Physics of Fluids

Non-autoregressive time-series methods for stable parametric reduced-order models

Cite as: Phys. Fluids **32**, 087115 (2020); https://doi.org/10.1063/5.0019884
Submitted: 26 June 2020 . Accepted: 07 August 2020 . Published Online: 25 August 2020

Romit Maulik, Bethany Lusch, and Prasanna Balaprakash

Physics of Fluids ARTICLE scitation.org/journal/phf

Non-autoregressive time-series methods for stable parametric reduced-order models

Cite as: Phys. Fluids 32, 087115 (2020); doi: 10.1063/5.0019884
Submitted: 26 June 2020 • Accepted: 7 August 2020 •
Published Online: 25 August 2020

View Online Export Citation CrossMark

Romit Maulik,[1,a)] Bethany Lusch,[1] and Prasanna Balaprakash[2]

AFFILIATIONS

[1]Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, Illinois 60439, USA
[2]Argonne Leadership Computing Facility and Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, Illinois 60439, USA

[a)]Author to whom correspondence should be addressed: rmaulik@anl.gov

ABSTRACT

Advection-dominated dynamical systems, characterized by partial differential equations, are found in applications ranging from weather forecasting to engineering design where accuracy and robustness are crucial. There has been significant interest in the use of techniques borrowed from machine learning to reduce the computational expense and/or improve the accuracy of predictions for these systems. These rely on the identification of a basis that reduces the dimensionality of the problem and the subsequent use of time series and sequential learning methods to forecast the evolution of the reduced state. Often, however, machine-learned predictions after reduced-basis projection are plagued by issues of stability stemming from incomplete capture of multiscale processes as well as due to error growth for long forecast durations. To address these issues, we have developed a *non-autoregressive* time series approach for predicting linear reduced-basis time histories of forward models. In particular, we demonstrate that non-autoregressive counterparts of sequential learning methods such as long short-term memory (LSTM) considerably improve the stability of machine-learned reduced-order models. We evaluate our approach on the inviscid shallow water equations and show that a non-autoregressive variant of the standard LSTM approach that is bidirectional in the principal component directions obtains the best accuracy for recreating the nonlinear dynamics of partial observations. Moreover—and critical for many applications of these surrogates—inference times are reduced by three orders of magnitude using our approach, compared with both the equation-based Galerkin projection method and the standard LSTM approach.

https://doi.org/10.1063/5.0019884

I. INTRODUCTION

Recently, researchers have shown sustained interest in using machine learning methods for bypassing traditional numerical methods.[1–11] This is due to the promise such methods hold in multiple applications ranging from engineering design to climate modeling, where forward model solves rely on nonlinear partial differential equations (PDEs). Frequently, these systems exhibit multiscale and advective behavior, which leads to very fine spatiotemporal discretization requirements. Consequently, PDE-based solutions of these systems become computationally expensive, causing a significant bottleneck in design and forecast tasks.[12] Data-driven reduced-order methods (ROMs) are promising since they allow for rapid predictions of nonlinear dynamics unencumbered by the limitations of numerical discretizations.[3,13] In almost all ROM applications, forecasts must be conditioned on time and several control parameters such as the initial conditions or the physical properties of the governing laws. Moreover, most systems need to be integrated in time for a large duration, and stable predictions throughout the lifetime of the dynamical system are essential. We address issues related to the stability of conditional surrogates by introducing physics-informed non-autoregressive methods for time series prediction. To that end, we propose a novel long short-term memory (LSTM) network method that performs bidirectional[14] gating in the dimension of the principal component analysis (PCA) coefficients while being globally connected in time. This method is compared with standard techniques such as the traditional LSTM,[15] a non-autoregressive version of a temporal convolutional network,[16] and a non-autoregressive multilayered perceptron. Our testing results show lower testing and reconstruction errors from

the proposed method as well as significant improvement in model stability compared with that of the traditional LSTM. Assessments are also made against the Galerkin projection (GP),[17] and our proposed approach is seen to provide more accurate results with shorter inference times. To summarize, the contributions of this article are as follows:

- A novel non-autoregressive LSTM-based method is proposed that performs bidirectional gating in the PCA dimension while remaining fully connected in time for predicting the spatiotemporal dynamics of the shallow water equations.
- We demonstrate that the proposed method can provide more accurate results for learning the trajectory of the nonlinear dynamical systems in addition to providing much lower inference times and exhibiting greater stability.
- We advance the state of the art for forecasting nonlinear dynamical systems from the point of view of stability and the ability to handle incomplete data.

