FlowDAS: A Stochastic Interpolant-based Framework for Data Assimilation

Siyi Chen[*]
University of Michigan
siyiche@umich.edu

Yixuan Jia[*]
University of Michigan
jiayx@umich.edu

Qing Qu
University of Michigan
qingqu@umich.edu

He Sun[†]
Peking University
hesun@pku.edu.cn

Jeffrey A. Fessler
University of Michigan
fessler@umich.edu

Abstract

Data assimilation (DA) integrates observations with a dynamical model to estimate states of PDE-governed systems. Model-driven methods (e.g., Kalman Filter, Particle Filter) presuppose full knowledge of the true dynamics, which is not always satisfied in practice, while purely data-driven solvers learn a deterministic mapping between observations and states and therefore miss the intrinsic stochasticity of real processes. Recently, score-based diffusion models have shown promise for DA by learning a global diffusion prior to represent stochastic dynamics. However, their one-shot generation lacks stepwise physical consistency and struggles with complex stochastic processes. To address these issues, we propose **FlowDAS**, a generative DA framework that employs stochastic interpolants to learn state transition dynamics through step-by-step stochastic updates. By incorporating observations into each transition, FlowDAS can produce stable, measurement-consistent forecasts. Experiments on Lorenz-63, Navier–Stokes super-resolution/sparse-observation scenarios, and large-scale weather forecasting——where dynamics are partly or wholly unknown—show that FlowDAS surpasses model-driven methods, neural operators, and score-based baselines in accuracy and physical plausibility. Our implementation is available at https://github.com/umjiayx/FlowDAS.

1 Introduction

Recovering state variables in complex dynamical systems is a fundamental problem in science and engineering. Accurate state estimation from noisy, incomplete data is critical in weather forecasting [1, 2], oceanography [3, 4], seismology [5, 6], and many other fields [7], where reliable predictions depend on understanding the underlying physics [8–10]. A representative example is fluid dynamics [11–13], where one aims to reconstruct a continuous velocity field from sparse, noisy observations governed by nonlinear, time-dependent partial differential equations (PDEs)—a task complicated by stochasticity and high dimensionality. To meet these challenges, **data assimilation** (DA) combines model forecasts with observations to produce physically consistent state estimates; developed first in atmospheric and oceanic forecasting, DA is now ubiquitous across many scientific and engineering domains [14–20].

Mathematically, a discrete-time stochastic dynamical system can be described by:

$$\boldsymbol{x}_{k+1} = \Psi(\boldsymbol{x}_k) + \boldsymbol{\xi}_k, \tag{1}$$

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

---

[*]Equal contribution. Work done while Siyi Chen was an undergraduate student at Peking University.

[†]Corresponding author. This author is affiliated with the National Biomedical Imaging Center, Peking University, Beijing and Academy for Advanced Interdisciplinary Studies, Peking University, Beijing.
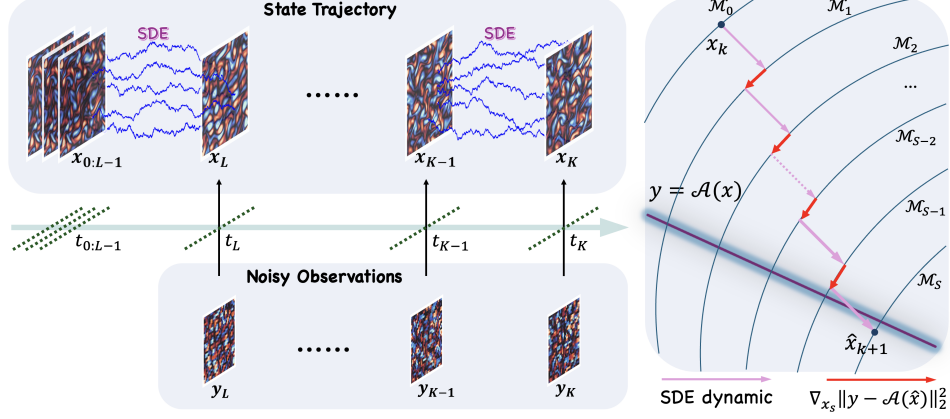
Figure 1: **An overview of FlowDAS.** We introduce a stochastic interpolant-based framework for data assimilation, named FlowDAS, to estimate states $\boldsymbol{x}_{L:K}$ from the noisy (sparse or low-resolution) observations $\boldsymbol{y}_{L:K}$. FlowDAS models the stochastic dynamics of the system with a flow-based stochastic differential equation (SDE) and incorporates the observations to improve the prediction accuracy. On the right, we show a conceptual illustration of the geometry of the process to estimate $\hat{\boldsymbol{x}}_{k+1}$ from $\boldsymbol{x}_k$. $\mathcal{M}_s$ denotes the generative manifold at interpolation step s. The gradient guidance $\nabla_{\boldsymbol{x}_s} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}})\|_2^2$ enforces observation consistency during the generation process.

where $\boldsymbol{x}_k \in \mathbb{R}^D$ is the state vector at time step k, $\Psi(\cdot)$ is the state transition map, and $\boldsymbol{\xi}_k \in \mathbb{R}^D$ denotes the stochastic force. The observations $\boldsymbol{y}_k \in \mathbb{R}^M$ are related to the state through the measurement map $\mathcal{A}(\cdot)$, with observation noise $\boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{0}, \gamma^2 \boldsymbol{I}_M)$. In many practical settings we have an initial window of known states $\boldsymbol{x}_{0:L-1}$, after which observations $\boldsymbol{y}_{L:K}$ will only be noisy and partial. The filtering problem in DA seeks to infer the posterior of the state trajectory $\boldsymbol{x}_{L:K}$ given observations $\boldsymbol{y}_{L:K}$ and the initial states $\boldsymbol{x}_{0:L-1}$, i.e., $p(\boldsymbol{x}_{L:K} \mid \boldsymbol{y}_{L:K}, \boldsymbol{x}_{0:L-1})$, as shown in Figure 1. Moreover, when the observations are absent, the task reduces to *probabilistic forecasting* [21], where we predict $p(\boldsymbol{x}_{L:K} \mid \boldsymbol{x}_{0:L-1})$.

Existing DA methods are split into two categories. Model-driven methods—Kalman variants for quasi-linear/Gaussian systems [22] and particle filters for fully nonlinear cases [23]—need an accurate physical model and become costly or unstable in high dimensions [24]. Data-driven surrogates learn unknown or partially known dynamics directly from observations and embed them within Bayesian inversion frameworks. However, the intrinsic stochasticity of complex systems exposes limitations: neural operators such as FNO [25] and Transolver [26] produce only deterministic forecasts and thus cannot quantify uncertainty, while diffusion models [21, 27, 28] struggle to learn physically faithful distributions due to the long Markov chain required to map Gaussian noise into realistic data.

Our contributions. We address the above limitations by building on recent flow-based *stochastic interpolant* methods [29–31], which learn a short-step conditional transition $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ instead of a global noise-to-data map. Our main contributions are:

- **FlowDAS**: We introduce a generative data-assimilation framework that treats a learned stochastic interpolant as the forward surrogate model and assimilates new observations on the fly at inference time, without retraining or auxiliary filtering steps.

- **Efficient, interpretable learning**: By modeling the transition between adjacent states, FlowDAS trains faster and more stably than long-horizon diffusion bridges, rolls out autoregressively, and provides clear physical insight into one-step dynamics.

- **Extensive validation**: Across the Lorenz-63 system, incompressible Navier–Stokes super-resolution and sparse-observation tasks, and large-scale SEVIR weather forecasting—where dynamics range from known to partially or fully unknown—FlowDAS consistently outperforms strong baselines in both accuracy and physical plausibility.

2 Preliminaries

2.1 Related Work

Data assimilation alternates between a *forecasting* step—propagating the current state forward under a dynamical model—and an *inverse* step that corrects this forecast with new observations.

Model-driven methods   Traditional model-driven approaches include the Kalman filter and its many variants [32]. The Kalman filter provides an optimal minimum-variance estimate when the dynamics are linear and all errors are Gaussian, but its performance degrades once these assumptions are not met. Also, there are some variants for nonlinear settings like the Extended Kalman Filter [33, 34], applying the nonlinear transition to predict states and its Jacobian to the covariance, struggling in discontinuity situations and causing computational expense to calculate the Jacobian. The Unscented Kalman Filters [35] bypass the computation of Jacobian by propagating a set of sample points with the nonlinear transformations and deriving the mean and covariance, making it sensitive to sample point parameters and non-Gaussian problems. For fully nonlinear, non-Gaussian problems, the bootstrap particle filter (BPF) [23] approximates the posterior with a set of weighted particles that are advanced by the dynamics, reweighted by the observation likelihood, and resampled to prevent weight collapse. Although flexible, the BPF requires an exponential number of particles as the state dimension grows, making it impractical for large physical systems [24]. Besides, both the Kalman family and particle filters rely on having accurate knowledge of the underlying dynamical models, a requirement that is often unmet in real-world applications.

Neural operator-based methods   Neural-operator solvers treat forecasting as a deterministic map from the current state to a single future state. The Fourier Neural Operator (FNO; [25]) learns this map in Fourier space, giving fast and accurate predictions, while Transolver ([26]) replaces Fourier layers with a physics-attention Transformer that can effectively capture complex physical correlations. Both models are deterministic and they are not designed for dynamics that are inherently stochastic. Recent work has sought to introduce stochasticity by fitting probabilistic surrogates—such as stochastic processes, graphical models, or stochastic Koopman operators [36, 37]—but these efforts typically focus on low-dimensional settings or assume simple Gaussian statistics, limiting their applicability to high-dimensional, strongly nonlinear systems. Researchers have also adapted neural operator-based models for DA tasks [38–40]. However, these adaptations are typically tailored to narrow use cases—such as arbitrary-resolution data assimilation—and have not demonstrated robust generalization across diverse real-world scenarios.

Generative model-based methods   Recent works have leveraged score-based methods, in particular diffusion models, for dynamical system modeling. For instance, the Conditional Diffusion Schrödinger Bridge (CDSB) framework extends diffusion models to conditional simulation and demonstrates strong performance on filtering in state-space models [41]. Score-based data assimilation (SDA) is a recent data-driven approach that employs score-based diffusion models to estimate state trajectories in dynamical systems [28]. 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 [42], SDA can generate entire state trajectories in zero-shot and non-autoregressive manners. However, SDA captures state transition implicitly from state concatenation, the learned dynamics of SDA may lack physical interpretability, leading to potential inaccuracies. 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 D.1. Additionally, non-autoregressive diffusion models may struggle with long sequence forecasting in high-dimensional systems, though autoregressive diffusion approaches, like GenCast [43], have shown promising results in such settings. PDEDiff extends SDA by introducing a universal amortized conditional diffusion framework for PDE dynamics, where a conditional score network is trained to predict local state transitions given variable-length histories, enabling both forecasting and data assimilation across different PDE systems [21]. Another closely related line of work is the stochastic interpolant-based forecasting framework recently proposed in [31], which formulates the prediction task as conditional sampling of future system states given the current state, using a fictitious, non-physical stochastic process governed by a learned SDE. While this approach enables probabilistic forecasting, it does not incorporate observation-driven corrections during inference. In contrast, FlowDAS augments the stochastic interpolant dynamics with a measurement-consistent correction term at each step, enabling integration of observational data and thus unifying generative modeling with data assimilation.

2.2 Stochastic Interpolants

Stochastic interpolants [29–31] is a generative modeling framework that unifies flow-based and diffusion-based models, providing 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 target state $\boldsymbol{X}_1 \sim q(\boldsymbol{X}_1)$. A stochastic interpolant can be described as:

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

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 $\alpha_1 = \beta_0 = \sigma_1 = 0$, 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, 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 [31]:

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

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 $\boldsymbol{R}_s = \dot{\alpha}_s \boldsymbol{X}_0 + \dot{\beta}_s \boldsymbol{X}_1 + \dot{\sigma}_s \boldsymbol{W}_s$. Furthermore, the drift term $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ is related to the score function $\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)$:

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) = \frac{\boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)}{\lambda_s \beta_s}, \tag{6}$$

where λ_s and $\boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ are defined to control the score-based dynamics (Appendix A.2).

Building on this framework, stochastic interpolants can estimate the transition $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ by evolving a latent path $\boldsymbol{X}_s$ smoothly from $s = 0$ (current state) to $s = 1$ (next state) in dynamical systems. It provides a probabilistic yet compact surrogate that is more aligned with the true step-to-step evolution of complex dynamical systems. While stochastic interpolants capture complex system dynamics, they do not yet enforce consistency with measurements.

3 Method

3.1 Stochastic Interpolants for Data Assimilation

Using stochastic interpolants as the engine of data assimilation is attractive but *not* a plug-and-play replacement for existing forward models: it raises *three fundamental challenges* that FlowDAS must overcome. Below we formulate each challenge and describe the corresponding FlowDAS solution.

