ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY OF MICHIGAN

Harnessing Low-dimensionality in Diffusion Models

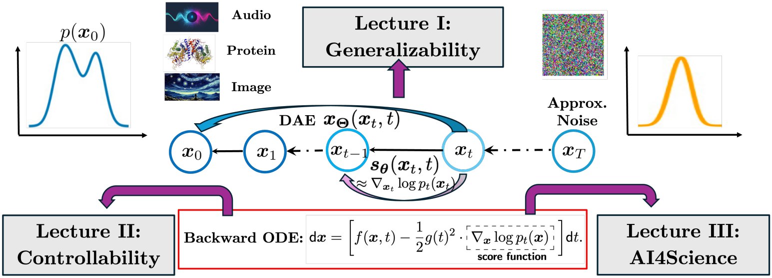
Lecture III: Solving Inverse Problems & AI for Science

Qing Qu

September 22, 2025

EECS, University of Michigan

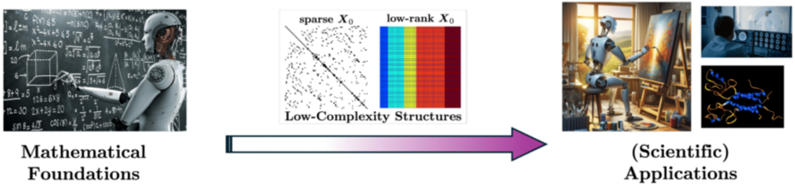
Lecture Schedule



We focus on the **mathematical foundations** of diffusion models through **low-dim structures** and their scientific applications:

- Introduction of Diffusion Models
- Lecture I: **Generalization** of Learning Diffusion Models
- Lecture II: **Controllability** of Diffusion Models
- Lecture III: From Theory to **Scientific Applications**

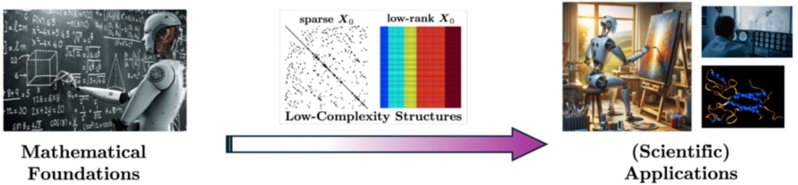
Application of Diffusion Models: Solving Inverse Problems



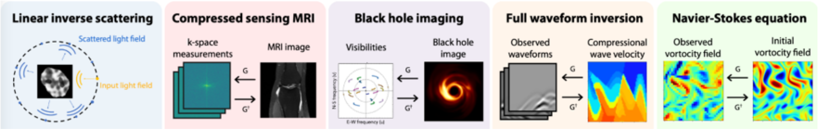
- Inverse problems are common across scientific applications.[1]
- Diffusion models learn strong priors to effectively solve them.

[1] Zheng et al., InverseBench: Benchmarking Plug-and-Play Diffusion Models for Inverse Problems in Physical Sciences, ICLR 2025.

Application of Diffusion Models: Solving Inverse Problems



- Inverse problems are common across scientific applications.[1]
- Diffusion models learn strong priors to effectively solve them.



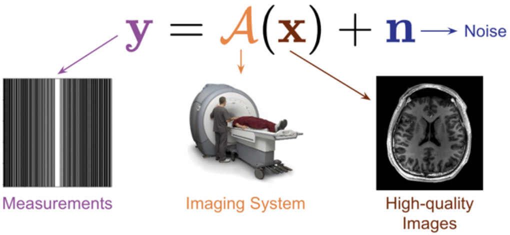
[1] Zheng et al., InverseBench: Benchmarking Plug-and-Play Diffusion Models for Inverse Problems in Physical Sciences, ICLR 2025.

Outline

1. Image Reconstruction Problems
2. Data Assimilation
3. Conclusion & Acknowledgement

Image Reconstruction Problems

Inverse Problems

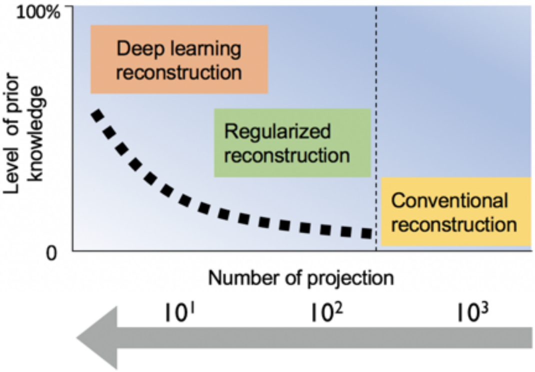


Goal: Recover signal $\boldsymbol{x} \in \mathbb{R}^n$ from noisy measurements $\boldsymbol{y} \in \mathbb{R}^m$ ($m \ll n$):

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$$

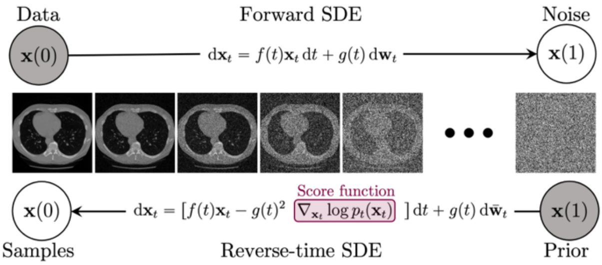
where $\mathcal{A}(\cdot)$ is a forward model, and $\boldsymbol{n}$ is some measurement noise.

Leverage Prior Knowledge for Solving Ill-Posed Inverse Problem



- **Conventional method:** requires dense sampling
- **Regularization method:** sparsity in transformed domain
- **Data driven method:** learned prior from a data-driven way

Recap: Score-based Diffusion Models



- **Forward diffusion process** as stochastic differential equation (SDE)
- **Generative reverse SDE**: uses score function to sample from prior $p(x)$

Diffusion models learn data prior by modeling **data distribution** through unsupervised training.

Solving Inverse Problems via Conditional Sampling

Goal: Recover signal $\boldsymbol{x} \in \mathbb{R}^n$ from $\boldsymbol{y} \in \mathbb{R}^m$ ($m \ll n$):

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$$

with forward model $\mathcal{A}(\cdot)$ and noise corruption $\boldsymbol{n}$.

Solving Inverse Problems via Conditional Sampling

Goal: Recover signal $\boldsymbol{x} \in \mathbb{R}^n$ from $\boldsymbol{y} \in \mathbb{R}^m$ ($m \ll n$):

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$$

with forward model $\mathcal{A}(\cdot)$ and noise corruption $\boldsymbol{n}$.

Idea of applying diffusion models: sampling from $p(\boldsymbol{x}|\boldsymbol{y})$ instead of $p(\boldsymbol{x})$ within the diffusion reverse process:

$$\nabla \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \underbrace{\nabla \log p_t(\boldsymbol{x}_t)}_{\text{we already have}} + \underbrace{\nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)}_{\text{missing \& intractable}} .$$

Solving Inverse Problems via Conditional Sampling

Goal: Recover signal $\boldsymbol{x} \in \mathbb{R}^n$ from $\boldsymbol{y} \in \mathbb{R}^m$ ($m \ll n$):

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$$

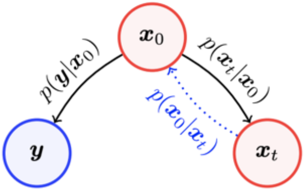
with forward model $\mathcal{A}(\cdot)$ and noise corruption $\boldsymbol{n}$.

