Computer Methods in Applied Mechanics and Engineering 432 (2024) 117447

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

An ensemble score filter for tracking high-dimensional nonlinear dynamical systems

Feng Bao [a], Zezhong Zhang [b], Guannan Zhang [b,*]

[a] *Department of Mathematics, Florida State University, 1017 Academic Way, Tallahassee, 32306, FL, USA*
[b] *Computer Science and Mathematics Division, Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, 37831, TN, USA*

ARTICLE INFO

Keywords:
Stochastic differential equations
Score-based diffusion models
Data assimilation
Curse of dimensionality
Nonlinear filtering

ABSTRACT

We propose an ensemble score filter (EnSF) for solving high-dimensional nonlinear filtering problems with superior accuracy. A major drawback of existing filtering methods, e.g., particle filters or ensemble Kalman filters, is the low accuracy in handling high-dimensional and highly nonlinear problems. EnSF addresses this challenge by exploiting the score-based diffusion model, defined in a pseudo-temporal domain, to characterize the evolution of the filtering density. EnSF stores the information of the recursively updated filtering density function in the score function, instead of storing the information in a set of finite Monte Carlo samples (used in particle filters and ensemble Kalman filters). Unlike existing diffusion models that train neural networks to approximate the score function, we develop a training-free score estimation method that uses a mini-batch-based Monte Carlo estimator to directly approximate the score function at any pseudo-spatial–temporal location, which provides sufficient accuracy in solving high-dimensional nonlinear problems while also saving a tremendous amount of time spent on training neural networks. High-dimensional Lorenz-96 systems are used to demonstrate the performance of our method. EnSF provides superior performance, compared with the state-of-the-art Local Ensemble Transform Kalman Filter, in reliably and efficiently tracking extremely high-dimensional Lorenz systems (up to 1,000,000 dimensions) with highly nonlinear observation processes.

1. Introduction

Tracking high-dimensional nonlinear dynamical systems, also known as nonlinear filtering, is a significant research area in data assimilation, with applications in weather forecasting, material science, biology, and finance [1–5]. The goal of addressing a filtering problem is to exploit noisy observational data streams to estimate the unobservable state of a stochastic dynamical system of interest. In linear filtering, where both the state and observation dynamics are linear, the Kalman filter provides an optimal estimate for the unobservable state, attainable analytically under the Gaussian assumption. However, maintaining the covariance matrix of a Kalman filter is computationally infeasible for high-dimensional systems. For this reason, ensemble Kalman filters (EnKF) were developed in [6–8] to represent the distribution of the system state using a collection of state samples, called an ensemble, and to replace the covariance matrix in the Kalman filter with the sample covariance computed from the ensemble. EnKF methods, especially the Local Ensemble Transform Kalman Filter (LETKF) [9,10], are deployed operationally [11,12] widely used to integrate observations for the purpose of understanding complex processes such as atmospheric convection [13,14]. Despite many successful applications, EnKFs suffer from fundamental limitations as they make Gaussian assumptions in their update step, which can lead to severe model

* Corresponding author.
E-mail addresses: bao@math.fsu.edu (F. Bao), zhangz2@ornl.gov (Z. Zhang), zhangg@ornl.gov (G. Zhang).

https://doi.org/10.1016/j.cma.2024.117447

Available online 22 October 2024

bias when solving highly nonlinear systems. Hyperparameters like localization and inflation have been added to EnKF to handle high-dimensional nonlinear problems. However, the accuracy of EnKF is highly sensitive to the hyperparameters, as demonstrated in Section 4, such that fine-tuning needs to be re-conducted whenever there is a small change, e.g., observation noise level, to the target filtering problem. Moreover, even though EnKF (e.g., LETKF [9,10]) is known to have good scalability for CPU-based parallel computing platforms [10], the existing EnKF algorithms are not suitable for modern GPU-based supercomputers because the parallelization of EnKF resulting from localization, i.e., decomposing the large covariance matrix into a large number of very small covariance matrices, cannot fully exploit the computing power of GPUs.

In addition to the EnKF, several effective methods have been developed to tackle nonlinearity in data assimilation. These methods include the particle filter [15–24], the Zakai filter [25,26], and so on. For example, the particle filter employs a set of random samples, referred to as particles, to construct an empirical distribution to approximate the filtering density of the target state. Upon receiving observational data, the particle filter uses Bayesian inference to assign likelihood weights to the particles. A resampling process is iteratively performed, generating duplicates of particles with large weights and discarding particles with small weights. Although particle filters emerged around the same time as the EnKF, their implementation to large-scale models has been difficult due to the curse of dimensionality (weight collapse). This means that particle filters require prohibitively large ensemble sizes (number of particles) to ensure long-term stability. While there have been significant advances in this direction [27–29], the resulting particle filters often provide marginal advantages over the state-of-the-art EnKFs used in operations.

In this work, we introduce a novel ensemble score filter (EnSF) that exploits the score-based diffusion model [30–35], defined in a pseudo-temporal domain to characterize the evolution of the filtering density. The score-based diffusion model is a popular generative machine learning model for generating samples from a target distribution. Diffusion models have been applied to nonlinear filtering problems in our previous work [36]. Despite the promising performance, a major drawback of the method in [36] is that the neural network used to learn the score function needs to be re-trained at every filtering step after assimilating new observational data. Even though we can store the checkpoint of the neural network weights from previous filtering step, the neural network training still takes several minutes at each filtering step for a 100-dimensional Lorenz-96 model. Moreover, training the score function will require storing all the paths of the forward stochastic processes of the diffusion model, which leads to high storage requirements for high-dimensional problems. To resolve these issues, the key idea of EnSF is to completely avoid training neural networks to learn the score function. Instead, we derive the closed form of the score function and develop a training-free score estimation that uses mini-batch-based Monte Carlo estimators to directly approximate the score function at any pseudo-spatial–temporal location in the process of solving the reverse-time diffusion sampler. Numerical examples in Section 4 demonstrate that the training-free score estimation approach can provide sufficient accuracy in solving high-dimensional nonlinear problems as well as save tremendous amount of time spent on training neural networks. Another essential aspect of EnSF is its analytical update step, which gradually incorporates data information into the score function. This step is crucial in mitigating the degeneracy issue faced when dealing with very high-dimensional nonlinear filtering problems. The main contributions of this work are summarized as follows:

- We develop a training-free score function estimator that allows the diffusion model to be updated in real-time without re-training when new observational data is collected.
- We showcase the remarkable robustness of the performance of EnSF with respect to its hyperparameters by using a set of fine-tuned hyperparameters to dramatically different scenarios, e.g., different dimensions, different observation noise, etc.
- We showcase the superior performance of EnSF by comparing it with the state-of-the-art LETKF method in tracking 1,000,000-dimensional Lorenz-96 models with highly nonlinear observations.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the nonlinear filtering problem. In Section 3, we provide a comprehensive discussion to develop our EnSF method. In Section 4, we demonstrate the performance of the EnSF method in solving a Lorenz-96 tracking problem in high-dimensional space with highly nonlinear observation processes, and we shall also conduct a series of comparison studies between EnSF and the state-of-the-art LETKF method. Concluding remarks and future directions are given in Section 5.

2. Problem setting

Nonlinear filtering is a process used to estimate the states of a system that evolves in time, where the system dynamics and the measurements are influenced by nonlinearities and noise. This estimation problem is complex because nonlinearity distorts the relationship between the observed measurements and the actual states, making linear prediction and update methods ineffective. Specifically, we are interested in tracking the state of the following discretized stochastic dynamical system:

$$X_t = f(X_{t-1}, \omega_{t-1}), \tag{1}$$

where $t \in \mathbb{Z}^+$ is the discrete time index, $X_t \in \mathbb{R}^d$ is the state of the system governed by a physical model $f : \mathbb{R}^d \times \mathbb{R}^k \mapsto \mathbb{R}^d$, and $\omega_{t-1} \in \mathbb{R}^k$ is a random variable representing the uncertainty in f. This uncertainty may be caused by natural perturbations to the physical model, incomplete knowledge, or unknown features of the model f. The existence of such uncertainty makes direct estimation and prediction of the state of the dynamical model infeasible. To filter out the uncertainty and make accurate estimations of the state, we rely on partial noisy observations of the state X_t given as follows:

$$Y_t = g(X_t) + \varepsilon_t, \tag{2}$$

where $Y_t \in \mathbb{R}^r$ is the observational data collected by nonlinear observation function $g(X_t)$, and the observation is also perturbed by a Gaussian noise $\varepsilon_t \sim \mathcal{N}(0, \Sigma_t)$.

The goal of the filtering problem is to find the best estimate, denoted by $\hat{X}_t$, of the hidden state X_t, given the observation information $\mathcal{Y}_t := \sigma(Y_{1:t})$, which is the σ-algebra generated by the observational data up to the time instant t. In mathematics, such optimal estimate for X_t is usually defined by the optimal filter, i.e., the conditional expectation for X_t conditioning on the observational information: $\hat{X}_t := \mathbb{E}[X_t|\mathcal{Y}_t]$. The standard approach for solving the optimal filtering problem is the Bayesian filter, in which we aim to approximate the conditional probability density function (PDF) of the state, denoted by $p_{X_t|Y_{1:t}}(x_t|y_{1:t})$, which is referred to as the filtering density. The general idea of the Bayesian filter is to recursively incorporate observational data to describe the evolution of the filtering density from time $t-1$ to t in two steps, i.e., the prediction step and the update step.

In the *prediction step*, we utilize the Chapman–Kolmogorov formula to propagate the state variable from $t-1$ to t and obtain the *prior filtering density*

$$p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1}) = \int_{\mathbb{R}^d} p_{X_t|X_{t-1}}(x_t|x_{t-1}) p_{X_{t-1}|Y_{1:t-1}}(x_{t-1}|y_{1:t-1}) dx_{t-1}, \tag{3}$$

where $p_{X_{t-1}|Y_{1:t-1}}(x_{t-1}|y_{1:t-1})$ is the posterior filtering density obtained at the time instant $t-1$, $p_{X_t|X_{t-1}}(x_t|x_{t-1})$ is the transition probability density derived from the state dynamics in Eq. (1), and $p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1})$ is the prior filtering density for the time instant t.

In the *update step*, we combine the likelihood function with the prior filtering density to obtain the *posterior filtering density*, i.e.,

$$p_{X_t|Y_{1:t}}(x_t|y_{1:t}) \propto p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1}) \, p_{Y_t|X_t}(y_t|x_t), \tag{4}$$

where $p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1})$ is the prior filtering density in Eq. (3), and the likelihood function $p_{Y_t|X_t}(y_t|x_t)$ is defined by

$$p_{Y_t|X_t}(y_t|x_t) \propto \exp\left[-\frac{1}{2}\left(g(x_t)-y_t\right)^\top \Sigma_t^{-1}\left(g(x_t)-y_t\right)\right], \tag{5}$$

with Σ_t being the covariance matrix of the random noise ε_t in Eq. (2). In this way, the filtering density is predicted and updated through formulas Eq. (3) to Eq. (4) recursively in time. Note that both the prior and the posterior filtering densities in Eq. (3) and (4) are defined as the continuum level, which is not practical. Thus, one important research direction in nonlinear filtering is to study how to accurately approximate the prior and the posterior filtering densities.

In the next section, we introduce how to utilize score-based diffusion models to solve the nonlinear filtering problem. The diffusion model was introduced into nonlinear filtering in our previous work [36] in which the score function is approximated by training a deep neural network. Although the use of score functions provides accurate results, there are several drawbacks resulting from training neural networks. First, the neural network needs to be re-trained/updated at each filtering time step, which makes it computationally expensive. For example, it takes several minutes to train and update the neural-network-based score function at each filtering time step for solving a 100-D Lorenz-96 model [36]. Second, since neural network models are usually over-parameterized, a large number of samples are needed to form the training set to avoid over-fitting. Third, hyperparameter tuning and validation of the trained neural network introduces extra computational overhead. These challenges motivated us to develop the EnSF method that completely avoids neural network training in score estimation, in order to greatly expand the powerfulness of the score-based diffusion model in nonlinear filtering.

3. The ensemble score filter (EnSF) method

We now describe the details of the proposed EnSF method. Section 3.1 introduces how to use the score-based diffusion model to store the information in the prior filtering density in the prediction step of nonlinear filtering. Section 3.2 recalls how to analytically incorporate the likelihood information to transform the prior score to the posterior score at the update step of nonlinear filtering. Section 3.3 introduces the implementation details of EnSF and the discussion on its computational complexity.

