FlowDAS: A Flow-Based Framework for Data Assimilation

Siyi Chen*[◇,†], Yixuan Jia*[◇], Qing Qu[◇], He Sun[†,§], and Jeffrey A. Fessler[◇]

[◇]Department of Electrical Engineering and Computer Science, University of Michigan
[†]Department of Physics, Peking University
[§]National Biomedical Imaging Center, Peking University

January 13, 2025

Abstract

Data assimilation (DA) is crucial for improving the accuracy of state estimation in complex dynamical systems by integrating observational data with physical models. Traditional solutions rely on either pure model-driven approaches, such as Bayesian filters that struggle with nonlinearity, or data-driven methods using deep learning priors, which often lack generalizability and physical interpretability. Recently, score-based DA methods have been introduced, focusing on learning prior distributions but neglecting explicit state transition dynamics, leading to limited accuracy improvements. To tackle the challenge, we introduce **FlowDAS**, a novel generative model-based framework using the stochastic interpolants to unify the learning of state transition dynamics and generative priors. FlowDAS achieves stable and observation-consistent inference by initializing from proximal previous states, mitigating the instability seen in score-based methods. Our extensive experiments demonstrate FlowDAS's superior performance on various benchmarks, from the Lorenz system to high-dimensional fluid super-resolution tasks. FlowDAS also demonstrates improved tracking accuracy on practical Particle Image Velocimetry (PIV) task, showcasing its effectiveness in complex flow field reconstruction.

Contents

*The first two authors contributed equally to the work. This work was completed while Siyi Chen was an intern at the University of Michigan.

†Corresponding author: hesun@pku.edu.cn

1 Introduction

Recovering state variables in complex dynamical systems is a fundamental problem in many scientific and engineering domains. Accurate state estimation from noisy and incomplete data is essential in applications like climate modeling, weather forecasting, and seismology to understand underlying physical processes and make reliable predictions. For example, in fluid dynamics, we want to recover a continuous velocity field from sparse, noisy observations, where the task is governed by nonlinear, time-dependent partial differential equations (PDEs). The problem is challenging to solve due to the inherent stochasticity and high dimensionality of the processes. Mathematically, a discrete-time stochastic dynamical system can be characterized by the state-space model:

$$\boldsymbol{x}_{k+1} = \Psi(\boldsymbol{x}_k) + \boldsymbol{\xi}_k, \tag{1}$$
$$\boldsymbol{y}_{k+1} = \mathcal{A}(\boldsymbol{x}_{k+1}) + \boldsymbol{\eta}_{k+1}, \tag{2}$$

where $\boldsymbol{x}_k \in \mathbb{R}^D$ is the state vector at time step k, $\Psi(\cdot)$ is the state transition map, and $\boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_D)$ denotes Gaussian noise. The observations $\boldsymbol{y}_k \in \mathbb{R}^M$ are related to the state through the measurement map $\mathcal{A}(\cdot)$, with observation noise $\boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \gamma^2 \boldsymbol{I}_M)$. Our goal is to estimate the posterior of the state trajectory $\boldsymbol{x}_{1:K}$ given observations $\boldsymbol{y}_{1:K}$ and the initial state $\boldsymbol{x}_0$, i.e., $p(\boldsymbol{x}_{1:K} \mid \boldsymbol{y}_{1:K}, \boldsymbol{x}_0)$, as shown in Figure 1.

This estimation problem, commonly referred to as **Data Assimilation** (DA) [Bouttier and Courtier, 2002, Lahoz and Menard, 2010, Law et al., 2015], originated in atmospheric [Bocquet et al., 2015, Reichle, 2008, Wang et al., 2000] and oceanic forecasting [Bertino et al., 2003, Carton et al., 2000, Cummings, 2005, Cummings and Smedstad, 2013, Derber and Rosati, 1989], and has recently found numerous applications across diverse scientific and engineering domains [Carrassi et al., 2018, Fletcher, 2017, Geer et al., 2018, Gustafsson et al., 2018, Rabier, 2005, Rodell et al., 2004, Van Leeuwen, 2010]. DA combines observations with model predictions to provide more accurate and physically consistent state estimates. Despite significant advancements, existing DA methods still face significant challenges due to high nonlinearity and high dimensionality of the dynamic processes.

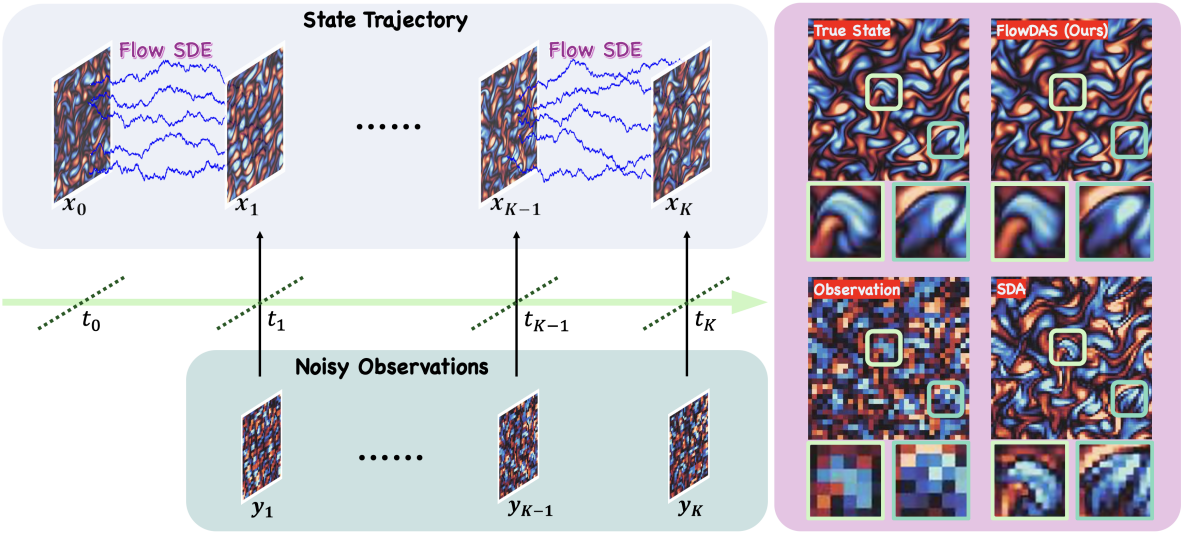


Figure 1: **An overview of FlowDAS.** We introduce a flow-based framework for data assimilation, named FlowDAS, to estimate the state trajectory $\boldsymbol{x}_{0:K}$ from the noisy observations $\boldsymbol{y}_{1:K}$. Our framework uses a flow-based stochastic differential equation to model the stochastic dynamics of the system and leverages the observations to improve the prediction accuracy. On the right, we show a comparison of Navier-Stokes flow results between FlowDAS and the Score-based Data Assimilation [Rozet and Louppe, 2023] method. FlowDAS produces more accurate prediction compared to the score-based method, demonstrating its strong capability for data assimilation of high-dimensional complex systems.

Existing DA techniques can be divided into model-driven and data-driven approaches. Model-driven methods, like Bayesian filters, rely on explicit physical models to describe dynamics for estimating the states. For example, Kalman filters and their variants (e.g., extended, unscented, and ensemble Kalman filters) [Evensen, 2003] assume (quasi-)linear state transitions and observations, while particle filters [Gordon et al., 1993] offer more flexibility by using Monte Carlo sampling for Bayesian estimation. However, model-driven methods face the curse of dimensionality in real-world applications [Bickel et al., 2008], because they typically require costly numerical solvers for explicit physical modeling. In contrast, data-driven approaches leverage machine learning methods to directly learn state dynamics from observational data, bypassing the usage of explicit physical models. For example, recent developments in score-based diffusion models demonstrate promise for capturing complex state transitions by learning from sequences of dynamic states [Qu et al., 2024, Rozet and Louppe, 2023]. These generative models approximate the joint distribution of entire state sequences, akin to popular video generation models, and can be employed as generative priors within a Bayesian framework to support DA. However, these models often encode dynamics in latent spaces that lack accuracy and physical interpretability, leading to instability during inference and yielding biased state estimates.

To overcome these limitations, we propose **FlowDAS**: a flow-based framework for data assimilation. FlowDAS utilizes stochastic interpolants [Albergo et al., 2023], a new class of flow-based generative models, to learn neural surrogates of complex system dynamics. Stochastic interpolant models employ SDEs to create interpolative pathways between arbitrary data distributions, providing flexible control over transition dynamics for smooth, stable generation that is well-suited for capturing nonlinear dynamics in DA tasks. This learned flow model is then integrated with obser-

vational data to enable accurate and efficient state estimation. Extensive experiments demonstrate that FlowDAS achieves state-of-the-art performance across diverse scientific applications, including the chaotic Lorenz system and Navier-Stokes equations in fluid mechanics. We further validate our approach in a practical DA application, Particle Image Velocimetry (PIV) [Raffel et al., 2018], where it effectively estimates dense fluid velocity maps from sparse particle motion observations.

2 Related Work

2.1 Existing Data Assimilation Approaches

DA approaches can be broadly categorized into model-driven and data-driven methods, with particle filters and score-based data assimilation (SDA) as representative approaches in each type.

Particle filters Gordon et al. [1993] are a class of sequential Monte Carlo (SMC) methods commonly used for DA in non-linear, non-Gaussian dynamical systems. They represent the state distribution as a set of weighted particles, each acting as a hypothesis for the state of the system. As new observations are received, particle states are updated on the basis of the dynamics of the system in Equation (1), and the particle weights are adjusted according to the likelihood of each state given the observation. Low-weight particles are periodically resampled to preserve computational resources on the most probable states. Particle filters are highly flexible, able to model complex distributions without requiring linearity or Gaussian assumptions. However, they face significant computational challenges in high-dimensional settings due to the curse of dimensionality [Bickel et al., 2008], limiting their scalability in large physical systems.

Score-based data assimilation (SDA) [Rozet and Louppe, 2023] is a recent data-driven approach that employs score-based diffusion models to estimate state trajectories in dynamical systems. It bypasses explicit physical modeling by learning the joint distribution of short state trajectory segments (e.g., $2k+1$ time steps) via a score network $\boldsymbol{s}_\theta(\boldsymbol{x}_{i-k:i+k})$. By integrating the score network with the observation model in a diffusion posterior sampling (DPS) framework [Chung et al., 2022], SDA can generate entire state trajectories in zero-shot and non-autoregressive manners. This approach is computationally efficient for high-dimensional systems, as generative priors on short segments allow for parallel inference, making it particularly useful for systems with sparse or limited observations. Despite these advantages, because the latent dynamics may fail to capture real-world physics, the learned dynamics of SDA may lack physical interpretability, leading to potential inaccuracies. SDA also requires substantial training data, and its posterior approximations may be less reliable in systems where the physical model is highly sensitive, as illustrated by the double well potential problem in Appendix B.2.

2.2 Stochastic Interpolants

Stochastic interpolants [Albergo and Vanden-Eijnden, 2022, Albergo et al., 2023, Chen et al., 2024b] is a recent generative modeling framework that unifies flow-based and diffusion-based models. It provides a smooth, controlled transition between arbitrary probability densities over a finite time horizon.