Challenge I: Observation-consistent 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 (6) with boundary conditions $\boldsymbol{X}_0, \boldsymbol{X}_1 = \boldsymbol{x}_k, \boldsymbol{x}_{k+1}$. Moreover, DA forward surrogate must respect the observation $\boldsymbol{y}_{k+1}$ while drawing the future state $\boldsymbol{x}_{k+1}$. For stochastic interpolants, this requires an *observation-conditioned* SDE. FlowDAS augments the original drift $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ via Bayes' rule (Appendix A.1):

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) = \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}. \tag{7}$$

Challenge II: Estimating the observation-informed drift   The term $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ captures the observation information, however, it is intractable because the observation model only directly links $\boldsymbol{y} = \boldsymbol{y}_{k+1}$ and $\boldsymbol{X}_1 = \boldsymbol{x}_{k+1}$. FlowDAS produces unbiased estimate of it by Monte-Carlo marginalization, i.e., integrating with respect to $\boldsymbol{X}_1$, which can be approximated by J Monte Carlo samples $\boldsymbol{X}_1^{(j)} \sim p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$:

$$\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) = \frac{\nabla \mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y} \mid \boldsymbol{X}_1)]}{\mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y} \mid \boldsymbol{X}_1)]} \approx \sum_{j=1}^J w_j \nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}), \tag{8}$$

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 $w_j = p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}) / \sum_{j=1}^J p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})$. We leave detailed derivation in Appendix A.3.

Challenge III: Efficient 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. FlowDAS accelerates this step with low-order stochastic integrators:

First-order Milstein method [44, p. 317]:

$$\hat{\boldsymbol{X}}_1 = \boldsymbol{X}_s + \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)(1-s) + \int_s^1 \sigma_\tau \, \mathrm{d}\boldsymbol{W}_\tau \tag{9}$$

and second-order stochastic Runge-Kutta method [44, p. 324]:

$$\hat{\boldsymbol{X}}_1' = \boldsymbol{X}_s + \frac{\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \boldsymbol{b}_1(\hat{\boldsymbol{X}}_1, \boldsymbol{X}_0)}{2}(1-s) + \int_s^1 \sigma_\tau \, \mathrm{d}\boldsymbol{W}_\tau. \tag{10}$$

Both approximations introduce slight numerical bias (i.e., $\mathcal{O}((1-s)^2)$ and $\mathcal{O}((1-s)^3)$, respectively) but significantly accelerated sampling speed. Appendix C.2 further compares these approximations.

Advantages of FlowDAS.   Compared with other data-driven surrogates, FlowDAS offers two key advantages.

- **Observation-consistent probabilistic forecasts.** Neural operators such as FNO and Transolver learn a deterministic map $\boldsymbol{x}_k \mapsto \boldsymbol{x}_{k+1}$; they yield a single forecast and require a separate optimisation step to reconcile that forecast with the observation $\boldsymbol{y}_{k+1}$. FlowDAS learns the full conditional distribution $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{y}_{k+1})$. It therefore generates an ensemble of forecasts that are already consistent with the incoming measurement, eliminating the need for any post-hoc update.
- **Local and physics-aligned transport.** FlowDAS directly learns a *local* bridge between adjacent states, so the diffusion process spans a short distance in state space. Training therefore remains numerically stable and provides clear physical insight into one-step dynamics. By contrast, score-based diffusion models construct a global path from Gaussian noise to data; the network must master a far longer transformation and learn the state transition dynamics through channel correlations, which often obscures physical interpretability and can degrade accuracy.

3.2   Implementation Details

Training   We train a user-defined neural network as the drift model $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$, which outputs the velocity given the current interpolant $\boldsymbol{X}_s$ and conditioned on the initial state $\boldsymbol{X}_0$. The network architecture of the drift model may vary across tasks, for example, a Multi-Layer Perceptron (MLP) is adopted for the low-dimensional Lorenz-63 system in Section 4.1, a U-Net for high-dimensional fluid system governed by incompressible Navier-Stokes equations in Section 4.2, and a FNO for the weather forecasting task in Section 4.3. The model is optimized using collected trajectories $\boldsymbol{x}_{0:K}$, minimizing an empirical loss between predicted and target velocities over sampled interpolation times $s \in [0, 1]$. The discrepancy metric (e.g., ℓ_1 or squared ℓ_2 norm, etc.) is also task-dependent.

Inference   Given a trained drift model $\hat{\boldsymbol{b}}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$, FlowDAS performs inference in an autoregressive manner to estimate the state trajectory $\hat{\boldsymbol{x}}_{1:K}$. At each time step k, we begin from the known state $\hat{\boldsymbol{x}}_k$ and predict the next state $\hat{\boldsymbol{x}}_{k+1}$ using a discretized stochastic interpolant over a time grid $s_0 = 0 < s_1 < \cdots < s_N = 1$. We use $N = 500$ in our experiments. We set the interpolant state $\boldsymbol{X}_{s_0} \leftarrow \hat{\boldsymbol{x}}_k$ and retrieve the next observation $\boldsymbol{y}_{k+1}$. For each interpolation step s_n, we simulate a forward transition using the learned drift and noise. To enforce observation consistency, we generate J posterior endpoint samples $\hat{\boldsymbol{X}}_1^{(j)}$, and compute their likelihoods under the measurement model, i.e., $\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$. A softmax over these values gives importance weights $\{w_{1:J}\}$. We then apply a correction to the interpolant using a weighted gradient $-\zeta_n \nabla_{\boldsymbol{X}_{s_n}} \sum_{j=1}^J w_j \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$, where ζ_n denotes the step size. After reaching $s = 1$, we set $\hat{\boldsymbol{x}}_{k+1} \leftarrow \boldsymbol{X}_{s_N}$ and proceed to the next time step. Repeating this procedure over all K yields the full trajectory.

Conditioning   While the original stochastic interpolants framework which only models the $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$, we take in a sequence of previous states to achieve the transition from $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}...\boldsymbol{x}_{k-l})$ to $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{x}_{k-1}...\boldsymbol{x}_{k-l})$ inspired by [45–48], thus achieving probabilistic prediction conditioned on several previous states. This straightforward extension allows FlowDAS to handle non-Markovian dynamics and empirically leads to markedly improved performance on weather-forecasting task (Section 4.3).

4 Experiments and Results

This section evaluates our proposed framework, FlowDAS, on a range of low- and high-dimensional stochastic dynamical systems, including low-dimensional problems with high-order observation models, such as the double-well potential (Appendix D.1.5) and the chaotic Lorenz 1963 system, as well as high-dimensional tasks involving the incompressible Navier-Stokes equations and a realistic problem: Particle Image Velocimetry (PIV). Additionally, we demonstrate its applicability on a real-world large-scale weather forecasting task, where governing dynamics are not available. These results underscore the versatility and robustness of FlowDAS.

4.1 Lorenz 1963

In this experiment, we evaluate the performance of FlowDAS using the Lorenz 1963 system, a simplied mathematical model for atmospheric convection that is widely studied in the DA community [28, 49, 50]. 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):

$$\begin{aligned}
\mathrm{d}a/\mathrm{d}t &= \mu(b - a) + \xi_1, \\
\mathrm{d}b/\mathrm{d}t &= a(\rho - c) - b + \xi_2, \\
\mathrm{d}c/\mathrm{d}t &= ab - \tau c + \xi_3,
\end{aligned} \tag{11}$$

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 Brownian process noise, with each component having a standard deviation $\sigma = 0.025$. This chaotic system poses a significant challenge for numerical methods, so we use the fourth-order Runge-Kutta (RK4) method [51] to simulate its state transition (see Appendix D.2 for details).

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

$$y = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\eta} = \arctan(a) + \eta, \tag{12}$$

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

Dataset and experiments 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 D.2. During inference, we independently estimate 64 trajectories over 15 time steps using FlowDAS and the baseline methods. A total of $L = 1$ previous state is conditioned for each autoregressive generation. 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 BPF [23]. Appendix D.2 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 $\mathbb{E}_{q(\boldsymbol{x}_{1:K}|\boldsymbol{y}_{1:K})}[\log p(\boldsymbol{x}_{2:K} \mid \boldsymbol{x}_1)]$; the expectation of log data likelihood $\mathbb{E}_{q(\boldsymbol{x}_{1:K}|\boldsymbol{y}_{1:K})}[\log p(\boldsymbol{y}_{1:K} \mid \boldsymbol{x}_{1:K})]$; the Wasserstein distance [52] $W_1(\cdot, \cdot)$ 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.

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

Table 1: **Data assimilation of Lorenz 1963 system.** 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**.
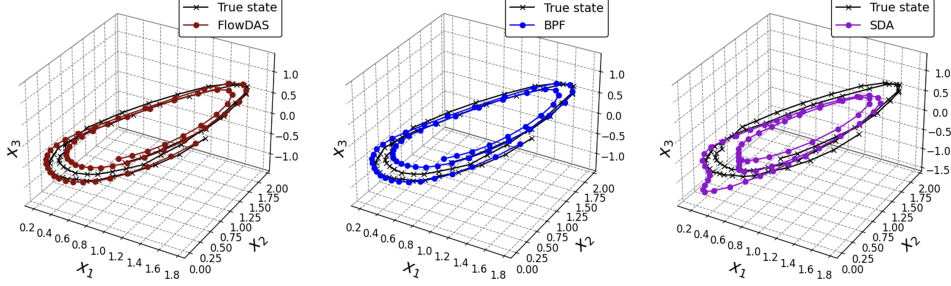
Figure 2: **Data assimilation of Lorenz 1963 system.** FlowDAS achieved results comparable to the state-of-the-art model-based BPF method, significantly outperforming the data-driven SDA method in recovering the underlying dynamics of this chaotic system. This highlights the efficiency and robustness 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 (11), respectively.

Results 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. Appendix D.2 provides additional comparisons and results.

The success of FlowDAS is primarily due to its accurate mapping from current to future states ($\boldsymbol{x}_k \rightarrow \boldsymbol{x}_{k+1}$). Despite lacking explicit transition equations, FlowDAS effectively captures the system dynamics through stochastic interpolants, enabling a closer approximation of state trajectories 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 D.1.5) 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,

$$d\boldsymbol{\omega} + \boldsymbol{v} \cdot \nabla \boldsymbol{\omega} \, dt = \nu \Delta \boldsymbol{\omega} \, dt - \alpha \boldsymbol{\omega} \, dt + \varepsilon \, d\boldsymbol{\xi}, \tag{13}$$

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 D.3.1.

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

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{\omega}) + \boldsymbol{\eta}, \tag{14}$$

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

Dataset and experiments In this experiment, system dynamics are simulated by solving Equation (13) using a pseudo-spectral method [53] 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.

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.
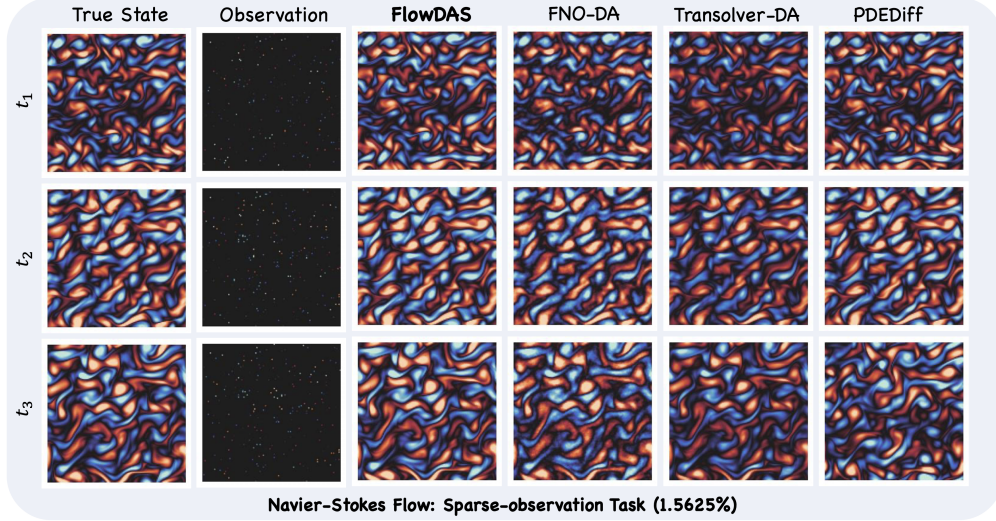
Figure 3: **Data assimilation of incompressible Navier-Stokes flow.** The positive values (red) of the state, i.e., vorticity field, indicate clockwise rotation and negative values (blue) indicate counter-clockwise rotation. FlowDAS achieved results with more accurate details and higher accuracy than all baselines, showing the efficiency of FlowDAS 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.8.

	$32^2 \rightarrow 128^2$	$16^2 \rightarrow 128^2$	5%	1%
FLOWDAS	**0.077** $\pm$ 0.005	**0.174** $\pm$ 0.006	**0.156** $\pm$ 0.005	**0.373** $\pm$ 0.012
PDEDIFF	0.112 $\pm$ 0.017	0.242 $\pm$ 0.035	0.221 $\pm$ 0.024	0.575 $\pm$ 0.068
TRANSOLVER-DA	0.597 $\pm$ 0.001	0.657 $\pm$ 0.001	0.660 $\pm$ 0.001	0.669 $\pm$ 0.001
FNO-DA	0.653 $\pm$ 0.001	0.732 $\pm$ 0.001	0.695 $\pm$ 0.001	0.753 $\pm$ 0.001

Table 2: **RMSE ($\pm$ std) of FlowDAS and baselines on incompressible Navier–Stokes super-resolution and sparse-observation tasks.** Each value is computed from 50 independently sampled states, with 30 repeated runs per state. All results correspond to single-step assimilation. Figure S.12 presents the RMSE and standard deviations evaluated along a FlowDAS-generated trajectory.

The model is evaluated on four unseen datasets ($2\times$ super-resolution, $2\times$ inpainting), with 64 samples for each configuration. A total of $L = 10$ previous states are conditioned for each autoregressive generation.