3.1. The prediction step of EnSF

The goal of the prediction step in EnSF is to develop a score-based diffusion model as a stochastic transport map between the prior filtering density $p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1})$ in Eq. (3) and the standard normal distribution. To proceed, we first define a pseudo-temporal variable

$$\tau \in \mathcal{T} = [0, 1), \tag{6}$$

which is different from the temporal domain for defining the state and observation processes in Eqs. (1) and (2). At the tth filtering step, we can define the following forward SDE [37] in the pseudo-temporal domain $\mathcal{T}$:

$$dZ_{t,\tau} = b(\tau) Z_{t,\tau} d\tau + \sigma(\tau) dW_\tau, \tag{7}$$

where W_τ is a standard d-dimensional Brownian motion, $b(\tau)$ is the drift coefficient, $\sigma(\tau)$ is the diffusion coefficient, and the subscript $(\cdot)_t$ indicates that the SDE is defined for the tth filtering step. Note that Eq. (7) is a *linear* SDE. Thus its solution can be derived as

$$Z_{t,\tau} = \exp\left[\int_0^\tau b(s) ds\right] \left(Z_{t,0} + \int_0^\tau \exp\left[-\int_0^s b(r) dr\right] \sigma(s) dW_s\right). \tag{8}$$

The goal is to construct a forward SDE to transport any given initial random variable $Z_{t,0} \sim q_{Z_{t,0}}(z_{t,0})$ at $\tau = 0$ to the standard normal distribution $Z_{t,1} \sim \phi_{(0,\mathbf{I}_d)}(z_{t,1})$ at $\tau = 1$, where $\phi_{(0,\mathbf{I}_d)}(\cdot)$ denotes the PDF of the standard normal distribution and $Z_{t,1} \equiv \lim_{\tau\to 1} Z_{t,\tau}$. There are many choices of $b(\tau)$ and $\sigma(\tau)$ given in the literature [31,33,38] to achieve the task. In this work, we use the following contruction of the parameters for the forward SDE:

$$b(\tau) = \frac{\mathrm{d}\log\alpha_\tau}{\mathrm{d}\tau} \quad \text{and} \quad \sigma^2(\tau) = \frac{\mathrm{d}\beta_\tau^2}{\mathrm{d}\tau} - 2\frac{\mathrm{d}\log\alpha_\tau}{\mathrm{d}\tau}\beta_\tau^2, \tag{9}$$

where the two processes α_τ and β_τ are defined by

$$\alpha_\tau = 1-\tau, \quad \beta_\tau^2 = \tau \quad \text{for} \quad \tau \in \mathcal{T} = [0,1). \tag{10}$$

Substituting the definitions of $b(\tau)$ and $\sigma(\tau)$ into Eq. (8), we can obtain that the conditional probability density function $q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0})$ for any fixed value $z_{t,0}$ is the following Gaussian distribution:

$$q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0}) = \phi_{(\alpha_\tau z_{t,0},\beta_\tau^2\mathbf{I}_d)}(z_{t,\tau}|z_{t,0}), \tag{11}$$

where $\phi_{(\alpha_\tau z_{t,0},\beta_\tau^2\mathbf{I}_d)}(\cdot)$ is the standard normal PDF with mean $\alpha_\tau z_{t,0}$ and covariance matrix $\beta_\tau^2\mathbf{I}_d$. The above equation immediately leads to

$$q_{Z_{t,1}|Z_{t,0}}(z_{t,1}|z_{t,0}) = \lim_{\tau\to 1}\phi_{(\alpha_\tau z_{t,0},\beta_\tau^2\mathbf{I}_d)}(z_{t,\tau}|z_{t,0}) = \phi_{(0,\mathbf{I}_d)}(z_{t,1}), \tag{12}$$

meaning that the forward SDE in Eq. (7) can transport any initial distribution to the standard normal distribution at $\tau = 1$.

Let the initial state $Z_{t,0}$ of the forward SDE in Eq. (7) follow the prior filtering density $p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1})$ in Eq. (3), we have

$$Z_{t,0} := X_t|Y_{1:t-1} \implies q_{Z_{t,0}}(z_{t,0}) = p_{X_t|Y_{1:t-1}}(x_t|y_{1:t-1}), \tag{13}$$

such that the forward SDE can transport the prior filtering density to the standard normal distribution. However, what we need is the transport model in the opposite direction, i.e., from $\tau = 1$ to $\tau = 0$. To do this, we construct the corresponding reverse SDE [33]

$$dZ_{t,\tau} = \left[b(\tau)Z_{t,\tau} - \sigma^2(\tau)S_{t|t-1}(Z_{t,\tau},\tau)\right]d\tau + \sigma(\tau)d\overleftarrow{W}_\tau, \tag{14}$$

where $\overleftarrow{W}_\tau$ is a standard d-dimensional Brownian motion running backward in time [39], $b(\tau)$ and $\sigma(\tau)$ are the same as in the forward SDE. The notation $S_{t|t-1}(Z_{t,\tau},\tau)$ defines the score function associated with the diffusion model for the prior filtering density $p_{X_t|Y_{1:t-1}}$ in Eq. (3), i.e.,

$$S_{t|t-1}(z_{t,\tau},\tau) := \nabla_{z_{t,\tau}}\log q_{Z_{t,\tau}}(z_{t,\tau}), \tag{15}$$

where the subscript $(\cdot)_{t|t-1}$ indicates that it is the prior score function without assimilating the observational data at the tth filtering step.

The methodology of EnSF is established based on the following derivation. First, we observe that the probability density function $q_{Z_{t,\tau}}$ can be expressed as follows:

$$q_{Z_{t,\tau}}(z_{t,\tau}) = \int_{\mathbb{R}^d} q_{Z_{t,\tau},Z_{t,0}}(z_{t,\tau},z_{t,0})dz_{t,0} = \int_{\mathbb{R}^d} q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0})q_{Z_{t,0}}(z_{t,0})dz_{t,0}$$

Then, by substituting this equation into Eq. (15) and exploiting the fact in Eq. (11), we can rewrite the score function in the form of the following integral:

$$\begin{aligned} &S_{t|t-1}(z_{t,\tau},\tau) \\ &= \nabla_{z_{t,\tau}}\log\left(\int_{\mathbb{R}^d} q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0})q_{Z_{t,0}}(z_{t,0})dz_{t,0}\right) \\ &= \frac{1}{\int_{\mathbb{R}^d} q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z'_{t,0})q_{Z_{t,0}}(z'_{t,0})dz'_{t,0}}\int_{\mathbb{R}^d} -\frac{z_{t,\tau}-\alpha_\tau z_{t,0}}{\beta_\tau^2}q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0})q_{Z_{t,0}}(z_{t,0})dz_{t,0} \\ &= \int_{\mathbb{R}^d} -\frac{z_{t,\tau}-\alpha_\tau z_{t,0}}{\beta_\tau^2}w(z_{t,\tau},z_{t,0})q_{Z_{t,0}}(z_{t,0})dz_{t,0}, \end{aligned} \tag{16}$$

where the weight function $w(z_{t,\tau},z_{t,0})$ is defined by

$$w(z_{t,\tau},z_{t,0}) := \frac{q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z_{t,0})}{\int_{\mathbb{R}^d} q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|z'_{t,0})q_{Z_{t,0}}(z'_{t,0})dz'_{t,0}}, \tag{17}$$

satisfying that $\int_{\mathbb{R}^d} w(z_{t,\tau},z_{t,0})q_{Z_{t,0}}(z_{t,0})dz_{t,0} = 1$.

Thus, the reserse SDE and the score function fully characterize the prior filtering density in Eq. (3). When applying proper numerical schemes to approximate the score function $S_{t|t-1}(z_{t,\tau},\tau)$ in Eq. (16) and the reserse SDE in Eq. (14), we can generate an unlimited number of samples from the prior filtering density. The implementation detail of the prediction step is given in Section 3.3.

3.2. The update step of EnSF

The goal of the update step in EnSF is to incorporate the new observational data Y_t, or more specifically the likelihood function in Eq. (5), obtained at the tth filtering step to update the prior filtering density in a Bayesian fashion. In the context of diffusion models, this task becomes how to update the prior score function $S_{t|t-1}$ to the posterior score function, denoted by $S_{t|t}$, by incorporating the likelihood function. In this work, we use a similar strategy as our previous work in [36] to analytically add the likelihood information to the prior score function $S_{t|t-1}$ to obtain a posterior score function $S_{t|t}$.

Specifically, we assume the posterior score function has the following structure:

$$S_{t|t}(z_{t,\tau},\tau) := S_{t|t-1}(z_{t,\tau},\tau) + h(\tau)\nabla_{z_{t,\tau}} \log p_{Y_t|X_t}(y_t|z_{t,\tau}), \tag{18}$$

where $S_{t|t-1}(z_{t,\tau},\tau)$ is the prior score function in Eq. (16), and $\nabla_{z_{t,\tau}} \log p_{Y_t|X_t}(y_t|z_{t,\tau})$ is the gradient of the log likelihood function evaluated at $z_{t,\tau}$. According to practical usage of nonlinear filtering in numerical weather forecasting (e.g., [40]), the analytical formula of the observation operator, i.e., the function g in Eq. (2) is usually known, such that it is reasonable to assume the gradient of the log-likelihood is accessible in EnSF. The key component in the update step is the damping function $h(\tau)$ satisfying

$$h(\tau) \text{ is monotonically decreasing in } [0,1] \text{ with } h(1)=0 \text{ and } h(0)=1, \tag{19}$$

which determines how the likelihood information is gradually introduced into the score function while solving the reserse SDE. In this work, we use $h(\tau) = 1-\tau$ in the numerical experiments. The likelihood has almost no influence on the prior score when the pseudo time τ is close to 1. As τ decreases, the diffusion term becomes less dominating and the likelihood information is gradually injected into the reserse SDE via the drift term.

We emphasize that even though the proposed structure of the posterior score works very well for the numerical examples in Section 4, there may be a model structure error in the proposed posterior score function in Eq. (18) depending on the choice of $h(\tau)$. In other words, the proposed $S_{t|t}$ in Eq. (18) may not be the score associated with the exact posterior filtering density in Eq. (4). Correcting the model error will ensure the theoretical rigor but may significantly increase the computational cost, especially for nonlinear filtering in which the score function needs to be updated dynamically in real time. Thus, how to develop a computationally efficient model error correction scheme is still an open question and will be considered in our future work.

Remark (*Avoiding the Curse of Dimensionality*)**.** Incorporating the analytical form of the likelihood information, i.e., $\nabla_{z_{t,\tau}} \log p_{Y_t|X_t}(y_t|z_{t,\tau})$, into the score function plays a critical role in avoiding performing high-dimensional approximation, i.e., the curse of dimensionality, in the update step. In other words, when $\nabla_{z_{t,\tau}} \log p_{Y_t|X_t}(y_t|z_{t,\tau})$ is given, either in the analytical form or via automatic differentiation, we do not need to perform any approximation in $\mathbb{R}^d$. In comparison, EnKF requires approximating the covariance matrix and the particle filter requires construction of empirical distributions, both of which involve approximation of the posterior distribution in $\mathbb{R}^d$.

3.3. Implementation of EnSF

We focus on how to discretize EnSF and approximate the score functions $S_{t|t}$ and $S_{t+1|t}$ in order to establish a practical implementation for EnSF. The classic diffusion model methods [31,33,38] train neural networks to learn the score functions. This approach works well for static problems that does not require fast evolution of the score function. However, this strategy becomes inefficient in solving the nonlinear filtering problem [36], especially for extremely high-dimensional problems. To address this challenge, we propose a training-free score estimation approach that uses the Monte Carlo method to directly approximate the expression of the score function in Eq. (16), which enables extremely efficient implementation of EnSF.

3.3.1. Introducing two hyperparameters into EnSF

The advantage of the choice of the drift and diffusion coefficients in Eqs. (9) and (10) is that the resulting forward SDE can map any distribution to the standard normal distribution within a bounded pseudo time interval $\mathcal{T} = [0,1)$. However, this approach also introduces several computational issues into the diffusion model. The first one is that the denominator in Eq. (16) goes to zero as $\tau \to 0$, which will cause the explosion of the score function when $\beta_\tau^2 = \tau$ at $\tau = 0$; the second one is that when $\alpha_\tau = 1-\tau$ and the reserse SDE is solved exactly, the conditional distribution in Eq. (11) indicates that the reserse SDE will drive each path of the state $Z_{t,\tau}$ of the diffusion model to the infinitesimal neighborhood of one of the samples of $Z_{t,0}$ as $\tau \to 0$, which will limit the exploration power of the diffusion model. To regularize the reserse SDE and extend the pseudo-temporal domain to $[0,1]$, we introduce two hyperparameters, denoted by ϵ_α and ϵ_β, to the definitions of α_τ and β_τ in Eq. (10), respectively. After re-parameterization, the actual α_τ and β_τ used in our EnSF implementation are

$$\bar{\alpha}_\tau = 1 - \tau(1-\epsilon_\alpha); \qquad \bar{\beta}_\tau^2 = \epsilon_\beta + \tau(1-\epsilon_\beta). \tag{20}$$

Based on the above equation, $\bar{\alpha}_\tau$ is a linear interpolation between $(0,1)$ and $(1,\epsilon_\alpha)$, and $\bar{\beta}_\tau^2$ is a linear interpolation between $(0,\epsilon_\beta)$ and $(1,1)$. The fine-tuning procedure shown in Section 4.2 indicates that even though the performance of EnSF is not sensitive to the two hyperparameters compared to LETKF, the fine-tuning can still provide significant performance improvement.