II. RELATED WORK

Neural networks have been used for ROMs for decades. One of the earliest examples[18] used a simple fully connected network for forecasting meteorological information. More recently, researchers have incorporated a single-layered feed-forward neural network into a nonlinear dynamical system and built a surrogate model for a high-dimensional aerodynamics problem;[19] radial basis function networks have been used to make forecasts for a nonlinear unsteady aerodynamics task;[20,21] and a simple fully connected network has been used for learning the dynamics of an advection-dominated system.[22,23] Data generated from PDE simulations can often be interpreted as images on a square grid, so convolutional neural networks have also been applied.[24–28] Forecast models constructed from data alone are often categorized as non-intrusive, and in recent times, several such techniques have been proposed to address situations with insufficient knowledge of the underlying governing equations.[23,29–36]

Although other ways can be used to reduce the dimensionality of dynamical system data, using the PCA projection means that the latent space can be interpreted as evolving coefficients in terms of physically relevant spectral content. Since 2018, there has been major growth in the use of LSTMs after projecting dynamical systems into latent space.[26,34,37–44] Since errors from the neural network model accumulate, the autoregressive approach may be unstable for long-term predictions. Most studies do not report on the robustness and stability of their trained networks for long prediction horizons.

In addition, such studies assume that all dependent variables (i.e., the full state of the dynamical system) are observable. This assumption limits the application of these methods to realistic forecasting scenarios where not all of the physical processes are observable. In practice, training recurrent networks for adhering to physical manifolds is nontrivial,[45] and it is made doubly difficult by having incomplete access to all the relevant information. Recent articles have tried to address these limitations[46–48] by constructing physics-aware networks that learn conservation laws, but their extension to complicated systems with incomplete observations remains unclear.

The method proposed in this article represents an improvement in the state of the art for nonintrusive ROMs based on recurrent neural networks. We find that a non-autoregressive approach improves long-term stability and inference time. It also allows a novel use of an LSTM where gating is performed in the PCA dimension instead of the time dimension. The error is further decreased by using a bidirectional LSTM. We perform a thorough analysis of the stability and robustness of the proposed framework, and we find improved performance in comparison with that of traditional methods. We also assess the performance of the framework for incomplete observations. GP, an *intrusive* and equation-based method,[17] is used as a baseline for the purpose of an additional comparison. We note that GP requires the solution of a partial differential equation in latent space and complete observations of the system dynamics. We demonstrate that our proposed method, which operates on incomplete observations, outperforms GP as well.

III. METHODS

The parameterized forecasting of a high-dimensional advection-dominated problem can be formulated as a supervised sequential learning problem. A common approach for solving this problem comprises four steps:

1. Collect time series data $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_N} \in \mathbb{R}^m$, evenly spaced in time. For example, the data could be from a spatiotemporal system governed by a PDE. However, we do not assume that we have "complete information" in the sense of measuring all the dependent variables of the original system.
2. Reduce the dimensionality of the problem by projecting data into the latent space defined by the first $\mathbf{r}$ PCA components of the data, producing $\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_N} \in \mathbb{R}^{\mathbf{r}}$.
3. Train a time series model in this latent space. The inputs are a historical sequence of inputs $\mathbf{z_{n-k}}$, $\mathbf{z_{n-k+1}}$, …, $\mathbf{z_n}$ and control variables w, such as the initial conditions. A trained model is tasked with predicting the sequence of outputs $\mathbf{z_{n+1}}$, $\mathbf{z_{n+2}}$, …, $\mathbf{z_{n+T}}$, where $\mathbf{T}$ is the total number of forecast steps.
4. Project the predictions $\mathbf{z_t}$ at any future time step $\mathbf{t}$ back to $\mathbb{R}^m$ using the saved PCA bases to assess reconstruction fidelity.

A. Galerkin projection

For comparison of all our machine-learned predictive strategies, we utilize the Galerkin projection[49,50] methodology, which has been used extensively for advection-dominated systems including the shallow water equations.[51,52] Briefly, GP involves the projection of the governing partial differential equations onto the truncated PCA space. The orthonormality of the PCA bases leads to a significant reduction in the number of coupled ordinary differential equations (for instance, the retention of $\mathbf{r}$ basis vectors implies that $\mathbf{r}$ coupled systems must be solved). However, some severe limitations are associated with this approach that hamper its utility as a surrogate model for dynamical systems. First, the utilization of GP necessitates complete observation of all dependent variables in the system. Second, the projection of the governing equations to the PCA space leads to a drop in accuracy since higher-order interactions between the truncated PCA bases are lost. Third, when the number of retained components grows, the computational cost of

GP becomes prohibitive.[53] These factors have contributed to growing interest in the use of machine learning methods for time series data for bypassing equation-based surrogates. GP results serve as a benchmark comparison for our proposed method against a state-of-the-art analytical technique, and a full formulation of the same for this test case may be found in our previous work.[54]

B. Time-series learning methods

Here, we introduce some common methods for the prediction of time series data, such as the LSTM network, and we describe our proposed non-autoregressive adaptation to these methods to achieve our goal of stable and accurate predictions on test datasets. Table I shows the nomenclature adopted in this study for the various time-series learning methods.

