# ENJOI: ENSEMBLE JOINT SCORE FILTER FOR GENERATIVE DATA ASSIMILATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Data Assimilation (DA) aims to recover the full state of a dynamical system that is only partially observed. A solution is to use Score-based models to generate physically consistent trajectories that agree with the observations. These Autoregressive Diffusion models are trained by conditioning on the previous state; however, they do not take into account the uncertainty of their past predictions. We propose a new diffusion-based assimilation algorithm that dynamically balances the confidence in the current state and the new observations. Crucially, we choose to learn the distribution of the *joint* state containing both the past and future. This allows us to use a modified version of En4DVar, a classical DA algorithm that relies on the covariance of an ensemble of particles. Experiments on fluid and traffic flow simulations show improved reconstruction performance, especially in situations where observations are sparse and non-homogeneous.

## 1 INTRODUCTION

Real world dynamical systems are often partially observed, making it challenging to estimate the full state at a given time. Scientists use numerical simulators to faithfully reproduce the dynamics, but they require an accurate starting point. Even with faithful initial conditions, the approximations in the physical model eventually lead to drift. A solution is to correct the simulation errors iteratively by ingesting sparse measurements; this practice is known as Data Assimilation (DA). The framework has applications across diverse domains, from weather prediction (Carrassi et al., 2018) to traffic engineering (Work et al., 2010).

DA problems are traditionally solved using Bayesian Filtering methods, applied to high-dimensional and non-linear systems. A classical algorithm is the Particle Filter (PF), which is asymptotically exact, but is notoriously subject to distribution collapse in large dimensions. Gaussian-linear methods (e.g. Kalman, 4DVar) are more robust but suffer from the linearization and mean-field assumption, resulting in overly smooth reconstructions. Modern systems often rely on a combination of the particle and Kalman formulation (i.e. LETKF (Hunt et al., 2007), En4DVar (Clayton et al., 2013)). However, the accuracy of the filtering depends on computational bottlenecks in the number of particles and scale of the discretized simulation.

Fortunately, Deep-Learning surrogates offer a compute efficient alternative to traditional solvers. Learning to reconstruct the state given a history of measurements would require an unreasonable amount of data. Thus, neural simulators used for DA are trained on windows of few timesteps and learn a Markovian model of the dynamics. Because the state information is not sufficient to get a deterministic prediction of the future timesteps, this work will focus on generative emulators that learn a distribution of the possible states.

Most generative DA methods train the surrogate like a standard *conditional* forecast model, i.e. they expect the full past state as input (Huang et al., 2024). In this work, we argue that this conditioning discards the uncertainty of the past distribution (Section 4). We therefore adopt a *joint* model that learns to generate both the past and future states. Our main contribution is a new diffusion-based assimilation algorithm that combines the score of the joint state with the En4DVar framework (Section 5). The solution uses an ensemble of particles, which offers a more robust a priori of the past. The superiority of the method is demonstrated on three challenging assimilation experiments (Section 6).



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

**Notations** We study the state of a physical system $\mathbf{x} \in \mathbb{R}^d$ with observations $\mathbf{y} \in \mathbb{R}^{d_y}$. The superscript $k \in [0, T]$ denotes the time index, with $T$ the trajectory length. When considering an ensemble of size $N$, each particle is indexed with the $(i)$ superscript, where $i \in [\![1, N]\!]$. Given a symmetric positive definite matrix $\mathbf{M} \in \mathcal{S}^d_{++}$, we define the Mahalanobis norm as $\|\mathbf{x}\|^2_{\mathbf{M}} = \mathbf{x}^\top \mathbf{M}^{-1}\mathbf{x}$. Finally, to denote an intermediate state of the diffusion process, we use the subscript $\mathbf{x}_t$, where $t$ is the noise level.

## 2 Data Assimilation and filtering

**Dynamics and observation** We consider a nonlinear state-space with forward model $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^d$ and linear observation matrix $\mathbf{H} \in \mathbb{R}^{d_y \times d}$. The error terms $\boldsymbol{\epsilon}^k \in \mathbb{R}^d$, $\boldsymbol{\eta}^k \in \mathbb{R}^{d_y}$ are assumed to be independent, $\mathbf{Q}$ and $\mathbf{R}$ are the model and observation error covariances, respectively.

$$\mathbf{x}^{k+1} = \mathcal{A}(\mathbf{x}^k) + \boldsymbol{\epsilon}^k \qquad \boldsymbol{\epsilon}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{1}$$

$$\mathbf{y}^{k+1} = \mathbf{H}\mathbf{x}^{k+1} + \boldsymbol{\eta}^{k+1}, \qquad \boldsymbol{\eta}^{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{2}$$

The goal of online data assimilation or filtering is to recover the distribution of the true state from previous observations $p(\mathbf{x}^{k+1}|\mathbf{y}^{1:k+1})$.

**Assimilation with 4DVar** Let us assume that we have access to the solution of the previous filtering step $p(\mathbf{x}^k|\mathbf{y}^{1:k})$, which we call the *background* distribution. At time $k+1$, we look for the expected future state $\hat{\mathbf{x}}^{k+1} := \mathbb{E}[\mathbf{x}^{k+1}|\mathbf{y}^{1:k+1}]$. When $\mathcal{A}$ is linear, estimating $\hat{\mathbf{x}}^{k+1}$ is equivalent to finding the Maximum a posteriori (MAP) of the joint variable $(\mathbf{x}^k, \mathbf{x}^{k+1}|\mathbf{y}^{1:k+1})$:

$$\check{\mathbf{x}}^k, \hat{\mathbf{x}}^{k+1} = \underset{\mathbf{x}^k, \mathbf{x}^{k+1}}{\arg\max} \quad p(\mathbf{y}^{k+1}|\mathbf{x}^{k+1})p(\mathbf{x}^{k+1}|\mathbf{x}^k)p(\mathbf{x}^k|\mathbf{y}^{1:k}) \tag{3}$$

where $\check{\mathbf{x}}^k := \mathbb{E}[\mathbf{x}^k|\mathbf{y}^{1:k+1}]$ denotes the smoothed past. Since we only need the most recent estimate $\hat{\mathbf{x}}^{k+1}$, we can discard the smoothed variable $\check{\mathbf{x}}^k$. For simplicity, we assume that the background is Gaussian, $p(\mathbf{x}^k|\mathbf{y}^{1:k}) = \mathcal{N}(\mathbf{x}^k|\hat{\mathbf{x}}^k, \mathbf{P})$, with constant covariance $\mathbf{P}$. In the general nonlinear case, an approximate MAP solution to (3) is obtained by differentiating the log-probability and applying iterative convex optimization (Sanz-Alonso et al., 2023). The weak-constraint **4DVar** assimilation minimizes the following loss:

$$J_{\text{4DVar}}(\mathbf{x}^k, \mathbf{x}^{k+1}) = \frac{1}{2}\|\mathbf{x}^{k+1} - \mathcal{A}(\mathbf{x}^k)\|^2_{\mathbf{Q}} + \frac{1}{2}\|\mathbf{x}^k - \hat{\mathbf{x}}^k\|^2_{\mathbf{P}} + \frac{1}{2}\|\mathbf{y}^{k+1} - \mathbf{H}\mathbf{x}^{k+1}\|^2_{\mathbf{R}} \tag{4}$$

**Ensemble filtering** In most real systems however, the background variance $\mathbf{P}$ is not stationary. An improvement is the **En4DVar** algorithm, which uses an ensemble of $N$ particles $\mathbf{x}^{k,(i)}$ where $i \in [\![1, N]\!]$. For each particle, we solve a modified 4DVar objective where $\mathbf{P}$ is replaced by the empirical covariance $\mathbf{P}^k = \frac{1}{N}\sum_{i=1}^N (\mathbf{x}^{k,(i)} - \bar{\mathbf{x}}^k)(\mathbf{x}^{k,(i)} - \bar{\mathbf{x}}^k)^\top$, where $\bar{\mathbf{x}}^k$ is the empirical mean.

$$\check{\mathbf{x}}^{k,(i)}, \mathbf{x}^{k+1,(i)} := \underset{\mathbf{x}^{k:k+1}}{\arg\max} \quad p(\mathbf{y}^{k+1}|\mathbf{x}^{k+1})p(\mathbf{x}^{k+1}|\mathbf{x}^k)\mathcal{N}(\mathbf{x}^k|\mathbf{x}^{k,(i)}, \mathbf{P}^k), \quad \forall i \in [\![1, N]\!] \tag{5}$$

Note that the background doesn't use the ensemble mean $\bar{\mathbf{x}}^k$, but the particle $\mathbf{x}^{k,(i)}$ to avoid ensemble collapse. The output of the ensemble filter is the empirical distribution $p(\mathbf{x}^k|\mathbf{y}^{1:k}) \approx \frac{1}{N}\sum_{i=1}^N \delta(\mathbf{x}^k - \mathbf{x}^{k,(i)})$.

## 3 Score-based data assimilation

### 3.1 Posterior Sampling with Diffusion models

**Denoising Score Matching** Diffusion models are a generative method to sample from a target data distribution $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ starting from initial Gaussian noise $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{x}_1; \mathbf{0}, \mathbf{I})$, where $(\mathbf{x}_0, \mathbf{x}_1) \in \mathbb{R}^d \times \mathbb{R}^d$. Let $t \in [0, 1]$, the random variable $\mathbf{x}_t$ follows the Gaussian probability path $\mathbf{x}_t = a_t\mathbf{x}_0 + b_t\mathbf{x}_1$ where $a_t, b_t$ are the scale and noise coefficients in $\mathbb{R}$. At all times, we have the relation $\mathbf{x}_t|\mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}_t \mid a_t\mathbf{x}_0, b_t^2\mathbf{I})$. For conciseness, we omit the details of the training and generation process and refer the interested reader to Appendix A or to the original papers for details (Ho et al., 2020; Song et al., 2021). We now assume that we have learned an approximation of the score function $\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ from a dataset of physical states $\mathbf{x}_0 \sim p_{\text{data}}$.

**Observation Guidance** Let us return to the observation equation (2). The following method is valid for any timestep, thus we omit the $k$ in our notation. Given the likelihood $p(\mathbf{y}|\mathbf{x}_0) = \mathcal{N}(\mathbf{y}|\mathbf{H}\mathbf{x}_0, \mathbf{R})$, we are interested in generating *posterior samples* $p(\mathbf{x}_0|\mathbf{y})$ that agree with the observations. This is an example of *inverse problem* solving, which has been extensively studied in both diffusion and assimilation literature. A popular solution is Diffusion Posterior Sampling (DPS) (Chung et al., 2024), a method to condition the generation of score-based models given a differentiable likelihood function $p(\mathbf{y}|\mathbf{x}_0)$. Interestingly, the conditioning is performed at inference time, without any training on $\mathbf{y}$. DPS adds a *guidance* term to the score function:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) \approx s_\theta(\mathbf{x}_t, t) - \frac{1}{2}\nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathbf{H}\tilde{\mathbf{x}}_0(\mathbf{x}_t)\|^2_{\tilde{\mathbf{R}}} \tag{6}$$

Where we approximated $\mathbf{x}_0$ with Tweedie's denoising expectation $\tilde{\mathbf{x}}_0(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] \approx \frac{1}{a_t}(\mathbf{x}_t + b_t^2 s_\theta(\mathbf{x}_t, t))$ and the gradient is computed with auto-differentiation. The variance $\tilde{\mathbf{R}}$ is chosen as a function of the observation error $\mathbf{R}$ (see Appendix A.2 for details). It is common practice to treat $\mathbf{R}$ as a hyperparameter and to choose a different value from the real observation noise $\mathbf{R}_{\text{true}}$.

