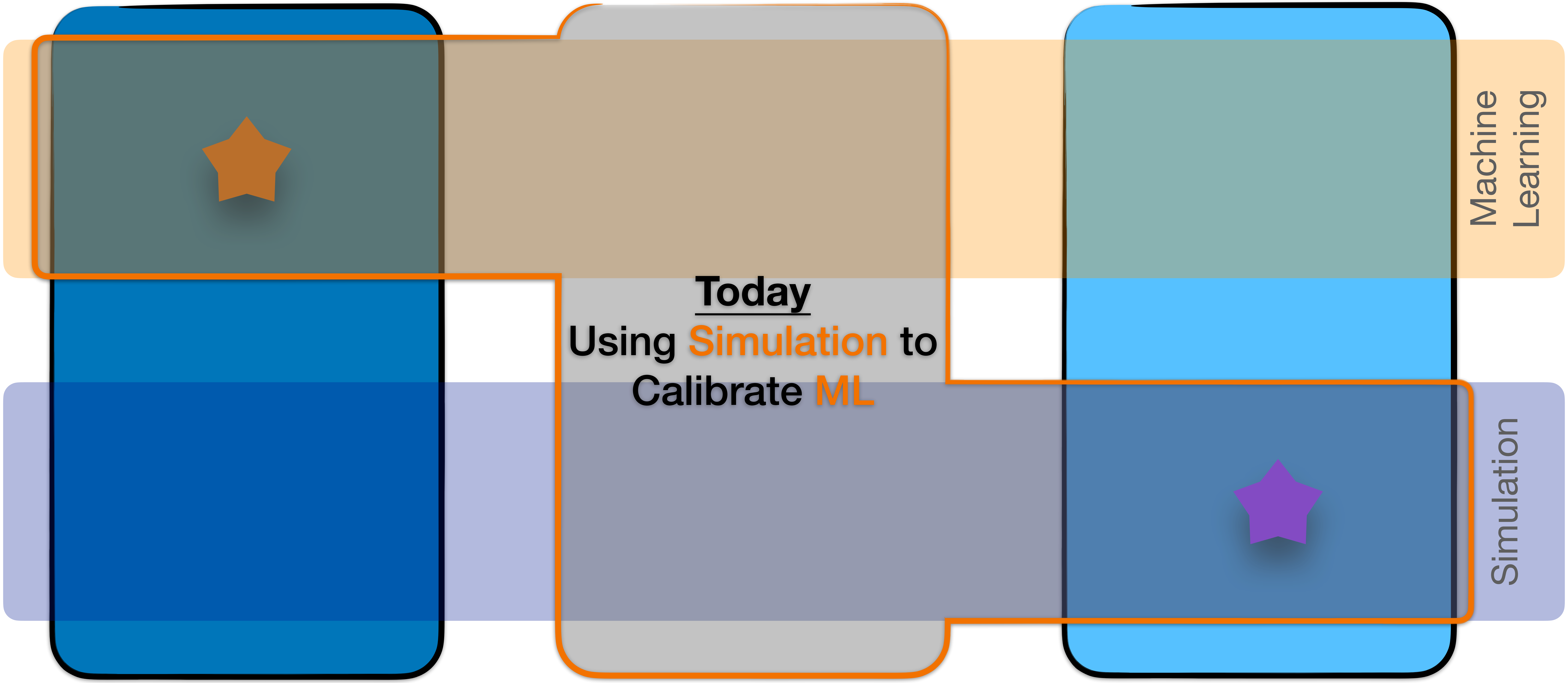
Our Aim Today : A Marriage



Our Aim Today : A Marriage



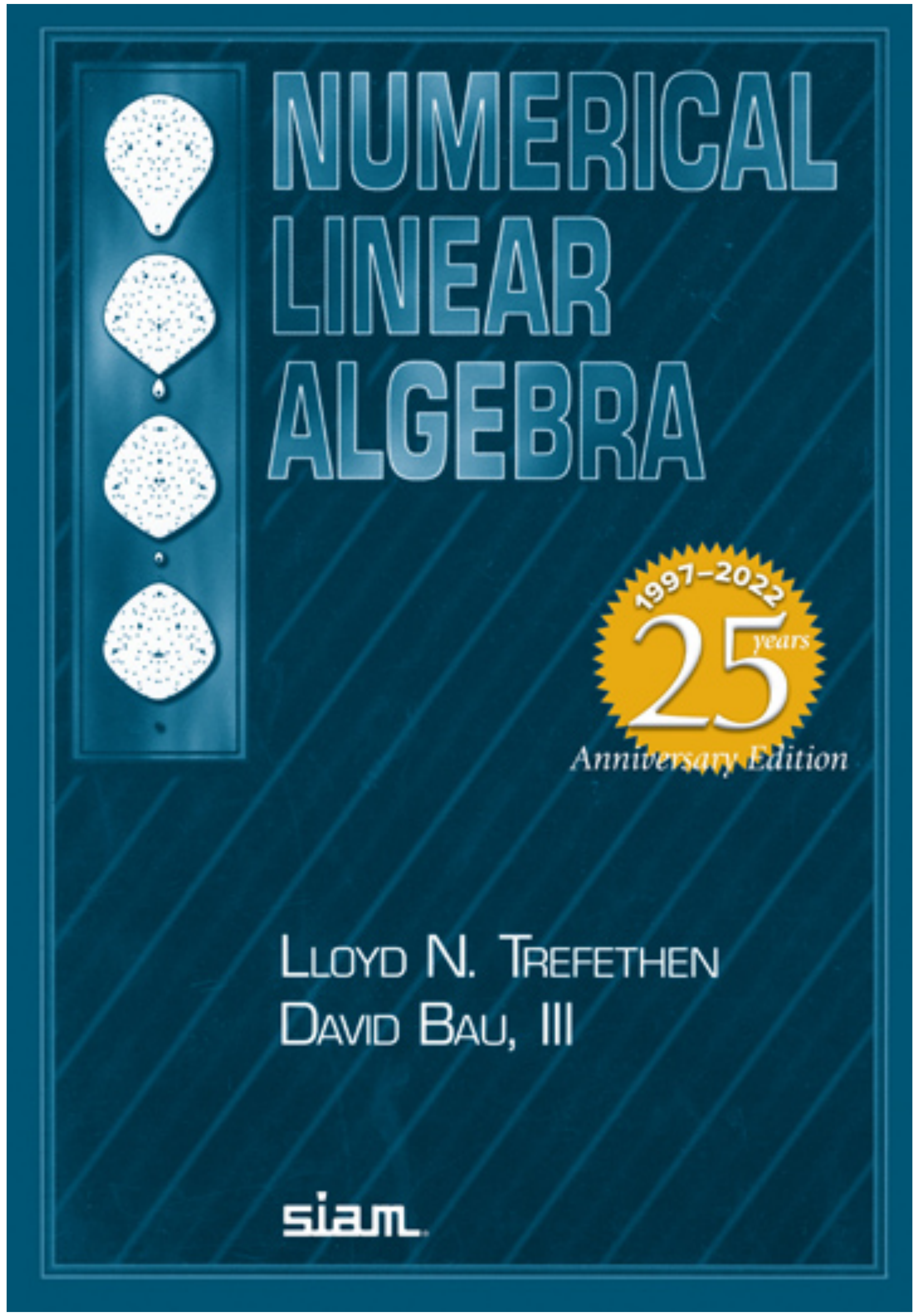
Our AIM Today: A Marriage



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 $BAx = Bb$

hardness depend on $\kappa(A)$

hardness depend on $\kappa(BA)$

Become easier when $B \approx A^{-1}$

A New Way to Implement Precondition

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

Error depends on $\|A^{-1}(A - B)\|$

A New Way to Implement **Precondition**

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

Error depends on $\|A^{-1}(A - B)\|$

  - $x - x_1$ 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 $Bx_1 = b$
  - $x - x_1$ satisfies the equation $A(x - x_1) = b - Ax_1$
  - Using the approximate solver to approximate $x - x_1$ via $Bx_2 = b - Ax_1$

Error depends on $\|A^{-1}(A - B)\|$

What is the error of $x_1 + x_2$?

$A(x_1 + x_2) = b - \underbrace{(A - B)}x_2$ Same level as $\|A^{-1}(A - B)\|$

Brings another $\|A^{-1}(A - B)\|$

$\|A^{-1}(A - B)\|^2$

Hardness depends on how $A^{-1}b$ near identity!