Idea of applying diffusion models: sampling from $p(\boldsymbol{x}|\boldsymbol{y})$ instead of $p(\boldsymbol{x})$ within the diffusion reverse process:

$$\nabla \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \underbrace{\nabla \log p_t(\boldsymbol{x}_t)}_{\text{we already have}} + \underbrace{\nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)}_{\text{missing \& intractable}}.$$

- **Advantages:** unsupervised learning with limited assumptions, and comparable or better performance to supervised learning.
- **Challenges:** Estimating the posterior score $\nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$?

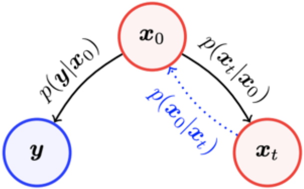
Diffusion Posterior Sampling (DPS, Chung et al. 2023)



- Posterior score decomposition:

$$\nabla \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \nabla \log p_t(\boldsymbol{x}_t) + \nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$$

Diffusion Posterior Sampling (DPS, Chung et al. 2023)



- Posterior score decomposition:

$$\nabla \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \nabla \log p_t(\boldsymbol{x}_t) + \nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$$

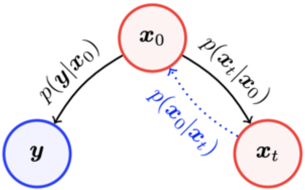
- Posterior mean estimation (Tweedie's Formula):

$$\hat{\boldsymbol{x}}_0 := \mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}} \left(\boldsymbol{x}_t + (1 - \bar{\alpha}(t))\nabla \log p_t(\boldsymbol{x}_t)\right)$$

- Approximating $\nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$ via

$$p(\boldsymbol{y} \mid \boldsymbol{x}_t) \simeq p(\boldsymbol{y} \mid \hat{\boldsymbol{x}}_0).$$

Diffusion Posterior Sampling (DPS, Chung et al. 2023)



- Decomposing posterior score:

$$\nabla \log p_t(\boldsymbol{x}_t \mid \boldsymbol{y}) = \nabla \log p_t(\boldsymbol{x}_t) + \nabla \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$$

- Take backpropagation through the network:

$$\begin{aligned}\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y} \mid \boldsymbol{x}_t) &\simeq \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y} \mid \hat{\boldsymbol{x}}_0) \\ &\simeq -\eta \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t))\|_2^2 \,.\end{aligned}$$

Diffusion Posterior Sampling (DPS, Chung et al. 2023)

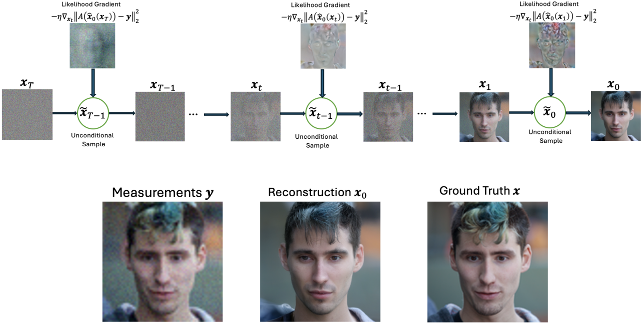
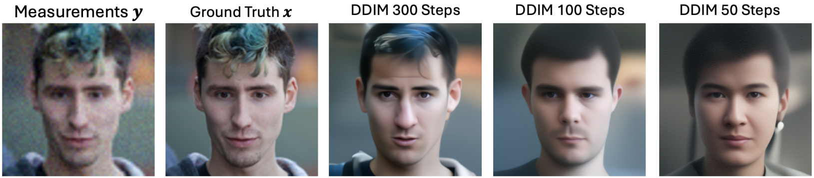


Figure 1: Diffusion Posterior Sampling Process

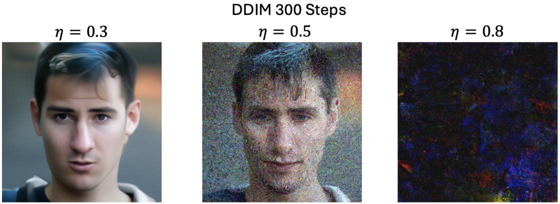
Guide the unconditional diffusion sampling process with the conditional gradient $-\eta\nabla_{\boldsymbol{x}_t}\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t))\|_2^2$.

Limitations of Coupled Sampling & Data Consistency



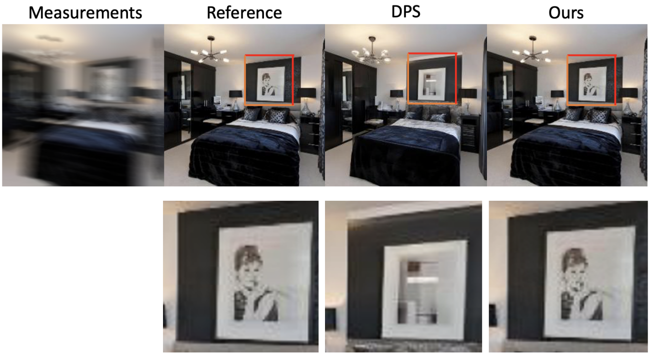
- **Limitation 1: large number of sampling steps.**
  - To ensure data consistency, it cannot benefit from faster samplers such as DDIM (Song et al. 2020) and Consistency Models (Song et al. 2023).
  - This is because **coupled** sampling and data consistency.

Limitations of DPS



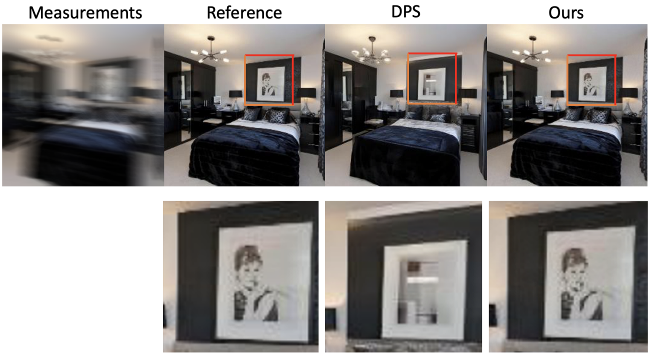
- **Limitation 1: large number of sampling steps.**
  - Increasing step size η of the likelihood gradient $-\eta \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t))\|_2^2$ **does not** help.
  - The likelihood gradient is also **expensive to compute** due to backpropagation.

Limitations of DPS



- **Limitation 2: Inconsistent reconstructions.**

Limitations of DPS



- **Limitation 2: Inconsistent reconstructions.**
- **Limitation 3: Difficult to adapt to latent diffusion.**

Our Method: Decoupled Data Consistency

Consider the regularized optimization formulation:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda\mathcal{R}(\boldsymbol{x})$$

Our Method: Decoupled Data Consistency

Consider the regularized optimization formulation:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda \mathcal{R}(\boldsymbol{x})$$

Question: How do we **better** utilize the diffusion model as the data prior?

Our Method: Decoupled Data Consistency

Consider the regularized optimization formulation:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda \mathcal{R}(\boldsymbol{x})$$

Question: How do we **better** utilize the diffusion model as the data prior?