### 3.2 Trajectory Sampling

Until now, we have reviewed generative models that ignore the temporal dynamics from the state space model. For trajectory generation, we can train our score-based model on a dataset of state pairs $(\mathbf{x}^k, \mathbf{x}^{k+1})$ [1]. At inference time, by letting the model access the previously generated state, we can iteratively generate a rollout of any length. In the literature, two architectures have been proposed to obtain an autoregressive (AR) diffusion. The model can either be trained to generate only the future state $\mathbf{x}^{k+1}$ *conditioned* on the past $\mathbf{x}^k$, or to generate a pair of *joint* states $\mathbf{x}^{k:k+1}$. The two modalities are usually regarded as equivalent.

**Conditional AR** The most common practice is to learn the *conditional* score $s_\theta(\mathbf{x}_t^{k+1}|\mathbf{x}^k)$, where a clean background state $\mathbf{x}^k$ is fed as additional input to the score network (Savary et al., 2026). The observations can be ingested with DPS guidance (6):

$$\nabla_{\mathbf{x}_t^{k+1}} \log p(\mathbf{x}_t^{k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) \approx \underbrace{s_\theta(\mathbf{x}_t^{k+1}|\mathbf{x}^k, t)}_{\text{conditional score}} - \underbrace{\frac{1}{2}\nabla_{\mathbf{x}_t^{k+1}} \|\mathbf{y}^{k+1} - \mathbf{H}\tilde{\mathbf{x}}_0^{k+1}(\mathbf{x}_t^{k+1}|\mathbf{x}^k)\|^2_{\tilde{\mathbf{R}}}}_{\text{guidance on observation } k+1} \tag{7}$$

**Joint AR** The other possibility is to denoise both timesteps jointly. An early version of this joint score was proposed in SDA (Rozet & Louppe, 2023) for smoothing, and then adapted to filtering by Shysheya et al. (2024) using reconstruction guidance (Ho et al., 2022) to obtain a joint autoregressive model:

$$\nabla \log p(\mathbf{x}_t^{k:k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) \approx \underbrace{s_\theta(\mathbf{x}_t^{k:k+1}, t)}_{\text{joint score}} - \underbrace{\frac{1}{2}\nabla \|\mathbf{x}^k - \tilde{\mathbf{x}}_0^k\|^2_{\tilde{\mathbf{P}}}}_{\text{reconstruction guidance}} - \underbrace{\frac{1}{2}\nabla \|\mathbf{y}^{k+1} - \mathbf{H}\tilde{\mathbf{x}}_0^{k+1}\|^2_{\tilde{\mathbf{R}}}}_{\text{guidance on observation } k+1} \tag{8}$$

For readability, we have omitted that the gradient is taken with respect to $\mathbf{x}_t^{k:k+1}$ and that the Tweedie expectations $\tilde{\mathbf{x}}_0^k(\mathbf{x}_t^{k:k+1})$ and $\tilde{\mathbf{x}}_0^{k+1}(\mathbf{x}_t^{k:k+1})$ depend on the state. The reconstruction guidance ensures that at the end of denoising, $\mathbf{x}_0^k \approx \mathbf{x}^k$ with variance $\mathbf{P} = \sigma_{\min}^2\mathbf{I}$ as penalty (the deviation $\sigma_{\min}$ is chosen as a small value). Additional motivation and sampling variants are available in Appendix B.

**Prediction and correction** Notice that both approaches view the previous state $\mathbf{x}^k$ as exact conditioning, not as a background distribution. Shysheya et al. (2024) show that the joint formulation is more flexible but that a conditional score is generally more efficient for long rollouts (without observations). However, if the state estimation $\mathbf{x}^k$ is wrong and rich observations $\mathbf{y}^{k+1}$ are available, the joint model will be better at *correcting* itself, because it is only weakly constrained by the reconstruction guidance. For conditional models, the correction mechanism has to be emulated by using an ensemble of particles and eliminating less probable states, as illustrated in recent studies

[1] Note that for simplification we focus on 2-step windows but the state $\mathbf{x}$ can be chosen as a concatenation of several time-lags for additional context.

(Savary et al., 2026; Wei et al., 2026). Note that another research direction is to learn a *static* model of a single timestep, and to rely on the numerical simulator for the dynamics (Andrae et al., 2026). More details on related works are in Appendix B.2.

## 4 Current Models Failure Modes

In this section, we identify the limitations of current filtering algorithms. We argue that the existing methods were designed for scenarios with perfect initial condition, low noise and homogeneous observations. Thus, they are not robust to more difficult settings encountered in real systems. We propose to study three toy scenarios on 1D Kuramoto-Sivashinsky fluid data to highlight the challenges of generative data assimilation. The algorithms are run with 10 particles and the performance is evaluated with CRPS, which measures the error of the ensemble empirical distribution (see Appendix D.4 for the metrics definition).

**Correcting an imperfect background** At the beginning of the trajectory, existing data assimilation benchmarks assume that the filters are initialized with a good first estimate of the state. However, it is common for Kalman Filters to start with a rough estimate and slowly increase the fidelity of the reconstruction by ingesting observations. We study this ability to warm-up and stabilize in **Scenario 1 (Warmup)**, the initial condition is blurred and the model must correct the state with sparse measurements. Each cell is observed with Bernoulli probability $p = 0.5$ and the observation noise is set to $\sigma_{\text{obs}} = 0.1$. In Figure 1 (left), we observe that the methods relying on exact conditioning (Joint and Conditional) have a large inertia and fail to correct their trajectory. A solution is to modify the guidance hyperparameters: either increase P or decrease R. We obtain a *volatile* model that quickly adapts to incoming observations. This correction mechanism is crucial when observations are more informative than the background, notably when the estimated state has drifted from the true trajectory.

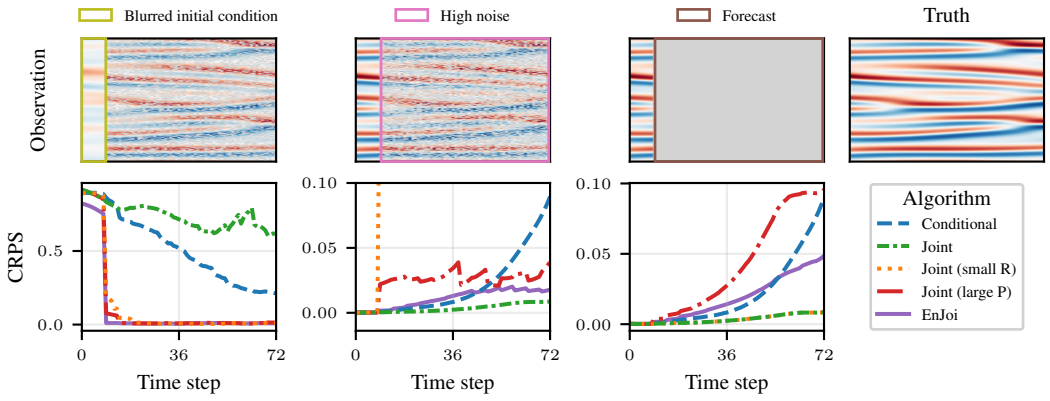


Figure 1: Three assimilation scenarios on 1D Kuramoto Sivashinsky data. **Left: Scenario 1 (Warmup)**, the initial condition is perturbed and the filter must correct itself with the observations. **Center: Scenario 2 (Noise)**, from perfect initial state, the filter should assimilate noisy measurements. **Right: Scenario 3 (Forecast)**, the model must predict the state without observations. Our proposed EnJoi Score Filter offers the best compromise. We note that our method isn't the most efficient to forecast from perfect initial conditions, but we argue that this is not an issue in the assimilation setting.

**Balancing observation and background** We have seen that current models are too stiff to rapidly correct their trajectory. However, this inertia can be beneficial when confronted with noisy observations. Modifying the guidance to obtain a *volatile* model (e.g. by choosing a small R) will cause the filter to be overly sensitive to measurements. In **Scenario 2 (Noise)**, we start from perfect initial conditions, use the same Bernoulli observations ($p = 0.5$) but increase the observation noise to $\sigma_{\text{obs}} = 0.5$. As seen in Figure 1 (center), the *volatile* models are not robust to the increased noise. The model with small $R$ inherits the errors of the measurements and the score network is unable to produce a smooth physical state. In **Scenario 3 (Forecast)**, we let the algorithms roll out their

prediction from perfect initial conditions without any observation. In Figure 1 (right), we observe that choosing a larger P causes the Joint model to quickly forget the background distribution and to drift from the true trajectory. These experiments show that a correct equilibrium must be found between the confidence on the past, trust in the physical model and sensitivity to observations. When the uncertainty is stable in time, it is common to use $\mathbf{P}$ and $\mathbf{R}$ that are constant across the trajectory length and the state dimension, but this is not always a good approximation in real settings.

**Spatial uncertainty** Once the correct tradeoff between observation and past has been found, there is still the issue of spatially-localized uncertainty. In real systems, it is common to have good coverage of some regions and poor information on others. In the case of moving sensors, data loss or missing values, the uncertainty pattern can evolve in space-time. It is also common to have highly chaotic regions with higher uncertainty due to the dynamics of the system (e.g. wave fronts, attractors). This spatial uncertainty is not incorporated into current diffusion algorithms: the joint model from Shysheya et al. (2024) uses a constant diagonal guidance, while particle-filter methods compute likelihood weights for the entire state-space at once.

These experiments motivate that our current approximation of the background distribution is not flexible enough for the assimilation setting. The exact conditioning hypothesis will not hold for long rollouts, the constant confidence in the past state is a simplification and the spatial dimension is not taken into account.

## 5 Proposed Method

Modern data assimilation algorithms use an ensemble of particles to account for the diversity of the filtering distribution. We propose to adapt the En4DVar (5) framework, which maintains an expressive background prior by computing the covariance of previous predictions. Instead of optimizing the state by differentiating the numerical model (4), we will use a joint generative model.