3.3.2. Training-free score estimation

Unlike existing methods that use neural network models to learn the score function, in this work we propose to directly discretize the score representation in Eq. (16). Specifically, we assume that we are given a set of samples $\{x_{t-1,j}\}_{j=1}^J$ drawn from the posterior filtering density function $p_{X_{t-1,\tau}|Y_{1:t-1}}(x_{t-1,\tau}|y_{1:t-1})$ from previous filtering time step $t-1$. For any fixed pseudo-time instant $\tau \in \mathcal{T}$ and $z_{t,\tau} \in \mathbb{R}^d$, the integral in Eq. (16) can be estimated by

$$S_{t|t-1}(z_{t,\tau},\tau) \approx \bar{S}_{t|t-1}(z_{t,\tau},\tau) := \sum_{n=1}^{N} -\frac{z_{t,\tau} - \bar{\alpha}_\tau f(x_{t-1,j_n},\omega_{t-1,j_n})}{\bar{\beta}_\tau^2}\bar{w}(z_{t,\tau}, f(x_{t-1,j_n},\omega_{t-1,j_n})), \tag{21}$$

using a mini-batch $\{x_{t-1,j_n}\}_{n=1}^N$ with batch size $N \leq J$, where $f(\cdot,\cdot)$ is the state equation in Eq. (1). The weight $w(z_{t,\tau}, f(x_{t-1,j_n}, \omega_{t-1,j_n}))$ in Eq. (17) is approximated by

$$\bar{w}(z_{t,\tau}, f(x_{t-1,j_n}, \omega_{t-1,j_n})) := \frac{q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|f(x_{t-1,j_n}, \omega_{t-1,j_n}))}{\sum_{m=1}^N q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|f(x_{t-1,j_m}, \omega_{t-1,j_m}))}, \tag{22}$$

which means $w(z_{t,\tau}, f(x_{t-1,j_n}, \omega_{t-1,j_n}))$ can be estimated by the normalized probability density values $\{q_{Z_{t,\tau}|Z_{t,0}}(z_{t,\tau}|f(x_{t-1,j_n}, \omega_{t-1,j_n}))\}_{n=1}^N$. In practice, the mini-batch $\{x_{t-1,j_n}\}_{n=1}^N$ could be a very small subset of $\{x_{t-1,j}\}_{j=1}^J$ to ensure sufficient accuracy in solving the filtering problems. In fact, we observe that the weighting function $\bar{w}(\cdot)$ in Eq. (22) tends to assign nearly all the weight to a single term in the summation of Eq. (21). In such cases, the approximated score function is primarily determined by that one sample, making additional score computations redundant. As a result, we use batch size one for our mini-batch in the numerical experiments in Section 4, which provides satisfactory performance in terms of accuracy and efficiency. The training-free score estimation is significantly more efficient than training neural networks to learn the score function [36] in nonlinear filtering where the posterior filtering density needs to be updated frequently.

3.3.3. *Summary of EnSF workflow*

Now we combine the aforementioned approximation schemes to develop a detailed algorithm to evolve the filtering density from $p_{X_{t-1}|Y_{1:t-1}}(x_{t-1}|y_{1:t-1})$ to $p_{X_t|Y_{1:t}}(x_t|y_{1:t})$. At the tth filtering step, we assume we are given a set of samples $\{x_{t-1,j}\}_{j=1}^J$ drawn from the posterior filtering density function $p_{X_{t-1}|Y_{1:t-1}}(x_{t-1}|y_{1:t-1})$ and the goal is to generate a set of samples $\{x_{t,j}\}_{j=1}^J$ from $p_{X_t|Y_{1:t}}(x_t|y_{1:t})$. This evolution involves the simulation of the reserse SDE of the diffusion model driven by the approximate score $S_{t|t}$. Even though the forward SDE is included in the diffusion model, the training-free score estimation approach allows us to skip the simulation of the forward SDE..

We use the Euler–Maruyama scheme to discretize the reserse SDE. Specifically, we introduce a partition of the pseudo-temporal domain $\mathcal{T} = [0,1]$, i.e.,

$$\mathcal{D}_K := \{\tau_k \mid 0 = \tau_0 < \tau_1 < \cdots < \tau_k < \tau_{k+1} < \cdots < \tau_K = 1\}$$

with uniform step-size $\Delta\tau = 1/K$. We first draw a set of samples $\{z_{t,1}^j\}_{j=1}^J$ from the standard normal distribution. For each sample $z_{t,1}^j$, we obtain the approximate solution $z_{t,0}^j$ by recursively evaluating the following scheme

$$z_{t,\tau_k}^j = z_{t,\tau_{k+1}}^j - \left[b(\tau_{k+1})z_{t,\tau_{k+1}}^j - \sigma^2(\tau_{k+1})\bar{S}_{t|t}(z_{t,\tau_{k+1}}^j, \tau_{k+1})\right]\Delta\tau + \sigma(\tau_{k+1})\Delta W_{\tau_{k+1}}^j, \tag{23}$$

for $k = K-1, K-2, \dots, 1, 0$, where $\Delta W_{\tau_{k+1}}^j$ is a realization of the Brownian increment, and the approximate score function is calculated by

$$\bar{S}_{t|t}(z_{t,\tau_{k+1}}^j, \tau_{k+1}) = \bar{S}_{t|t-1}(z_{t,\tau_{k+1}}^j, \tau_{k+1}) + h(\tau_{k+1})\nabla_{z_{t,\tau_{k+1}}^j} \log p_{Y_t|X_t}(y_t|z_{t,\tau_{k+1}}^j). \tag{24}$$

Since the reserse SDE is driven by $\bar{S}_{t|t}$, we treat $\{z_{t,0}^j\}_{j=1}^J$ as the desired sample set $\{x_{t,j}\}_{j=1}^J$ from the posterior filtering density $p_{X_t|Y_{1:t}}(x_t|y_{1:t})$. The EnSF workflow is summarized as a pseudo-algorithm in Algorithm 1.

Algorithm 1: the pseudo-algorithm for EnSF

```
1: **Input**: the state equation $f(X_t, \omega_t)$, the initial distribution $p_{X_0}(x_0)$;
2: Generate $J$ samples $\{x_{0,j}\}_{j=1}^J$ from the initial distribution $p_{X_0}(x_0)$;
3: for $t = 1, \dots,$
4:     Run the state equation in Eq. (1) to get predictions $\{f(x_{t-1,j}, \omega_{t-1,j})\}_{j=1}^J$;
5:     for $k = K-1, \dots, 0$
6:         Compute the weight $\{\{\bar{w}(z_{t,\tau}^j, f(x_{t-1,j_n}, \omega_{t-1,j_n}))\}_{n=1}^N\}_{j=1}^J$ using Eq. (22);
7:         Compute $\{\bar{S}_{t|t-1}(z_{t,\tau_{k+1}}^j, \tau_{k+1})\}_{j=1}^J$ using Eq. (21);
8:         Compute $\{\bar{S}_{t|t}(z_{t,\tau_{k+1}}^j, \tau_{k+1})\}_{j=1}^J$ using Eq. (24);
9:         Compute $\{z_{t,\tau_k}^j\}_{j=1}^J$ using Eq. (23);
11:     end
10:     Let $\{x_{t,j}\}_{j=1}^J = \{z_{t,0}^j\}_{j=1}^J$;
11: end
```

3.3.4. *Discussion on the computational complexity of EnSF*

Since the cost of running the state equation $f(X_t, \omega_t)$ in Eq. (1) is problem-dependent, we only discuss the cost of the matrix operations for Line 6 – 9 in Algorithm 1. In terms of the storage cost, the major storage of EnSF is used to store the two sample sets, i.e., $\{x_{t,j}\}_{j=1}^J$ from the posterior filtering density of the previous time step and $\{z_{\tau,j}\}_{j=1}^J$ for the states of the diffusion model. Each set is stored as a matrix of size $J \times d$ where J is the number of samples and d is the dimension of the filtering problem. The storage requirement is suitable for conducting all the computations on modern GPUs. In terms of the number of floating point operations, Line 6 – 9 in Algorithm 1 for fixed t and τ involves $\mathcal{O}(J \times N \times d)$ operations including element-wise operations and matrix summations, where $N < J$ is the size of the mini-batch used to estimate the weights in Eq. (22). So the total number of floating point operations is on the order of $\mathcal{O}(J \times N \times d \times K)$ to update the filtering density from t to $t+1$, where K is the number of time steps for discretizing the reserse SDE. The numerical experiments in Section 4 show that the number of samples J can grow very slowly with the dimension d while maintaining a satisfactory performance for tracking the Lorenz-96 model, which indicates the superior efficiency of EnSF in handling extremely high-dimensional filtering problems.

4. Numerical experiments: tracking the 1,000,000-dimensional Lorenz-96 model

We demonstrate EnSF's capability in handling the high-dimensional Lorenz-96 model. Specifically, we track the state of the Lorenz-96 model described as follows:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} + F, \qquad i = 1, 2, \ldots, d, \quad d \geq 4, \tag{25}$$

where $X_t = [x_1(t), x_2(t), \ldots, x_d(t)]^\top$ is a d-dimensional target state, and it is assumed that $x_{-1} = x_{d-1}$, $x_0 = x_d$, and $x_{d+1} = x_1$. The term F is a forcing constant. When $F = 8$, the Lorenz-96 dynamics (25) becomes a chaotic system, which makes tracking the state X_t a challenging task for all the existing filtering techniques, especially in high dimensional spaces. In our numerical experiments, we discretize Eq. (25) through the Runge–Kutta (RK4) scheme. To avoid NaN values in the experiments, we clip the solutions of the forward solver at a magnitude of 50, as extreme values can lead to numerical instabilities. Specifically, we set the ensemble values to 50 or −50 when they exceed this range. To initialize the Lorenz-96 system, we first pick a random sample from $N(\mathbf{0}, 3^2\mathbf{I}_d)$ and then run 1000 burn-in simulation steps through the RK4 scheme to obtain our true initial state X_0. Our initial guess for the initial ensemble X_0 is a standard Gaussian random variable $N(\mathbf{0}, \mathbf{I}_d)$, which means that we do not possess any effective information about X_0 at the beginning.

Since EnSF is designed as a nonlinear filter for high-dimensional problems, we carry out experiments on one million-dimensional Lorenz-96 system, i.e., $d = 1,000,000$, where the observational process in Eq. (2) is an arctangent function of the state, i.e.,

$$Y_{t+1} = \arctan(X_{t+1}) + \varepsilon_{t+1}. \tag{26}$$

The chaotic state dynamics in Eq. (25) along with the highly linear observation in Eq. (26) would make the tracking of the Lorenz-96 system extremely challenging, especially in such high-dimensional space. In what follows, we shall demonstrate the performance of EnSF in solving the above Lorenz-96 tracking problems in one-million-dimensional space, and we shall also carry out a series of experiments to compare our method with the state-of-the-art optimal filtering method, i.e., the Local Ensemble Transform Kalman Filter (LETKF), which is the method adopted by the European Center for Medium-Range Weather Forecasts for hurricane forecasting.

Remark (***Reproducibility***)**.** EnSF method for the high-dimensional Lorenz-96 problem is implemented in Pytorch with GPU. The source code is publicly available at https://github.com/zezhongzhang/EnSF. The numerical results in this section can be exactly reproduced using the code on Github.

4.1. *Illustration of EnSF's accuracy*

In the first experiment, we illustrate the accuracy of EnSF in tracking the 1,000,000-dimensional Lorenz-96 model, and we track the target state 800 time steps with temporal step size $\Delta t = 0.01$ and observational noise $\varepsilon_t \sim (\mathbf{0}, 0.05^2\mathbf{I}_d)$.

In Fig. 1, we illustrate the nonlinearity of the observation system by comparing the true state X_t and the observation Y_t along four randomly selected directions. Due to the nonlinearity of $arctan()$, the observation Y_t does not provide sufficient information of the state when X_t is outside the domain $[-\pi/2, \pi/2]$. When it happens, the partial derivative of Y_t is very close to zero such that there is very little update of the score function in Eq. (18) along the directions with states outside $[-\pi/2, \pi/2]$. In other words, there may be only a small subset of informative observations at each filtering time step.

