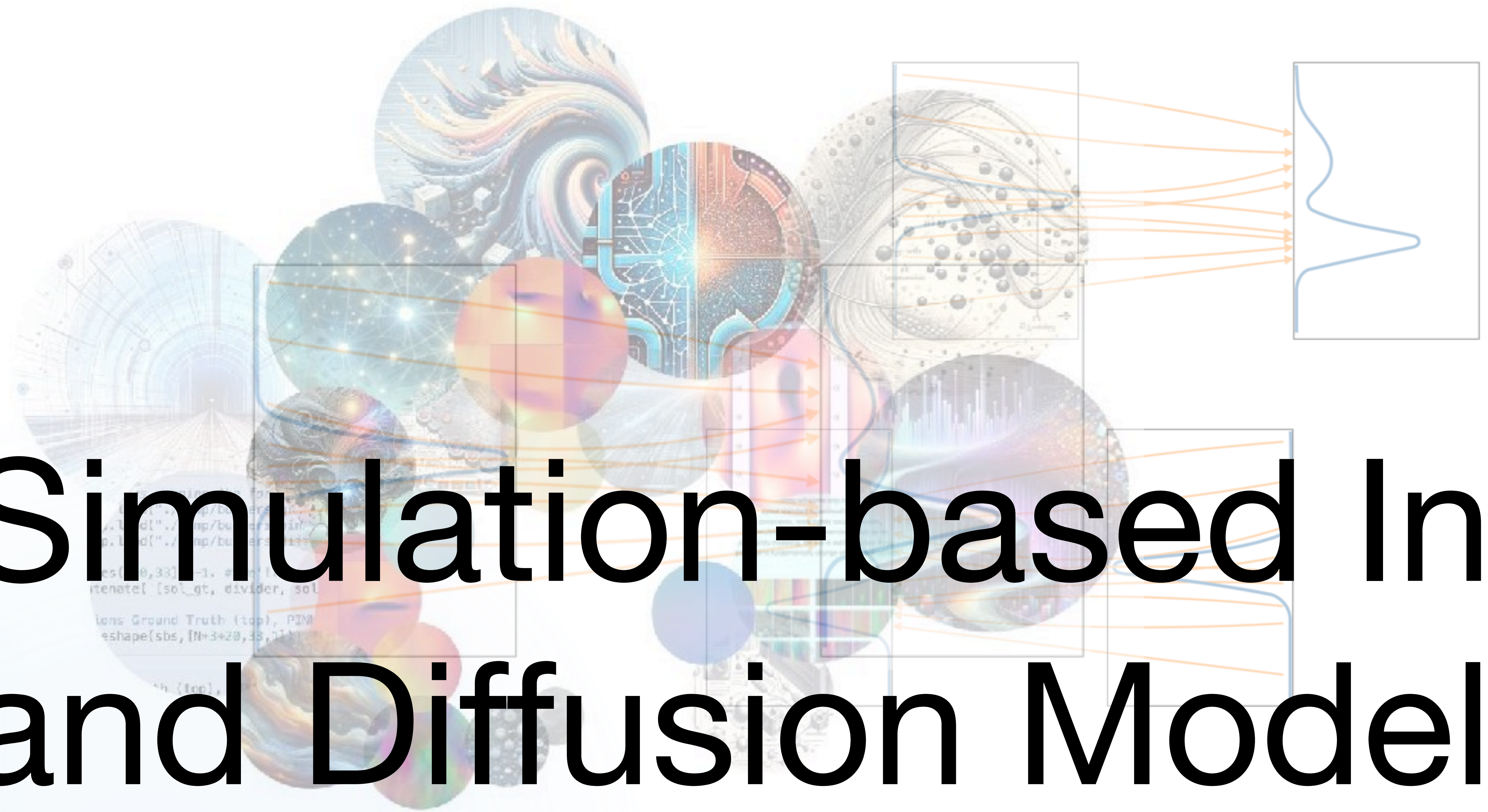


Simulation-based Inference and Diffusion Models

PROBABILISTIC INVERSE PROBLEMS, PART 2/2

Overview

- Normalizing Flows
- Score Matching
- Langevin Dynamics
- Denoising Diffusion
- Flow Matching
- Physics Constraints

Flow Matching & Rectified Flows

Flow Matching

- Revisit score matching: integrate flow (ODE) to transform probability densities
- But: Compute reference trajectories via denoising framework (instead of score)
- Main formulation: transform x from p_0 into a sample from p_1 by training integrating a simple ODE: $dx/dt = v_\theta(x, t)$
- Train v_θ with $\mathcal{L}_{\text{CFM}}(\phi) = \mathbb{E}_{q(z,t),p_t(x|z)} \mid v_\phi(x, t) - u_t(x \mid z) \mid^2$ with conditional likelihoods over the helper latent variable z

Mappings

- Which mapping from p_0 to p_1 to aim for?
- Straight, linear paths are ideal: correspond to optimal transport
- Slight complication: minimal noise amount σ_{min} needed to ensure a continuous distribution (instead of discrete samples)
- Generating velocity given by:

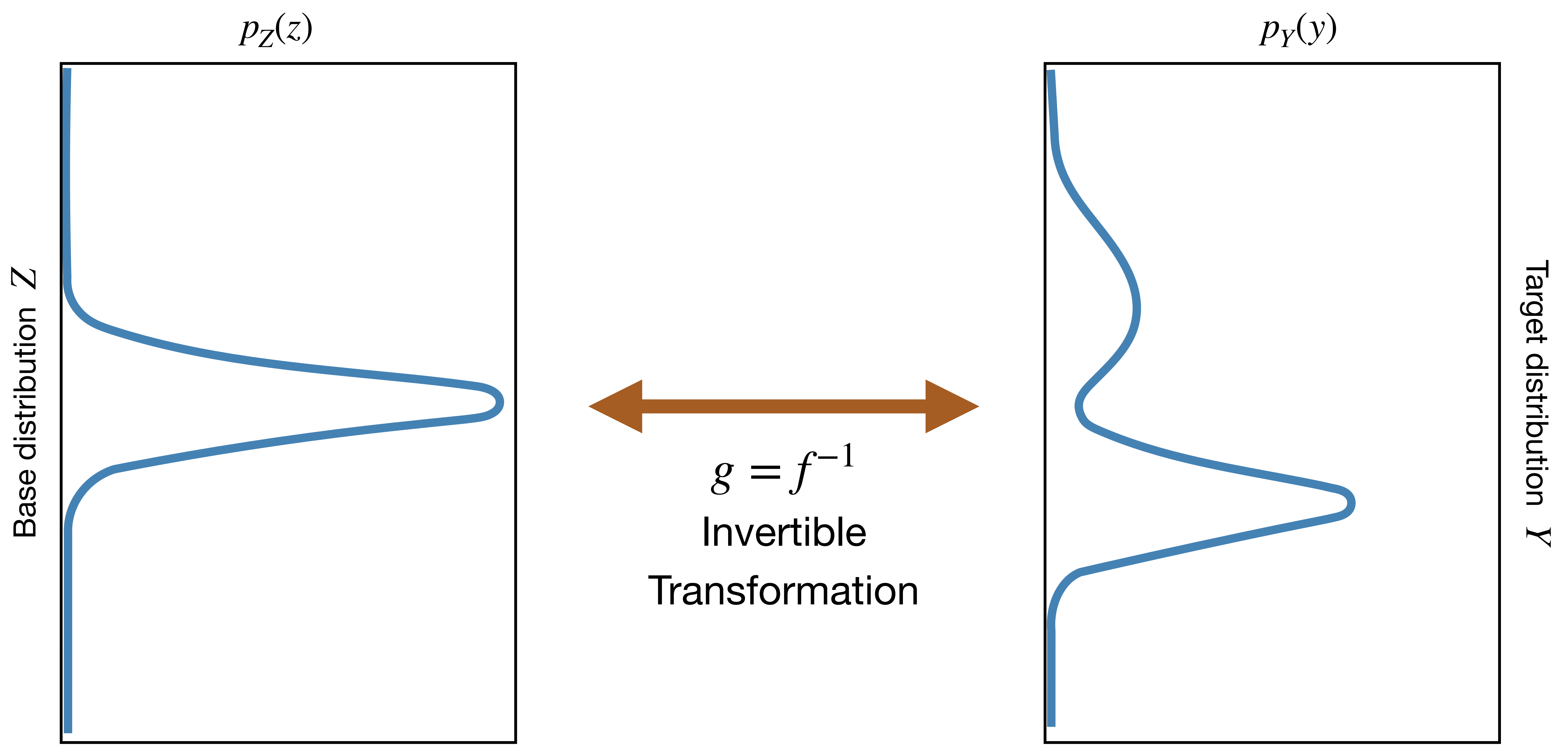
$$p_t(x|x_1) = \mathcal{N}\left(tx_1, (1-(1-\sigma_{\min})t)I\right)$$

$$u_t(x|x_1) = \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}$$

Note: without σ_{min} is simply: $p_t(x|x_1) = \mathcal{N}(tx_1, (1-t)I), \quad u_t(x|x_1) = (x_1 - x)/(1-t)$