### 5.1 Guidance with ensemble covariance

We first need to adapt 4DVar (3) to a setting with diffusion noise. This can be achieved with a series of approximations to extract one guidance for the past and one for the future.

$$
\begin{aligned}
p(\mathbf{x}_t^{k:k+1}|\mathbf{y}^{1:k+1}) &\propto p(\mathbf{x}_t^{k:k+1}) p(\mathbf{y}^{1:k+1}|\mathbf{x}_t^{k:k+1}) \\
&\approx p(\mathbf{x}_t^{k:k+1}) \overbrace{p(\mathbf{y}^{1:k}|\mathbf{x}_t^{k:k+1}) p(\mathbf{y}^{k+1}|\mathbf{x}_t^{k:k+1})}^{\approx \text{independent}} \\
&\propto p(\mathbf{x}_t^{k:k+1}) \overbrace{p(\mathbf{x}_t^{k:k+1}|\mathbf{y}^{1:k})}^{\approx \text{use prior as likelihood}} p(\mathbf{y}^{k+1}|\mathbf{x}_t^{k:k+1}) \\
&\underset{\text{DPS}}{\approx} p(\mathbf{x}_t^{k:k+1}) \overbrace{\mathcal{N}(\tilde{\mathbf{x}}_0^k|\hat{\mathbf{x}}^k, \mathbf{P}^k)}^{\text{Kalman recurrence}} \mathcal{N}(\mathbf{y}^{k+1}|\mathbf{H}\tilde{\mathbf{x}}_0^{k+1}, \mathbf{R})
\end{aligned} \tag{9}
$$

Where we have used DPS to replace the state $\mathbf{x}_t^k$ with $\tilde{\mathbf{x}}_0^k$. Note that the Gaussian background prior acts as a likelihood. This is for practical considerations and is further motivated in Appendix C.1. We can then derive a score that is similar to Joint AR (8), but the guidance strength is now dependent on the ensemble covariance $\mathbf{P}^k$.

**An issue when guidance is small** Equation (9) allows us to approximately sample from the filtering distribution. However this is not enough to fulfill our objective, since the En4DVar algorithm (5) requires the MAP. With posterior sampling, there is no guarantee that a sample will be from the most probable mode. If we are in a long term rollout, $\mathbf{P}^k$ will grow, and if no observation is available, the observation guidance will vanish (infinite $\mathbf{R}$). In this situation with low guidance signal, the diffusion model will generate unconditional samples and the filter will tend to forget its past.

### 5.2 Scaling the Guidance

**Balancing guidance terms** To maintain strong guidance at all times, we propose to scale both guidance terms together. This is a way to preserve the balance between observation noise and past

uncertainty. We rewrite the noisy MAP problem as the sum of two conflicting objectives:

$$\arg\max_{\mathbf{x}_t^{k:k+1}} \underbrace{\log p(\mathbf{x}_t^{k:k+1})}_{\text{consistency}} + \underbrace{\log \mathcal{N}(\tilde{\mathbf{x}}_0^k | \hat{\mathbf{x}}^k, \mathbf{P}^k)\mathcal{N}(\mathbf{y}^{k+1} | \mathbf{H}\tilde{\mathbf{x}}_0^{k+1}, \mathbf{R})}_{\approx -J^\star_{\text{4DVar}}(\tilde{\mathbf{x}}_0^{k:k+1})}$$

Where the diffusion score is acting as a regularizer, pushing the noisy estimates towards solutions that are consistent with both the physics and the noise level. The second term can be interpreted as the 4DVar objective (4) without a physical model ($\mathbf{Q} = +\infty$). We normalize the guidance term with the root mean square to ensure that the two objectives are balanced.

$$\mathbf{s} = \nabla_{\mathbf{x}_t^{k:k+1}} \log p(\mathbf{x}_t^{k:k+1}) - \gamma\sqrt{2d}\frac{\nabla_{\mathbf{x}_t^{k:k+1}} J^\star_{\text{4DVar}}(\tilde{\mathbf{x}}_0^{k:k+1})}{\left\|\nabla_{\mathbf{x}_t^{k:k+1}} J^\star_{\text{4DVar}}(\tilde{\mathbf{x}}_0^{k:k+1})\right\|_2} \tag{10}$$

Where $2d$ is the dimension of the joint state $\mathbf{x}_t^{k:k+1}$ and we introduce the guidance strength hyper-parameter $\gamma$ which we set to 1. Similar to Rozet & Louppe (2023), we apply an additional step of Langevin sampling after the SDE integration step, to correct the approximation of the guidance signal.

$$\mathbf{x}_t^{k:k+1} \leftarrow \mathbf{x}_t^{k:k+1} + \tau \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|_2} + \sqrt{\tau}\epsilon_\mathbf{c}, \quad \epsilon_c \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \tau \in \mathbb{R} \tag{11}$$

**Related Guidance methods** Rescaling the guidance and keeping only directional information has been proposed in other works. In particular, Spherical Guidance (Yang et al., 2024) proposes to project the state to the sphere of center $\mathbb{E}[\mathbf{x}_{t-1}|\mathbf{x}_t]$ with a radius proportional to $\sqrt{d}$ to stay in the denoising manifold. Later work DMAP (Xu et al., 2025) adopts an optimization perspective and proposes to alternate spherical projection with DPS steps. Their algorithms are very close to our scaled guidance with Langevin corrections.

**Maintaining ensemble spread** Because of the random noise initialization, the model will generate various scenarios in the areas where the guidance is not constraining the generation. If there are multiple local maxima in the posterior distribution, using the MAP formulation will not make the filter collapse to a single particle. This bears resemblance to a stochastic En4DVar framework where model noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is injected into the simulation to increase ensemble spread.

### 5.3 The Ensemble Joint Score Filter algorithm

We refer to our method as Ensemble Joint Score Filter (*Enjoi*), since it is the combination of En4DVar with a Joint score function. The iterative filtering process is described in Algorithm 1 which closely resembles the classical En4DVar algorithm. The key difference is that the approximate MAP is generated using a joint score neural network with our scaled guided sampling. We detail this modified generation process in Algorithm 2.

Compared to other diffusion filters, the novelty of our approach is to incorporate the ensemble covariance into the sampling. This has several benefits, which stem from the En4DVar framework. Our algorithm takes full advantage of the **empirical past uncertainty**: the reconstruction chooses the right balance between the uncertain past and the noisy observation. This is an improvement over auto-regressive methods that set a constant confidence to the past state (e.g. using a specific noise level (Andrae et al., 2026; Jia et al., 2026) or guidance strength (Shysheya et al., 2024)). The other strength of our method is its ability to perform **local updates**, which alleviates the curse of dimensionality of Particle Filters (Savary et al., 2026; Wei et al., 2026; Bao et al., 2023). Particles that diverge from the MAP are locally course-corrected instead of being eliminated. Thus, we do not lose the entire state information. This allows the filter to track multiple sources of uncertainty with a reduced ensemble size.

**Background Covariance** To avoid spurious correlations in the empirical covariance, we apply Gaspari-Cohn localization (Gaspari & Cohn, 1999). This classical DA method applies a spatio-temporal kernel function with bounded support to the covariance matrix. The localization radius must be chosen empirically considering the temporal and spatial scale of the dynamical process. A preconditioning diagonal term $\sigma_{\min}$ is also added for stability, we use the same value as the reconstruction guidance (8). The constant covariance term is meant to represent stationary or climatic variance. Operational DA systems often rely on *hybrid* covariance matrix for robustness.

**Algorithm 1** EnJoi: Ensemble Joint Score Filter

**Require:** Initial ensemble $\{\mathbf{x}^{k=0,(i)}\}_{i=1}^{N}$, observations $\{\mathbf{y}^{k+1}\}_{k=0}^{T-1}$, $\mathbf{P}^0$
1: **for** $k = 0, \dots, T-1$ **do**
2: $\quad \mathbf{x}^{k+1,(i)} \leftarrow \mathrm{JointDiffusion}(\mathbf{x}^{k,(i)}, \mathbf{P}^k, \mathbf{y}^{k+1}), \quad \forall i \in [\![1, N]\!]$ ▷ Sample approximate MAP
3: $\quad \bar{\mathbf{x}}^{k+1} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^{k+1,(i)}$
4: $\quad \mathbf{P}^{k+1} \leftarrow \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{k+1,(i)} - \bar{\mathbf{x}}^{k+1})(\mathbf{x}^{k+1,(i)} - \bar{\mathbf{x}}^{k+1})^\top$ ▷ Estimate background covariance
5: $\quad \mathbf{P}^{k+1} \leftarrow \mathrm{GaspariCohn}(\mathbf{P}^{k+1}) + \sigma_{\min}$ ▷ Apply localization (Gaspari & Cohn, 1999)
6: **end for**
7: **return** Empirical distribution $\{\mathbf{x}^{1:T,(i)}\}_{i=1}^{N}$ and covariance $\{\mathbf{P}^k\}_{k=1}^{T}$

**Algorithm 2** Guided Joint Diffusion Sampling

**Require:** Background state $\mathbf{x}^{k,(i)}$, state covariance $\mathbf{P}^k$, observation $\mathbf{y}^{k+1}$, Observation covariance $R$, score $s_\theta$, correction stepsize $\tau$, guidance strength $\gamma$.
1: $\mathbf{x}_t^{k:k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Sample random noise
2: **for** $t = 1, 1-\Delta t, \dots, \Delta t$ **do**
3: $\quad \tilde{\mathbf{x}}_0^{k:k+1} \leftarrow \frac{1}{a_t}(\mathbf{x}_t^{k:k+1} + b_t^2 s_\theta(\mathbf{x}_t^{k:k+1}))$ ▷ Approximate reconstruction with Tweedie
4: $\quad \mathbf{s} \leftarrow s_\theta(\mathbf{x}_t^{k:k+1}, t) - \gamma\sqrt{2d}\frac{\nabla_{\mathbf{x}_t^{k:k+1}} J^\star_{\text{4DVar}}(\tilde{\mathbf{x}}_0^{k:k+1})}{\|\nabla_{\mathbf{x}_t^{k:k+1}} J^\star_{\text{4DVar}}(\tilde{\mathbf{x}}_0^{k:k+1})\|}$ ▷ Add guidance (10) to the score
5: $\quad \mathbf{x}_{t-\Delta t}^{k:k+1} \leftarrow \mathrm{SDESolverStep}(\mathbf{x}_t^{k:k+1}, \mathbf{s}, t)$ ▷ Integrate backward SDE (Lu et al., 2022)
6: $\quad \mathbf{s}' \leftarrow$ recompute lines 3–4 at $t - \Delta t$ ▷ Recompute the guided score
7: $\quad \mathbf{x}_{t-\Delta t}^{k:k+1} \leftarrow \mathrm{LangevinStep}(\mathbf{x}_{t-\Delta t}^{k:k+1}, \mathbf{s}', \tau)$ ▷ Apply one Langevin correction (11)
8: **end for**
9: **return** Denoised state $\mathbf{x}_0^{k+1}$

## 6 Experiments

We compare our **EnJoi** algorithm to two approaches from the recent literature. The first baseline consists of i.i.d. runs of Conditional AR Diffusion from Shysheya et al. (2024). We use their implementation of **Joint AR** (8) and **Cond. AR** (7). For comparison with a recent ensemble method, we re-implement the particle filter from Savary et al. (2026) which we refer to as **Cond. P.F.**. We also generalize the method to the joint score, resulting in a **Joint P.F.**.