Consider a stochastic process $\boldsymbol{X}_s$ defined over the interval $s \in [0, 1]$, which evolves from an initial state $\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)$ to the final state $\boldsymbol{X}_1 \sim q(\boldsymbol{X}_1)$. This process defines a smooth path from

the base distribution to the target distribution, facilitating generative modeling. A stochastic interpolant can be described as [Chen et al., 2024b]:

$$\tilde{\boldsymbol{I}}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \boldsymbol{X}_1 + \sigma_s \boldsymbol{W}_s, \tag{3}$$

where $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0, \boldsymbol{X}_1)$. $\boldsymbol{W}_s$ is a Wiener process for $s \in [0,1]$ introduced after $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$ are sampled, ensuring that $\boldsymbol{W}_s$ is independent of $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$. The time-varying coefficients $\alpha_s, \beta_s, \sigma_s \in C^1([0,1])$ satisfy boundary conditions $\alpha_0 = \beta_1 = 1$ and $\alpha_1 = \beta_0 = \sigma_1 = 0$. A typical choice for these coefficients is $\alpha_s = 1 - s$, $\sigma_s = 1 - s$, and $\beta_s = s^2$, ensuring that $\tilde{\boldsymbol{I}}_0 = \boldsymbol{X}_0$ and $\tilde{\boldsymbol{I}}_1 = \boldsymbol{X}_1$, thereby creating a smooth interpolation from $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. Moreover, Chen et al. [2024b] further showed that, for all $(s, \boldsymbol{X}_0) \in [0,1] \times \mathbb{R}^D$, $\tilde{\boldsymbol{I}}_s \mid \boldsymbol{X}_0$ has the same distribution as $\boldsymbol{X}_s$, which is the solution to the following SDE:

$$\mathrm{d}\boldsymbol{X}_s = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)\,\mathrm{d}s + \sigma_s\,\mathrm{d}\boldsymbol{W}_s, \tag{4}$$

where the drift term $\boldsymbol{b}_s(\boldsymbol{X}, \boldsymbol{X}_0)$ is optimized by minimizing the cost function:

$$\mathcal{L}_b(\hat{\boldsymbol{b}}_s) = \int_0^1 \mathbb{E}\left[\|\hat{\boldsymbol{b}}_s(\tilde{\boldsymbol{I}}_s, \boldsymbol{X}_0) - \boldsymbol{R}_s\|^2\right] \mathrm{d}s. \tag{5}$$

The "velocity" of the interpolant path, $\boldsymbol{R}_s$, is given by:

$$\boldsymbol{R}_s = \dot{\alpha}_s \boldsymbol{X}_0 + \dot{\beta}_s \boldsymbol{X}_1 + \dot{\sigma}_s \boldsymbol{W}_s. \tag{6}$$

This formulation captures the instantaneous rate of change along the interpolation path, ensuring smooth transitions that are well-suited for modeling complex distributions.

Furthermore, the score function $\nabla_{\boldsymbol{X}_s} \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)$ for this SDE can be expressed as:

$$\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0) = \lambda_s \left[\beta_s \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) - \boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)\right], \tag{7}$$

where λ_s and $\boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ are defined to control the score-based dynamics.

Stochastic interpolants provides a robust method for estimating the conditional distribution $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$, which offers a powerful surrogate state transition model. In DA, given the current state $\boldsymbol{x}_k = \boldsymbol{X}_0$, this framework can evolve the process $\boldsymbol{X}_s$ smoothly from $s = 0$ to $s = 1$, to capture the transition dynamics to $\boldsymbol{x}_{k+1}$. However, achieving observation-consistent predictions requires conditioning these transitions on observational data. The next section introduces our approach, which leverages stochastic interpolants for observation-informed state prediction, enabling more accurate and reliable DA.

3 Methods

This section presents our approach, which extends the original stochastic interpolants framework to incorporate observational data for effective DA. Inspired by DPS [Chung et al., 2022] and related works [Alkhouri et al., 2024, Li et al., 2024, Song et al., 2024], our method adapts the score function into a conditional form, aligning data-driven state transition modeling with observed data to enable accurate, observation-consistent state estimation in complex dynamical systems.

Algorithm 1 Training

Input: Dataset $\boldsymbol{x}_{0:K}$; minibatch size $K' \leq K$; coefficients α_s, β_s, σ_s
repeat
  Compute $\tilde{\boldsymbol{I}}_s^k$ and $\boldsymbol{R}_s^k$ using (14) for $k \in \mathcal{B}_{K'}$
  Compute the empirical loss $\mathcal{L}_b^{\text{emp}}(\hat{\boldsymbol{b}})$ in Equation (13)
  Take the gradient step on $\mathcal{L}_b^{\text{emp}}(\hat{\boldsymbol{b}})$ to update $\hat{\boldsymbol{b}}_s$
until converged
return drifts $\hat{\boldsymbol{b}}_s$

3.1 Stochastic Interpolants for Data Assimilation

Conditional state generation Stochastic interpolants approximate the state transition $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ by interpolating between the state variables $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ using the SDE defined in Equation (4) and Equation (7) with boundary conditions $\boldsymbol{X}_0, \boldsymbol{X}_1 = \boldsymbol{x}_k, \boldsymbol{x}_{k+1}$. The drift term $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ is related to the score function $\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)$ of state transition distribution:

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) = \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)}{\lambda_s \beta_s}. \tag{8}$$

To generate observation-consistent states, we modify the SDE to a process conditioned on observational data, where the drift term $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0)$ incorporates observation information via Bayes' rule:

$$\begin{aligned}
\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) =& \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{y}, \boldsymbol{X}_0)}{\lambda_s \beta_s} \\
=& \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) + \nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)}{\lambda_s \beta_s} \\
=& \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}.
\end{aligned} \tag{9}$$

Estimation of $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ The term $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ captures the observation information. Since the observation model only directly links $\boldsymbol{y} = \boldsymbol{y}_{k+1}$ and $\boldsymbol{X}_1 = \boldsymbol{x}_{k+1}$, we compute this term by integrating with respect to $\boldsymbol{X}_1$:

$$\begin{aligned}
\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) = \frac{\nabla\, p(\boldsymbol{y} | \boldsymbol{X}_s, \boldsymbol{X}_0)}{p(\boldsymbol{y} | \boldsymbol{X}_s, \boldsymbol{X}_0)} &= \frac{\nabla \int p(\boldsymbol{y} | \boldsymbol{X}_1) p(\boldsymbol{X}_1 | \boldsymbol{X}_s, \boldsymbol{X}_0)\, \mathrm{d}\boldsymbol{X}_1}{\int p(\boldsymbol{y} | \boldsymbol{X}_1) p(\boldsymbol{X}_1 | \boldsymbol{X}_s, \boldsymbol{X}_0)\, \mathrm{d}\boldsymbol{X}_1} \\
&= \frac{\nabla\, \mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1 | \boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y} | \boldsymbol{X}_1)]}{\mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1 | \boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y} | \boldsymbol{X}_1)]}.
\end{aligned} \tag{10}$$

In practice, we approximate the above expectations by J Monte Carlo samples, $\boldsymbol{X}_1^{(j)} \sim p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$:

$$\begin{aligned}
\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) \approx \frac{\nabla\, \frac{1}{J} \sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})}{\frac{1}{J} \sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})} &= \frac{\sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})\, \nabla\, \log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})}{\sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})} \\
&= \sum_{j=1}^J w_j \nabla\, \log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}),
\end{aligned} \tag{11}$$

Algorithm 2 Inference

Input: Observation $\boldsymbol{y}_{1:K}$, 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 $\hat{\boldsymbol{x}}_0 \leftarrow \boldsymbol{x}_0$
Set the $(\Delta s)_n = s_{n+1} - s_n,\ n = 0 : N-1$
for $k = 0$ **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 $(\hat{\boldsymbol{b}}_s, s_n, \boldsymbol{X}_0, \boldsymbol{X}_{s_n})$ by Equation (12)
    $\{w_j\}_{j=1}^J \leftarrow \mathrm{Softmax}\Big(\{\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2\}_{j=1}^J\Big)$
    $\boldsymbol{X}_{s_{n+1}} = \boldsymbol{X}'_{s_{n+1}} - \zeta_n \nabla_{\boldsymbol{X}_{s_n}} \sum_{j=1}^J w_j \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$
  end for
  $\hat{\boldsymbol{x}}_{k+1} \leftarrow \boldsymbol{X}_{s_N}$
end for
return $\{\hat{\boldsymbol{x}}_k\}_{k=1}^K$

where we apply a softmax function to the J scalars $\{\log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})\}_{j=1}^J$ to compute the sample weights $w_j = p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}) / \sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})$.

Acceleration of Monte Carlo sampling Accurate sampling from $p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ requires solving the SDE in Equation (4), which can be computationally intensive. To improve efficiency, we propose using approximate numerical integration methods such as Euler's method [Süli and Mayers, 2003, p. 317] or Heun's method [Süli and Mayers, 2003, p. 324]:

$$\begin{aligned} \text{Euler: } \hat{\boldsymbol{X}}_1 &= \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)(1-s) + \int \sigma_s \, \mathrm{d}\boldsymbol{W}_s, \\ \text{Heun: } \hat{\boldsymbol{X}}'_1 &= \frac{\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \boldsymbol{b}_1(\hat{\boldsymbol{X}}_1, \boldsymbol{X}_0)}{2}(1-s) + \int \sigma_s \, \mathrm{d}\boldsymbol{W}_s. \end{aligned} \tag{12}$$

These methods offer first-order and second-order approximations, respectively, introducing slight numerical bias but significantly accelerated sampling speed (i.e., $\mathcal{O}((1-s)^2)$ for Euler and $\mathcal{O}((1-s)^3)$ for Heun). In practice, Heun's method consistently demonstrated better performance than Euler's method. Appendix A.2 further compares these approximations.

3.2 Implementation Details

Training We used a dataset consisting of multiple simulated state trajectories $\boldsymbol{x}_{0:K}$ to train the drift function $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ in stochastic interpolants (see Appendix B.1 for more details). We approximate the cost function in Equation (5) by the empirical loss:

$$\mathcal{L}_b^{\text{emp}}(\hat{\boldsymbol{b}}) = \frac{1}{K'} \sum_{k \in \mathcal{B}_{K'}} \int_0^1 \|\hat{\boldsymbol{b}}_s(\tilde{\boldsymbol{I}}_s^k, \boldsymbol{x}_k) - \boldsymbol{R}_s^k\|^2 \, \mathrm{d}s, \tag{13}$$

where $K' \leq K$ and $\mathcal{B}_{K'} \subset \{0:K\}$ is a subset of indices of cardinality K', with

$$\begin{aligned} \tilde{\boldsymbol{I}}_s^k &= \alpha_s \boldsymbol{x}_k + \beta_s \boldsymbol{x}_{k+1} + \sqrt{s}\sigma_s \boldsymbol{z}_k \\ \boldsymbol{R}_s^k &= \dot{\alpha}_s \boldsymbol{x}_k + \dot{\beta}_s \boldsymbol{x}_{k+1} + \sqrt{s}\dot{\sigma}_s \boldsymbol{z}_k, \end{aligned} \tag{14}$$

where $\boldsymbol{z}_k \sim \mathcal{N}(0, \boldsymbol{I}_D)$ and satisfy $\boldsymbol{W}_s \overset{d}{=} \sqrt{s}\boldsymbol{z}$ with $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_D)$ for all $s \in [0,1]$. We approximate the integral over s in Equation (13) via an empirical expectation sampling from $s \sim U([0,1])$. Algorithm 1 provides a detailed description of the model training process.