Fig. 2 shows the comparison between the true state trajectories and the estimated trajectories, each sub-figure shows the trajectories along randomly selected three directions in the 1,000,000-dimensional state space. EnSF is implemented with 500 pseudo time steps when discretizing the reserse SDE, and the ensemble size that we picked is 20 samples. Since EnSF's initial estimate is randomly sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, it is far from the true initial state. After several filtering steps, EnSF gradually captures the true state by assimilating the observational data. Even though there are some discrepancy between the true and the estimated states, the accuracy of EnSF is sufficient for capturing such a high-dimensional chaotic system.

4.2. *Comparison between EnSF and LETKF*

In the following numerical experiments, we compare EnSF with LETKF in tracking the $1,000,000$-dimensional Lorenz-96 model, and we track the target state 1500 time steps with temporal step size $\Delta t = 0.01$ and observational noise $\varepsilon_t \sim (\mathbf{0}, 0.05^2\mathbf{I}_d)$. To allow gaps between prediction and update, we implement the Bayesian update procedure with time step size 0.1, i.e., we implement EnSF or LETKF to update the posterior filtering density after simulating the Lorenz-96 model 10 time steps.

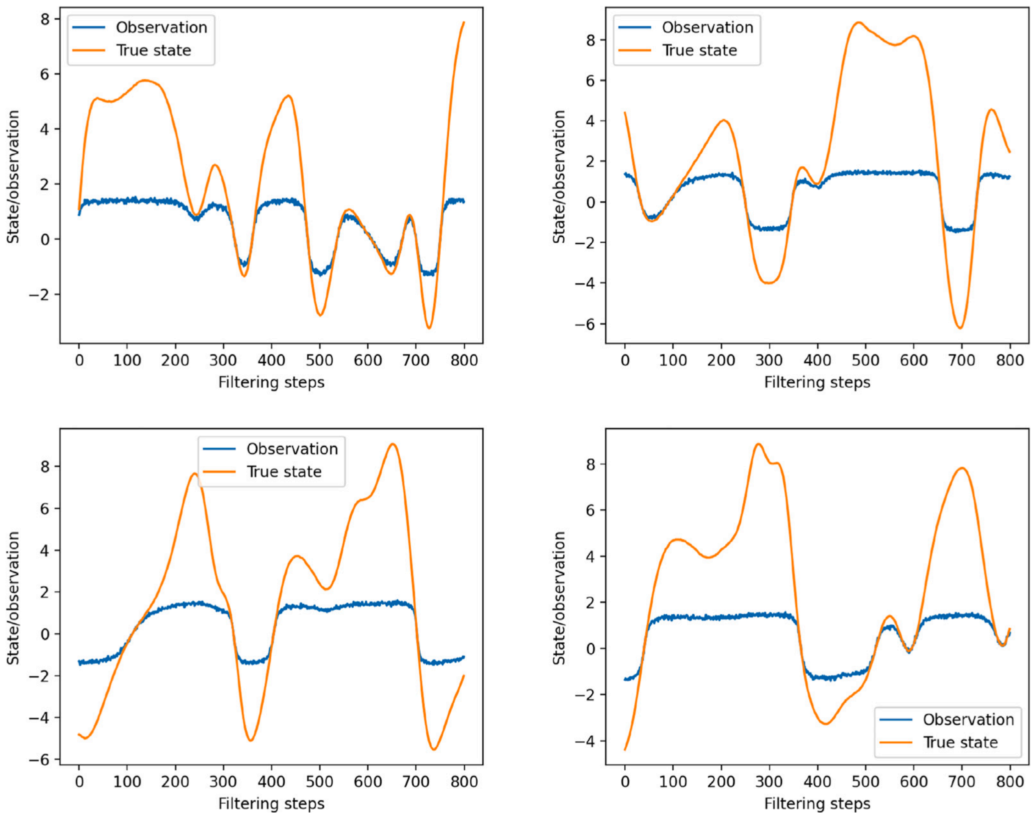


Fig. 1. Illustration of the nonlinearity of observation process by comparing the true state X_t and the observation Y_t along four randomly selected directions. Due to the nonlinearity of $arctan()$, the observation Y_t does not provide sufficient information of the state when X_t is outside the domain $[-\pi/2, \pi/2]$.

4.2.1. Hyperparameter fine tuning

An important feature of LETKF is that it utilizes an inflation factor and a localization factor to fine-tune the behavior of the filter so that the fine-tuned LETKF can be adapted to a specific optimal filtering problem. Therefore, before we conduct the comparison experiments, we first fine-tune LETKF. According to the computational cost shown in Fig. 7, it takes around 300 s for LETKF to perform one filtering step in tracking the 1,000,000-dimensional Lorenz-96 model. This means finishing the fine-tuning chart for LETKF in Figs. 3 and 4 for the one million dimensional cases will cost about 520 days using a single RTX 3070 GPU, which is not practical. Therefore, we perform LETKF fine-tuning in the 100 dimensional space ($d = 100$), which costs around 2 h to generate the fine-tuning charts. Additionally, fine-tuning in 100-dimensional space and testing in 1,000,000-dimensional space will demonstrate the transferability of LETKF and EnSF.

Specifically, we let the inflation factor vary from $0.9 - 1.8$, and the localization factor is tested from $0.0001 - 9$,[1] and the ensemble size that we picked for LETKF is 20. Then, we solve the corresponding Lorenz-96 tracking problem repeatedly 10 times, and the overall RMSEs averaged on all data assimilation times are presented in Fig. 3. We also use a colorbar to visually represent the various RMSEs and provide an intuitive understanding of the fine-tuned results. In Fig. 4, we present the average RMSEs (over 10 repetitions) on the last 50 data assimilation times, which indicates the converged performance of LETKF in the tuning procedure. Based on Fig. 3, we choose the best three parameter combinations for LETKF:

- LETKF (No. 1): Inflation=1.1, localization=4;
- LETKF (No. 2): Inflation=1.0, localization=2;
- LETKF (No. 3): Inflation=1.1; localization=3.

The selected LETKF parameters are highlighted in both Figs. 3 and 4, and will be used for further comparisons with EnSF.

To compare with LETKF, we also fine-tune EnSF's hyperparameters, which are introduced in Eq. (20). In Figs. 5 and 6, we present the fine-tune charts for EnSF under the same setup as LETKF, with the RMSEs presented in each block marked by the same colorbar

[1] The corresponding testing ranges for inflation and localization are already optimized based on our experience. In practice, one may need to test the inflation factor and the localization factor in much larger ranges.

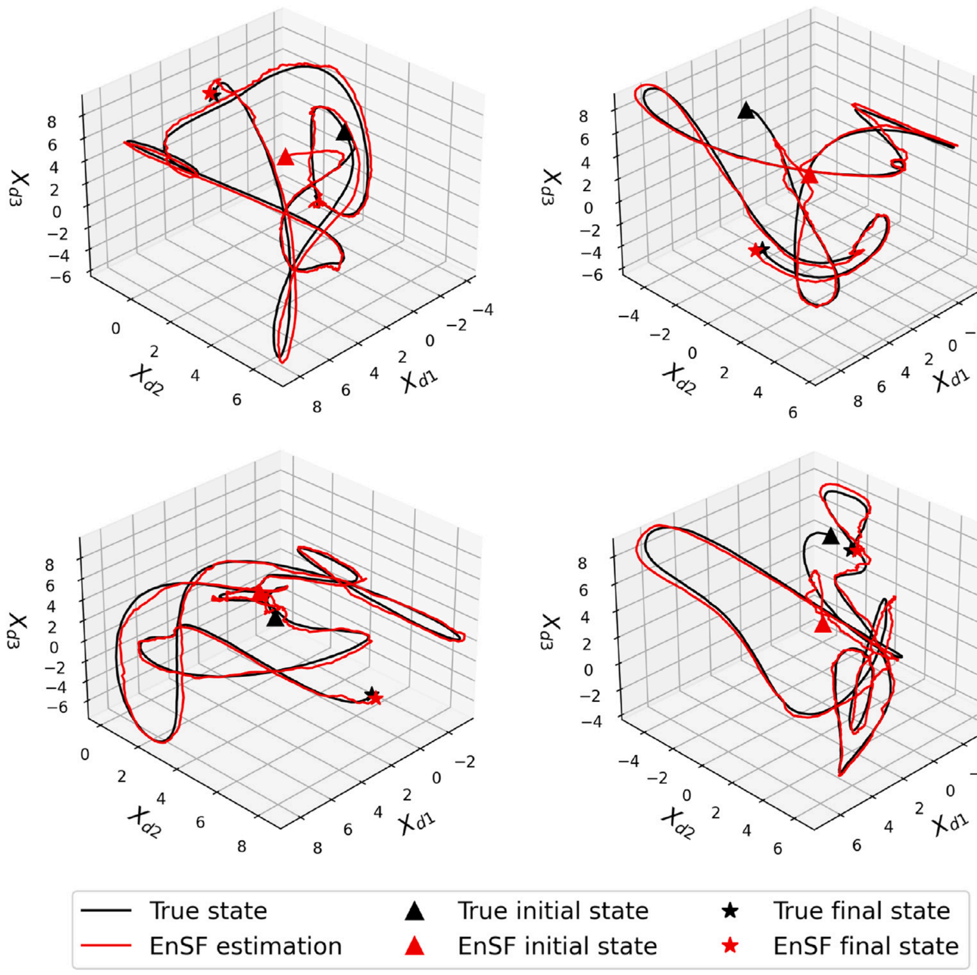


Fig. 2. Comparison between the true state trajectories and the estimated trajectories obtained by EnSF, each sub-figure shows the trajectories along randomly selected three directions in the 1,000,000-dimensional state space. We observe that even though the initial guess for EnSF is far from the true initial state, EnSF gradually captures the true state by assimilating the observational data after several filtering steps, providing sufficient accuracy in capturing such a high-dimensional chaotic system.

as used for LETKF fine-tune charts. The best three parameter combinations for EnSF are highlighted in both figures and will be used for comparison. They are as follows:

- EnSF (No. 1): $\epsilon_\alpha = 0.5$, $\epsilon_\beta = 0.025$;
- EnSF (No. 2): $\epsilon_\alpha = 0.6$, $\epsilon_\beta = 0.025$;
- EnSF (No. 3): $\epsilon_\alpha = 0.5$, $\epsilon_\beta = 0.05$.

We can see from the fine-tune charts for LETKF and EnSF that both methods achieve comparable accuracy with their optimal hyperparameters. However, EnSF is less sensitive to these hyperparameters with a wide range of configurations yielding good performance. On the other hand, the performance of LETKF varies dramatically, which indicates that it is very sensitive to the choice of inflation factor and localization factor. Such sensitivity may cause additional difficulty when attempting to fine-tune LETKF in more complex problems.

4.2.2. Efficiency comparison between EnSF and LETKF

To proceed, we first carry out an efficiency comparison between EnSF and then show the performance comparison in solving the one million dimensional problem. In Fig. 7, we showcase the computational cost for implementing *one data assimilation step* of EnSF and LETKF in solving problems ranging from 100 dimensions up to 1,000,000 dimensions. The ensemble size chosen for both methods is 20. The CPU used is a 6-core AMD Ryzen™ 5 5600X, and the GPU used is an NVIDIA RTX 3070. Both EnSF and LETKF are implemented on CPU and GPU. The LETKF is tested with one-sided neighbor sizes of 3 (LETKF_n3) and 17 (LETKF_n17). The

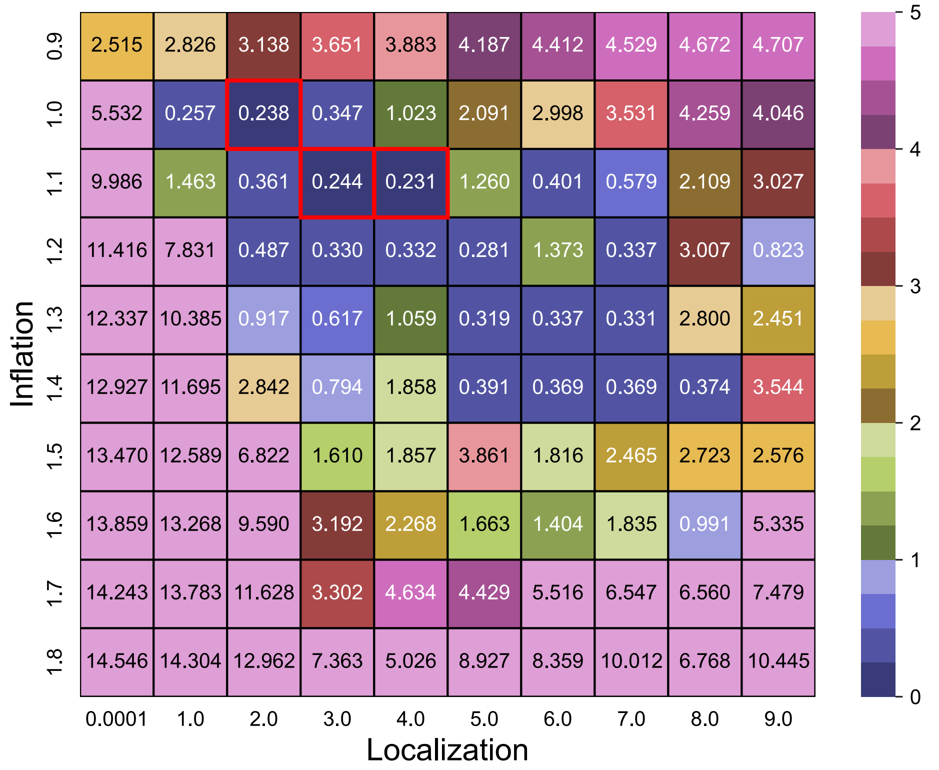


Fig. 3. LETKF's fine-tuning chart where the RMSE is averaged on all data assimilation times with 10 repetitions. The highlighted cells are the best three parameter combinations selected for LETKF.