Motivated by our experiments in Section 4, we explore the possibility of selecting a smaller variance $R$ than the real measurement noise $\sigma^2_{\text{obs}}$. We denote this modification as *volatile* (v) models. For clarity, the algorithms using the true variance $\sigma^2_{\text{obs}}$ are tagged with an (s) for *steady*. We use *(g=SDA)* to denote the guidance from Rozet & Louppe (2023), *(g=scaled)* is our scaled guidance (10) with constant $\mathbf{P}$, and *(g=MMPS)* is the Moment-Matching Posterior Sampling from Rozet et al. (2024). Unless specified otherwise, the number of particles is set to 10 for all algorithms.

### 6.1 1D Kuramoto Sivashinsky

**Data** We use the KS dataset from Shysheya et al. (2024). This synthetic experiment considers a 1D fluid simulation with a chaotic behaviour, which makes the assimilation challenging. In the *Bernoulli* setting, each cell of the space-time grid has probability $p = 0.1$ to be observed. We experiment with different observation noise levels to showcase the sensitivity of the baseline algorithms. In addition, we propose two new observation patterns that are inhomogeneous in space-time: *Sine* (a single sensor with a sine trajectory) and *Exponential* (a full scan with exponentially distributed waiting times) (see Figure 2). The full details of the experimental setup are presented in Appendix D.

**Results** Table 1 presents the evaluation scores of the filtering algorithms on a dataset of 10 trajectories of 640 steps. The approaches that are too confident in the past state (e.g. the *conditional* models) are failing, while the *volatile* models perform better. The baselines are not equally robust to the different observation noise levels, in particular the volatile Joint AR fails in the high noise setting. The Joint PF obtains robust results in situations where the observations are homogeneous

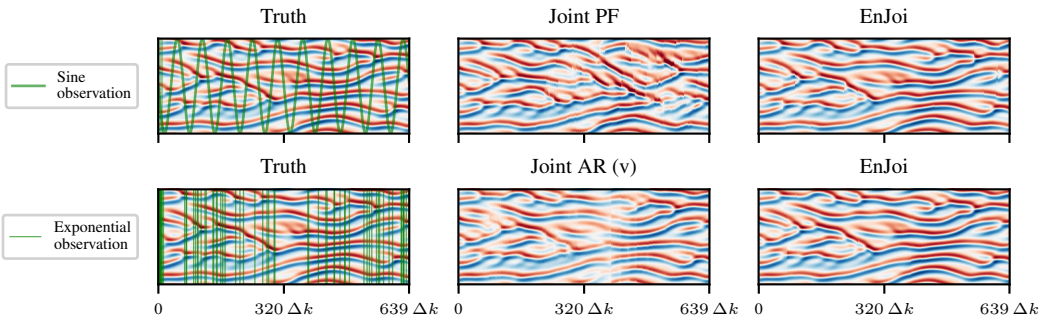


Figure 2: Kuramoto Sivashinsky assimilation experiment with non-homogneous observation patterns. The two left columns show the mean of an ensemble of 10 particles.

Table 1: Kuramoto Sivashinsky CRPS (↓) average ±3 standard errors on the mean. We denote *volatile* models with (v) and regular *steady* models with (s), the guidance method is (g)

| Obs. shape $H(k)$ | Bernoulli | | | Sine | Exponential |
|---|---|---|---|---|---|
| Obs. noise std. $\sigma_{\text{obs}}$ | 0.1 | 0.5 | 2 | 0.5 | 0.5 |
| Cond. P.F. (s) (g=MMPS) | 0.568 ± 0.000 | 1.042 ± 0.133 | 1.029 ± 0.094 | 0.976 ± 0.144 | 0.921 ± 0.126 |
| Joint AR (s) (g=SDA) | 0.491 ± 0.136 | 0.619 ± 0.092 | 0.694 ± 0.031 | 0.598 ± 0.058 | 0.591 ± 0.051 |
| Cond. AR (s) (g=SDA) | 0.457 ± 0.096 | 0.606 ± 0.128 | 0.576 ± 0.087 | 0.574 ± 0.086 | 0.571 ± 0.078 |
| Joint AR (v) (g=scaled) | 0.088 ± 0.010 | 0.093 ± 0.008 | **0.200** ± 0.017 | 0.516 ± 0.036 | 0.213 ± 0.026 |
| Joint P.F. (v) (g=scaled) | 0.013 ± 0.001 | 0.063 ± 0.003 | 0.239 ± 0.014 | 0.546 ± 0.129 | 0.120 ± 0.019 |
| Joint AR (v) (g=SDA) | 0.012 ± 0.001 | 0.091 ± 0.003 | 0.407 ± 0.011 | 0.293 ± 0.049 | 0.136 ± 0.017 |
| EnJoi (s) (ours) | **0.010** ± 0.000 | **0.050** ± 0.004 | 0.229 ± 0.054 | **0.162** ± 0.069 | **0.097** ± 0.029 |

in space. However, it fails in the *Sine* experiment with the moving sensor. We attribute this failure to its incapacity to correct the ensemble locally in space. Overall, our EnJoi algorithm is the most robust to all scenarios.

### 6.2 2D Kolmogorov Flow

**Data** This experiment considers a 2D simulation of Navier Stokes flow with random forcing. The observations are laid out on a sparse grid (see Figure 3) and we add a small observation noise $\sigma_{\text{obs}} = 0.1$. The setup is similar to Savary et al. (2026), although our domain is half as wide (64x64). See Appendix D.2 in Shysheya et al. (2024) for the data generation process.

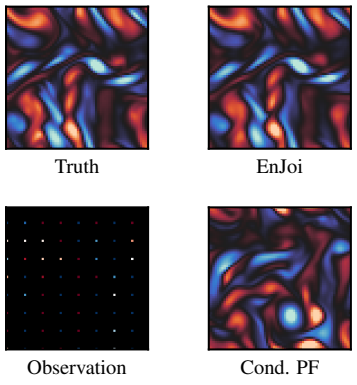


Figure 3: Final timestep of a Kolmogorov Flow assimilation experiment with sparse (8x8) observations. The left column displays one sample from the filter ensemble.

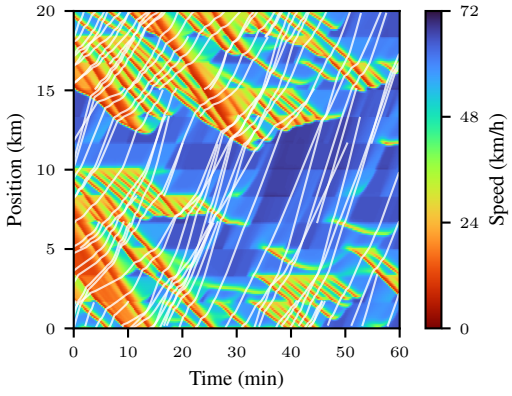


Figure 4: ARZ traffic flow simulation on a ring road with ramps. The velocity field is observed by only 2.5% of vehicles, their trajectory on the domain is shown in white.

Table 2: Kolmogorov CRPS (↓) average ±3 standard errors on the mean.

| Algorithm | Observed grid points (out of 64x64) | | |
|---|---|---|---|
| | $(4, 4)$ | $(8, 8)$ | $(16, 16)$ |
| Cond. P.F. (s) (g=MMPS) | 0.32 ± 0.04 | 0.245 ± 0.082 | 0.043 ± 0.009 |
| Cond. AR (s) (g=SDA) | 0.23 ± 0.03 | 0.186 ± 0.024 | 0.074 ± 0.018 |
| Joint AR (v) (g=SDA) | <u>0.18</u> ± 0.02 | **0.043** ± 0.002 | <u>0.036</u> ± 0.001 |
| EnJoi (s) (ours) | **0.12** ± 0.02 | <u>0.047</u> ± 0.002 | **0.029** ± 0.001 |

**Results** Table 2 shows averaged results over 10 trajectories of 64 steps. The conditional PF obtains a better performance than i.i.d. runs of Conditional AR in the dense observation regime (16x16), demonstrating the usefulness of particle resampling. However the results do not hold for sparse observations and the Joint models reach better CRPS. In the denser regimes, the performance of Joint AR and EnJoi is comparable, which is expected because of the low noise and static sensor positions.

### 6.3 Highway traffic PDE

**Data** We use a macroscopic second-order traffic simulation based on the PDE from Aw & Rascle (2000); Zhang (2002). The domain is set to a 20km-long ring road with 12 equally spaced ramps. The exit-entry ratios are simulated with random step functions and act as perturbations. We simulate 1 hour of traffic using a finite-volume solver (more details in Appendix D). The observations consist of simulated trajectories from a small portion of probe vehicles that move over the domain, following the mean velocity with a probe-specific variance (See Figure 4)

**Results** This experiment highlights the heterogeneity of real world measurements. With moving sensors, all the sections of the highway are not equally observed, creating a moving uncertainty pattern. The traffic PDE exhibits shockwaves and rapid shift of conditions, making it challenging for algorithms with high inertia. To demonstrate that local correction is more effective than resampling, we use 128 particles for the particle filter and 10 for the other methods. Our algorithm obtains the best reconstruction results, see Table 3.

Table 3: ARZ traffic CRPS (↓) average ± 3 standard errors on the mean

| Algorithm | Penetration rate | |
|---|---|---|
| | 2.5% | 5% |
| Joint AR (s) (g=scaled) | <u>0.21</u> ± 0.14 | 0.21 ± 0.14 |
| Joint P.F. (s) (g=MMPS) | 0.23 ± 0.05 | <u>0.18</u> ± 0.05 |
| EnJoi (s) (ours) | **0.15** ± 0.06 | **0.11** ± 0.03 |

## 7 Conclusion

In this work we have argued that current generative data assimilation algorithms are limited by their strict auto-regressive conditioning. We proposed to take advantage of a flexible *joint* generative model which we condition on a more expressive distribution of background states. Our new filtering algorithm is able to better balance past and future uncertainty and outperforms all baselines on simulated experiments. Its robustness to non-homogeneous observation patterns makes it applicable to a wide range of domains.

One limitation of our method is the increased computation cost: joint scores almost double the inference time and the covariance inverse also adds overhead (see Appendix E). Another limit is that scaling the guidance with the empirical covariance degrades the rollout quality (see the forecast in Figure 1) and necessitates Langevin corrections. Similarly to En4DVar, we make the simplifying assumption that the background is a Multivariate Gaussian, thus we are not able to handle multimodal past priors. An interesting future direction would be to consider more expressive background priors, using tools from Stochastic Interpolants or Optimal Transport.

### AI USE STATEMENT

In this work, we used generative AI tools to implement methods. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results and design or provide feedback on research methodology or experiments. The following are not applicable to this work: formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This paper uses generative AI for the prediction of dynamical systems. Careful evaluation of the model bias and robustness should be conducted before using these predictions for real-world decision making.