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 $A(u) = b$
- **Step 2:** Build an approximate solver $A(\hat{u}) \approx b$
  - Via machine learning/sketching/finite element….
- **Step 3:** Solve $u - \hat{u}$

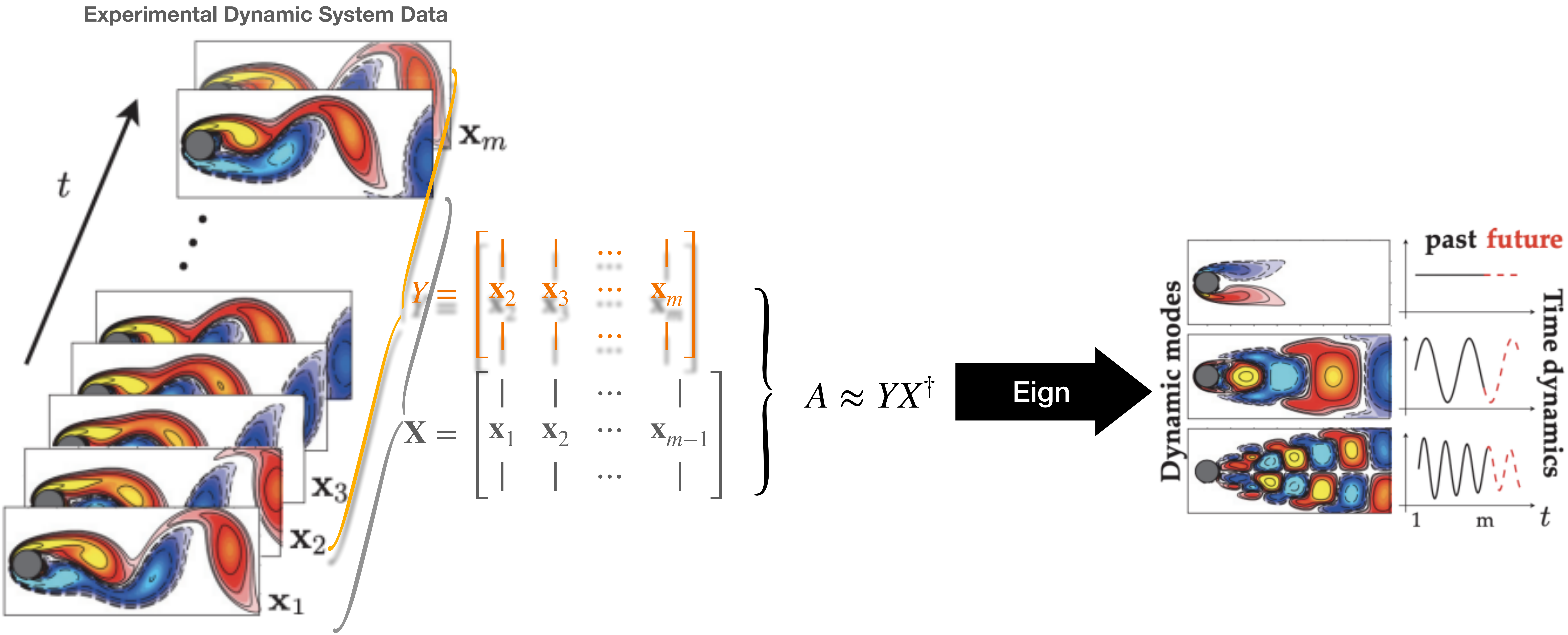
use Machine Learning

Unrealiable approximate solver as preconditioner

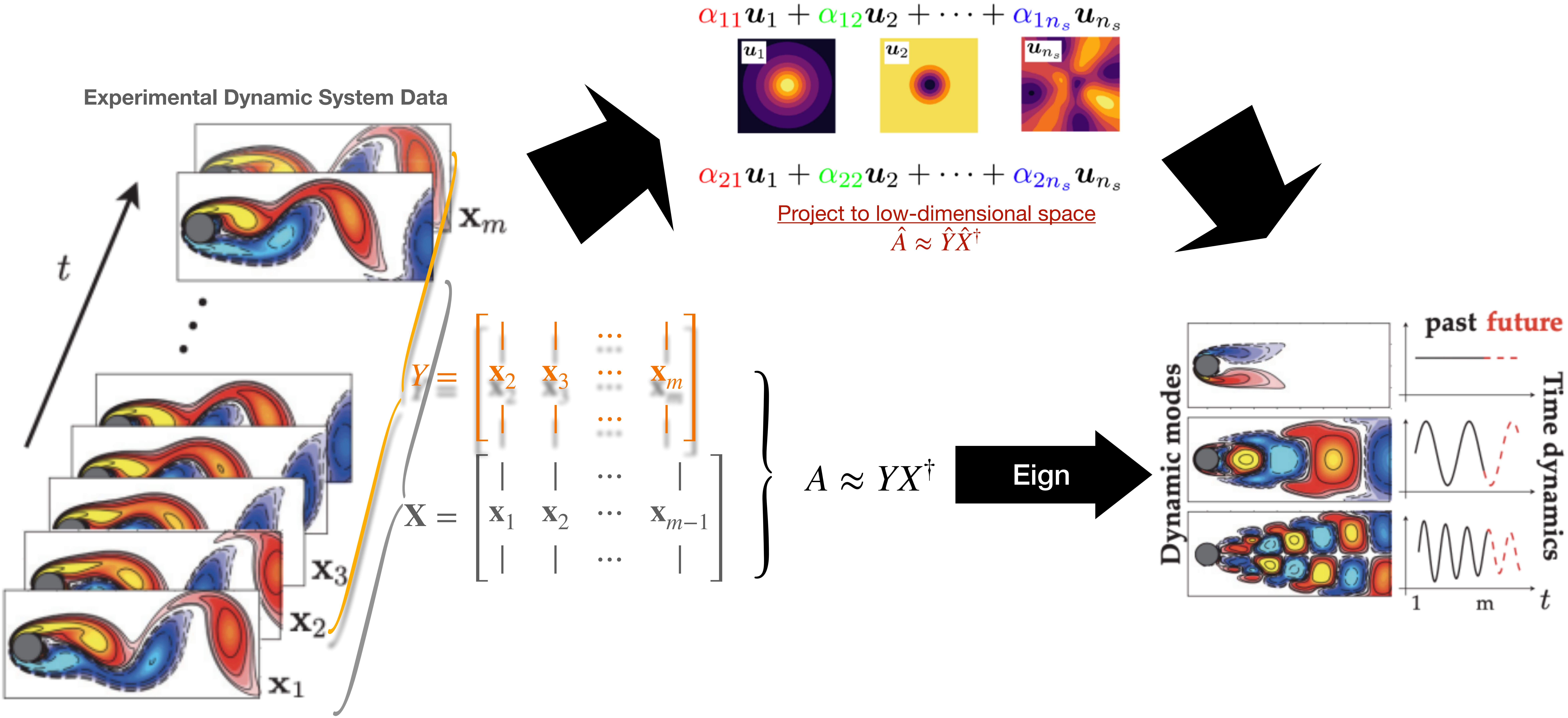
Connection with control variate, doubly robust estimator, Multifidelity Monte Carlo