1. Autoregressive methods

Our first method is the traditional LSTM network.[15] In this method, gating is applied in the time dimension, and the framework seeks to predict the time-varying coefficients of the PCA bases. In addition, the outputs of the network are fed back into the window required for predicting the next step. This *sliding-window* prediction strategy is denoted *autoregressive*. A vast majority of nonintrusive surrogate modeling strategies employ this methodology for one-step-ahead prediction of dynamics. The choice is motivated primarily by the methodological similarities with most ordinary differential integration techniques,[55] which integrate dynamics one step at a time because of issues of *numerical* stability. We denote this method A-LSTM-T (short for autoregressive LSTM with gating in time and prediction in PCA space). The evolution equations for a standard LSTM are given as follows:

TABLE I. Methods starting with "A-" are those where outputs are fed back into the network for recursive prediction, "NA-" methods directly forecast all dynamics at once, and "T/P" indicates which dimension is being gated (time/PCA, respectively). Here, **N** refers to the total number of timesteps in the time-series and **k** refers to an input window of available data.

Method	Gating space	Gating type	Prediction steps
	Autoregressive		
A-LSTM-T	Time	Standard	1
A-LSTM-T-R	Time	Standard	2
	Non-autoregressive		
NA-LSTM-T	Time	Standard	**N–k**
NA-TCN	N/A	N/A	**N–k**
NA-LSTM-P	PCA	Standard	**N–k**
NA-BLSTM-P	PCA	Bidirectional	**N–k**
NA-MLP	N/A	N/A	**N–k**
	Analytical		
GP	N/A	N/A	1

$$\begin{aligned}
\text{input gate: } \boldsymbol{G}_i &= \boldsymbol{\varphi}_S \circ \mathcal{F}_i^{N_c}(\mathbf{z_n}),\\
\text{forget gate: } \boldsymbol{G}_f &= \boldsymbol{\varphi}_S \circ \mathcal{F}_f^{N_c}(\mathbf{z_n}),\\
\text{output gate: } \boldsymbol{G}_o &= \boldsymbol{\varphi}_S \circ \mathcal{F}_o^{N_c}(\mathbf{z_n}),\\
\text{internal state: } \boldsymbol{s}_n &= \boldsymbol{G}_f \odot \boldsymbol{s}_{n-1} + \boldsymbol{G}_i \odot \left(\boldsymbol{\varphi}_T \circ \mathcal{F}_{is}^{N_c}(\mathbf{z})\right),\\
\text{output: } \mathbf{h}_{n+1} &= \boldsymbol{G}_o \circ \boldsymbol{\varphi}_T(\boldsymbol{s}_n),
\end{aligned} \tag{1}$$

where $\mathbf{z_n}$ is an input at a current time step and $\mathbf{a}\circ\mathbf{b}$ refers to a Hadamard product of two vectors. The above set of operations are *unrolled* in the temporal dimension to allow for the effect of $\mathbf{z_{n-k}}$, $\mathbf{z_{n-k+1}}$, …, $\mathbf{z_{n-1}}$ on $\mathbf{z_n}$ for making a prediction for $\mathbf{z_{n+1}}$. Note that $\boldsymbol{\varphi}_S$ and $\boldsymbol{\varphi}_L$ refer to tangent sigmoid and tangent hyperbolic activation functions, respectively, and N_c is the number of hidden layer units in the LSTM network. $\mathcal{F}^n$ refers to a linear operation given by a matrix multiplication and subsequent bias addition, i.e.,

$$\mathcal{F}^n(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B}, \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{B} \in \mathbb{R}^n$ for $\mathbf{x} \in \mathbb{R}^m$. Conventionally, the output of the standard LSTM $\mathbf{h_{n+1}}$ may be fed into another set of LSTM operations if multiple cells are stacked. If there is only one cell or these operations represent those for the final one, the output is acted upon by another operation akin to a linear operation followed by tangent sigmoid activation to obtain $\mathbf{z_{n+1}}$, i.e.,

$$\mathbf{z_{n+1}} = \varphi_T(\mathcal{F}_{op}^{\mathbf{r}}\mathbf{h_{n+1}}). \tag{3}$$

A key disadvantage of the autoregressive time series models is that they suffer from problems related to error propagation during recursive predictions. This often results from a lack of an *analytical* notion of stability[43] as opposed to the use of an equation-based method where several numerical techniques (such as filtering or adaptive time-stepping) may be used to ensure that error growth is arrested. Some recent attempts at characterizing the chaotic behavior of the A-LSTM-T include a study to assess the Lyapunov spectra of these types of networks[56] to quantify the error propagation rate. Other time-series methods have also looked at using the Lyapunov coefficient[57] for accurate evolution of dynamical systems over long horizons.