### REPRODUCIBILITY STATEMENT

We release the code necessary for reproducing experiments of Section 6. Details on how to generate the synthetic datasets are in Appendix D.

## REFERENCES

Martin Andrae, Erik Larsson, So Takao, Tomas Landelius, and Fredrik Lindsten. DAISI: Data Assimilation with Inverse Sampling using Stochastic Interpolants. In *ICML 2026*. arXiv, 2026. doi: 10.48550/arXiv.2512.00252.

A. Aw and M. Rascle. Resurrection of "Second Order" Models of Traffic Flow. *SIAM Journal on Applied Mathematics*, 60(3):916–938, 2000. doi: 10.1137/S0036139997332099.

Feng Bao, Zezhong Zhang, and Guannan Zhang. A Score-based Nonlinear Filter for Data Assimilation, 2023.

Feng Bao, Hristo G. Chipilski, Siming Liang, Guannan Zhang, and Jeffrey S. Whitaker. Nonlinear Ensemble Filtering with Diffusion Models: Application to the Surface Quasigeostrophic Dynamics. *Monthly Weather Review*, 153(7):1155–1169, 2025. doi: 10.1175/MWR-D-24-0069.1.

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. doi: 10.1002/wcc.535.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *Advances in Neural Information Processing Systems*, number 37, pp. 24081–24125. arXiv, 2024. doi: 10.48550/arXiv.2407.01392.

Siyi Chen, Yixuan Jia, Qing Qu, He Sun, and Jeffrey A. Fessler. FlowDAS: A Stochastic Interpolant-based Framework for Data Assimilation. In *Advances in Neural Information Processing Systems*, volume 2026, pp. 103061–103102. arXiv, 2025. doi: 10.48550/arXiv.2501.16642.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *ICLR 2023*. arXiv, 2024. doi: 10.48550/arXiv.2209.14687.

A. M. Clayton, A. C. Lorenc, and D. M. Barker. Operational implementation of a hybrid ensemble/4D-Var global data assimilation system at the Met Office. *Quarterly Journal of the Royal Meteorological Society*, 139(675):1445–1461, 2013. doi: 10.1002/qj.2054.

Gregory Gaspari and Stephen E Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757, 1999.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Langwen Huang, Lukas Gianinazzi, Yuejiang Yu, Peter D. Dueben, and Torsten Hoefler. DiffDA: A Diffusion Model for Weather-scale Data Assimilation. In *ICML 2024*. arXiv, 2024. doi: 10.48550/arXiv.2401.05932.

Brian R. Hunt, Eric J. Kostelich, and Istvan Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena*, 230(1):112–126, 2007. doi: 10.1016/j.physd.2006.11.008.

Yixuan Jia, Siyi Chen, Yida Pan, Xiao Li, Lianghe Shi, Chanyong Jung, Haijie Yuan, Ismail Alkhouri, Yue Cynthia Wu, Saiprasad Ravishankar, Jeffrey A. Fessler, and Qing Qu. Forcing-DAS: Unified and Robust Data Assimilation via Diffusion Forcing, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *NeurIPS 2022*, 2022. doi: 10.48550/arXiv.2206.00364.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models. *Machine Intelligence Research*, 22(4):730–751, 2025. doi: 10.1007/s11633-025-1562-4.

Julien Moreau, Mustafa Al Bakri, and Marc Lelarge. 4DVar Assimilation of Vehicle Probe Data with a Second-Order Macroscopic Traffic Model. In *IEEE ITSC 2026 - IEEE International Conference on Intelligent Transportation Systems*, Naples, Italy, 2026. IEEE ITSS.

Elise Özalp, Andrea Nóvoa, and Luca Magri. Real-time forecasting of chaotic dynamics from sparse data and autoencoders. *Computer Methods in Applied Mechanics and Engineering*, 450:118600, 2026. doi: 10.1016/j.cma.2025.118600.

François Rozet and Gilles Louppe. Score-based Data Assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

François Rozet, Ruben Ohana, Michael McCabe, Gilles Louppe, François Lanusse, and Shirley Ho. Lost in Latent Space: An Empirical Study of Latent Diffusion Models for Physics Emulation.

François Rozet, Gérôme Andry, François Lanusse, and Gilles Louppe. Learning Diffusion Priors from Observations by Expectation Maximization, 2024.

Daniel Sanz-Alonso, Andrew Stuart, and Armeen Taeb. *Inverse Problems and Data Assimilation*. Cambridge University Press, Cambridge, 2023.

Thomas Savary, François Rozet, and Gilles Louppe. Training-Free Data Assimilation with GenCast. In *Tackling Climate Change with Machine Learning: Workshop at NeurIPS 2025*. arXiv, 2025. doi: 10.48550/arXiv.2509.18811.

Thomas Savary, François Rozet, and Gilles Louppe. Training-Free Bayesian Filtering with Generative Emulators. In *ICML 2026*, 2026.

Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E. Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 23246–23300. arXiv, 2024. doi: 10.48550/arXiv.2410.16415.

Phillip Si and Peng Chen. Latent-EnSF: A latent ensemble score filter for high-dimensional data assimilation with sparse observation data. In *International Conference on Learning Representations*, volume 2025, pp. 94879–94895, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *ICLR 2021*, 2022. doi: 10.48550/arXiv.2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*. arXiv, 2021. doi: 10.48550/arXiv.2011.13456.

Lifu Wei, Yinuo Ren, Naichen Shi, and Yiping Lu. SURGE: Approximation-free Training Free Particle Filter for Diffusion Surrogate. In *ICML 2026*. arXiv, 2026. doi: 10.48550/arXiv.2605.18745.

Daniel B. Work, Sébastien Blandin, Olli-Pekka Tossavainen, Benedetto Piccoli, and Alexandre M. Bayen. A Traffic Model for Velocity Data Assimilation. *Applied Mathematics Research eXpress*, 2010(1):1–35, 2010. doi: 10.1093/amrx/abq002.

Tongda Xu, Xiyan Cai, Xinjie Zhang, Xingtong Ge, Dailan He, Ming Sun, Jingjing Liu, Ya-Qin Zhang, Jian Li, and Yan Wang. Rethinking Diffusion Posterior Sampling: From Conditional Score Estimator to Maximizing a Posterior. In *ICLR 2025*. arXiv, 2025. doi: 10.48550/arXiv.2501.18913.

Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with Spherical Gaussian Constraint for Conditional Diffusion. In *ICML 2024*. arXiv, 2024. doi: 10.48550/arXiv.2402.03201.

H. M. Zhang. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological*, 36(3):275–290, 2002. doi: 10.1016/S0191-2615(00)00050-3.

# A Details on Diffusion Models

## A.1 Score-based Generative Models

Denoising Diffusion models (Ho et al., 2020; Song et al., 2021) generate samples $\mathbf{x}_0 \in \mathbb{R}^d$ of a target data distribution $p_{\text{data}}(\mathbf{x}_0)$ starting from initial Gaussian noise $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{x}_1; 0, I)$. Let $t \in [0, 1]$, we define $a_t, b_t \in \mathbb{R}$ the scale and noise coefficients and $\mathbf{x}_t$ a random variable following the Gaussian probability path $\mathbf{x}_t = a_t \mathbf{x}_0 + b_t \mathbf{x}_1$. At all times, we have the relation $\mathbf{x}_t | \mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}_t \mid a_t \mathbf{x}_0, b_t^2 I)$. From these coefficients, we can define the forward diffusion SDE (12) which noises the input data until it becomes pure Gaussian noise. The associated family of backward SDEs (13) generates clean samples from noise, for any $\eta \in \mathbb{R}$.

$$\mathrm{d}\overrightarrow{\mathbf{x}_t} = f_t \mathbf{x}_t \,\mathrm{d}t + g_t \,\mathrm{d}\mathbf{w}_t \tag{12}$$

$$\mathrm{d}\overleftarrow{\mathbf{x}_t} = \left[ f_t \mathbf{x}_t - \frac{1+\eta^2}{2} g_t^2 \nabla_{x_t} \log p(\mathbf{x}_t) \right] \mathrm{d}t + \eta g_t \,\mathrm{d}\mathbf{w}_t \tag{13}$$

Where $w_t$ is the standard Wiener process in $\mathbb{R}^d$, $f_t \in \mathbb{R}$ and $g_t \in \mathbb{R}^+$ are derived from $a_t$ and $b_t$. In this paper we use the Variance Preserving SDE with coefficients $a_t = \sqrt{\alpha_t}$ and $b_t = \sqrt{1 - \alpha_t}$, $\alpha_t \in \mathbb{R}^+$, which grants the property $a_t^2 + b_t^2 = 1$. The score $\nabla_{x_t} \log p(x_t)$ is approximated with a Neural Network $s_\theta(x_t, t)$ trained by minimizing the score-matching loss (14). In practice, it is more stable to parametrize the score network as a function of the noise-predictor $\epsilon_\theta(x_t, t) = -b_t s_\theta(x_t, t)$, which shares the same minimizer.

$$\mathcal{L}(\theta) = \mathbb{E}_{t, x_0, x_1} \left[ \| \epsilon_\theta(a_t x_0 + b_t x_1, t) - x_1 \|_2^2 \right] \tag{14}$$

Once the score has been trained, there exists a variety of sampling methods to numerically integrate Equation (13) (Song et al., 2022; Karras et al., 2022; Lu et al., 2025). Following Shysheya et al. (2024), we use the DPM solver Lu et al. (2022).