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

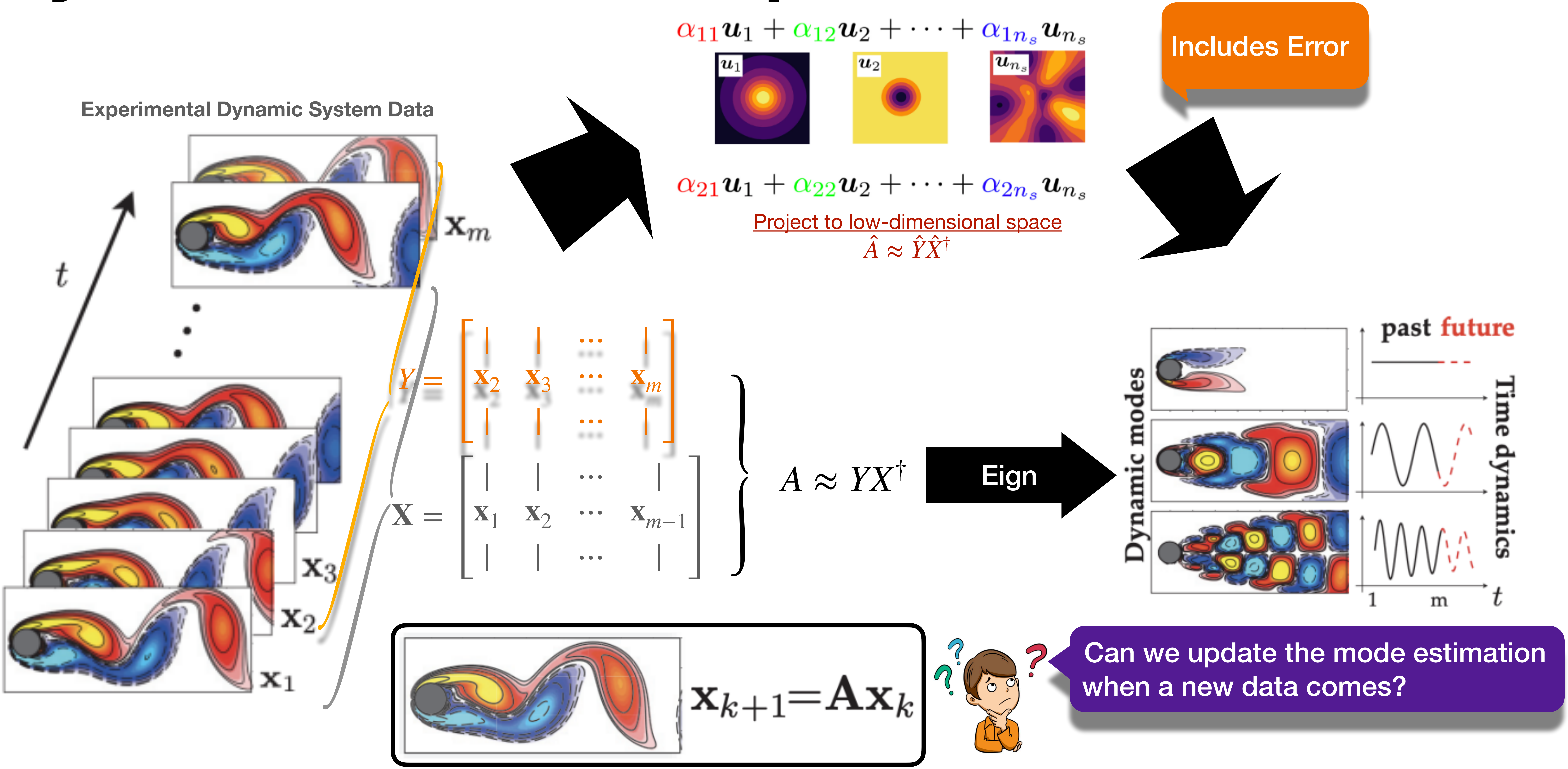
Dynamic Mode Decomposition



Dynamic Mode Decomposition



Dynamic Mode Decomposition

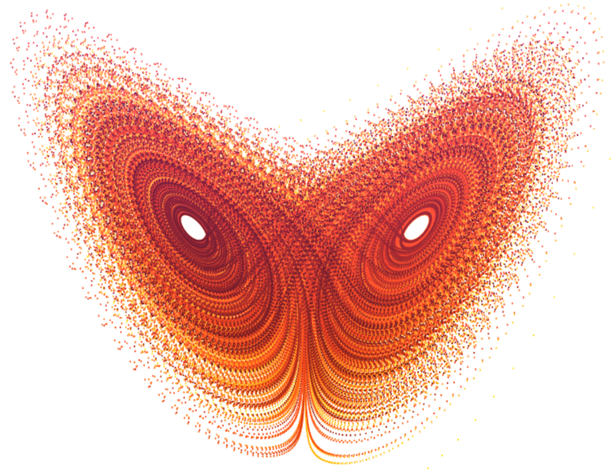


A Data-Driven View

Machine Learning ➡ $\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \to \hat{\theta} \in \Theta$

A Data-Driven View

Machine Learning ➡ $\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \to \hat{\theta} \in \Theta$



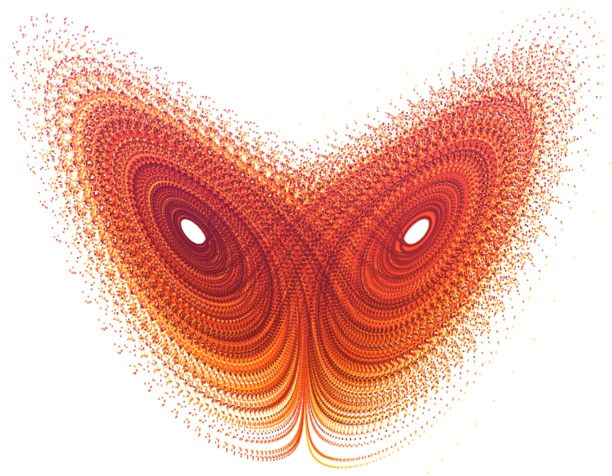
$$\frac{dx(t)}{dt} = Ax(t)$$

$\theta = A$

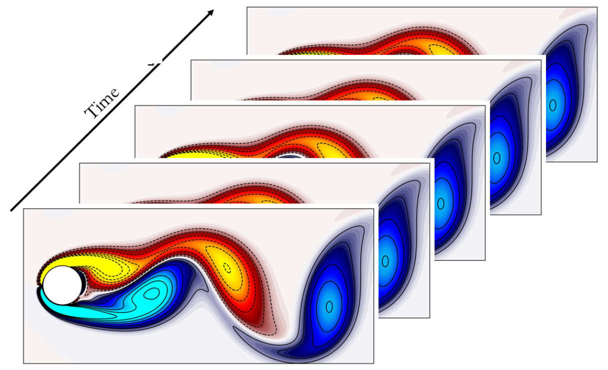
A Data-Driven View

Machine Learning →

$$\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \rightarrow \hat{\theta} \in \Theta$$



$$\frac{dx(t)}{dt} = Ax(t)$$



Snapshot Data

$$\underbrace{\{(x_i, Ax_i)\}_{i=1}^n \qquad \theta = A}$$

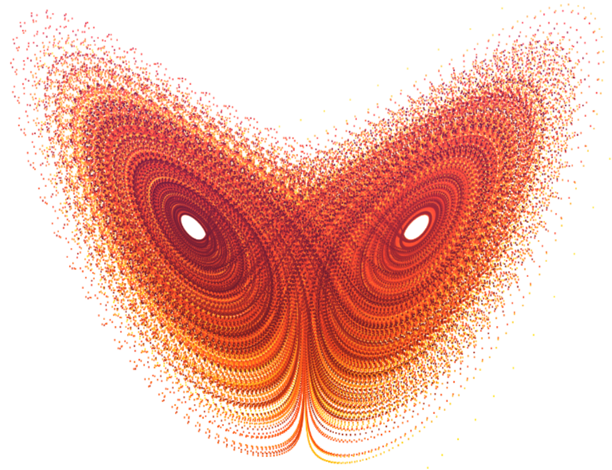
Project to a subspace

Dynamic Mode Decomposition/Randomized SVD

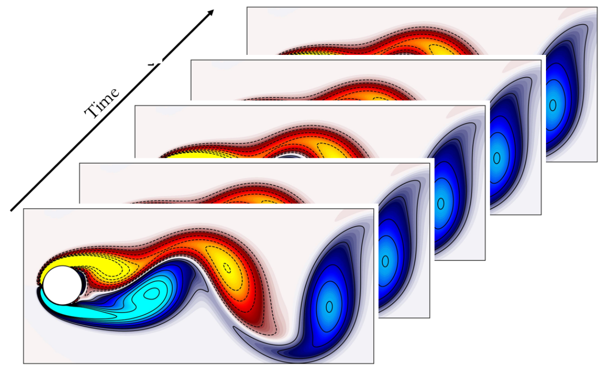
A Data-Driven Debias View

Machine Learning → $\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \rightarrow \hat{\theta} \in \Theta \rightarrow \Phi(\hat{\theta})$

Eigendecomposition of A



$$\frac{dx(t)}{dt} = Ax(t)$$



Snapshot Data

$\underbrace{\{(x_i, Ax_i)\}_{i=1}^n \qquad \theta = A}$

Project to a subspace

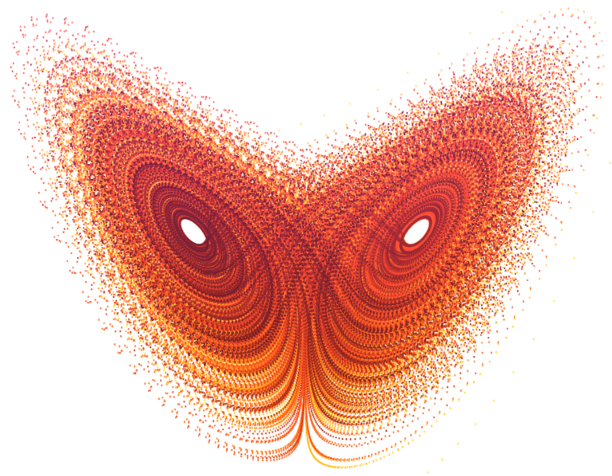
Dynamic Mode Decomposition/Randomized SVD

How can we the error $\Phi(\theta) - \Phi(\hat{\theta})$?