Therefore, we consider a modified version of the A-LSTM-T method where the training involves a metamodeling strategy allowing for output feedback in the training process. The computational graph is set up in such a way that any error propagation due to prediction feedback is penalized. This may be represented as

$$\begin{aligned}
\mathbf{z_{n+1}} &= \mathcal{M}_A(\mathbf{z_{n\text{-}k}}, \mathbf{z_{n\text{-}k+1}}, \ldots, \mathbf{z_n}),\\
\mathbf{z_{n+2}} &= \mathcal{M}_A(\mathbf{z_{n\text{-}k+1}}, \mathbf{z_{n\text{-}k+2}}, \ldots, \mathbf{z_{n+1}}),
\end{aligned} \tag{4}$$

where $\mathcal{M}_A$ is an aggregation of the LSTM operations and the computation of $\mathbf{z_{n+2}}$ depends on the *prediction* of $\mathbf{z_{n+1}}$ by the current state of the network. We note that this technique has previously been utilized for training latent space representations for nonlinear dynamical systems.[58] We assess whether it can enhance the stability of the standard A-LSTM-T. We denote this method as A-LSTM-T-R. Both A-LSTM-T and A-LSTM-T-R require a window of inputs to make a one-step prediction. In addition, parameter information (i.e., w) is concatenated to this window-augmented state vector. To summarize, our training dataset for the autoregressive methods has multiple examples of inputs of a window of state vectors. The output

from these methods is a forecast of the state of the dynamical system at the next time step.

2. Non-autoregressive methods

Here, we present non-autoregressive methods that help improve the stability and accuracy of surrogate models. For this reason, the methods are prepended with "NA." The equations of a general non-autoregressive method are given by a *direct prediction* as follows:

$$\mathbf{z_{n+1}}, \mathbf{z_{n+2}}, \ldots, \mathbf{z_{n+T}} = \mathcal{M}_{NA}(\mathbf{z_{n-k}}, \mathbf{z_{n-k+1}}, \ldots, \mathbf{z_n}), \tag{5}$$

where $\mathcal{M}_{NA}$ is a non-autoregressive map. A schematic outlining the difference between autoregressive and non-autoregressive methods is provided in Fig. 1.

Our first approach is to introduce the NA counterpart of the A-LSTM-T method. Here, a standard LSTM is configured to return a sequence of all forecasts, given a burn-in sequence of $\mathbf{k}$ inputs and parameter information. A direct prediction of the dynamics precludes the necessity for any autoregressive feedback. We denote this method NA-LSTM-T, and the sole difference from the autoregressive methods of Sec. III B 1 is due to the final operation on the LSTM output, i.e.,

$$\mathbf{z_{NA}} = \varphi_T(\mathcal{F}_{op}^{\mathbf{r\times N\text{-}k}}\mathbf{h_{n+1}}), \tag{6}$$

where $\mathbf{z_{NA}}$ is a vector that stacks the states at different time steps into one target. Effectively, we predict all future states at once by managing the output dimension of the standard LSTM.

More importantly, the non-autoregressive formulation allows one to explore alternative strategies in terms of the interpretation of the dataset. For instance, we may interpret the dataset to be *sequential in PCA space* rather than time. A framework leveraging this interpretation can be devised simply by switching the gating dimension of the dataset. This aligns with the sequential nature of the PCA coefficients in terms of the proportion of variance capture of the dataset. The first method that leverages this is thus called NA-LSTM-P. The equations of the NA-LSTM-P method are given as follows. First, let us consider a matrix $\mathbf{Z} \in \mathbb{R}^{\mathbf{k\times r}}$ where each row corresponds to the $\mathbf{r}$-dimensional reduced state at each time step k of the burn-in window $\mathbf{k}$. We may transpose this matrix to obtain $\mathbf{Z'} \in \mathbb{R}^{\mathbf{r\times k}}$ where each row now corresponds to the PCA coefficient r (of a total of $\mathbf{r}$ coefficients) of our reduced state. Our NA-LSTM-P method would then perform the following operations:

$$\begin{aligned}
\text{input gate: } \boldsymbol{G}_i &= \boldsymbol{\varphi}_S \circ \mathcal{F}_i^{N_c}(\mathbf{z'}_r),\\
\text{forget gate: } \boldsymbol{G}_f &= \boldsymbol{\varphi}_S \circ \mathcal{F}_f^{N_c}(\mathbf{z'}_r),\\
\text{output gate: } \boldsymbol{G}_o &= \boldsymbol{\varphi}_S \circ \mathcal{F}_o^{N_c}(\mathbf{z'}_r),\\
\text{internal state: } \boldsymbol{s}_r &= \boldsymbol{G}_f \odot \boldsymbol{s}_{r-1} + \boldsymbol{G}_i \odot \left(\boldsymbol{\varphi}_T \circ \mathcal{F}_{is}^{N_c}(\mathbf{z'}_r)\right),\\
\text{output: } \mathbf{h}_{r+1} &= \boldsymbol{G}_o \circ \boldsymbol{\varphi}_T(\boldsymbol{s}_r),
\end{aligned} \tag{7}$$

where $\mathbf{z'}_r \in \mathbb{R}^{\mathbf{k}}$ is row r of $\mathbf{r}$ rows in matrix $\mathbf{Z'}$. The above set of operations are *unrolled* in the *PCA dimension* to allow for the effect of history. At this point, we have interpreted data in the PCA dimension to be sequential, which align with the well-known variance-ordered nature of principal components obtained from snapshot data. We may now use this directional information to predict the dynamics at all future time steps at once (contained in the vectors $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h_r}$). This is represented as