- Introducing an auxiliary variable $\boldsymbol{v}$:

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda \mathcal{R}(\boldsymbol{v}), \text{ s.t. } \boldsymbol{x} = \boldsymbol{v}$$

Our Method: Decoupled Data Consistency

Consider the regularized optimization formulation:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda \mathcal{R}(\boldsymbol{x})$$

Question: How do we **better** utilize the diffusion model as the data prior?

- Introducing an auxiliary variable $\boldsymbol{v}$:

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \lambda \mathcal{R}(\boldsymbol{v}), \text{ s.t. } \boldsymbol{x} = \boldsymbol{v}$$

- Applying half quadratic splitting (HQS):

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}||_2^2 + \lambda \mathcal{R}(\boldsymbol{v})$$

Decouple the Diffusion Prior from Data Consistency

Alternating minimization between x and v for solving

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}||_2^2 + \lambda\mathcal{R}(\boldsymbol{v})$$

Decouple the Diffusion Prior from Data Consistency

Alternating minimization between x and v for solving

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}||_2^2 + \lambda\mathcal{R}(\boldsymbol{v})$$

- Data consistency optimization via gradient descent:

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

Decouple the Diffusion Prior from Data Consistency

Alternating minimization between $\boldsymbol{x}$ and $\boldsymbol{v}$ for solving

$$\min_{\boldsymbol{x},\boldsymbol{v}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}||_2^2 + \lambda\mathcal{R}(\boldsymbol{v})$$

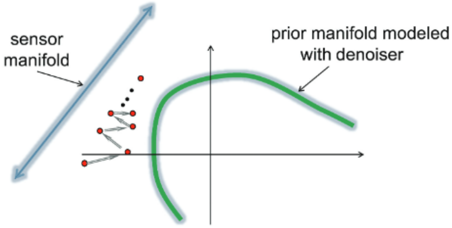
- Data consistency optimization via gradient descent:

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

- Enforcing image prior (**through diffusion models**):

$$\boldsymbol{v}_k = \arg\min_{\boldsymbol{v}_k} \mu||\boldsymbol{x}_k - \boldsymbol{v}||_2^2 + \lambda\mathcal{R}(\boldsymbol{v})$$

Decouple the Diffusion Prior from Data Consistency



Decoupling via variable splitting and alternating minimization:

- Data consistency optimization via gradient descent:

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

- Enforcing image prior (**through diffusion models**):

$$\boldsymbol{v}_k = \arg\min_{\boldsymbol{v}_k} \mu||\boldsymbol{x}_k - \boldsymbol{v}||_2^2 + \lambda\mathcal{R}(\boldsymbol{v})$$

Step I: Data Consistency Optimization

- **Data fidelity optimization.** Generate measurement-consistent reconstructions (suffer from artifacts):

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2,$$

which can be solved via gradient descent.

Step II: Enforcing Image Prior via Diffusion Purification

Question: Can we refine x_k with pre-trained diffusion models?

[2] Nie et al. Diffusion Models for Adversarial Purification, 2022.

Step II: Enforcing Image Prior via Diffusion Purification

Question: Can we refine x_k with pre-trained diffusion models?

Solution: Diffusion purification[2] by adding noise and then denoising

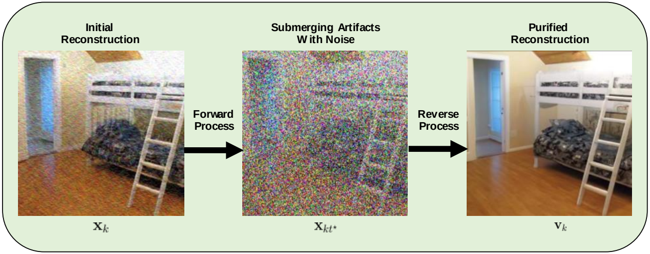
- Run forward process to a certain intermediate noise level t^*.

$$\boldsymbol{x}_{k,t^*} = \boldsymbol{x}_k + \sigma_{t^*}\boldsymbol{\epsilon},$$

- Run the reverse sampling process from $\boldsymbol{x}_{k,t^\star}$ to obtain $\boldsymbol{v}_k$.

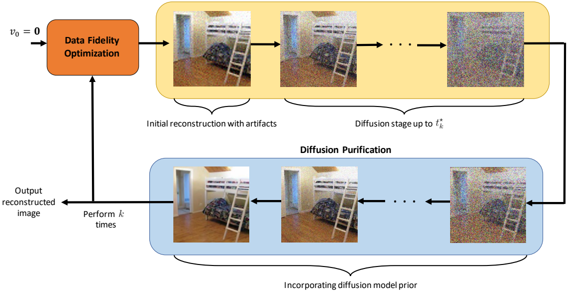
[2]Nie et al. Diffusion Models for Adversarial Purification, 2022.

Step II: Enforcing Image Prior via Diffusion Purification



- **Effectiveness:** With a properly chosen $t^\star$, the image artifacts can be effectively removed while the overall image structures are preserved.
- **Efficiency:** Diffusion purification can be efficiently implemented through fast samplers, such as DDIM and consistency models. It can also be easily adapted to the latent space.

Decoupled Data Consistency via Diffusion Purification (DCDP)



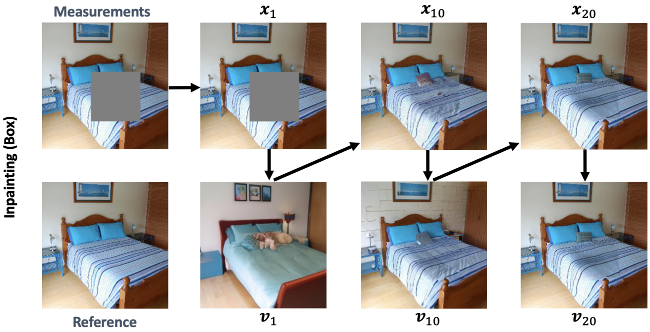
- Step I: Enforcing data consistency:

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

- Step II: Applying diffusion purification:

$$\boldsymbol{v}_k = \text{DPUR}(\boldsymbol{x}_k, t_k^*)$$

Decoupled Data Consistency via Diffusion Purification (DCDP)



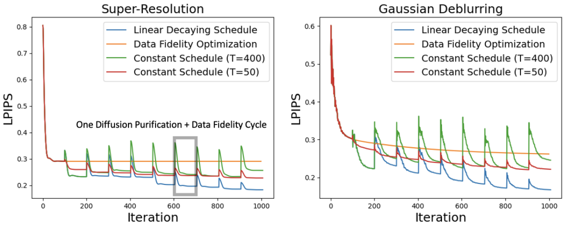
- Enforcing data consistency:

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

- Applying diffusion purification:

$$\boldsymbol{v}_k = \text{DPUR}(\boldsymbol{x}_k, t_k^*)$$

Decoupled Data Consistency via Diffusion Purification (DCDP)



- Enforcing data consistency:

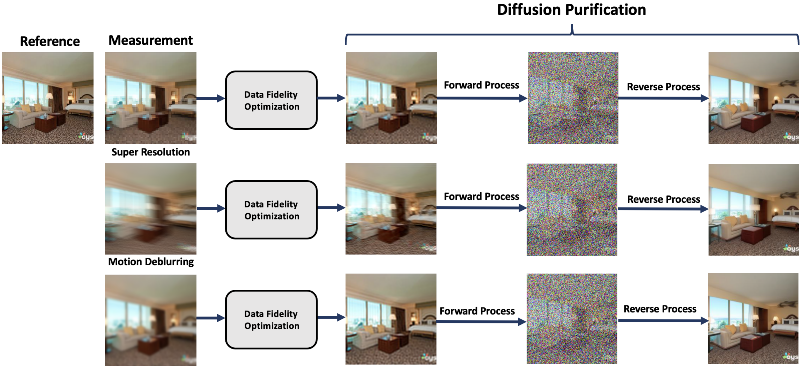
$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}||_2^2 + \mu||\boldsymbol{x} - \boldsymbol{v}_{k-1}||_2^2$$

- Applying diffusion purification:

$$\boldsymbol{v}_k = \text{DPUR}(\boldsymbol{x}_k, t_k^*)$$

Use a **decaying** purification strength $t_k^\star$ across iteration k for stable convergence.