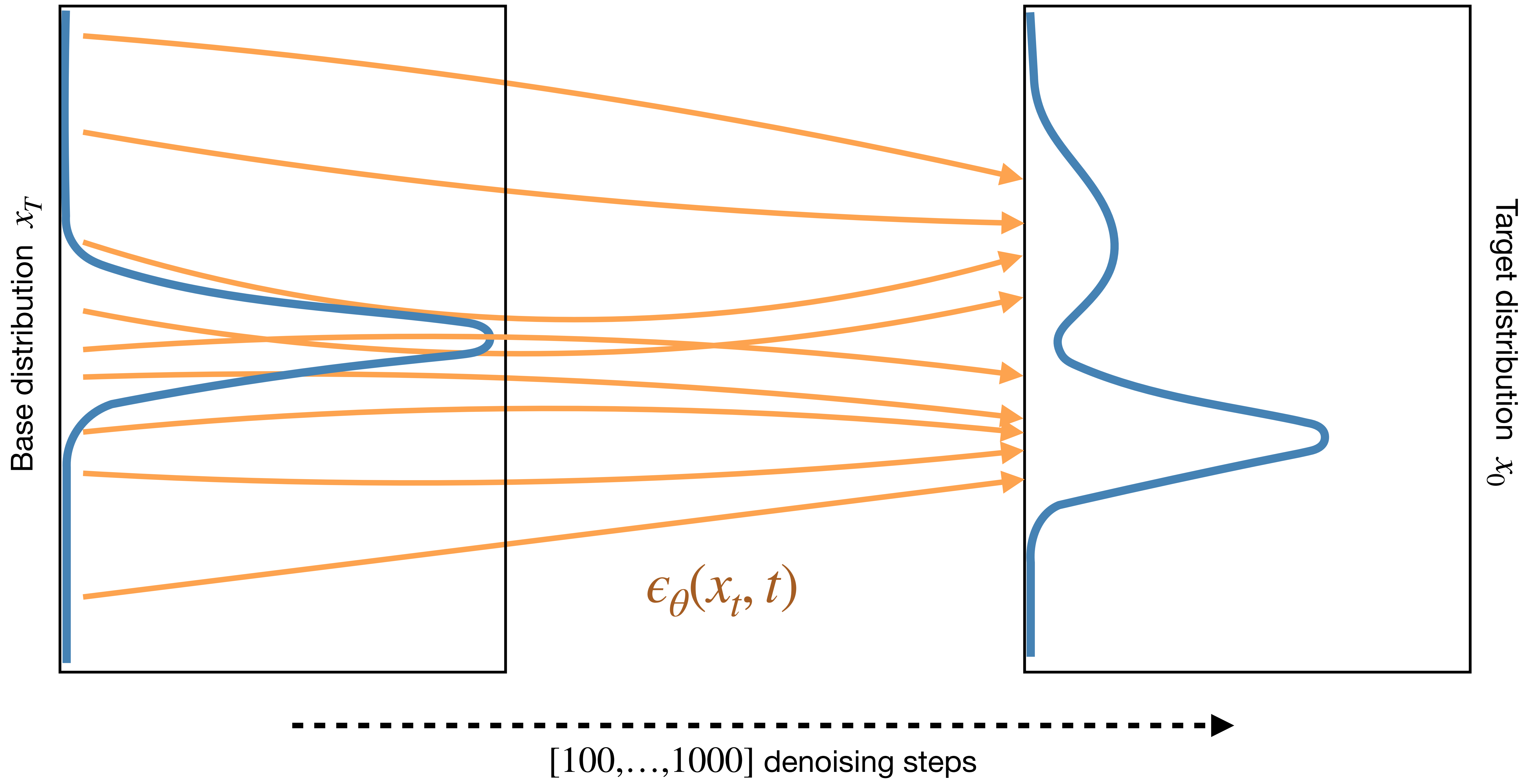
Denoising Diffusion Probabilistic Model

Previously: Normalizing Flows



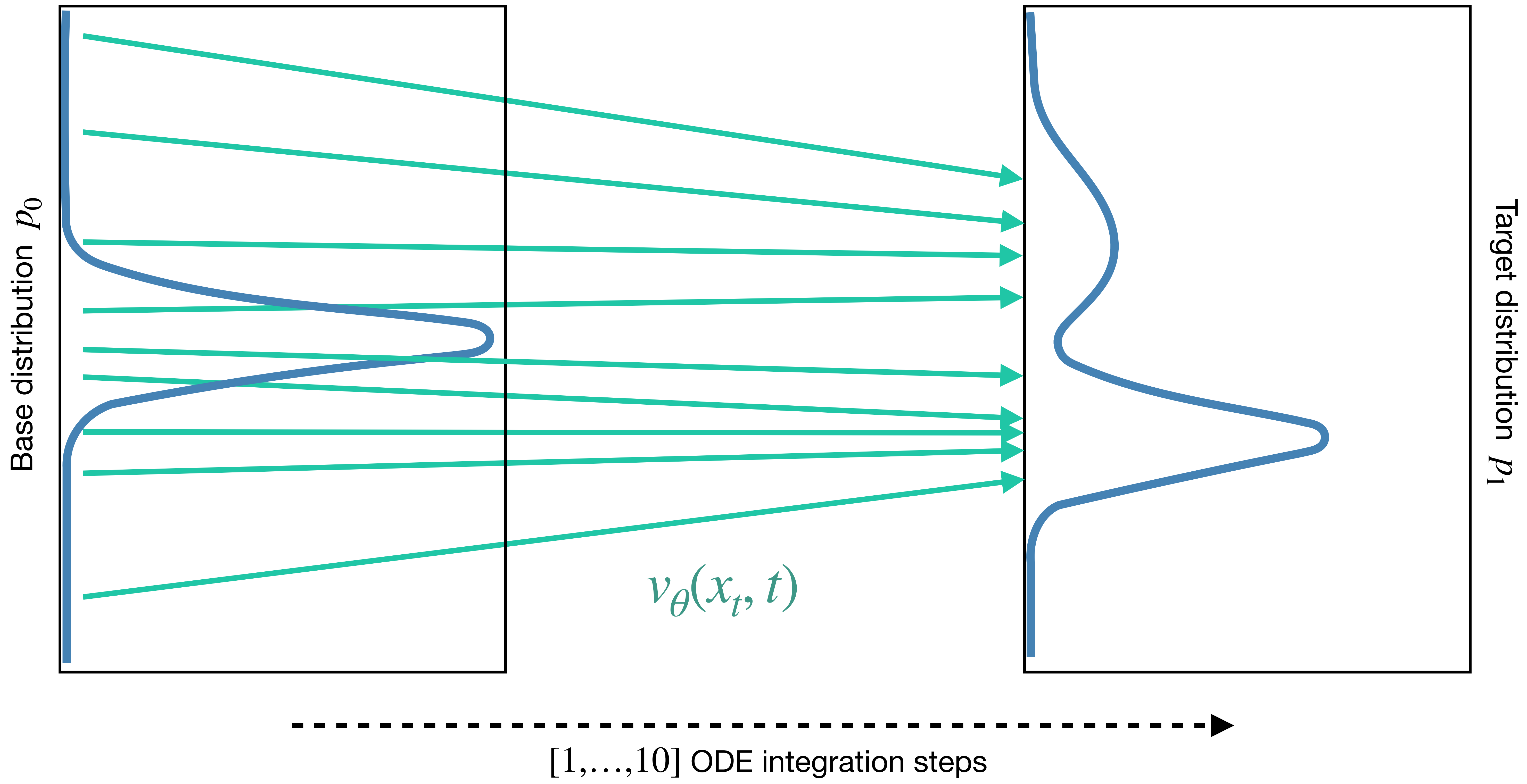
Flow Matching

Previously: DDPM



Flow Matching

Rectified Flows / Flow Matching



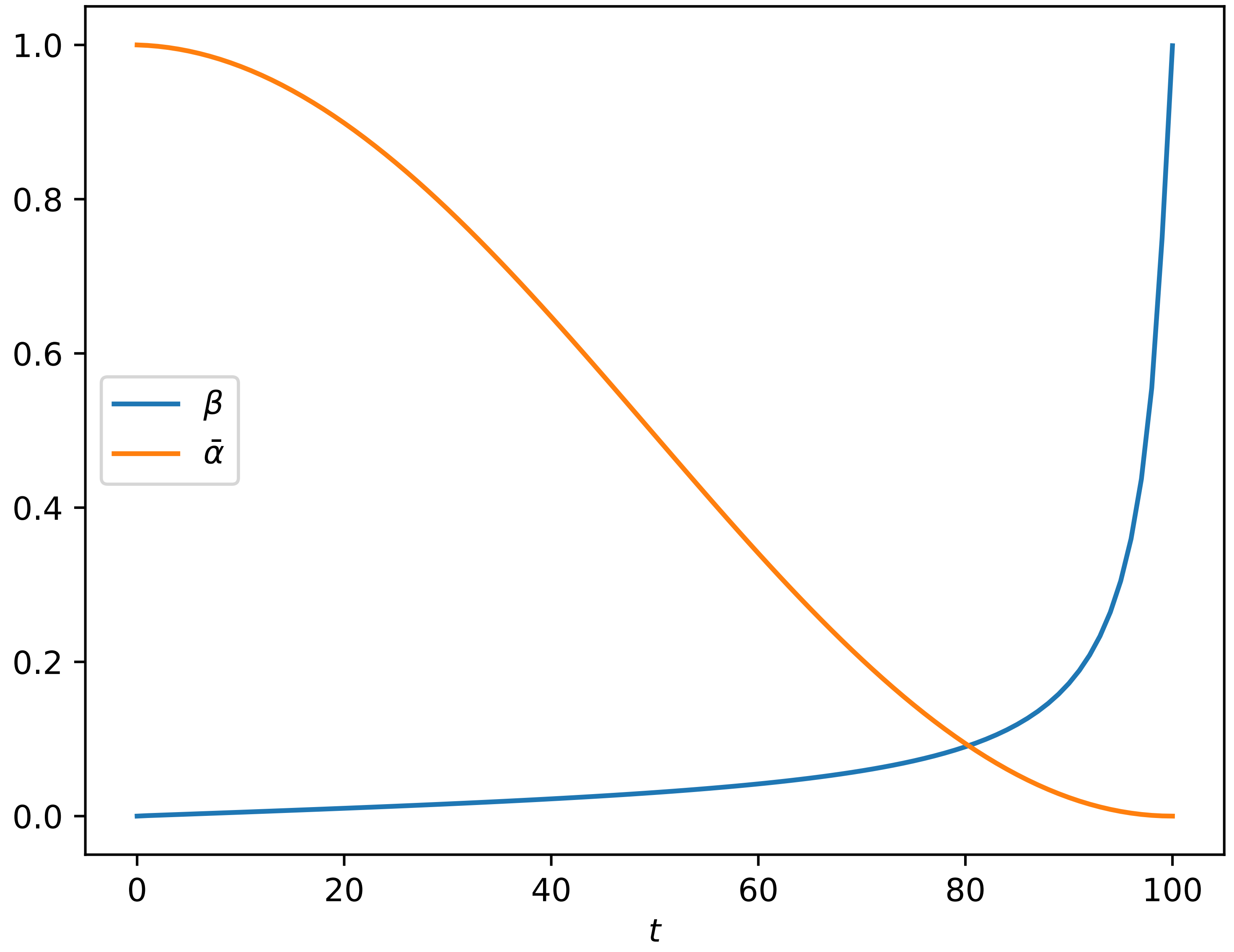
Ideally, converges in a single step!

Flow Matching why is it better?

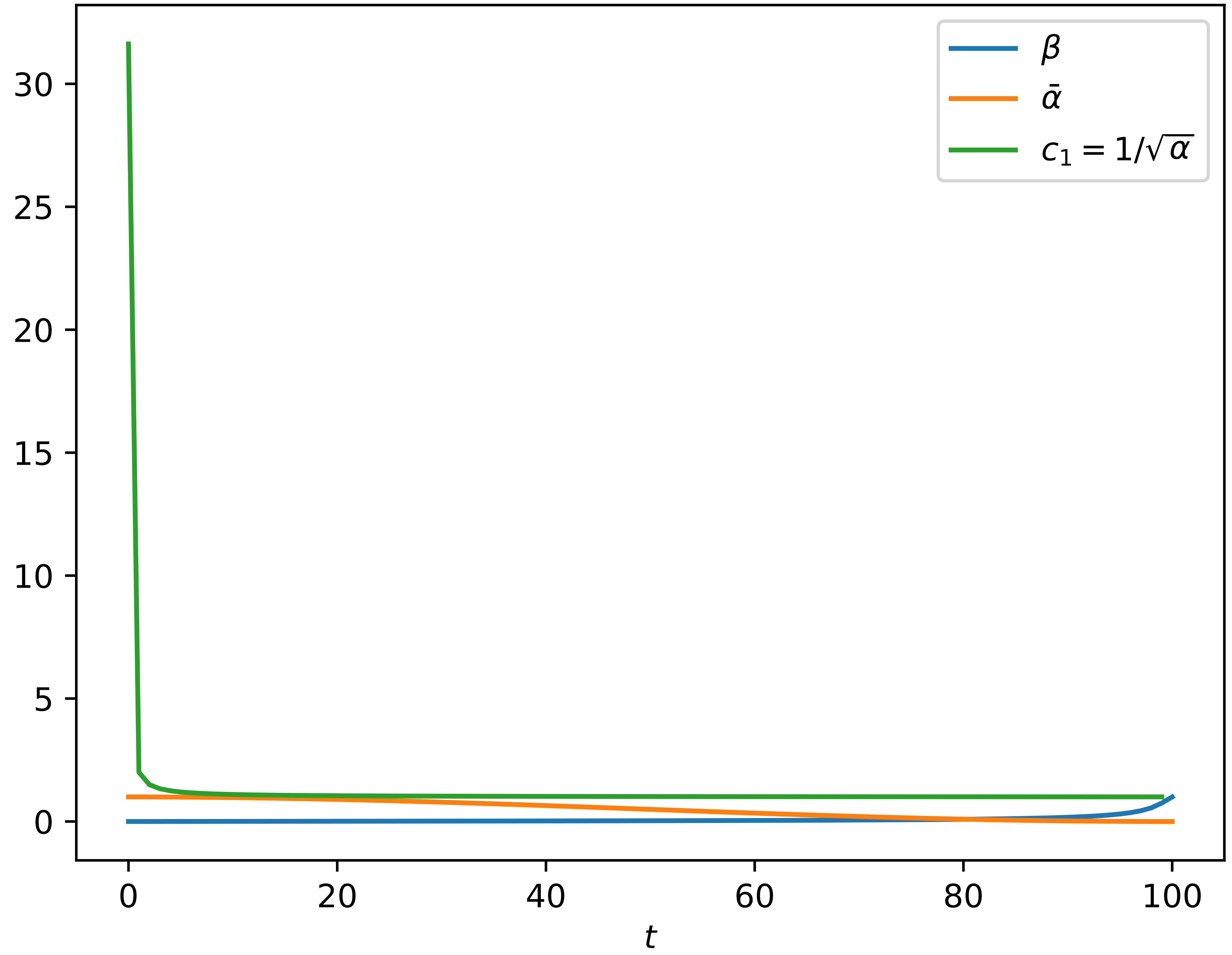
Reminder: Sampling with DDPM

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \boxed{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)} + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$