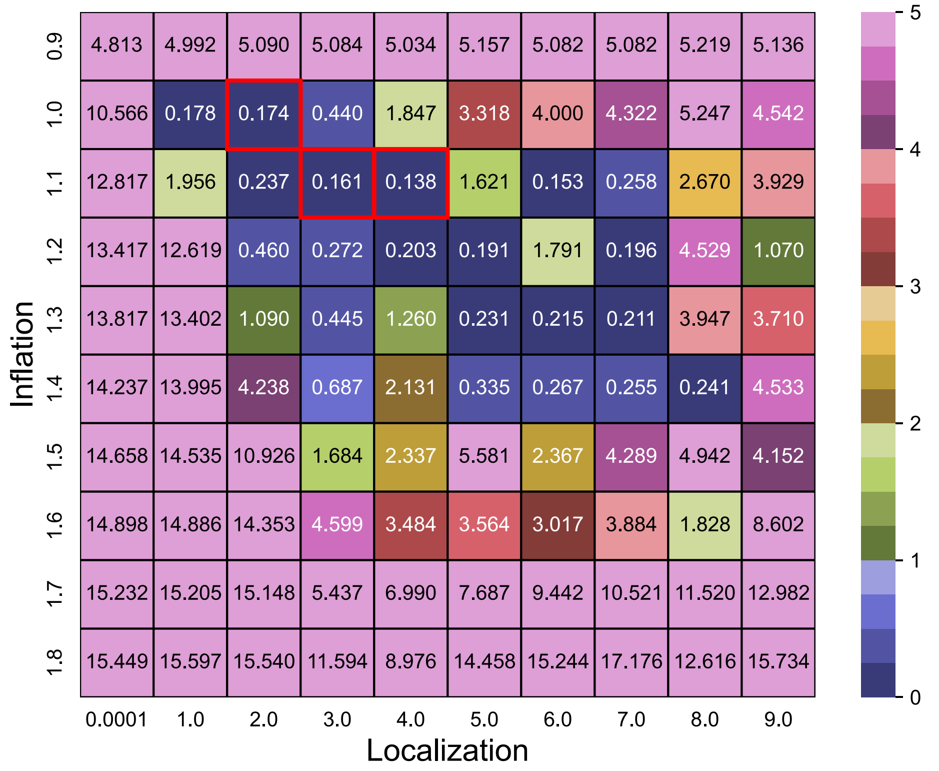
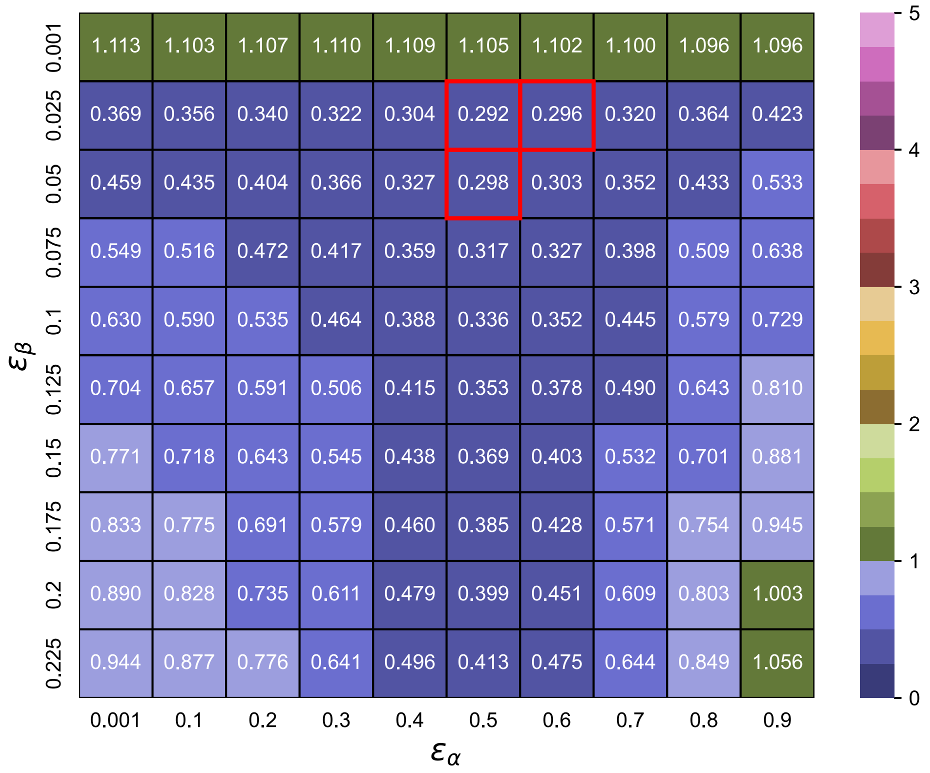


Fig. 4. LETKF's fine-tuning chart where the RMSE is averaged on the last 50 data assimilation times with 10 repetitions. The highlighted cells are the best three parameter combinations selected for LETKF.



ε_β \ ε_α	0.001	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.001	1.113	1.103	1.107	1.110	1.109	1.105	1.102	1.100	1.096	1.096
0.025	0.369	0.356	0.340	0.322	0.304	0.292	0.296	0.320	0.364	0.423
0.05	0.459	0.435	0.404	0.366	0.327	0.298	0.303	0.352	0.433	0.533
0.075	0.549	0.516	0.472	0.417	0.359	0.317	0.327	0.398	0.509	0.638
0.1	0.630	0.590	0.535	0.464	0.388	0.336	0.352	0.445	0.579	0.729
0.125	0.704	0.657	0.591	0.506	0.415	0.353	0.378	0.490	0.643	0.810
0.15	0.771	0.718	0.643	0.545	0.438	0.369	0.403	0.532	0.701	0.881
0.175	0.833	0.775	0.691	0.579	0.460	0.385	0.428	0.571	0.754	0.945
0.2	0.890	0.828	0.735	0.611	0.479	0.399	0.451	0.609	0.803	1.003
0.225	0.944	0.877	0.776	0.641	0.496	0.413	0.475	0.644	0.849	1.056

Fig. 5. EnSF's fine-tuning chart where the RMSE is averaged on all data assimilation times with 10 repetitions. The highlighted cells are the best three parameter combinations selected for EnSF. Compared to LETKF, EnSF's performance is much more stable with respect to small changes of the hyperparameters.

neighbor size refers to the number of state variables retained for calculating the localized covariance, which depends on the value of the localization parameter. The one-step data assimilation computing time is calculated as the average of 20 repetitions.

From this efficiency figure, we can see that EnSF is much more efficient than LETKF. From the 100 dimension to the 10,000 dimension, the main computational cost for EnSF is essentially the background computation, and it is only 0.17 s per step on average. For the $1,000,000$ dimensional problem, the average computational cost for EnSF is only approximately 5 seconds per step. In addition, Fig. 7 demonstrates that EnSF is particularly well-suited for GPU computing on a large scale. Due to the sequential nature of CPU computing, the computing time grows almost linearly with the problem size. In contrast, GPU computing is much faster for large-scale matrix operations because of its inherent parallelization. Since all computations in EnSF can be structured as large matrix operations, it can efficiently utilize GPU computing power. In fact, GPU computation time remains constant as long as the matrix size fits within the GPU's tensor core capacity, which explains why the computing time for EnSF (GPU) remains constant for problems with dimensions up to 10,000. The computing time starts to grow linearly only after the matrix size surpasses the GPU's tensor core capacity, which occurs at approximately 100,000 dimensions. It is also worth noting that all GPU benchmarks were conducted using an RTX 3070, an older mid-tier gaming GPU. The scalability of EnSF would be significantly enhanced with modern, professional-grade GPUs, further improving its performance in large-scale applications.

On the other hand, the computational cost of LETKF is approximately 300 seconds per step when implemented with 20 Kalman filter samples. Although a 6-core CPU could potentially run LETKF faster, the total computational cost of LETKF is still much higher than that of EnSF, and LETKF is not suitable for modern GPU machines, which makes it difficult to further scale LETKF algorithms in practical implementations. Due to the extremely high computational cost of LETKF in solving the 1,000,000-dimensional problem, it is not feasible to fine-tune LETKF in the one-million-dimensional space. Therefore, we utilize the optimal hyperparameters for both methods that we obtained in the 100-dimensional space to run the 1,000,000-dimensional problem.

4.2.3. Experimental setting 1: baseline test

We first conduct a baseline comparison using the same problem setting that was used to fine-tune LETKF and EnSF, except that the dimension of the Lorenz-96 model is now $d = 1,000,000$. The comparison of root mean square errors (RMSEs) at data assimilation times is presented in Fig. 8, where we have selected the top three sets of hyperparameters for each method, and the RMSEs are plotted with respect to time. In our numerical experiments, RMSEs are calculated by repeating the same test 10 times with different random initial conditions, and we average the estimation errors over all 1,000,000 directions and 10 repetitions. We

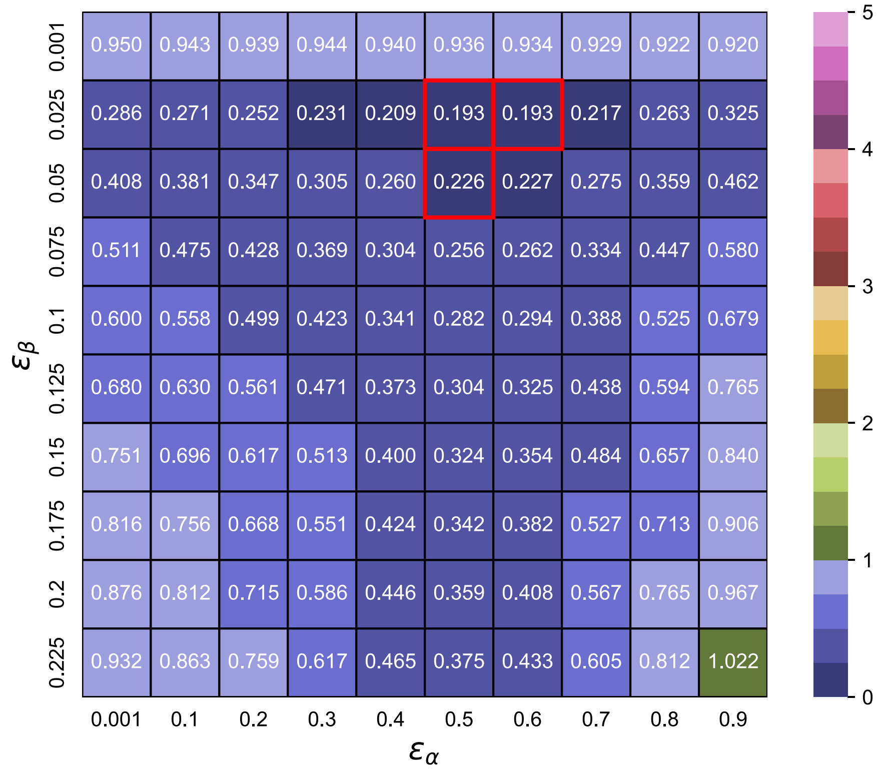


Fig. 6. EnSF's fine-tuning chart where the RMSE is averaged on the last 50 data assimilation times with 10 repetitions. The highlighted cells are the best three parameter combinations selected for EnSF. Compared to LETKF, EnSF's performance is much more stable with respect to small changes of the hyperparameters.

can see that both EnSF and LETKF provide good accuracy for tracking the target Lorenz-96 state. EnSF performs consistently well with three different sets of hyperparameters, while the accuracy of LETKF varies.