Effectiveness of Diffusion Purification



- **Effectiveness:** Diffusion Purification is effective for various inverse problems.

Efficiency of Diffusion Purification via DDIM

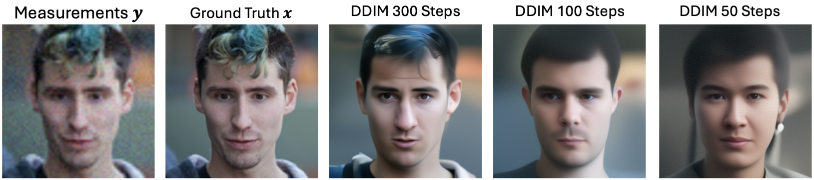


Figure 2: DPS: insufficient sampling Steps leads to inconsistent samples

Efficiency of Diffusion Purification via DDIM

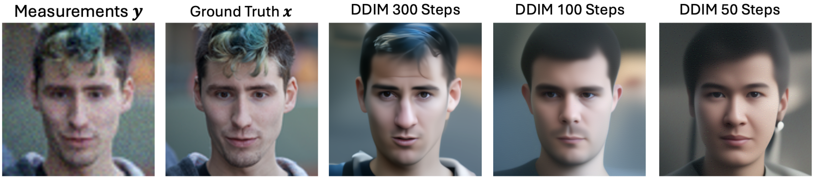
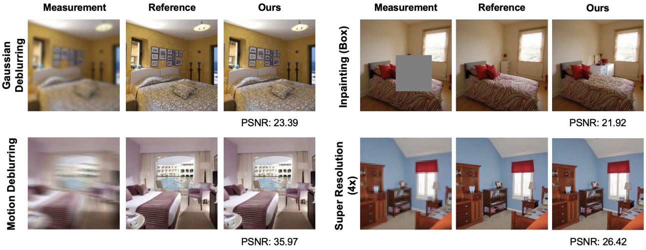


Figure 2: DPS: insufficient sampling Steps leads to inconsistent samples

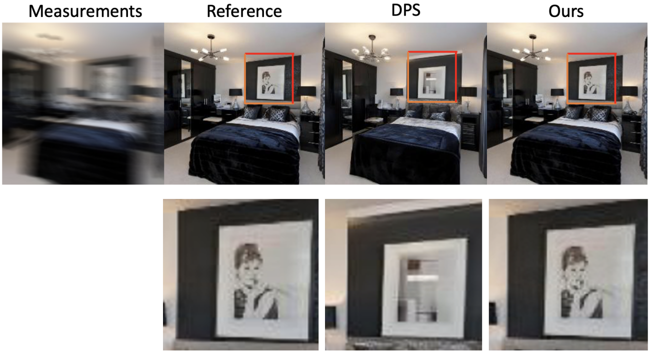
Figure 3: DCDP: Fast diffusion purification via DDIM.

Efficiency of Diffusion Purification via Consistency Models



- Diffusion Purification can efficiently be performed using Consistency Models (one step).

Improved Data Consistency via DCDP



- DCDP is more data consistent with the measurements.

Competitive Performance in the Latent Space

Competitive Performance: Quantitative Comparison

Method	Gaussian Deblurring			Motion Deblurring		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
DPS [14]	0.213 ± 0.08	24.45 ± 3.72	0.691 ± 0.132	0.182 ± 0.03	24.45 ± 2.93	0.736 ± 0.06
MCG [11]	0.311 ± 0.13	17.54 ± 5.06	0.551 ± 0.191	0.365 ± 0.11	20.17 ± 3.73	0.515 ± 0.36
ADMM-PnP [49]	0.437 ± 0.04	20.76 ± 1.94	0.595 ± 0.09	0.524 ± 0.04	18.05 ± 2.05	0.493 ± 0.10
DDNM [13]	0.217 ± 0.07	$\underline{27.30} \pm 1.67$	$\underline{0.815} \pm 0.08$	–	–	–
RED-Diff [45]	0.228 ± 0.12	23.98 ± 1.23	0.687 ± 0.067	0.178 ± 0.06	24.15 ± 2.34	0.704 ± 0.12
DiffPIR [50]	0.214 ± 0.013	27.04 ± 2.67	0.801 ± 0.12	0.114 ± 0.12	27.45 ± 1.89	0.866 ± 0.65
DCDP-DDIM (Ours)	$\mathbf{0.196} \pm 0.05$	27.13 ± 3.26	0.804 ± 0.07	$\underline{0.065} \pm 0.01$	$\underline{33.56} \pm 3.92$	$\underline{0.947} \pm 0.02$
DCDP-Tweedie (Ours)	$\underline{0.212} \pm 0.05$	$\mathbf{27.74} \pm 3.34$	$\mathbf{0.825} \pm 0.07$	$\mathbf{0.067} \pm 0.02$	$\mathbf{34.47} \pm 4.07$	$\mathbf{0.956} \pm 0.02$
Latent-DPS	0.337 ± 0.05	23.75 ± 2.53	0.622 ± 0.10	0.425 ± 0.06	21.90 ± 2.31	0.539 ± 0.10
PSLD [18]	0.373 ± 0.07	24.26 ± 2.84	0.683 ± 0.11	0.469 ± 0.06	20.58 ± 2.32	0.562 ± 0.11
ReSample [19]	$\underline{0.240} \pm 0.05$	25.76 ± 3.02	0.731 ± 0.09	0.188 ± 0.04	27.96 ± 3.07	0.806 ± 0.07
DCDP-LDM-DDIM (Ours)	0.246 ± 0.05	$\underline{26.08} \pm 3.02$	$\underline{0.766} \pm 0.07$	$\underline{0.145} \pm 0.03$	$\underline{28.87} \pm 3.77$	$\underline{0.856} \pm 0.06$
DCDP-LDM-Tweedie (Ours)	$\mathbf{0.217} \pm 0.05$	$\mathbf{27.11} \pm 3.26$	$\mathbf{0.807} \pm 0.07$	$\mathbf{0.121} \pm 0.03$	$\mathbf{29.40} \pm 3.81$	$\mathbf{0.875} \pm 0.05$
DCDP-CMs (Ours)	0.202 ± 0.05	26.91 ± 3.11	0.805 ± 0.07	0.061 ± 0.02	34.10 ± 3.80	0.953 ± 0.02