Table 4: Comparison between diffusion trajectory sampling methods.

| Sampling Method $\nabla \log p =$ | Learned score $s_\theta(., t)$ | Observation guidance $+\frac{1}{2}\nabla\|\mathbf{y} - H\tilde{\mathbf{x}}_0\|^2_R$ | Past guidance $+\frac{1}{2}\nabla\|\mathbf{x}^k - \tilde{\mathbf{x}}_0^k\|^2_P$ |
|---|---|---|---|
| Rozet & Louppe (2023) | joint | on all timesteps | no |
| Savary et al. (2026) | conditional | on $k+1$ | no |
| Shysheya et al. (2024) | joint | on $k+1$ | constant $\mathbf{P}$ |
| Ours | joint | on $k+1$ | empirical $\mathbf{P}^k$ of particles |

### A.2 Diffusion Posterior Sampling

Guidance is a test-time method to sample from $p(\mathbf{x}|\mathbf{y})$ with $\mathbf{y}$ not seen during training. Assuming that we have access to a likelihood function $p(\mathbf{y}|\mathbf{x}_0)$, we can condition the score on observations by adding a guidance term:

$$p(\mathbf{x}_t|\mathbf{y}) = p(\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_t)/p(\mathbf{y})$$
$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

The last term is intractable and needs to be approximated. DPS (Chung et al., 2024) proposes to substitute $\mathbf{x}_t$ with the denoising expectation obtained with Tweedie's formula $\tilde{\mathbf{x}}_0(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{a_t}(\mathbf{x}_t + b_t^2 s_\theta(\mathbf{x}_t, t))$. It is common to model $p(\mathbf{x}_0|\mathbf{x}_t)$ as a Gaussian to account for the denoising error.

$$p(\mathbf{y}|\mathbf{x}_t) \approx \int p(\mathbf{y}|\mathbf{x}_0)\mathcal{N}(\mathbf{x}_0|\tilde{\mathbf{x}}_0(\mathbf{x}_t), \tilde{\mathbf{\Sigma}}(\mathbf{x}_t))\,\mathrm{d}\mathbf{x}_0 \tag{15}$$

Where $\tilde{\mathbf{\Sigma}}(\mathbf{x}_t)$ is an approximation of the variance $\mathbb{V}[\mathbf{x}_0|\mathbf{x}_t]$. SDA (Rozet & Louppe, 2023) uses $\tilde{\mathbf{\Sigma}}(\mathbf{x}_t) = (b_t^2/a_t^2)\,\mathbf{\Gamma}$, where $\mathbf{\Gamma}$ is a constant. In the case of Gaussian observation error $p(\mathbf{y}|\mathbf{x}_0) = \mathcal{N}(\mathbf{y}|\mathbf{H}\mathbf{x}_0, \mathbf{R})$, the guided score is

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) \approx s_\theta(\mathbf{x}_t, t) - \frac{1}{2}\nabla_{\mathbf{x}_t}\|\mathbf{y} - \mathbf{H}\tilde{\mathbf{x}}_0(\mathbf{x}_t)\|^2_{\tilde{\mathbf{R}}} \quad \text{where } \tilde{\mathbf{R}} = \mathbf{R} + \mathbf{H}\tilde{\mathbf{\Sigma}}\mathbf{H}^T \tag{16}$$

The gradient is computed by auto-differentiating through the denoiser. The approximation (15) and the sampler discretization often cause the state to deviate from the exact denoising manifold induced by $p(\mathbf{x}_t)$. This issue is exacerbated when the guidance signal is large, thus a solution is to stabilize the sampling with Langevin corrections (see Equation (11))

$$\mathbf{x}_t \leftarrow \mathbf{x}_t + \tau \cdot \frac{\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y})}{\|\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y})\|} + \sqrt{\tau}\epsilon_{\mathbf{c}}, \quad \epsilon_c \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{17}$$

## B Related works

### B.1 Trajectory sampling

This section provides additional details on the related sequential diffusion sampling approaches. Also see Table 4 for comparison.

**SDA** Rozet & Louppe (2023) introduce the smoothing score:

$$p(\mathbf{x}_t^{k:k+1}|\mathbf{y}^{k:k+1}) \propto p(\mathbf{x}_t^{k:k+1})p(\mathbf{y}^{k:k+1}|\mathbf{x}_t^{k:k+1})$$
$$\underset{\text{DPS}}{\approx} p(\mathbf{x}_t^{k:k+1})\mathcal{N}(\mathbf{y}^{k:k+1}|H\tilde{\mathbf{x}}_0^{k:k+1}, \tilde{R})$$
$$\nabla_{\mathbf{x}_t^{k:k+1}} \log p(\mathbf{x}_t^{k:k+1}|\mathbf{y}^{k:k+1}) \approx \underbrace{s_\theta(\mathbf{x}_t^{k:k+1})}_{\text{joint score}} - \underbrace{\frac{1}{2}\nabla_{\mathbf{x}_t^{k:k+1}}\|\mathbf{y}^{k:k+1} - H\tilde{\mathbf{x}}_0^{k:k+1}(\mathbf{x}_t^{k:k+1})\|^2_{\tilde{R}}}_{\text{guidance on all observations}}$$

Notice that the probability only depends on observations $\mathbf{y}^{k:k+1}$ not $\mathbf{y}^{1:k+1}$, thus there is no background (past) information. This can be extended to any horizon $k:k+n$, and is meant to be used on the full window $0:T$ with observations at all timesteps. The algorithm isn't meant for online filtering, or simulation rollout.

**Joint AR** To include the background state into an auto-regressive scheme, Shysheya et al. (2024) combines the joint score with reconstruction guidance (Ho et al., 2022).

$$
\begin{aligned}
p(\mathbf{x}_t^{k:k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) &\propto p(\mathbf{x}_t^{k:k+1}) p(\mathbf{x}^k, \mathbf{y}^{k+1}|\mathbf{x}_t^{k:k+1}) \\
&\approx p(\mathbf{x}_t^{k:k+1}) \overbrace{p(\mathbf{x}^k|\mathbf{x}_t^{k:k+1}) p(\mathbf{y}^{k+1}|\mathbf{x}_t^{k:k+1})}^{\approx \text{independent}} \\
&\underset{\text{DPS}}{\approx} p(\mathbf{x}_t^{k:k+1}) \mathcal{N}(\mathbf{x}^k|\tilde{\mathbf{x}}_0^k, \tilde{\Gamma}) \mathcal{N}(\mathbf{y}^{k+1}|H\tilde{\mathbf{x}}_0^{k+1}, \tilde{R}) \\
\nabla_{\mathbf{x}_t^{k:k+1}} \log p(\mathbf{x}_t^{k:k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) &\approx \underbrace{s_\theta(\mathbf{x}_t^{k:k+1})}_{\text{joint score}} \underbrace{- \frac{1}{2} \nabla_{\mathbf{x}_t^{k:k+1}} \|\mathbf{x}^k - \tilde{\mathbf{x}}_0^k\|^2_{\tilde{\Gamma}}}_{\text{auto-regressive guidance}} \\
&\quad \underbrace{- \frac{1}{2} \nabla_{\mathbf{x}_t^{k:k+1}} \|\mathbf{y}^{k+1} - H\tilde{\mathbf{x}}_0^{k+1}\|^2_{\tilde{R}}}_{\text{guidance on observation } k+1} \qquad (18)
\end{aligned}
$$

With $\tilde{\Gamma} = \mu_t^2/\sigma_t^2 \Gamma + \zeta$ where $\zeta$ is a small value. Because the guidance signal needs to be strong enough to condition the generation, sampling needs to be stabilized with Langevin corrections.

**Conditional AR** Shysheya et al. (2024) and Savary et al. (2025) make use of an *amortized* score - i.e. conditioned by architecture. The input of the score network is a clean past state $\mathbf{x}^k$ and a noisy future state $\mathbf{x}_t^{k+1}$.

$$
\begin{aligned}
p(\mathbf{x}_t^{k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) &\propto p(\mathbf{x}_t^{k+1}|\mathbf{x}^k) p(\mathbf{y}^{k+1}|\mathbf{x}_t^{k+1}, \mathbf{x}^k) \qquad (19) \\
&\underset{\text{DPS}}{\approx} p(\mathbf{x}_t^{k+1}|\mathbf{x}^k) \mathcal{N}(\mathbf{y}^{k+1}|H\tilde{\mathbf{x}}_0^{k+1}, \tilde{R}) \qquad (20) \\
\nabla_{\mathbf{x}_t^{k+1}} \log p(\mathbf{x}_t^{k+1}|\mathbf{x}^k, \mathbf{y}^{k+1}) &\approx \underbrace{s_\theta(\mathbf{x}_t^{k+1}|\mathbf{x}^k)}_{\text{conditional score}} \underbrace{- \frac{1}{2} \nabla_{\mathbf{x}_t^{k+1}} \|\mathbf{y}^{k+1} - H\tilde{\mathbf{x}}_0^{k+1}(\mathbf{x}_t^{k+1}|\mathbf{x}^k)\|^2_{\tilde{R}}}_{\text{guidance on observation } k+1} \qquad (21)
\end{aligned}
$$

This is ideal for building an Autoregressive diffusion, where we fully trust the previously generated sample $\mathbf{x}^k$. Shysheya et al. (2024) report superior performance on forecasting tasks, compared to a model that is conditioned via guidance 18. Conditioning via architecture ensures stabler rollouts, there is no need to use Langevin corrections since the general direction is already well determined. However, it cannot easily deviate from its established course.

**Stochastic Interpolants** FlowDAS (Chen et al., 2025) uses a variant of conditional SDA for stochastic interpolants, where the model learns a path from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$. The follow-up **ForcingDAS** (Jia et al., 2026) combines FlowDAS to Diffusion Forcing (Chen et al., 2024) to handle arbitrary noise levels. The filter "re-noises" the past states, enabling the background to be progressively forgotten. However the noise schedule is based on a fixed heuristic and does not adapt to the model uncertainty.

### B.2 Ensemble of trajectories

In the past section, we have reviewed sampling algorithms for generating a single trajectory. For completeness, we now present the result of recent research on *ensemble* filters that operate with a distribution of trajectories. **FA-APF** (Savary et al., 2026) is a particle filter that uses the optimal proposal from a conditioned diffusion model (7). This is a practical algorithm that works out-of-the-box with a diffusion model that has already been trained on massive data (e.g. GenCast). However, this framework doesn't allow for much deviation from the past state, hence particles with incorrect background are eliminated. **SURGE** (Wei et al., 2026) is a particle filter variant that uses a stochastic interpolant to sample $p(\mathbf{x}^{k+1}|\mathbf{x}^k)$. The novelty is to use particle weighting and resampling at each step of the SDE integration which results in better observation guidance. These particle filter methods suffer from the high-dimensionality curse and localization issues. There is a low probability that a particle background will be correct in all dimensions, but because particle likelihoods are computed globally, particles that are only locally correct are eliminated.

### B.3 Numerical Solver hybrids

Another research direction is to assume that we have access to a numerical forward model of the dynamics at inference time. The following methods use a generative framework in conjonction with a simulator.

**Non-Deep Models** **EnSF** (Bao et al., 2025) is a Particle Filter that approximates the diffusion score using the numerical simulator instead of a neural netwok. This Monte Carlo approach is easier to implement but we believe that using a learned physical model is more efficient in terms of ensemble size.

**Latent Data Assimilation** Prior work has been conducted on hybridation of Assimilation algorithms with deep latent states. Notably using Variational Autoencoders to transpose the problem to a low-dimensional space with Gaussian distribution (Özalp et al., 2026). These reduced-order methods are less compute-intensive, however diffusion-based approaches have shown better generative performance. Thus, a solution is to combine latent-space encoding with score-based sampling Rozet et al.; Si & Chen (2025). In the same vein, **DAISI** (Andrae et al., 2026) projects the prediction of a simulator to a noisy latent space, then uses a guided denoiser to assimilate measurements. Adding noise to the latent reduces the trust in the background, but there is no principled method to determine the correct level.

