Physics of Fluids

Long short-term memory embedded nudging schemes for nonlinear data assimilation of geophysical flows

Cite as: Phys. Fluids **32**, 076606 (2020); https://doi.org/10.1063/5.0012853
Submitted: 06 May 2020 . Accepted: 24 June 2020 . Published Online: 15 July 2020

Suraj Pawar, Shady E. Ahmed, Omer San, Adil Rasheed, and Ionel M. Navon

Long short-term memory embedded nudging schemes for nonlinear data assimilation of geophysical flows

Cite as: Phys. Fluids 32, 076606 (2020); doi: 10.1063/5.0012853
Submitted: 6 May 2020 • Accepted: 24 June 2020 •
Published Online: 15 July 2020

Suraj Pawar,[1] Shady E. Ahmed,[1] Omer San,[1,a)] Adil Rasheed,[2] and Ionel M. Navon[3]

AFFILIATIONS

[1] School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma 74078, USA
[2] Department of Engineering Cybernetics, Norwegian University of Science and Technology, N-7465 Trondheim, Norway
[3] Department of Scientific Computing, Florida State University, Tallahassee, Florida 32306, USA

[a)] Author to whom correspondence should be addressed: osan@okstate.edu

ABSTRACT

Reduced rank nonlinear filters are increasingly utilized in data assimilation of geophysical flows but often require a set of ensemble forward simulations to estimate forecast covariance. On the other hand, predictor–corrector type nudging approaches are still attractive due to their simplicity of implementation when more complex methods need to be avoided. However, optimal estimate of the nudging gain matrix might be cumbersome. In this paper, we put forth a fully nonintrusive recurrent neural network approach based on a long short-term memory (LSTM) embedding architecture to estimate the nudging term, which plays a role not only to force the state trajectories to the observations but also acts as a stabilizer. Furthermore, our approach relies on the power of archival data, and the trained model can be retrained effectively due to the power of transfer learning in any neural network applications. In order to verify the feasibility of the proposed approach, we perform twin experiments using the Lorenz 96 system. Our results demonstrate that the proposed LSTM nudging approach yields more accurate estimates than both the extended Kalman filter (EKF) and ensemble Kalman filter (EnKF) when only sparse observations are available. With the availability of emerging artificial intelligence friendly and modular hardware technologies and heterogeneous computing platforms, we articulate that our simplistic nudging framework turns out to be computationally more efficient than either the EKF or EnKF approaches.

I. INTRODUCTION

Data assimilation (DA) is a methodology where the observations are utilized to correct the results from a mathematical model to reconstruct the spatiotemporal dynamics of a system.[1–3] DA is used extensively for weather forecasting, where there is a growing number of observations coming from satellites and *in situ* monitoring. Variational and sequential schemes are two of the most widely used approaches in dynamical data assimilation. For the former, DA is formulated as a minimization problem, where the objective function is defined as the discrepancy between real observations and model's predictions based on a given set of initial conditions and parameters. The argument of this minimization problem is the set of model's initial conditions and parameters that need to be tuned to drive the predictions toward the observations. On the other hand, sequential methods usually rely on statistical inference using Bayesian analysis, where the current measurements are used to correct the prior model forecasts to get a better posterior estimate, usually called the analysis in DA terminology.

One of the key limitations of DA methods is that they rely on a forward model whose dynamics is known. For high-dimensional systems such as geophysical flows, standard DA methods suffer from the curse of dimensionality. With the increasing resolution of numerical models, the nonlinearities are likely to become so strong that DA algorithms based on linearization might fail.[4] In recent years, with an explosion of data generated from observations, experimental measurements, and numerical simulations, there is a growing interest in applying data-driven methods along with DA.[5]

Mostly, efforts were focused on using data-driven models *in lieu* of conventional (physics-based) models in order to accelerate the DA computations. Tang, Liu, and Durlofsky[6] developed a surrogate-based model based on convolutional and recurrent neural networks for predicting dynamical subsurface flows and employed it in the DA framework as an emulator to the forward dynamical model. There have been several other studies that demonstrated the potential of data-driven methods in the accurate prediction of complex physical systems such as flooding,[7] global atmospheric model,[8] quasi-geostrophic flows,[9] chaotic systems,[10,11] soil water dynamics,[12] and tsunami modeling.[13] Recent works have also drawn ideas to synthesize DA with reduced order models.[14–23] Bocquet *et al.*[24] proposed a hybrid framework by combining DA and machine learning (ML) to estimate the model, the state trajectory, and model error statistics for high-dimensional chaotic systems from partial and noisy observations. Brajard *et al.*[25] proposed an algorithm where neural networks provide a surrogate forward model to DA, and DA provides a time series of complete states to train the neural network. They illustrated the convergence of the proposed algorithm for the Lorenz 96 system and achieved the accurate forecasts up to two Lyapunov time units.

Correspondingly, ML tools can also benefit from DA algorithms. Abarbanel, Rozdeba, and Shirman[26] offer a perspective on the equivalence between ML and statistical data assimilation and discuss how methods developed in DA can be potentially useful for ML. Bocquet *et al.*[27] proposed DA as a learning tool to infer ordinary differential equations for dynamical systems solely from noisy data and showed its connection with deep learning methods. Pérez-Ortiz *et al.*[28] showed that the long short-term memory (LSTM) network can be trained efficiently with better generalization using the decoupled extended Kalman filter (EKF).[29]

As an extension to the current efforts of using ML tools in DA context, we propose a modular neural-network based DA framework. Specifically, we utilize ML to achieve the fusion between the model's estimates and noisy observations to provide more accurate predictions, rather than using ML as a facilitator to just accelerate the existing DA algorithms. To accomplish this, we train a LSTM neural network to "nudge" model's forecast given a set of sparse observations. Nudging is a relatively simple DA approach that uses the forecast error, defined as the difference between model predictions and measurements, to constrain and correct the model evolution. Nudging was introduced by Anthes[30] for the initialization of hurricane models from real observational data. In nudging methods, the state analysis is approximated as a linear superposition between its model forecast and forecast error. Despite its conceptual simplicity, nudging schemes often require *ad hoc* approximation of the nudging (or weighting) matrix. In our framework, we relax this linear superposition assumption and avoid those *ad hoc* approximations by training an LSTM neural network to *nonlinearly* blend model's forecast and sparse observations.

We demonstrate and test the proposed LSTM-DA framework using the Lorenz 96 system as a benchmark problem in geophysical science applications. We illustrate the success of LSTM-DA using different sets of observations with varying levels of noise and sparsity. In particular, we consider combinations between data-rich, data-deficient, observation-rich, and observation-deficient settings. We also compare our results against some of the common DA techniques. Specifically, we discuss the results of the extended Kalman filter (EKF), ensemble Kalman filter (EnKF), deterministic ensemble Kalman filter (DEnKF), and a simple forward nudging method. Our LSTM-DA framework can be considered very much similar to the methodology proposed by Zhu *et al.*[31] in which the fully connected neural network was used to learn the uncertainty in the mathematical model arising from linearization, discretization, and model reduction. The difference in our proposed framework is that we employ the LSTM neural network to learn the nudging correction term in order to cure the discrepancy between prior predictions and measurements that might arise due to inaccurate initial conditions, boundary conditions, or model parameters.

The rest of the manuscript is outlined here. In Sec. II, we discuss the nudging method as a simple alternative to nonlinear filters, which is then extended as a base for our proposed LSTM-DA framework in Sec. II B. We define the DA setup using Lorenz 96 system in Sec. III. After that, we provide our results in Sec. IV as well as relevant discussions and comparisons using different sets of historical data and observations. Finally, we draw our conclusions as well as the limitations and potential extensions of the present study in Sec. V. In Appendix A, we describe three of the most common nonlinear filtering techniques as benchmarks to compare our framework against. In particular, we briefly outline the extended Kalman filter, which is a first-order adaptation of the standard Kalman filter to deal with nonlinear models. We then introduce the ensemble Kalman filter and its deterministic version as reduced rank variants of nonlinear filters.

II. DATA ASSIMILATION METHODOLOGY

The central goal of DA is to extract the information from observational data to correct dynamical models and improve their prediction. There are different approaches such as variational methods such as 4D-Var and stochastic methods such as ensemble filters that are widely used in DA. Several textbooks on data assimilation offer academic explanations and discussion on these methods.[1–3,32–34]

For demonstration, we consider the dynamical system whose evolution is governed by

$$\mathbf{x}_{k+1} = \mathbf{M}(\mathbf{x}_k) + \mathbf{w}_{k+1}, \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state of the dynamical system at discrete time t_k and $\mathbf{M} : \mathbb{R}^n \to \mathbb{R}^n$ is the nonlinear model operator that defines the temporal evolution of the system. The term $\mathbf{w}_{k+1}$ denotes the model noise that takes into account the mathematical model error, numerical approximations, and the boundary conditions. In our study, we assume that the model noise is drawn from a multivariate normal distribution with zero mean and a covariance matrix $\mathbf{Q}_k$, i.e., $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$.

Let $\mathbf{z}_k \in \mathbb{R}^m$ be observations of the state vector obtained through noisy measurements procedure as follows:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \tag{2}$$

where $h(\cdot)$ is a nonlinear function that maps $\mathbb{R}^n \to \mathbb{R}^m$, also known as the observational operator defining a map between state space and measurement space, and $\mathbf{v}_k \in \mathbb{R}^m$ is the measurement noise.

We assume that the measurement noise is a white Gaussian noise with zero mean and the covariance matrix $\mathbf{R}_k$, i.e., $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Furthermore, we assume that the noise vectors $\mathbf{w}_k$ and $\mathbf{v}_k$ at two different time steps are uncorrelated, which is a common assumption in data assimilation problems. In this section, first, we review nudging-based data assimilation methods and then present our approach of long short-term memory nudging method that can be used for a variety of problems. Along with nudging methods, for comparison purposes, we also discuss briefly different nonlinear filtering algorithms for sequential data assimilation in Appendix A.

A. Nudging dynamics

Nudging is a data assimilation method that was introduced by Anthes[30] for the initialization of hurricane models from real observational data. Contrary to variational and sequential data assimilation methods that minimize the cost function based on the error between model forecast and observations, nudging methods utilize the forecast error as a constraint to the model evolution equation. The evolution of the dynamical system based on nudging methods can be written as

$$\mathbf{x}_{k+1} = \mathbf{M}(\mathbf{x}_k) + G_k \mathbf{e}_k, \tag{3}$$

where $G_k \in \mathbb{R}^{n\times m}$ is called the time varying nudging coefficient matrix. The forecast error $\mathbf{e}_k$ in Eq. (3) is computed as follows:

$$\mathbf{e}_k = \mathbf{z}_k - h(\mathbf{x}_k). \tag{4}$$

The correction term in Eq. (3) is proportional to $\mathbf{e}_k \in \mathbb{R}^m$ (i.e., in the observation space), and therefore, this form of nudging is called observation nudging. The literature on nudging can be divided into different classes/versions based on how the nudging coefficient matrix is computed. Lakshmivarahan and Lewis[35] offer an overview of the theoretical aspect of nudging methods and present promising directions of research on the nudging process of dynamic data assimilation. Nudging methods have been applied for different applications such as the forecast of Indian Monsoon,[36] diagnostic studies of mesoscale processes in mid-latitude weather systems,[37,38] and operational predictions in meteorology and oceanography.[39,40] Nudging methods have also been shown to increase the long-term accuracy of time-dependent partial differential equations.[41]

Zou, Navon, and Le Dimet[42] proposed a parameter-estimation approach to obtain optimal nudging coefficients using a variational data assimilation method. They estimated the parameters of the nudging coefficient matrix by solving the constrained minimization problem utilizing the Lagrangian formulation. Their cost function consists of two parts: the first part corresponds to the misfit between the model results and observations and the second part was related to keeping the new estimate of nudging coefficients close to its prior estimate. They enforced the nudged dynamics given in Eq. (3) as a strong constraint to the optimization problem. They demonstrated the performance of the optimal nudging method for an adiabatic version of the National Meteorological Center (NMC) spectral model with 18 vertical layers. Vidard, Le Dimet, and Piacentini[43] introduced another approach to estimate the optimal nudging coefficient matrix using the Kalman filter. They illustrated the proposed approach for the Burgers equation and shallow-water equations in a twin experiment framework and showed noticeable improvement in the prediction. Auroux and Blum[44] introduced the back and forth nudging (BFN) algorithm where the set of observations are incorporated into the model by running it forward in time, starting with some initial condition. After the forward run is completed, the model is again run backward in time, starting from the final state obtained by the standard nudging method. During the backward integration, the use of opposite sign for the nudging term to forward integration makes this algorithm numerically stable. This procedure is repeated in the BFN algorithm until the convergence. Therefore, it helps reduce forecast error on a finite time window. One of the advantages of the BFN algorithm is that it does not require the linearization of nonlinear equations in order to have the adjoint model or to solve any optimization problem. The BFN algorithm was tested for the Lorenz 63 model and for the quasi-geostrophic model in the presence of perfect and noisy observations[45] and showed comparable prediction skills to the 4D-VAR algorithm.

Spectral nudging is another technique where the nudging term is added in the spectral domain with maximum efficiency for large scales and no effect for small scales.[46] This method has been successfully applied to force large-scale atmospheric states from global climate models onto a regional climate model.[47–51] The main idea in spectral nudging is that small-scale details for weather prediction are governed by the interplay between larger-scale atmospheric flow and geographic features such as mountains and land–sea distribution. It is computationally impractical to resolve these small scales in global climate models. Therefore, spectral nudging is applied to match overlapping scales in global and regional climate models by forcing the regional model to behave as a global model. Spectral nudging method has also been applied for inferring flow parameters for turbulent flows,[52] and for three-dimensional homogeneous isotropic turbulence.[53] There are also nudging methods that make use of the present and past observations in the formulation of the forcing term to drive the model evolution toward observation.[54,55] An *et al.*[56] used the time delayed nudging method[54] for estimating the state of the geophysical system from sparse observation data.

B. Long short-term memory nudging

With the huge amount of data generated from high-fidelity numerical simulations, non-invasive experimental techniques such as particle image velocimetry (PIV), and satellite data, there is a growing interest in using machine learning for data assimilation.[26,57] One of the difficulties in weather and climate prediction is that atmospheric flows are multiscale in nature and their dynamics are typically chaotic. Several data-driven algorithms can address these challenges. The recurrent neural networks (RNNs) are particularly attractive for complex dynamical systems due to their ability to capture temporal dependencies and to take state history into account for future state prediction. One of the problems with RNN is that the gradient vanishes during the learning procedure. Long short-term memory[58] is a type of RNN that alleviates this issue of vanishing gradient[59] by employing cell architecture that remembers or forgets information.

There is a rich literature on the application of LSTM for modeling chaotic dynamical systems. Vlachas *et al.*[11] proposed a data-driven forecasting method for the high-dimensional chaotic system by modeling their temporal dynamics on reduced order space using LSTM. They also integrated the LSTM with a mean stochastic model to ensure convergence and demonstrated its improved prediction performance compared to the Gaussian process. In Wan *et al.*,[60] the LSTM was employed to learn the mismatch between the imperfect Galerkin based reduced order model and the actual dynamics projected onto the reduced order space. They showed the improved performance of the proposed framework for the prediction of extreme events. Jia *et al.*[61] introduced the physics-guided RNN that combines the LSTM and physics-based model to model the dynamics of temperature in lakes. They utilized a physics-based regularization as a penalty term to the optimization cost function to enforce physics into the training. Apart from LSTM, other machine learning algorithms such as reservoir computing have been used for modeling chaotic dynamical systems[10,62] and residual network for predicting dynamical system evolution.[63,64] In a recent study, Vlachas *et al.*[65] investigated the performance of LSTM trained with backpropagation through time and reservoir computing for long term forecasting of chaotic dynamical systems.

Zhang *et al.*[66] presented an LSTM based Kalman filter for data assimilation of two-dimensional spatio-temporal varying depth of ocean field for underwater glider path planning. In their study, the temporal evolution of the spatial basis function was modeled using LSTM. They train the LSTM network to predict the future temporal coefficients based on the historical states of these coefficients. Jin *et al.*[67] utilized LSTM to perform observation bias correction for data assimilation of dust storm prediction. They showed that with the LSTM model for bias corrections, existing measurements are used precisely and that improves the resulting prediction accuracy. In the work by Loh, Omrani, and van der Linden,[68] the LSTM was deployed as a prediction model for their EnKF approach to achieve real-time production forecast in natural gas wells. Xingjian *et al.*[69] proposed a convolutional LSTM framework to predict the rainfall intensity over a short period of time and illustrated its ability to capture more improved correlation than the existing methods.