Inference Our inference procedure generates trajectories conditioned on observations $\boldsymbol{y}_{1:K}$ using the learned drift model $\hat{\boldsymbol{b}}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$. We start by setting a specific state $\boldsymbol{x}_k$ as initial $\boldsymbol{X}_0$ and iterate over a predefined temporal grid $s_0 = 0 < s_1 < \cdots < s_N = 1$. Within each iteration, we first compute posterior estimates $\{\hat{\boldsymbol{X}}_1^{(j)}\}_{j=1}^J$ using Equation (12) for J times. Then, we move one step further towards s_N on $\boldsymbol{X}_{s_n}$ by solving Equation (9) which involves backpropagating the gradient $\nabla_{\boldsymbol{X}_{s_n}} \sum_{j=1}^J w_j \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$ to enforce consistency with observations $\boldsymbol{y}_k$. We iterated this process autoregressively, by setting $\boldsymbol{X}_{s_0}^{k+1} = \boldsymbol{X}_{s_N}^k$. Empirically, we found that using a constant step size ζ_n across the inference process produced generally good results, although fine-tuning ζ_n at each step can slightly improve the performance [Bai et al., 2024, Chung et al., 2022]. The chosen J values, reported in Table S.6, were sufficient large for stable performance, with subtle variation across the dimensionality of problems. An ablation study, detailed in Appendix A.1, further confirmed that larger J offers diminishing returns. Overall, Algorithm 2 summarizes the inference process.

4 Experiments and Results

This section evaluates our proposed framework, FlowDAS, on a range of tasks with a consistent parameter setting ($\alpha_s = 1 - s, \sigma_s = 1 - s, \beta_s = s^2$). We test FlowDAS on both low- and high-dimensional stochastic dynamical systems, including low-dimensional problems with high-order observation models, such as the double-well potential (Appendix B.2.5) and the chaotic Lorenz 1963 system, as well as high-dimensional tasks involving the incompressible Navier-Stokes equations. Additionally, we demonstrate its practical applicability in a realistic Particle Image Velocimetry (PIV) simulation. These results underscore the versatility and robustness of FlowDAS.

4.1 Lorenz 1963

In this experiment, we evaluate the performance of our FlowDAS framework using the Lorenz 1963 system, a widely studied benchmark in the DA community [Bao et al., 2023, Lorenz, 1963, Rozet and Louppe, 2023]. The state vector of the Lorenz system, $\boldsymbol{x} = (a, b, c) \in \mathbb{R}^3$, evolves according to the following nonlinear stochastic ordinary differential equations (ODEs):

$$\begin{aligned} \mathrm{d}a/\,\mathrm{d}t &= \mu(b - a) + \xi_1, \\ \mathrm{d}b/\,\mathrm{d}t &= a(\rho - c) - b + \xi_2, \\ \mathrm{d}c/\,\mathrm{d}t &= ab - \tau c + \xi_3, \end{aligned} \tag{15}$$

where $\mu = 10$, $\rho = 28$, and $\tau = 8/3$ define the ODE parameters, and $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3$ is the process noise, with each component having a standard deviation $\sigma = 0.25$. This chaotic system poses a significant challenge for numerical methods, so we use the fourth-order Runge-Kutta (RK4) method [Noar et al., 2024] to simulate its state transition (see Appendix B.3 for details).

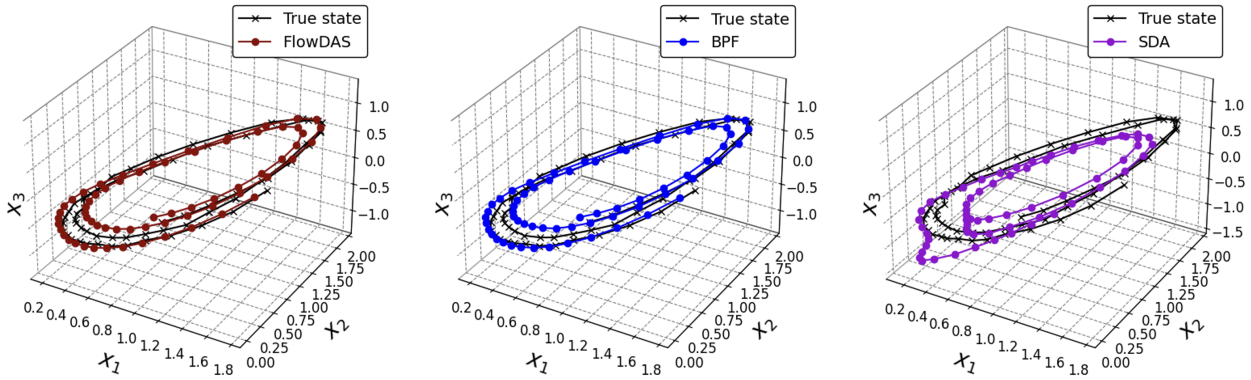


Figure 2: **Data assimilation of Lorenz 1963 system.** FlowDAS achieved results comparable to the state-of-the-art model-based BPF method, significantly outperforming the data-driven SDA method in recovering the underlying dynamics of this chaotic system. This highlights the efficiency and robustness of FlowDAS in capturing complex, nonlinear dynamics while maintaining accuracy and stability. The variables x_1, x_2 and x_3 correspond to a, b and c in the ODEs of the Lorenz system Equation (15), respectively.

We observe only the arctangent-transformed value of the first state component a, so the observation model of the system is defined as

$$y = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\eta} = \arctan(a) + \eta, \tag{16}$$

wehre η is the observation noise with a standard deviation $\gamma = 0.25$.

Dataset We generate 1,024 independent trajectories, each containing 1,024 states, and split the data into training (80%), validation (10%), and evaluation (10%) sets. Initial states are sampled from the statistically stationary regime of the Lorenz system, with additional data generation details provided in Appendix B.3. For this low-dimensional problem, we use a fully connected neural network to approximate the drift term in stochastic interpolants; the network architecture is also described in the same appendix.

Baselines and metrics We compare our method against two baselines: the SDA solver with a fixed window size of 2 and the classic bootstrap particle filter (BPF) [Gordon et al., 1993]. Appendix B.3 details the score network architecture for SDA and particle density settings for BPF.

We evaluate the performance of FlowDAS and baselines using four metrics: the expectation of log-prior $\mathbb{E}_{q(\boldsymbol{x}_{1:K}|\boldsymbol{y}_{1:K})}\left[\log p(\boldsymbol{x}_{2:K} \mid \boldsymbol{x}_1)\right]$; the expectation of log data likelihood $\mathbb{E}_{q(\boldsymbol{x}_{1:K}|\boldsymbol{y}_{1:K})}\left[\log p(\boldsymbol{y}_{1:K} \mid \boldsymbol{x}_{1:K})\right]$; the Wasserstein distance [Villani, 2009] $W_1(\cdot,\cdot)$ between the true trajectory $\boldsymbol{x}_{1:K}$ and the estimated trajectory $\hat{\boldsymbol{x}}_{1:K}$; and the RMSE between the true and estimated states.

Results We independently estimate 64 trajectories over 15 time steps using FlowDAS and the baseline methods. As shown in Table 1 and Figure 2, FlowDAS outperforms SDA across all metrics. FlowDAS is only slightly less effective than BPF in the expected log-prior, as BPF directly incorporates the true system dynamics into its state estimation.

The success of FlowDAS is primarily due to its accurate mapping from current to future states ($\boldsymbol{x}_k \to \boldsymbol{x}_{k+1}$). Despite lacking explicit transition equations, FlowDAS effectively captures the system dynamics through stochastic interpolants, enabling a closer approximation of state trajectories

		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
↓	$\text{RMSE}(\boldsymbol{x}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$	**0.202**	1.114	0.270

Table 1: **Lorenz tracking results.** This table summarizes the performance of FlowDAS, SDA, and BPF on the Lorenz 1963 experiment over 15 time steps. FlowDAS outperforms SDA across all evaluation metrics and is competitive with BPF, despite BPF utilizing the true transition equations, which are unknown to FlowDAS and SDA. The best results for each metric are highlighted in **bold**.

compared to SDA, which models joint distributions across sequential states using diffusion models. Additionally, stochastic interpolants allow FlowDAS to produce accurate state estimates while managing inherent variability, avoiding over-concentration on high-probability regions, and effectively dealing with rare events. This advantage is further illustrated in the double well potential experiment (Appendix B.2.5) where FlowDAS outperforms BPF, because BPF tends to be trapped by high-probability point estimates.

4.2 Incompressible Navier-Stokes Flow

This section considers a high-dimensional dynamical system: incompressible fluid flow governed by the 2D Navier-Stokes (NS) equations with random forcing on the torus $\mathbb{T}^2 = [0, 2\pi]^2$. The state transition, Ψ, is described using the stream function formulation,

$$\mathrm{d}\boldsymbol{\omega} + \boldsymbol{v} \cdot \nabla \boldsymbol{\omega} \, \mathrm{d}t = \nu \Delta \boldsymbol{\omega} \, \mathrm{d}t - \alpha \boldsymbol{\omega} \, \mathrm{d}t + \varepsilon \, \mathrm{d}\boldsymbol{\xi}, \tag{17}$$

where $\boldsymbol{\omega}$ represents the vorticity field, the state variable in this fluid dynamics system ($\boldsymbol{x} = \boldsymbol{\omega}$). The velocity $\boldsymbol{v} = \nabla^{\perp}\psi = (-\partial_y \psi, \partial_x \psi)$ is expressed in terms of the stream function $\psi(x, y)$, which satisfies $-\Delta\psi = \boldsymbol{\omega}$. The term $d\boldsymbol{\xi}$ represents white-in-time random forcing acting on a few Fourier modes, with parameters $\nu, \alpha, \varepsilon > 0$ specified in Appendix B.4.1.

The observation operator $\mathcal{A}$ linearly downsamples or selects partial pixels from the simulated vorticity fields ($\boldsymbol{\omega}$),

$$y = \mathcal{A}(\boldsymbol{\omega}) + \boldsymbol{\eta}, \tag{18}$$

where the observation noise $\boldsymbol{\eta}$ has a standard deviation of $\gamma = 0.05$.

Dataset In this experiment, system dynamics are simulated by solving Equation (17) using a pseudo-spectral method [Peyret, 2002] with a resolution of 256^2 and a timestep $\Delta t = 10^{-4}$. We simulate 200 flow conditions over $t \in [0, 100]$, saving snapshots of fluid vorticity field ($\boldsymbol{\omega} = \nabla \times v$) at the second half of each trajectory ($t \in [50, 100]$) at intervals of $\Delta t = 0.5$ with a reduced resolution of 128^2. The data are divided into training (80%), validation (10%), and evaluation (10%) sets.

Baselines and metrics We compare our method against a SDA solver with a fixed window size of 2. BPF is not included in our testing, as its particle requirements grow exponentially with system dimensions, making it impractical for high-dimensional fluid dynamics systems. Additional details on the model architecture and training for both SDA and FlowDAS are provided in Appendix B.4.3.