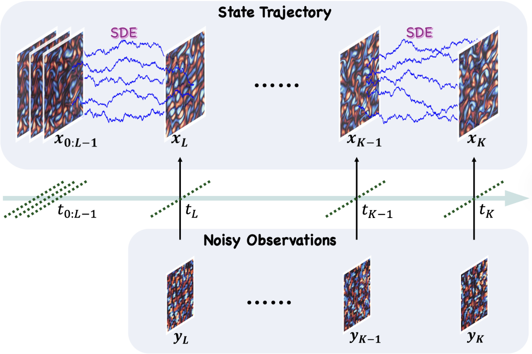
Discussion

- Diffusion models can serve as an implicit regularization for solving ill-posed inverse problems.
- Decoupling the data consistency can significantly improve the efficiency and flexibility.
- Limitation: Perception-distortion tradeoff (a potential direction to explore)

1. Xiang Li, Soo Min Kwon, Ismail R. Alkhouri, Saiprasad Ravishankar, Qing Qu. Decoupled Data Consistency with Diffusion Purification for Image Restoration. *Under Review at IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, 2025.

Data Assimilation

Background: Data Assimilation



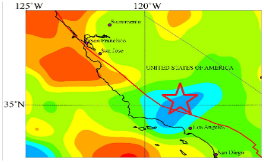
- **Stochastic dynamic systems:**

$$x_k = \underbrace{\Psi(x_{k-1})}_{\text{transition map}} + \underbrace{\xi_{k-1}}_{\text{stochastic force}}$$

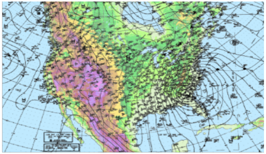
- **(Partial) nosiy observation models:**

$$y_k = \underbrace{\mathcal{A}(x_k)}_{\text{partial observation}} + \eta_k$$

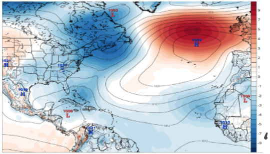
Background: Data Assimilation



Earthquake Prediction



Weather Forecasting



Hurricane Prediction

Data Assimilation (DA): combine observations y_k with numerical models to **predict states** x_k in stochastic dynamic systems.

$$x_k = \underbrace{\Psi(x_{k-1})}_{\text{transition map}} + \underbrace{\xi_{k-1}}_{\text{stochastic force}},$$

$$y_k = \underbrace{\mathcal{A}(x_k)}_{\text{partial observation}} + \eta_k.$$

Classical Approaches for DA

Model-based methods (require state transition model Ψ):

- **Kalman filter:** assumption of Gaussian noise and **linear** dynamics

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}$$
$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}\right)$$

Classical Approaches for DA

Model-based methods (require state transition model Ψ):

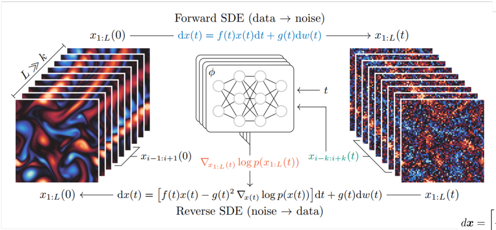
- **Kalman filter:** assumption of Gaussian noise and **linear** dynamics

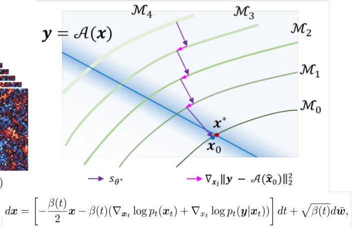
$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}$$
$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}\right)$$

- **Particle filter:** curse of dimensionality, computationally **too expensive** in high-dimension

$$\boldsymbol{x}_k^i = \Psi(\boldsymbol{x}_{(k-1)}^i) + \boldsymbol{\xi}_k^i$$
$$\hat{\boldsymbol{x}}_k = \sum_{i=1}^{N} \omega_k^i \boldsymbol{x}_k^i, \quad \omega_k^i \propto p(\boldsymbol{y}_k \mid \boldsymbol{x}_k^i)$$

Data-driven Methods for DA

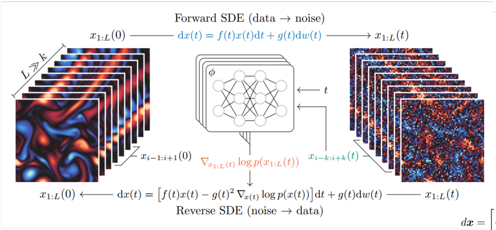


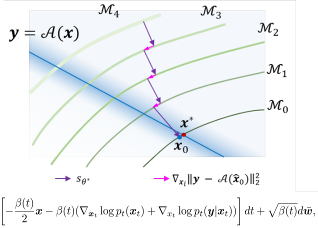


Data-driven methods:

- **Score-based diffusion models (SDA):** learn joint distribution $p(\boldsymbol{x}_{k-\tau},, \cdots, \boldsymbol{x}_k, \cdots, \boldsymbol{x}_{k+\tau} \mid \boldsymbol{y}_{k+\tau})$ of state priors (non-autoregressive)

Data-driven Methods for DA

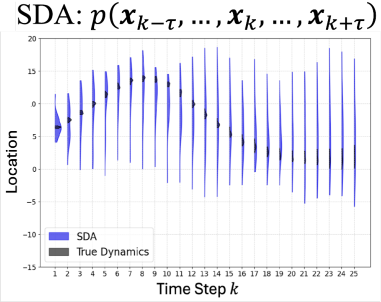


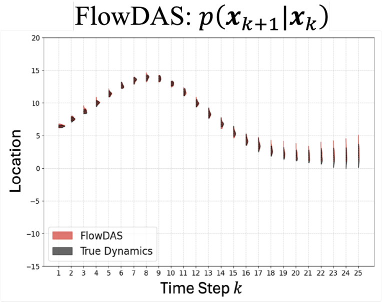


Data-driven methods:

- **Score-based diffusion models (SDA):** learn joint distribution $p(\boldsymbol{x}_{k-\tau},, \cdots, \boldsymbol{x}_k, \cdots, \boldsymbol{x}_{k+\tau} \mid \boldsymbol{y}_{k+\tau})$ of state priors (non-autoregressive)
- **Our Method: FlowDAS** explicitly models **system transition** dynamics $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$ (autoregressive)

Data-driven Methods for DA





Data-driven methods:

- **Score-based diffusion models (SDA):** learn joint distribution $p(\boldsymbol{x}_{k-\tau}, , \cdots, \boldsymbol{x}_k, \cdots, \boldsymbol{x}_{k+\tau} \mid \boldsymbol{y}_k)$ of state priors (non-autoregressive)
- **Our Method: FlowDAS** explicitly models **system transition dynamics** $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$ (autoregressive)

FlowDAS: A Flow-Based Framework for Data Assimilation

Goal: Given the state variable $\boldsymbol{x}_{k-1} \sim p(\boldsymbol{x}_{k-1})$ at k, estimate $q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$

- **Learning a predictor:** estimate $\boldsymbol{x}'_k \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{x}'_k \mid \boldsymbol{x}_{k-1})$ via stochastic interpolant (SI).[3]

[3] M. S. Albergo et al., Stochastic Interpolants: A Unifying Framework for Flows and Diffusions.