Motivated by the previous successes of employing neural networks for making better predictions in geophysical applications, in the present study, we introduce an LSTM nudging scheme. The LSTM network is trained to learn the correction term based on the background state of the system and observations. To train the LSTM network, we initialize the state of the system for different training sets from prior distribution of the true initial state. This step is similar to initializing different ensemble members in the case of the EnKF algorithm. We then evolve the system with erroneous initial conditions and compute the correction term at all observation points as follows:

$$\boldsymbol{\epsilon}_k(i) = \widehat{\mathbf{X}}_k(i) - \bar{\mathbf{x}}_k. \tag{5}$$

The input features to the LSTM network (denoted by $\mathcal{X}_k$) consist of full state of the system (from erroneous initial conditions) and current observations, i.e., $\mathcal{X}_k = \{\widehat{\mathbf{X}}_k(i); \mathbf{z}_k\} \in \mathbb{R}^{n+m}$, where m is the number of observations. Based on these input features, the LSTM is trained to learn the correction term for all states, i.e., the output of the LSTM is $\mathcal{Y}_k = \{\boldsymbol{\epsilon}_k(i)\} \in \mathbb{R}^n$. The LSTM network is capable of capturing the temporal dependencies and utilizing it to forecast the system's future state. Therefore, we can also train the LSTM network by including the temporal history of the system's states and observations as input features. For further details on incorporating the temporal history of the system's state into training, refer to the work of Rahman *et al.*[9] The procedure for the training phase of the LSTM nudging scheme is outlined in Algorithm 1.

We adopt the predictor–corrector approach during online deployment. Since the LSTM network is trained to learn the mapping from the state of the system generated with the erroneous initial condition, we start with two systems. We use the superscript E to denote the system with the erroneous initial condition and C to denote the evolution of the system whose state is corrected at each observation point. The procedure for online deployment is reported in Algorithm 2. We start with initializing two systems with the same initial condition based on some educated guess. The forward dynamics of erroneous and corrected systems are evolved simultaneously as

$$\mathbf{X}^E_{k+1} = \mathbf{M}(\mathbf{X}^E_k), \tag{6}$$

$$\mathbf{X}^C_{k+1} = \mathbf{M}(\mathbf{X}^C_k). \tag{7}$$

Once the observations are available, we determine the correction term using the trained LSTM network as follows:

$$\boldsymbol{\epsilon}_{k+1} = \mathcal{M}(\{\mathbf{X}^E_{k+1}; \mathbf{z}_{k+1}\}). \tag{8}$$

This correction is for the state of the system generated with the erroneous initial condition. Therefore, the correction is added to the erroneous system's state at that time and assigned to the corrected system using the following superposition:

$$\mathbf{X}^C_{k+1} = \mathbf{X}^E_{k+1} + \boldsymbol{\epsilon}_{k+1}. \tag{9}$$

The schematic of the LSTM nudging framework is depicted in Fig. 1. We highlight some of the features of the LSTM nudging framework here. The LSTM nudging framework is highly modular, and it can be implemented with other types of neural network

ALGORITHM 1. LSTM Nudging (training phase).

```
1: Initialize the state of the system for different training sets from
   prior distribution of $\mathbf{x}_0 \sim N(\mathbf{m}_0, \mathbf{P}_0)$.
       $\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i)$,
   where $\mathbf{y}_0(i) \sim N(0, \mathbf{P}_0)$.
2: Integrate the dynamical system and store the system's state at all
   observation points, i.e., at time $t_1, \ldots, t_k$.
3: Compute the correction term at time $t_1, \ldots, t_k$ with respect to the
   true state of the system as follows:
       $\boldsymbol{\epsilon}_k(i) = \widehat{\mathbf{X}}_k(i) - \bar{\mathbf{x}}_k$,
   where $\bar{\mathbf{x}}_k$ is the true state of the system at $t_k$.
4: Each sample of the input training matrix $\mathcal{X}_k$ and corresponding
   output data matrix $\mathcal{Y}_k$ is constructed as follows:
       $\mathcal{X}_k = \{\widehat{\mathbf{X}}_k(i); \mathbf{z}_k\} \in \mathbb{R}^{n+m}$,
       $\mathcal{Y}_k = \{\boldsymbol{\epsilon}_k(i)\} \in \mathbb{R}^n$,
   where $m$ is the number of observations.
5: Train the LSTM model to learn the mapping from input to output
       $\mathcal{M} : \mathcal{X}_k \Rightarrow \mathcal{Y}_k$.
```

ALGORITHM 2. LSTM nudging (online deployment).

```
1: Initialize the state of the system for two members with an educated
   guess for an initial condition.
       $\mathbf{X}_0^E = \mathbf{x}_0,$
       $\mathbf{X}_0^C = \mathbf{x}_0.$
2: For $k = 0, 1, \ldots,$ proceed as follows:
   • Forecast step: integrate the state estimate for two systems from
     time $t_k$ to $t_{k+1}$ as follows:
       $\mathbf{X}_{k+1}^E = \mathbf{M}(\mathbf{X}_k^E),$
       $\mathbf{X}_{k+1}^C = \mathbf{M}(\mathbf{X}_k^C).$
   • Data assimilation step: once the observations are available at
     time $t_{k+1}$, they are used to determine the correction term with
     the trained LSTM network and correct the state estimate as
     follows:
       $\boldsymbol{\epsilon}_{k+1} = \mathcal{M}(\{\mathbf{X}_{k+1}^E; \mathbf{z}_{k+1}\}),$
       $\mathbf{X}_{k+1}^C = \mathbf{X}_{k+1}^E + \boldsymbol{\epsilon}_{k+1}.$
```

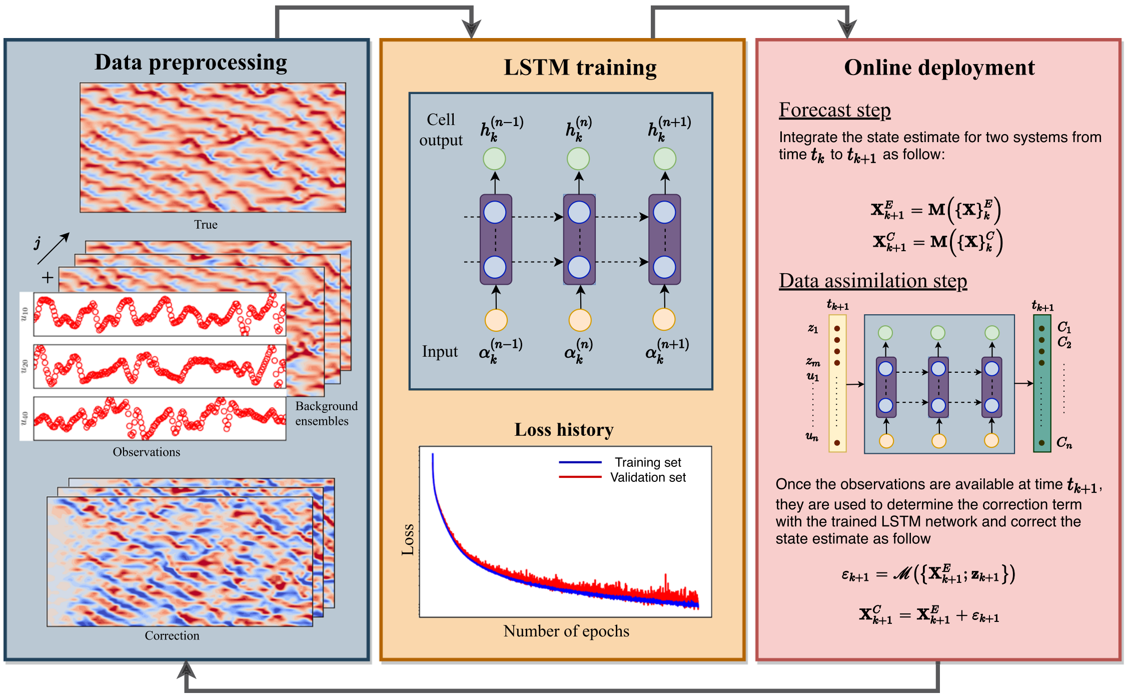
architectures also based on the size or type of problems. For example, convolutional autoencoders are gaining popularity to find the nonlinear basis functions of complex physical systems and they are complemented with the LSTM network for learning the latent-space dynamics.[70–76] The LSTM nudging framework can be easily applied to high dimensional systems, where convolutional autoencoders are employed for dimensionality reduction and the LSTM is trained to learn the nudging dynamics in latent-space instead of high-dimensional space. Novel neural network architectures such as generative adversarial networks (GANs)[77,78] can also be applied to learn the nudging dynamics. Another feature of the LSTM nudging scheme is that once the network is trained with the archival or background data, it can be retrained efficiently with transfer learning as the new observation data become available. Therefore, training the LSTM network for the first time is the only computationally heavier part of the LSTM nudging scheme. Our main goal in this study is to illustrate that neural networks can be effectively trained to provide accurate and stable nudging dynamics.

III. DATA ASSIMILATION PROBLEM SETUP

In this section, we describe the Lorenz 96 model proposed by Lorenz,[79] which is commonly used as a prototypical test case in data assimilation. This model describes the temporal evolution of atmospheric quantity discretized spatially over a single latitude circle. The system of ordinary differential equations governing the Lorenz 96 model can be written as

$$\frac{du_i}{dt} = u_{i-1}(u_{i+1} - u_{i-2}) - u_i + F \tag{10}$$

for $i \in \{1, 2, \ldots, n\}$. The first term on the right-hand side of Eq. (10) is the nonlinear advection term, the second term presents an



Integrate the state estimate for two systems from time t_k to t_{k+1} as follow:

$\mathbf{X}_{k+1}^E = \mathbf{M}(\{\mathbf{X}\}_k^E)$

$\mathbf{X}_{k+1}^C = \mathbf{M}(\{\mathbf{X}\}_k^C)$

Once the observations are available at time t_{k+1}, they are used to determine the correction term with the trained LSTM network and correct the state estimate as follow

$\epsilon_{k+1} = \mathcal{M}(\{\mathbf{X}_{k+1}^E; \mathbf{z}_{k+1}\})$

$\mathbf{X}_{k+1}^C = \mathbf{X}_{k+1}^E + \epsilon_{k+1}$

FIG. 1. Overview of the LSTM-DA framework. The LSTM-DA framework consists of three main steps: data preprocessing, training the neural network, and the deployment of trained network.

internal dissipation, and the third term presents an external forcing. We use $n = 40$ and $F = 10$ in our analysis. The forcing $F = 10$ is chosen as a strongly supercritical value to make the system sufficiently chaotic[80] and have a similar doubling time that is more compatible with larger models.[81] We apply the periodic boundary conditions at ghost points, i.e., $u_0 = u_n$, $u_{-1} = u_{n-1}$ and $u_{n+1} = u_1$.

We use the fourth-order Runge–Kutta scheme for time integration with a time step of $\Delta t = 0.005$. To generate a physical initial condition for the forward run, we start with an equilibrium condition at time $t = -5$. The equilibrium condition for the model is $u_i = F$ for $i \in \{1, 2, \ldots, n\}$. We introduce a very little perturbation to the equilibrium state for the state u_{20}, i.e., we set $u_{20} = F + 0.01$ to generate chaotic dynamics and then do the time integration up to $t = 0$. Once the true initial condition is generated, we run the forward solver up to time $t = 10$.

The twin experiment is one of the most commonly used methods to validate any data assimilation algorithm before it can be applied to real-life applications.[82] For twin experiments, we first generate the n-dimensional data for the Lorenz 96 model and select m observations. These observations are obtained by adding some noise to the true state of the system to take experimental uncertainties and measurement error into account. The observations are also sparse in time, meaning that the time interval between two observations can be different from the time step of the model. For our twin experiments, we assume that observations are recorded at every tenth time step of the model. Therefore, the time difference between two observations is $\delta t = 0.05$. The analysis time step $\delta t = 0.05$ is representative of 6 h of a data assimilation cycle of global meteorological models. The accurate estimation of the full state of the system depends upon the number of observations that are assimilated by the model.[83] We assume that observation locations are constant throughout the time unlike asynchronous observations where they can be rotated.[84] We compare the performance of traditional data assimilation algorithms and the proposed LSTM nudging algorithm for three sets of observations. The first set of observations is very sparse with only 10% of the full state of the system (i.e., $m = 4$), utilizing observations for states $[u_{10}, u_{20}, u_{30}, u_{40}] \in R^4$. In a second set of observations ($m = 8$), we employ observations at $[u_5, u_{10}, \ldots, u_{40}] \in R^8$ for the assimilation. The third set of observations consists of 50% of the full state of the system ($m = 20$), i.e., observations at states $[u_2, u_4, \ldots, u_{40}] \in R^{20}$ for the assimilation.

IV. RESULTS

In this section, we describe the results of numerical experiments with the Lorenz 96 model using algorithms discussed in Secs. II, II A, and II B. We assume that our model is perfect for all numerical experiments except for the EKF and EnKF algorithms. For these two algorithms, it is found that an introduction of small uncertainty in the model provides more accurate predictions than the assumption of a perfect model. For the aforementioned two algorithms, we assume that the model noise is drawn from the Gaussian distribution with zero mean and variance 1×10^{-4}. The observations are created by adding random noise from Gaussian distribution with zero mean and variance 1×10^{-2} to the true state of the system. The erroneous initial condition is generated by adding a noise from Gaussian distribution of zero mean and 1×10^{-2} variance to the true initial condition. To ensure a fair comparison between EnKF and DEnKF, we use an equal number of ensembles in both algorithms. For the comparison, we plot time evolution of states u_{10}, u_{21}, u_{39} and also the full state trajectory of the Lorenz 96 model. We use black lines to denote true states, blue dashed lines to denote states with the erroneous initial condition, and green dashed-dotted lines for assimilated states. The observations for the state u_{10} are shown with red circles in all the time series plots.

In Fig. 2, we present the time evolution of the selected states for three different numbers of observations included in the assimilation of the EKF algorithm. There is an excellent agreement between

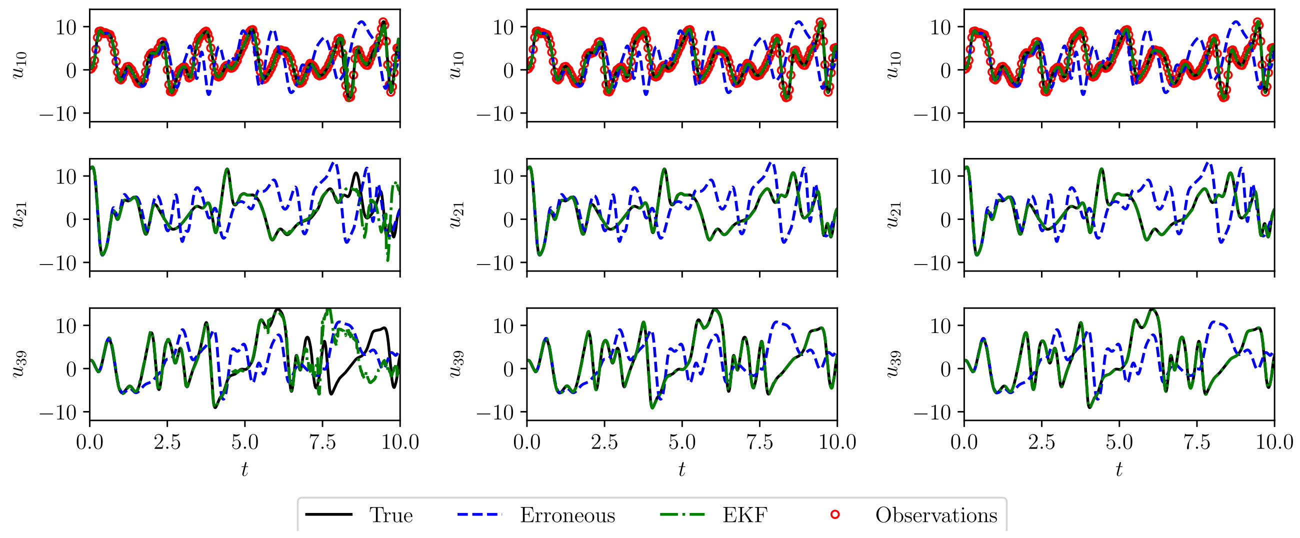


FIG. 2. Selected trajectories of the Lorenz 96 model with the analysis performed by the extended Kalman filter (EKF) using observations from $m = 4$ (left), $m = 8$ (middle), and $m = 20$ (right) state variables at every ten time steps.

true and assimilated states u_{21} and u_{39} when more than 20% observations are utilized for assimilation. We also provide the full state trajectory of the Lorenz 96 model in Fig. 3. The results obtained clearly show that the EKF algorithm can determine the correct state trajectory with more than 20% observations, i.e., for $m \geq 8$. We observe a discrepancy in prediction after $t \sim 7$ when only four observations are used in the assimilation step. Figure 4 shows the time evolution of the selected states predicted by using the EnKF algorithm with $N = 40$ ensemble members. We notice some discrepancy between the true and predicted states with $m = 12$ observations after $t \sim 7.5$. If we compare the full state trajectory prediction by the EnKF algorithm in Fig. 5, we can conclude that there is almost a perfect match between true and assimilated states with more than eight observations. Since the EnKF algorithm is based on the Monte Carlo framework, its accuracy can be improved by applying an increased number of ensembles. The typical number of ensembles is $O(100)$ for high-dimensional systems,[85–87] and we often use localization approaches. Considering that the Lorenz 96 model is a lower-dimensional system with $n = 40$ states, we apply only 40 ensemble members without using any localization kernel. If we consider the computational cost of the EKF algorithm, the major bottleneck is the propagation of the error covariance matrix as given in Eq. (A5). The computational overhead of the EnKF algorithm goes up with an increase in the number of ensembles. However, with the advancement in parallel algorithms and high-performance computing, ensemble Kalman filter algorithms are particularly attractive data assimilation of complex physical systems.[88]

As we observed in Fig. 5, the use of virtual observations in the EnKF algorithm leads to suboptimal performance when fewer observations are used for assimilation with a small number of ensembles. The EnKF data solution converges toward a true solution with an increase in the number of ensembles. The DEnKF algorithm is the deterministic version of the EnKF algorithm where no virtual observations are used. Instead of using virtual observations, the DEnKF algorithm updates the ensemble mean with the standard analysis equation, and ensemble anomalies are updated separately with half the Kalman gain in the same equation.[89] In Fig. 6, we illustrate the time evolution of selected states for different percentages of observations used in the assimilation step. We notice that even with just four observations, the DEnKF algorithm is able to correct the erroneous states up to the final time $t = 10$. From the results depicted in Fig. 7, we can deduce that the DEnKF algorithm leads to a better performance than the EnKF algorithm when the number of observations is smaller.