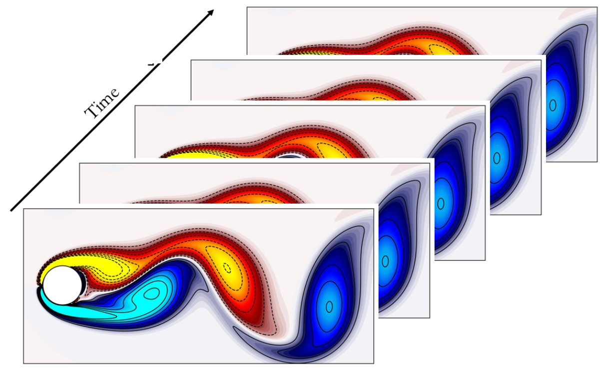
A Data-Driven Debias View

Eigendecomposition of A

Machine Learning → $\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \rightarrow \hat{\theta} \in \Theta \rightarrow \Phi(\hat{\theta})$



$$\frac{dx(t)}{dt} = Ax(t)$$



Snapshot Data

$\underbrace{\{(x_i, Ax_i)\}_{i=1}^n \qquad \theta = A}$

Project to a subspace

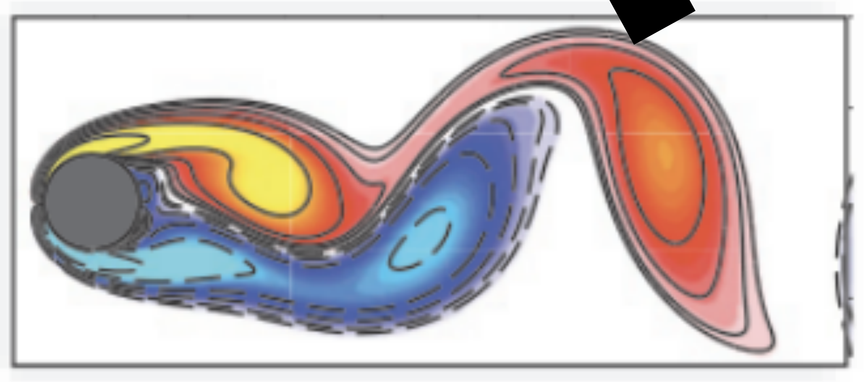
Dynamic Mode Decomposition/Randomized SVD

Debiasing using Taylor Expansion

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

Our Observation:

Taylor Expansion can be computed by snapshot data.

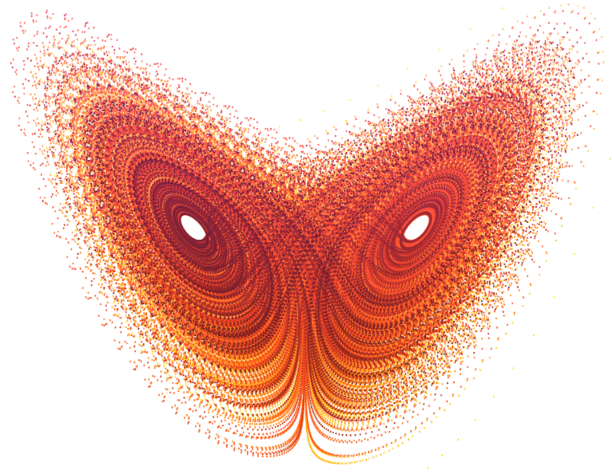


$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$$

A Data-Driven Debias View

Eigendecomposition of A

$$\underbrace{\{X_i\}_{i=1}^{n}, X_i \sim P_\theta}_{\text{data}} \rightarrow \hat{\theta} \in \Theta \rightarrow \Phi(\hat{\theta})$$



$$\frac{dx(t)}{dt} = Ax(t)$$

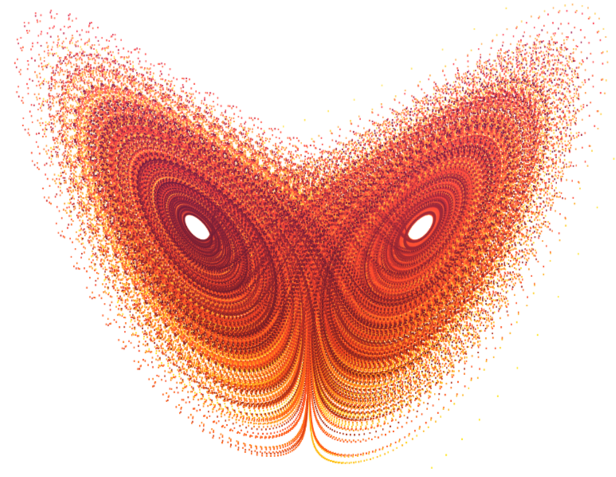
What is $\nabla \Phi(\hat{A})$?

$$\nabla \Phi(\hat{A}) = (\lambda I - \hat{A})^{\dagger}$$

A Data-Driven Debias View

Eigendecomposition of A

$$\underbrace{\{X_i\}_{i=1}^n, X_i \sim P_\theta}_{\text{data}} \rightarrow \hat{\theta} \in \Theta \rightarrow \Phi(\hat{\theta})$$



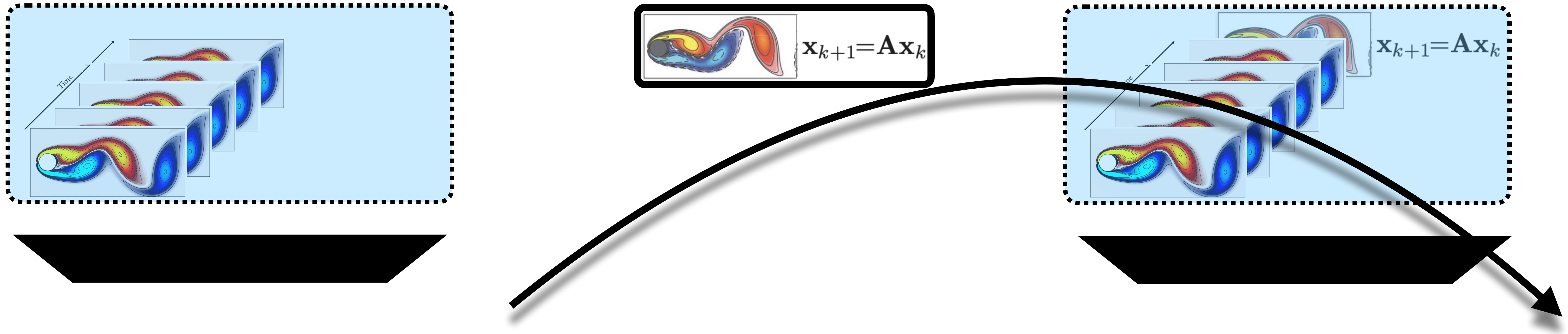
$$\frac{dx(t)}{dt} = Ax(t)$$

What is $\nabla \Phi(\hat{A})$?