Baselines and metrics We benchmark FlowDAS against three data-driven baselines. First, we implement PDEDiff solver with a fixed eleven-step window (10 previous states as conditions). Second, we adapt the deterministic neural operators FNO and Transolver to the DA setting, namely 'FNO-DA' and 'Transolver-DA'. Each network is trained to minimise RMSE between adjacent states; at inference time the raw prediction $x'_{k+1} = f_\theta(x_k)$ is reconciled with the observation y_{k+1} by solving $\hat{x}_{k+1} = \arg\min_x \alpha_1 \|x - x'_{k+1}\|_2^2 + \alpha_2 \|y_{k+1} - \mathcal{A}(x)\|_2^2$, an optimization step analogous to a Kalman update, where α_1 and α_2 are weighting coefficients. The α_1 and α_2 are chosen by grid searching, and we report the best results. 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 baselines and FlowDAS are provided in Appendix D.3.3.

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 the physical characteristics of the fluid are accurately preserved. Appendix D.3.5 and Figure S.8 provide the definitions and results for the kinetic energy spectrum metric.
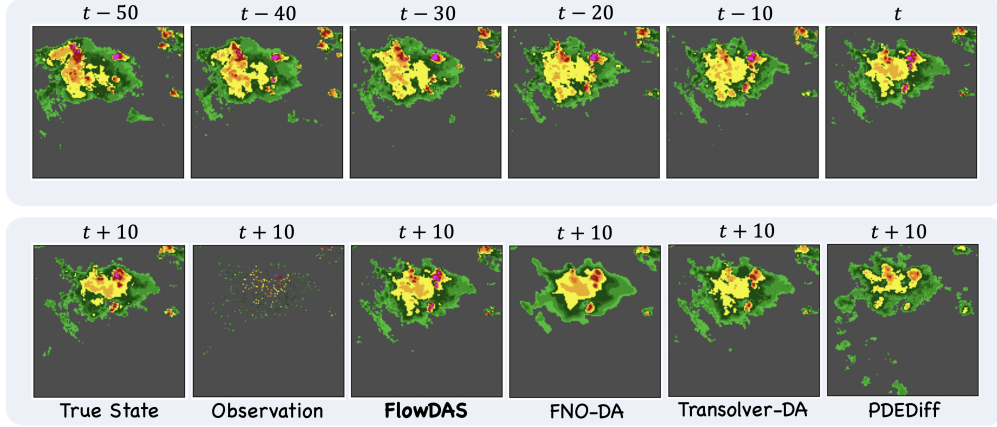
8

Figure 4: **Data assimilation of weather forecasting on SEVIR Vertical Integrated Liquid dataset under sparse observations.** All DA models take previous six states (displayed in the first row; $t-50$ min to t min) as conditions and estimate the future state at $t + 10$ min. FlowDAS preserves storm cores and spatial texture better than score-based diffusion models and neural operator-based methods.

Results Figure 3 compares FlowDAS with baselines under the 1.5625% sparse-observation setting. See Appendix D.3 for additional results. Quantitative metrics, including RMSE and kinetic energy spectrum comparisons, are provided in Table 2 and Figure S.8. Our method consistently outperforms all baselines in terms of reconstruction accuracy, capturing high-frequency information with greater precision. This advantage is further validated by the kinetic energy spectrum in Figure S.8. The RMSE scores in Table 2 further highlight the effectiveness of FlowDAS in accurately estimating the underlying fluid dynamics from observational data. Appendix D.3.7 provides additional results on more challenging observation cases. Appendix D.3.8 presents the performance of FlowDAS when evaluated along an estimated trajectory.

Particle Image Velocimetry To demonstrate that our framework generalizes beyond the synthetic incompressible NS setting, we also apply FlowDAS to Particle Image Velocimetry, a widely used optical technique that infers sparse planar velocity vectors by tracking tracer particles in sequential images. In this scenario the DA problem is to reconstruct the full vorticity (or velocity) field from noisy, sparsely sampled velocities. We generate synthetic PIV frames from the same NS simulations, extract particle displacements with a standard PIV pipeline, and feed the resulting observations into the pretrained FlowDAS model. A comparison with baseline methods confirms that FlowDAS maintains its performance advantage in this practical setting. The full experimental protocol, quantitative metrics, and qualitative results are reported in Appendix D.4.

4.3 Weather Forecasting

This experiment evaluates FlowDAS on a large-scale, real-world weather–forecasting problem using the Storm EVent Imagery and Radar (SEVIR) dataset [54]. SEVIR provides multi-modal observations of severe convective storms across the continental United States. We focus on the *Vertically Integrated Liquid* (VIL) product, a 2-D proxy for precipitation intensity. Each sample is a 128×128 grid covering $384\,\text{km} \times 384\,\text{km}$ at 2 km resolution and recorded every 10 min for four hours. Similar to [48], six consecutive VIL frames ($t-50$ min to t min) constitute the input, and the task is to predict the next frame at $t + 10$ min. In our DA settings, the state variable is the VIL field x, and the observation is aquired by randomly sampling 10% of grid cells to emulate sparse radar coverage. A total of $L = 6$ previous states are conditioned for next-frame generation. We split the dataset into 80%/10%/10% for training, validation, and testing.

Baselines and metrics We compare FlowDAS with the PDEDiff solver (fixed window size of 7 with 6 previous states as conditions). Neural-operator baselines, e.g., FNO and Transolver, are similarly obtained as Navier-Stokes experiments. Both models takes a concatenation (along channel dimension) of states as inputs and output the state at target time step. Performance is measured by RMSE and the Critical Success Index (CSI) at thresholds of τ_{20} and τ_{40} dBZ, which are standard verification metrics for precipitation nowcasting.

METHOD	RMSE ↓	CSI(τ_{20}) (0.3) ↑	CSI(τ_{40}) (0.5) ↑
FLOWDAS	**0.053±0.004**	**0.746±0.022**	**0.614±0.044**
PDEDIFF	0.071±0.007	0.549±0.033	0.387±0.065
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

Table 3: **Comparison of FlowDAS and other baselines on RMSE and Critical Success Index (CSI) of the weather forecasting task.** All metrics are averaged over 30 repeated runs with standard deviations.

Inplementation Details Different from [31], we adopted FNO [25] rather than U-Net as the backbone of the stochastic interpolants framework. We provide the comparison between the FNO-based and the U-Net-based FlowDAS in the Appendix D.5, and found that the FNO-based FlowDAS demonstrates better performance.

Results FlowDAS achieves lower RMSE and higher CSI at both thresholds, indicating more accurate intensity estimates and better hit rates for heavy precipitation as shown in Appendix D.5. Qualitative comparisons in Figure 4 (and Figures S.15 to S.17) show that FlowDAS reconstructs coherent *precipitation structures* and *peak intensities* that other baselines either smooth out or mis-localize. These results demonstrate that the stochastic-interpolant surrogate scales to high-resolution, real meteorological data, even when underlying governed PDEs are unknown, and retains its advantage over diffusion models and neural operators baselines.

5   Limitations and Future Work

FlowDAS has so far been validated on controlled dynamical systems (e.g., Lorenz–63, Navier–Stokes), and its generalizability to more complex, real-world environments such as numerical weather prediction remains to be evaluated. In particular, we have not yet examined its performance under Sim2Real conditions or applied it to operational-scale, high-dimensional systems such as those targeted by GEN_BE [55]. Although FlowDAS supports probabilistic inference through sampling, its current sampling speed is a limitation. A promising direction is to employ post-hoc distillation, training a lightweight surrogate model to approximate the FlowDAS sampler. Another avenue is to explore hybrid formulations that combine stochastic interpolants with variational inference techniques to further improve computational efficiency.

6   Conclusion and Future Work

This work introduced **FlowDAS**, a stochastic interpolant-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 weather forecasting—FlowDAS demonstrated strong performance in recovering accurate state variables from sparse, noisy observations. These results highlight Flow-DAS as a robust alternative to traditional model-driven methods (e.g., particle filters) and data-driven approaches (e.g., score-based diffusion models and neural operators) for data assimilation, offering improved accuracy, efficiency, and adaptability.

References

[1] Marc Bocquet, H Elbern, H Eskes, M Hirtl, R Žabkar, GR Carmichael, J Flemming, A Inness, M Pagowski, JL 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.

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

[3] James 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.

[4] James A Cummings and Ole Martin Smedstad. Variational data assimilation for the global ocean. In *Data assimilation for atmospheric, oceanic and hydrologic applications (Vol. II)*, pages 303–343. Springer, 2013.

[5] M. J. Werner, K. Ide, and D. Sornette. Earthquake forecasting based on data assimilation: Sequential monte carlo methods for renewal processes, 2009.

[6] Arundhuti Banerjee, Ylona van Dinther, and Femke C Vossepoel. On parameter bias in earthquake sequence models using data assimilation. *Nonlinear Processes in Geophysics*, 30(2):101–115, 2023.

[7] Bin Wang, Xiaolei Zou, and Jiang Zhu. Data assimilation and its applications. *Proceedings of the National Academy of Sciences*, 97(21):11143–11144, 2000.

[8] Marc Bocquet, Carlos A Pires, and Lin Wu. Beyond gaussian statistical modeling in geophysical data assimilation. *Monthly Weather Review*, 138(8):2997–3023, 2010.

[9] Ross N Bannister. A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):607–633, 2017.

[10] Sebastian Reich and Colin Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[11] Yuichiro Taira, Shinichi Sagara, and Masahiro Oya. Model-based motion control for underwater vehicle-manipulator systems with one of the three types of servo subsystems. *Artificial life and robotics*, 25:133–148, 2020.

[12] Steven L Brunton and Bernd R Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67(5):050801, 2015.

[13] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51(1):357–377, 2019.

[14] Florence 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.

[15] Alan J Geer, Katrin Lonitz, Peter Weston, Masahiro Kazumori, Kozo Okamoto, Yanqiu Zhu, Emily Huichun Liu, Andrew Collard, William Bell, Stefano 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.

[16] Nils Gustafsson, Tijana Janjić, Christoph Schraff, Daniel Leuenberger, Martin Weissmann, Hendrik Reich, Pierre Brousseau, Thibaut Montmerle, Eric Wattrelot, Antonín 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.

[17] Matthew Rodell, PR Houser, UEA Jambor, J Gottschalck, Kieran Mitchell, C-J Meng, Kristi 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.

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

[19] Steven J Fletcher. *Data assimilation for the geosciences: From theory to application*. Elsevier, 2017.

[20] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[21] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard Turner, and Emile Mathieu. On conditional diffusion models for pde simulations. *Advances in Neural Information Processing Systems*, 37:23246–23300, 2024.

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

[23] Neil J Gordon, David J Salmond, and Adrian FM 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.

[24] Peter Bickel, Bo Li, and Thomas 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.

[25] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[26] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries, 2024.

[27] Yongquan Qu, Juan Nathaniel, Shuolin Li, and Pierre 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.

[28] François Rozet and Gilles Louppe. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

[29] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

[30] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[31] Yifan Chen, Mark Goldstein, Mengjian Hua, Michael Samuel Albergo, Nicholas Matthew Boffi, and Eric Vanden-Eijnden. Probabilistic forecasting with stochastic interpolants and Föllmer processes. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix 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 2024.

[32] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.

[33] K. Senne. Stochastic processes and filtering theory. *IEEE Transactions on Automatic Control*, 17(5):752–753, 1972.

[34] Mohinder Grewal and Angus Andrews. Kalman filtering: Theory and applications. 01 1985.

[35] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[36] Mathias Wanner and Igor Mezić. Robust approximation of the stochastic koopman operator, 2022.

[37] Meng Zhao and Lijian Jiang. Data-driven probability density forecast for stochastic dynamical systems, 2022.

[38] Kun Chen, Peng Ye, Hao Chen, Tao Han, Wanli Ouyang, Tao Chen, LEI BAI, et al. Fnp: Fourier neural processes for arbitrary-resolution data assimilation. *Advances in Neural Information Processing Systems*, 37:137847–137872, 2024.

[39] Xiaoze Xu, Xiuyu Sun, Wei Han, Xiaohui Zhong, Lei Chen, Zhiqiu Gao, and Hao Li. Fuxi-da: A generalized deep learning data assimilation framework for assimilating satellite observations. *npj Climate and Atmospheric Science*, 8(1):156, 2025.

[40] Yanfei Xiang, Weixin Jin, Haiyu Dong, Mingliang Bai, Zuliang Fang, Pengcheng Zhao, Hongyu Sun, Kit Thambiratnam, Qi Zhang, and Xiaomeng Huang. Adaf: An artificial intelligence data assimilation framework for weather forecasting. *arXiv preprint arXiv:2411.16807*, 2024.

[41] Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Conditional simulation using diffusion schrödinger bridges, 2022.

[42] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

[43] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

[44] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[45] Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in neural information processing systems*, 35:23371–23385, 2022.

[46] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10459–10469, June 2023.

[47] Zhicheng Zhang, Junyao Hu, Wentao Cheng, Danda Paudel, and Jufeng Yang. Extdm: Distribution extrapolation diffusion model for video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19310–19320, June 2024.

[48] Zhihan Gao, Xingjian Shi, Boran Han, Hao Wang, Xiaoyong Jin, Danielle Maddix, Yi Zhu, Mu Li, and Yuyang Bernie Wang. Prediff: Precipitation nowcasting with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:78621–78656, 2023.

[49] Edward Norton Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[50] Feng Bao, Zezhong Zhang, and Guannan Zhang. A score-based nonlinear filter for data assimilation, 2023.