We evaluate performance using the RMSE between the predicted and ground-truth vorticity fields. Additionally, we assess the reconstruction of the kinetic energy spectrum to determine whether

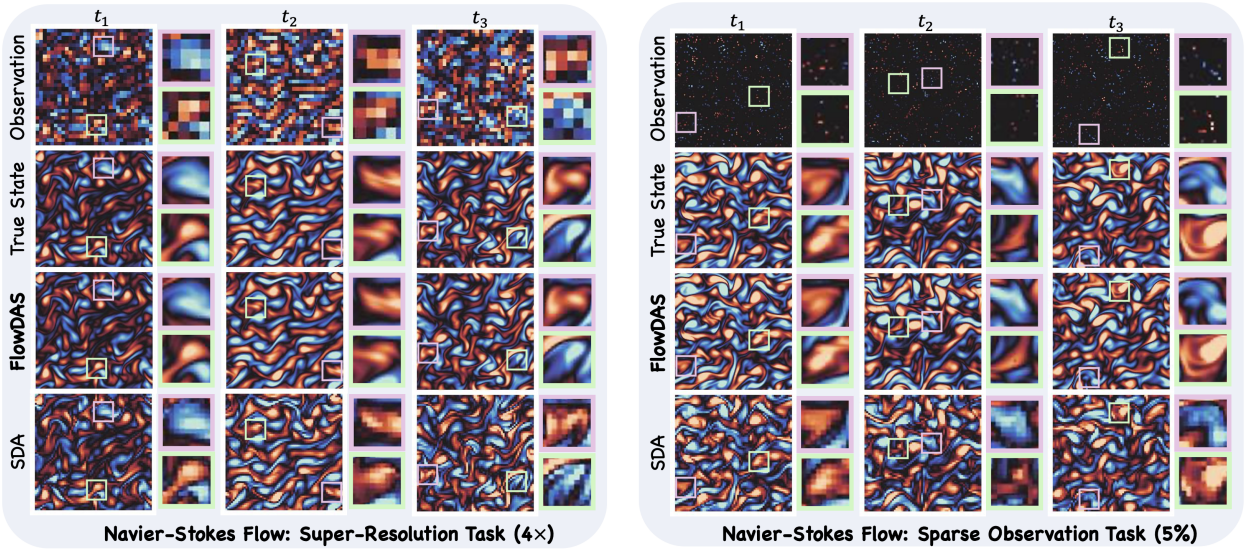


Figure 3: **Data assimilation of incompressible Navier-Stokes flow.** The positive values (red) of the state, i.e., vorticity field, indicate clockwise rotation and negative values (blue) indicate counter-clockwise rotation. FlowDAS achieved results with more details and higher accuracy than the SDA both in the super-resolution task and the sparse observation task, showing FlowDAS's efficiency in tackling DA tasks with highly non-linear complex systems. Additionally, FlowDAS is also better at recovering high-frequency information, evidenced by the spectral analysis in Figure S.9.

the physical characteristics of the fluid are accurately preserved. Appendix B.4.5 and Figure S.9 provide the definitions and results for the kinetic energy spectrum metric.

Results We conduct experiments across different observation resolutions (32^2, 16^2) and observation sparsity levels (5%, 1.5625%). For the super-resolution task, the goal is to reconstruct high-resolution vorticity data (128^2) from low-resolution observations. In the inpainting task, only 5% or 1.5625% of pixel values are retained, with the rest set to zero, and we attempt to recover the complete vorticity field. The model is evaluated on four unseen datasets (2× super-resolution, 2× inpainting), with 64 samples for each configuration. Quantitative metrics, including RMSE and kinetic energy spectrum comparisons, are provided in Table 2 and Figure S.9.

Figure 3 presents a visual comparison between FlowDAS and SDA for the flow super-resolution and inpainting tasks. Our method consistently outperforms SDA in terms of reconstruction accuracy and resolution, capturing high-frequency information with greater precision. This advantage is further validated by the kinetic energy spectrum in Figure S.9. The RMSE scores in Table 2 further highlight the effectiveness of FlowDAS in accurately estimating the underlying fluid dynamics from observational data.

4.3 Particle Image Velocimetry

This section presents a realistic application of our method: Particle Image Velocimetry (PIV). PIV is a widely used optical technique for measuring velocity fields in fluids, with many scientific applications in aerodynamics [Koschatzky et al., 2011, Taylor et al., 2010, Van Oudheusden, 2013], biological flow studies [Ergin et al., 2018, King et al., 2007, Stamhuis, 2006], and medical research

	$32^2 \to 128^2$	$16^2 \to 128^2$	5%	1.5625%
FLOWDAS	**0.038**	**0.067**	**0.071**	**0.123**
SDA	0.073	0.133	0.251	0.258

Table 2: **RMSE of FlowDAS and SDA on incompressible Naiver-Stokes super-resolution and sparse observation tasks.** Results are reported for various settings: super-resolution from 32^2 and 16^2 to 128^2 and sparse observation recovery at 5% and 1.5625% observed pixels. The best results are highlighted in **bold**.

[Chen et al., 2014, Özcan et al., 2023, Tan et al., 2009].

In a standard PIV setup, as shown in Figure 4, fluorescent tracer particles are seeded into a fluid flowing through a channel with transparent walls. A laser sheet illuminates the fluid, and particle movements are recorded by a high-speed camera with adjustable temporal resolution. By analyzing the displacement of these tracer particles, the velocity field within the fluid can be determined at sparse locations. Unlike the task in Section 4.2, which involves recovering dense vorticity fields from sparse vorticity observations, PIV introduces a slightly different DA task: recovering dense vorticity fields ($\boldsymbol{x} = \boldsymbol{\omega}$) from sparse velocity measurements. This observation model is defined by

$$y = \mathcal{A}(\boldsymbol{v}(\boldsymbol{\omega})) + \boldsymbol{\eta}, \tag{19}$$

where the $\mathcal{A}(\cdot)$ sparsely samples the velocity field $\boldsymbol{v}$ and the observation noise $\boldsymbol{\eta}$ has a standard deviation of $\gamma = 0.25$. The relationship between the velocity $\boldsymbol{v}$ and the state (vorticity) $\boldsymbol{\omega}$ is given by $\boldsymbol{\omega} = \nabla \times \boldsymbol{v}$. To derive the velocity $\boldsymbol{v}$ from the vorticity $\boldsymbol{\omega}$, we first solve the Poisson equation $\Delta\psi = -\boldsymbol{\omega}$ to obtain the stream function ψ, and then compute the velocity $\boldsymbol{v}$ as the gradient of ψ. This process is performed using the Fast Fourier Transform.

Dataset In this experiment, we used the same fluid dynamics simulation data from the NS experiments described in Section 4.2. However, we convert the vorticity data to velocity fields via Fourier transform to create synthetic PIV datasets [Lagemann et al., 2021]. The particle positions in our simulation were randomly initialized and then perturbed according to the simulated flow motion pattern. In these synthetic images, we assumed a particle density of 0.03 particles per pixel, a particle diameter of 3 pixels, and a peak intensity of 255 for each particle in grayscale. The images were processed through a standard PIV pipeline to extract particle locations, match corresponding particles across frames, and compute sparse velocity observations. These sparse measurements were then used in DA to reconstruct the full vorticity fields.

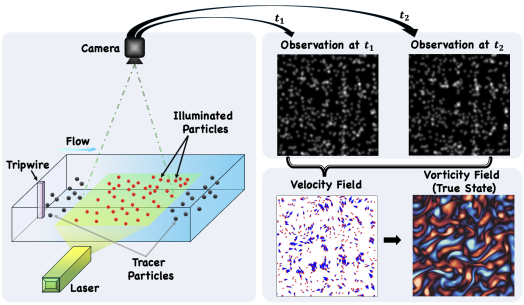


Figure 4: **Illustration of the real-world Particle Image Velocimetry (PIV) experiment**: The flow is seeded with tracer particles illuminated by a laser sheet, and their movements are captured by a camera to derive the sparse velocity field. The goal here is to recover dense vorticity field from the sparse velocity measurement.

Baselines and metrics As in Section 4.2, we compared our method against the SDA solver with a fixed window size of 2. Instead of training new scores or stochastic

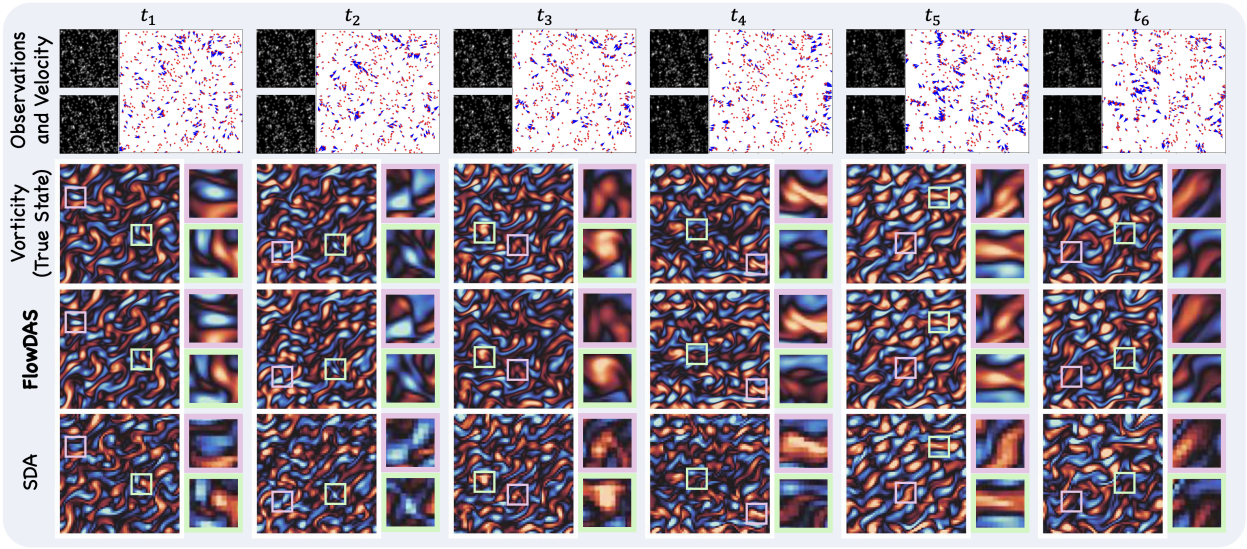


Figure 5: **Data assimilation of Particle Image Velocimetry.** The vorticity field is visualized in the same way as in Figure 3. FlowDAS significantly outperforms SDA in terms of reconstruction resolution and RMSE, recovering more detailed features even when direct observations are not available. These improvements highlight the potential of FlowDAS for real-world applications.

interpolant networks, we directly adopted the networks
trained on vorticity data from the incompressible NS flow experiment to evaluate FlowDAS and SDA on the PIV task. The same quantitative metrics from Section 4.2 were employed to validate the performance of each method.

Results FlowDAS accurately recovers the underlying fluid motion from the observed particle images, yielding vorticity estimates with minimal error relative to the ground truth. As shown in Figure 5 and Appendix B.5.3, FlowDAS significantly outperformed SDA in terms of reconstruction resolution and RMSE. This improvement underscores FlowDAS's robustness and applicability in fluid dynamics research.

5 Conclusion and Future Work

In this work, we introduced **FlowDAS**, a flow-based data assimilation framework designed to address the challenges of high-dimensional, nonlinear dynamical systems. By leveraging stochastic interpolants, FlowDAS effectively integrates complex transition dynamics with observational data, enabling accurate state estimation without relying on explicit physical simulations. Through experiments on both low- and high-dimensional systems—including the Lorenz 1963 system, incompressible Navier-Stokes flow, and Particle Image Velocimetry—FlowDAS demonstrated strong performance in recovering accurate state variables from sparse, noisy observations. These results highlight FlowDAS as a robust alternative to traditional model-driven methods (e.g., particle filters) and data-driven approaches (e.g., score-based data assimilation), offering improved accuracy, efficiency, and adaptability. Future work will focus on scaling the framework to more complex real-world systems [Huang et al., 2024, Seabra et al., 2024] and further optimizing its performance with Sim2Real setting [Chen et al., 2024a].

References

M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

I. Alkhouri, S. Liang, C.-H. Huang, J. Dai, Q. Qu, S. Ravishankar, and R. Wang. Sitcom: Step-wise triple-consistent diffusion sampling for inverse problems. *arXiv preprint arXiv:2410.04479*, 2024.

W. Bai, S. Chen, W. Chen, and H. Sun. Blind inversion using latent diffusion priors, 2024. URL https://arxiv.org/abs/2407.01027.

F. Bao, Z. Zhang, and G. Zhang. A score-based nonlinear filter for data assimilation, 2023. URL https://arxiv.org/abs/2306.09282.

L. Bertino, G. Evensen, and H. Wackernagel. Sequential data assimilation techniques in oceanography. *International Statistical Review*, 71(2):223–241, 2003.