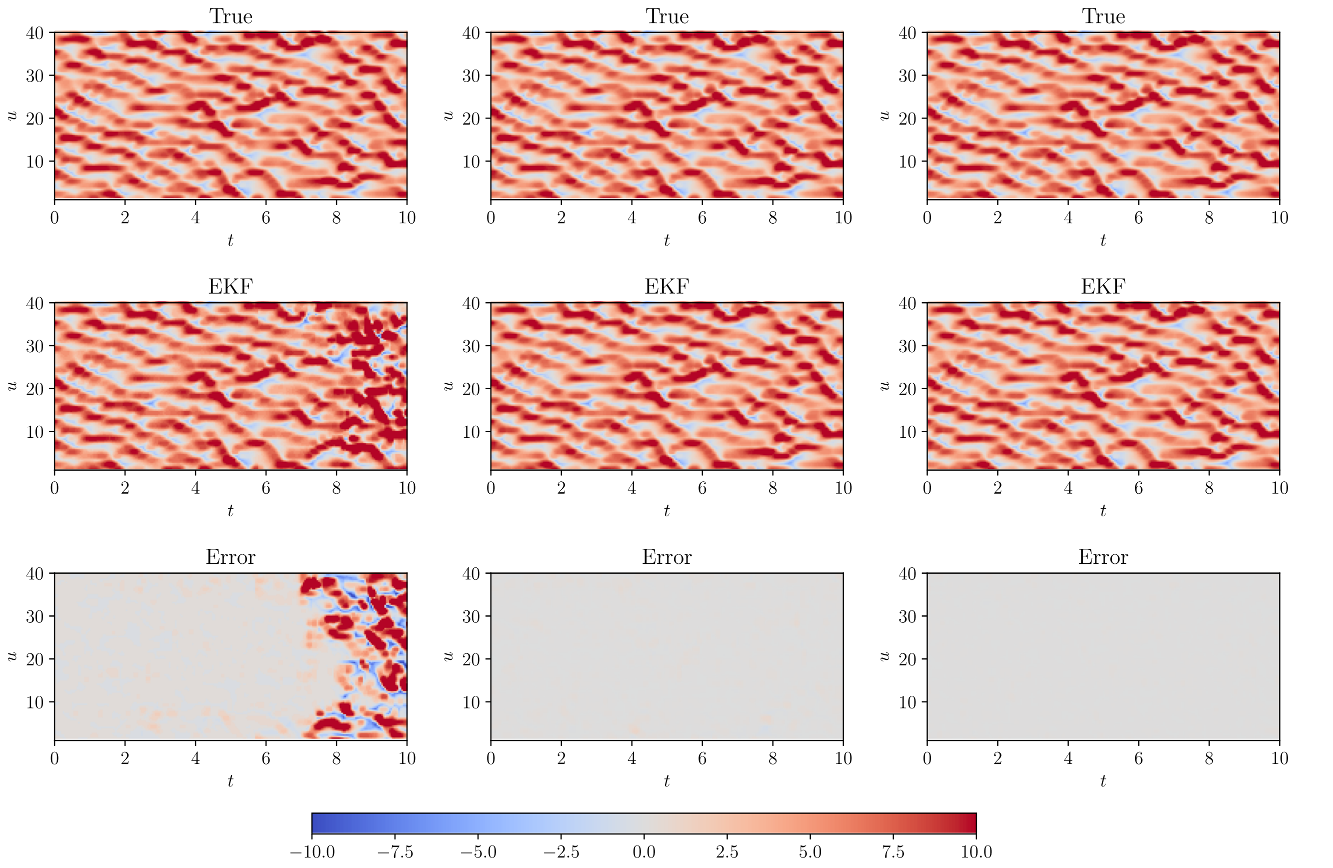


FIG. 3. Full state trajectory of the Lorenz 96 model with the analysis performed by the extended Kalman filter (EKF) using observations from $m = 4$ (left), $m = 8$ (middle), and $m = 20$ (right) state variables at every ten time steps.

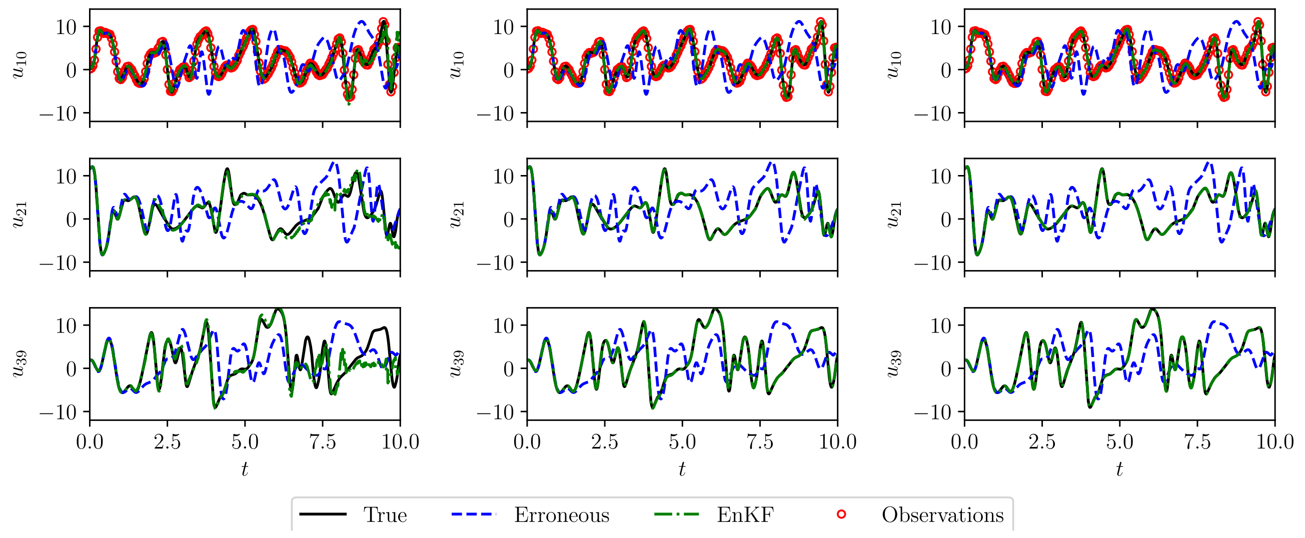


FIG. 4. Selected trajectories of the Lorenz 96 model with the analysis performed by the ensemble Kalman filter (EnKF) with N = 40 member ensemble using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

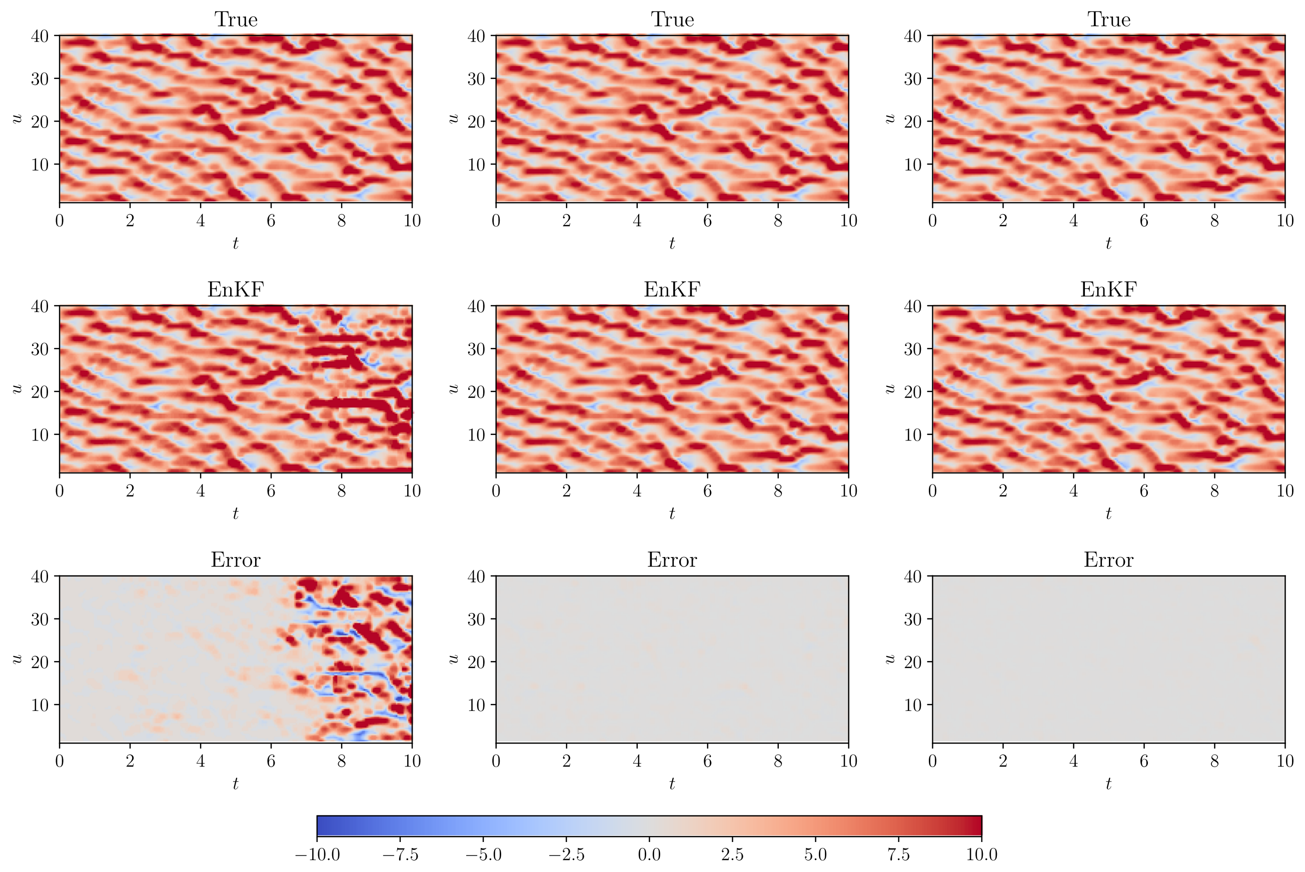


FIG. 5. Full state trajectory of the Lorenz 96 model with the analysis performed by the ensemble Kalman filter (EnKF) with N = 40 member ensemble using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

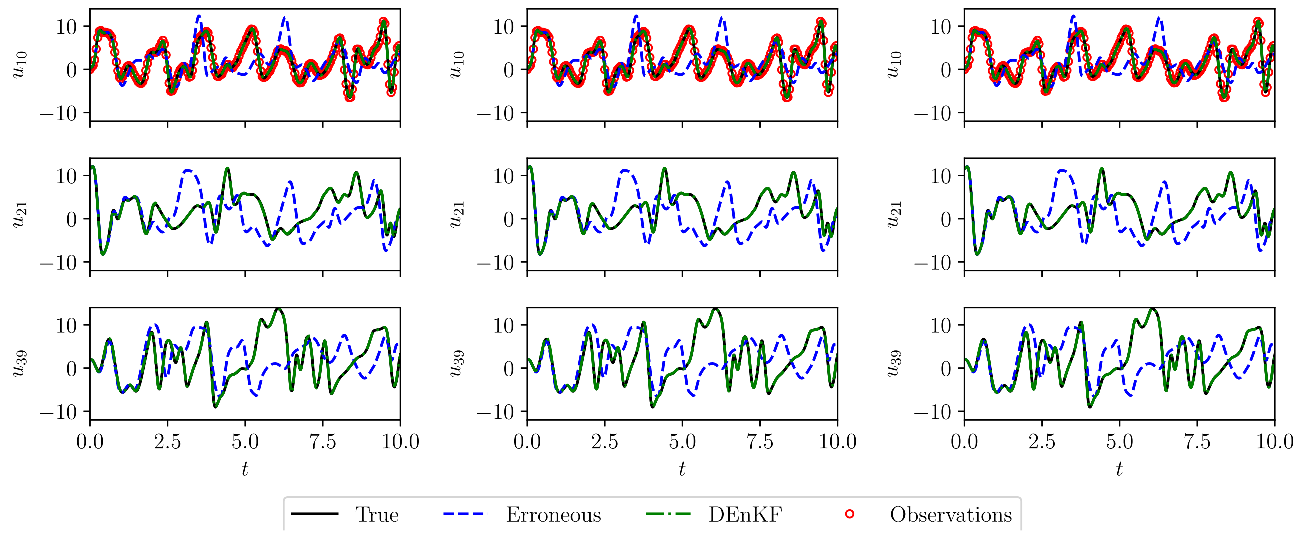


FIG. 6. Selected trajectories of the Lorenz 96 model with the analysis performed by the deterministic ensemble Kalman filter (DEnKF) with the N = 40 member ensemble using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

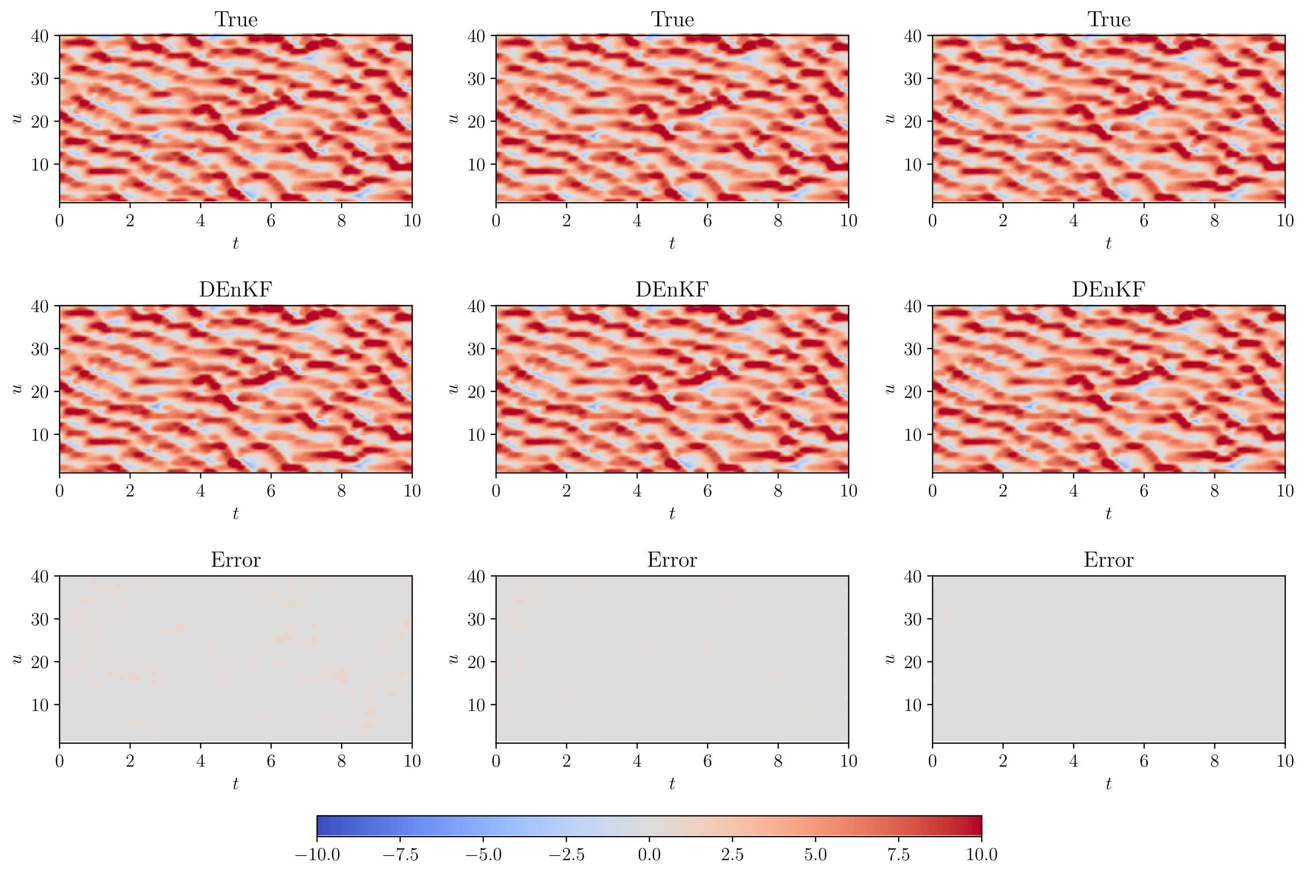


FIG. 7. Full state trajectory of the Lorenz 96 model with the analysis performed by the deterministic ensemble Kalman filter (DEnKF) with the N = 40 member ensemble using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