[51] Nor Aida Zuraimi Md Noar, Nur Ilyana Anwar Apandi, and Norhayati 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.

[52] Cédric Villani. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[53] Roger Peyret. *Spectral methods for incompressible viscous flow*, volume 148. Springer, 2002.

[54] Mark Veillette, Siddharth Samsi, and Chris Mattioli. Sevir: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *Advances in Neural Information Processing Systems*, 33:22009–22019, 2020.

[55] Gael Descombes, T Auligné, Francois Vandenberghe, DM Barker, and Jerome Barre. Generalized background error covariance matrix model (gen_be v2. 0). *Geoscientific Model Development*, 8(3):669–696, 2015.

[56] Weimin Bai, Siyi Chen, Wenzheng Chen, and He Sun. Blind inversion using latent diffusion priors, 2024.

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

[58] Feng Bao, Zezhong Zhang, and Guannan Zhang. An ensemble score filter for tracking high-dimensional nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 432:117447, 2024.

[59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[60] Zachary J Taylor, Roi Gurka, Gregory A Kopp, and Alex Liberzon. Long-duration time-resolved PIV to study unsteady aerodynamics. *IEEE Transactions on Instrumentation and Measurement*, 59(12):3262–3269, 2010.

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

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

[63] Colin King, Edmond Walsh, and Ronan Grimes. PIV measurements of flow within plugs in a microchannel. *Microfluidics and Nanofluidics*, 3:463–472, 2007.

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

[65] Fahrettin Gökhan Ergin, Bo Beltoft Watz, and Nicolai Fog Gade-Nielsen. A review of planar PIV systems and image processing tools for lab-on-chip microfluidics. *Sensors*, 18(9):3090, 2018.

[66] Chia-Yuan Chen, Raúl Antón, Ming-yang Hung, Prahlad Menon, Ender A Finol, and Kerem 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.

[67] FPP Tan, A Borghi, RH Mohiaddin, NB Wood, S Thom, and XY Xu. Analysis of flow patterns in a patient-specific thoracic aortic aneurysm model. *Computers & Structures*, 87(11-12):680–690, 2009.

[68] Can Özcan, Özgür Kocatürk, Civan Işlak, and Cengizhan Öztürk. Integrated particle image velocimetry and fluid–structure interaction analysis for patient-specific abdominal aortic aneurysm studies. *BioMedical Engineering OnLine*, 22(1):113, 2023.

[69] Christian Lagemann, Kai Lagemann, Sach Mukherjee, and Wolfgang Schröder. Deep recurrent optical flow learning for particle image velocimetry data. *Nature Machine Intelligence*, 3:641 – 651, 2021.

A Mathematical Derivation

A.1 Conditional Drift

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 (6) 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 (Appendix A.2):

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) = \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)}{\lambda_s \beta_s}. \tag{S.15}$$

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:

$$\begin{aligned}
\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{y}, \boldsymbol{X}_0) &= \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{y}, \boldsymbol{X}_0)}{\lambda_s \beta_s} \\
&= \frac{c_s(\boldsymbol{X}_s, \boldsymbol{X}_0)}{\beta_s} + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) + \nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)}{\lambda_s \beta_s} \\
&= \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) + \frac{\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)}{\lambda_s \beta_s}.
\end{aligned} \tag{S.16}$$

A.2 Relating Drift and Score

By extending Stein's formula [30], we have:

$$\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0) = -\frac{1}{\sqrt{s}\sigma_s} \mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{z} \mid \boldsymbol{X}_s], \tag{S.17}$$

where $\boldsymbol{X}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \boldsymbol{X}_1 + \sqrt{s}\sigma_s \boldsymbol{z}$. In addition,

$$\begin{aligned}
\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) &= \dot{\alpha}_s \boldsymbol{X}_0 + \dot{\beta}_s \mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{X}_1 \mid \boldsymbol{X}_s] + \sqrt{s}\dot{\sigma}_s \mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{z} \mid \boldsymbol{X}_s], \\
\boldsymbol{X}_s &= \alpha_s \boldsymbol{X}_0 + \beta_s \mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{X}_1 \mid \boldsymbol{X}_s] + \sqrt{s}\sigma_s \mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{z} \mid \boldsymbol{X}_s].
\end{aligned} \tag{S.18}$$

Solve $\mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{z} \mid \boldsymbol{X}_s]$ from Equation (S.18):

$$\mathbb{E}_{\boldsymbol{X}_0 \sim \pi(\boldsymbol{X}_0)}[\boldsymbol{z} \mid \boldsymbol{X}_s] = \frac{\beta_s \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) - \dot{\beta}_s \boldsymbol{X}_s - (\beta_s \dot{\alpha}_s - \dot{\beta}_s \alpha_s) \boldsymbol{X}_0}{\sqrt{s}(\dot{\sigma}_s \beta_s - \sigma_s \dot{\beta}_s)}. \tag{S.19}$$

Define $\lambda_s = \frac{1}{\sqrt{s}(\dot{\sigma}_s \beta_s - \sigma_s \dot{\beta}_s)}$ and $\boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) = \dot{\beta}_s \boldsymbol{X}_s + (\beta_s \dot{\alpha}_s - \dot{\beta}_s \alpha_s) \boldsymbol{X}_0$, and insert Equation (S.19) into Equation (S.17), we relate the score function $\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0)$ and the drift $\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)$ [31] by:

$$\nabla \log p(\boldsymbol{X}_s \mid \boldsymbol{X}_0) = \lambda_s[\beta_s \boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) - \boldsymbol{c}_s(\boldsymbol{X}_s, \boldsymbol{X}_0)]. \tag{S.20}$$

A.3 Estimation of the Gradient Log Likelihood of Observation

The term $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ in Equation (7) 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 $\boldsymbol{X}_1$:

$$\begin{aligned}
\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) &= \frac{\nabla p(\boldsymbol{y}|\boldsymbol{X}_s, \boldsymbol{X}_0)}{p(\boldsymbol{y}|\boldsymbol{X}_s, \boldsymbol{X}_0)} = \frac{\nabla \int p(\boldsymbol{y}|\boldsymbol{X}_1)p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)\, \mathrm{d}\boldsymbol{X}_1}{\int p(\boldsymbol{y}|\boldsymbol{X}_1)p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)\, \mathrm{d}\boldsymbol{X}_1} \\
&= \frac{\nabla \mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y}|\boldsymbol{X}_1)]}{\mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y}|\boldsymbol{X}_1)]}.
\end{aligned} \tag{S.21}$$

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

$$\mathbb{E}_{\boldsymbol{X}_1 \sim p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)}[p(\boldsymbol{y}|\boldsymbol{X}_1)] \approx \frac{1}{J} \sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}). \tag{S.22}$$

Algorithm 1 Training

1: **Input:** Dataset $\boldsymbol{x}_{0:K}$; minibatch size $K' \leq K$; coefficients α_s, β_s, σ_s
2: **repeat**
3:     Compute $\tilde{\boldsymbol{I}}_s^k$ and $\boldsymbol{R}_s^k$ using (S.25) for $k \in \mathcal{B}_{K'}$
4:     Compute the empirical loss $\mathcal{L}_b^{\mathrm{emp}}(\hat{\boldsymbol{b}})$ in Equation (S.24)
5:     Take the gradient step on $\mathcal{L}_b^{\mathrm{emp}}(\hat{\boldsymbol{b}})$ to update $\hat{\boldsymbol{b}}_s$
6: **until** converged
7: **return** drifts $\hat{\boldsymbol{b}}_s$

Plug Equation (S.22) into Equation (S.21):

$$
\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0) \approx \frac{\nabla \frac{1}{J} \sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})}{\frac{1}{J} \sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})} = \frac{\sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}) \nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})}{\sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})}
$$
$$
= \sum_{j=1}^{J} w_j \nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}),
\tag{S.23}
$$

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 $w_j = p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)}) / \sum_{j=1}^{J} p(\boldsymbol{y} \mid \boldsymbol{X}_1^{(j)})$.

B   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:

$$
\mathcal{L}_b^{\mathrm{emp}}(\hat{\boldsymbol{b}}) = \frac{1}{K'} \sum_{k \in \mathcal{B}_{K'}} \int_0^1 \ell\big(\hat{\boldsymbol{b}}_s(\tilde{\boldsymbol{I}}_s^k, \boldsymbol{x}_k), \boldsymbol{R}_s^k\big) \, \mathrm{d}s,
\tag{S.24}
$$

where $\ell(\cdot, \cdot)$ denotes a user-chosen discrepancy metric, depending on the experiment. $\mathcal{B}_{K'} \subset \{0:K\}$ is a subset of indices of cardinality $K' \leq K$, with

$$
\begin{aligned}
\tilde{\boldsymbol{I}}_s^k &= \alpha_s \boldsymbol{x}_k + \beta_s \boldsymbol{x}_{k+1} + \sqrt{s}\sigma_s \boldsymbol{z}_k \\
\boldsymbol{R}_s^k &= \dot{\alpha}_s \boldsymbol{x}_k + \dot{\beta}_s \boldsymbol{x}_{k+1} + \sqrt{s}\dot{\sigma}_s \boldsymbol{z}_k,
\end{aligned}
\tag{S.25}
$$

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 (S.24) 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 $\boldsymbol{y}_{L:K}$ 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 Equations (9) and (10) for J times. Then, we move one step further towards s_N on $\boldsymbol{X}_{s_n}$ by solving Equation (7) 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 [42, 56]. The chosen J values crossed all experiments, reported in Table S.4, were sufficient large for stable performance, with subtle variation across the dimensionality of problems. An ablation study, detailed in Appendix C.1, further confirmed that larger J offers diminishing returns. Overall, Algorithm 2 summarizes the inference process.

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:

Algorithm 2 Inference

1: **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
2: Set $\hat{\boldsymbol{x}}_{L-1} \leftarrow \boldsymbol{x}_{L-1}$
3: Set the $(\Delta s)_n = s_{n+1} - s_n$, $n = 0 : N - 1$
4: **for** $k = L - 1$ **to** $K - 1$ **do**
5: $\quad \boldsymbol{X}_{s_0}, \boldsymbol{y} \leftarrow \hat{\boldsymbol{x}}_k, \boldsymbol{y}_{k+1}$
6: $\quad$ **for** $n = 0$ **to** $N - 1$ **do**
7: $\quad\quad \boldsymbol{X}'_{s_{n+1}} = \boldsymbol{X}_{s_n} + \hat{\boldsymbol{b}}_s(\boldsymbol{X}_{s_n}, \boldsymbol{X}_{s_0})(\Delta s)_n + \sigma_{s_n}\sqrt{(\Delta s)_n}\boldsymbol{z}_n$
8: $\quad\quad \{\hat{\boldsymbol{X}}_1^{(j)}\}_{j=1}^J \leftarrow$ Posterior estimation $(\hat{\boldsymbol{b}}_s, s_n, \boldsymbol{X}_0, \boldsymbol{X}_{s_n})$
9: $\quad\quad \{w_j\}_{j=1}^J \leftarrow$ Softmax $\left(\{ \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2 \}_{j=1}^J \right)$
10: $\quad\quad \boldsymbol{X}_{s_{n+1}} = \boldsymbol{X}'_{s_{n+1}} - \zeta_n \nabla_{\boldsymbol{X}_{s_n}} \sum_{j=1}^J w_j \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{X}}_1^{(j)})\|_2^2$
11: $\quad$ **end for**
12: $\quad \hat{\boldsymbol{x}}_{k+1} \leftarrow \boldsymbol{X}_{s_N}$
13: **end for**
14: **return** $\{\hat{\boldsymbol{x}}_k\}_{k=L}^K$

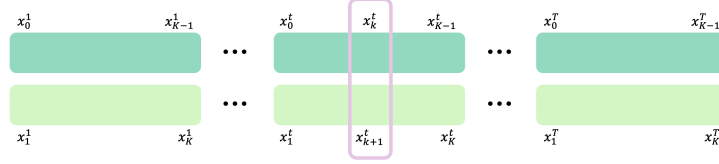Figure S.5: **Structure of training data**. Consecutive states are paired across multiple simulated trajectories to construct the $\tilde{\boldsymbol{I}}_s$ and $\boldsymbol{R}_s$ defined in Equation (S.24) for training the velocity model $\boldsymbol{b}_s$.

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 (S.24) for training the velocity model $\boldsymbol{b}_s$ as described in Algorithm 1.

In summary, we draw many state pairs $(\boldsymbol{X}_0, \boldsymbol{X}_1)$ and estimate the integral over s in Equation (S.24) 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 (S.25) with S samples of $\boldsymbol{z}_k$, so the training loss can be rewritten as:

$$\mathcal{L}_b^{\text{emp}}(\hat{\boldsymbol{b}}) = \frac{1}{K'} \sum_{k \in \mathcal{B}_{K'}} \frac{1}{S} \sum_{s=1}^S \|\hat{\boldsymbol{b}}_s(\tilde{\boldsymbol{I}}_s^k, \boldsymbol{x}_k) - \boldsymbol{R}_s^k\|^2, \tag{S.26}$$

		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
WEATHER FORECASTING		25	0.1

Table S.4: **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.

where we randomly draws s from $U(0,1)$. Note that the training batch size equals $S \times K'$. Figure S.5 illustrates the structure of training data, showing how consecutive states are paired across trajectories.

B.2 FlowDAS Inference Hyperparameters

We present key hyperparameters for FlowDAS in the inference stage, including the Monte Carlo sampling times J and the sampling step size ξ in Table S.4.