[4] H. Chung et al., Diffusion Posterior Sampling for General Noisy Inverse Problems

FlowDAS: A Flow-Based Framework for Data Assimilation

Goal: Given the state variable $\boldsymbol{x}_{k-1} \sim p(\boldsymbol{x}_{k-1})$ at k, estimate $q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$

- **Learning a predictor:** estimate $\boldsymbol{x}'_k \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{x}'_k \mid \boldsymbol{x}_{k-1})$ via stochastic interpolant (SI).[3]
- **Correction using partial observation:** leverage $\boldsymbol{y}_k$ to estimate $q(\boldsymbol{x}_k \mid \boldsymbol{x}'_k, \boldsymbol{y}_k)$ inspired by *denoising posterior sampling* (DPS)[4].

[3] M. S. Albergo et al., Stochastic Interpolants: A Unifying Framework for Flows and Diffusions.

[4] H. Chung et al., Diffusion Posterior Sampling for General Noisy Inverse Problems

FlowDAS: A Flow-Based Framework for Data Assimilation

Goal: Given the state variable $\boldsymbol{x}_{k-1} \sim p(\boldsymbol{x}_{k-1})$ at k, estimate $q(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$

- **Learning a predictor:** estimate $\boldsymbol{x}'_k \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{x}'_k \mid \boldsymbol{x}_{k-1})$ via stochastic interpolant (SI).[3]
- **Correction using partial observation:** leverage $\boldsymbol{y}_k$ to estimate $q(\boldsymbol{x}_k \mid \boldsymbol{x}'_k, \boldsymbol{y}_k)$ inspired by *denoising posterior sampling* (DPS)[4].
- **Autoregressive:** estimate $\boldsymbol{x}_{k+1} \sim q(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{y}_{k+1})$ and repeat.

[3] M. S. Albergo et al., Stochastic Interpolants: A Unifying Framework for Flows and Diffusions.
[4] H. Chung et al., Diffusion Posterior Sampling for General Noisy Inverse Problems

Stochastic Interpolants (SI) for Probabilistic Forecasting

Consider a stochastic process $\boldsymbol{X}_s$ defined over the interval $s \in [0, 1]$, evolving from initial state $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. The SI can be described as

$$\boldsymbol{X}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \boldsymbol{X}_1 + \sigma_s \boldsymbol{W}_s$$

where $\alpha_s, \beta_s, \sigma_s$ are drift coefficients, and $\boldsymbol{W}_s$ is a Wiener process.

Stochastic Interpolants (SI) for Probabilistic Forecasting

Consider a stochastic process $\boldsymbol{X}_s$ defined over the interval $s \in [0, 1]$, evolving from initial state $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. The SI can be described as

$$\boldsymbol{X}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \boldsymbol{X}_1 + \sigma_s \boldsymbol{W}_s$$

where $\alpha_s, \beta_s, \sigma_s$ are drift coefficients, and $\boldsymbol{W}_s$ is a Wiener process.

- Let $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ be the “velocity” of the interpolant path, then $\boldsymbol{X}_s$ follows the SDE

$$\mathrm{d}\boldsymbol{X}_s = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)\,\mathrm{d}s + \sigma_s\,\mathrm{d}\boldsymbol{W}_s,$$

Stochastic Interpolants (SI) for Probabilistic Forecasting

Consider a stochastic process $\boldsymbol{X}_s$ defined over the interval $s \in [0,1]$, evolving from initial state $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. The SI can be described as

$$\boldsymbol{X}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \boldsymbol{X}_1 + \sigma_s \boldsymbol{W}_s$$

where $\alpha_s, \beta_s, \sigma_s$ are drift coefficients, and $\boldsymbol{W}_s$ is a Wiener process.

- Let $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ be the "velocity" of the interpolant path, then $\boldsymbol{X}_s$ follows the SDE

$$\mathrm{d}\boldsymbol{X}_s = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)\,\mathrm{d}s + \sigma_s\,\mathrm{d}\boldsymbol{W}_s,$$

- The drift (velocity) term $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ can be learned through

$$\mathcal{L}_b(\hat{\boldsymbol{b}}_s) = \int_0^1 \mathbb{E}\left[\|\hat{\boldsymbol{b}}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) - \boldsymbol{R}_s\|^2\right] ds.$$

  where $\boldsymbol{R}_s = \dot{\alpha}_s \boldsymbol{X}_0 + \dot{\beta}_s \boldsymbol{X}_1 + \dot{\sigma}_s \boldsymbol{W}_s$.

Observation Consistent Prediction

To ensure measurement consistency $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x})$, FlowDAS augments the original drift $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ via Bayes' rule

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}.$$

Observation Consistent Prediction

To ensure measurement consistency $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x})$, FlowDAS augments the original drift $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ via Bayes' rule

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}.$$

- The conditional score $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ captures the observation information, but intractable.

Observation Consistent Prediction

To ensure measurement consistency $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x})$, FlowDAS augments the original drift $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ via Bayes' rule

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}.$$

- The conditional score $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ captures the observation information, but intractable.
- In practice, we estimate it by Monte-Carlo marginalization and do Monte Carlo sampling during inference.

Algorithm 2 Inference

Input: Observation $\boldsymbol{y}_{L:K}$, the measurement map $\mathcal{A}$, initial state $\boldsymbol{x}_0$, model $\hat{\boldsymbol{b}}_s(\boldsymbol{X}, \boldsymbol{X}_0)$, noise coefficient σ_s, grid $s_0 = 0 < s_1 < \cdots < s_N = 1$, i.i.d. $\boldsymbol{z}_n \sim \mathcal{N}(0, \boldsymbol{I}_D)$ for $n = 0 : N-1$, step size ζ_n, Monte Carlo sampling times J

Set $\hat{\boldsymbol{x}}_{L-1} \leftarrow \boldsymbol{x}_{L-1}$

Set the $(\Delta s)_n = s_{n+1} - s_n,\ n = 0 : N-1$

for $k = L-1$ **to** $K-1$ **do**

    $\boldsymbol{X}_{s_0}, \boldsymbol{y} \leftarrow \hat{\boldsymbol{x}}_k, \boldsymbol{y}_{k+1}$

    for $n = 0$ **to** $N-1$ **do**

        $\boldsymbol{X}'_{s_{n+1}} = \boldsymbol{X}_{s_n} + \hat{\boldsymbol{b}}_s(\boldsymbol{X}_{s_n}, \boldsymbol{X}_{s_0})(\Delta s)_n + \sigma_{s_n}\sqrt{(\Delta s)_n}\boldsymbol{z}_n$

        $\{\hat{\boldsymbol{X}}_1^{(j)}\}_{j=1}^J \leftarrow$ Posterior estimation ($\hat{\boldsymbol{b}}_s$, s_n, $\boldsymbol{X}_0$, $\boldsymbol{X}_{s_n}$)

        $\{w_j\}_{j=1}^J \leftarrow \text{Softmax}\Big(\{\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2\}_{j=1}^J\Big)$

        $\boldsymbol{X}_{s_{n+1}} = \boldsymbol{X}'_{s_{n+1}} - \zeta_n \nabla_{\boldsymbol{X}_{s_n}} \sum_{j=1}^J w_j \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$