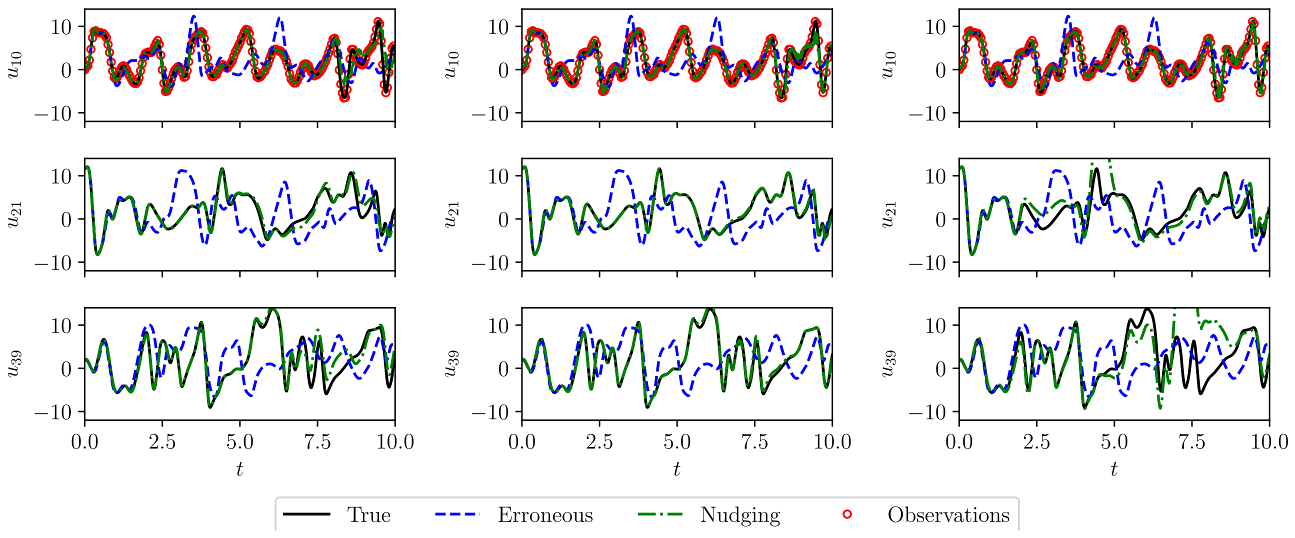


FIG. 8. Selected trajectories of the Lorenz 96 model with the analysis performed by the forward nudging with m = 20 observations state variables at every ten time steps for τ = 50Δt (left), τ = 100Δt (middle), and τ = 200Δt (right).

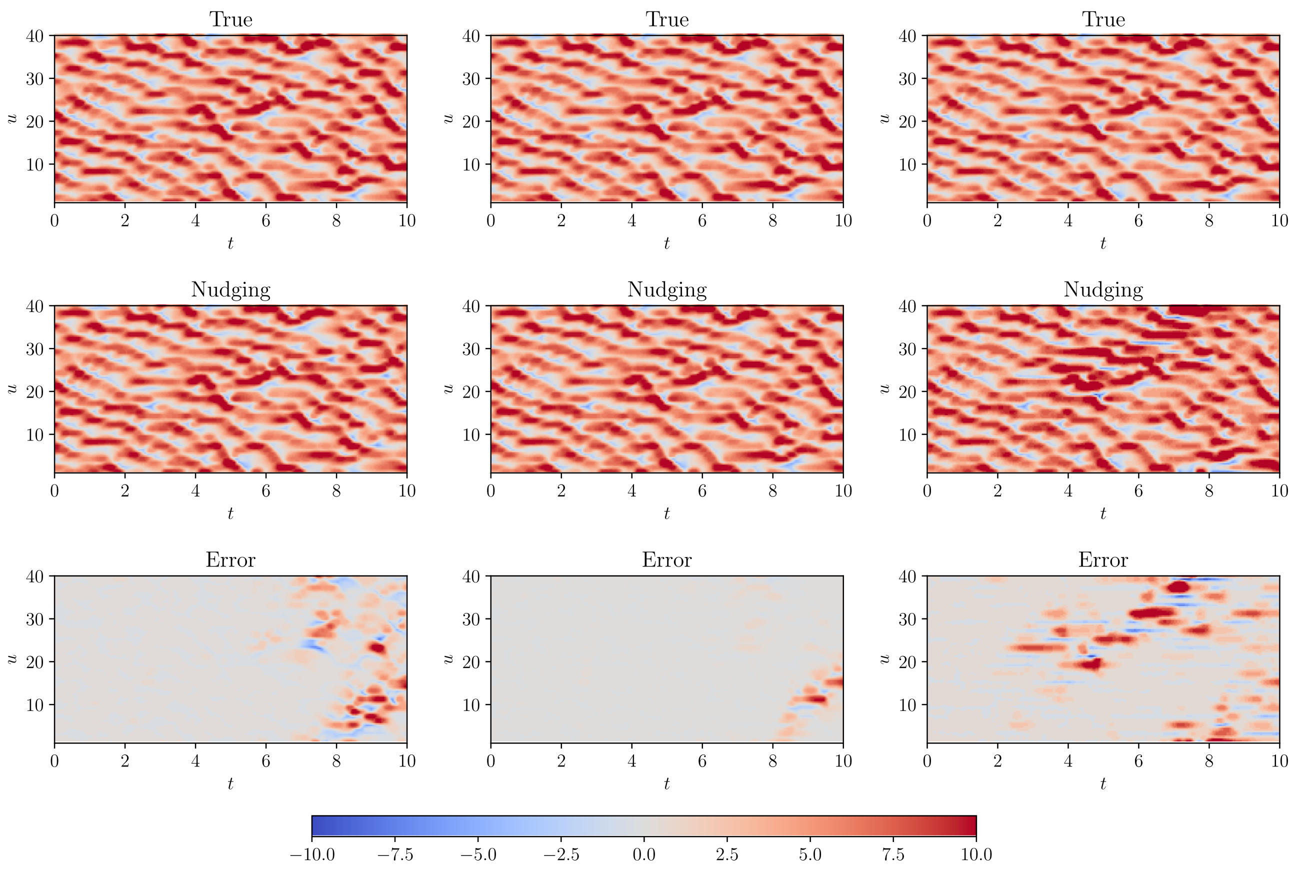


FIG. 9. Full state trajectory of the Lorenz 96 model with the analysis performed by the forward nudging with m = 20 observations state variables at every ten time steps for τ = 50Δt (left), τ = 100Δt (middle), and τ = 200Δt (right).

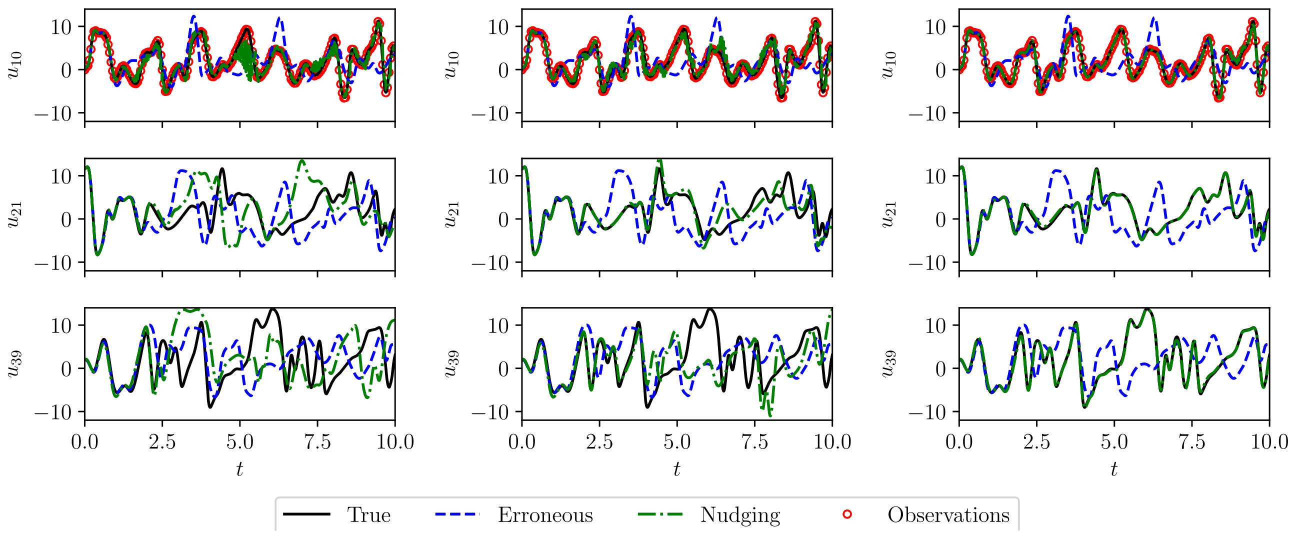


FIG. 10. Selected trajectories of the Lorenz 96 model with the analysis performed by the forward nudging with τ = 150Δt using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

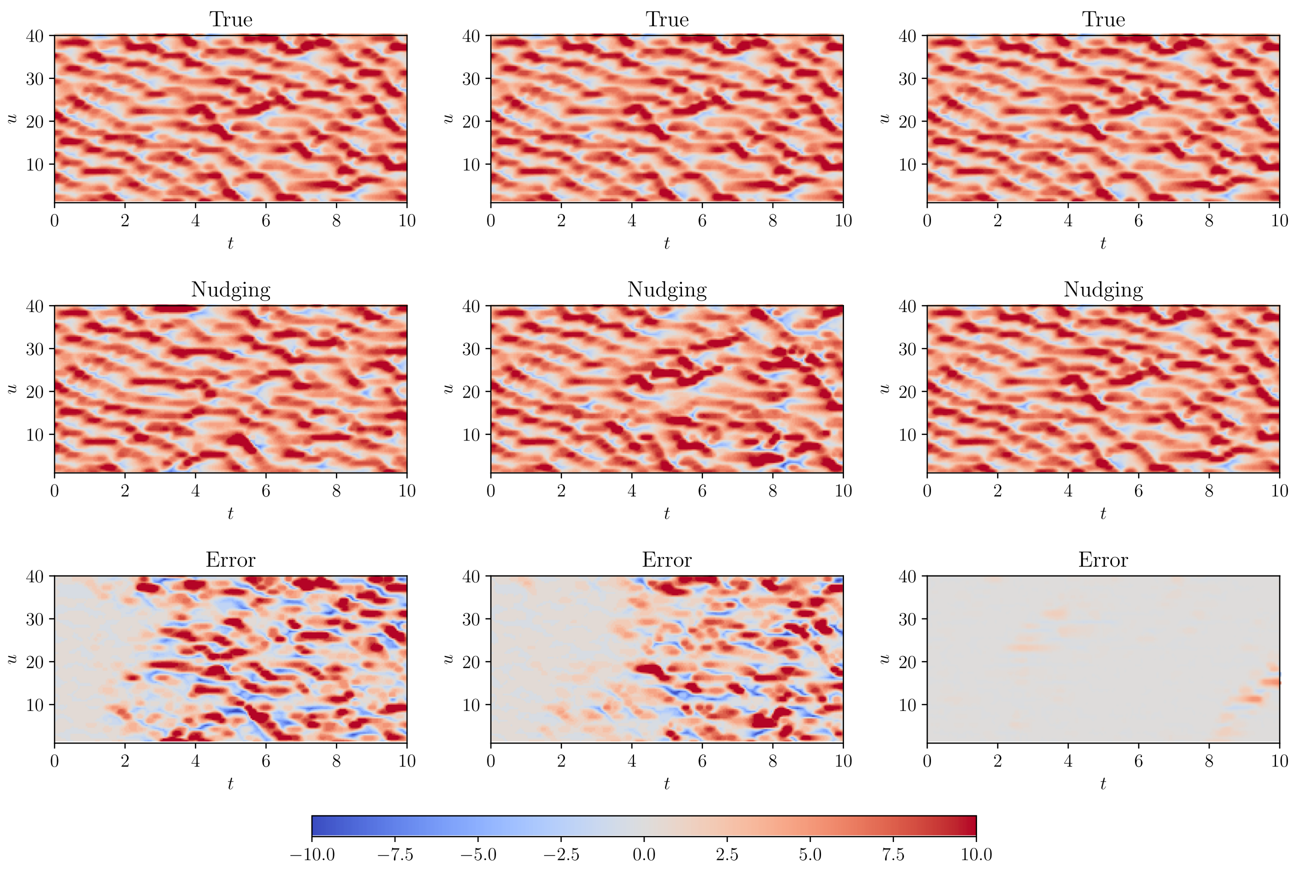


FIG. 11. Full state trajectory of the Lorenz 96 model with the analysis performed by the forward nudging with τ = 150Δt using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

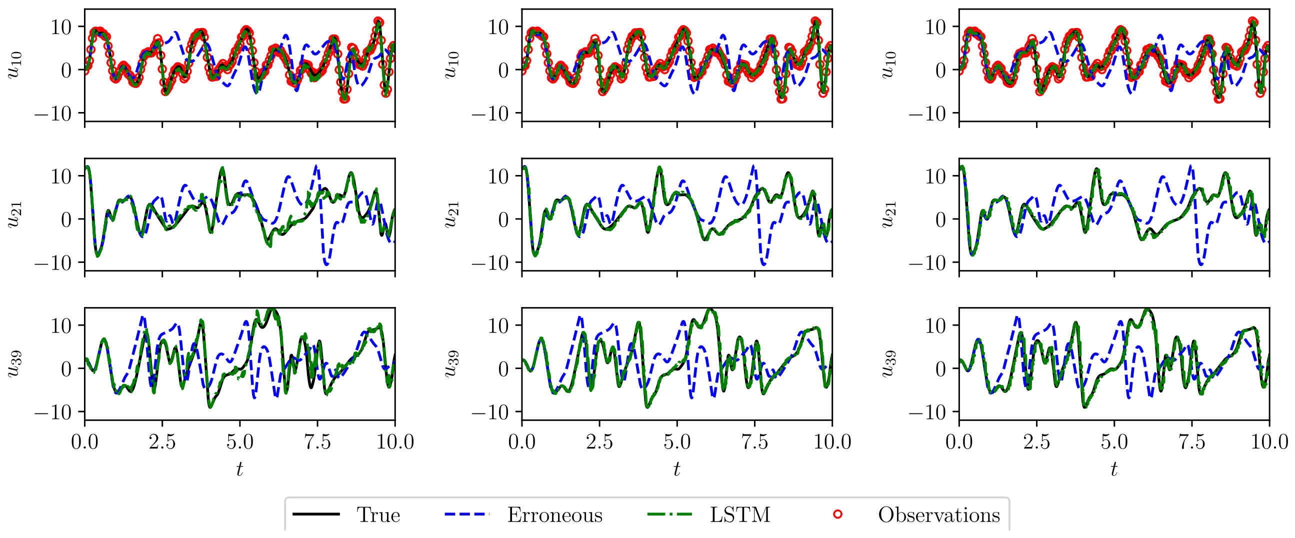


FIG. 12. Selected trajectories of the Lorenz 96 model with the analysis performed by the LSTM nudging with the N = 40 member ensemble for training using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