C 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.

C.1 Monte Carlo Sampling and An Alternative

In Appendix A.3, we estimate $\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right)$ by

$$\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right) = \frac{\nabla \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]}{\mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right]}, \tag{S.27}$$

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

$$\mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] \approx \frac{1}{J}\sum_{j=1}^{J} p\left(\boldsymbol{y}|\boldsymbol{X}_1^{(j)}\right), \boldsymbol{X}_1^{(j)} \sim p\left(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0\right). \tag{S.28}$$

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:

$$\begin{aligned}\nabla \log p\left(\boldsymbol{y} \mid \boldsymbol{X}_s, \boldsymbol{X}_0\right) &= \nabla \log \int p\left(\boldsymbol{y}|\boldsymbol{X}_1\right) p\left(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0\right) \mathrm{d}\boldsymbol{X}_1 \\ &= \nabla \log \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] \geq \nabla \mathbb{E}_{\boldsymbol{X}_1}\left[\log p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right].\end{aligned} \tag{S.29}$$

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 [57] and given by

$$\left|\log \mathbb{E}_{\boldsymbol{X}_1}\left[p\left(\boldsymbol{y}|\boldsymbol{X}_1\right)\right] - \mathbb{E}_{\boldsymbol{X}_1}\left[\log p(\boldsymbol{y}|\boldsymbol{X}_1)\right]\right| \leq M\gamma_1^1 \tag{S.30}$$

18

		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.5: **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.

where M is a constant that satisfying $|\log Z - \log \mathbb{E}[Z]| \leq M |Z - \mathbb{E}[Z]|$ and $\gamma_1^1 = \mathbb{E}[|Z - \mathbb{E}[Z]|]$. Additionally, because $Z \in [0, \frac{1}{\sqrt{2\pi}\gamma}]$, we have $M < 1$ and $\gamma_1^1 \leq \max |Z - \mathbb{E}[Z]| \leq \frac{1}{\sqrt{2\pi}\gamma}$.

And for this upper-bound $M\gamma_1^1$, when $s \rightarrow 1$, $p(\boldsymbol{X}_1 \mid \boldsymbol{X}_s, \boldsymbol{X}_0)$ 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

$$|\log \mathbb{E}_{\boldsymbol{X}_1}[p(\boldsymbol{y}|\boldsymbol{X}_1)] - \mathbb{E}_{\boldsymbol{X}_1}[\log p(\boldsymbol{y}|\boldsymbol{X}_1)]| \leq \frac{1}{\sqrt{2\pi}\gamma} \tag{S.31}$$

Although this bias is theoretically bounded, it still results in a slight degradation in performance. Table S.5 shows the comparison for the Lorenz experiment to illustrate this point.

Hyperparameters for Monte Carlo Sampling We examine the estimation process and associated hyperparameters in Equation (S.28), 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.6, 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.6: **Effect of J**. The RMSE of generated state trajectories for FlowDAS is evaluated with different Monte Carlo sampling times J in Equation (S.28) for the Lorenz 1963 experiment. As J increases, the RMSE initially decreases, indicating improved performance, and then stabilizes.

C.2 Posterior Estimation Methods

We also evaluate the impact of different methods for posterior estimation: as defined in Equations (9) and (10). The results are presented in Table S.7, where "1st-order" and "2nd-order" refer to 1st-order Milstein method and 2nd-order stochastic Runge-Kutta method, 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 consistently delivers better performance. This suggests that employing more accurate estimations of $p(\boldsymbol{X}_1|\boldsymbol{X}_s, \boldsymbol{X}_0)$ 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.

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

Table S.7: **Effect of posterior estimation**. The RMSE of vorticity estimate from FlowDAS is evaluated on the incompressible Navier-Stokes task using different posterior estimation methods as defined in Equations (9) and (10), and forecasting without observations (i.e., no correction). For the super-resolution tasks $16^2 \to 128^2$ and $32^2 \to 128^2$, both posterior estimation methods significantly outperform forecasting without observations. Among them, the 2nd-order method achieves the lowest RMSE.

C.3 Particle Collapse and Robustness of Likelihood-based Sampling

The likelihood-based sampling in Equation (8) can, in principle, suffer from particle collapse, which is a well-known issue in high-dimensional importance sampling where a few weights dominate the normalization sum, leading to an effective sample size close to one. We provide further analysis and our mitigation strategy below.

Observed behavior. During the stochastic interpolation process, we observed that the likelihood score $\|\boldsymbol{y} - \mathcal{A}\boldsymbol{x}_s\|_2^2$ typically decreases as the sampling proceeds, but in some cases starts to increase sharply near the end of the trajectory (typically within the final 30 to 50 steps). This abnormal increase coincides with the particle weights w_j becoming heavily concentrated on a single particle $\hat{\boldsymbol{X}}_1^{(j)}$, indicating a collapse of the Monte Carlo estimator.

Empirically, we observed two distinct regimes:

- When the likelihood term decreases as expected, the weights remain balanced across particles, ensuring stable estimation.
- When the likelihood begins to rise unexpectedly, one or two particles dominate the weights (approaching 100%), leading to collapse and potential divergence.

Mitigation strategies. To prevent collapse, we employ a two-step remedy:

- **Resampling:** When abnormally concentrated weights are detected, we immediately perform resampling to regenerate a balanced particle set. This is a standard correction technique in sequential importance sampling.
- **Alternative sampling strategy:** If resampling does not stabilize the process, we switch to the alternative sampling strategy detailed in Appendix C.1. Although this approach may yield slightly less precise estimates, it significantly enhances robustness and prevents divergence in high-dimensional settings.

C.4 Alternative Likelihood Score Approximation

In the main formulation, the likelihood score term $\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_0, \boldsymbol{X}_s)$ is estimated via Monte Carlo sampling over particles $\{\boldsymbol{X}_1^{(j)}\}_{j=1}^K$, which provides an unbiased approximation at the cost of potential particle collapse in high-dimensional settings. An alternative approach is to adopt a *DPS-type approximation* [42]:

$$\nabla \log p(\boldsymbol{y} \mid \boldsymbol{X}_0, \boldsymbol{X}_s) \approx \nabla \log p(\boldsymbol{y} \mid \hat{\boldsymbol{X}}_1), \quad \text{where } \hat{\boldsymbol{X}}_1 := \mathbb{E}[\boldsymbol{X}_1 \mid \boldsymbol{X}_0, \boldsymbol{X}_s].$$

This estimator replaces the Monte Carlo sampling with the conditional expectation $\hat{\boldsymbol{X}}_1$, thereby avoiding weight collapse but introducing a bias bounded by Jensen's inequality.

To assess the effect of this approximation, we directly substituted the DPS-type term into Equation (8) of the main paper and re-ran the weather forecasting experiment. The quantitative comparison is summarized in Table S.8.

The DPS estimate can mitigate collapse and sometimes improve numerical stability. However, the denominator in Equation (S.34) (see derivation below) can approach zero when $s \to 0$, causing the

Method	RMSE ± Std
FlowDAS	0.056 ± 0.002
FlowDAS-DPS	0.050 ± 0.002

Table S.8: **Comparison of likelihood approximations on the weather forecasting task.** While the DPS approximation slightly improves RMSE, it introduces theoretical bias due to Jensen's gap and requires additional stabilization.

optimization to become unstable. To prevent this, we select a cutoff value s_c and apply the DPS-type update only for $s > s_c$. In the weather forecasting experiment, we set $s_c = 0.04$ via grid search. This parameter may vary across datasets and is problem dependent, with no closed-form theoretical criterion for selection. Further tuning of parameters $(\alpha_s, \beta_s, \sigma_s)$ could improve the method, which we leave for future work.

Overall, FlowDAS-DPS provides a biased yet more stable alternative that complements our main FlowDAS formulation and demonstrates the flexibility of the framework.

Short derivation of FlowDAS-DPS. To derive the DPS-type approximation, we notice that

$$\boldsymbol{b}_s(\boldsymbol{X}_s, \boldsymbol{X}_0) = \dot{\alpha}_s \boldsymbol{X}_0 + \dot{\beta}_s \mathbb{E}[\boldsymbol{X}_1 \mid \boldsymbol{X}_0, \boldsymbol{X}_s] + \sqrt{s} \dot{\sigma}_s \mathbb{E}[\boldsymbol{z} \mid \boldsymbol{X}_0, \boldsymbol{X}_s], \tag{S.32}$$

Besides,

$$\boldsymbol{X}_s = \alpha_s \boldsymbol{X}_0 + \beta_s \mathbb{E}[\boldsymbol{X}_1 \mid \boldsymbol{X}_0, \boldsymbol{X}_s] + \sqrt{s} \sigma_s \mathbb{E}[\boldsymbol{z} \mid \boldsymbol{X}_0, \boldsymbol{X}_s]. \tag{S.33}$$

By combining the above two equations, we obtain

$$\mathbb{E}[\boldsymbol{X}_1 \mid \boldsymbol{X}_0, \boldsymbol{X}_s] = \frac{\sigma_s b_s(\boldsymbol{X}_s, \boldsymbol{X}_0) - \dot{\sigma}_s \boldsymbol{X}_s - (\sigma_s \dot{\alpha}_s - \alpha_s \dot{\sigma}_s) \boldsymbol{X}_0}{\sigma_s \dot{\beta}_s - \beta_s \dot{\sigma}_s}, \tag{S.34}$$

C.5 Ablation on Conditioning Horizon

To study the effect of the conditioning horizon, we perform an ablation by varying the number of observed past states used during inference. We evaluate performance using the Continuous Ranked Probability Score (CRPS), a proper scoring rule that jointly measures accuracy and uncertainty calibration of probabilistic forecasts—the lower the CRPS, the better the performance.

Navier–Stokes: CRPS vs. Conditioning Horizon				
# Conditioning States	10	6	3	1
CRPS	**0.538**	0.634	0.663	0.776

SEVIR: CRPS vs. Conditioning Horizon		
# Conditioning States	6	3	1
CRPS	**0.015**	0.021	0.021

Table S.9: **Effect of conditioning horizon.** Longer conditioning horizons consistently improve CRPS, indicating better probabilistic forecasting performance.

The results clearly show that incorporating a longer conditioning horizon leads to consistent improvements in forecast accuracy and uncertainty estimation. In the Navier–Stokes task, extending the horizon from 1 to 10 steps yields approximately a 30% reduction in CRPS, highlighting the benefit of temporal context in capturing complex dynamics. Similarly, in the SEVIR precipitation forecasting task, conditioning on six previous frames achieves the best predictive accuracy. These findings confirm that longer temporal conditioning enables FlowDAS to leverage sequential dependencies more effectively, which is crucial for accurate data assimilation in high-dimensional and nonlinear dynamical systems.

MODEL	PARAMS (M)	TRAINING TIME (HR)	INFERENCE TIME (S/SAMPLE)
FLOWDAS	30.0	22	30.9
PDEDIFF	22.9	45	28.9
TRANSOLVER	11.2	80	12.7
FNO	1880.0	80	2.4

Table S.10: Comparison of model size, total training time, and inference time on the weather forecasting task using a single A100 GPU.

METHOD	COMPUTATION TIME (S)	RMSE $\pm$ STD	MAX MEMORY (GB)
1ST ORDER	17.100	0.0490 ± 0.0026	1.8
2ND ORDER	30.871	0.0485 ± 0.0035	3.9
HYBRID	205.271	0.0478 ± 0.0033	70.0

Table S.11: **Accuracy–speed trade-off for stochastic integrators.** The hybrid "2nd order $\times$ 400 + Flow $\times$ 100" improves RMSE only slightly (absolute gain 0.0007) but increases computation time by $\sim 7\times$ and peak memory by $\sim 18\times$ compared with the second–order integrator.

C.6 Runtime and Parameter Comparison

We report the model sizes, total training times, and inference costs for the SEVIR weather forecasting task using a single NVIDIA A100 GPU in Table S.10. FlowDAS achieves the shortest training time among all baselines, converging roughly twice as fast as PDEDiff and nearly four times faster than Transolver, despite having a comparable model size.

The iterative correction mechanism in FlowDAS introduces a modest increase in inference time compared to purely feedforward models. However, this process is essential for producing observation-consistent predictions and stabilizing long-horizon forecasts. Importantly, FlowDAS exhibits strong training efficiency, reaching convergence significantly faster than other baselines.

C.7 Bias–efficiency trade-offs of stochastic integrators

Our stochastic integrators in Equations (9) and (10) approximate the observation–conditioned $\hat{X}_1$ and therefore introduce discretization bias. We quantify the accuracy–speed trade-off on the weather forecasting task by comparing three settings: a first–order integrator, a second–order integrator, and a hybrid method that follows the second–order scheme for the first 400 steps and then computes the full stochastic–interpolant path to the endpoint X_1 for the last 100 steps ("2nd order $\times$ 400 + Flow $\times$ 100"). The hybrid reduces bias but is much more costly.

We also considered a full-path variant ("Flow$\times$500") that computes X_1 exactly at *every* step by rolling out the leftward stochastic interpolant. This requires $500 + (500 + 499 + \cdots + 1) \approx 2.5 \times 10^5$ forward passes and retaining the entire computation graph to backpropagate $\log p(y \mid X_1)$, which leads to out-of-memory failures even on A100 GPUs. In contrast, the first– and second–order schemes provide stable inference with much lower cost while maintaining competitive accuracy.