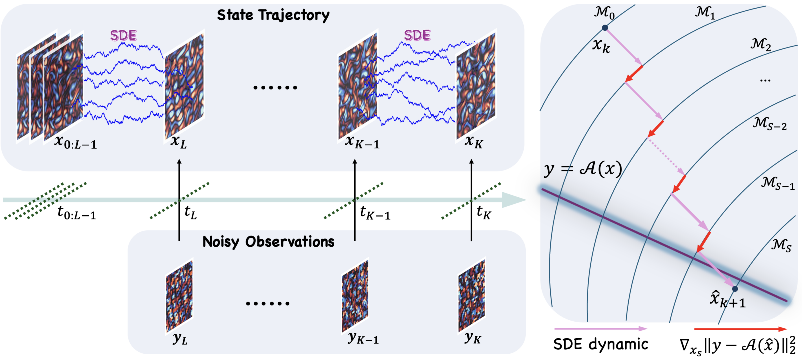
    end for

    $\hat{\boldsymbol{x}}_{k+1} \leftarrow \boldsymbol{X}_{s_N}$

end for

return $\{\hat{\boldsymbol{x}}_k\}_{k=L}^K$

FlowDAS: A Flow-Based Framework for Data Assimilation



- **Physics aligned transport:** directly learns **transition** between adjacent states, enabling faster inference and stable training
- **Observation-consistency:** learns the full **conditional** distribution, with no post-hoc update needed

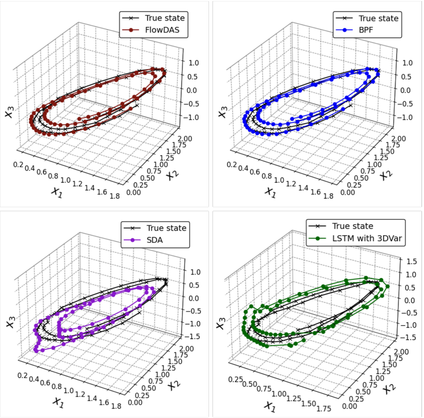
Low-dimensional Problem: Lorenz 1963

- Stuyd the 3D Lorenz 1963 tracking problem, with $\boldsymbol{x}(t) = (x_1(t), x_2(t), x_3(t))$.
- **System dynamics:**

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = \mu(x_2 - x_1) + \xi_1$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = x_1(\rho - x_3) - x_2 + \xi_2$$
$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = x_1 x_2 - \tau x_3 + \xi_3$$

- **Observation model:**

$$y = \arctan(x_1) + \eta$$



		FlowDAS	SDA	BPF
↑	$\log p(\hat{\boldsymbol{x}}_{2:K} \mid \hat{\boldsymbol{x}}_1)$	17.29	-332.7	**17.88**
↑	$\log p(\boldsymbol{y} \mid \hat{\boldsymbol{x}}_{1:K})$	**-0.228**	-6.112	-1.572
↓	$W_1(\boldsymbol{x}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$	**0.106**	0.528	0.812
↓	$\mathrm{RMSE}(\boldsymbol{x}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$	**0.202**	1.114	0.270

Solving Navier-Stokes (NS) Equation

- Consider the **incompressible fiuld flow** governed by 2D NS equations. Let $x = \omega$ to be the *vorticity field*.
- **State transition dynamics:**

$$\mathrm{d}\boldsymbol{\omega} + \boldsymbol{v} \cdot \nabla \boldsymbol{\omega}\, \mathrm{d}t = \nu \Delta \boldsymbol{\omega}\, \mathrm{d}t - \alpha \boldsymbol{\omega}\, \mathrm{d}t + \epsilon\, \mathrm{d}\boldsymbol{\xi}.$$

- **Observation model:** $\boldsymbol{y} = \mathcal{A}(\boldsymbol{\omega}) + \boldsymbol{\eta}$, where $\mathcal{A}$ could be downsampling operator or random mask.

Solving Navier-Stokes (NS) Equation

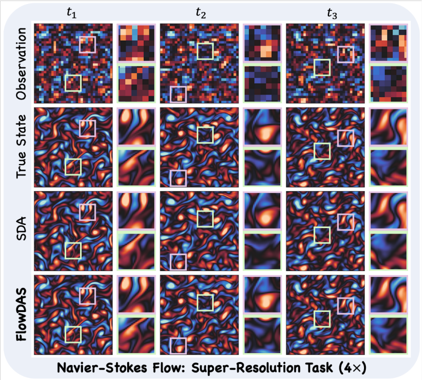
	$32^2 \to 128^2$	$16^2 \to 128^2$	5%	1.5625%
FLOWDAS	**0.038**	**0.067**	**0.071**	**0.123**
FNO-DA	0.158	0.166	0.165	0.183
TRANSOLVER-DA	0.159	0.176	0.161	0.180
SDA	0.073	0.133	0.251	0.258

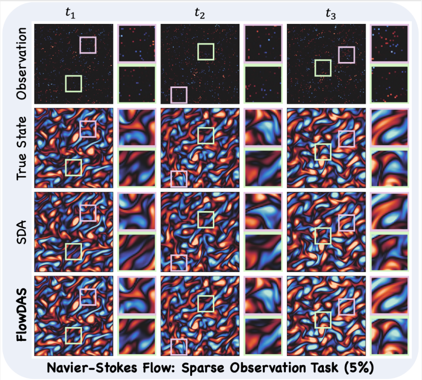
- Consider the **incompressible fiuld flow** governed by 2D NS equations. Let $x = \omega$ to be the *vorticity field*.
- **State transition dynamics:**

$$\mathrm{d}\boldsymbol{\omega} + \boldsymbol{v} \cdot \nabla \boldsymbol{\omega} \, \mathrm{d}t = \nu \Delta \boldsymbol{\omega} \, \mathrm{d}t - \alpha \boldsymbol{\omega} \, \mathrm{d}t + \epsilon \, \mathrm{d}\boldsymbol{\xi}.$$

- **Observation model:** $y = \mathcal{A}(\boldsymbol{\omega}) + \boldsymbol{\eta}$, where $\mathcal{A}$ could be downsampling operator or random mask.

Solving Navier-Stokes (NS) Equation





- **State transition dynamics:**

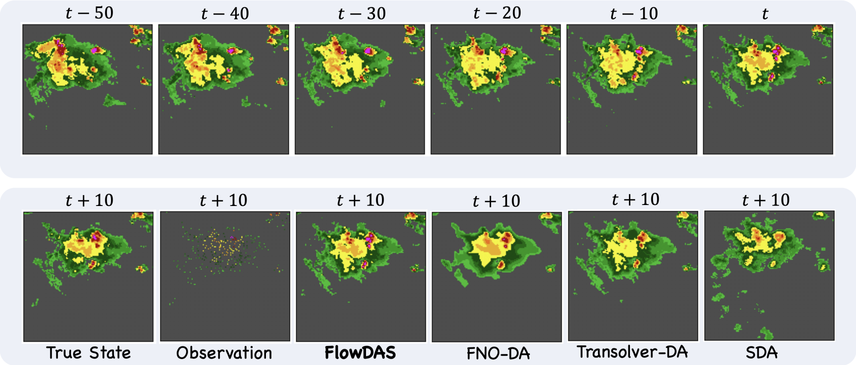
$$\mathrm{d}\omega + \boldsymbol{v} \cdot \nabla\omega \, \mathrm{d}t = \nu\Delta\omega \, \mathrm{d}t - \alpha\omega \, \mathrm{d}t + \epsilon \, \mathrm{d}\boldsymbol{\xi}.$$

- **Observation model:** $y = \mathcal{A}(\omega) + \eta$, where $\mathcal{A}$ could be downsampling operator or random mask.