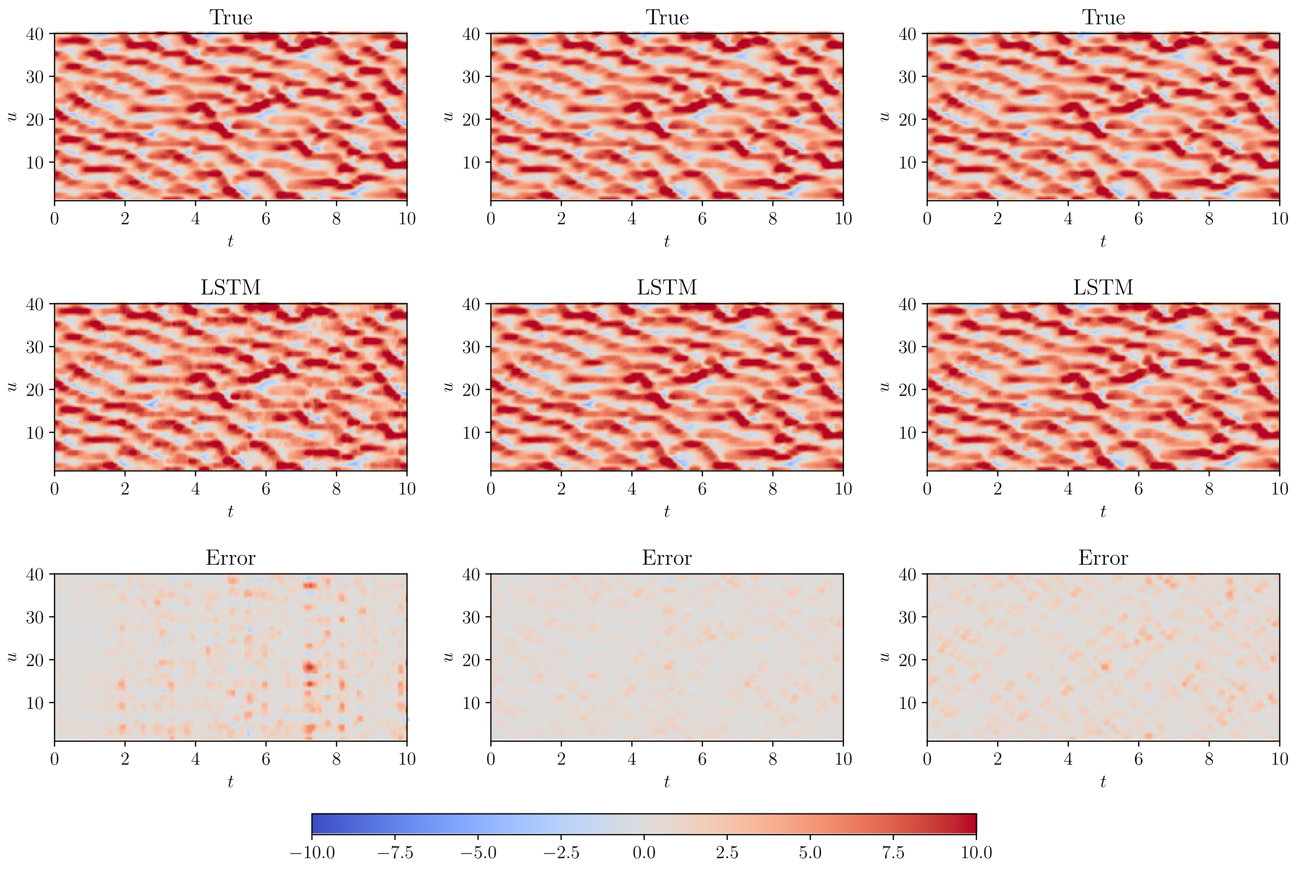


FIG. 13. Full state trajectory of the Lorenz 96 model with the analysis performed by the LSTM nudging with the N = 40 member ensemble for training using observations from m = 4 (left), m = 8 (middle), and m = 20 (right) state variables at every ten time steps.

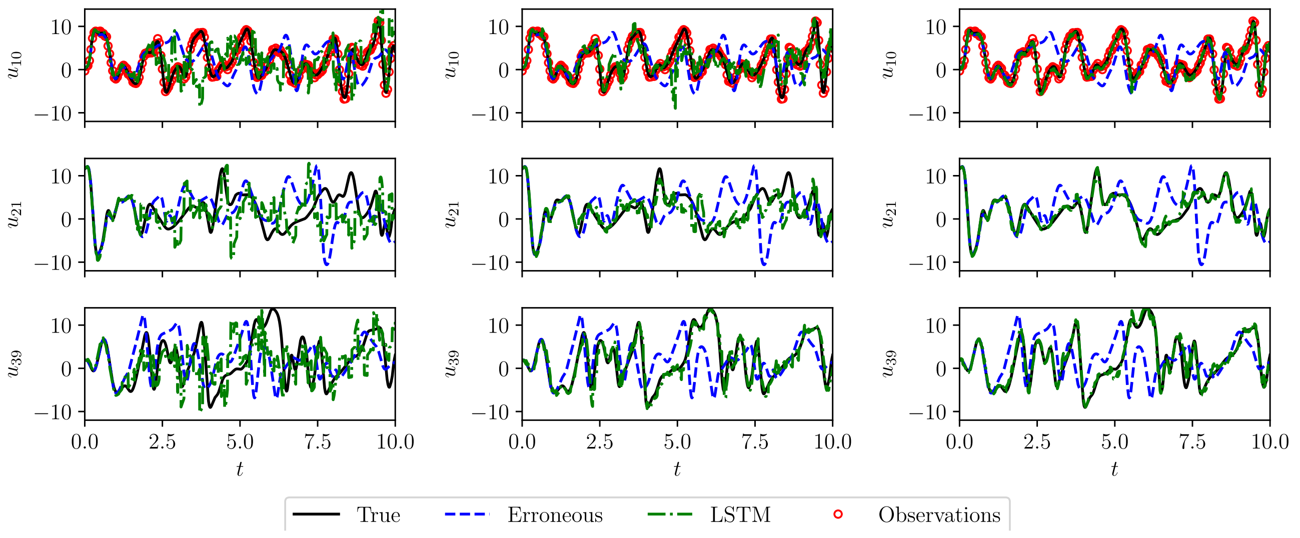


FIG. 14. Selected trajectories of the Lorenz 96 model with the analysis performed by the LSTM nudging with the N = 40 member ensemble for training using observations from m = 2 (left), m = 3 (middle), and m = 4 (right) state variables at every ten time steps.

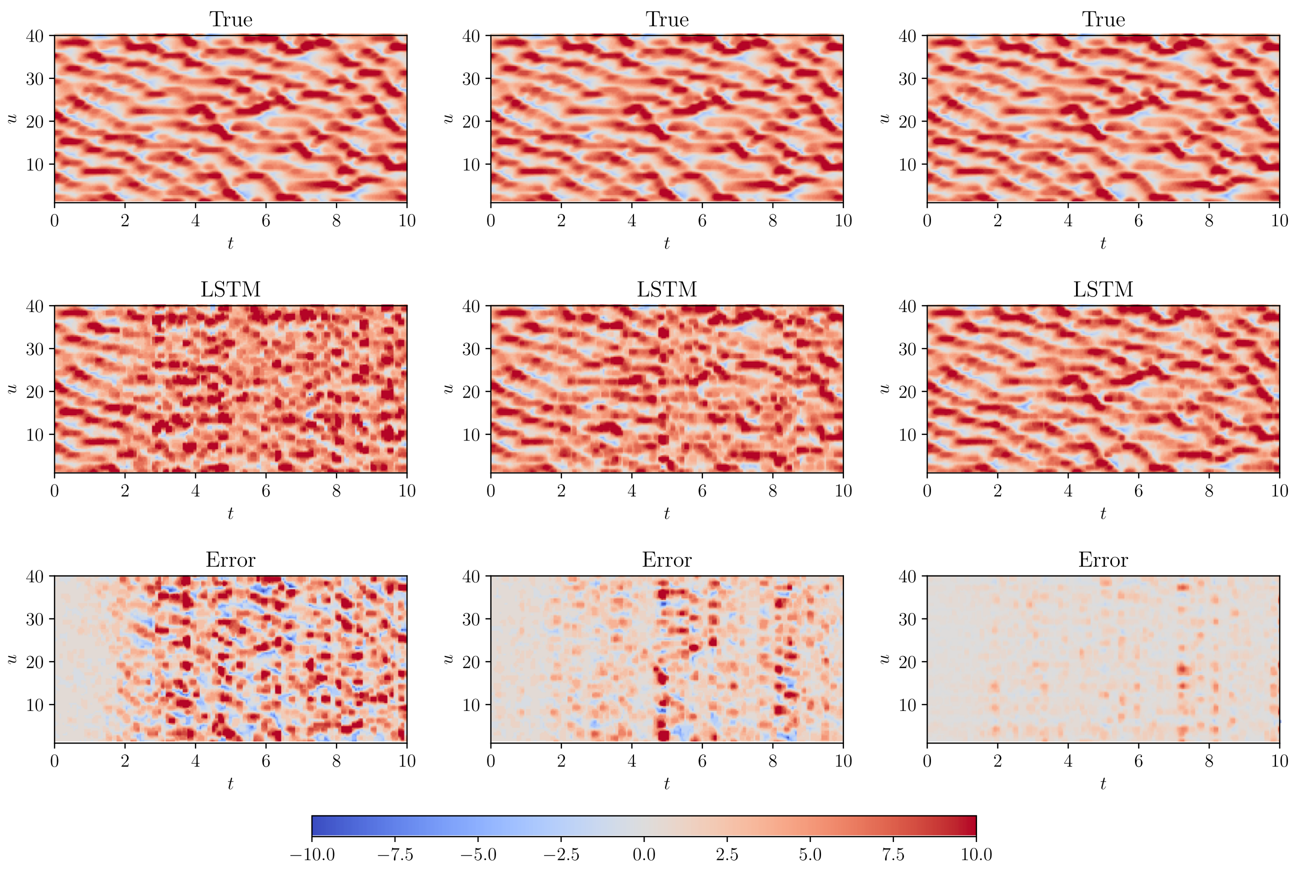


FIG. 15. Full state trajectory of the Lorenz 96 model with the analysis performed by the LSTM nudging with the N = 40 member ensemble for training using observations from m = 2 (left), m = 3 (middle), and m = 4 (right) state variables at every ten time steps.

The nonlinear filtering methods discussed in Appendix A are computationally expensive and are prone to the curse of dimensionality with an increase in the resolution of the forward numerical model. Nudging methods, on the other hand, are computationally inexpensive and straightforward to implement. As described in Sec. II A, nudging is accomplished by adding a correction term to the dynamical model, which is proportional to the difference between observations and model forecast. One of the main limitations of nudging methods is the *ad hoc* specification of the nudging relaxation coefficient, and it is not clear how to choose this coefficient to obtain an optimal solution.[90] Here, we demonstrate how the choice of nudging coefficient affects the prediction when 20 observations are available for assimilation. Since the nudging coefficient represents the relaxation of time scale, we use a constant value for the nudging coefficient that is a function of the time step of the model. Also, the nudging coefficient is assumed to be constant throughout the time integration, i.e., $G_k = \tau$, where τ is a function of the time step of the model. Figure 8 displays the time evolution of the selected states for three different values of the nudging coefficient. We notice that for a higher value of τ, the nudging method is not able to correct the model forecast accurately. Figure 9 provides the full state trajectory of the Lorenz 96 model with different nudging coefficient matrices. We observe that the error is sufficiently low at $\tau = 100\Delta t$. Although, the prediction can be further refined by fine-tuning of the nudging coefficient matrix.

Figures 10 and 11 present the time evolution of the selected states and full Lorenz 96 system for different numbers of observations with $\tau = 150\Delta t$. We can easily see that the prediction capability of the nudging scheme is poor when less number of observations are available for the assimilation. Indeed, the performance of the nudging scheme can be improved by the optimal specification of the nudging coefficient[42] or by using the back and forth nudging algorithm.[44] However, the optimal nudging coefficient computation involves obtaining an adjoint model and solving a constrained minimization problem. Also, the back and forth nudging algorithm requires $O(10)$ iterations for convergence, and the computational cost will be large for high-dimensional systems. Therefore, machine learning algorithms that are successful in finding the nonlinear mapping between two quantities can be exploited to learn the nudging dynamics.

Now, we describe the results of numerical experiments with the LSTM nudging scheme described in Sec. II B. For the fair comparison with the EnKF and DEnKF algorithms, we use the data generated from $N = 40$ perturbed initial conditions for the training of

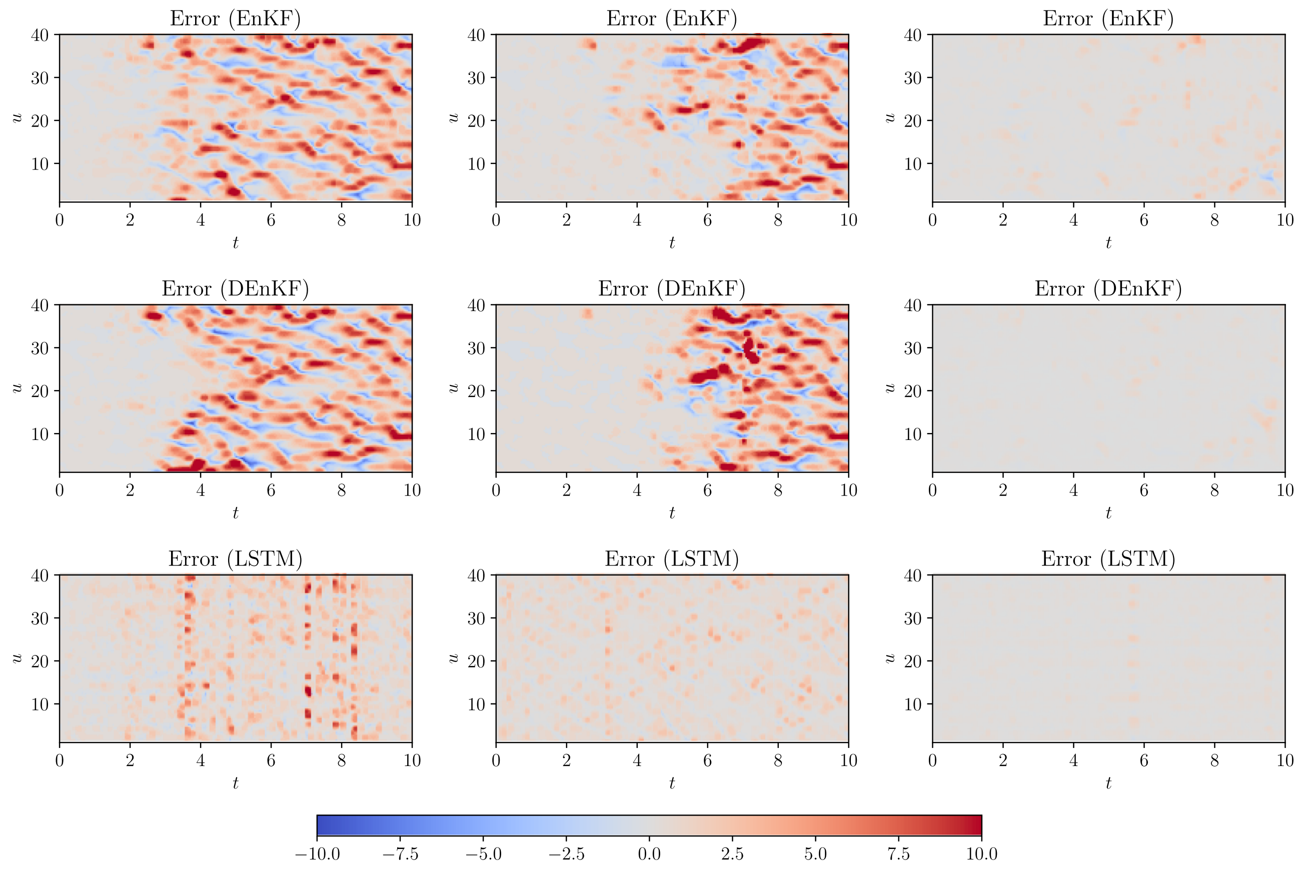


FIG. 16. Full state error of the Lorenz 96 model with the analysis performed by the ensemble Kalman filter (EnKF), deterministic ensemble Kalman filter (DEnKF), and LSTM nudging with the $N = 200$ member ensemble using observations from $m = 2$ (left), $m = 3$ (middle), and $m = 4$ (right) state variables at every ten time steps.

the LSTM network. These perturbed initial conditions are created by adding noise from the Gaussian distribution of zero mean and 1×10^{-2} variance to the erroneous initial condition. The training data are obtained by integrating the model with these perturbed initial conditions from time $t = 0$ to $t = 10$ with $\Delta t = 5 \times 10^{-3}$ and then storing the states at all times where observations are present. Therefore, there will be 40 000 samples available for training the LSTM network. The LSTM network is trained using the procedure described in Algorithm 1. We use fairly simple LSTM architecture with two hidden layers consisting of 80 LSTM cells each and train the network for 2500 epochs. We apply the ReLU activation function and Adam optimizer for the optimization. We found that our training is not highly sensitive to neural network hyperparameters, and a similar level of accuracy can be achieved with other sets of hyperparameters. Figure 12 presents the time evolution of selected states for three different numbers of observations. We see that the LSTM network has learned the mapping from input data to the correction term and is able to produce the correct trajectory even for those states for which observations are not available. In Fig. 13, we provide the full state trajectory of the Lorenz 96 model for the LSTM nudging method. We get a sufficient level of accuracy comparable to nonlinear filtering algorithms with 20% observations.