These results shown in Table S.11 support our choice: conditioning on observations at each interpolation step delivers most of the DA benefit, whereas reducing the small integrator bias further yields only marginal accuracy gains at a very high computational and memory cost. We include this analysis and the table above in the appendix for clarity.

D Additional Experiment Results

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

D.1 Double-well Potential Problem

D.1.1 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:

$$\mathrm{d}x = -4x(x^2 - 1)\,\mathrm{d}t + \beta_d\,\mathrm{d}\xi_t \tag{S.35}$$

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

$$y = \mathcal{A}(x) + \eta = x^3 + \eta \tag{S.36}$$

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.6a.

D.1.2 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 $(x \rightarrow -x)$.
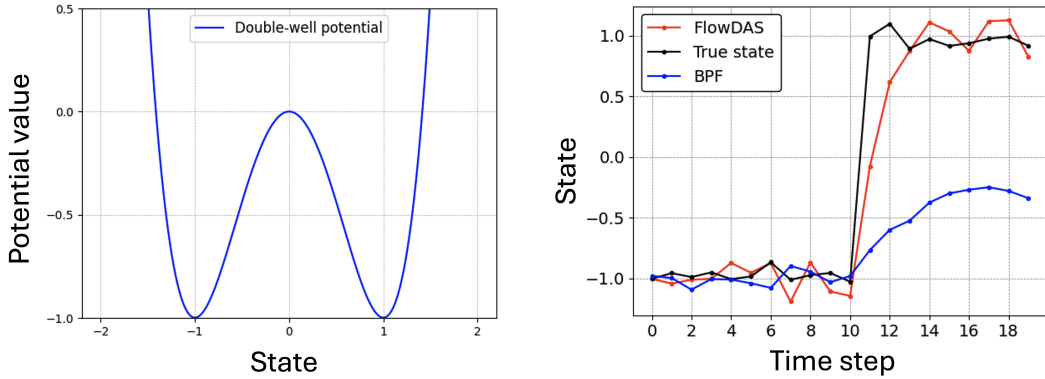
(a) **Illustration of the Double-Well Potential Problem**. In the double-well potential system, particles are typically captured at the bottoms of the wells ($x = 1$ and $x = -1$). With low probability, particles can transition between wells due to the stochastic term in the transition equation.

(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.6: Illustrations and visualization results for the double-well potential problem.

D.1.3 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.

D.1.4 Hyperparameters During Inference

Hyperparameters of the inference procedure, specified for the double-well potential task, are presented in Table S.4.

D.1.5 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 [50]. 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.

D.1.6 Results

The visual results are shown in Figure S.6b. 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 [50].

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.

D.2 Lorenz 1963

D.2.1 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:

$$
\begin{aligned}
k_{1a_n} &= h \cdot \mu(b_n - a_n), \\
k_{1b_n} &= h \cdot \left(a_n(\rho - c_n) - b_n \right), \\
k_{1c_n} &= h \cdot \left(a_n b_n - \tau c_n \right),
\end{aligned}
\tag{S.37}
$$

$$
\begin{aligned}
k_{2a_n} &= h \cdot \mu \left((b_n + \frac{k_{1b_n}}{2}) - (a_n + \frac{k_{1a_n}}{2}) \right), \\
k_{2b_n} &= h \cdot \left((a_n + \frac{k_{1a_n}}{2})(\rho - (c_n + \frac{k_{1c_n}}{2})) - (b_n + \frac{k_{1b_n}}{2}) \right), \\
k_{2c_n} &= h \cdot \left((a_n + \frac{k_{1a_n}}{2})(b_n + \frac{k_{1b_n}}{2}) - \tau(c_n + \frac{k_{1c_n}}{2}) \right),
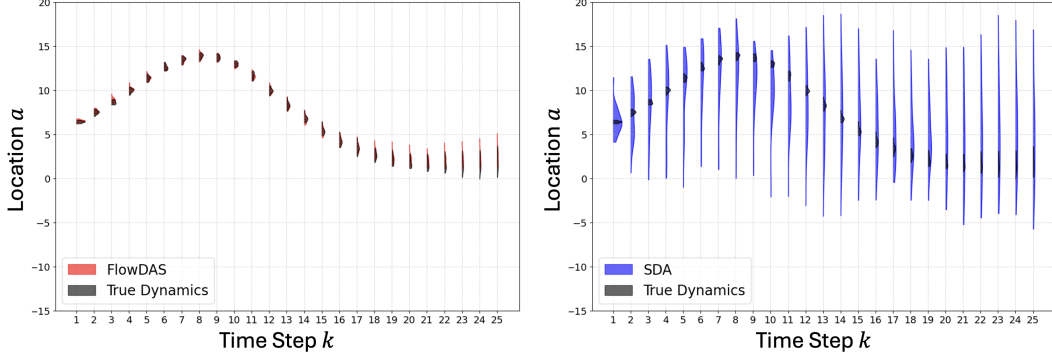\end{aligned}
\tag{S.38}
$$

24

Figure S.7: **Forecaseting 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.

$$k_{3a_n} = h \cdot \mu \left((b_n + \frac{k_{2b_n}}{2}) - (a_n + \frac{k_{2a_n}}{2}) \right),$$

$$k_{3b_n} = h \cdot \left((a_n + \frac{k_{2a_n}}{2})(\rho - (c_n + \frac{k_{2c_n}}{2})) - (b_n + \frac{k_{2b_n}}{2}) \right), \tag{S.39}$$

$$k_{3c_n} = h \cdot \left((a_n + \frac{k_{2a_n}}{2})(b_n + \frac{k_{2b_n}}{2}) - \tau(c_n + \frac{k_{2c_n}}{2}) \right),$$

$$k_{4a_n} = h \cdot \mu \left((b_n + k_{3b_n}) - (a_n + k_{3a_n}) \right),$$
$$k_{4b_n} = h \cdot \left((a_n + k_{3a_n})(\rho - (c_n + k_{3c_n})) - (b_n + k_{3b_n}) \right), \tag{S.40}$$
$$k_{4c_n} = h \cdot \left((a_n + k_{3a_n})(b_n + k_{3b_n}) - \tau(c_n + k_{3c_n}) \right).$$

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

$$a_{n+1} = a_n + \frac{1}{6}(k_{1a_n} + 2k_{2a_n} + 2k_{3a_n} + k_{4a_n}),$$

$$b_{n+1} = b_n + \frac{1}{6}(k_{1b_n} + 2k_{2b_n} + 2k_{3b_n} + k_{4b_n}), \tag{S.41}$$

$$c_{n+1} = c_n + \frac{1}{6}(k_{1c_n} + 2k_{2c_n} + 2k_{3c_n} + k_{4c_n}).$$

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 (11).

D.2.2   Training and Testing Data Generation

We apply the RK4 method described in Appendix D.2.1 to generate the simulated training and testing data.

D.2.3   Hyperparameters During Inference

Hyperparameters of the inference procedure, specified for the Lorenz 1963 task, are presented in Table S.4. 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.

D.2.4   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 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.

D.2.5 Baseline and Additional Results

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.

We also implemented EnSF [58] and its performance is on par with BPF. This is expected since EnSF uses a training-free Monte Carlo estimator for the score function, which faces the same particle collapse issues, i.e., samples concentrate in high-probability regions of the transition dynamics, leading to biased score estimates. This reflects a train-sampling tradeoff: EnSF saves training time but cannot learn accurate scores, especially under highly nonlinear dynamics and observation models like in our noisy Lorenz-63 settings(e.g., our $\mathrm{atan}(x_1)$ observation). FlowDAS, by learning the drift with collected data, avoids this issue and achieves superior performance. For the EnSF, we largely follow the official implementation and adopt the settings summarized below. The ensemble size is set to 20000 to ensure stable Monte Carlo estimation of the ensemble mean and covariance. The diffusion SDE is integrated using 500 steps with the Euler solver. Both ε_a and ε_b are fixed to 0.001. The initial data variance is set to zero, and no covariance inflation is applied (inflation factor $= 1$).

METHOD	FLOWDAS	SDA	BPF	ENSF
RMSE	**0.202**	1.114	0.270	0.298

Table S.12: **Comparison of FlowDAS with diffusion-based and ensemble-based filters.** Reported RMSEs are computed on the same problem setting as described in the Section 4.1. FlowDAS achieves the lowest RMSE, demonstrating robustness under non-Gaussian dynamics and non-linear observation observations.

D.2.6 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 (11) using the RK4 method, starting from the same initial point. The visualization of location a across 25 time steps is shown in Figure S.7.

D.3 Incompressible Navier-Stokes Flow

D.3.1 Incompressible Navier-Stokes Flow Problem Settings

For the incompressible Navier-Stokes flow problem, we adopted the problem setting from [31] 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:

$$
\begin{aligned}
\xi(t, x, y) = {} & W_1(t)\sin(6x) + W_2(t)\cos(7x) + W_3(t)\sin(5(x+y)) + W_4(t)\cos(8(x+y)) \\
& + W_5(t)\cos(6x) + W_6(t)\sin(7x) + W_7(t)\cos(5(x+y)) + W_8(t)\sin(8(x+y))
\end{aligned}
\tag{S.42}
$$

For more details and insights about this problem, see [31].

D.3.2 Hyperparameters During Inference

Table S.4 presents the hyperparameters of the inference procedure for the incompressible Navier-Stokes flow task.

D.3.3 Neural Network for Learning the Drift Velocity

We use a U-Net architecture to approximate b_s, following the network proposed in [31]. The conditioning on X_0 is implemented through channel concatenation in the input. The architectural details are as follows:

- Number of states conditioned on: 10.
- Number of initial channels: 128.
- 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 [59] 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 1000 epochs.

D.3.4 Baseline

We compare FlowDAS to FNO-DA, Transolver-DA, and PDEDiff.

FNO-DA   The model is configured with 7 stacked Fourier layers. Each layer processes data in a 128-dimensional feature space and truncates the Fourier series to include 6 modes in each spatial direction. A padding of 6 is applied before the Fourier operations and removed afterward. The model is provided with the previous 10 states to predict the state at the next time step. We train the model for 1000 epochs, using a batch size of 64 and the AdamW optimizer with an initial learning rate of 0.0001.

Transolver-DA   The model has 8 layers of transolver blocks whose hidden channels are 128 and have 32 slices, with 8 attention heads. The model is provided with the previous 10 states to predict the state at the next time step. We train the model for 1000 epochs using a batch size of 64, the AdamW optimizer, and a OneCycleLR learning rate scheduler with an initial learning rate of $5e^{-4}$.

PDEDiff   The score neural network is configured with a temporal window of 11, of which the first 10 inputs are conditions and the forecast horizon is 1; an embedding layer dimensionality of 32; and a base number of feature channels 256. The network depth is 5, and the activation function used is SiLU [59]. We train the model using a batch size of 64 and the AdamW optimizer, with a linear learning rate scheduler (initial learning rate 0.001) and a weight decay of 0.001.

During inference, the α_1 and α_2 used is varied from task to task for both the FNO-DA and the Transolver-DA. We show the detailed parameter settings in Tables S.13 and S.14. We use an AdamW optimizer for Kalman update with a learning rate of $1e^{-4}$ and a maximum iteration time of 2000 to ensure the loss converges. Without other indications, we adopt the same optimizer settings for both FNO-DA and Transolver-DA.

D.3.5 Kinetic Energy Spectrum Analysis

We evaluate 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:

$$E(k_n) = \sum_{\substack{p,q \\ k_{p,q} \in \text{bin } n}} E(k_{x_p}, k_{y_q}), \tag{S.43}$$

			α_1	α_2
	SR	4x	0.63	1
INCOMPRESSIBLE	SR	8x	1.59	1
NAVIER STOKES	SO	5%	0.10	1
	SO	1.5625%	0.29	1
WEATHER FORECASTING			1.22	1

Table S.13: **Hyperparameters for FNO-DA in the inference stage** of all experiments presented in this study. SR stands for super-resolution task and SO represents sparse observation (inpainting) task

			α_1	α_2
	SR	4x	0.53	1
INCOMPRESSIBLE	SR	8x	1.21	1
NAVIER STOKES	SO	5%	0.14	1
	SO	1.5625%	0.22	1
WEATHER FORECASTING			5.00	1

Table S.14: **Hyperparameters for Transolver-DA in the inference stage** of all experiments presented in this study. SR stands for super-resolution task and SO represents sparse observation (inpainting) task

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 PDEDiff 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.

D.3.6 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.8 (a), and additional visualization results are provided in Figure S.9. FlowDAS effectively reconstructs high-resolution vorticity data from low-resolution observations, while PDEDiff struggles to capture high-frequency physics.

$32^2 \rightarrow 128^2$ **Super-Resolution** The kinetic energy spectrum is shown in Figure S.8 (b), and additional visualization results are provided in Figure S.10. Similar to the $16^2 \rightarrow 128^2$ task, FlowDAS achieves superior alignment with the true state compared to PDEDiff.

5% **Sparse Observation** The kinetic energy spectrum is presented in Figure S.8 (d), with additional visualization results in Figure S.11. FlowDAS accurately recovers the vorticity field despite the highly sparse observations, significantly outperforming PDEDiff, demonstrating strong recovery of physical information.