Application I: Weather Forecasting on SEVIR Dataset

- **Storm Event ImageRy (SEVIR)** is a spatiotemporal Earth observation dataset which consists of 384 km $\times$ 384 km image sequences spanning over 4 hours.
- **Task:** predict the future Vertically Integrated Liquid (VIL) given of pervious context VIL and some sparse observation (10%)

Application I: Weather Forecasting on SEVIR Dataset



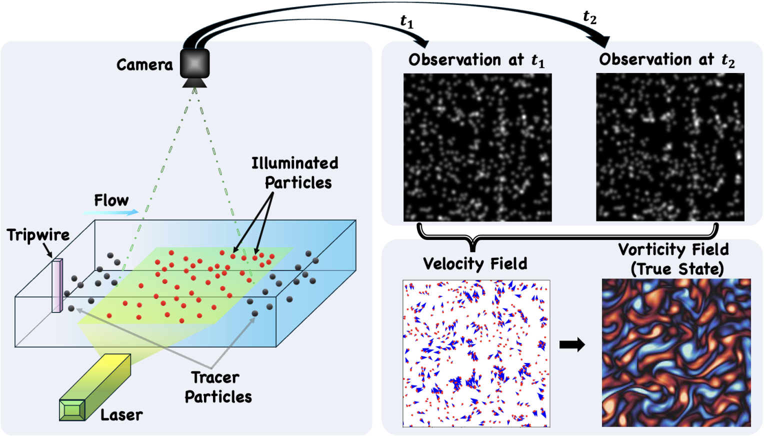
- **Storm Event ImageRy (SEVIR)** is a spatiotemporal Earth observation dataset which consists of 384 km $\times$ 384 km image sequences spanning over 4 hours.
- **Task:** predict the future Vertically Integrated Liquid (VIL) given of pervious context VIL and some sparse observation (10%)

Application I: Weather Forecasting on SEVIR Dataset

Method	**RMSE** $\downarrow$	**CSI(τ_{20}) (0.3)** $\uparrow$	**CSI(τ_{40}) (0.5)** $\uparrow$
Transolver-DA	0.062±0.001	0.663±0.001	0.499±0.002
FNO-DA	0.064±0.001	0.641±0.001	0.493±0.002
SDA	0.071±0.007	0.549±0.033	0.387±0.065
FlowDAS	**0.053±0.004**	**0.746±0.022**	**0.614±0.044**

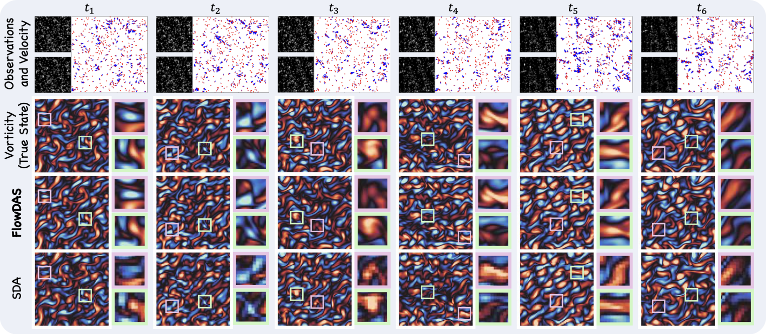
- **Storm Event ImageRy (SEVIR)** is a spatiotemporal Earth observation dataset which consists of 384 km $\times$ 384 km image sequences spanning over 4 hours.
- **Task:** predict the future Vertically Integrated Liquid (VIL) given of pervious context VIL and some sparse observation (10%)

Application II: Particle Image Velocimetry (PIV)



- For many scientific applications, PIV aims at measuring dense vorticity fields from sparse velocity measurements.
- Here, the fluid flow is seeded with tracer particles are captured by a camera to derive the sparse velocity field.

Application II: Particle Image Velocimetry (PIV)



- For many scientific applications, PIV aims at measuring dense vorticity fields from sparse velocity measurements.
- Here, the fluid flow is seeded with tracer particles, which are captured by a camera to derive the sparse velocity field.

Discussion

- **Method:** We introduce FlowDAS, a flow-based autoregressive method for solving the data assimilation problem with many scientific applications.
- **Significance:** the first DA framework built on stochastic interpolants that learns step-to-step state transitions and conditions on observations during rollout.
- **Future directions:** multimodality and multiphyics, long-term prediction, efficiency, and robustness.

2. Siyi Chen*, Yixuan Jia*, Qing Qu, He Sun, Jeffrey Fessler. FlowDAS: A Stochastic Interpolant-based Framework for Data Assimilation. *Neural Information Processing Systems (NeurIPS'25)*, 2025.

Conclusion & Acknowledgement

Take-Home Message

- **Solving image reconstruction problems:** We developed an efficient method for solving image inverse problems through decoupling via diffusion purification.
- **Data assimilation:** We explored stochastic interpolant methods for data assimilation in scientific applications, by better modeling the underlying physics.

Major References

1. Xiang Li, Soo Min Kwon, Ismail R. Alkhouri, Saiprasad Ravishankar, Qing Qu. Decoupled Data Consistency with Diffusion Purification for Image Restoration. *Under Review at IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, 2025.
2. Siyi Chen*, Yixuan Jia*, Qing Qu, He Sun, Jeffrey Fessler. FlowDAS: A Stochastic Interpolant-based Framework for Data Assimilation. *Neural Information Processing Systems (NeurIPS'25)*, 2025.
3. Ismail Alkhouri, Shijun Liang, Cheng-Han Huang, Jimmy Dai, Qing Qu, Saiprasad Ravishankar, Rongrong Wang. SITCOM: Step-wise Triple-Consistent Diffusion Sampling for Inverse Problems. *International Conference on Machine Learning (ICML'25)*, 2025.
4. Bowen Song*, Soo Min Kwon*, Zecheng Zhang, Xinyu Hu, Qing Qu, Liyue Shen. Solving Inverse Problems with Latent Diffusion Models via Hard Data Consistency. *International Conference on Learning Representations (ICLR'24)*, 2024. (**spotlight, top 5%**)

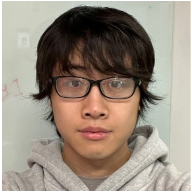
Acknowledgement

Siyi Chen
(UMich)

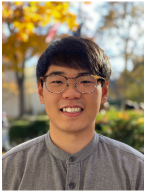
Yixuan Jia
(UMich)

He Sun
(Peking U)

Jeff Fessler
(UMich)



Xiang Li
(UMich)



Soo Min Kwon
(UMich)

Ismail Alkhouri
(DARPA)

Sai
Ravishankar
(MSU)

Acknowledgement

MICHIGAN INSTITUTE FOR COMPUTATIONAL DISCOVERY & ENGINEERING
UNIVERSITY OF MICHIGAN

National Science Foundation

amazon

DARPA

Office of Naval Research
Science & Technology

Google

Thank You!