From our analysis of numerical experiments with three sets of observations, we can conclude that the LSTM network can learn the nudging dynamics efficiently. Some of the other questions that we want to investigate in this study are as follows: How sparse can the observations be for an accurate prediction? How much training data are required for training the network effectively? Figure 14 displays the time evolution of the selected states for the LSTM nudging scheme with very sparse observations, i.e., $m = 2$, 3, and 4, and we observe a large discrepancy between true and predicted states with less than 10% observations. Figure 15 reports the full state trajectory for the Lorenz 96 model with very sparse observations. The results in Figs. 14 and 15 suggest that at least 10% observations are necessary for producing the correct prediction with low error. We point out here that we utilized the data created from only 40 perturbed initial conditions for training, and it is well known that the performance of the neural network can be improved by training with more data.

In Figs. 16 and 17, we illustrate the improvement in prediction for highly sparse observations as the amount of data employed for training the LSTM network is increased. We show only the error plot (the difference between true and predicted states) for conciseness. We can easily observe that the error is large for the EnKF

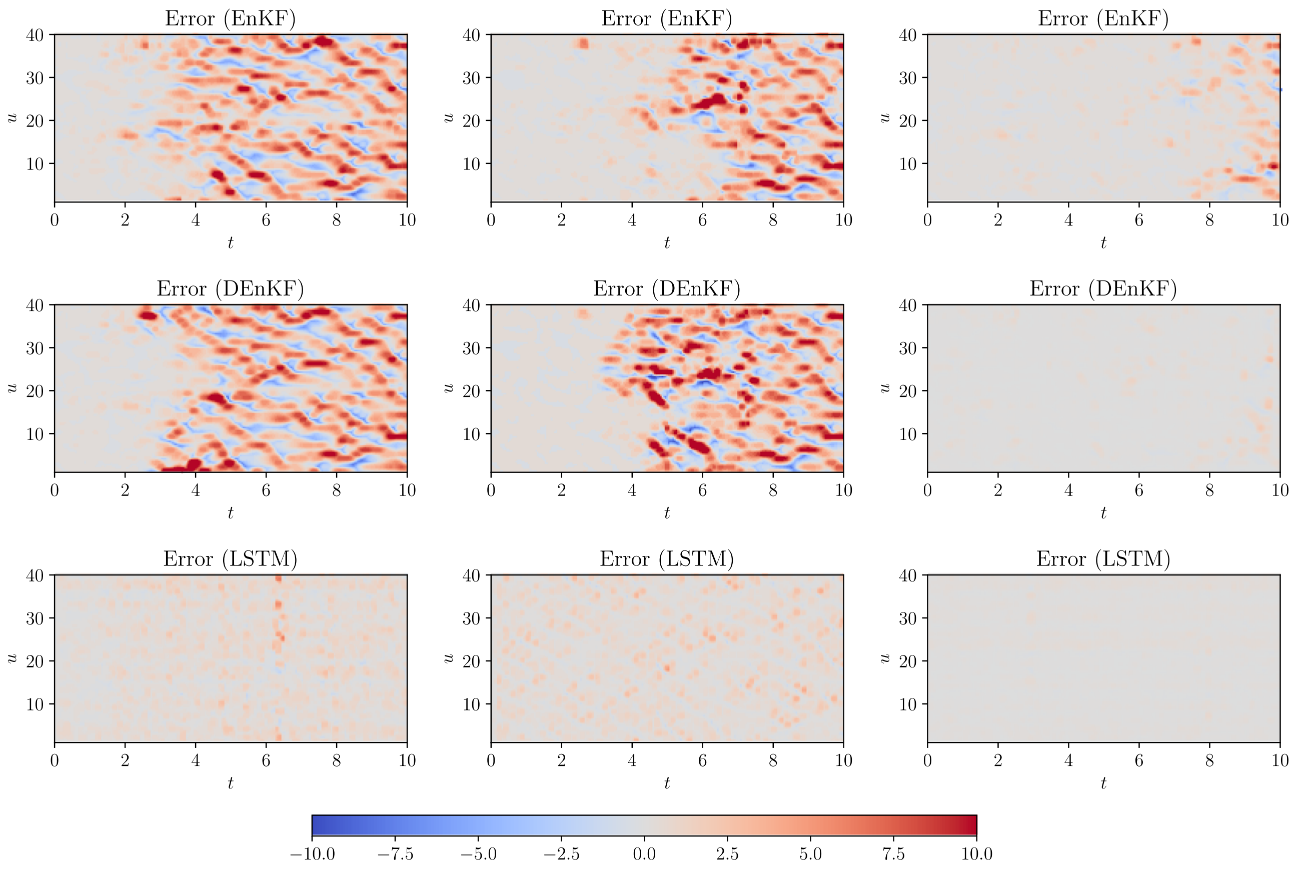


FIG. 17. Full state error of the Lorenz 96 model with the analysis performed by the ensemble Kalman filter (EnKF), deterministic ensemble Kalman filter (DEnKF), and LSTM nudging with $N = 400$ member ensemble using observations from $m = 2$ (left), $m = 3$ (middle), and $m = 4$ (right) state variables at every ten time steps.

and DEnKF algorithms compared to the LSTM nudging scheme when only two or three observations are available for assimilation. When four observations are present, we see a similar level of accuracy for EnKF, DEnKF, and LSTM nudging method. If we compare the error in Figs. 16 and 17, there is an improvement in the prediction as we increase the training data. The results presented in Figs. 16 and 17 are obtained by utilizing $N = 200$ and $N = 400$ ensemble members for the EnKF and DEnKF algorithms. The same number of perturbed initial conditions are also used for training the LSTM network. Therefore, in terms of computational cost, all three methods can be considered equivalent because the same number of forward numerical models are integrated from initial time to final time for all three methods. In terms of the storage, the LSTM nudging is more demanding as it requires the storage of full state for all training sets (i.e., perturbed initial condition) at all observation points for the training. However, there is no need to store the solution of all ensemble members in the EnKF and DEnKF algorithm. This limitation can be addressed by transfer learning, where the weights and biases of the neural network are updated by training its last few layers with new data. Therefore, training the LSTM network for the first time is a computationally intensive task, and the LSTM network can be retrained as new observations become available.

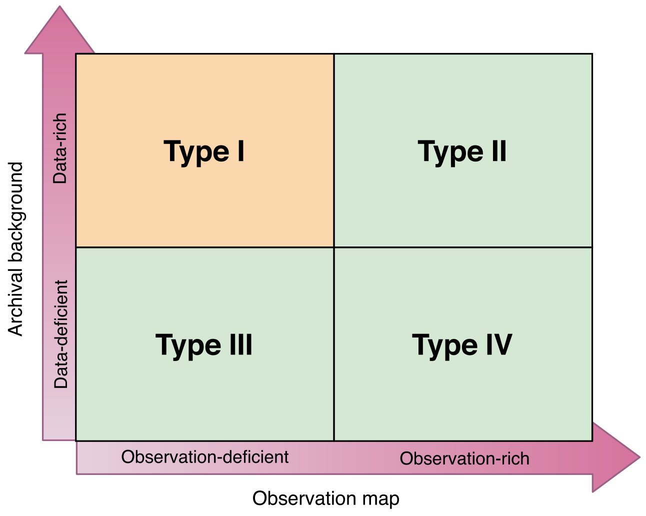


FIG. 18. Segregation of problems encountered in data-assimilation based on the observations and archival/ensemble background data. LSTM nudging method is particularly suitable where there is a rich amount archival or background information available for training the network and observations are sparse.

V. CONCLUSIONS

In the present study, we introduced the LSTM nudging scheme that learns the nudging dynamics from the full state of the system and partial observations. We illustrate the approach for the Lorenz 96 system and compare its performance against extended Kalman filter (EKF), ensemble Kalman filter (EnKF), and deterministic ensemble Kalman filter (DEnKF) approaches. We consider different aspects of the LSTM nudging scheme such as sparsity in observations and the amount of available data for training the LSTM network. We successfully demonstrate that the LSTM network can be trained to learn the nudging dynamics with extremely sparse observations provided that there is a large amount of training data. In terms of computational overhead, training the neural network is the most demanding task. However, this is a one time task, and future observations can be incorporated by retraining the neural network with transfer learning at a much less computational cost.

The results of our numerical experiments with the LSTM nudging scheme indicate its potential benefit of assimilation from very sparse observations. Another benefit is that there are no matrix computational operations such as Kalman gain calculation. One of the important caveats of the LSTM nudging scheme is that the neural networks are data-hungry, and hence, a large amount of archival or background data will be necessary to train the neural network. The suitability of the LSTM nudging scheme for DA problems is summarized in Fig. 18, where the DA problems are classified based on the sparsity of observations and the amount of archival background information. The LSTM nudging scheme is well suited for problems where observations are very sparse and there is an availability of archival background information (i.e., type I problems). These problems can arise when obtaining observations is very challenging or expensive, such as the sparse rain gauge network over complex topography.[91] Another limitation is that the training procedure in the present form will not be feasible for very high-dimensional systems. One of the solutions to address this constraint is to utilize reduced order modeling (ROM) approaches for dimensionality reduction, and recently, machine learning methods are found to give accurate, stable, and robust ROMs for physical systems. Since the LSTM nudging scheme is flexible, we foresee that this approach can be extended to large scale systems by blending it with ROM approaches. One more reservation of the LSTM nudging method is that it does not predict the uncertainty in analyzed states.

In the present work, we focus on a relatively simple chaotic system that adequately captures the essence of a variety of prediction problems. On the other hand, we can arguably discuss that many methods in chaotic synchronization might work well only for some specific systems. To elucidate this further, although not shown in this report, we verify that our proposed method is robust in parametric variability (e.g., we can reproduce similar conclusions using $F = 8$ in our model). With this in mind, however, we highlight that the robustness of the LSTM nudging scheme needs further investigation for problems that exhibit higher level of complexity than the Lorenz 96 model. Therefore, in our future studies, we intend to explore the feasibility of the LSTM nudging approach for more complex settings such as the two-scale Lorenz 96 system[79] and shell models for turbulence.[92–96]

We re-emphasize here that the significance of the proposed LSTM nudging method on the prototype model does not mean that it can be directly extended to higher-dimensional and more complex problems. In this work, we assumed that the model is perfect and the noise is Gaussian, which is a very idealized condition. In actual scenarios, real weather forecast models are approximate and contain a lot of parameterizations for subgrid scale processes. Therefore, one can look at the results of numerical experiments presented in this study as the early findings, and substantial future work is required for the demonstration of the proposed method in a realistic situation. As a part of future studies, we plan to illustrate the LSTM nudging method for a two-dimensional quasi-geostrophic model with an application of the convolutional autoencoder for dimensionality reduction. Neural networks have also been shown to be capable of discovering hidden information about the physical processes embedded in the data,[97,98] and we will integrate these methods with the LSTM nudging scheme for imperfect models.

ACKNOWLEDGMENTS

We thank Professor Sivaramakrishnan Lakshmivarahan for his archival NPTEL lectures for covering the mechanics of the nudging and nonlinear filtering methods. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award No. DE-SC0019290. O.S. gratefully acknowledges the U.S. Department of Energy Early Career Research Program support.

APPENDIX A: NONLINEAR FILTERING

In this appendix, we outline the algorithms for extended Kalman filter (EKF), ensemble Kalman filter (EnKF), and deterministic ensemble Kalman filter (DEnKF). In sequential data assimilation problems, the objective is to estimate the state $\mathbf{x}_k$ given the observations up to time t_k, i.e., $\mathbf{z}_1, \ldots, \mathbf{z}_k$. When we use observations to estimate the state of the system, we say that the data are assimilated into the model. There is a number of studies that deal with non-Gaussian distributions for noise vectors.[99–101] However, this is outside the scope of this study, and we restrict to the assumption of Gaussian noise for model and measurement errors.

We will use the notation $\widehat{\mathbf{x}}_k$ to denote an analyzed state of the system at time t_k when all of the observations up to and including time t_k are used in determining the state of the system. When all the observations before (but not including) time t_k are utilized for estimating the state of the system, then we call it the forecast estimate and denote it as $\mathbf{x}_k^f$. We use the notation $\mathbf{P}_k$ to denote the error covariance matrix. The error covariance matrix for the state vector $\mathbf{x}_k$ is defined as

$$\mathbf{P}_k = \mathrm{E}[(\mathbf{x}_k - \mathrm{E}[\mathbf{x}_k])(\mathbf{x}_k - \mathrm{E}[\mathbf{x}_k])^{\mathrm{T}}], \tag{A1}$$

where $\mathrm{E}[\cdot]$ denotes the expected value. We use $\widehat{\mathbf{P}}_k$ to denote the error covariance for an analyzed state $\widehat{\mathbf{x}}_k$ and $\mathbf{P}_k^f$ denotes the error covariance for the forecast estimate $\mathbf{x}_k^f$.

1. Extended Kalman filter

To start with an EKF algorithm, we initialize the state of the system and error covariance matrix,

$$\widehat{\mathbf{x}}_0 = E[\mathbf{x}_0], \tag{A2}$$

$$\widehat{\mathbf{P}}_0 = \mathbf{P}_0. \tag{A3}$$

Then, we evolve the state of the system between two observation points (from time t_k to t_{k+1}) using the known nonlinear dynamics,

$$\mathbf{x}_{k+1}^f = \mathbf{M}(\widehat{\mathbf{x}}_k), \tag{A4}$$

and the error covariance matrix is propagated between two observation points using

$$\mathbf{P}_{k+1}^f = \mathbf{D_M}\widehat{\mathbf{P}}_k\mathbf{D_M}^{\mathrm{T}} + \mathbf{Q}_{k+1}. \tag{A5}$$

Here, $\mathbf{D_M} \in \mathbb{R}^{n\times n}$ is the Jacobian of the model $\mathbf{M}(\cdot)$ and the superscript T denotes the transpose of the matrix. Once the observation $\mathbf{z}_{k+1}$ becomes available at time t_{k+1}, we assimilate it into the forecast state using

$$\widehat{\mathbf{x}}_{k+1} = \mathbf{x}_{k+1}^f + \mathbf{K}[\mathbf{z}_{k+1} - h(\mathbf{x}_{k+1}^f)]. \tag{A6}$$

The matrix $\mathbf{K} \in \mathbb{R}^{n\times m}$ refers to the Kalman gain matrix and is computed as

$$\mathbf{K} = \mathbf{P}_{k+1}^f\mathbf{D_h}^{\mathrm{T}}[\mathbf{D_h}\mathbf{P}_{k+1}^f\mathbf{D_h}^{\mathrm{T}} + \mathbf{R}_{k+1}]^{-1}, \tag{A7}$$

where $\mathbf{D_h} \in \mathbb{R}^{m\times n}$ is the Jacobian of observation function $h(\cdot)$. The Kalman gain matrix decides the influence of measurements on the estimated state. When the measurement error covariance $\mathbf{R}_{k+1}$ approaches zero, the Kalman gain $\mathbf{K}$ gives more weight to the residual defined as $[\mathbf{z}_{k+1} - h(\mathbf{x}_{k+1}^f)]$. On the other hand, when the error covariance $\mathbf{P}_{k+1}^f$ is very small, the Kalman gain $\mathbf{K}$ weights the residual less heavily. Each row of the Kalman gain matrix contains the influence of all observation points on one element of the state $\mathbf{x}_{k+1}$ corresponding to that row. The analyzed error covariance matrix is calculated by

$$\widehat{\mathbf{P}}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{D_h})\mathbf{P}_{k+1}^f, \tag{A8}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix. The complete procedure for the EKF is summarized in Algorithm 3.