Across all tasks, FlowDAS consistently outperforms PDEDiff, 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.
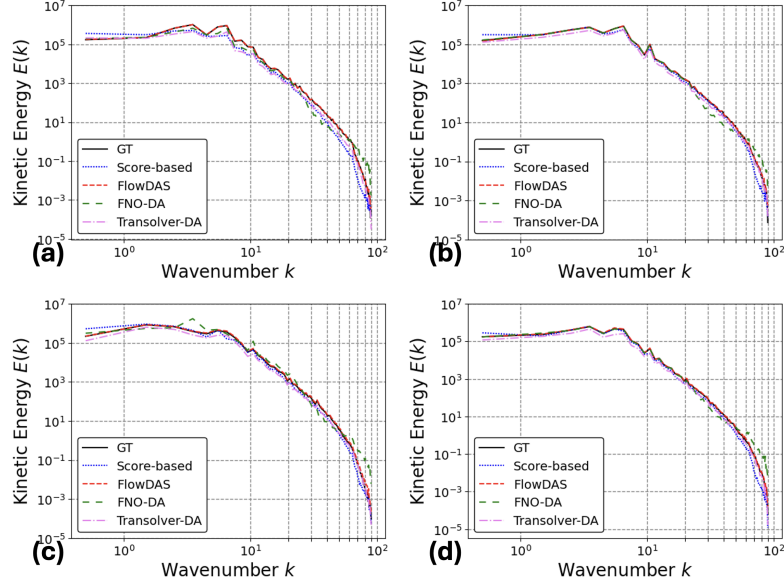
Figure S.8: **The kinetic energy spectrum of:** (a) super-resolution task: $16^2 \to 128^2$; (b) super-resolution task: $32^2 \to 128^2$; (c) sparse observation task: 1.5625%; (d) sparse observation task: 5%, in the incompressible Navier-Stokes flow task. We present the kinetic energy spectrum of the true state alongside the estimations from FlowDAS and baseline methods. FlowDAS can produce results that better aligned with the true state in terms of the kinetic energy spectrum, evidenced by the oscillations in the spectrum of baselines, indicating FlowDAS's superiority in recovering the physics information and effectiveness as a surrogate model for stochastic dynamic systems.
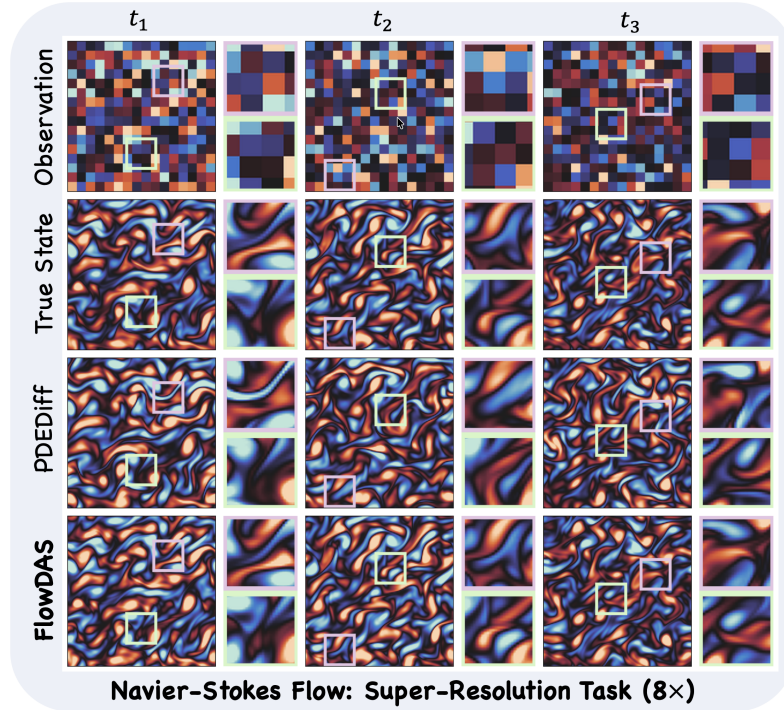
Figure S.9: **Data assimilation of incompressible Navier-Stokes flow—Super-Resolution task ($8\times$).** Experiments were conducted at observation resolution of 16^2. The goal of super-resolution task is to reconstruct high-resolution vorticity data (128^2) from low-resolution observations.
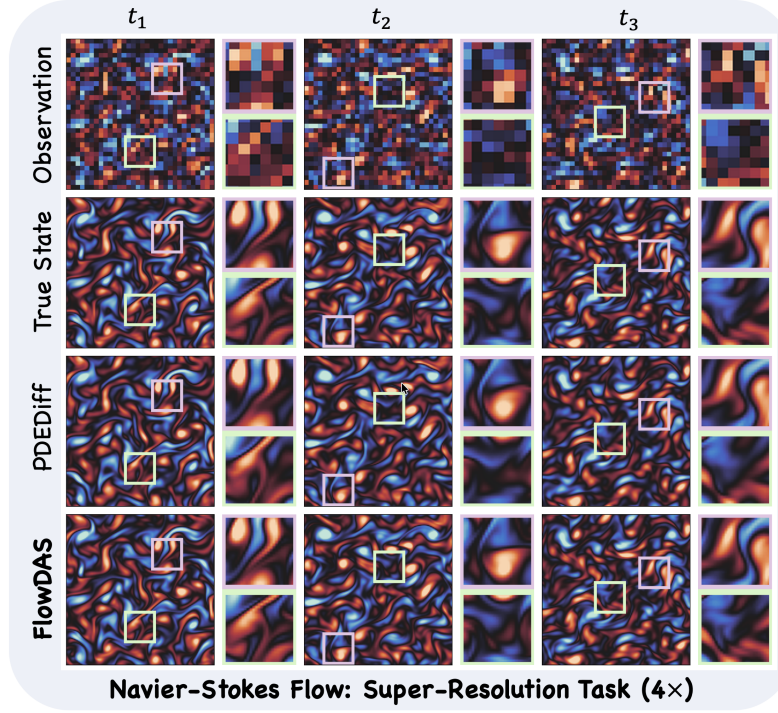
Figure S.10: **Data assimilation of incompressible Navier-Stokes flow—Super-Resolution task** ($4\times$). Experiments were conducted at observation resolution of 32^2. The goal of super-resolution task is to reconstruct high-resolution vorticity data (128^2) from low-resolution observations.
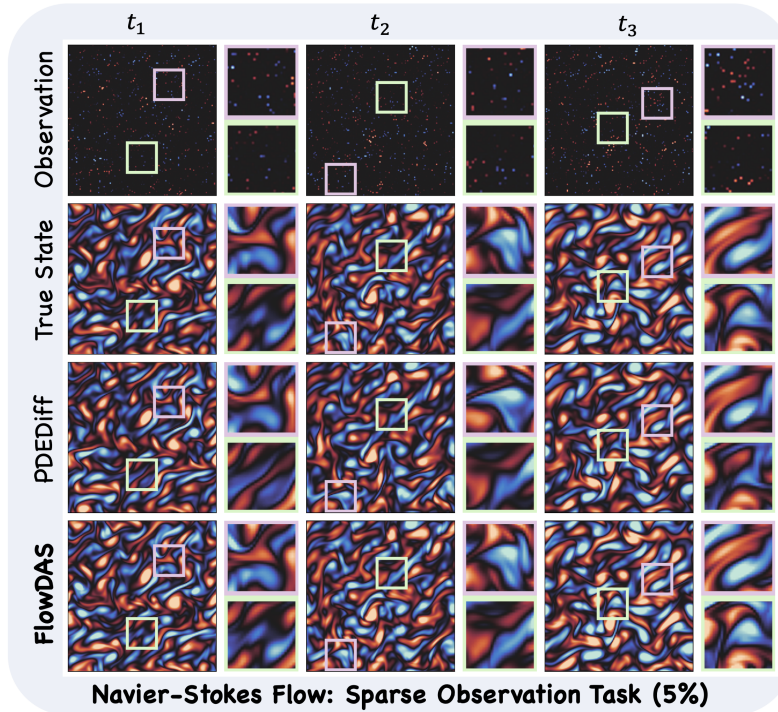
Figure S.11: **Data assimilation of incompressible Navier-Stokes flow—Sparse-Observation task** (5%). Experiments were conducted at observation sparsity level at 5%. The goal of sparse observation task is to recover the complete vorticity field.

D.3.7 Performance Degradation Under More Challenging Cases

In the main paper, FlowDAS was evaluated on the incompressible Navier–Stokes system under moderate super-resolution (SR) and sparse-observation (SO) conditions, specifically SR $4\times$, SR $8\times$, SO 5%, and SO 1.5625%. To evaluate the performance under more extreme conditions, we further test FlowDAS on SR $16\times$ and SO settings with observation coverages as low as 0.75%.

These results demonstrate a *graceful degradation* of performance as the observation coverage decreases or the upscaling factor increases, confirming the robustness of FlowDAS even under highly underdetermined measurement scenarios.

	SR $16\times$	SO 1%	SO 0.75%
RMSE $\pm$ STD	0.398 ± 0.015	0.395 ± 0.016	0.452 ± 0.019

Table S.15: **RMSE ($\pm$ std) of FlowDAS on Navier–Stokes under more challenging super-resolution and sparse-observation conditions.** Performance decreases smoothly as the observation ratio is reduced, supporting the stability and generalization of FlowDAS in limited-measurement regimes.

D.3.8 Trajectory-level Evaluation

To further evaluate the temporal stability of FlowDAS, we examine its performance over full generated trajectories on the incompressible Navier–Stokes system. While the main results in Table 2 focus on single-step assimilation accuracy, this analysis quantifies how reconstruction errors and their variability evolve over multiple forecasting timesteps. At each timestep, RMSE and standard deviation are computed over 50 independently sampled initial states, each propagated through 30 stochastic realizations.
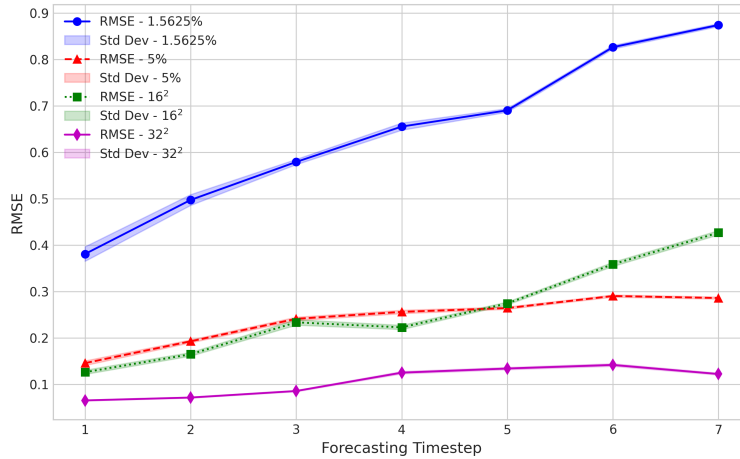

Figure S.12: **Trajectory-level RMSE and standard deviation of FlowDAS on the incompressible Navier–Stokes system.** The plot shows the evolution of RMSE (solid lines) and corresponding standard deviations (shaded areas) across forecasting timesteps. FlowDAS maintains stable performance over several assimilation steps, with gradual error accumulation consistent with stochastic propagation.

D.4 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 [60–62], biological flow studies [63–65], and medical research [66–68].
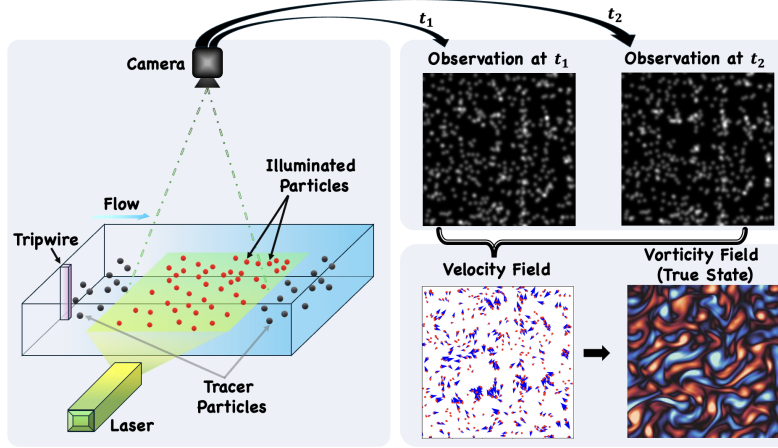
Figure S.13: **Illustration of the real-world Particle Image Velocimetry (PIV) experiment**: The flow is seeded with tracer particles illuminated by a laser sheet, and their movements are captured by a camera to derive the sparse velocity field. The goal here is to recover dense vorticity field from the sparse velocity measurement.

D.4.1 Task Description

In a standard PIV setup, as shown in Figure S.13, 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 ($x = \omega$) from sparse velocity measurements. This observation model is defined by

$$y = \mathcal{A}(v(\omega)) + \eta, \tag{S.44}$$

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

D.4.2 Dataset

In this experiment, we use 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 [69]. The particle positions in our simulation are randomly initialized and then perturbed according to the simulated flow motion pattern. In these synthetic images, we assume 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 are processed through a standard PIV pipeline to extract particle locations, match corresponding particles across frames, and compute sparse velocity observations. These sparse measurements are then used in DA to reconstruct the full vorticity fields.

D.4.3 Baselines

We compare our method against the SDA solver. Instead of training new scores or stochastic interpolant networks, we directly adopt the networks trained on vorticity data from the incompressible NS flow experiment to evaluate FlowDAS and SDA on the PIV task.