$$\mathbf{Z_{NA}} = \varphi_T(\mathcal{F}_{op}^{\mathbf{N\text{-}k}}[\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_{r+1}}]), \tag{8}$$

where the double brackets imply column vector concatenation. The final output of this framework, $\mathbf{Z_{NA}} \in \mathbb{R}^{\mathbf{N\text{-}k\times r}}$, contains all the information of the evolution of the $\mathbf{r}$-dimensional state for all future time steps $\mathbf{N–k}$. Note that the length of $\mathbf{h}_r \in \mathbb{R}^{N_c}$ is a function of the number of hidden-layer neurons N_c in this LSTM cell. In a manner similar to that demonstrated in the autoregressive case, parameter information w is concatenated into the gating (i.e., PCA) dimension. We then extend the implementation of the NA-LSTM-P formulation allowing for a bidirectional gating mechanism in the PCA dimension.[14] Through the use of this augmentation, any output $\mathbf{h}_r$ is affected by all inputs $\mathbf{z'_1}, \mathbf{z'_2}, \ldots, \mathbf{z'_r}$. This decision is physics-informed due to the common knowledge of energy interchange

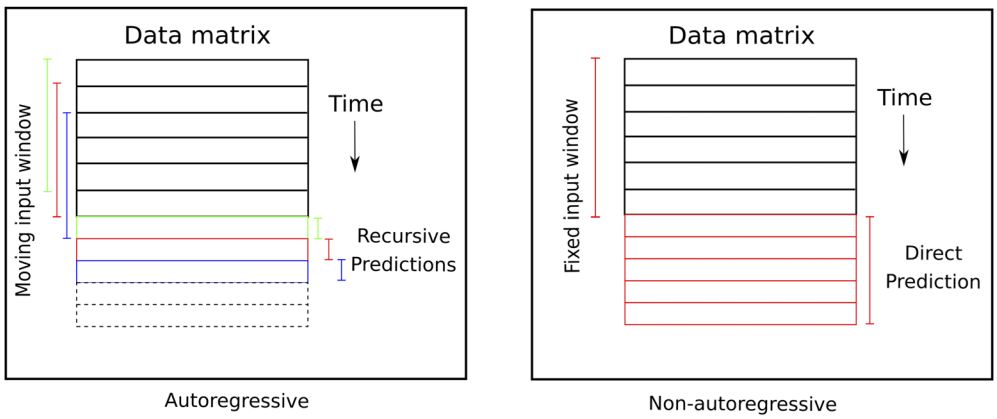

FIG. 1. Schematic outlining the difference between autoregressive and non-autoregressive methods for forecasting. The solid rectangles in each data matrix indicate the state of the dynamical system, and their dotted versions indicate states that need to be predicted. Non-autoregressive methods have been proposed to deal with issues of noise accumulation in regular time-series forecasting techniques.

between the different frequencies (and correspondingly, the different PCA basis vectors) in multiscale nonlinear dynamical systems.[59] We denote this method by NA-BLSTM-P.

The *sequential in PCA space* interpretation of the dataset also allows for the use of a comparable architecture given by the one-dimensional temporal convolution network,[16] where convolutions are performed on the parametric information and the PCA coefficients comprising the **k** input time steps. The reader may find an excellent review of the temporal convolutional network and its applications for reduced-order modeling in Ref. 60. As in the previous case, we then obtain a sequence of PCA coefficients for all future time steps as the output of the network. We denote this method by NA-TCN. As a baseline, we also use a fully connected network for predicting from the input time steps and PCA coefficients. In essence, the PCA coefficients for all input time steps are flattened and concatenated with parameter information to obtain an input signal that is used to directly predict a flattened vector of PCA coefficients. We denote this method by NA-MLP. To summarize this section, given a burn-in sequence of **k** inputs and control variables w, these frameworks produce the PCA coefficients for all future time steps directly.

IV. EXPERIMENTS

We describe the data generation methodology from the inviscid shallow water equations of an advection-dominated system and present the experimental results comparing different learning methods.

A. Data generation for training and inference

The inviscid shallow water equations belong to a prototypical system of equations for geophysical flows. In particular, the shallow water equations admit solutions where advection dominates dissipation and poses challenges for conventional ROMs.[59] These governing equations are given by

$$\frac{\partial(\rho\eta)}{\partial t} + \frac{\partial(\rho\eta u)}{\partial x} + \frac{\partial(\rho\eta v)}{\partial y} = 0, \tag{9}$$

$$\frac{\partial(\rho\eta u)}{\partial t} + \frac{\partial}{\partial x}\left(\rho\eta u^2 + \frac{1}{2}\rho g\eta^2\right) + \frac{\partial(\rho\eta uv)}{\partial y} = 0, \tag{10}$$

$$\frac{\partial(\rho\eta v)}{\partial t} + \frac{\partial(\rho\eta uv)}{\partial x} + \frac{\partial}{\partial y}\left(\rho\eta v^2 + \frac{1}{2}\rho g\eta^2\right) = 0, \tag{11}$$