$\bar{\alpha}, \beta$ are well behaved



$1/\sqrt{\alpha}$ **is not!**

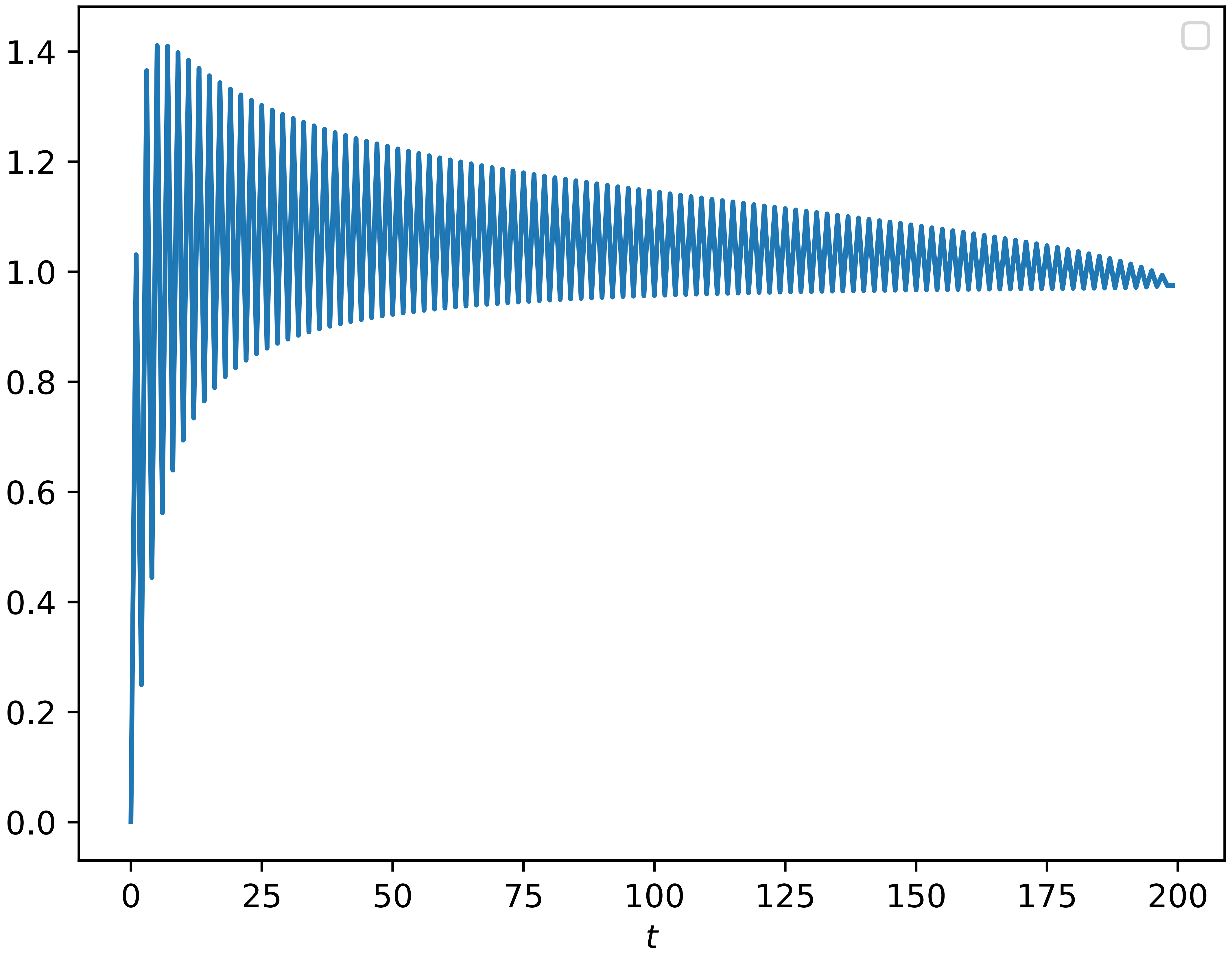
Flow Matching why is it better?

Reminder: Sampling with DDPM

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
#1
4: $\quad \mathbf{x}_{t-1} = \boxed{\frac{1}{\sqrt{\alpha_t}}\left(\boxed{\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}\right) + \sigma_t \mathbf{z}}$
#2
5: **end for**
6: **return** $\mathbf{x}_0$

- DDPM: Great training objective, but inference relies on high accuracy of NN
- Zig-zagging behavior (shown right)
- Flow matching: more stable, strictly linear!



Progression of x_t over time (values #1, #2 per step shown)

Flow Matching

- Very simple implementation:

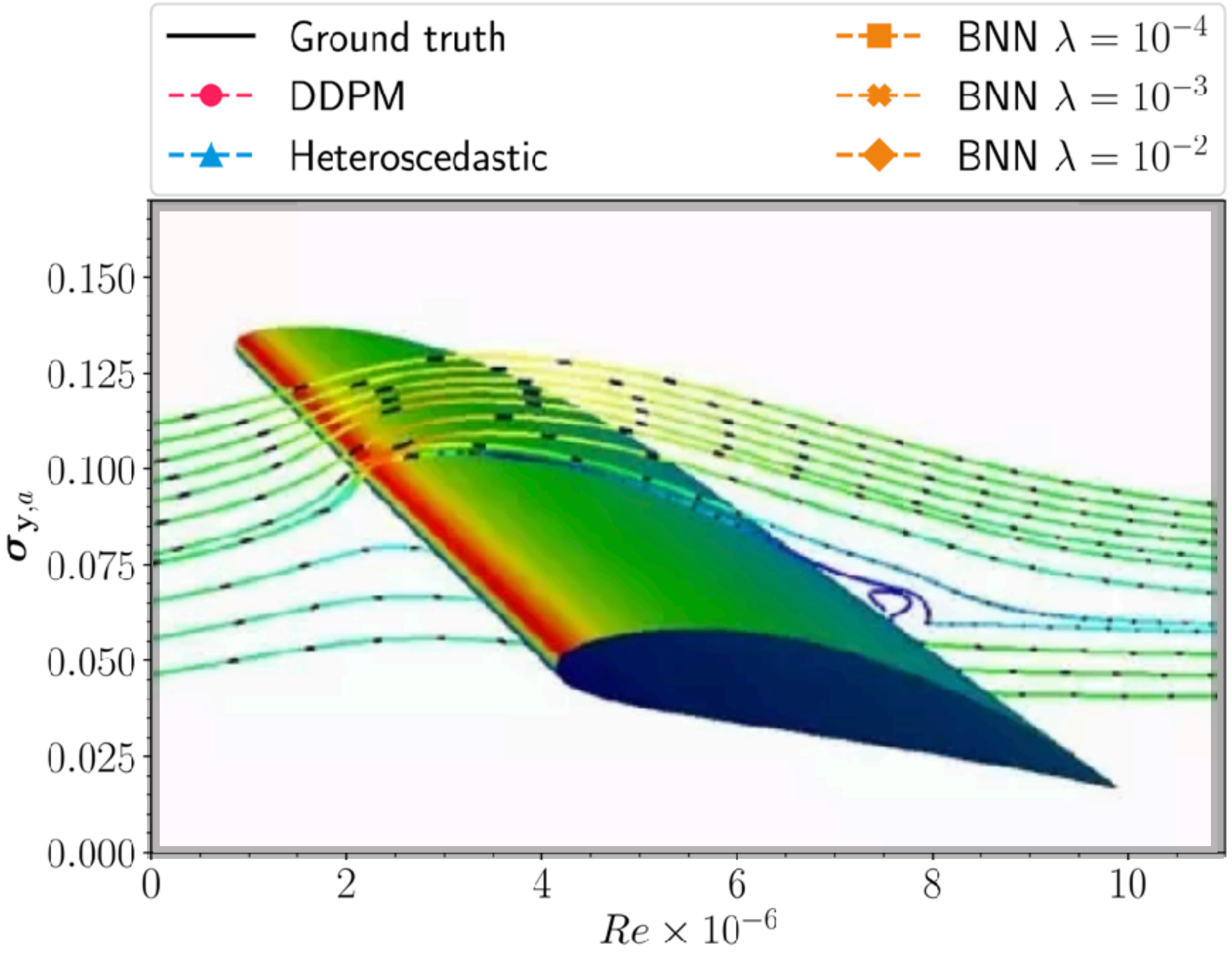
```
t = np.random.rand()
x_t = (1 - ( 1 - self.sigma_min) * t) * x0 + t * x1
u_t = (x1 - x0)
```

- At inference time, simply integrate: `x += dt * model(x,t)`
- Fancier (higher-order) integration also possible…
- Resulting method: fast, and gives full access to posterior distribution!
- Try yourself: **https:// ... /probmodels-flowmatching.ipynb**

Flow Matching & Co. in Action

Known Ground Truth Distribution

Turbulent NS case with varying Reynolds number:



Parameter varied in dataset

Liu et. al: Uncertainty Prediction of Airfoil Flow Simulations with DDPM

Known Ground Truth Distribution

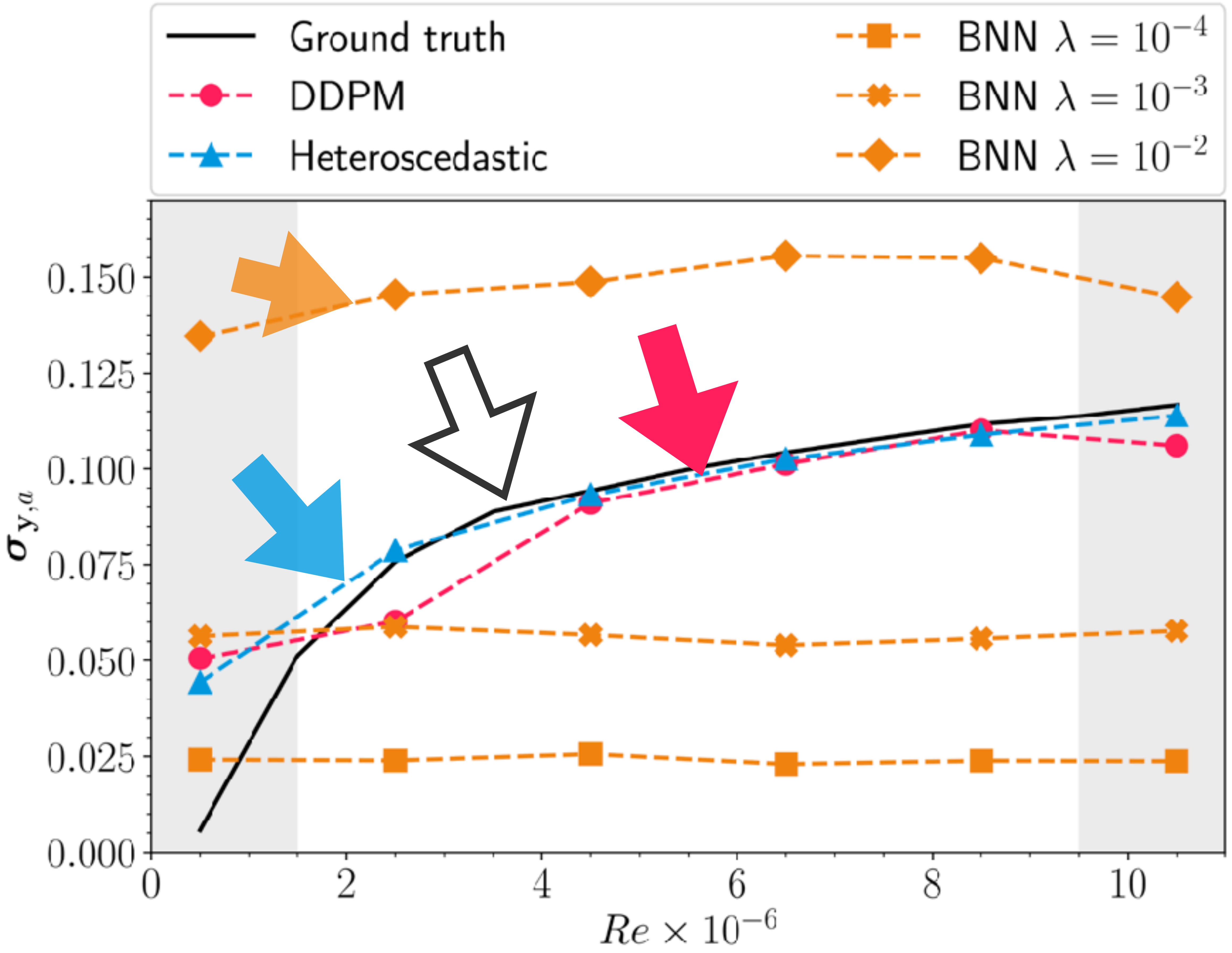
Turbulent NS case with varying Reynolds number:

Ground Truth

Bayesian NN

Heteroscedastic

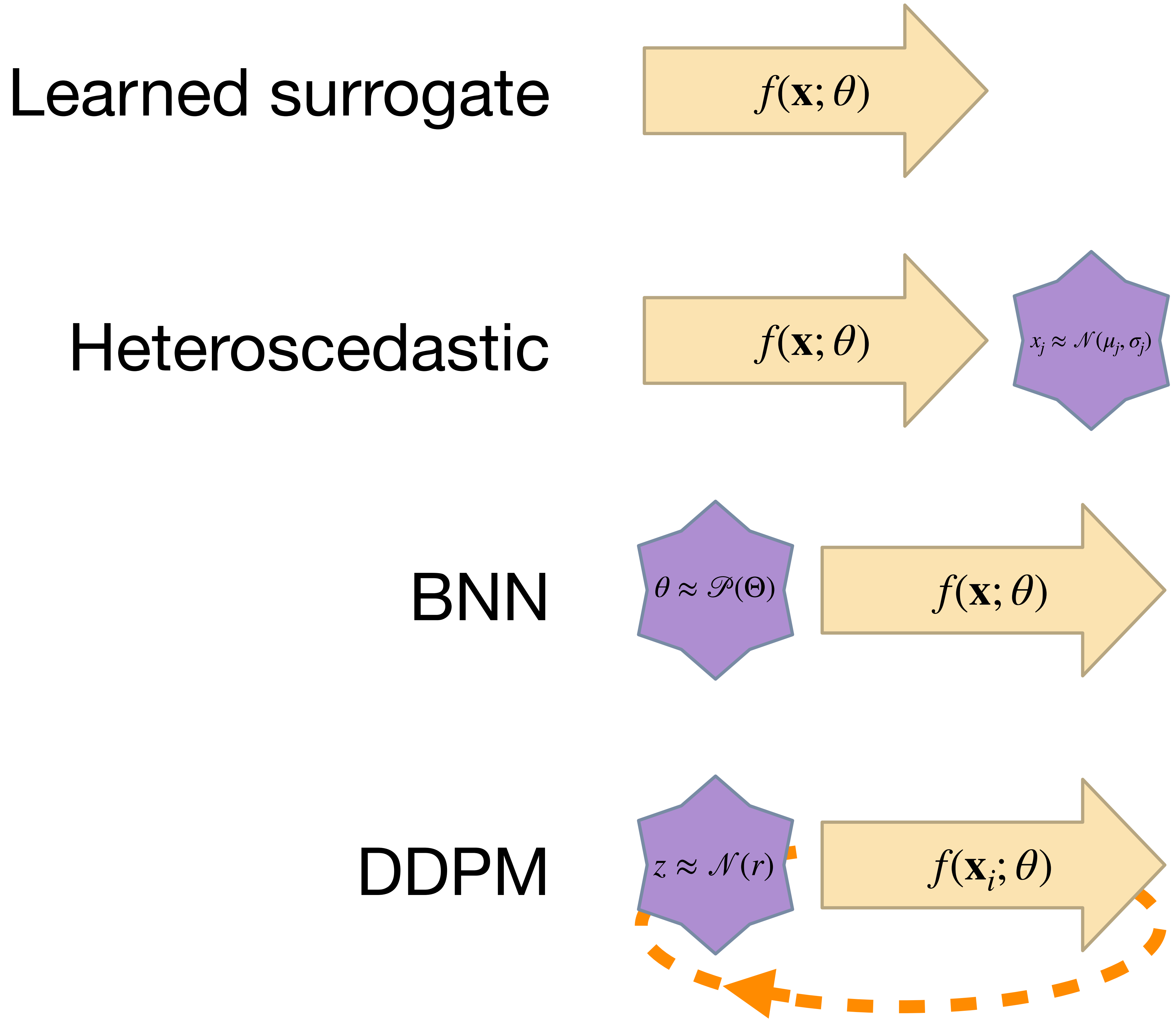
Diffusion Model



Parameter varied in dataset

Liu et. al: Uncertainty Prediction of Airfoil Flow Simulations with DDPM

Known Ground Truth Distribution

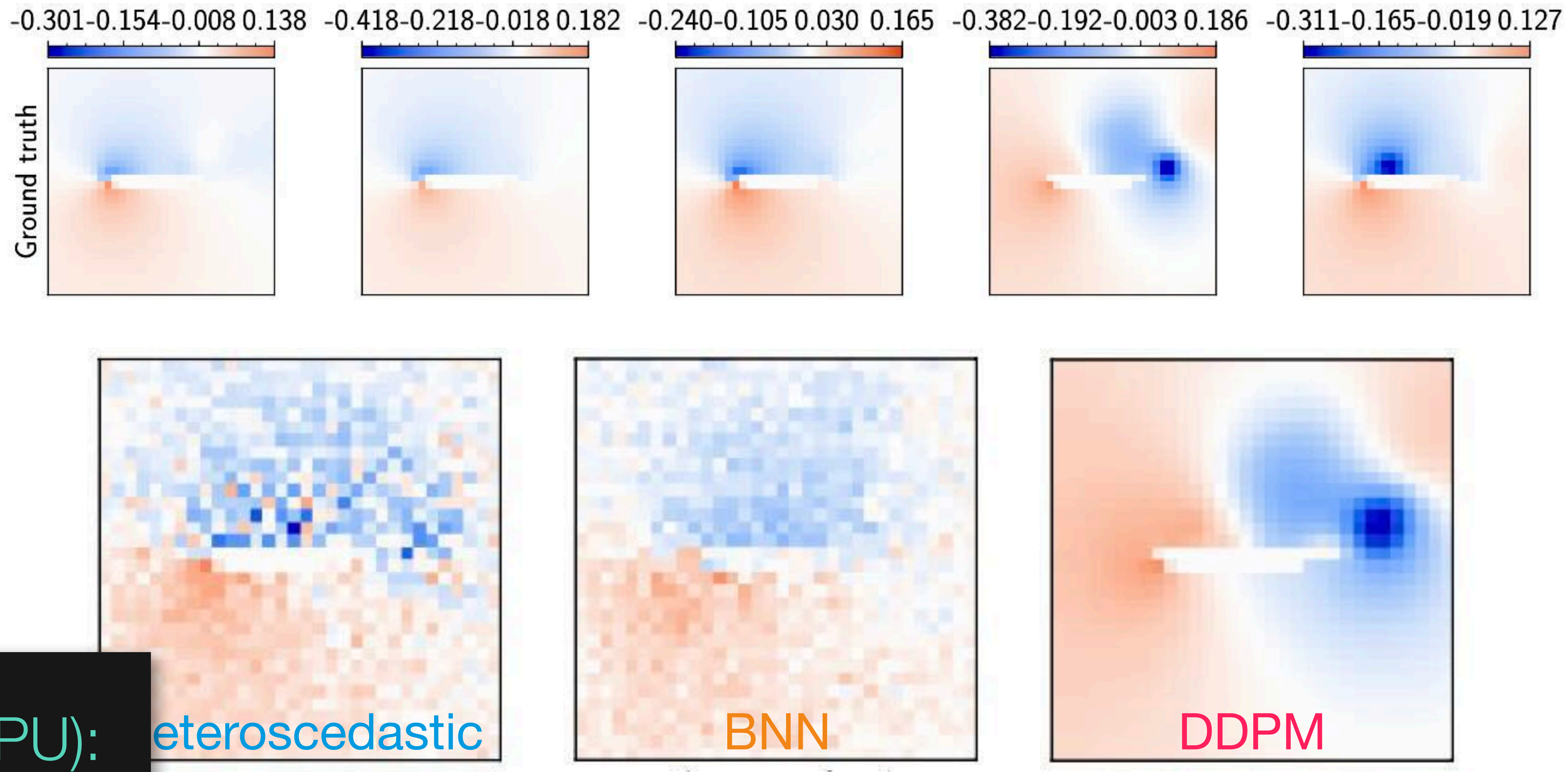


Liu et. al: Uncertainty Prediction of Airfoil Flow Simulations with DDPM

Known Ground Truth Distribution

Turbulent NS case with varying Reynolds number:

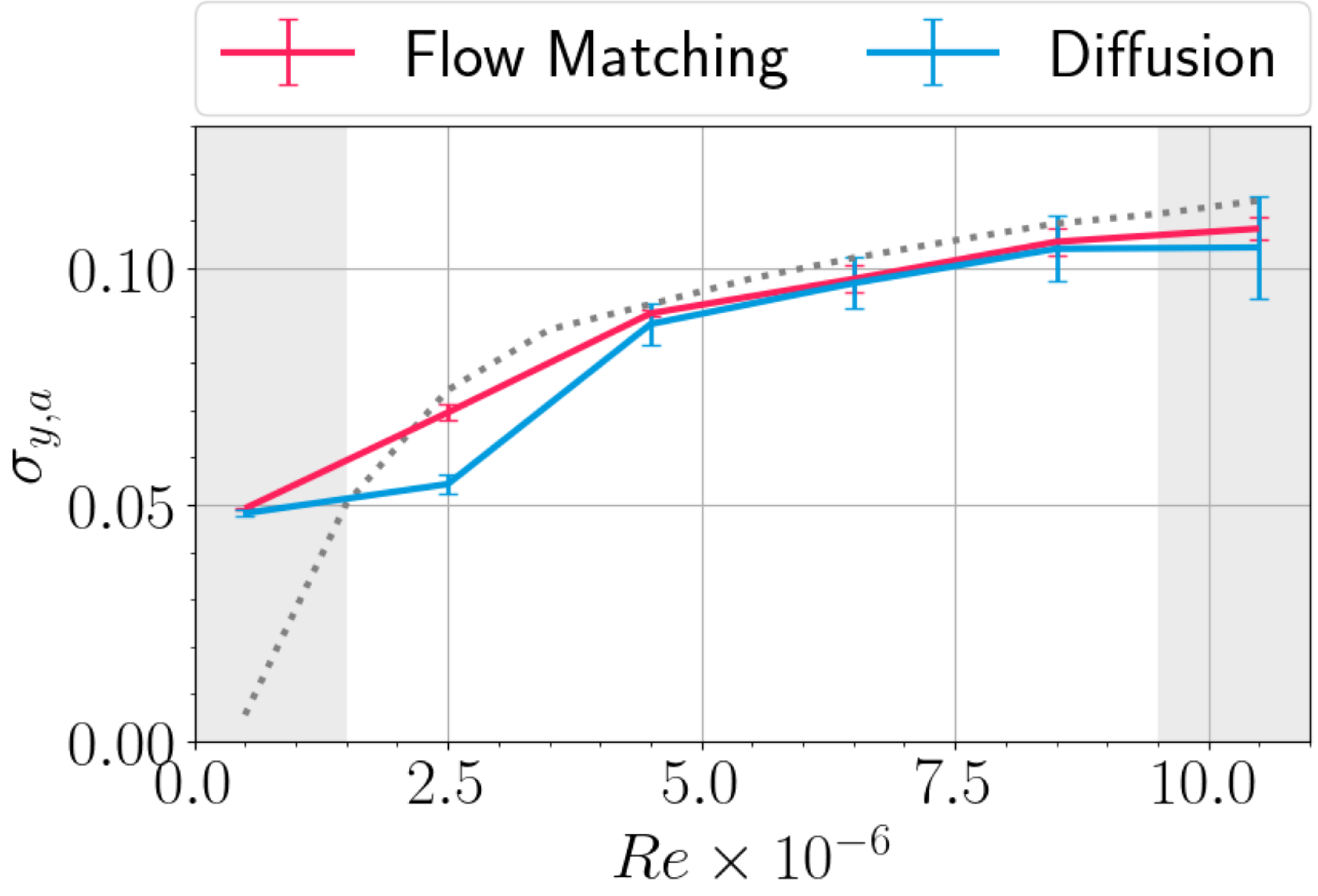
- *Heteroscedastic*
- *Bayesian NNs*
- *DDPM*



Fair comparison with solver (CPU):
DDPM ca. **9.5x** faster

Liu et. al: Uncertainty Prediction of Airfoil Flow Simulations with DDPM

DDPM vs Flow Matching

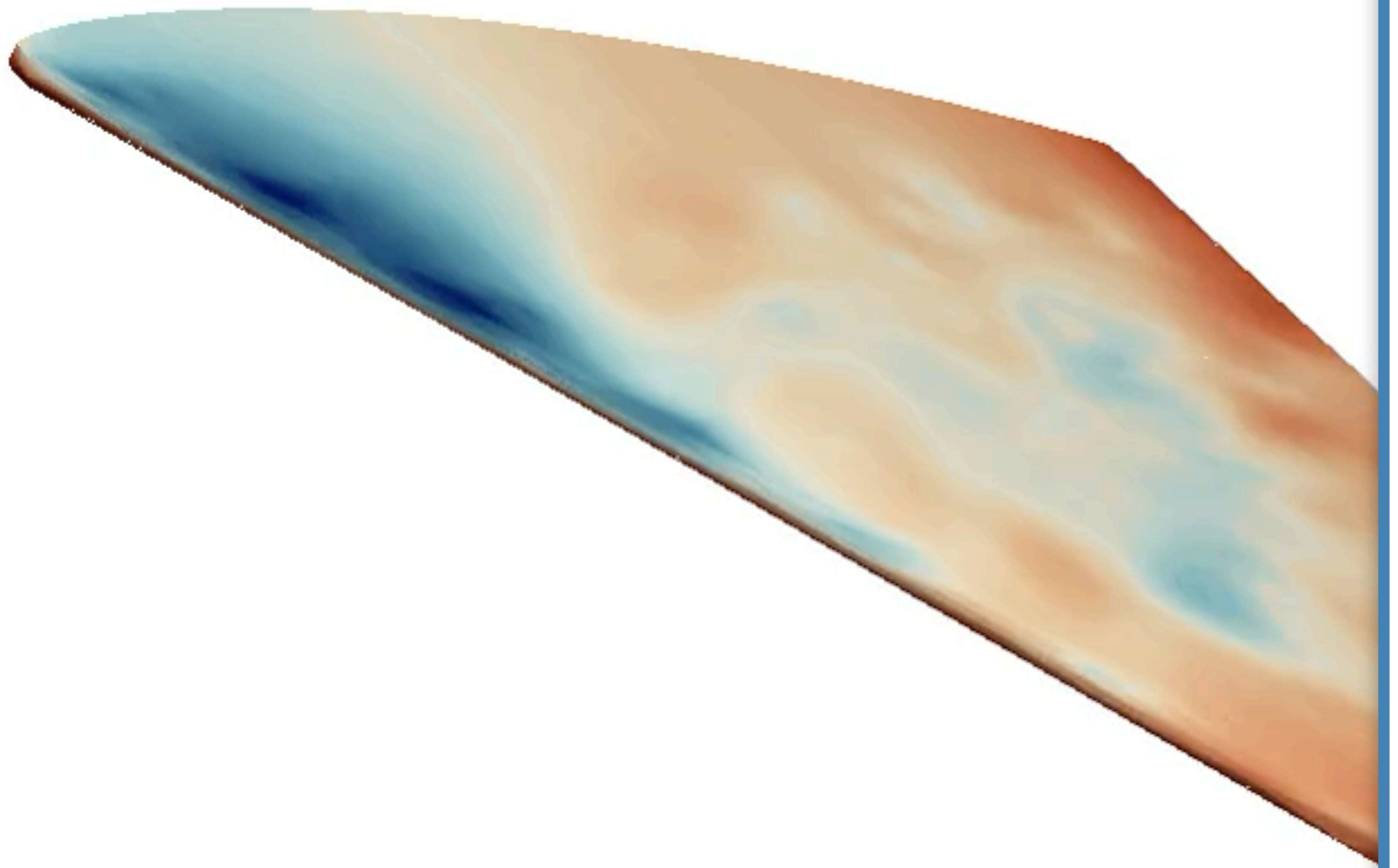


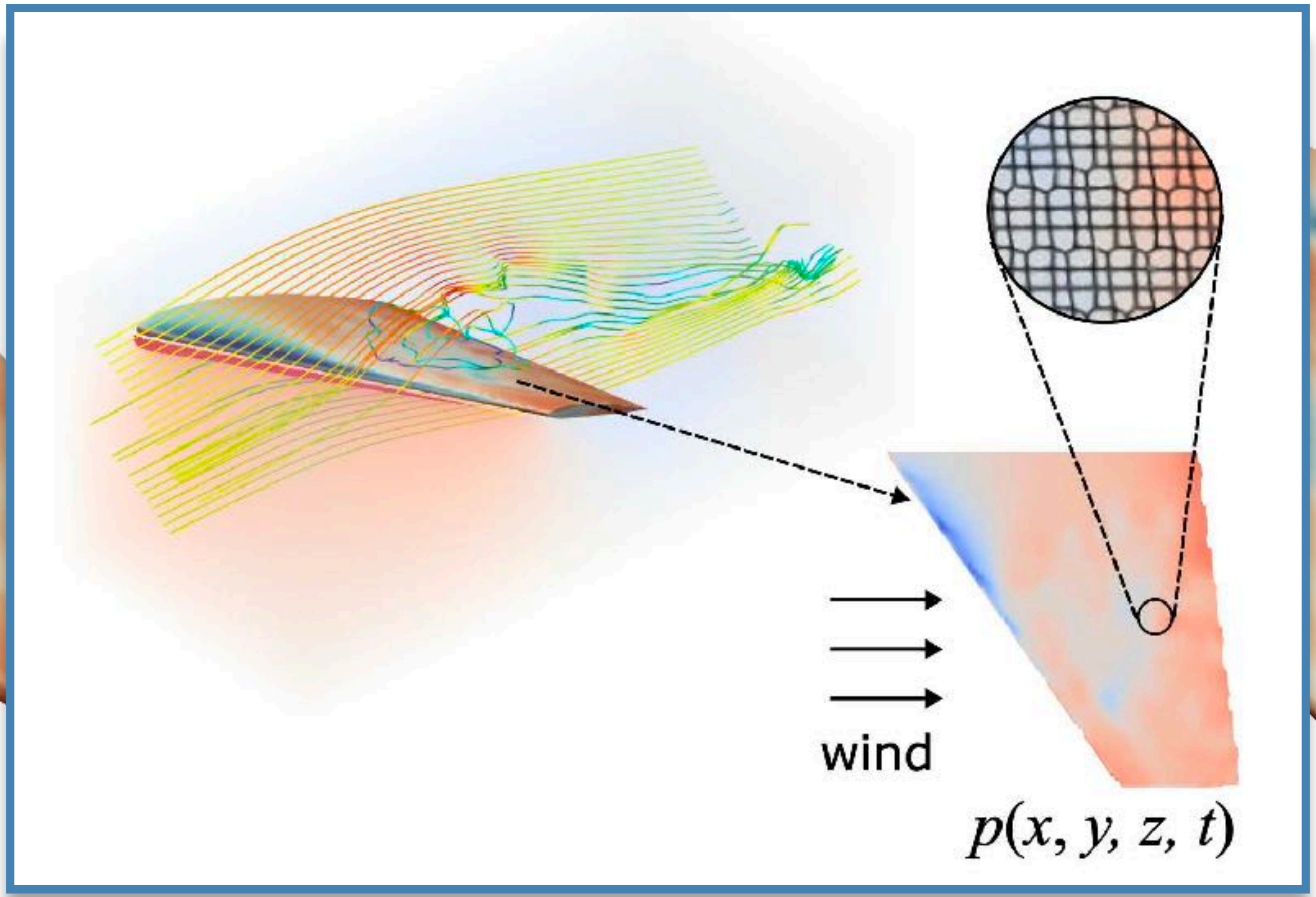
- 1D Airfoil Case again
- Diffusion iterations **n=10**
- Flow matching with n=10 roughly on-par with DDPM at n=200
- … plus improved training stability