$$\nabla \Phi(\hat{A}) = (\lambda I - \hat{A})^\dagger$$

Our AIM

update using online snapshot data



Embed dynamic to space Φ

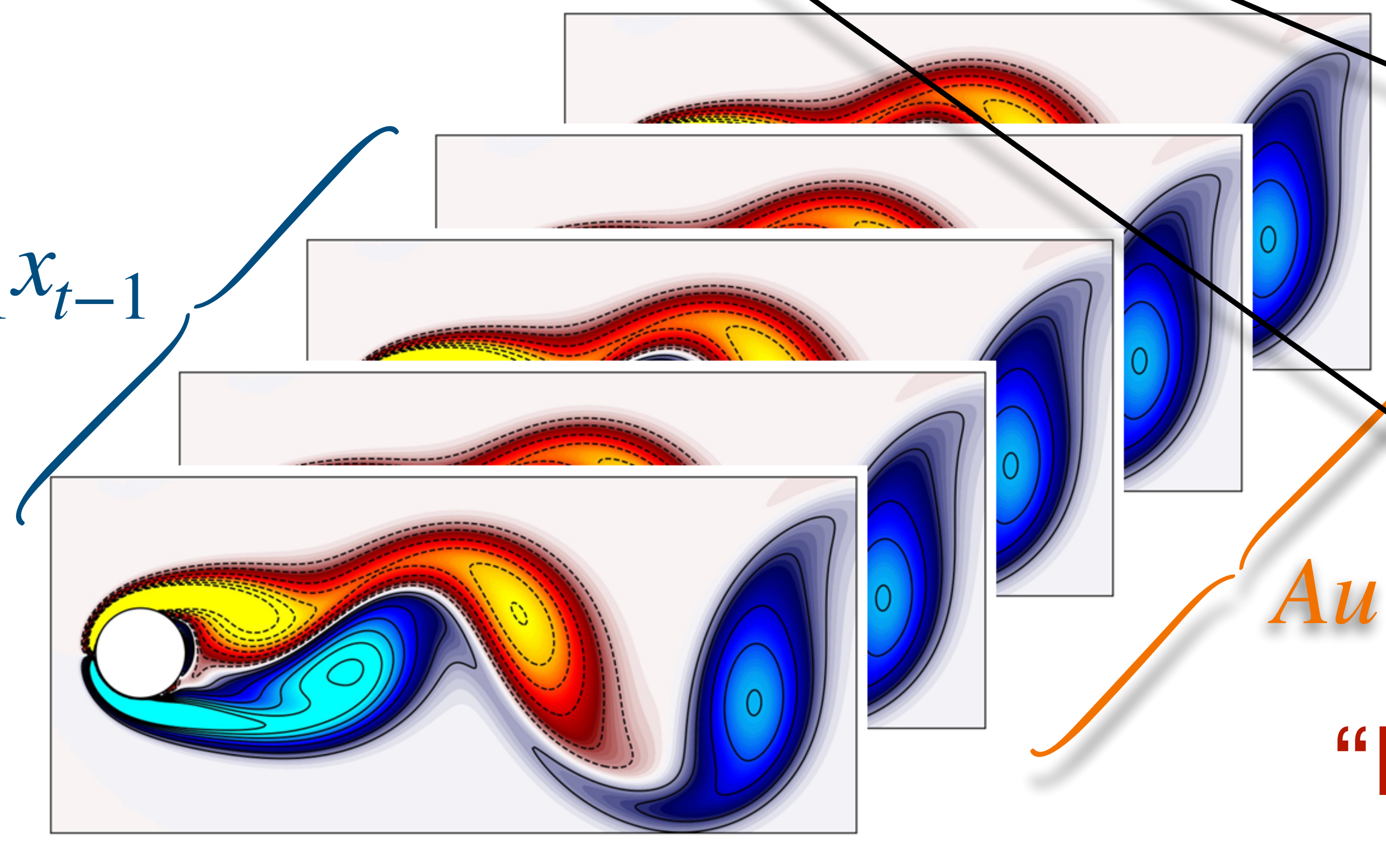
Embed dynamic to space $\Phi + \nabla \Phi d\Phi$

Computation of Taylor Expansion

$$\nabla \Phi(\hat{A})(A - \hat{A})u = (\lambda I - \hat{A})^{\dagger}(A - \hat{A})u$$

"Inverse Power Method"

$u = a_1 x_1 + \cdots + a_{t-1} x_{t-1}$



$Au = a_2 x_2 + \cdots + a_t x_t$

Prediction by DMD

"How much Error DMD have made"

Proposition The estimated mode at time t lies in span$\{x_1, \cdots, x_t\}$

Computation of Taylor Expansion

$$\nabla\Phi(\hat{A})(A-\hat{A})u = (\lambda I - \hat{A})^{\dagger}(A-\hat{A})u$$

"Inverse Power Method"

$$= \frac{1}{\lambda}(I - UU^{\dagger}) + U(\lambda I - \Lambda)^{\dagger}U^{\dagger}$$

Orthogonal to modes — Decrease λ times

Span of modes — "Inverse Power method"

when we know the Eigen decomposition $\hat{A} = U\Lambda U^{\dagger}$ (U: Mode, Λ: Eigen)

Enables computation using snapshot data!

Proposition The estimated mode at time t lies in span$\{x_1, \cdots, x_t\}$

Relationship with Inverse Power Methods

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

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

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

Relationship with Inverse Power Methods

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

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

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

How you construct such iteration?
What is the rule of $\hat{A}$?

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

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, Iteratative 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

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

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

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

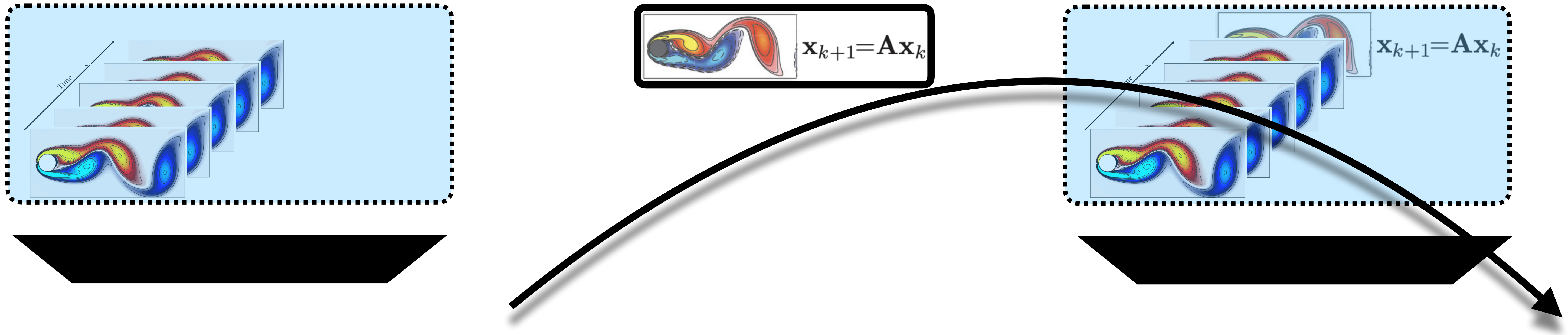
Thus Our convergence is linear-quadratic

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?

Online Dynamic Mode Decomposition

update using online snapshot data



Embed dynamic to space Φ

Computational cost:

$$O\left(nTk\log\left(\frac{\lambda_k}{\epsilon\lambda_{k+1}}\right)+nk^2+k^3\right)$$

SVD of Dynamic (By Krylov Iteration) — $nTk\log\left(\frac{\lambda_k}{\epsilon\lambda_{k+1}}\right)$