where η corresponds to the total fluid column height; (u, v) is the fluid's horizontal flow velocity, averaged across the vertical column; g is acceleration due to gravity; and ρ is the fluid density, typically set to 1.0. Here, t, x, and y are the independent variables: time and the spatial coordinates of the two-dimensional system. Equation (9) captures the law of mass conservation, whereas Eqs. (10) and (11) denote the conservation of momentum. The initial conditions of the problem are given by

$$\rho\eta(x, y, t = 0) = 1 + e^{-\left(\frac{(x-\bar{x})^2}{2(5e+4)^2} + \frac{(y-\bar{y})^2}{2(5e+4)^2}\right)}, \tag{12}$$

$$\rho\eta u(x, y, t = 0) = 0, \tag{13}$$

$$\rho\eta v(x, y, t = 0) = 0, \tag{14}$$

i.e., a Gaussian perturbation at a particular location on the grid $[\bar{x}, \bar{y}] \equiv w$. We solve the system of equations until $t = 0.5$ with a time step of 0.001 s on a square two-dimensional grid with 64 collocation points to completely capture the advection and gradual decay of this perturbation. Note that these numbers may vary according to the forecasting and fidelity requirements of a particular problem and perturbation. An additional challenge is introduced when we seek to build predictive models solely from observations of $\rho\eta$ conditioned on w mimicking a real-world scenario where complete observations of all relevant variables (in this case, velocities) are unavailable. Equation-based models are thus impossible to construct because of the absence of information from the other variables of the partial differential equation.

Five hundred snapshots of ρ and η each are generated from 20 different vectors w obtained by Latin hypercube sampling from w. In this case, since w corresponds to the physical location of the initial Gaussian pulse, the two-dimensional domain for sampling was restricted between −0.5 and 0.5 in both x and y. These snapshots are used to obtain the global PCA bases that span all these simulations. The PCA bases constructed from field (or image) snapshot data capture information in a linear least-squares sense.[61] Linear combinations of PCA bases may be used to reconstruct the dynamics of the partial differential equation with the use of projection methods. A schematic detailing the generation of training and testing data is shown in Fig. 2. At this juncture, we provide some remarks on the nature of the system that is emulated. The initial and boundary conditions for this particular shallow-water equation experiment represent a tightly controlled traveling wave problem that is translation invariant. Different realizations of the initial condition lead

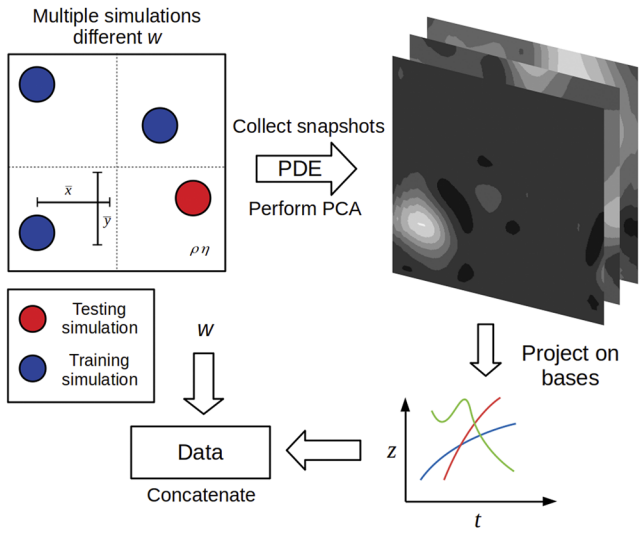

FIG. 2. Schematic outlining the generation of training and testing data for the forecasting problem. Multiple simulations are used to generate training and testing snapshots. The training snapshots are used to compute a set of PCA bases on which individual snapshots are projected to obtain time-varying PCA coefficients. These coefficients are predicted for the testing simulation by a data-driven method. Here, z, t, and $w \equiv [\bar{x}, \bar{y}]$ are the PCA coefficients, time, and control parameters, respectively.

to translationally shifted trajectories. We also note the presence of mirror symmetries with respect to $x = \bar{x}$ and $y = \bar{y}$ coupled with a rotational symmetry of $\pi/2$ radians about the origin. However, our motivation for a first assessment of our emulators on this system stems from the well-known fact that proper orthogonal decomposition (POD)-based methods are severely limited in their ability to forecast on these simple traveling-wave systems[32,62] and require special treatment with intrinsic knowledge of the flow dynamics. Contours representing the final time flow-field for various choices of w are shown in Fig. 3.

Instead of utilizing all possible PCA bases, because of issues of computational complexity (20 simulations × 500 time steps = 10 000 components), only $\mathbf{r} = 40$ PCA bases corresponding to the most energetic structures in the data are retained. Therefore, only $\mathbf{r}$ coefficients are required in order to reconstruct a field, given these bases. We note that the choice to set $\mathbf{r} = 40$ coefficients is made by assessing the reconstruction error in the transformed bases. This number allows us to reconstruct the original solution with root mean square errors (RMSEs) around the order of 1×10^{-5} and captures more than 95% of the total variance by measuring the magnitude of singular values in the PCA decomposition. In addition, the coefficients we retain are conditioned on w as well as time. This study seeks to learn the underlying trends of their evolution in time and assesses this learning for different time-series prediction methods.