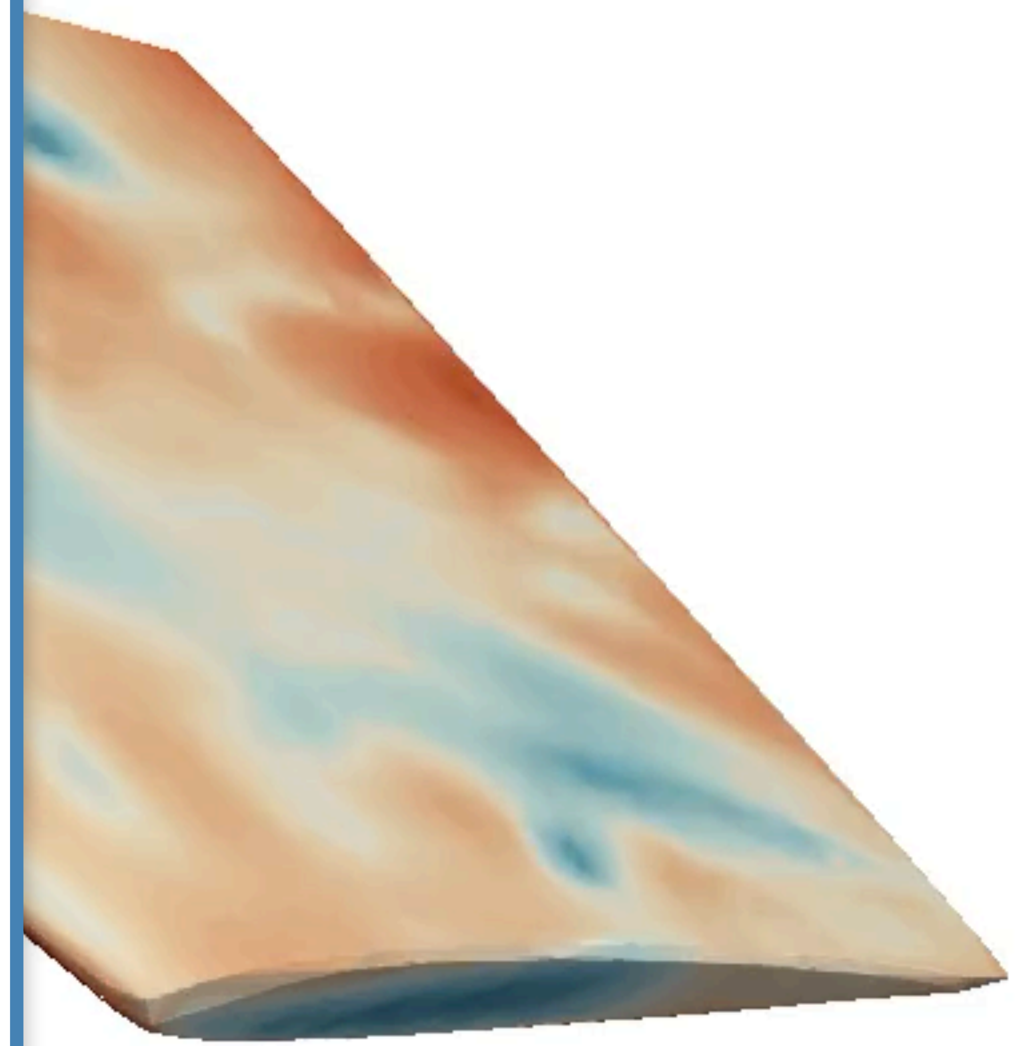
Liu et. al: Uncertainty Prediction of Airfoil Flow Simulations with DDPM

Flow Prediction for Unstructured Meshes

Targeting complex unsteady dynamics (no single mean solution)

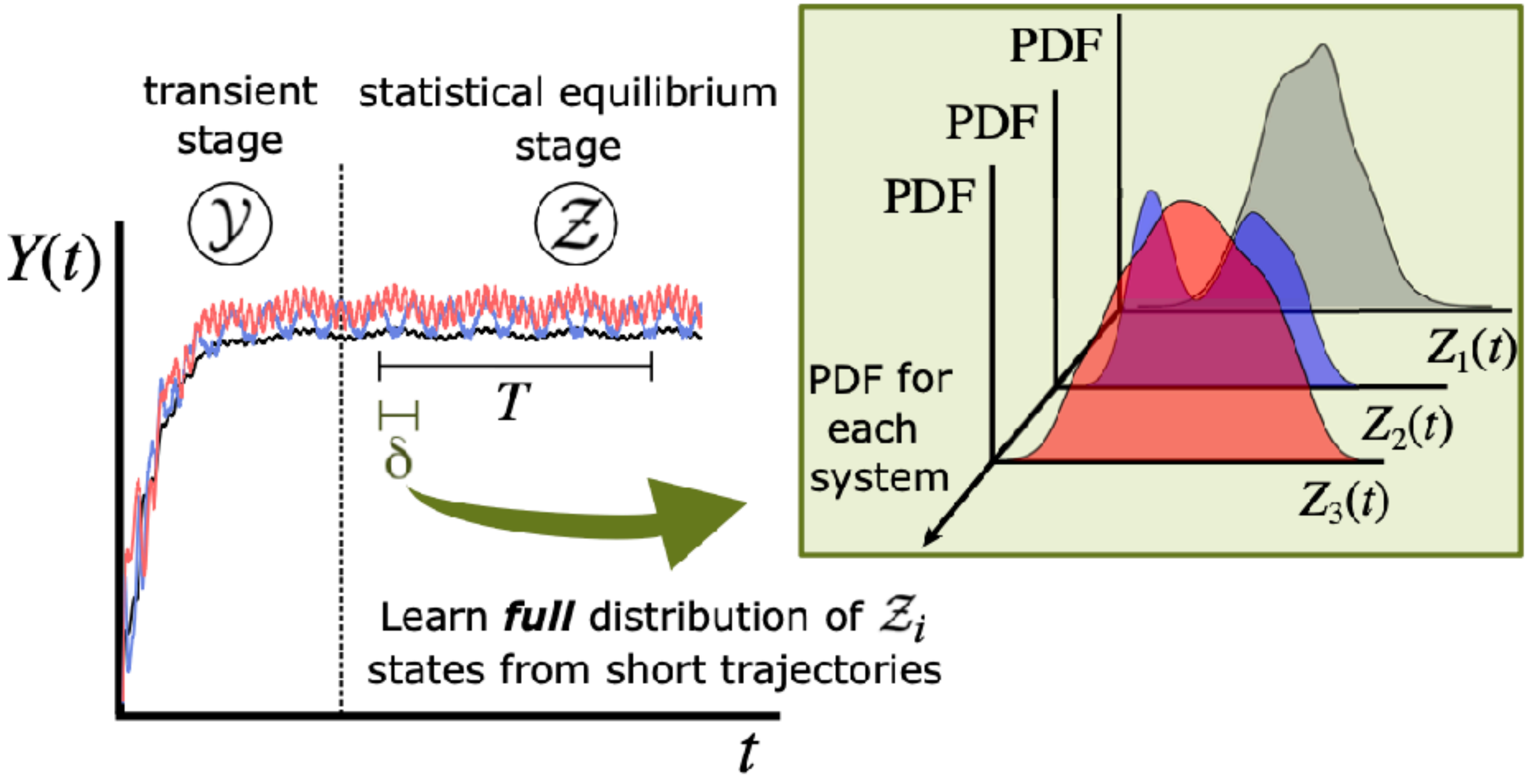






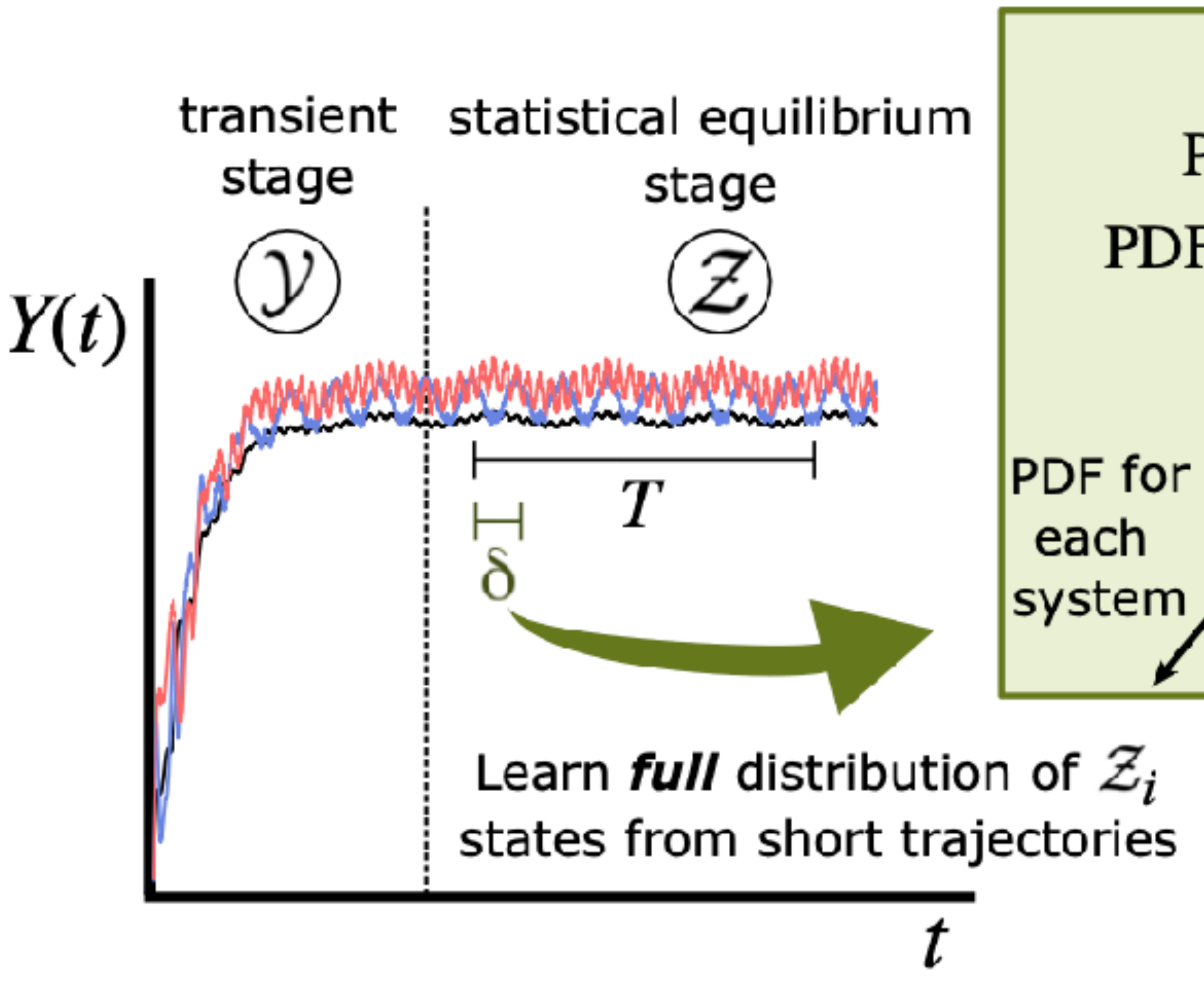
Lino et. al: Learning Distributions of Complex Fluid Simulations with Diffusion Graph Networks