Fig. 9 shows the comparison of RMSEs at every time step — including prediction-only steps and data assimilation steps where prediction and Bayesian updates are performed. From this figure, we can see fluctuations in estimation errors for both methods. When providing good accuracy, the difference between prediction-only errors and data assimilation errors is small. However, for one LETKF test, which provided the least accurate result, there is a larger variance between prediction-only steps and data assimilation steps. This partially explains why the hyperparameter choice "No. 1" of LETKF did not work as well as the other two.

4.2.4. Experimental setting 2: reduced observation noise test

Next, we modify the problem setting by reducing the observational noise from $\varepsilon_t \sim (\mathbf{0}, 0.05^2\mathbf{I}_d)$ to $\varepsilon \sim (\mathbf{0}, 0.03^2\mathbf{I}_d)$ and conduct the same comparison experiment. The corresponding RMSEs at data assimilation time steps are presented in Fig. 10, and the RMSEs at all time steps are presented in Fig. 11.

Although the observational data are more accurate in this experiment, two of the top choices of hyperparameters for LETKF diverge, and the hyperparameter that provides the best result in this experiment is actually the third-best choice from the fine-tune chart. This result shows that while the fine-tuned LETKF provides higher accuracy, it is very sensitive to the problem setting. Even slight modifications to the problem may cause severe divergence in LETKF, with no indication beforehand which set of hyperparameters will fail. In addition, the all-time RMSEs presented in Fig. 11 verify that LETKF failed with hyperparameters No. 1 and No. 2. In comparison, EnSF continues to provide very accurate estimates for the target state for the top-three choices of its hyperparameters.

To better illustrate the performance of EnSF and LETKF, we plot the average ensemble spread in Fig. 12. The ensemble spread is the square root of the average ensemble variance, i.e., $\sqrt{\|Var(\boldsymbol{x}_{ensemble})\|_1/d}$, where the variance is calculated for each dimension of the ensemble. While a larger ensemble spread allows a filtering method to better cover the true signal, overly wide spread state samples provide less useful information about the true target state, which makes the predicted state less reliable. This issue is also reflected in the all-time RMSEs shown in Fig. 11, where the hyperparameters causing widely spread LETKF samples correspond to large fluctuations in the RMSEs. On the other hand, EnSF maintains very stable ensemble spread sizes, and the best-performing LETKF shows a converging ensemble spread.

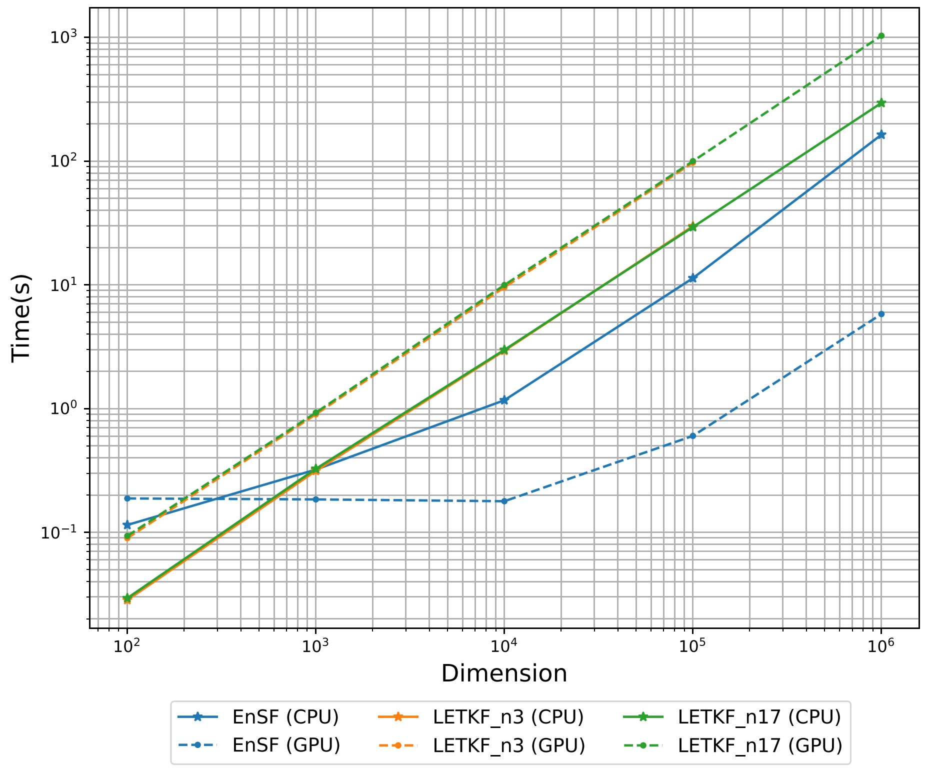


Fig. 7. One step data assimilation computational cost with ensemble size = 20. EnSF with GPU implementation is much more efficient than LETKF, and EnSF is more suitable for modern GPU machines.

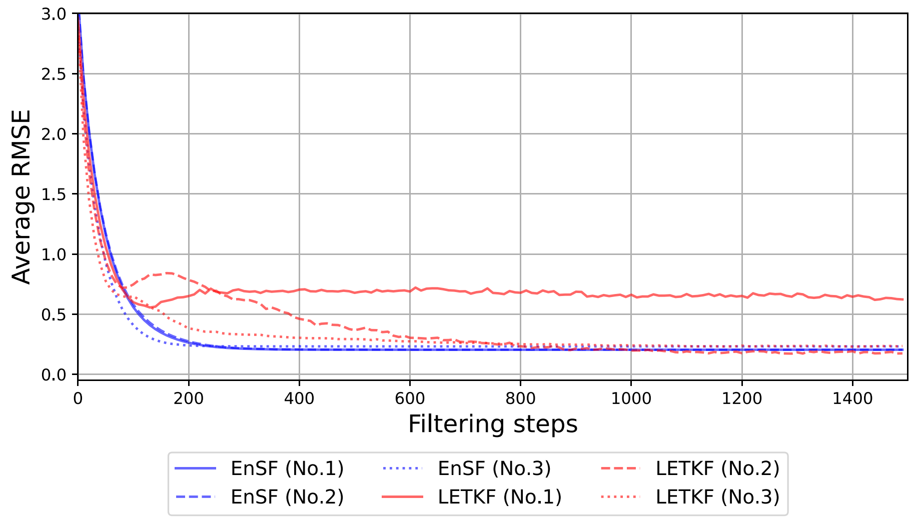


Fig. 8. RMSEs comparison between EnSF and LETKF at data assimilation times. RMSEs are calculated by repeating the same test 10 times with different random initial conditions, and we average the estimation errors over all 1,000,000 directions and 10 repetitions. No. 1, No. 2, and No. 3 in the legend correspond to the first, second, and third-best hyperparameters, respectively. We observe that EnSF performs consistently well with the top three sets of hyperparameters, while the accuracy of LETKF varies.

4.2.5. Experimental setting 3: incomplete knowledge of the model error

In the last experimental setting, we address a more challenging but realistic scenario involving *an imperfect model due to incomplete knowledge*. In this scenario, we assume that the state model may not fully reflect the true state propagation, and we model this unknown portion as a mixture of three levels of independent Gaussian-type random shocks. On the evolution of the true state trajectory, we introduce independent shocks with probabilities of 2%, 1%, and 0.5%, with corresponding shock sizes of 5%, 20%,

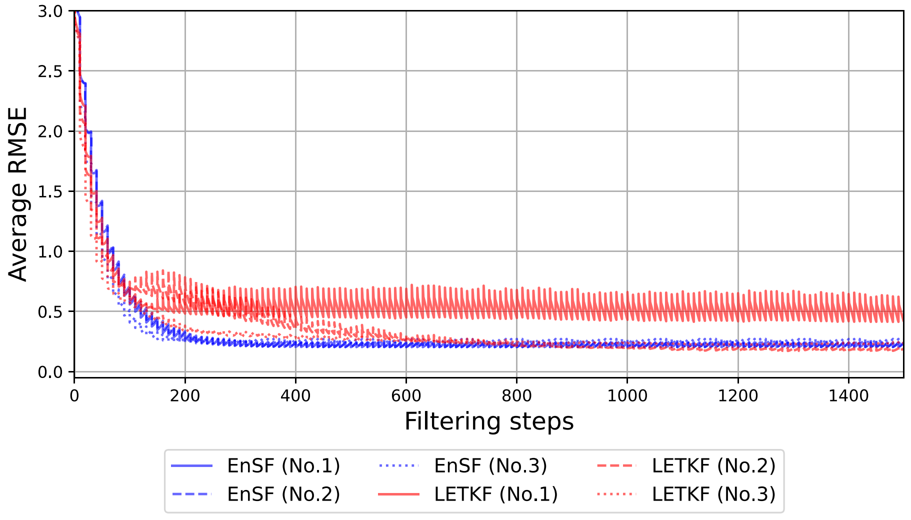


Fig. 9. RMSEs comparison between EnSF and LETKF at every time step — including prediction-only steps and data assimilation steps. Compared to Fig. 8, we observe fluctuations in estimation errors for both methods.

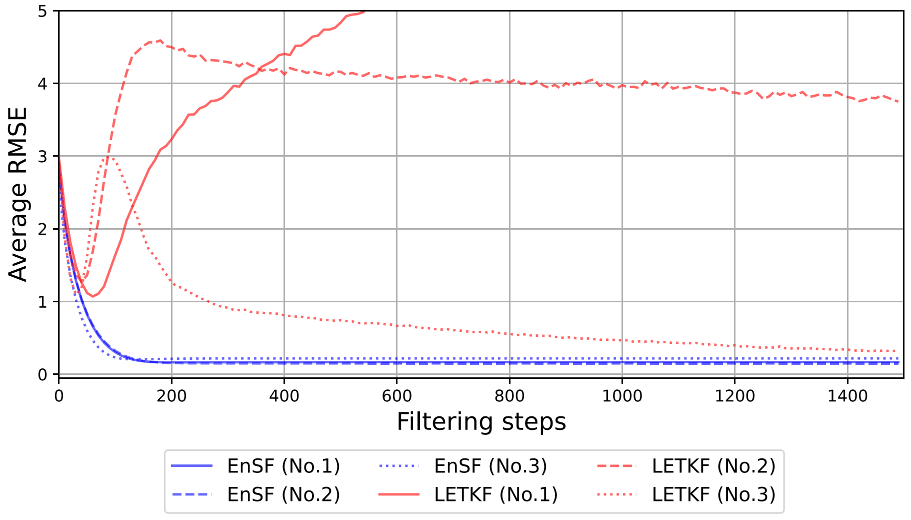


Fig. 10. RMSEs comparison between EnSF and LETKF at data assimilation time steps with smaller observational noise $\epsilon_t \sim (\mathbf{0}, 0.03^2\mathbf{I}_d)$. We observe that although the observational data are more accurate, two of the top choices of hyperparameters for LETKF diverge. In comparison, EnSF continues to provide very accurate estimates for the target state.

and 50% relative to the current state magnitude of the Lorenz-96 model, respectively. For example, when a size 50% shock happens, every component of the true state $\boldsymbol{x}_t^i$ is perturbed by an additive term $0.5Z_i|\boldsymbol{x}_t^i|$, where $\boldsymbol{x}_t^i$ is the ith dimension of the true state and Z_i are i.i.d. standard Gaussian noise.

This problem setting mimics a situation where there is a small chance (2%) that the model is inaccurate with a 5% error. There is an even smaller chance (1%) that the model error is larger, at 20% level, and a very small chance (0.5%) that a large-scale unexpected error occurs in the model. In practical applications, this scenario is quite common due to the limited knowledge we have about the real world. For example, in weather forecasting, this variety of unknown model errors is used to simulate the effects of flow-dependent model uncertainties (see discussions in [41]).