For testing, assessments are made on an *unseen* w, and a successful reconstruction of the dynamics for these parameters implies that a viable surrogate has been obtained. Moreover, we assume that a short duration of observations, $\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_k}$, is available (corresponding to a first window of inputs to our methods). This burn-in window $\mathbf{k}$ corresponds to a very small observation set compared with the full 500 time steps. We set $\mathbf{k} = 20$ for all experiments. This choice was determined through manual experimentation with the objective of having the shortest burn-in duration. We observed that values less than $\mathbf{k} = 20$ led to very high deterioration in the results for each method. For training, the time sequence of each PCA component is scaled by the minimum and maximum value of all its counterparts using only the training dataset. We also note that the data generated from these finite-volume simulations can be interpreted to be images on a square grid. While this lends to the use of convolutional

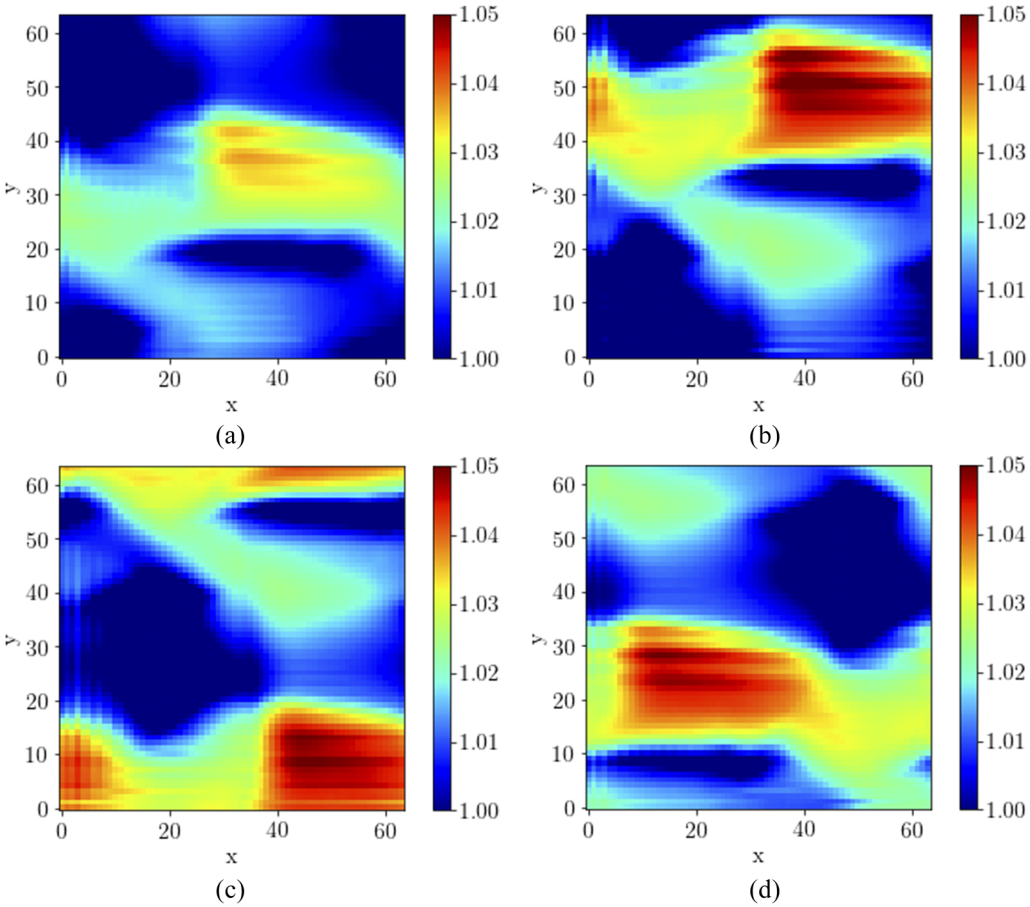

FIG. 3. Final time contours [(a)–(d)] of various choices of w. The symmetries inherent in this problem are visible as the contours are translationally and rotationally shifted across the periodic boundaries.

neural networks for modeling dynamics,[24,25,63,64] the choice of the PCA projection allows for interpretability of the evolving coefficients in terms of physically relevant spectral content.

B. Experimental setup

All the learning methods investigated here utilized only two hidden layers to allow for low training and inference times. All our data generation and trained model assessments used Python 3.6.8 and TensorFlow 1.14.0 on an Ubuntu 18.04 operating system with 8 GB of RAM.