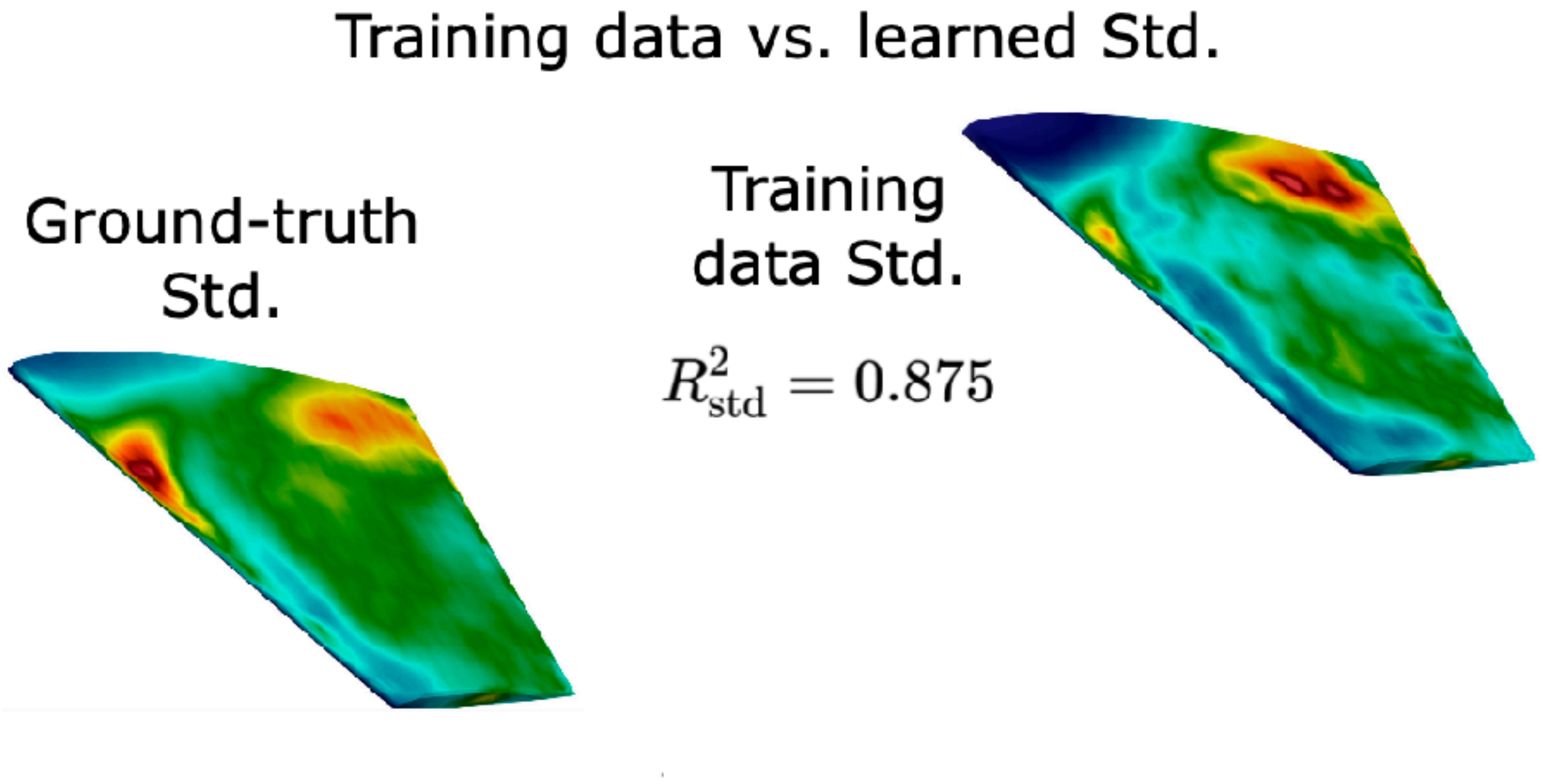
Flow Prediction for Unstructured Meshes



Lino et. al: Learning Distributions of Complex Fluid Simulations with Diffusion Graph Networks

Flow Prediction for Unstructured Meshes

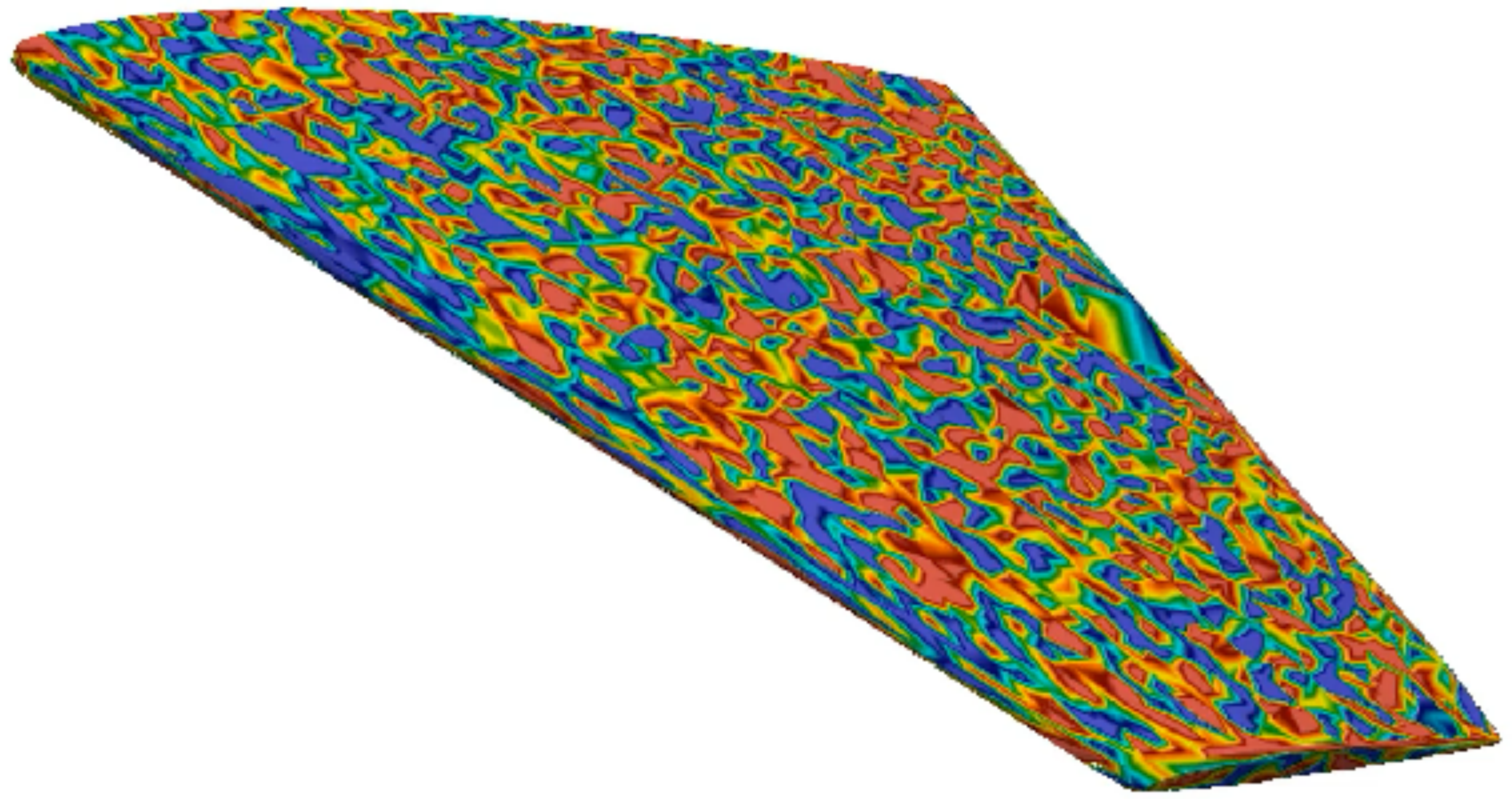




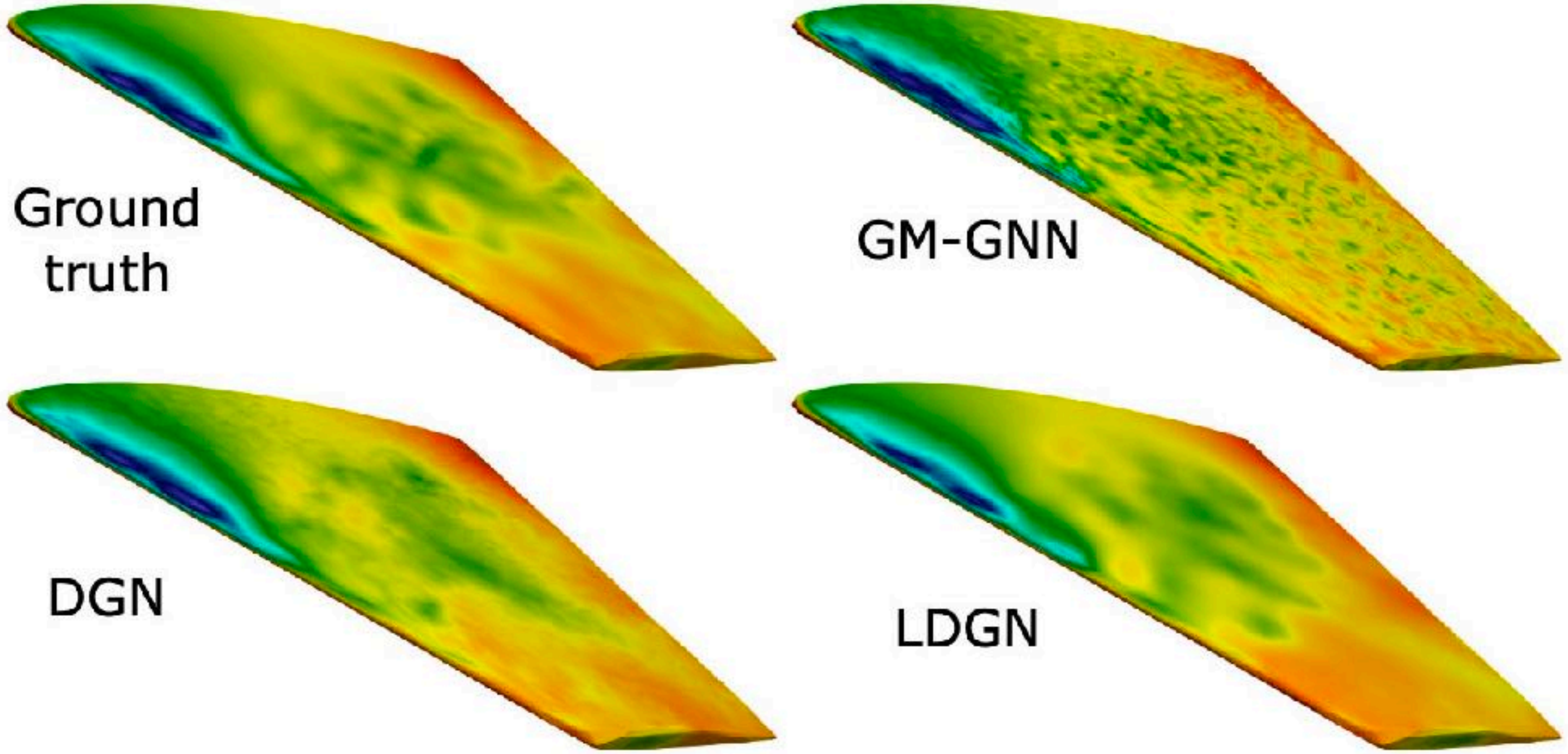
Lino et. al: Learning Distributions of Complex Fluid Simulations with Diffusion Graph Networks