P. Bickel, B. Li, and T. Bengtsson. Sharp failure rates for the bootstrap particle filter in high dimensions. In *Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh*, volume 3, pages 318–330. Institute of Mathematical Statistics, 2008.

M. Bocquet, H. Elbern, H. Eskes, M. Hirtl, R. Žabkar, G. Carmichael, J. Flemming, A. Inness, M. Pagowski, J. Pérez Camaño, et al. Data assimilation in atmospheric chemistry models: current status and future prospects for coupled chemistry meteorology models. *Atmospheric chemistry and physics*, 15(10):5325–5358, 2015.

F. Bouttier and P. Courtier. Data assimilation concepts and methods march 1999. *Meteorological training course lecture series. ECMWF*, 718:59, 2002.

A. Carrassi, M. Bocquet, L. Bertino, and G. Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9 (5):e535, 2018.

J. A. Carton, G. Chepurin, X. Cao, and B. Giese. A simple ocean data assimilation analysis of the global upper ocean 1950–95. part i: Methodology. *Journal of Physical Oceanography*, 30(2):294–309, 2000.

C.-Y. Chen, R. Antón, M.-y. Hung, P. Menon, E. A. Finol, and K. Pekkan. Effects of intraluminal thrombus on patient-specific abdominal aortic aneurysm hemodynamics via stereoscopic particle image velocity and computational fluid dynamics modeling. *Journal of biomechanical engineering*, 136(3):031001, 2014.

J. Chen, P. Li, Y. Wang, P.-C. Ku, and Q. Qu. Sim2Real in reconstructive spectroscopy: Deep learning with augmented device-informed data simulation. *APL Machine Learning*, 2(3):036106, 08 2024a. ISSN 2770-9019. doi: 10.1063/5.0209339. URL https://doi.org/10.1063/5.0209339.

Y. Chen, M. Goldstein, M. Hua, M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Probabilistic forecasting with stochastic interpolants and Föllmer processes. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st*

International Conference on Machine Learning, volume 235 of *Proceedings of Machine Learning Research*, pages 6728–6756. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/chen24n.html.

H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

J. A. Cummings. Operational multivariate ocean data assimilation. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3583–3604, 2005.

J. A. Cummings and O. M. Smedstad. Variational data assimilation for the global ocean. In *Data assimilation for atmospheric, oceanic and hydrologic applications (Vol. II)*, pages 303–343. Springer, 2013.

J. Derber and A. Rosati. A global oceanic data assimilation system. *Journal of physical oceanography*, 19(9):1333–1347, 1989.

F. G. Ergin, B. B. Watz, and N. F. Gade-Nielsen. A review of planar PIV systems and image processing tools for lab-on-chip microfluidics. *Sensors*, 18(9):3090, 2018.

G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53:343–367, 2003.

S. J. Fletcher. *Data assimilation for the geosciences: From theory to application*. Elsevier, 2017.

A. J. Geer, K. Lonitz, P. Weston, M. Kazumori, K. Okamoto, Y. Zhu, E. H. Liu, A. Collard, W. Bell, S. Migliorini, et al. All-sky satellite data assimilation at operational weather forecasting centres. *Quarterly Journal of the Royal Meteorological Society*, 144(713):1191–1217, 2018.

N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

N. Gustafsson, T. Janjić, C. Schraff, D. Leuenberger, M. Weissmann, H. Reich, P. Brousseau, T. Montmerle, E. Wattrelot, A. Bučánek, et al. Survey of data assimilation methods for convective-scale numerical weather prediction at operational centres. *Quarterly Journal of the Royal Meteorological Society*, 144(713):1218–1256, 2018.

L. Huang, L. Gianinazzi, Y. Yu, P. D. Dueben, and T. Hoefler. DiffDA: a diffusion model for weather-scale data assimilation. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 19798–19815. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huang24h.html.

C. King, E. Walsh, and R. Grimes. PIV measurements of flow within plugs in a microchannel. *Microfluidics and Nanofluidics*, 3:463–472, 2007.

V. Koschatzky, P. Moore, J. Westerweel, F. Scarano, and B. Boersma. High speed PIV applied to aerodynamic noise investigation. *Experiments in fluids*, 50:863–876, 2011.

C. Lagemann, K. Lagemann, S. Mukherjee, and W. Schröder. Deep recurrent optical flow learning for particle image velocimetry data. *Nature Machine Intelligence*, 3:641 – 651, 2021. URL https://api.semanticscholar.org/CorpusID:237869288.

B. K. W. Lahoz and R. Menard. *Data assimilation*. Springer, 2010.

K. Law, A. Stuart, and K. Zygalakis. Data assimilation. *Cham, Switzerland: Springer*, 214:52, 2015.

X. Li, S. M. Kwon, I. R. Alkhouri, S. Ravishanka, and Q. Qu. Decoupled data consistency with diffusion purification for image restoration. *arXiv preprint arXiv:2403.06054*, 2024.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963. URL https://api.semanticscholar.org/CorpusID:15359559.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:53592270.

N. A. Z. M. Noar, N. I. A. Apandi, and N. Rosli. A comparative study of Taylor method, fourth order Runge-Kutta method and Runge-Kutta Fehlberg method to solve ordinary differential equations. *AIP Conference Proceedings*, 2895(1):020003, 03 2024. ISSN 0094-243X. doi: 10.1063/5.0192085. URL https://doi.org/10.1063/5.0192085.

C. Özcan, Ö. Kocatürk, C. Işlak, and C. Öztürk. Integrated particle image velocimetry and fluid–structure interaction analysis for patient-specific abdominal aortic aneurysm studies. *BioMedical Engineering OnLine*, 22(1):113, 2023.

R. Peyret. *Spectral methods for incompressible viscous flow*, volume 148. Springer, 2002.

Y. Qu, J. Nathaniel, S. Li, and P. Gentine. Deep generative data assimilation in multimodal setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 449–459, 2024.

F. Rabier. Overview of global data assimilation developments in numerical weather-prediction centres. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3215–3233, 2005.

M. Raffel, C. E. Willert, F. Scarano, C. J. Kähler, S. T. Wereley, and J. Kompenhans. *Particle image velocimetry: a practical guide*. springer, 2018.

R. H. Reichle. Data assimilation methods in the earth sciences. *Advances in water resources*, 31(11): 1411–1418, 2008.

M. Rodell, P. Houser, U. Jambor, J. Gottschalck, K. Mitchell, C.-J. Meng, K. Arsenault, B. Cosgrove, J. Radakovich, M. Bosilovich, et al. The global land data assimilation system. *Bulletin of the American Meteorological society*, 85(3):381–394, 2004.

F. Rozet and G. Louppe. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

G. S. Seabra, N. T. Mücke, V. L. S. Silva, D. Voskov, and F. C. Vossepoel. AI enhanced data assimilation and uncertainty quantification applied to geological carbon storage. *ArXiv*, abs/2402.06110, 2024. URL https://api.semanticscholar.org/CorpusID:267616732.

S. Simic. On a global upper bound for jensen's inequality. *Journal of mathematical analysis and applications*, 343(1):414–419, 2008.

B. Song, S. M. Kwon, Z. Zhang, X. Hu, Q. Qu, and L. Shen. Solving inverse problems with latent diffusion models via hard data consistency. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=j8hdRqOUhN.

E. J. Stamhuis. Basics and principles of particle image velocimetry (PIV) for mapping biogenic and biologically relevant flows. *Aquatic Ecology*, 40(4):463–479, 2006.

E. Süli and D. F. Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

F. Tan, A. Borghi, R. Mohiaddin, N. Wood, S. Thom, and X. Xu. Analysis of flow patterns in a patient-specific thoracic aortic aneurysm model. *Computers & Structures*, 87(11-12):680–690, 2009.

Z. J. Taylor, R. Gurka, G. A. Kopp, and A. Liberzon. Long-duration time-resolved PIV to study unsteady aerodynamics. *IEEE Transactions on Instrumentation and Measurement*, 59(12):3262–3269, 2010.

P. J. Van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

B. Van Oudheusden. PIV-based pressure measurement. *Measurement Science and Technology*, 24(3): 032001, 2013.

C. Villani. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL https://doi.org/10.1007/978-3-540-71050-9_6.

B. Wang, X. Zou, and J. Zhu. Data assimilation and its applications. *Proceedings of the National Academy of Sciences*, 97(21):11143–11144, 2000.

Appendices

A Ablation Study

This section examines different aspects of the proposed FlowDAS, including an alternative to the method and the hyperparameter settings. We also provide various evaluation results. To simplify notation, we omit the explicit sampling distribution $p(\boldsymbol{x})$ in the expectation operator, writing $\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[f(\boldsymbol{x})]$ simply as $\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]$. The sampling distribution for $\boldsymbol{x}$ will be specified at the end of the equation where necessary.

A.1 Monte Carlo Sampling and An Alternative

In Equation (10) and Equation (11), we estimate $\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right)$ by

$$\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right) = \frac{\nabla \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]}{\mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]}, \tag{S.20}$$

where we approximate the expectation term by averaging J Monte Carlo samples:

$$\mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] \approx \frac{1}{J}\sum_{j=1}^{J} p\left(\boldsymbol{y}|\boldsymbol{X}_1^{(j)}\right), \boldsymbol{X}_1^{(j)} \sim p\left(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0\right). \tag{S.21}$$

This Monte Carlo estimate is an unbiased estimate and the error is proportional to $\frac{1}{\sqrt{J}}$. When the number of Monte Carlo samples, i.e., J, is sufficiently large, the estimation error converges to zero. Alternatively, one can also apply Jensen's inequality to estimate $\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right)$, which provides a biased estimate:

$$\begin{aligned} \nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right) &= \nabla \log \int p\left(\boldsymbol{y}|\boldsymbol{X}_1\right) p\left(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0\right) \mathrm{d}\boldsymbol{X}_1 \\ &= \nabla \log \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] \geq \nabla \mathbb{E}_{\boldsymbol{X}_1}\left[\log p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]. \end{aligned} \tag{S.22}$$

The "$\geq$" arises from Jensen's inequality.

Unbiased vs. biased estimation Let $Z = p(\boldsymbol{y}|\boldsymbol{X}_1) = \frac{1}{\sqrt{2\pi\gamma}} e^{-\frac{(\boldsymbol{y}-\mathcal{A}(\boldsymbol{X}_1))^2}{2\gamma^2}}$, where Z is a bounded random variable within the finite range $[0, \frac{1}{\sqrt{2\pi\gamma}}]$. As a result, the logarithm function is α-Hölder continuous with $\alpha = 1$ and the gap introduced by Jensen's inequality, i.e., the Jensen gap, can be explicitly bounded [Simic, 2008] and given by

$$\left|\log \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] - \mathbb{E}_{\boldsymbol{X}_1}\left[\log p(\boldsymbol{y}|\boldsymbol{X}_1)\right]\right| \leq M\gamma_1^1 \tag{S.23}$$

where M is a constant that satisfying $|\log Z - \log \mathbb{E}[Z]| \leq M\,|Z - \mathbb{E}[Z]|$ and $\gamma_1^1 = \mathbb{E}\left[|Z - \mathbb{E}\left[Z\right]|\right]$. Additionally, because $Z \in [0, \frac{1}{\sqrt{2\pi\gamma}}]$, we have $M < 1$ and $\gamma_1^1 \leq \max|Z - \mathbb{E}\left[Z\right]| \leq \frac{1}{\sqrt{2\pi\gamma}}$.
And for this upper-bound $M\gamma_1^1$, when $s \to 1$, $p\left(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right)$ will become a delta distribution concentrated on $\hat{\boldsymbol{X}}_1$ and $Z = p(\boldsymbol{y}|\boldsymbol{X}_1)$ will also have a delta distribution concentrated on $p(\boldsymbol{y}|\hat{\boldsymbol{X}}_1)$, and the $\gamma_1^1 \approx 0$ finally. In conclusion, this bias should be controlled by

$$\left|\log \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] - \mathbb{E}_{\boldsymbol{X}_1}\left[\log p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]\right| \leq \frac{1}{\sqrt{2\pi\gamma}} \tag{S.24}$$

Although this bias is theoretically bounded, it still results in a slight degradation in performance. Table S.3 shows the comparison for the Lorenz experiment to illustrate this point.