32

PARAMETER	VALUE	UNIT
PARTICLE DENSITY	0.03	PARTICLE PER PIXEL
PARTICLE DIAMETER	3	PIXEL
PEAK INTENSITY	255	GRAY VALUE

Table S.16: **PIV task: parameters for simulation.** This table summarizes parameters for the PIV experiments in detail.
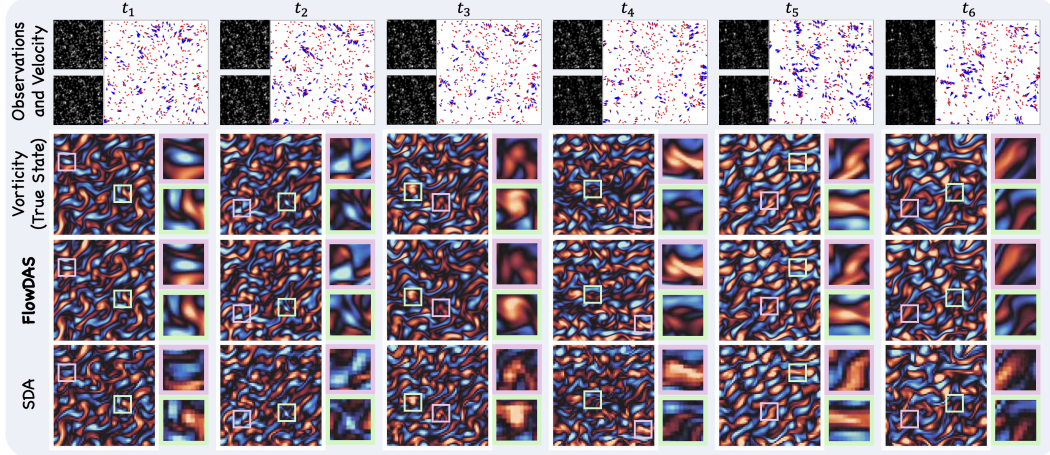

Figure S.14: **Data assimilation of Particle Image Velocimetry.** The vorticity field is visualized in the same way as in Section 4.2. FlowDAS outperforms SDA in terms of reconstruction fidelity and RMSE, recovering more detailed features even when direct observations are not available. These improvements highlight the potential of FlowDAS for real-world applications.

D.4.4 Experiment Setting

The training data, neural network (including FlowDAS and SDA) is the same as those in the incompressible Navier-stokes flow simulation. Figure S.13 shows the standard PIV set up. Hyperparameters of the inference procedure, specified for the PIV task, are presented in Table S.4.

D.4.5 Results

FlowDAS accurately reconstructs the underlying fluid dynamics from observed particle images, producing vorticity estimates with high precision. As shown in Figure S.14, it outperforms the baseline SDA in terms of reconstruction fidelity. Quantitatively, FlowDAS achieves a lower average RMSE of 0.118, compared to 0.154 for SDA. This performance gap highlights the robustness of FlowDAS and its potential utility in fluid dynamics applications.

D.5 Weather Forecasting

D.5.1 Baselines

We compared FlowDAS to FNO-DA, Transolver-DA, and PDEDiff.

FNO-DA   The model is configured with 7 stacked Fourier layers. Each layer processes data in a 128-dimensional feature space and truncates the Fourier series to include 64 modes in each spatial direction. A padding of 6 is applied before the Fourier operations and removed afterward. The model is provided with the previous 6 states to predict the state at the next time step. We train the model for 1000 epochs, using a batch size of 200 and the AdamW optimizer with an initial learning rate of 0.0001.

BACKBONE OF FLOWDAS	RMSE ↓	CSI(τ_{20}) (0.3) ↑	CSI(τ_{40}) (0.5) ↑
FNO	**0.053±0.001**	**0.718±0.015**	**0.568±0.024**
TRANSOLVER	0.056±0.002	0.702±0.018	0.540±0.028
UNET	0.056±0.002	0.703±0.017	0.540±0.023

Table S.17: Comparison of backbone network architectures of the drift model on the weather forecasting task. All metrics are averaged over multiple runs. Subtle improvements are observed when using a FNO network as the backbone of the drift model.

Transolver-DA   The model has 8 layers of transolver blocks whose hidden channels are 128 and have 32 slices, with 8 attention heads. The model is provided with the previous 6 states to predict the state at the next time step. We train the model for 1000 epochs using a batch size of 200, the AdamW optimizer, and a OneCycleLR learning rate scheduler with an initial learning rate of $5e^{-4}$.

PDEDiff   The score neural network is configured with a temporal window of 7, of which the first 6 inputs are conditions, and the forecast horizon is 1; an embedding layer dimensionality of 32; and a base number of feature channels 256. The network depth is 5, and the activation function used is SiLU [59]. We train the model using a batch size of 200 and the AdamW optimizer, with a linear learning rate scheduler (initial learning rate 0.001) and a weight decay of 0.001.

During inference, the α_1 and α_2 used is varied from task to task for both the FNO-DA and the Transolver-DA. We show the detailed parameter settings in Tables S.13 and S.14. We use an AdamW optimizer for Kalman update with a learning rate of $1e^{-4}$ and a maximum iteration time of 2000 to ensure the loss finally becomes stable. Without other indications, we adopt the same optimizer settings for both FNO-DA and Transolver-DA.

D.5.2   Training Details

We experiment with three backbone architectures—U-Net, FNO, and Transolver—to learn the drift term and compare their performance and provide numerical results in Table S.17. The architectural details are as follows:

- UNet: as shown in Appendix D.3.3, the only change is the number of states conditioned on. Here we condition on previous 6 states.

- FNO: this model is configured with 7 stacked Fourier layers. Each layer processes data in a 128-dimensional feature space and truncates the Fourier series to include 64 modes in each spatial direction. A padding of 6 is applied before the Fourier operations and removed afterward. The model is provided with the previous 6 states, the $\boldsymbol{X}_s$ and the time s as additional channel concatenated to the previous states and the $\boldsymbol{X}_s$ to predict the drift term.

- Transolver: this model has 5 layers of transolver blocks whose hidden channels are 128 and have 32 slices, with 8 attention heads. The sinusoidal timestep embeddings are applied. The inputs are the previous 6 states, the $\boldsymbol{X}_s$ and the time s and the output is the drift term.

D.5.3   Results

We present additional visualizations to demonstrate FlowDAS's ability of capturing unknown system dynamics on this weather forecasting task in Figures S.15 to S.17. FlowDAS successfully generates accurate estimates of the weather in the future frames in an autoregressive manner. Table 3 provides numerical results of FlowDAS as well as other baseline methods. FlowDAS consistently outperformed DA baselines both in terms of RMSE and CSI metrics.
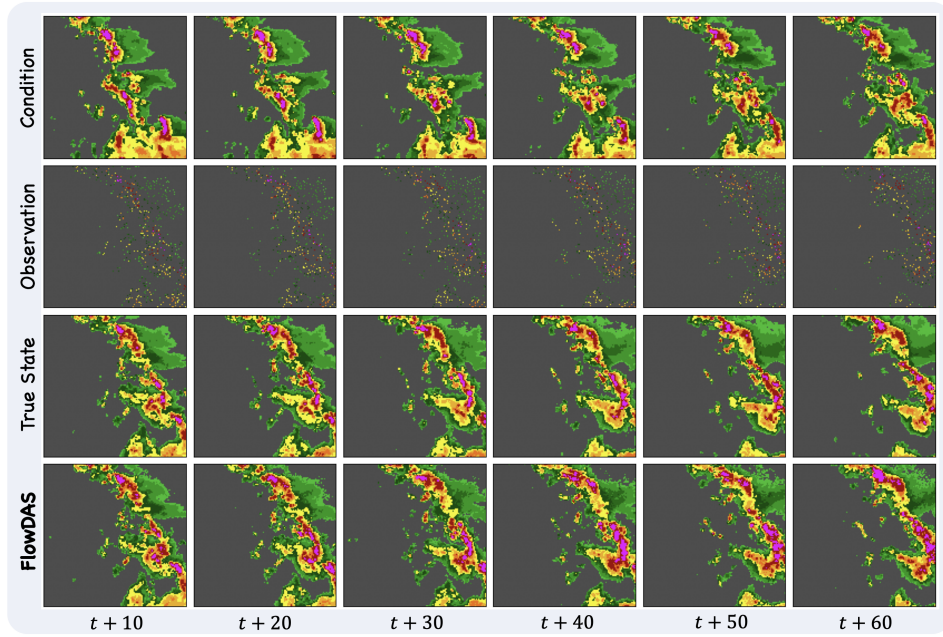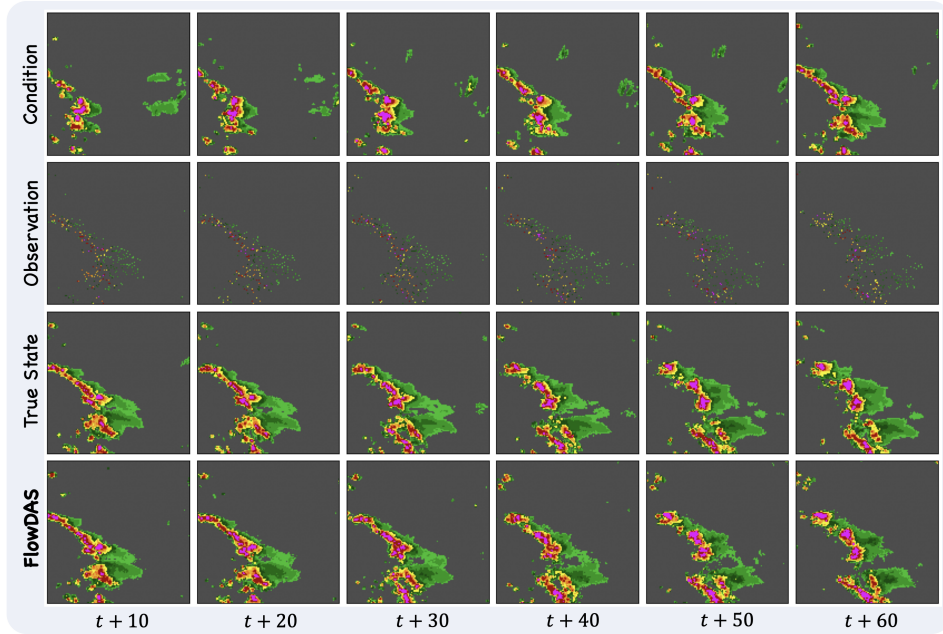
Figure S.15: **Additional result 1: Data assimilation of weather forecasting (Sparse-Observation task)**. The underlying PDE of the dynamical system is unknown. This experiment was conducted at an observation sparsity level of 10%, showing the capacity of long-trajectory forecasting.
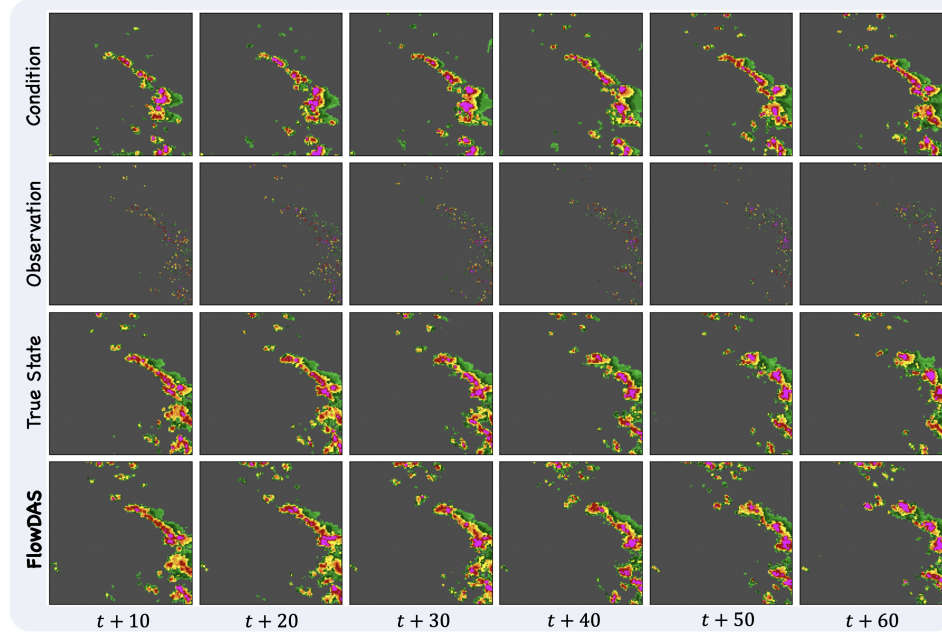

Figure S.16: **Additional result 2: Data assimilation of weather forecasting (Sparse-Observation task)**. The underlying PDE of the dynamical system is unknown. This experiment was conducted at an observation sparsity level of 10%, showing the capacity of long-trajectory forecasting.

Figure S.17: **Additional result 3: Data assimilation of weather forecasting (Sparse-Observation task)**. The underlying PDE of the dynamical system is unknown. This experiment was conducted at an observation sparsity level of 10%, showing the capacity of long-trajectory forecasting.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state the claims we made in this paper in the abstract and introduction, as well as the contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in appendix and state the plan of future work in the final section in the main paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We include proofs to all mathematical claims we made in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We will provide a link to the codebase and data once published for reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide a link to the codebase and data once published for reproducibility.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have specified all necessary experimental details both in the main text and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We have included the standard deviation in the experiment section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide detailed computing resources used for each experiment in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All research conducted in the paper conforms in every respect with the NeurIPS code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We state the proposed method may benefit the data assimilation community.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly cited all assets used in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We will provide a link to the new assets introduced in the paper once published for reproducibility.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:
    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.