Flow Prediction for Unstructured Meshes

Denoising Process



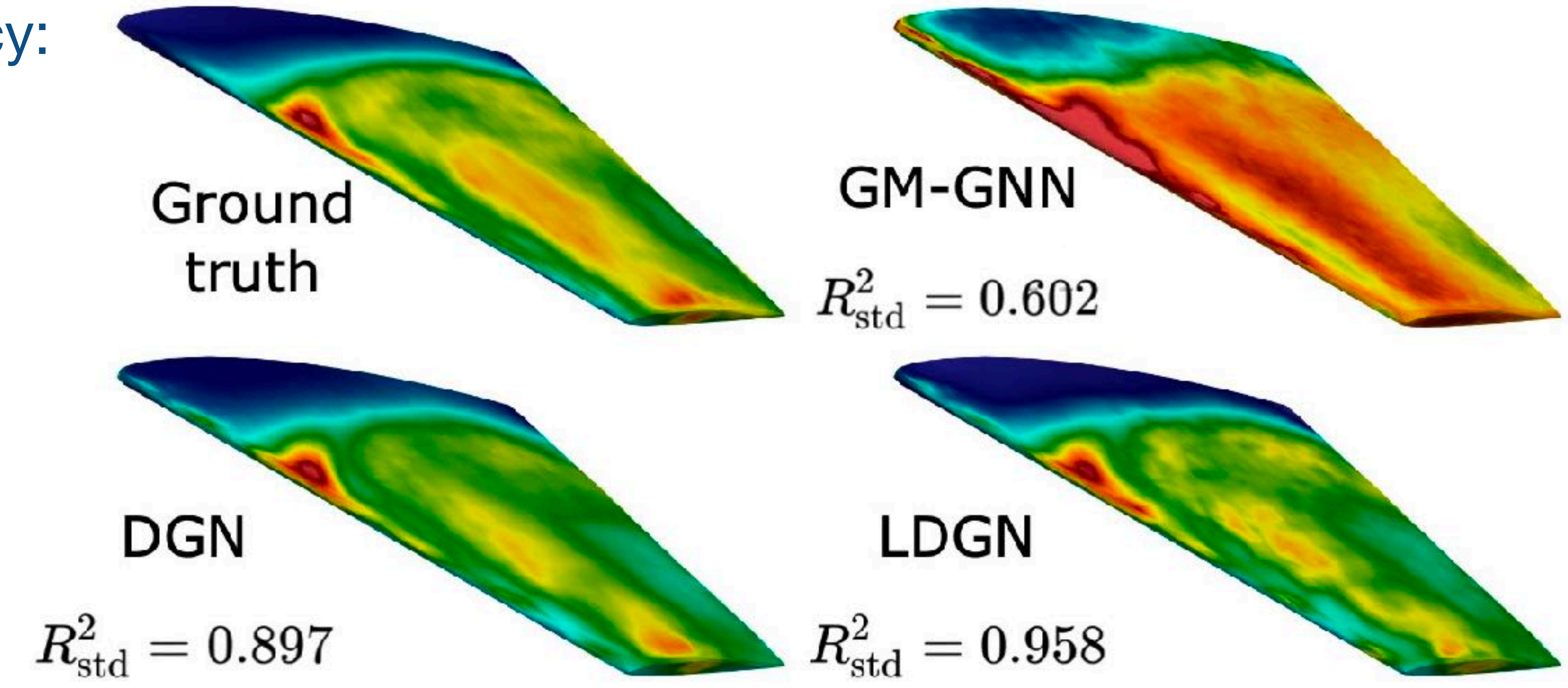
Produced Samples



Lino et. al: Learning Distributions of Complex Fluid Simulations with Diffusion Graph Networks

Flow Prediction for Unstructured Meshes

Excellent Distributional Accuracy:
(Inferred std. dev. on test case)



Model	**CPU** s/sample	**CPU** min/distribution (speedup)	**GPU** s/sample	**GPU** min/distribution (speedup)
DGN	6.81	340 (×8.8)	0.59	19 (×152)
LDGN	0.98	49 (×61)	0.20	2.43 (×1235)

Lino et. al: Learning Distributions of Complex Fluid Simulations with Diffusion Graph Networks

Probabilistic Models & Physical Constraints

Physics Constraints

- *So far, purely focused on learning complex distributions. Powerful tool, but (akin to supervised learning) no physics priors involved*

- Back to differentiable simulations & PDEs: how to include prior knowledge?

- Main options: invoke $\mathscr{P}$ at **training or inference time**

- **1) Inference time:** train diffusion model (DM) as usual; follow physics gradient when de-noising

- **2) Training time:** add physics gradient at training time; inference unmodified

1) Physics Constraints at Inference Time

- (From *Physics-informed Diffusion Models*) [Bastek et al. '25]
- For reference: regular DDPM on the right
- After step 4 add:

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

  - Predict approximate target $\tilde{x}_0$ (e.g., Tweedie's or PIDM) , the less noisy the better
  - Evaluate physics residual and make GD step with $\lambda_{\mathcal{P}}$ as step size: $x_{t-1} -= \lambda_{\mathcal{P}} \nabla_{\mathcal{P}}(\tilde{x}_0)$
  - Iterate…

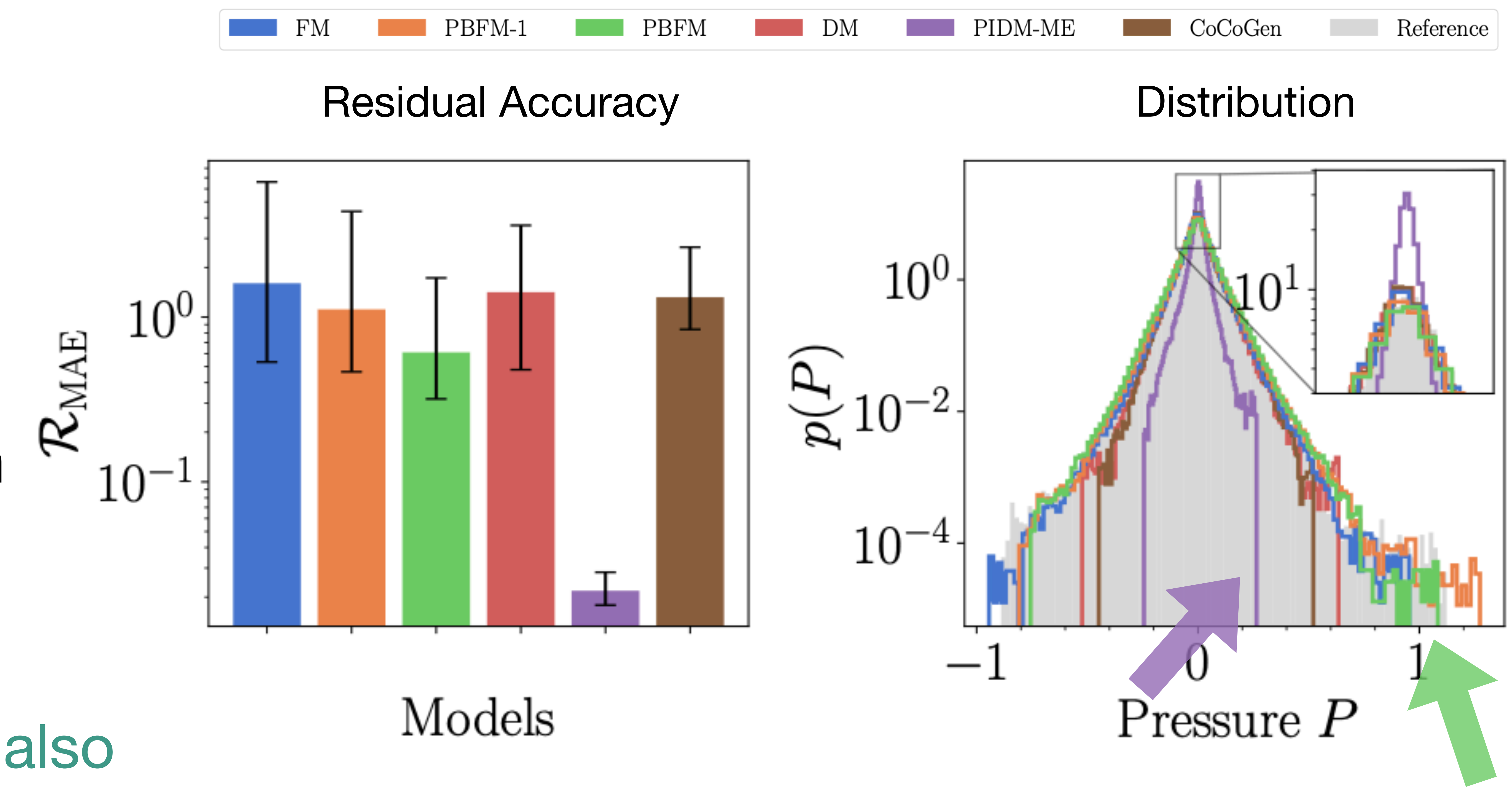
2) Physics Constraints at Training Time

- (From *Physics-based Flow Matching*)
  [Baldan et al. '25]
- Similar, but evaluate $\nabla_{\mathscr{P}}$ at for each batch
- Unrolling to better predict target $\tilde{x}_1$ for FM
- Evaluate physics residual to make "conflict-free" step
- Advantage: no additional cost at inference time.
  Disadvantage: slower training

$n \leftarrow$ number of unrolling steps
$dt \leftarrow (1-t)/n$
$\widetilde{t} \leftarrow t$
$u_t^\theta \leftarrow \text{model}(x_t,\ t)$
$\widetilde{x}_1 \leftarrow x_t + dt \cdot u_t^\theta$
for $i = 1,\ i < n$ **do**
  $\widetilde{t} = \widetilde{t} + dt$
  $\widetilde{u}_t^\theta \leftarrow \text{model}(\widetilde{x}_1,\ \widetilde{t})$
  $\widetilde{x}_1 \leftarrow \widetilde{x}_1 + dt \cdot \widetilde{u}_t^\theta$
end for
$\mathcal{R} \leftarrow \text{compute residual}(\widetilde{x}_1)$
$\mathcal{L}_\mathcal{R} \leftarrow \|t^p \cdot \mathcal{R}\|_2$
$\mathcal{L}_{\text{FM}} \leftarrow \|u_t^\theta - u_t\|_2$
$\nabla_\theta \leftarrow$ compute $\mathbf{g}_{\text{update}}$ via Eq. 3
AdamW optimizer step with ∇_θ

2) PBFM: Model Comparison

- Standard test case: Darcy flow (porous medium)
- PIDM (purple) has very low residual error, huge errors in distribution
- PBMF (green) better residual error than e.g. regular FM, still good distribution at same comp. cost
- Main conclusion: differentiable solvers also improve probabilistic learning!



Baldan et. al: Flow Matching meets PDEs: A Unified Framework for Physics-Constrained Generation

Summary & Outlook

Probabilistic Learning

✅ Powerful methods for learning complex distributions

✅ High training stability, and can handle large dimensionalities

✅ Important downstream tasks enabled, e.g., SBI and uncertainty quantification

❌ Increased training and inference cost

(❌ Unnecessarily slow for learning unique mappings)

Outlook

- Diffusion models can be transformed into continuous normalizing flows via the probability flow ODE → likelihood evaluation + deterministic sampling (Song et al. 2021, Lipman et al. 2023)

- Reduce the model size and cost by modeling high-dimensional data in a latent space (Rombach et al. 2021)

- How to condition models on different inputs, e.g. text (Rombach et al. 2019)

- Reduce the number of inference steps: network distillation and rectified flows (Ho et al. 2022, Liu et al. 2023)

- How to use diffusion models as priors for inverse problems? (Kawar et al. 2022, Chung et al. 2022)

- For simulation-based inference: include physics-based controls as self-conditioning to improve quality of samples (Holzschuh et al., 2025)

End