		UNBIASED	BIASED
↑	$\log p(\hat{\boldsymbol{x}}_{2:K} \mid \hat{\boldsymbol{x}}_1)$	**17.29**	-36.98
↑	$\log p(\boldsymbol{y} \mid \hat{\boldsymbol{x}}_{1:K})$	**-0.228**	-1.530
↓	$W_1(\boldsymbol{x}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$	**0.106**	0.111
↓	$\text{RMSE}(\boldsymbol{x}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$	**0.202**	0.363

Table S.3: **Evaluation of unbiased vs. biased estimation**. Comparison of metrics between unbiased and biased estimation in the Lorenz experiments. The results demonstrate that the unbiased estimation outperforms the biased estimation.

Hyperparameters for Monte Carlo Sampling We examine the estimation process and associated hyperparameters in Equation (S.21), where the expectation is computed using a Monte Carlo method. The hyperparameter J, is referred to as the number of Monte Carlo sampling iterations in Algorithm 2. It is important to note that increasing J does not lead to an increase in neural network evaluations but only involves additional Gaussian noise simulations, which are computationally lightweight. To illustrate the effect of J, we use numerical results from the Lorenz experiment. As shown in Table S.4, increasing J can improve performance by approximately 20%, with negligible impact on computational time.

$J =$	3	6	12	21	30	50
RMSE	0.167	0.148	0.153	0.142	0.150	0.138

Table S.4: **Effect of** J. The RMSE of generated state trajectories for FlowDAS is evaluated with different Monte Carlo sampling times J in Equation (S.21) for the Lorenz 1963 experiment. As J increases, the RMSE initially decreases, indicating improved performance, and then stabilizes.

A.2 The Method For Posterior Estimation

We also evaluate the impact of different methods for posterior estimation: Euler's method (1st-order estimation) and Heun's method (2nd-order estimation), as defined in Equation (12). The results are presented in Table S.5, where "1st-order" and "2nd-order" refer to Euler's and Heun's methods, respectively. "No correction" indicates forecasting purely based on the model without incorporating observation information. The results show that both 1st-order and 2nd-order estimations provide reasonable accuracy. However, the 2nd-order estimation (Heun's method) consistently delivers better performance. This suggests that employing more accurate estimations of $p\left(\boldsymbol{X}_1 | \boldsymbol{X}_s, \boldsymbol{X}_0\right)$ can effectively enhance model performance. Beyond the 2nd-order method, higher-order approaches like the Runge-Kutta 4th-order (RK4) method could further improve accuracy. However, these methods come with increased computational cost: 2nd-order estimation requires two neural network evaluations per step compared to one for 1st-order estimation, while RK4 requires four evaluations per step. In our experiments, we find that the 2nd-order estimation strikes a good balance between performance and efficiency, making it a practical choice. Further exploration of higher-order methods will be left for future research.

B Experiment Details and Results

This section provides the details of our experiments, including additional experiments and results.

	1ST-ORDER	2RD-ORDER	NO CORRECTION
$32^2 \rightarrow 128^2$	0.048	**0.038**	0.206
$16^2 \rightarrow 128^2$	0.101	**0.067**	0.206

Table S.5: **Effect of posterior estimation**. The RMSE of vorticity estimate from FlowDAS is evaluated on the incompressible Navier-Stokes task using different posterior estimation methods: Euler's method (1st-order estimation), Heun's method (2rd-order estimation), as defined in Equation (12), and forecasting without observations (i.e., no correction). For the super-resolution tasks $16^2 \rightarrow 128^2$ and $32^2 \rightarrow 128^2$, both posterior estimation methods significantly outperform forecasting without observations. Among them, the 2nd-order method achieves the lowest RMSE.

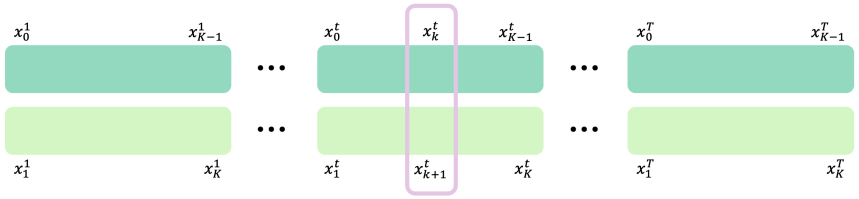


Figure S.6: **Structure of training data**. Consecutive states are paired across multiple simulated trajectories to construct the $\tilde{\boldsymbol{I}}_s$ and $\boldsymbol{R}_s$ defined in Equation (13) for training the velocity model $\boldsymbol{b}_s$.

B.1 Constructing Training Dataset

In the training stage for all experiments, we require pairs of consecutive states to train the velocity model $\boldsymbol{b}_s$. To generate these pairs, we proceed as follows:

1. **Trajectory simulation.** We simulate T independent trajectories, denoted as $\boldsymbol{x}_{0:K}^{1:T}$, where each trajectory starts from a unique initial state $\boldsymbol{x}_0^t$. The state transitions within each trajectory follow the dynamics defined in Equation (1).

2. **Consecutive state pairs formation.** For each trajectory t, form two aligned sequences:

   (a) $\boldsymbol{x}_{0:K-1}^t$: The original sequence of states with each last state x_K^t discarded.

   (b) $\boldsymbol{x}_{1:K}^t$: The sequence of states shifted by one time step with each initial state x_0^t discarded.

   The two sequences form pairs of consecutive states $(\boldsymbol{x}_k^t, \boldsymbol{x}_{k+1}^t)$ for $k = 0, 1, \cdots, K-1$.

3. **Concatenating trajectories.** Then, we concatenate T sequences of $\boldsymbol{x}_{0:K-1}^t$ and T sequences $\boldsymbol{x}_{1:K}^t$ end-to-end, respectively, into two long sequences: $\boldsymbol{x}_{0:K-1}^{1:T}$ and $\boldsymbol{x}_{1:K}^{1:T}$, where each state in the second sequence is the corresponding successor states in the first sequence.

4. **Sampling training data.** During training, batches of paired consecutive states are sampled. For each batch, we sample training data pairs as follows:

   (a) sample states from $\boldsymbol{x}_{0:K-1}^{1:T}$ as $\boldsymbol{X}_0$'s.

   (b) retrieve the counterpart states from $\boldsymbol{x}_{1:K}^{1:T}$ as $\boldsymbol{X}_1$'s.

   The batches of state pairs $(\boldsymbol{X}_0, \boldsymbol{X}_1)$ are used to construct $\tilde{\boldsymbol{I}}_s$ and $\boldsymbol{R}_s$ in Equation (13) for training the velocity model $\boldsymbol{b}_s$ as described in Algorithm 1.

		Monte Carlo Sampling Times J	Sampling Step Size ζ
Double-well		17	1
Lorenz 1963		21	0.0002
Incompressible Navier Stokes	SR 4x	25	1
	SR 8x	25	2
	SO 5%	25	1
	SO 1.5625%	25	1.75
Particle Image Velocimetry		25	1

Table S.6: **Hyperparameters for FlowDAS in the inference stage** of all experiments presented in this study. SR stands for super-resolution task and SO represents sparse observation (inpainting) task.

In summary, we draw many state pairs $(\boldsymbol{X}_0, \boldsymbol{X}_1)$ and estimate the integral over s in Equation (13) by approximated via an empirical expectation over draws of $s \sim U(0,1)$, and for every s and every state pairs $(\boldsymbol{X}_0, \boldsymbol{X}_1)$, we independently compute Equation (14) with S samples of $\boldsymbol{z}_k$, so the training loss can be rewritten as:

$$\mathcal{L}_b^{\text{emp}}(\hat{\boldsymbol{b}}) = \frac{1}{K'} \sum_{k \in \mathcal{B}_{K'}} \frac{1}{S} \sum_{s=1}^{S} \|\hat{\boldsymbol{b}}_s(\tilde{\boldsymbol{I}}_s^k, \boldsymbol{x}_k) - \boldsymbol{R}_s^k\|^2 , \tag{S.25}$$

where we randomly draws s from $U(0,1)$. Note that the training batch size equals $S \times K'$. Figure S.6 illustrates the structure of training data, showing how consecutive states are paired across trajectories.

B.2 Double-well Potential Problem

B.2.1 The Double-well Potential System

In this experiment, we investigate a 1D tracking problem in a dynamical system driven by the double-well potential. The system is governed by the following stochastic dynamics:

$$\mathrm{d}x = -4x(x^2 - 1)\,\mathrm{d}t + \beta_d\,\mathrm{d}\xi_t \tag{S.26}$$

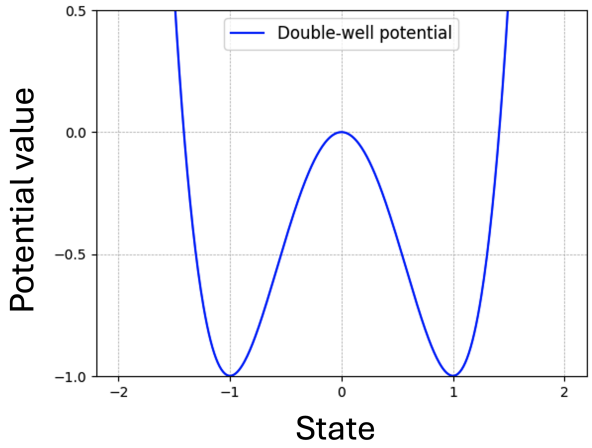
with observation model defined by $\mathcal{A}(x) = x^3$. The observations are given by

$$y = \mathcal{A}(x) + \eta = x^3 + \eta \tag{S.27}$$

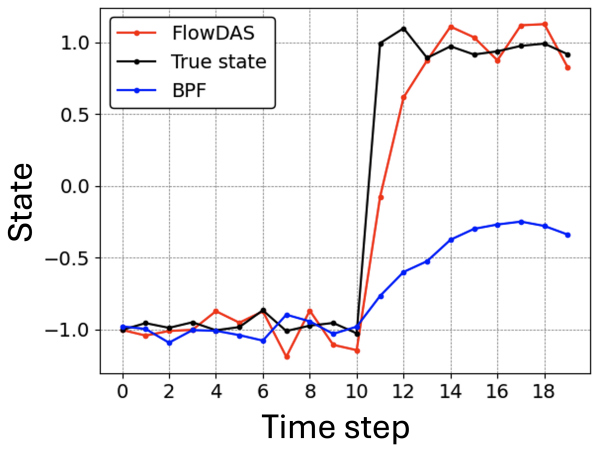
where the stochastic force ξ_t is a standard Brownian motion with diffusion coefficient of $\beta_d = 0.2$. η is standard Gaussian noise with standard deviation of 0.2. The Ψ is the derivative of the potential $U(x) = x^4 - 2x^2 + f$, where f is a function independent of x. The system describes a particle trapped in the wells located at $x = 1$ and $x = -1$, with small fluctuations around these points, as illustrated in Figure S.7a.

B.2.2 Training and Testing Data Generation

We trained the model on simulated trajectories generated by numerically solving the transition equation using a temporal step size of 0.1. The training dataset consisted of 500 trajectories, each of length 100, with initial points uniformly sampled from the range $[-2, 2]$. In the testing stage, we introduced stronger turbulence to the system, causing the particle to occasionally switch wells $(x \to -x)$.