2. Ensemble Kalman filter

When the system is high-dimensional, i.e., n is very large, then the computations for the EKF algorithm are practically infeasible. In addition, the EKF algorithm requires computation of Jacobians, and it might be numerically difficult to compute Jacobians for complex models. Ensemble filtering techniques are attractive for such systems where the approximate state of the system is estimated using the standard Monte Carlo framework. In EKF, the mean estimate of the state $\widehat{\mathbf{x}}_k$ and the error covariance matrix $\widehat{\mathbf{P}}_k$ are updated sequentially. In contrast to an EKF algorithm, we apply the forecast step to an ensemble of states in the EnKF algorithm.[1] The sample mean and

ALGORITHM 3. Extended Kalman filter.

```
1: Initialize the state of the system and error covariance.
       $\widehat{\mathbf{x}}_0 = E[\mathbf{x}_0]$,
       $\widehat{\mathbf{P}}_0 = \mathbf{P}_0$.
2: For $k = 0, 1, \ldots$, proceed as follows:
   • Forecast step: integrate the state estimate and its error
     covariance from time $t_k$ to $t_{k+1}$ as follows:
       $\mathbf{x}^f_{k+1} = \mathbf{M}(\widehat{\mathbf{x}}_k)$,
       $\mathbf{P}^f_{k+1} = \mathbf{D_M}\widehat{\mathbf{P}}_k\mathbf{D}^{\mathrm{T}}_{\mathbf{M}} + \mathbf{Q}_{k+1}$.
   • Data assimilation step: once the observations are available at
     time $t_{k+1}$, they are incorporated into the state estimate and error
     covariance estimation as follows:
       $\widehat{\mathbf{x}}_{k+1} = \mathbf{x}^f_{k+1} + \mathbf{K}[\mathbf{z}_{k+1} - h(\mathbf{x}^f_{k+1})]$,
       $\mathbf{K} = \mathbf{P}^f_{k+1}\mathbf{D}^{\mathrm{T}}_{\mathbf{h}}[\mathbf{D_h}\mathbf{P}^f_{k+1}\mathbf{D}^{\mathrm{T}}_{\mathbf{h}} + \mathbf{R}_{k+1}]^{-1}$,
       $\widehat{\mathbf{P}}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{D_h})\mathbf{P}^f_{k+1}$.
```

covariance of the ensembles analyses represent the analyzed state estimate $\widehat{\mathbf{x}}_k$ and error covariance matrix $\widehat{\mathbf{P}}_k$.

Let x_0 be an initial condition drawn from the Gaussian distribution with mean $\mathbf{m}_0$ and the covariance matrix $\mathbf{P}_0$, i.e., $x_0 \sim \mathcal{N}(\mathbf{m}_0, \mathbf{P}_0)$. In our notation, we use $\mathbf{X}_k(i)$ to denote the ith member of ensembles and N is the size of ensembles, i.e., $i = 1, 2, \ldots, N$. We initialize the state of the system for all ensemble members from known distribution of the initial condition for the system as

$$\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i). \tag{A9}$$

Then, we forecast the state of the system for all ensemble members between two observation points (i.e., from time t_k to t_{k+1}) using the nonlinear model dynamics as

$$\mathbf{X}^f_{k+1}(i) = \mathbf{M}(\widehat{\mathbf{X}}_k(i)) + \mathbf{w}_{k+1}. \tag{A10}$$

The forecast state estimate and error covariance are calculated based on the sample mean and sample variance of all ensembles as follows:

$$\mathbf{x}^f_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}^f_{k+1}(i), \tag{A11}$$

$$\mathbf{P}^f_{k+1} = \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{E}^f_{k+1}(i)[\mathbf{E}^f_{k+1}(i)]^{\mathrm{T}}, \tag{A12}$$

where

$$\mathbf{E}^f_{k+1}(i) = \mathbf{X}^f_{k+1}(i) - \mathbf{x}^f_{k+1}. \tag{A13}$$

Once the observations $\mathbf{z}_{k+1}$ are available at time t_{k+1}, we create N different virtual observations using

$$\mathbf{Z}_{k+1}(i) = \mathbf{z}_{k+1} + \mathbf{v}_{k+1}(i). \tag{A14}$$

In the original formulation of the EnKF algorithm proposed by Evensen,[102] virtual observations were not used in the assimilation step. However, Burgers, Jan van Leeuwen, and Evensen[103] showed that it is essential to include random perturbations to observations to ensure that the analyzed covariance is not underestimated. Once the virtual observations are generated, the forecast state estimate for all ensembles are assimilated by

$$\widehat{\mathbf{X}}_{k+1}(i) = \mathbf{X}^f_{k+1}(i) + \mathbf{K}[\mathbf{Z}_{k+1}(i) - h(\mathbf{X}^f_{k+1}(i))]. \tag{A15}$$

The Kalman gain $\mathbf{K}$ is computed using the same formula as the EKF algorithm. The analysis state estimate is calculated using the sample mean of the analyzed state estimate for all ensemble members as follows:

$$\widehat{\mathbf{x}}_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\widehat{\mathbf{X}}_{k+1}(i). \tag{A16}$$

The complete procedure for the EnKF is summarized in Algorithm 4.

ALGORITHM 4. Ensemble Kalman filter.

```
1: Initialize the state of the system for different ensemble members.
       $\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i)$,
   where $\mathbf{y}_0(i) \sim N(0, \mathbf{P}_0)$.
2: For $k = 0, 1, \ldots$, proceed as follows:
   • Forecast step:
     – Integrate the state estimate all ensemble members from time
       $t_k$ to $t_{k+1}$ as follows:
           $\mathbf{X}^f_{k+1}(i) = \mathbf{M}(\widehat{\mathbf{X}}_k(i)) + \mathbf{w}_{k+1}$.
     – Compute the sample mean and error covariance as follows:
           $\mathbf{x}^f_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}^f_{k+1}(i)$,
           $\mathbf{E}^f_{k+1}(i) = \mathbf{X}^f_{k+1}(i) - \mathbf{x}^f_{k+1}$,
           $\mathbf{P}^f_{k+1} = \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{E}^f_{k+1}(i)[\mathbf{E}^f_{k+1}(i)]^{\mathrm{T}}$.
   • Data assimilation step:
     – Once the observations are available at time $t_{k+1}$, generate $N$
       realizations of virtual observations as follows:
           $\mathbf{Z}_{k+1}(i) = \mathbf{z}_{k+1} + \mathbf{v}_{k+1}(i)$,
       where $\mathbf{v}_{k+1}(i) \sim N(0, \mathbf{R}_{k+1})$.
     – Assimilate the state estimate with virtual observations for all
       ensemble members as follows:
           $\widehat{\mathbf{X}}_{k+1}(i) = \mathbf{X}^f_{k+1}(i) + \mathbf{K}[\mathbf{Z}_{k+1}(i) - h(\mathbf{X}^f_{k+1}(i))]$,
           $\mathbf{K} = \mathbf{P}^f_{k+1}\mathbf{D}^{\mathrm{T}}_{\mathbf{h}}[\mathbf{D_h}\mathbf{P}^f_{k+1}\mathbf{D}^{\mathrm{T}}_{\mathbf{h}} + \mathbf{R}_{k+1}]^{-1}$.
     – Compute the sample mean to get analysis state estimate at time
       $t_{k+1}$,
           $\widehat{\mathbf{x}}_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\widehat{\mathbf{X}}_{k+1}(i)$.
```

3. Deterministic ensemble Kalman filter

Sakov and Oke[89] proposed a modification in traditional EnKF that results into matching the analyzed error covariance to that of the standard Kalman filter without the need for virtual observations. We start the DEnKF algorithm in a similar manner as the EnKF algorithm by initializing the state estimate for all ensemble members using

$$\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i). \tag{A17}$$

The anomalies between the forecast estimate of all ensembles and its sample mean are computed utilizing

$$\mathbf{A}^f_{k+1}(i) = \mathbf{X}^f_{k+1}(i) - \mathbf{x}^f_{k+1}, \tag{A18}$$

where

$$\mathbf{x}^f_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}^f_{k+1}(i). \tag{A19}$$

The error covariance matrix becomes

$$\mathbf{P}^f_{k+1} = \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{A}^f_{k+1}(i)[\mathbf{A}^f_{k+1}(i)]^{\mathrm{T}}. \tag{A20}$$

Once the observations are available at time t_{k+1}, the forecast state estimate is assimilated as

$$\widehat{\mathbf{x}}_{k+1} = \mathbf{x}^f_{k+1} + \mathbf{K}[\mathbf{z}_{k+1} - h(\mathbf{x}^f_{k+1})], \tag{A21}$$

where the Kalman gain $\mathbf{K}$ is computed in a similar manner as the EKF algorithm as follows:

$$\mathbf{K} = \mathbf{P}^f_{k+1}\mathbf{D}_{\mathbf{h}}^{\mathrm{T}}[\mathbf{D}_{\mathbf{h}}\mathbf{P}^f_{k+1}\mathbf{D}_{\mathbf{h}}^{\mathrm{T}} + \mathbf{R}_{k+1}]^{-1}. \tag{A22}$$

The anomalies for all ensemble members are updated separately with half the Kalman gain. Therefore, the analyzed anomalies for all ensemble members are calculated using the following expression:

$$\widehat{\mathbf{A}}_{k+1}(i) = \mathbf{A}^f_{k+1}(i) - \frac{1}{2}\mathbf{K}\mathbf{D}_{\mathbf{h}}\mathbf{A}^f_{k+1}(i). \tag{A23}$$

The analyzed state estimate for all ensemble members is then obtained by offsetting the analyzed anomalies with the analyzed state estimate and is computed using

$$\widehat{\mathbf{X}}_{k+1}(i) = \widehat{\mathbf{A}}_{k+1}(i) + \widehat{\mathbf{x}}_{k+1}. \tag{A24}$$

The procedure for the deterministic EnKF (DEnKF) is summarized in Algorithm 5. In practice (e.g., when $n \gg N$), we compute Eq. (A22) using its square root version (without storing or computing $\mathbf{P}^f_{k+1}$ explicitly),

$$\mathbf{K} = \frac{1}{N-1}\mathbf{A}^f(\mathbf{D}_{\mathbf{h}}\mathbf{A}^f)^{\mathrm{T}}\left[\frac{1}{N-1}(\mathbf{D}_{\mathbf{h}}\mathbf{A}^f)(\mathbf{D}_{\mathbf{h}}\mathbf{A}^f)^{\mathrm{T}} + \mathbf{R}\right]^{-1}, \tag{A25}$$

where a size of $\mathbb{R}^{n\times N}$ matrix is concatenated as follows:

$$\mathbf{A}^f = [\mathbf{A}^f_{k+1}(1), \mathbf{A}^f_{k+1}(2), \ldots, \mathbf{A}^f_{k+1}(N)]. \tag{A26}$$

ALGORITHM 5. Deterministic ensemble Kalman filter.

```
1: Initialize the state of the system for different ensemble members.
       $\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i)$,
   where $\mathbf{y}_0(i) \sim N(0, \mathbf{P}_0)$.
2: For $k = 0, 1, \ldots$ proceed as follows:
   • Forecast step:
     – Integrate the state estimate all ensemble members from time $t_k$ to $t_{k+1}$ as follows:
           $\mathbf{X}^f_{k+1}(i) = \mathbf{M}(\widehat{\mathbf{X}}_k(i))$.
     – Compute the sample mean, ensemble anomalies, and error covariance as follows:
           $\mathbf{x}^f_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}^f_{k+1}(i)$,
           $\mathbf{A}^f_{k+1}(i) = \mathbf{X}^f_{k+1}(i) - \mathbf{x}^f_{k+1}$,
           $\mathbf{P}^f_{k+1} = \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{A}^f_{k+1}(i)[\mathbf{A}^f_{k+1}(i)]^{\mathrm{T}}$.
   • Data assimilation step:
     – Once the observations are available at time $t_{k+1}$, assimilate the forecast state estimate with the observation as follows:
           $\widehat{\mathbf{x}}_{k+1} = \mathbf{x}^f_{k+1} + \mathbf{K}[\mathbf{z}_{k+1} - h(\mathbf{x}^f_{k+1})]$,
           $\mathbf{K} = \mathbf{P}^f_{k+1}\mathbf{D}_{\mathbf{h}}^{\mathrm{T}}[\mathbf{D}_{\mathbf{h}}\mathbf{P}^f_{k+1}\mathbf{D}_{\mathbf{h}}^{\mathrm{T}} + \mathbf{R}_{k+1}]^{-1}$.
     – Compute the analyzed anomalies as follows:
           $\widehat{\mathbf{A}}_{k+1}(i) = \mathbf{A}^f_{k+1}(i) - \frac{1}{2}\mathbf{K}\mathbf{D}_{\mathbf{h}}\mathbf{A}^f_{k+1}(i)$.
     – Calculate the analyzed ensemble using the analyzed state estimate and analyzed anomalies as follows:
           $\widehat{\mathbf{X}}_{k+1}(i) = \widehat{\mathbf{A}}_{k+1}(i) + \widehat{\mathbf{x}}_{k+1}$.
```

In other words, we skip computing Eq. (A20) and use its reduced-rank square root definition given by

$$\mathbf{P}^f_{k+1} = \frac{1}{N-1}\mathbf{A}^f(\mathbf{A}^f)^T. \tag{A27}$$

APPENDIX B: JACOBIAN OF THE MODEL AND OBSERVATION MATRIX

We apply a fourth-order Runge–Kutta (RK4) numerical scheme for temporal integration of the Lorenz 96 model, and it can be written as follows:

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \frac{\Delta t}{6}(\mathbf{g}_1 + 2\mathbf{g}_2 + 2\mathbf{g}_3 + \mathbf{g}_4), \tag{B1}$$

where

$$\mathbf{g}_1 = \mathbf{f}(\mathbf{u}^k), \tag{B2}$$

$$\mathbf{g}_2 = \mathbf{f}(\mathbf{u}^k + \frac{\Delta t}{2}\cdot\mathbf{g}_1), \tag{B3}$$

$$\mathbf{g}_3 = \mathbf{f}\left(\mathbf{u}^k + \frac{\Delta t}{2} \cdot \mathbf{g}_2\right), \tag{B4}$$

$$\mathbf{g}_4 = \mathbf{f}\left(\mathbf{u}^k + \Delta t \cdot \mathbf{g}_3\right). \tag{B5}$$

The function **f** is the right-hand side of the Lorenz 96 model, and in the discrete form, it can be written as

$$f_i = u_{i-1}(u_{i+1} - u_{i-2}) - u_i + F. \tag{B6}$$

The Jacobian of the function **f** is defined as follows:

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \left[\frac{\partial f_i}{\partial u_j}\right] \quad \text{for } 1 \le i, j \le n. \tag{B7}$$

The Jacobian **J** will be a $\mathbb{R}^{n\times n}$ matrix. The Jacobian of the model $\mathbf{D_M} \in \mathbb{R}^{n\times n}$ can be computed by applying the chain rule to Eq. (B1) and is given as follows:

$$\mathbf{D_M} = \mathbf{I} + \Delta t \cdot \mathbf{J} + \frac{1}{2}\Delta t^2 \cdot \mathbf{J}^2 + \frac{1}{6}\Delta t^3 \cdot \mathbf{J}^3 + \frac{1}{24}\Delta t^4 \cdot \mathbf{J}^4, \tag{B8}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix.

The Jacobian of observation is denoted by $\mathbf{D_h} \in \mathbb{R}^{m\times n}$ and is computed as follows:

$$\mathbf{D_h} = \left[\frac{\partial h_i}{\partial u_j}\right], \tag{B9}$$

where $1 \le i \le m$ and $1 \le j \le n$. Since we use linear observations, $\mathbf{D_h}$ will be a constant sparse matrix. Each row of the matrix $\mathbf{D_h}$ will consist of all zeros except for the corresponding observation location, where it will have the value of one.

DATA AVAILABILITY

The data that support the findings of this study are available within the article. For interested readers, we also provide a GitHub repository (https://github.com/surajp92/LSTM_Nudging) describing the algorithms' Python implementations for the reproduction of the numerical experiments discussed in the present study.

REFERENCES

[1] J. M. Lewis, S. Lakshmivarahan, and S. Dhall, *Dynamic Data Assimilation: A Least Squares Approach* (Cambridge University Press, 2006), Vol. 104.

[2] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches* (John Wiley & Sons, 2006).

[3] G. Evensen, *Data Assimilation: The Ensemble Kalman Filter* (Springer Science & Business Media, 2009).

[4] P. J. Van Leeuwen, "Particle filtering in geophysical systems," Mon. Weather Rev. **137**, 4089–4114 (2009).