Reconstruct Mode — nk^2

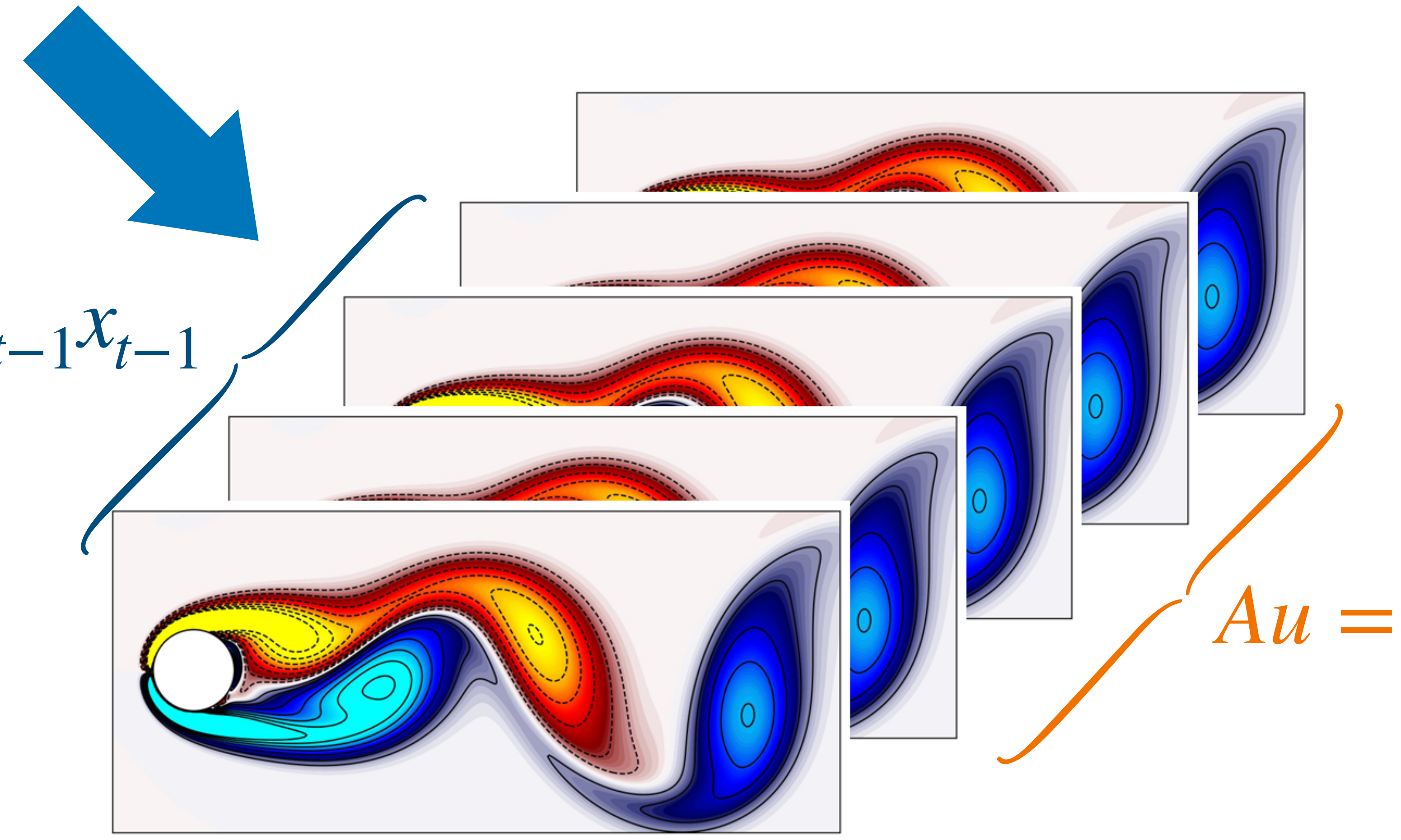
Eigen Decomposition — k^3

Embed dynamic to space $\Phi + \nabla\Phi d\Phi$

Computational cost:

$$O(nTk + nk^2 + k^3)$$

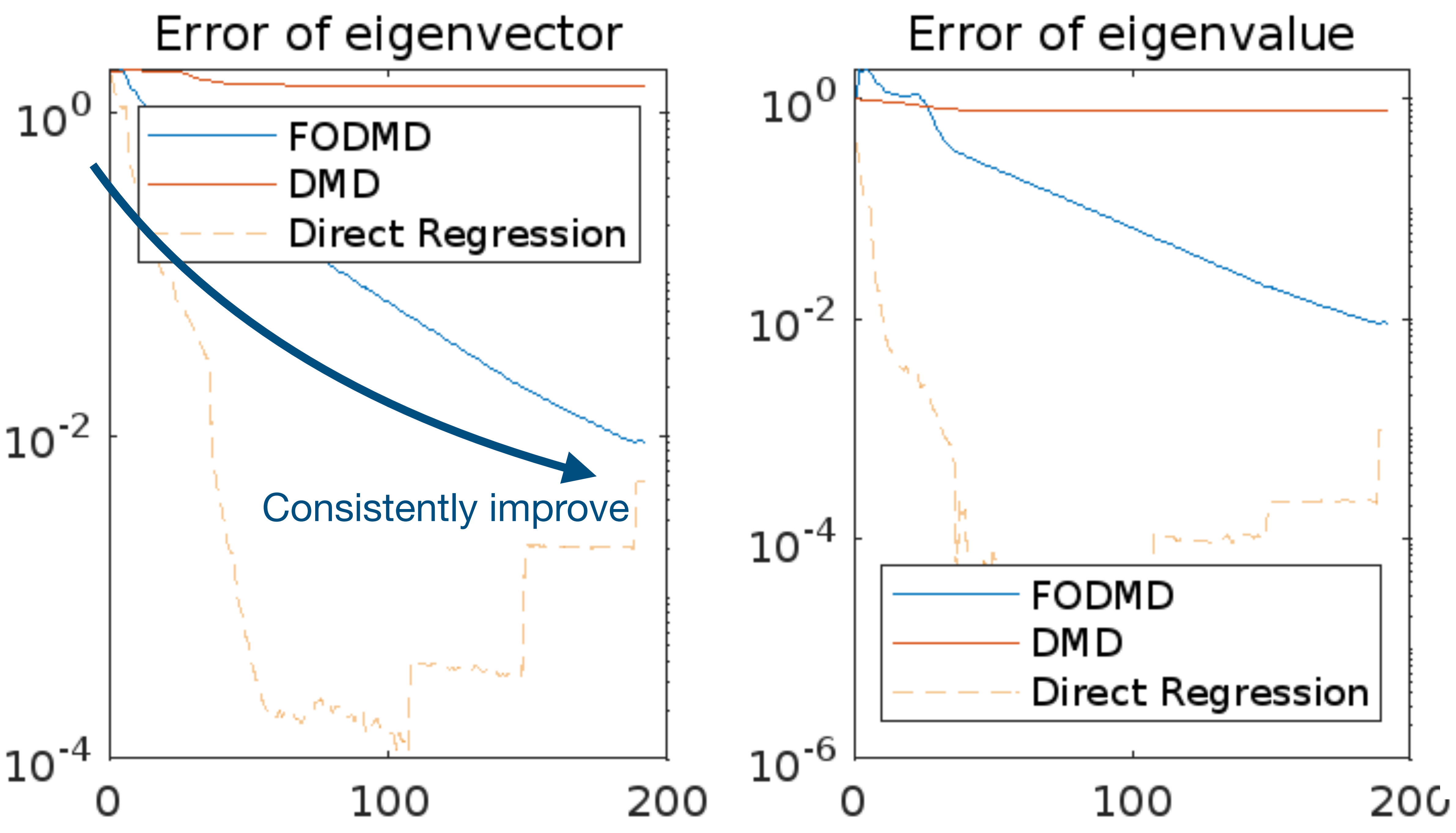
$u = a_1x_1 + \cdots + a_{t-1}x_{t-1}$

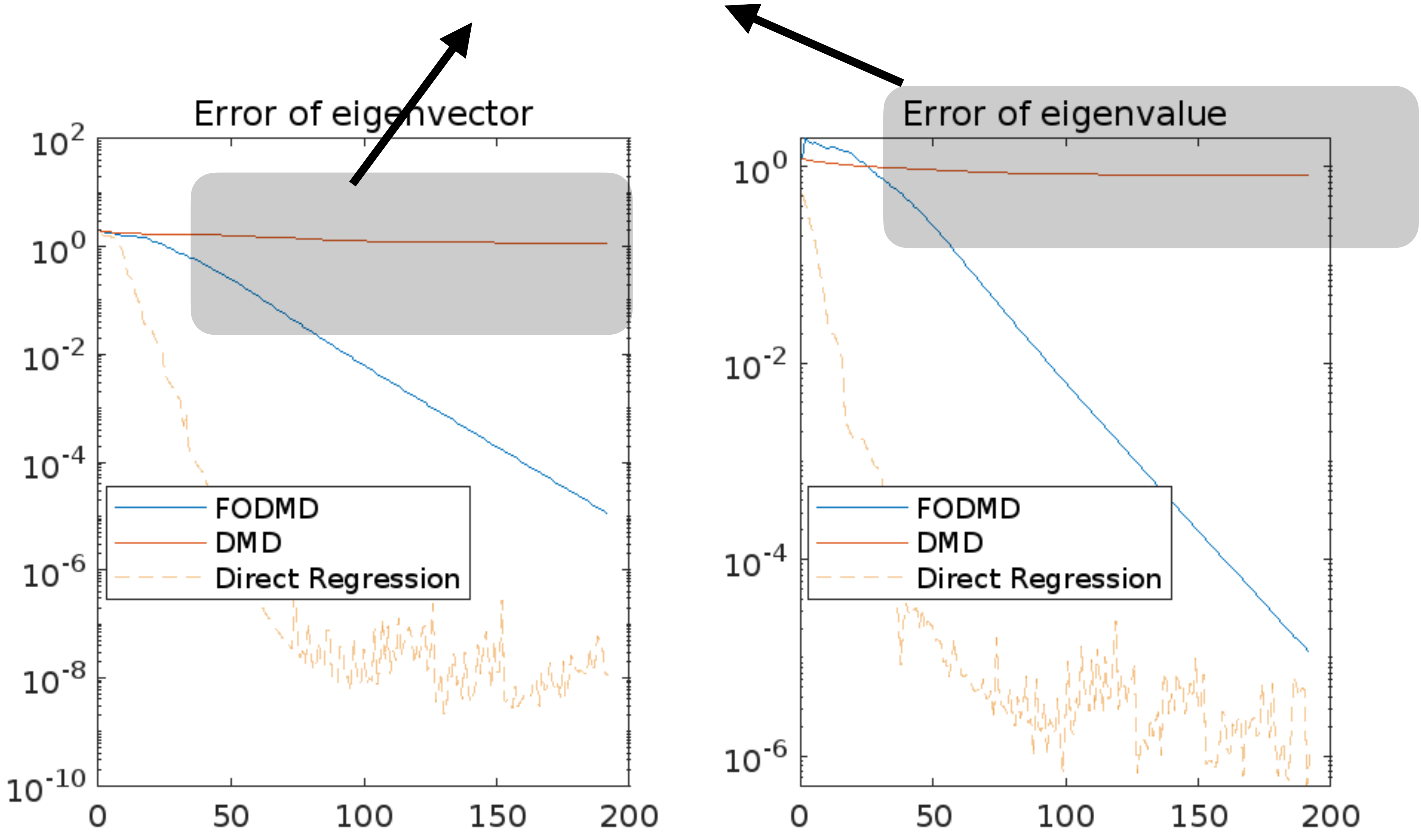


$Au = a_2x_2 + \cdots + a_tx$

Experimental Results

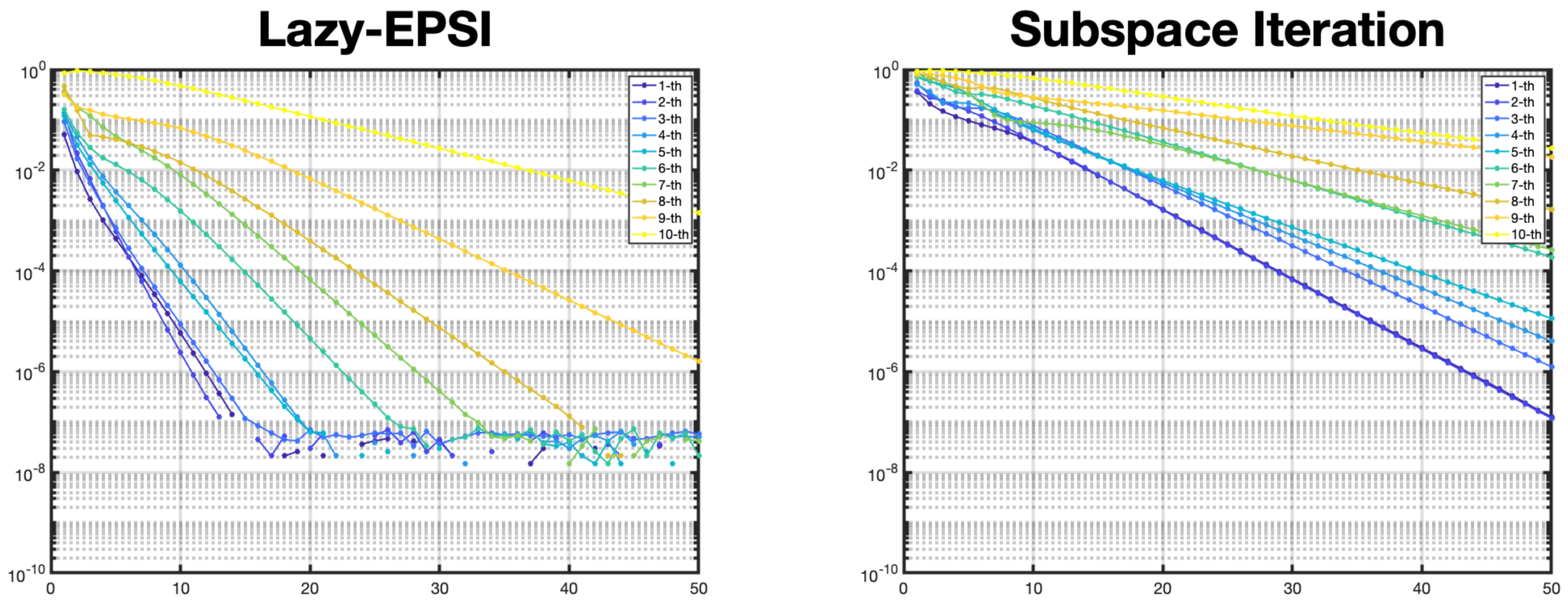
Error dominated by truncation of Krylov subspace
The help of more data is limited