## C Limitations of the joint formulation

### C.1 Background prior

In Bayesian Filtering, there is a fundemental difference between the observation and the background information: the measurements provide a likelihood while the past acts a prior. This is because the model of the dynamics is usually given as a conditional probability $p(\mathbf{x}^{k+1}|\mathbf{x}^k)$. This is seen in the classical 4DVar decomposition:

$$p(\mathbf{x}^k, \mathbf{x}^{k+1}|\mathbf{y}^{1:k+1}) = \underbrace{p(\mathbf{y}^{k+1}|\mathbf{x}^{k+1})}_{\text{obs. likelihood}} \underbrace{p(\mathbf{x}^{k+1}|\mathbf{x}^k)}_{\text{conditional model}} \underbrace{\mathcal{N}(\mathbf{x}^k|\hat{\mathbf{x}}^k, \mathbf{P}^k)}_{\text{background prior}} \tag{22}$$

In our sampling algorithm, we choose to see the background as a likelihood.

$$p(\mathbf{x}^k, \mathbf{x}^{k+1}|\mathbf{y}^{1:k+1}) \approx \underbrace{p(\mathbf{y}^{k+1}|\mathbf{x}^{k+1})}_{\text{obs. likelihood}} \underbrace{p(\mathbf{x}^k, \mathbf{x}^{k+1})}_{\text{joint model}} \underbrace{\mathcal{N}(\mathbf{x}^k|\hat{\mathbf{x}}^k, \mathbf{P}^k)}_{\text{background likelihood}} \tag{23}$$

We have chosen this approximation because it is straight-forward for a diffusion model to learn the joint noised score $\nabla \log p(\mathbf{x}_t^k, \mathbf{x}_t^{k+1})$. On the other hand, learning the score $\nabla \log p(\mathbf{x}_t^{k+1}|\mathbf{x}_t^k)$ with a noisy conditional $\mathbf{x}_t^k$ is non-standard (see the next section). Using a conditional score with clean past $\nabla \log p(\mathbf{x}_t^{k+1}|\mathbf{x}^k)$ is also straight-forward but doesn't offer the same flexibility.

Because of this choise, our proposed sampling algorithm has a bias, even in the linear Gaussian setting. It uses the model $p(\mathbf{x}^k, \mathbf{x}^{k+1})$ to generate $\mathbf{x}^k$ rather than sampling from the background Gaussian prior $\mathcal{N}(\mathbf{x}^k|\hat{\mathbf{x}}^k, \mathbf{P}^k)$. As a result, it will be biased towards the static (i.e. climatological) distribution rather than trusting the background trajectory. However, because the Gaussian background assumption is physically inacurrate, being biased towards the learned model could instead be beneficial to our algorithm.

One hypothetical drawback is for trajectories that fall in the tails of the training distribution. In this situation, we believe that the state will eventually return to a domain that it has seen more frequently in training. For completeness, we describe an alternative method that avoids the prior-likelihood swap. The next section describes a more expensive and arguably less-biased version of our method that could be explored in future work.

### C.2 Noisy Conditional score

In an attempt to apply 4DVar to a diffusion model, we have proposed to use the joint formulation (23). However, it is more natural to try and generalize Equation (22) to the noisy setting. An

alternative to the *joint* denoising of Equation (9) is to perform a *conditional* denoising $\mathbf{x}_t^{k+1}|\mathbf{x}_t^k$.

$$p(\mathbf{x}_t^{k:k+1}|\mathbf{y}^{1:k+1}) \approx \underbrace{p(\mathbf{y}^{k+1}|\tilde{\mathbf{x}}_0^{k+1})}_{\text{obs. likelihood}} \underbrace{p(\mathbf{x}_t^{k+1}|\mathbf{x}_t^k)}_{\text{conditional model}} \underbrace{\mathcal{N}(\tilde{\mathbf{x}}_0^k|\hat{\mathbf{x}}^k, \mathbf{P}^k)}_{\text{background prior}} \tag{24}$$

In the sampling algorithm, we will need the score $\nabla \log p(\mathbf{x}_t^{k+1}|\mathbf{x}_t^k)$, where the gradient is taken with respect to $\mathbf{x}_t^k$ and $\mathbf{x}_t^{k+1}$. Fortunately, the conditional score can be expressed as a sum of two denoising-diffusion score networks:

$$\nabla_{\mathbf{x}_t^k} \log p(\mathbf{x}_t^{k+1}|\mathbf{x}_t^k) = \underbrace{\nabla_{\mathbf{x}_t^k} \log p(\mathbf{x}_t^k, \mathbf{x}_t^{k+1})}_{\text{left part of joint score}} - \underbrace{\nabla_{\mathbf{x}_t^k} \log p(\mathbf{x}_t^k)}_{\text{marginal score}} \tag{25}$$

$$\nabla_{\mathbf{x}_t^{k+1}} \log p(\mathbf{x}_t^{k+1}|\mathbf{x}_t^k) = \underbrace{\nabla_{\mathbf{x}_t^{k+1}} \log p(\mathbf{x}_t^k, \mathbf{x}_t^{k+1})}_{\text{right part of joint score}} \tag{26}$$

The marginal score $\nabla_{\mathbf{x}_t^k} \log p(\mathbf{x}_t^k)$ can be approximated by a different neural-network, using the classical denoising-score matching objective (14). Essentially, the marginal score network erases the climatological prior on $\mathbf{x}^k$, so it can be replaced by the background prior.

This opens the way for a generalization of the current method, although we have not verified how it performs in practice. The marginal score requires a second neural network in both training and inference which creates a computation overhead.

# D Experiment details

## D.1 Kuramoto-Sivashinsky

The dataset consists in synthetic 1D fluid data generated with the following PDE:

$$\frac{\partial u}{\partial k} + u\frac{\partial u}{\partial z} + \frac{\partial^2 u}{\partial z^2} + \frac{\partial^4 u}{\partial z^4} = 0 \tag{27}$$

The spatial axis is decomposed into 256 cells $\{i\Delta z\}_{i=0}^{L}$ of length $\Delta z = 1/4$, the total time is 120 seconds which is split into 640 sub-timesteps $\Delta k$. The joint score network is trained on temporal windows of width $9\Delta k$. For conditional scores and inference, we decompose the window into two timesteps: conditioning $k = \{0, 1, 2, 3\}$ and prediction $k + 1 = \{4, 5, 6, 7, 8\}$.

The Neural Networks are 1D UNet with 12 hidden layers, we use the base implementation of the "SDA" architecture from Shysheya et al. (2024) which we train for 1000 epochs of 4096 windows on an Nvidia T4 GPU. In their implementation, the temporal steps are simply represented as channels of the discretized field. The conditional model uses the same architecture as the Joint and learns to decode both past and future, but is fed the past state as an extra input.

The observations are sampled from $\mathbf{y} = \mathbf{H}\mathbf{u} + \boldsymbol{\rho}$ where $\mathbf{H}$ is a boolean mask and $\boldsymbol{\rho} \sim \mathcal{N}(\mathbf{0}_Y, r\mathbf{I}_Y)$ is i.i.d noise. We experiment with three observation shapes:

$$\forall i, j \quad \mathbf{H}[i, j] \sim \mathcal{B}(p) \qquad \text{Sparse Bernoulli} \tag{28}$$

$$\forall j, \Delta j \quad p(\mathbf{H}[:, j + \Delta j] \mid \mathbf{H}[:, j]) = \mathcal{E}(\lambda \Delta j) \qquad \text{Exponential waiting time} \tag{29}$$

$$\forall i, j \quad \mathbf{H}[i, j] = 1 \text{ if } i\Delta z \approx L/2 + L\sin(\omega j) \qquad \text{Oscillating observer} \tag{30}$$

The Bernoulli probability is set to $p = 0.1$, the waiting time is $\lambda = 10$ and the frequency is $\omega = 1/64$.

## D.2 Kolmogorov Flow

The experiment uses a Navier-Stokes simulation on a periodic 2D domain. The velocity field $\mathbf{u}$ is described by the following system:

$$\partial_k \mathbf{u} + \mathbf{u} \cdot \Delta \mathbf{u} - \frac{1}{\mathrm{Re}}\Delta^2 \mathbf{u} + \frac{1}{\rho}\Delta p = \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = 0$$

Where $p$ is the pressure field, $\mathrm{Re} = 1000$ is the Reynolds number, $\rho = 1$ is the density and $\mathbf{f}$ is a forcing term. We refer to the paper from Rozet & Louppe (2023) for details on the generation process. The discrete velocity field is of size (2x64x64), the total duration of the trajectories is 64 timesteps.

We train the models on windows of length 5 with $k = [\![1, 2]\!]$ and $k + 1 = [\![3, 4, 5]\!]$. The Neural Network is a 2D UNet with 24 hidden layers, we use the base implementation of the "SDA" architecture from Shysheya et al. (2024) which we train for 1000 epochs of 4096 windows on an Nvidia A100 GPU.

### D.3 Road Traffic

The ARZ equations are a classic macroscopic model of motorway traffic on a one-dimensional road $z \in [0, L]$. The PDE system describes the evolution of vehicle density $\rho(z, k)$ (vehicles/meter) and average velocity $v(z, k)$. The first equation imposes the conservation of vehicles, the second equation couples the density $\rho$ with the velocity $v$ using a traffic pressure $h(\rho)$.

$$\begin{cases} \partial_k \rho + \partial_z(v\rho) = 0 \\ \partial_k(v + h(\rho)) + v\, \partial_z(v + h(\rho)) = \frac{1}{T_{\text{relax}}}(v - V_{\text{eq}}(\rho)) \end{cases} \tag{31}$$

We include 12 equally-spaced ramps to the simulated ring road. The ramps allow a fraction ($0.8 \leq \beta_r \leq 1.2$) of the traffic flow to enter or exit the road, this acts as an unpredictable exterior perturbation. We use the pressure term $h(\rho)$ and non-concave flux function $\rho V_{\text{eq}}(\rho)$ from David Ketcheson's codebase (https://github.com/ketch/aw-rascle-zhang) which we find to generate more stop-and-go waves than a classical Greenshields flux in our experiments.

The simulation domain is divided in a grid with time discretization $\Delta\kappa = 0.1$s and cell length $\Delta z = 50$m. We use the setup described in Moreau et al. (2026), the numerical solver is a finite-volume Rusanov scheme. The intial state and ramp entry-exit ratios are generated using a random combination of step functions. The data is then downscaled to $\Delta k = 10$s and $\Delta z = 100$m and the velocity is normalized to $[-1, 1]$ for learning and inference. The discretized state of the velocity field is of shape $(200)$ and the density field is not seen by the model.

We generate 600 trajectories of traffic flow as training data, 200 for validation and 200 for testing. We use a window of 12 time steps, $k = [\![1, 6]\!]$ for conditioning and $k + 1 = [\![7, 12]\!]$ for prediction. Although the observation pattern is not strictly linear and uncorrelated, the guidance is approximated as i. i. d. Gaussian taking the empirical observation RMSE at around $0.05$. The Neural Network is the same architecture as the KS dataset and we train for 1000 epochs of 4096 windows on an Nvidia A100 GPU. Because of the extended compute time, we only evaluate the models on 10 full simulations from the test dataset.