(a) **Illustration of the Double-Well Potential Problem**. In the double-well potential system, particles are typically captured at the bottoms of the wells ($x = 1$ and $x = -1$). With low probability, particles can transition between wells due to the stochastic term in the transition equation.



(b) **Visualization Results of Double-Well Potential Problem**. We show the visualization results of FlowDAS and BPF, while the score-based solver SDA fails to produce reasonable results. FlowDAS can track the dramatic change of the particles that BPF struggles to immediately react to.

Figure S.7: Illustrations and visualization results for the double-well potential problem.

B.2.3 Neural Network for Learning the Drift Velocity

In this low-dimensional double-well potential task, the drift velocity $\boldsymbol{b}_s$ is approximated using a fully connected neural network with 3 hidden layers, each having a hidden dimension of 50. Both the input and output dimensions of the network are 1. For the condition $\boldsymbol{X}_0$ and timestep s, we empirically find that embedding $\boldsymbol{X}_0$, similar to how s is embedded, outperforms directly using $\boldsymbol{X}_0$ as an additional input to the network. The intuition behind this approach is that embedding $\boldsymbol{X}_0$ helps the network better distinguish between the two variables, $\boldsymbol{X}_s$ and $\boldsymbol{X}_0$. The model is trained using the Adam optimizer with a base learning rate of 0.005, along with a linear rate scheduler. Training is conducted for 5000 epochs.

B.2.4 Hyperparameters During Inference

Hyperparameters of the inference procedure, specified for the double-well potential task, are presented in Table S.6.

B.2.5 Baseline

For this task, we compare our method with SDA and the classic BPF method. For SDA, we fix the window size to two and use the same training data as FlowDAS to ensure fairness. The local score network is implemented as a fully connected neural network, following the architecture proposed in [Bao et al., 2023]. The score network consists of 3 hidden layers, each with a hidden dimension of 50. The model is trained using the AdamW optimizer with a base learning rate of 0.001, a weight decay of 0.001, and a linear learning rate scheduler. Training is conducted for 5000 epochs. For the BPF method, the particle density is set to 16384.

B.2.6 Results

The visual results are shown in Figure S.7b. Surprisingly, while both FlowDAS and the classic BPF method produce reasonable results, FlowDAS demonstrates superior performance in tracking dramatic changes in the trajectory. In contrast, the score-based solver SDA fails to produce reasonable results. This failure arises because the starting point of SDA during optimization is purely Gaussian noise, leading to a poor initial estimation of the target state. Furthermore, the cubic observation model amplifies the differences, causing the optimization gradients to explode and resulting in recovery failure.

In FlowDAS, the error is bounded by $||\boldsymbol{y}_{k+1} - \mathcal{A}(\boldsymbol{x}_k)||_2^2$ because the generation process begins with the previous state, which serves as a proximal estimate of the target state. Consequently, FlowDAS undergoes a more stable optimization process.

Compared to the classic BPF method, we find that it struggles to capture dramatic changes in the trajectory. This limitation is primarily due to the small diffusion coefficient β_d, such as $\beta_d = 0.2$, which causes the predicted filtering density in BPF to concentrate around the mean value dictated by the deterministic part of the transition equation. As a result, extreme cases lying in the tail of the future state distribution $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ are often missed. This phenomenon can be explained by the truncation error arising from the finite particle space [Bao et al., 2023].

In contrast, FlowDAS is capable of immediately sampling from the true conditional distribution as the steps become sufficiently large. This allows FlowDAS to better capture the tail region of the true distribution and react to dramatic changes, even those in low-probability areas. Additionally, FlowDAS can effectively incorporate observation information, enabling it to balance prediction and observation even when the true underlying state occurs in a low-probability region.

B.3 Lorenz 1963

B.3.1 The Lorenz 1963 Dynamic System

To simulate this system, we use the RK4 method, which updates the solution from time t_n to $t_{n+1} = t_n + h$ using the following formulas for each variable a, b, and c:

$$\begin{aligned} k_{1a_n} &= h \cdot \mu(b_n - a_n), \\ k_{1b_n} &= h \cdot (a_n(\rho - c_n) - b_n), \\ k_{1c_n} &= h \cdot (a_n b_n - \tau c_n), \end{aligned} \tag{S.28}$$

$$\begin{aligned} k_{2a_n} &= h \cdot \mu \left((b_n + \frac{k_{1b_n}}{2}) - (a_n + \frac{k_{1a_n}}{2}) \right), \\ k_{2b_n} &= h \cdot \left((a_n + \frac{k_{1a_n}}{2})(\rho - (c_n + \frac{k_{1c_n}}{2})) - (b_n + \frac{k_{1b_n}}{2}) \right), \\ k_{2c_n} &= h \cdot \left((a_n + \frac{k_{1a_n}}{2})(b_n + \frac{k_{1b_n}}{2}) - \tau(c_n + \frac{k_{1c_n}}{2}) \right), \end{aligned} \tag{S.29}$$

$$\begin{aligned} k_{3a_n} &= h \cdot \mu \left((b_n + \frac{k_{2b_n}}{2}) - (a_n + \frac{k_{2a_n}}{2}) \right), \\ k_{3b_n} &= h \cdot \left((a_n + \frac{k_{2a_n}}{2})(\rho - (c_n + \frac{k_{2c_n}}{2})) - (b_n + \frac{k_{2b_n}}{2}) \right), \\ k_{3c_n} &= h \cdot \left((a_n + \frac{k_{2a_n}}{2})(b_n + \frac{k_{2b_n}}{2}) - \tau(c_n + \frac{k_{2c_n}}{2}) \right), \end{aligned} \tag{S.30}$$

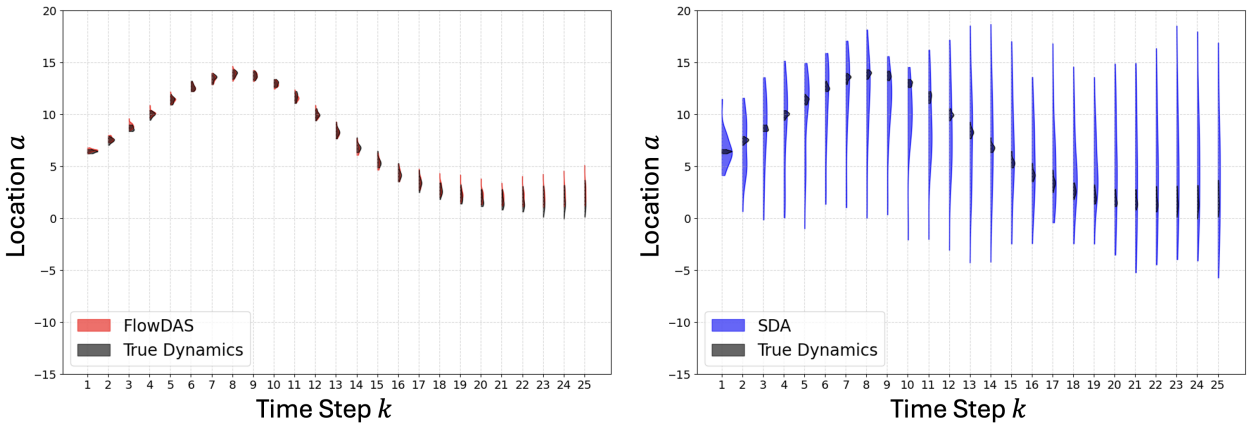


Figure S.8: **Dynamics modeling evaluation in the Lorenz 1963 task: FlowDAS vs. SDA**. FlowDAS produces results that closely match the true states, demonstrating its ability to learn the underlying transition dynamics $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$. In contrast, SDA exhibits rapid divergence from the true states. This divergence arises because SDA focus on modeling the joint distribution $p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k)$ rather than directly learn the transition dynamics $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$, which is inherently less suitable for capturing the system's underlying dynamics.

$$
\begin{aligned}
k_{4a_n} &= h \cdot \mu \left((b_n + k_{3b_n}) - (a_n + k_{3a_n}) \right), \\
k_{4b_n} &= h \cdot \left((a_n + k_{3a_n})(\rho - (c_n + k_{3c_n})) - (b_n + k_{3b_n}) \right), \\
k_{4c_n} &= h \cdot \left((a_n + k_{3a_n})(b_n + k_{3b_n}) - \tau(c_n + k_{3c_n}) \right).
\end{aligned} \tag{S.31}
$$

Then, the updates for a_n, b_n, and c_n are:

$$
\begin{aligned}
a_{n+1} &= a_n + \frac{1}{6}(k_{1a_n} + 2k_{2a_n} + 2k_{3a_n} + k_{4a_n}), \\
b_{n+1} &= b_n + \frac{1}{6}(k_{1b_n} + 2k_{2b_n} + 2k_{3b_n} + k_{4b_n}), \\
c_{n+1} &= c_n + \frac{1}{6}(k_{1c_n} + 2k_{2c_n} + 2k_{3c_n} + k_{4c_n}).
\end{aligned} \tag{S.32}
$$

After solving these deterministic updates, the stochastic force (ξ_1, ξ_2, ξ_3) is added to $(a_{n+1}, b_{n+1}, c_{n+1})$, which leads to the Lorenz 1963 system dynamic equations described in Equation (15).

B.3.2 Training and Testing Data Generation

We apply the RK4 method described in Appendix B.3.1 to generate the simulated training and testing data.

B.3.3 Hyperparameters During Inference

Hyperparameters of the inference procedure, specified for the Lorenz 1963 task, are presented in Table S.6. Noticeably, the step size ζ is smaller than in other experiments due to the chaotic nature of the Lorenz 1963 system, which requires finer step size to accurately capture its dynamics.

B.3.4 Neural Network for Learning the Drift Velocity

In this task, we use a fully connected neural network with 5 hidden layers, each with a hidden dimension of 256, to approximate the velocity field $\boldsymbol{b}_s$. The input and output dimensions are both

3. For the condition $\boldsymbol{X}_0$ and timestep s, we use embeddings of dimension 4. The model is trained using the Adam optimizer with a base learning rate of 0.005, and a linear learning rate scheduler is applied. Training is conducted over 23000 epochs.

B.3.5 Baseline

We compare FlowDAS with SDA and BPF. For the SDA, we use a score neural network with a fixed window size of 2. This local score function is implemented using a fully connected neural network with 5 hidden layers, each having a hidden dimension of 256. The model is trained using the AdamW optimizer with a base learning rate of 0.001 and a linear scheduler over 23000 epochs. For BPF, the particle density is set to 16384.

B.3.6 Accuracy of the Learned Dynamics

We evaluate the dynamic learning performance of FlowDAS and SDA, focusing on their ability to forecast future states **without** using observations. For BPF, we do not evaluate its performance in this context since it explicitly incorporates the true system dynamics.

As SDA is designed to rely on observations and does not work in an auto-regressive manner, we adapt it for this evaluation by using the previous state as a pseudo-observation. The SDA then forecasts the next state based on this input. For FlowDAS, we disable the observation-based update step by setting the step size $\zeta_n = 0$.

Both methods are initialized with the same initial state, and we simulate 64 independent trajectories of length 25. True states are generated by solving Equation (15) using the RK4 method, starting from the same initial point. The visualization of location a across 25 time steps is shown in Figure S.8.