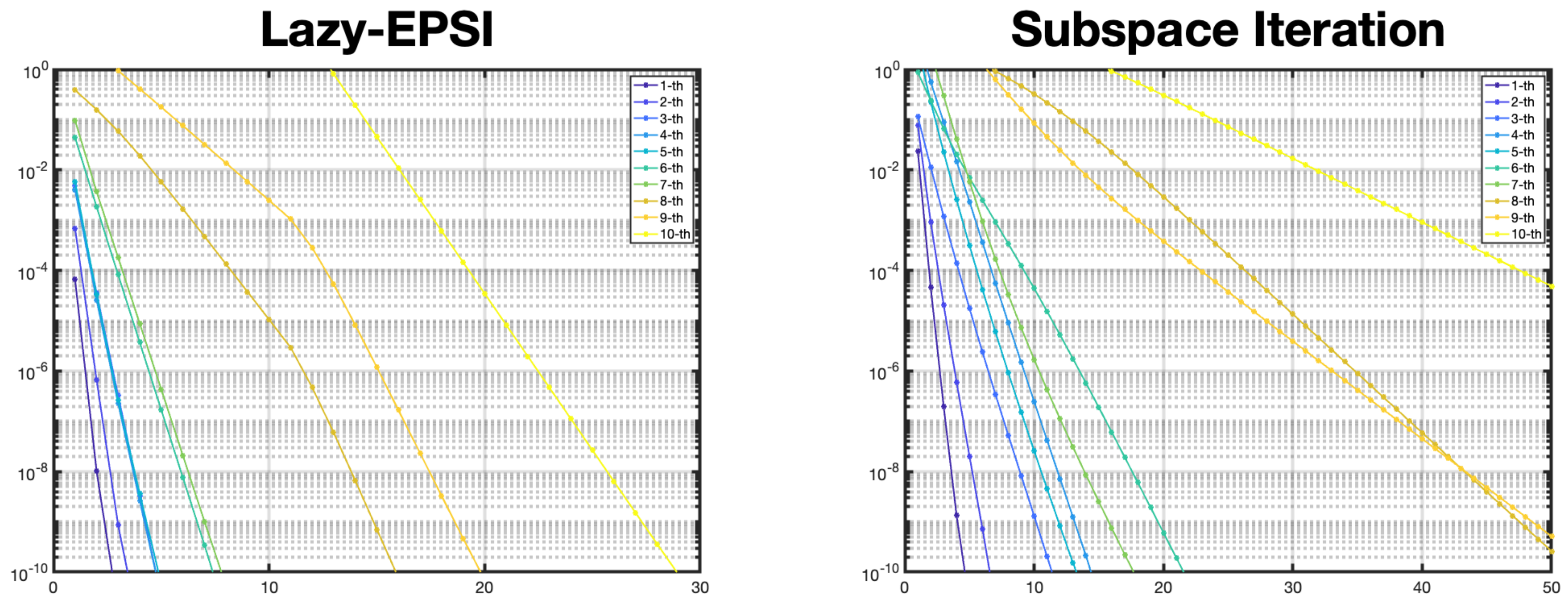


A has four eigenvalue 1 and then decreasing to 0.1

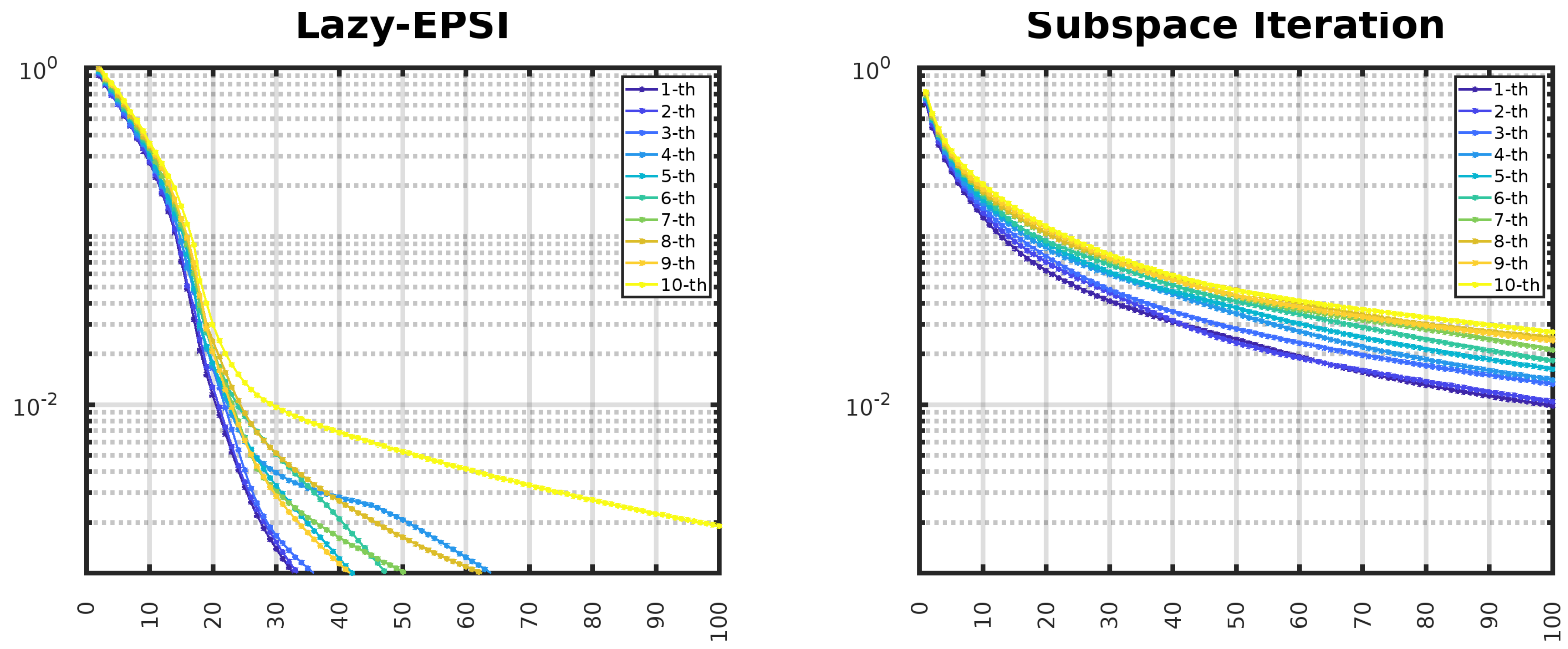
Eigenvalue Computation



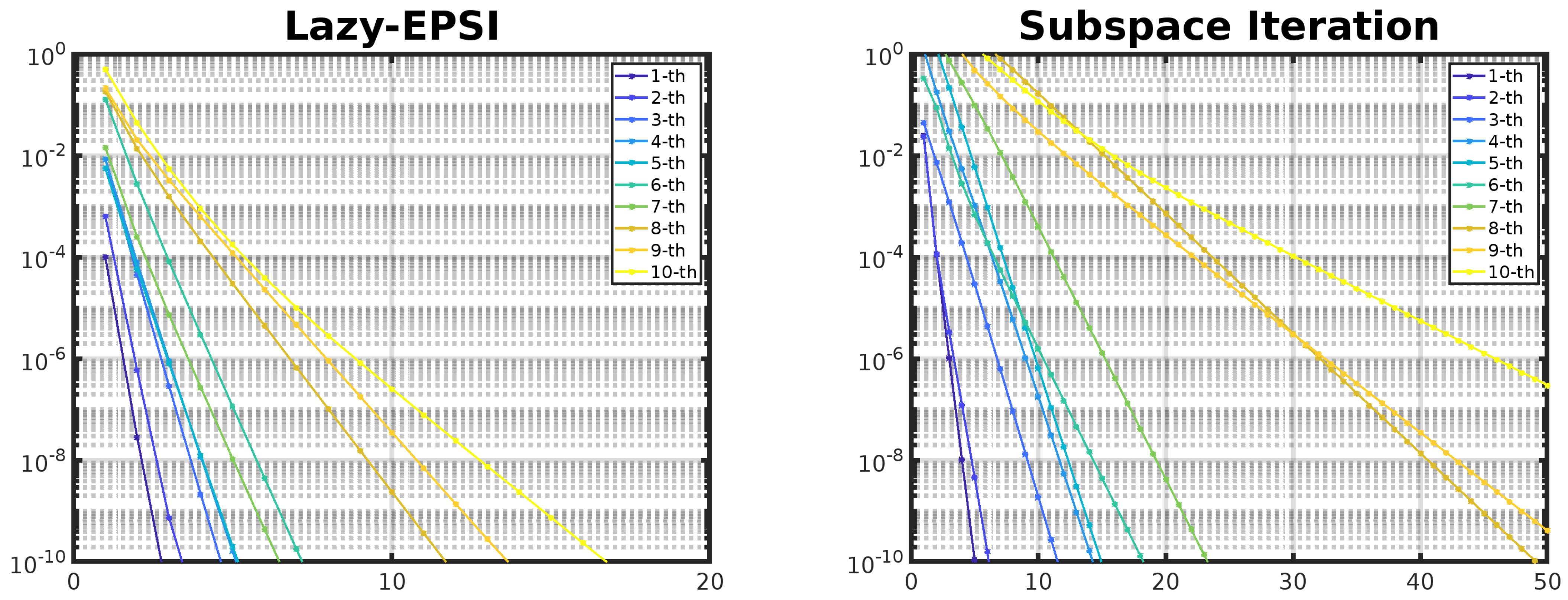
Random ill-conditioned matrix



Amazon (SNAP)



PDE (Laplacian)



Web Stanford (SNAP)

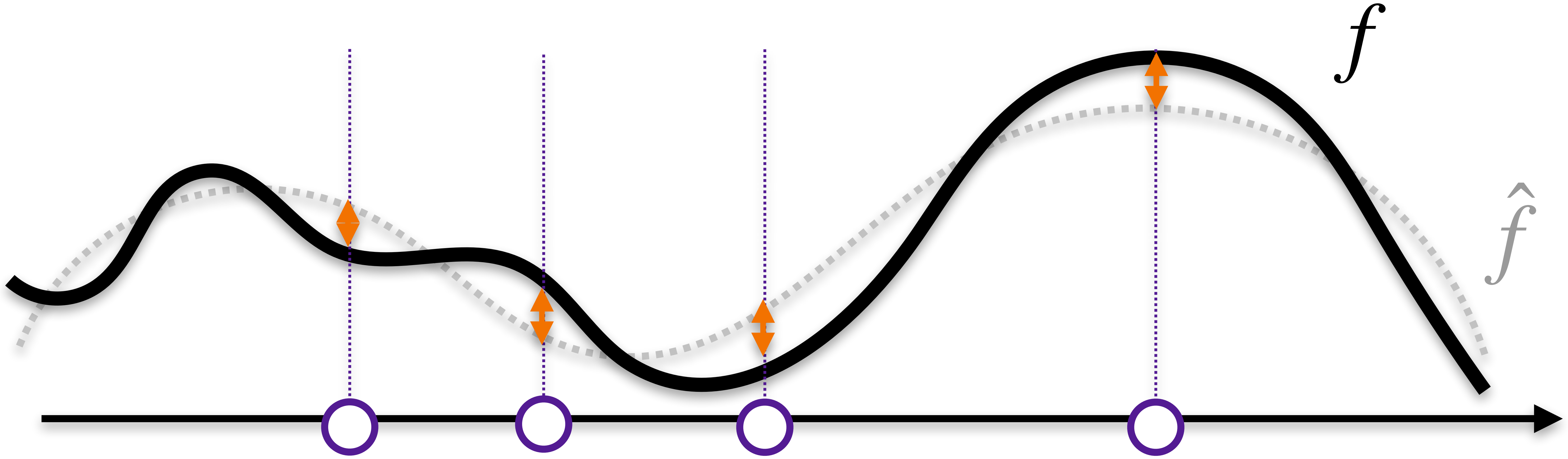
What is SCaSML about?

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

Examples Later!