### D.4 Metrics

We now present the metrics used in this study. We denote the predicted state $\mathbf{x}$ and ground truth $\mathbf{x}^\star$, both vectors contain space and time dimensions.

**Skill** This is a direct measure of the prediction performance, it consists in the RMSE of the mean of the ensemble.

$$\text{Skill} = \sqrt{\text{Mean}\left((\mathbf{x}^\star - \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^{(i)})^2\right)} \tag{32}$$

Where the Mean is taken with respect to the spatial and temporal dimension to obtain a scalar average.

**Spread** This indicator of the sample variety is obtained by taking the standard deviation of the ensemble members, which is then averaged over the temporal and state dimension.

$$\text{Spread} = \sqrt{\text{Mean}\left(\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}^{(j)})^2\right)} \tag{33}$$

We use the biased standard deviation in our calculations.

**CRPS** The Continuous Ranked Probability Score measures the error between the distribution derived from the prediction ensemble $\mathbf{x}^{(i)}, 1 \leq i \leq N$ and the true trajectory $\mathbf{x}$. The result is then averaged over the temporal and state dimension.

$$\text{CRPS} = \text{Mean}\left(\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}^{(i)} - \mathbf{x}^{\star}\right\|_1 - \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\mathbf{x}^{k,(i)} - \mathbf{x}^{k,(j)}\right\|_1\right) \tag{34}$$

### D.5 Implementation details

**Sampling** We use the DPM solver (Lu et al., 2022) with 128 steps for all methods. We use one Langevin step (11) for every method except when using MMPS, the stepsize is chosen as $\tau = 0.25$, the guidance strength for SDA is $\mathbf{\Gamma} = 0.1\mathbf{I}$. The *volatile* models (small R) use a modified observation guidance $\mathbf{R} = \sigma_{\text{min}}^2\mathbf{I}$ with $\sigma_{\text{min}} = 0.01$. In Figure 1, the model with (large P) has a modified reconstruction guidance $\mathbf{P} = \sigma_{\text{large}}^2\mathbf{I}$ with $\sigma_{\text{large}} = 0.5$.

**Covariance localization** For all experiments, we use a spatial bandwidth of $3\Delta z$. The temporal bandwidth is chosen as $1\Delta k$ for Kolmogorov, $5\Delta k$ for ARZ and $10\Delta k$ for KS.

**Joint PF** When computing likelihood weights in the Joint Particle Filter, we can't easily estimate the expected state $\mathbb{E}[\mathbf{x}^{k+1}|\mathbf{x}^k]$ like in the original Conditional version. Thus we sample a prediction $\mathbf{x}^{k+1} \sim p(\mathbf{x}^{k+1}|\mathbf{x}^k)$ for each particle and compute the likelihood of observations $\mathbf{y}^{k+1}$ given the generated $\mathbf{x}^{k+1}$.

## E Compute times

Table 5 reports the wall-clock time to assimilate a single trajectory for each method, averaged over 10 runs.

Table 5: Assimilation time per single trajectory (h:mm). The KS and Kolmogorov experiments are run on an Nvidia P4 GPU and the ARZ experiment on an Nvidia T4 GPU. All methods use 10 particles, except the ARZ particle filter which uses 128.

| | KS | | Kolmogorov | ARZ |
|---|---|---|---|---|
| | P4 | T4 | P4 | T4 |
| Method | | | | |
| Cond. AR | 0:34 | – | 0:35 | – |
| Joint AR | 0:56 | – | 1:23 | 0:15 |
| Joint PF | – | 0:54 | – | – |
| Joint PF (N=128) | – | – | – | 2:48 |
| Cond. PF | 0:43 | – | 0:48 | – |
| EnJoi | 1:02 | – | 1:32 | 0:17 |

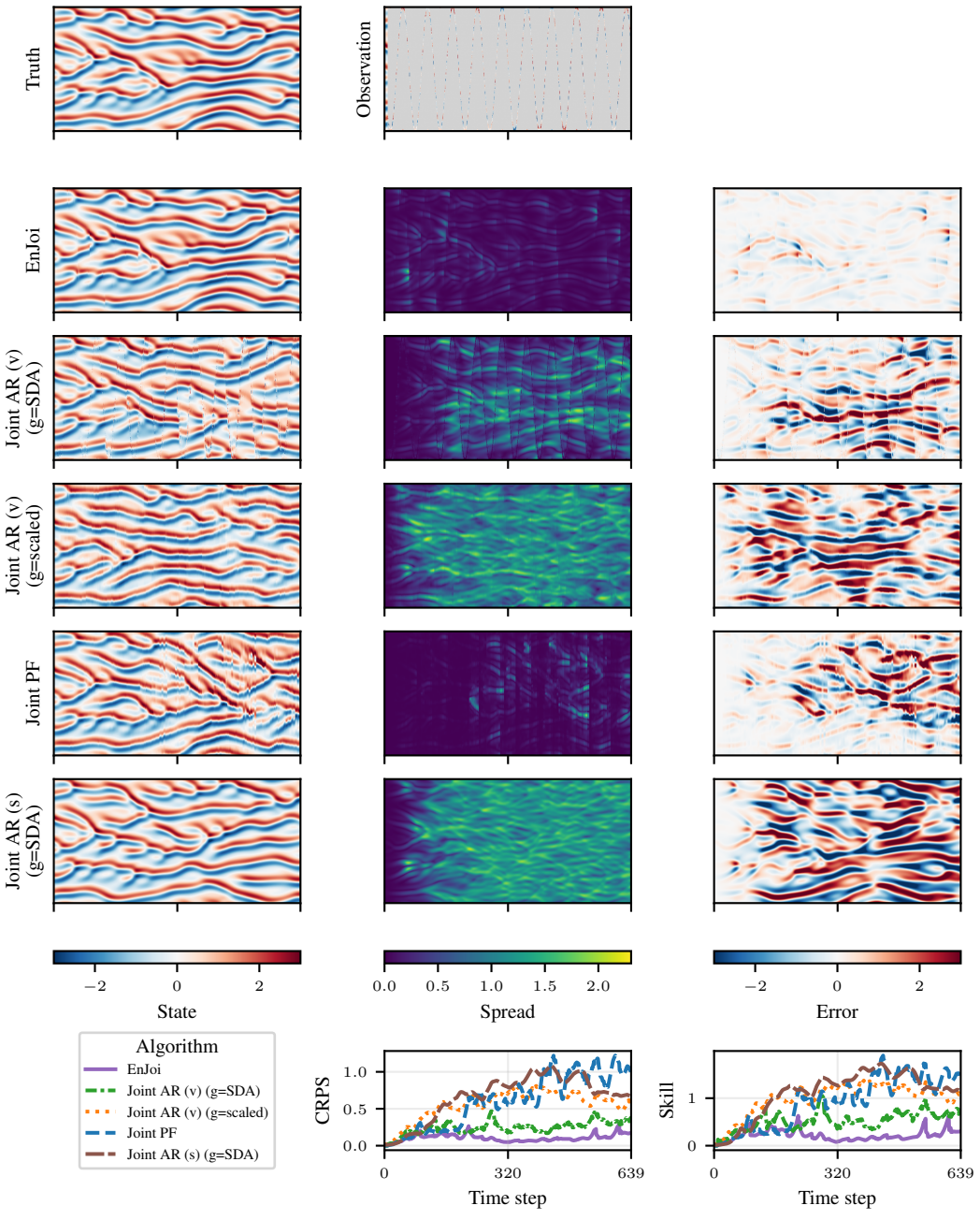


Figure 5: KS assimilation experiment with sine pattern observation. The left column displays one sample from the ensemble filters.

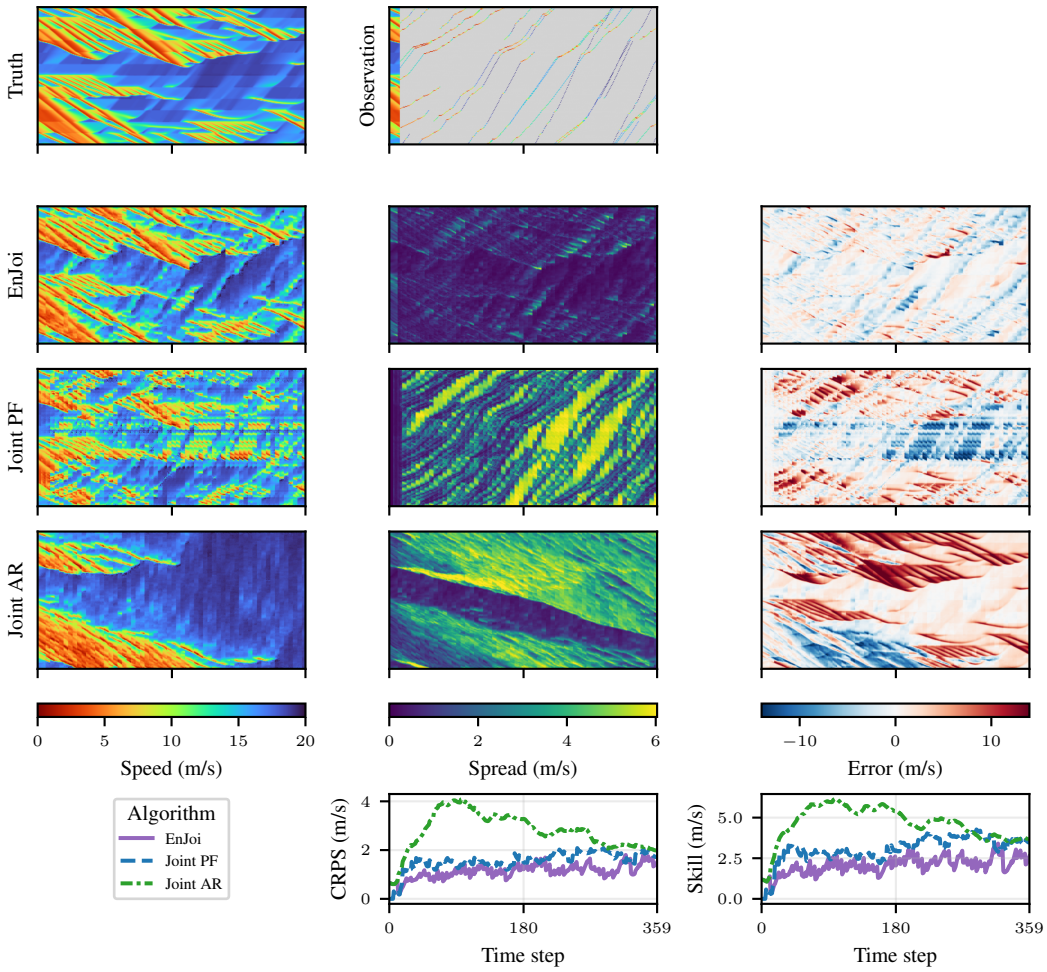


Figure 6: ARZ assimilation experiment. The left column displays one sample from the ensemble filters.