In Fig. 13, we compare the RMSEs, which are calculated by averaging the estimation errors over all 1,000,000 directions, of EnSF with LETKF in the imperfect model scenario at data assimilation time steps. The observational noise is kept at $\epsilon_t \sim (\mathbf{0}, 0.05^2\mathbf{I}_d)$, which is the setting used for fine-tuning. The figure also shows the time instants when the unexpected shocks occur during the data assimilation period. We can see from Fig. 13 that all three settings of EnSF can quickly recover from unexpected shocks. However, LETKF either diverges or struggles to recover from the shocks. Unlike the previous test, where fine-tuning is possible for a smaller observational noise value, the unexpected shocks in this experiment are caused by the "unknown" portion of the state model. Since we lack information about these unknown shocks, we cannot fine-tune either EnSF or LETKF. To further validate the

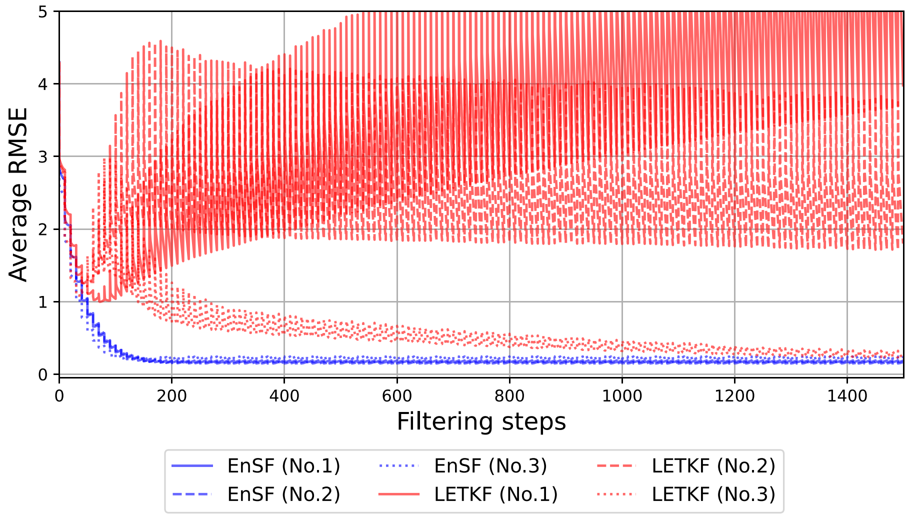


Fig. 11. RMSEs comparison between EnSF and LETKF at every time step with smaller observation noise $\epsilon_t \sim (\mathbf{0}, 0.03^2\mathbf{I}_d)$.

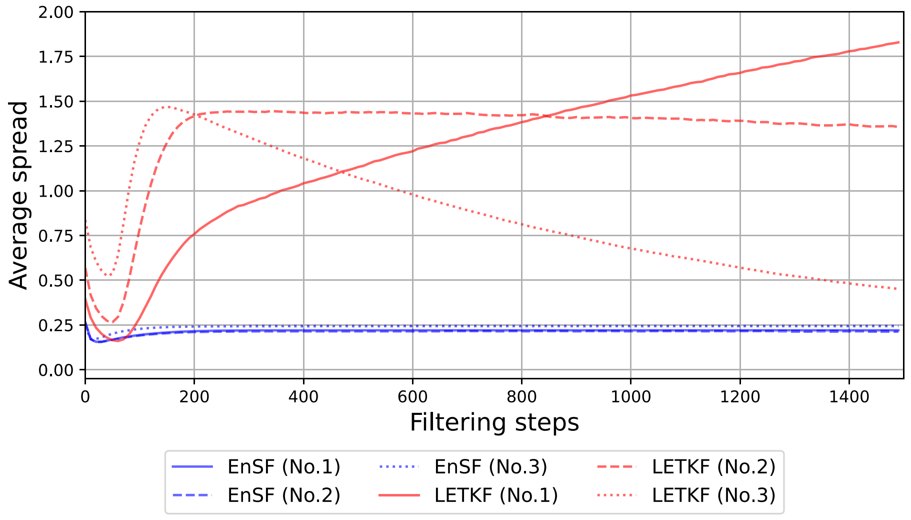


Fig. 12. Average ensemble spread in the smaller observation noise test. While a larger ensemble spread allows a filtering method to better cover the true signal, overly wide spread state samples provide less useful information about the true target state.

reliable performance of EnSF, we repeat the above experiments four times with different occurrences of random shocks and show the corresponding tracking RMSEs of EnSF in Fig. 16. From this figure, we can see that EnSF consistently generates low errors and quickly recovers from unexpected shocks.

In Fig. 14, we present the RMSE comparison at every time step, and in Fig. 15 we show the comparison of average ensemble spreads in this incomplete knowledge experiment. From these figures, we can see that LETKF has highly fluctuating estimation errors, and its average ensemble spreads are generally wide or even divergent. These two pieces of evidence indicate that LETKF is not stable enough to handle unknown model errors due to incomplete knowledge or information. To further validate the reliable performance of EnSF, we repeat the above experiments four times with different realizations of random shocks and show the corresponding tracking RMSEs of EnSF in Fig. 16. From this figure, we can see that EnSF constantly generates low errors, and it always recovers from unexpected shocks quickly.

In this experimental setting, we introduce an extra metric to evaluate the performance of EnSF and LETKF, namely the continuous ranked probability score (CRPS). Specifically, at a given time step, for a marginal dimension i, let $F^i_{ensmeble}(z)$ and $F^i_{true}(z)$ be the empirical cumulative distribution functions of the ensemble and the true state, respectively, where $F^i_{true}(z) = 1_{z>\mathbf{x}^i_{true}}(z)$. Then, we let $CRPS^i := \int (F^i_{ensmeble}(z) - F^i_{true}(z))^2 dz$, and we average the CRPS over all $1,000,000$ dimensions. The CRPS measures how well the ensemble distribution matches with the true state. Lower CRPS values indicate that the ensemble distribution is well-aligned with the true state with a small uncertainty, while higher CRPS values indicate otherwise. In this experiment, we plot the CRPS

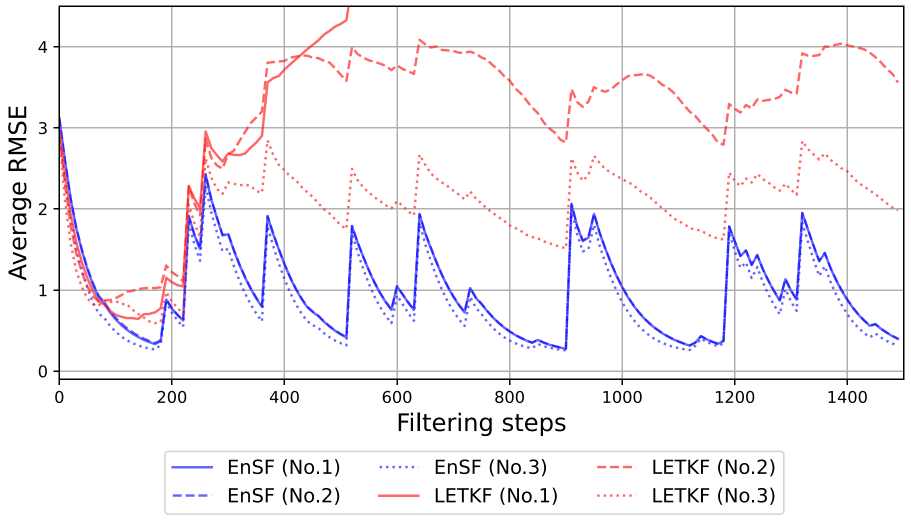


Fig. 13. RMSEs comparison between EnSF and LETKF in the incomplete knowledge experiment, where the unknown model error is injected into the state equation as random shocks. We observe that EnSF can quickly recover from unexpected shocks, but LETKF either diverges or struggles to recover from the shock.

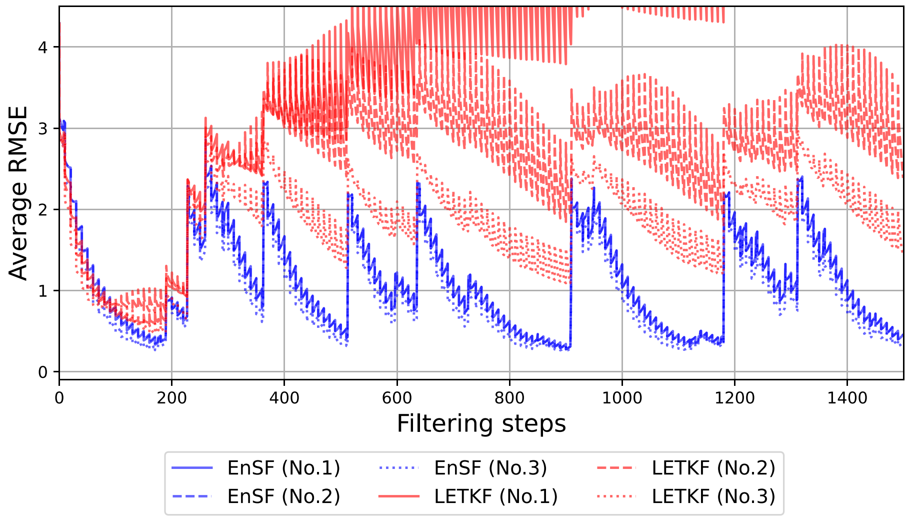


Fig. 14. Comparison between EnSF and LETKF at every time step in the incomplete knowledge experiment, where the unknown model error is injected into the state equation as random shocks. We observe that LETKF has highly fluctuating estimation errors, but EnSF's error fluctuation between two filtering steps is much smaller.

comparison between EnSF and LETKF at data assimilation steps in Fig. 17. From this figure, we can see that EnSF outperforms LETKF in this CRPS comparison.

5. Conclusion

We propose the EnSF method to solve very high-dimensional nonlinear filtering problems. The avoidance of training neural networks to approximate the score function makes it computationally feasible for EnSF to efficiently solve the 1,000,000-dimensional Lorenz-96 problem. We observe in the numerical experiments that one million dimensions is definitely not the upper limit of EnSF's capability, especially with the help of modern high-performance computing. In addition to exploring higher-dimensional cases, several key aspects of EnSF can be improved in future work. First, we will investigate how fast the number of samples, i.e., J in Algorithm 1, needs to grow with the dimensionality to ensure robust performance. Second, we will expand the capability of the current EnSF to handle partial observations, i.e., only a subset of the state variables are involved in the observation process, which is critical to real-world data assimilation problems. In fact, Fig. 1 shows that the $arctan()$ observation function can be reviewed as a

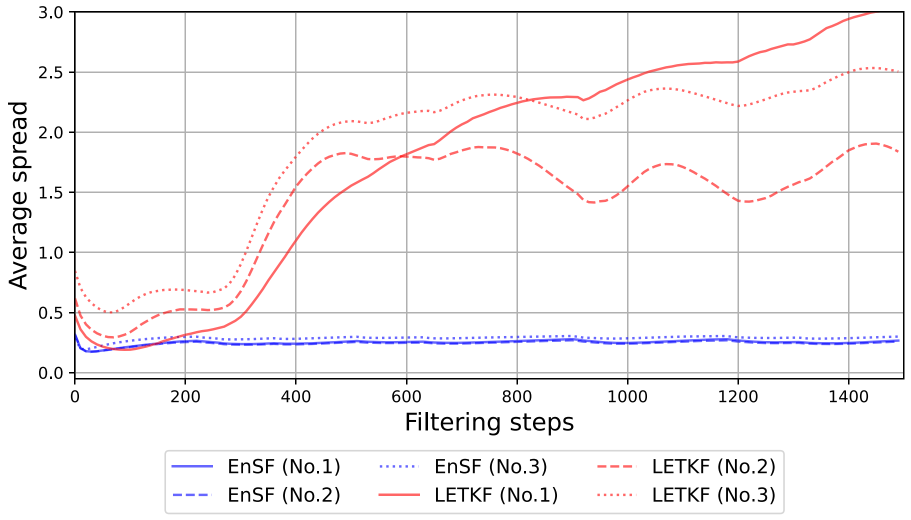


Fig. 15. Average ensemble spread in the incomplete knowledge experiment, where the unknown model error is injected into the state equation as random shocks. We observe that the ensemble spreads of LETKF are generally wide or even divergent.