[5] R. Lguensat, P. Tandeo, P. Ailliot, M. Pulido, and R. Fablet, "The analog data assimilation," Mon. Weather Rev. **145**, 4093–4107 (2017).

[6] M. Tang, Y. Liu, and L. J. Durlofsky, "A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems," J. Comput. Phys. **413**, 109456 (2020).

[7] R. Hu, F. Fang, C. C. Pain, and I. M. Navon, "Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method," J. Hydrol. **575**, 911–920 (2019).

[8] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, "A machine-learning-based global atmospheric forecast model," Geophys. Res. Lett. **47**(9), e2020GL087776, https://doi.org/10.1029/2020gl087776 (2020).

[9] S. M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu, "Nonintrusive reduced order modeling framework for quasigeostrophic turbulence," Phys. Rev. E **100**, 053306 (2019).

[10] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).

[11] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," Proc. R. Soc. A **474**, 20170844 (2018).

[12] P. Li, Y. Zha, L. Shi, C.-H. M. Tso, Y. Zhang, and W. Zeng, "Comparison of the use of a physical-based model with data assimilation and machine learning methods for simulating soil water dynamics," J. Hydrol. **584**, 124692 (2020).

[13] M. Cheng, F. Fang, C. C. Pain, and I. M. Navon, "Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network," Comput. Methods Appl. Mech. Eng. **365**, 113000 (2020).

[14] B. Protas, B. R. Noack, and J. Östh, "Optimal nonlinear eddy viscosity in Galerkin models of turbulent flows," J. Fluid Mech. **766**, 337–367 (2015).

[15] C. Zerfas, L. G. Rebholz, M. Schneier, and T. Iliescu, "Continuous data assimilation reduced order models of fluid flow," Comput. Methods Appl. Mech. Eng. **357**, 112596 (2019).

[16] D. Xiao, J. Du, F. Fang, C. Pain, and J. Li, "Parameterised non-intrusive reduced order methods for ensemble Kalman filter data assimilation," Comput. Fluids **177**, 69–77 (2018).

[17] D. N. Daescu and I. M. Navon, "Efficiency of a POD-based reduced second-order adjoint model in 4D-VAR data assimilation," Int. J. Numer. Methods Fluids **53**, 985–1004 (2007).

[18] R. Ştefănescu, A. Sandu, and I. M. Navon, "POD/DEIM reduced-order strategies for efficient four dimensional variational data assimilation," J. Comput. Phys. **295**, 569–595 (2015).

[19] Y. Cao, J. Zhu, I. M. Navon, and Z. Luo, "A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition," Int. J. Numer. Methods Fluids **53**, 1571–1583 (2007).

[20] C. Robert, S. Durbiano, E. Blayo, J. Verron, J. Blum, and F.-X. Le Dimet, "A reduced-order strategy for 4D-VAR data assimilation," J. Mar. Syst. **57**, 70–82 (2005).

[21] R. Arcucci, L. Mottet, C. Pain, and Y.-K. Guo, "Optimal reduced space for variational data assimilation," J. Comput. Phys. **379**, 51–69 (2019).

[22] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2019).

[23] V. Puzyrev, M. Ghommem, and S. Meka, "pyROM: A computational framework for reduced order modeling," J. Comput. Sci. **30**, 157–173 (2019).

[24] M. Bocquet, J. Brajard, A. Carrassi, and L. Bertino, "Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization," Found. Data Sci. **2**, 55 (2020).

[25] J. Brajard, A. Carassi, M. Bocquet, and L. Bertino, "Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model," arXiv:2001.01520 (2020).

[26] H. D. Abarbanel, P. J. Rozdeba, and S. Shirman, "Machine learning: Deepest learning as statistical data assimilation problems," Neural Comput. **30**, 2025–2055 (2018).

[27] M. Bocquet, J. Brajard, A. Carrassi, and L. Bertino, "Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models," Nonlinear Process. Geophys. **26**, 143–162 (2019).

[28] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber, "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets," Neural Networks **16**, 241–250 (2003).

[29] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," IEEE Trans. Neural Networks **5**, 279–297 (1994).

[30] R. A. Anthes, "Data assimilation and initialization of hurricane prediction models," J. Atmos. Sci. **31**, 702–719 (1974).

[31] J. Zhu, S. Hu, R. Arcucci, C. Xu, J. Zhu, and Y.-k. Guo, "Model error correction in data assimilation by integrating neural networks," Big Data Min. Anal. **2**, 83–91 (2019).

[32]E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability* (Cambridge University Press, 2003).
[33]A. Gelb, *Applied Optimal Estimation* (MIT Press, 1974).
[34]G. Welch and G. Bishop, "An introduction to the Kalman filter," University of North Carolina, Department of Computer Science, TR 95-041, 1995.
[35]S. Lakshmivarahan and J. M. Lewis, "Nudging methods: A critical overview," in *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications* (Springer, 2013), Vol. II, pp. 27–57.
[36]T. Krishnamurti, J. Xue, H. Bedi, K. Ingles, and D. Oosterhof, "Physical initialization for numerical weather prediction over the tropics," Tellus B **43**, 53–81 (1991).
[37]D. R. Stauffer and N. L. Seaman, "Use of four-dimensional data assimilation in a limited-area mesoscale model. Part I: Experiments with synoptic-scale data," Mon. Weather Rev. **118**, 1250–1277 (1990).
[38]D. R. Stauffer, N. L. Seaman, and F. S. Binkowski, "Use of four-dimensional data assimilation in a limited-area mesoscale model. Part II: Effects of data assimilation within the planetary boundary layer," Mon. Weather Rev. **119**, 734–754 (1991).
[39]A. Lorenc, R. Bell, and B. Macpherson, "The meteorological office analysis correction data assimilation scheme," Q. J. R. Metereol. Soc. **117**, 59–89 (1991).
[40]J. Derber and A. Rosati, "A global oceanic data assimilation system," J. Phys. Oceanogr. **19**, 1333–1347 (1989).
[41]A. Azouani, E. Olson, and E. S. Titi, "Continuous data assimilation using general interpolant observables," J. Nonlinear Sci. **24**, 277–304 (2014).
[42]X. Zou, I. M. Navon, and F. Le Dimet, "An optimal nudging data assimilation scheme using parameter estimation," Q. J. R. Metereol. Soc. **118**, 1163–1186 (1992).
[43]P. Vidard, F. Le Dimet, and A. Piacentini, "Determination of optimal nudging coefficients," Tellus A **55**, 1–15 (2003).
[44]D. Auroux and J. Blum, "Back and forth nudging algorithm for data assimilation problems," C. R. Math. **340**, 873–878 (2005).
[45]D. Auroux and J. Blum, "A nudging-based data assimilation method: The back and forth nudging (BFN) algorithm," Nonlinear Process. Geophys. **15**, 305–319 (2008).
[46]K. M. Waldron, J. Paegle, and J. D. Horel, "Sensitivity of a spectrally filtered and nudged limited-area model to outer model options," Mon. Weather Rev. **124**, 529–547 (1996).
[47]H. von Storch, H. Langenberg, and F. Feser, "A spectral nudging technique for dynamical downscaling purposes," Mon. Weather Rev. **128**, 3664–3673 (2000).
[48]R. Radu, M. Déqué, and S. Somot, "Spectral nudging in a spectral regional climate model," Tellus A **60**, 898–910 (2008).
[49]G. Miguez-Macho, G. L. Stenchikov, and A. Robock, "Spectral nudging to eliminate the effects of domain position and geometry in regional climate model simulations," J. Geophys. Res.: Atmos. **109**, D13104, https://doi.org/10.1029/2003jd004495 (2004).
[50]B. Rockel, C. L. Castro, R. A. Pielke, Sr., H. von Storch, and G. Leoncini, "Dynamical downscaling: Assessment of model system dependent retained and added variability for two different regional climate models," J. Geophys. Res.: Atmos. **113**, D21107, https://doi.org/10.1029/2007jd009461 (2008).
[51]M. Schubert-Frisius, F. Feser, H. von Storch, and S. Rast, "Optimal spectral nudging for global dynamic downscaling," Mon. Weather Rev. **145**, 909–927 (2017).
[52]P. C. Di Leoni, A. Mazzino, and L. Biferale, "Inferring flow parameters and turbulent configuration with physics-informed data assimilation and spectral nudging," Phys. Rev. Fluids **3**, 104604 (2018).
[53]P. C. Di Leoni, A. Mazzino, and L. Biferale, "Synchronization to big data: Nudging the Navier-Stokes equations for data assimilation of turbulent flows," Phys. Rev. X **10**, 011023 (2020).
[54]D. Rey, M. Eldridge, M. Kostuk, H. D. Abarbanel, J. Schumann-Bischoff, and U. Parlitz, "Accurate state and parameter estimation in nonlinear systems with sparse observations," Phys. Lett. A **378**, 869–873 (2014).
[55]D. Pazó, A. Carrassi, and J. M. López, "Data assimilation by delay-coordinate nudging," Q. J. R. Metereol. Soc. **142**, 1290–1299 (2016).
[56]Z. An, D. Rey, J. Ye, and H. D. I. Abarbanel, "Estimating the state of a geophysical system with sparse observations: Time delay methods to achieve accurate initial states for prediction," Nonlinear Process. Geophys. **24**, 9–22 (2017).
[57]R. C. Gilbert, M. B. Richman, T. B. Trafalis, and L. M. Leslie, "Machine learning methods for data assimilation," in *Computational Intelligence in Architecturing Complex Engineering Systems* (ASME Press, 2010), pp. 105–112.
[58]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).
[59]S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," Int. J. Uncertainty, Fuzziness Know.-Based Syst. **6**, 107–116 (1998).
[60]Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis, "Data-assisted reduced-order modeling of extreme events in complex dynamical systems," PLoS One **13**(5), e0197704 (2018).
[61]X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar, "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," arXiv:2001.11086 (2020).
[62]J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," Chaos **27**, 121102 (2017).
[63]A. Chashchin, M. Botchev, I. Oseledets, and G. Ovchinnikov, "Predicting dynamical system evolution with residual neural networks," arXiv:1910.05233 (2019).
[64]Z. Chen and D. Xiu, "On generalized residue network for deep learning of unknown dynamical systems," arXiv:2002.02528 (2020).
[65]P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Networks **126**, 191–217 (2020).
[66]Z. Zhang, M. Hou, F. Zhang, and C. R. Edwards, "An LSTM based Kalman filter for spatio-temporal ocean currents assimilation," in *Proceedings of the International Conference on Underwater Networks and Systems* (Association for Computing Machinery, 2019), pp. 1–7.
[67]J. Jin, H. X. Lin, A. Segers, Y. Xie, and A. Heemink, "Machine learning for observation bias correction with application to dust storm data assimilation," Atmos. Chem. Phys. **19**, 10009–10026 (2019).
[68]K. Loh, P. S. Omrani, and R. van der Linden, "Deep learning and data assimilation for real-time production prediction in natural gas wells," arXiv:1802.05141 (2018).
[69]S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems* (MIT Press, 2015), pp. 802–810.
[70]F. J. Gonzalez and M. Balajewicz, "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems," arXiv:1808.01346 (2018).
[71]A. Mohan, D. Daniel, M. Chertkov, and D. Livescu, "Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence," arXiv:1903.00033 (2019).
[72]R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders," arXiv:2002.00470 (2020).
[73]K. Lee and K. T. Carlberg, "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," J. Comput. Phys. **404**, 108973 (2020).
[74]N. B. Erichson, M. Muehlebach, and M. W. Mahoney, "Physics-informed autoencoders for Lyapunov-stable fluid flow prediction," arXiv:1905.10866 (2019).
[75]L. Agostini, "Exploration and prediction of fluid dynamical systems using autoencoder technology," Phys. Fluids **32**, 067103 (2020).
[76]P. Wu, J. Sun, X. Chang, W. Zhang, R. Arcucci, Y. Guo, and C. C. Pain, "Data-driven reduced order model with temporal convolutional neural network," Comput. Methods Appl. Mech. Eng. **360**, 112766 (2020).
[77]I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (MIT Press, 2014), pp. 2672–2680.

[78]J. Amirian, W. Van Toll, J.-B. Hayet, and J. Pettré, "Data-driven crowd simulation with generative adversarial networks," in *Proceedings of the 32nd International Conference on Computer Animation and Social Agents* (Association for Computing Machinery, 2019), pp. 7–10.

[79]E. N. Lorenz, "Predictability: A problem partly solved," in *Proceedings of Seminar on Predictability* (ECMWF, 1996), Vol. 1.

[80]E. N. Lorenz and K. A. Emanuel, "Optimal sites for supplementary weather observations: Simulation with a small model," J. Atmos. Sci. **55**, 399–414 (1998).

[81]A. Simmons, R. Mureau, and T. Petroliagis, "Error growth and estimates of predictability from the ECMWF forecasting system," Q. J. R. Metereol. Soc. **121**, 1739–1771 (1995).

[82]H. Abarbanel, *Predicting the Future: Completing Models of Observed Complex Systems* (Springer, 2013).

[83]W. G. Whartenby, J. C. Quinn, and H. D. Abarbanel, "The number of required observations in data assimilation for a shallow-water flow," Mon. Weather Rev. **141**, 2502–2518 (2013).

[84]E. J. Fertig, J. Harlim, and B. R. Hunt, "A comparative study of 4D-VAR and a 4D ensemble Kalman filter: Perfect model simulations with Lorenz-96," Tellus A **59**, 96–100 (2007).

[85]P. L. Houtekamer and H. L. Mitchell, "A sequential ensemble Kalman filter for atmospheric data assimilation," Mon. Weather Rev. **129**, 123–137 (2001).

[86]E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. A. Yorke, "A local ensemble Kalman filter for atmospheric data assimilation," Tellus A **56**, 415–428 (2004).

[87]M. Jardak, I. M. Navon, and M. Zupanski, "Comparison of sequential data assimilation methods for the Kuramoto–Sivashinsky equation," Int. J. Numer. Methods Fluids **62**, 374–402 (2010).

[88]J. L. Anderson and N. Collins, "Scalable implementations of ensemble filter algorithms for data assimilation," J. Atmos. Oceanic Technol. **24**, 1452–1463 (2007).

[89]P. Sakov and P. R. Oke, "A deterministic formulation of the ensemble Kalman filter: An alternative to ensemble square root filters," Tellus A **60**, 361–371 (2008).

[90]J. Blum, F.-X. Le Dimet, and I. M. Navon, "Data assimilation for geophysical fluids," in *Handbook of Numerical Analysis* (Elsevier, 2009), Vol. 14, pp. 385–441.

[91]M. J. M. Cheema and W. G. Bastiaanssen, "Local calibration of remotely sensed rainfall from the TRMM satellite for different periods and spatial scales in the Indus Basin," Int. J. Remote Sens. **33**, 2603–2627 (2012).

[92]L. Kadanoff, D. Lohse, J. Wang, and R. Benzi, "Scaling and dissipation in the GOY shell model," Phys. Fluids **7**, 617–629 (1995).

[93]P. D. Ditlevsen and I. Mogensen, "Cascades and statistical equilibrium in shell models of turbulence," Phys. Rev. E **53**, 4785 (1996).

[94]A. Gluhovsky and C. Tong, "The structure of energy conserving low-order models," Phys. Fluids **11**, 334–343 (1999).

[95]P. D. Ditlevsen, *Turbulence and Shell Models* (Cambridge University Press, New York, 2010).

[96]A. Gluhovsky and K. Grady, "Effective low-order models for atmospheric dynamics and time series analysis," Chaos **26**, 023119 (2016).

[97]M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," J. Comput. Phys. **357**, 125–141 (2018).

[98]S. Pawar, S. E. Ahmed, O. San, and A. Rasheed, "Data-driven recovery of hidden physics in reduced order modeling of fluid flows," Phys. Fluids **32**, 036602 (2020).

[99]W. Li, W. S. Rosenthal, and G. Lin, "Trimmed ensemble Kalman filter for nonlinear and non-Gaussian data assimilation problems," arXiv:1808.05465 (2018).

[100]J. L. Anderson, "A non-Gaussian ensemble filter update for data assimilation," Mon. Weather Rev. **138**, 4186–4198 (2010).

[101]A. Apte, M. Hairer, A. M. Stuart, and J. Voss, "Sampling the posterior: An approach to non-Gaussian data assimilation," Physica D **230**, 50–64 (2007).

[102]G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," J. Geophys. Res.: Oceans **99**, 10143–10162, https://doi.org/10.1029/94jc00572 (1994).

[103]G. Burgers, P. Jan van Leeuwen, and G. Evensen, "Analysis scheme in the ensemble Kalman filter," Mon. Weather Rev. **126**, 1719–1724 (1998).