B.4 Incompressible Navier-Stokes Flow

B.4.1 Incompressible Navier-Stokes Flow Problem Settings

For the incompressible Navier-Stokes flow problem, we adopted the problem setting from [Chen et al., 2024b] and used the provided training data. The choices of experiment parameters are as follows: $\nu = 10^{-3}$, $\alpha = 0.1$, $\varepsilon = 1$. The stochastic force ξ is defined as:

$$\begin{aligned}\xi(t,x,y) = W_1(t)\sin(6x) + W_2(t)\cos(7x) + W_3(t)\sin(5(x+y)) + W_4(t)\cos(8(x+y)) \\ + W_5(t)\cos(6x) + W_6(t)\sin(7x) + W_7(t)\cos(5(x+y)) + W_8(t)\sin(8(x+y))\end{aligned} \tag{S.33}$$

For more details and insights about this problem, see [Chen et al., 2024b].

B.4.2 Hyperparameters During Inference

Table S.6 presents the hyperparameters of the inference procedure for the incompressible Navier-Stokes flow task.

B.4.3 Neural Network for Learning the Drift Velocity

We used a U-Net architecture to approximate $\boldsymbol{b}_s$, following the network proposed in [Chen et al., 2024b]. The conditioning on $\boldsymbol{X}_0$ was implemented through channel concatenation in the input. The architectural details are as follows:

- Number of initial channels: 128.

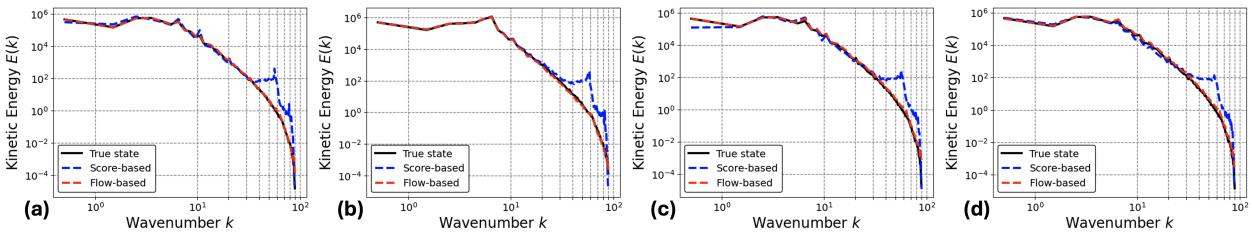


Figure S.9: **The kinetic energy spectrum of:** (a) super-resolution task: $16^2 \rightarrow 128^2$; (b) super-resolution task: $32^2 \rightarrow 128^2$; (c) sparse observation task: 1.5625%; (d) sparse observation task: 5%, in the incompressible Navier-Stokes flow task. We present the kinetic energy spectrum of the true state alongside the estimations from FlowDAS and SDA. FlowDAS can produce results that better aligned with the true state in terms of the kinetic energy spectrum, indicating FlowDAS's superiority in recovering the physics information and effectiveness as a surrogate model for stochastic dynamic systems. In contrast, SDA fails to produce reasonable high-frequency physics, evidenced by the oscillations in the spectrum.

- Multiplication factor for the number of channels at each stage: $(1, 2, 2, 2)$.
- Number of groups for group normalization in ResNet blocks: 8.
- Dimensionality of learned sinusoidal embeddings: 32.
- Dimensionality of each attention head in the self-attention mechanism: 64.
- Number of attention heads in the self-attention layers: 4.

We employ the AdamW optimizer [Loshchilov and Hutter, 2017] with a cosine annealing schedule to reduce the learning rate during training. The base learning rate is set to 2×10^{-4}. Training is conducted with a batch size of 32 for 10000 epochs.

B.4.4 Baseline

We compared our method with SDA. For SDA, the score neural network was configured with a temporal window of 2, an embedding layer dimensionality of 32, and a base number of feature channels set to 256. The network depth was 5, and the activation function used was SiLU [Loshchilov and Hutter, 2017]. The training process used a batch size of 64 and the AdamW optimizer. A linear learning rate scheduler was applied with an initial learning rate of 0.001, and the weight decay was set to 0.001. The model was trained for 10000 epochs.

B.4.5 Kinetic Energy Spectrum Analysis

We evaluated the methods from a physics perspective using the kinetic energy spectrum. A better method produces results that align more closely with the kinetic energy spectrum of the true state. The kinetic energy spectrum is computed as follows:

$$E(k_n) = \sum_{\substack{p,q \\ k_{p,q} \in \text{bin } n}} E(k_{x_p}, k_{y_q}), \tag{S.34}$$

where $k_{p,q}$ represents the wavenumbers grouped into bin n, $E(k_{x_p}, k_{y_q})$ is the kinetic energy at each wavenumber and p,q are the index representing a specific discrete wavenumber along the k_x or k_y axis.

Since the direct outputs of both FlowDAS and SDA are vorticity fields, it is necessary to first convert the vorticity into velocity before computing the kinetic energy spectrum. This conversion is achieved by solving the Poisson equation $\Delta\psi = -\omega$ to obtain ψ, and then calculating its gradient $\boldsymbol{v} = -\nabla\psi$ to derive the velocity $\boldsymbol{v}$.

We present the results of the kinetic energy spectrum analysis for the Navier-Stokes flow super-resolution and sparse observation tasks.

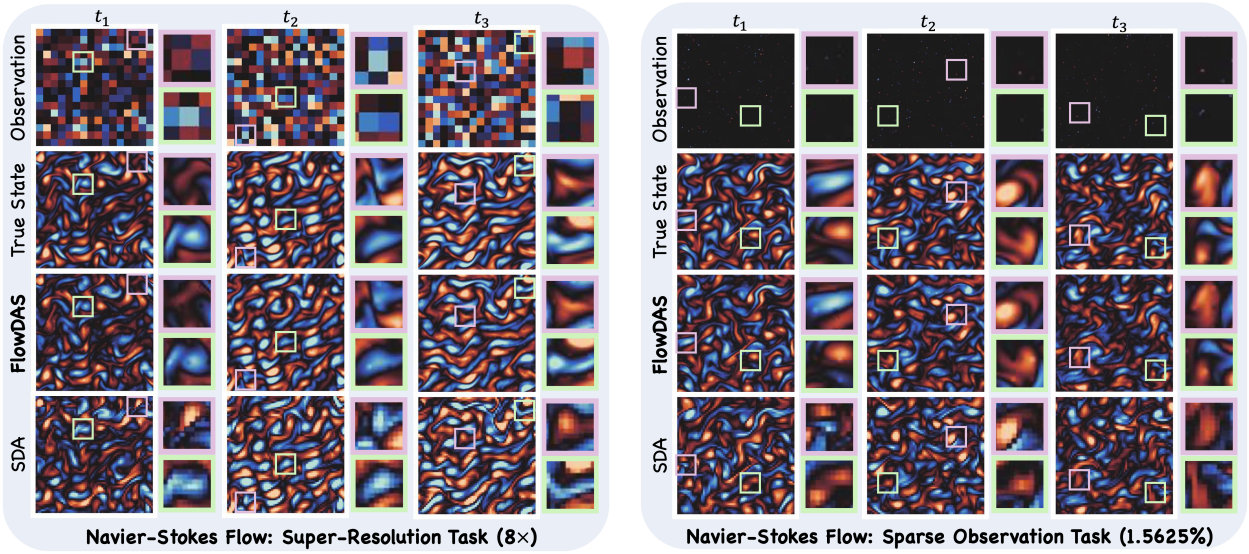


Figure S.10: **Data assimilation of incompressible Navier-Stokes flow**. Experiments were conducted at observation resolution of 16^2 (left) and observation sparsity level at 1.5625% (right). The goal of super-resolution task is to reconstruct high-resolution vorticity data (128^2) from low-resolution observations and the goal of sparse observation task is to recover the complete vorticity field.

B.4.6 Additional Results

In this section, we present additional results to Section 4.2, including the kinetic energy spectrum and visualization results for the Navier-Stokes flow super-resolution and sparse observation tasks.

$16^2 \rightarrow 128^2$ **Super-Resolution** The kinetic energy spectrum is shown in Figure S.9 (a), and additional visualization results are provided in Figure S.10 (left). FlowDAS effectively reconstructs high-resolution vorticity data from low-resolution observations, while SDA struggles to capture high-frequency physics.

$32^2 \rightarrow 128^2$ **Super-Resolution** The kinetic energy spectrum is shown in Figure S.9 (b). Similar to the $16^2 \rightarrow 128^2$ task, FlowDAS achieves superior alignment with the true state compared to SDA.

1.5625% **Sparse Observation** The kinetic energy spectrum is presented in Figure S.9 (c), with additional visualization results in Figure S.10 (right). FlowDAS accurately recovers the vorticity field despite the highly sparse observations, significantly outperforming SDA.

5% **Sparse Observation** The kinetic energy spectrum is shown in Figure S.9 (d). FlowDAS continues to outperform SDA, demonstrating strong recovery of physical information.

Across all tasks, FlowDAS consistently outperforms SDA, demonstrating superior alignment with the true kinetic energy spectrum and effective recovery of the vorticity field. FlowDAS not only excels in super-resolution tasks but also handles sparse observation challenges with robustness, maintaining physical coherence and accurately capturing high-frequency dynamics.

B.5 Particle Image Velocimetry

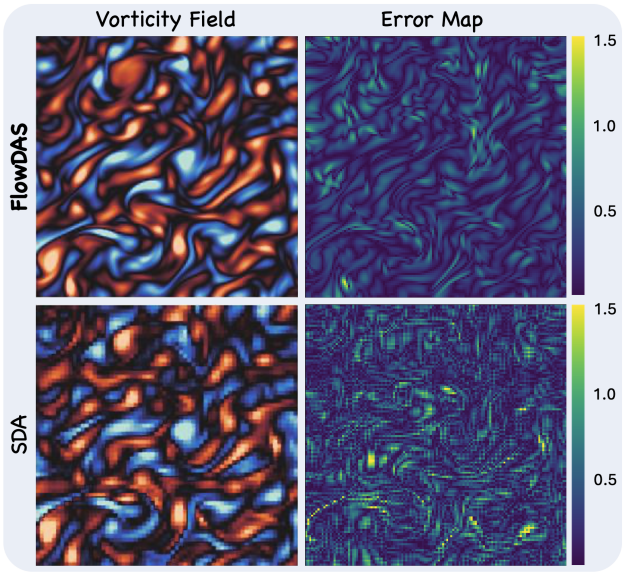


Figure S.11: **Reconstructed Vorticity Fields and Error Maps from FlowDAS and SDA**. For the example shown in Figure 4, FlowDAS demonstrated superior reconstruction quality compared to the score-based method SDA. Numerically, FlowDAS achieved a lower RMSE of 0.12, outperforming SDA with an RMSE of 0.15.

B.5.1 Experiment Setting

The training data, neural network (including FlowDAS and SDA) is the same as those in the incompressible Navier-stokes flow simulation. Figure 4 shows the standard PIV set up.

B.5.2 Hyperparameters During Inference

Hyperparameters of the inference procedure, specified for the PIV task, are presented in Table S.6.

B.5.3 Additional Results

RMSE We compared the RMSE of the reconstructed vorticity fields with the true states for the PIV task. Using FlowDAS and the score-based method SDA, we independently generated 10 results and computed the RMSE for each. FlowDAS achieved a lower average RMSE of 0.12, compared

Parameter	Value	Unit
Particle density	0.03	particle per pixel
Particle diameter	3	Pixel
Peak intensity	255	Gray value

Table S.7: **PIV task: parameters for simulation.** This table summarizes parameters for the PIV experiments in detail.

to 0.15 for SDA. This highlights the ability of FlowDAS in accurately reconstructing vorticity fields for the PIV task.

Error map Figure S.11 provides the error maps of FlowDAS and SDA for the example shown in Figure 4. In the FlowDAS error map, most regions are darker (indicating lower errors), with fewer bright spots. In contrast, the SDA error map shows more scattered bright areas and generally higher values. These results suggest that FlowDAS recovers the PIV vorticity field more accurately.