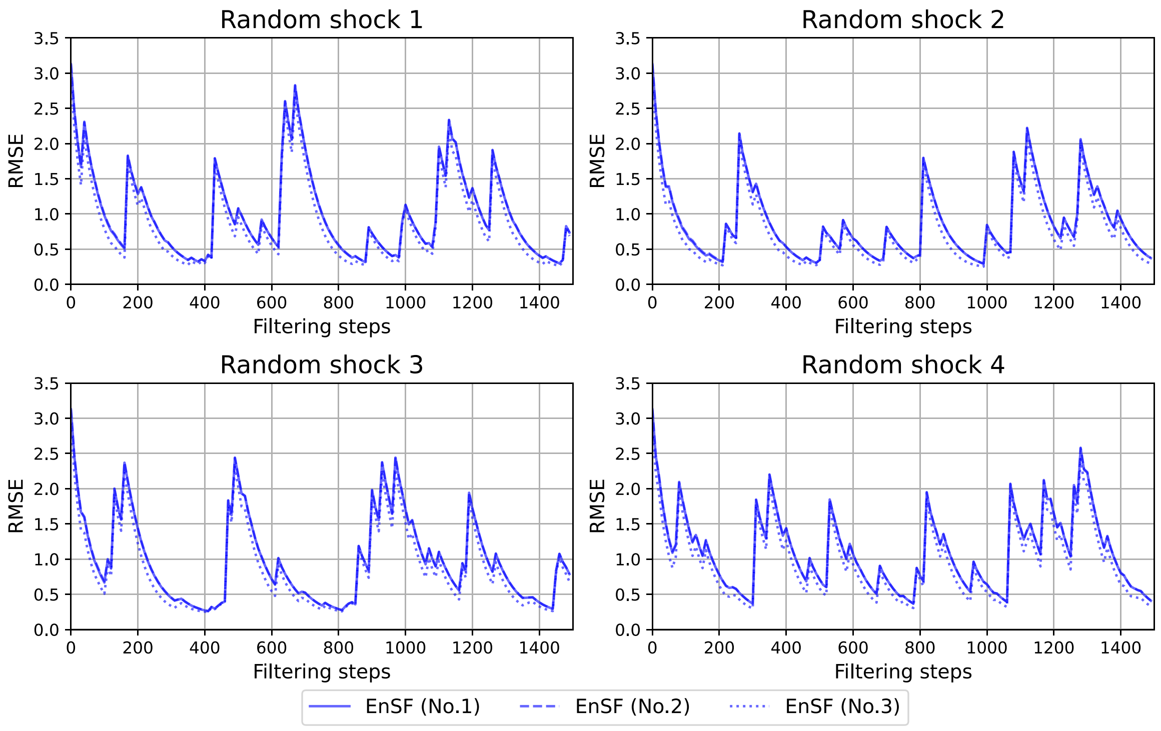


Fig. 16. Repeated experiments of EnSF in the incomplete knowledge scenarios, each subfigure shows the RMSE of EnSF for a different occurrence of random shocks. We observe that EnSF performs stably with different random shock patterns.

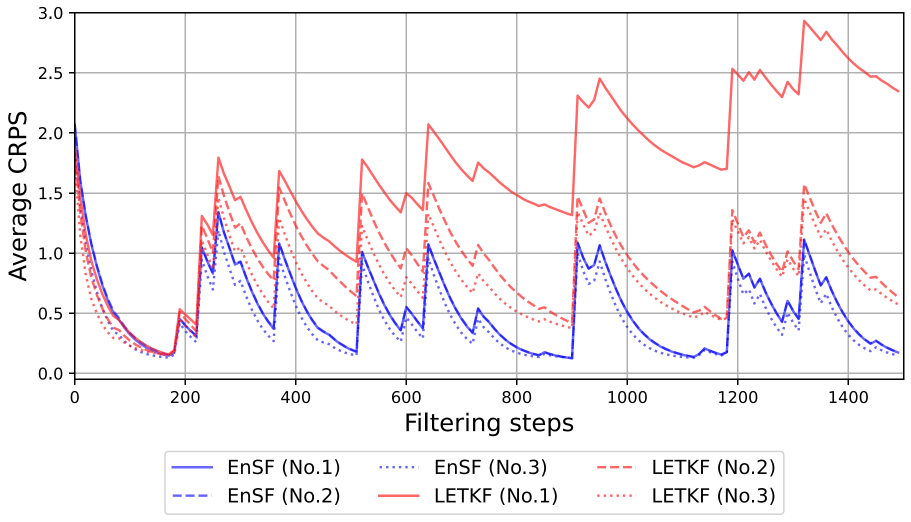


Fig. 17. CRPS comparison for posterior filtering densities. Lower CRPS values indicate that the distribution is well-aligned with the true state. We observe that EnSF stably outperforms LETKF in this experiment.

partial observation in the sensing that there is no observational information when the state is outside $[-\pi/2, \pi/2]$. Third, the current definition of the weight function $h(\tau)$ in Eq. (18) for incorporating the likelihood into the score function is empirical. The current choice of $h(\tau)$ may introduce a bias into the posterior state estimation. We will investigate whether there is an optimal weight function to gradually incorporate the likelihood information into the reserse SDEs. Fourth, the efficiency of reserse sampling can also be improved by incorporating advanced stable time-stepping schemes, e.g., the exponential integrator, to significantly reduce the number of time steps in the discretization of the reserse process in the diffusion model. Fifth, we will test the performance of EnSF for real-world models, e.g., the IFS model developed by ECMWF, and the existing AI-based weather models, e.g., FourCastNet, GraphCast.

CRediT authorship contribution statement

Feng Bao: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Conceptualization. **Zezhong Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Guannan Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under the contract ERKJ443 at the Oak Ridge National Laboratory, which is operated by UT-Battelle, LLC, for the U.S. Department of Energy under Contract DE-AC05-00OR22725. The first author (FB) would also like to acknowledge the support from U.S. National Science Foundation through project DMS-2142672 and the support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Grant DE-SC0022297.

Data availability

No data was used for the research described in the article.

References

[1] F. Bao, Y. Cao, P. Maksymovych, Backward sde filter for jump diffusion processes and its applications in material sciences, Commun. Comput. Phys. 27 (2020) 589–618.
[2] F. Bao, N. Cogan, A. Dobreva, R. Paus, Data assimilation of synthetic data as a novel strategy for predicting disease progression in alopecia areata, Math. Med. Biol. J. IMA 06 (2021).
[3] M.F. Bugallo, T. Lu, P.M. Djuric, Target tracking by multiple particle filtering, in: 2007 IEEE Aerospace Conference, 2007, pp. 1–7.
[4] G. Evensen, The ensemble Kalman filter for combined state and parameter estimation: Monte Carlo techniques for data assimilation in large systems, IEEE Control Syst. Mag. 29 (3) (2009) 83–104.
[5] B. Ramaprasad, Stochastic filtering with applications in finance, 2010.
[6] G. Evensen, Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte carlo methods to forecast error statistics, J. Geophys. Res.: Oceans 99 (1994) 10143–10162.
[7] P.L. Houtekamer, H.L. Mitchell, Data assimilation using an ensemble kalman filter technique, Mon. Weather Rev. 126 (3) (1998) 796–811.
[8] G. Evensen, Data Assimilation: The Ensemble Kalman Filter, Springer-Verlag, Berlin, Heidelberg, 2006.
[9] B.R. Hunt, E.J. Kostelich, I. Szunyogh, Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter, Physica D 230 (2007) 112–126.
[10] T. Miyoshi, A. Amemiya, S. Otsuka, Y. Maejima, J. Taylor, T. Honda, H. Tomita, S. Nishizawa, K. Sueki, T. Yamaura, Y. Ishikawa, S. Satoh, T. Ushio, K. Koike, A. Uno, Big data assimilation: Real-time 30-second-refresh heavy rain forecast using fugaku during tokyo olympics and paralympics, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '23, New York, NY, USA, Association for Computing Machinery, 2023.
[11] P.L. Houtekamer, X. Deng, H.L. Michell, S.-J. Baek, N. Gagnon, Higher resolution in an operational ensemble Kalman filter, Mon. Weather Rev. 142 (2014) 1143–1162.
[12] C. Schraff, H. Reich, A. Rhodin, A. Schomburg, K. Stephan, A. Periáñez, R. Potthast, Kilometre-scale ensemble data assimilation for the COSMO model (KENDA), Q. J. R. Meteorol. Soc. 142 (2016) 1453–1472.
[13] A. Aksoy, D. Dowell, C. Snyder, A multicase comparative assessment of the ensemble Kalman filter for assimilation of radar observations. Part I: Storm-scale analyses, Mon. Weather Rev. 137 (2009) 1805–1824.
[14] A. Aksoy, D. Dowell, C. Snyder, A multicase comparative assessment of the ensemble Kalman filter for assimilation of radar observations. Part II: Short-range ensemble forecasts, Mon. Weather Rev. 138 (2010) 1273–1292.
[15] N. Gordon, D. Salmond, A. Smith, Novel approach to nonlinear/non-gaussian bayesian state estimation, IEE Proc.-F 140 (2) (1993) 107–113.
[16] A.J. Chorin, X. Tu, Implicit sampling for particle filters, Proc. Natl. Acad. Sci. 106 (41) (2009) 17249–17254.
[17] C. Andrieu, A. Doucet, R. Holenstein, Particle markov chain Monte carlo methods, J. R. Stat. Soc. Ser. B Stat. Methodol. 72 (3) (2010) 269–342.
[18] A.J. Chorin, X. Tu, Implicit sampling for particle filters, Proc. Nat. Acad. Sc. USA 106 (2009) 17249–17254.
[19] K. Kang, V. Maroulas, I. Schizas, F. Bao, Improved distributed particle filters for tracking in a wireless sensor network, Comput. Statist. Data Anal. 117 (2018) 90–108.
[20] M.K. Pitt, N. Shephard, Filtering via simulation: auxiliary particle filters, J. Amer. Statist. Assoc. 94 (446) (1999) 590–599.
[21] C. Snyder, T. Bengtsson, P. Bickel, J. Anderson, Obstacles to high-dimensional particle filtering, Mon. Weather Rev. 136 (2008) 4629–4640.
[22] A. Spantini, R. Baptista, Y. Marzouk, Coupling techniques for nonlinear ensemble filtering, SIAM Rev. 64 (4) (2022) 921–953.
[23] A. Solonen, T. Cui, J. Hakkarainen, Y. Marzouk, On dimension reduction in gaussian filters, Inverse Problems 32 (4) (2016) 045003.
[24] A. Chorin, M. Morzfeld, X. Tu, Implicit particle filters for data assimilation, Commun. Appl. Math. Comput. Sci. 5 (2) (2010) 221–240.
[25] F. Bao, Y. Cao, C. Webster, G. Zhang, A hybrid sparse-grid approach for nonlinear filtering problems based on adaptive-domain of the Zakai equation approximations, SIAM/ASA J. Uncertain. Quantif. 2 (1) (2014) 784–804.
[26] M. Zakai, On the optimal filtering of diffusion processes, Z. Wahrscheinlichkeitstheor. Verwandte Geb. 11 (1969) 230–243.
[27] J. Tödter, P. Kirchgessner, L. Nerger, B. Ahrens, Assessment of a nonlinear ensemble transform filter for high-dimensional data assimilation, Mon. Weather Rev. 144 (2016) 409–427.
[28] J. Poterjoy, R.A. Sobash, J.L. Anderson, Convective-scale data assimilation for the weather research and forecasting model using the local particle filter, Mon. Weather Rev. 145 (2017) 1897–1918.
[29] A. Rojahn, N. Schenk, P.J. van Leeuwen, R. Potthast, Particle filtering and Gaussian mixtures - on a localized mixture coefficients particle filter (LMCPF) for global NWP, J. Meteorol. Soc. Japan 101 (2023) 233–253.
[30] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, in: Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 8780–8794.
[31] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851.
[32] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, in: Advances in Neural Information Processing Systems, vol. 32, 2019.
[33] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, in: International Conference on Learning Representations, 2021.
[34] D. Lu, Y. Liu, Z. Zhang, F. Bao, G. Zhang, A diffusion-based uncertainty quantification method to advance e3sm land model calibration, J. Geophys. Res. Mach. Learn. Comput. 1 (3) (2024) e2024JH000234 e2024JH000234 2024JH000234.
[35] Y. Liu, M. Yang, Z. Zhang, F. Bao, Y. Cao, G. Zhang, Diffusion-model-assisted supervised learning of generative models for density estimation, J. Mach. Learn. Model. Comput. 5 (1) (2024) 25–38.
[36] F. Bao, Z. Zhang, G. Zhang, A score-based nonlinear filter for data assimilation, J. Comput. Phys. 514 (2024) 113207.
[37] Diederik Kingma, Tim Salimans, Ben Poole, Jonathan Ho, Variational diffusion models, Adv. Neural Inf. Process. Syst. 34 (2021) 21696–21707.
[38] P. Vincent, A connection between score matching and denoising autoencoders, Neural Comput. 23 (7) (2011) 1661–1674.
[39] F. Bao, Y. Cao, A. Meir, W. Zhao, A first order scheme for backward doubly stochastic differential equations, SIAM/ASA J. Uncertain. Quantif. 4 (1) (2016) 413–445.
[40] ECMWF, IFS documentation CY48R1 - Part I: Observations. Number 1. ECMWF, 06/2023, 2023.
[41] I.M. Held, R.T. Pierrehumbert, S.T. Garner, K.L. Swanson, Surface quasi-geostrophic dynamics, J. Fluid Mech. 282 (1995) 1–20.