An asynchronous hyperparameter optimization of the methods that we consider was performed on 256 Intel Knights Landing compute nodes for 6 h with the best models being chosen by their validation loss. Subsequent evaluations of the trained models were performed on an eighth-generation Intel Core-I7 processor. For optimizing the hyperparameters of the different methods, we use DeepHyper,[65] a hyperparameter search package that leverages Bayesian black-box optimization from scikit-optimize.[66] This also allows for more confident conclusions about the results of our models and removes bias due to default or manual parameter settings. While the total computational cost for obtaining the optimal hyperparameters prior to training an emulator represents a prohibitive one-time cost, we emphasize that the use of the hyperparameter search is to determine the *best-suited* emulation technique among the different methods studied in this article. In practice, the development of the ML-ROM would be satisfactory with manual tuning. Online costs, to be outlined later, demonstrate that the ML methods outperform POD-GP.

The search space for all hyperparameters was limited to the number of neurons (for each of two hidden layers), the batch size, and the learning rate. A smaller search space ensured more model evaluations to allow for effective search-space exploration. Our search-space bounds are [10 200] for the hidden-layer neurons and $[1 \times 10^{-4}, 1 \times 10^{-1}]$ for the learning rates. Batch size bounds were set according to the choice of method: autoregressive methods have batch sizes between 64 and 256, and non-autoregressive methods have batch sizes between 1 and 10. Two hyperparameter optimization experiments are performed for each method in this study. The first utilizes a standard training, and the second incorporates dropout regularization[67] for stability analyses of methods. Both searches utilize the same bounds. On average, each method was evaluated for more than 400 unique hyperparameter configurations. The high-performing hyperparameters obtained by using DeepHyper are shown in Table II. Models were trained with convergence callbacks based on training loss (less than 1×10^{-5} RMSE) or training time out (corresponding to 5000 epochs).

C. Comparison of different methods

Here, we compare the different autoregressive and non-autoregressive learning methods with GP and show that non-autoregressive methods are superior to other methods with respect to forecasting accuracy and stability.

The different methods are compared based on the mean squared error (MSE) of PCA coefficient predictions for the test simulation. We also show error variance for each method. The training loss, testing error, and field reconstruction error results are shown in Table II. First, we compare the autoregressive variants. We observe that both A-LSTM-T and the A-LSTM-T-R perform poorly. While the latter exhibits a saturation after a certain prediction horizon, the former displays compounding errors that diverge from the true trajectory. These results are reflected in their test MSEs of 2.26×10^{-1} and 8.77×10^{-3}, the two highest in our set of experiments. We also note that A-LSTM-T shows a very low training loss of around 5.03×10^{-4}, which is far lower than those obtained by the other methods; however, its testing performance is poor, hinting at issues of stability for long-term feedback predictions.

Next, we look at the non-autoregressive methods. For these methods, testing MSEs show that the proposed method NA-BLSTM-P has the lowest magnitude of error (1.26×10^{-5}) compared with NA-LSTM-T, NA-LSTM-P, NA-TCN, and NA-MLP (1.79×10^{-3}, 1.41×10^{-3}, 2.10×10^{-3}, and 1.36×10^{-3}, respectively). This may be attributed to the physics-aware nature of the gating in PCA space where information from multiple PCA coefficients interacts globally across the $\mathbf{r} = 40$ components. The gating in PCA space has a positive effect on the performance of the non-autoregressive methods, with both NA-LSTM-P and NA-BLSTM-P showing the two best test results. The sequential nature of the ordered PCA coefficients is thus leveraged successfully. In terms of training losses, NA-LSTM-P has the lowest training loss among non-autoregressive methods of 2.35×10^{-3}, whereas NA-BLSTM-P has the next best (but comparable) training loss of 2.81×10^{-3}. These are followed by NA-LSTM-T

TABLE II. High-performing hyperparameter values obtained by DeepHyper and corresponding training, test, and final-time reconstruction errors. The proposed method NA-BLSTM-P can be seen to provide the lowest mean squared error (MSE) for this problem.

Method	Neurons	Batch size	Learning rate	Training loss MSE	Testing loss MSE/variance	Field error MSE
GP	N/A	N/A	N/A	N/A	$2.85 \times 10^{-3}/6.15 \times 10^{-5}$	0.000 146
A-LSTM-T	88	152	0.000 258	0.000 503	$2.26 \times 10^{-1}/4.59 \times 10^{-3}$	0.000 794
A-LSTM-T-R	29	178	0.651	0.018 7	$8.77 \times 10^{-3}/5.61 \times 10^{-4}$	0.000 352
NA-LSTM-T	102	6	0.034 2	0.003 98	$1.79 \times 10^{-3}/2.15 \times 10^{-5}$	5.77×10^{-5}
NA-LSTM-P	88	9	0.000 258	0.002 35	$1.41 \times 10^{-3}/1.26 \times 10^{-5}$	4.12×10^{-5}
NA-BLSTM-P	145	6	0.005 23	0.002 81	$\mathbf{1.26 \times 10^{-5}/5.23 \times 10^{-6}}$	$\mathbf{3.96 \times 10^{-5}}$
NA-TCN	145	6	0.005 23	0.004 55	$2.10 \times 10^{-3}/2.02 \times 10^{-5}$	4.33×10^{-5}
NA-MLP	72	4	0.001 84	0.004 77	$1.36 \times 10^{-3}/1.18 \times 10^{-5}$	5.23×10^{-5}

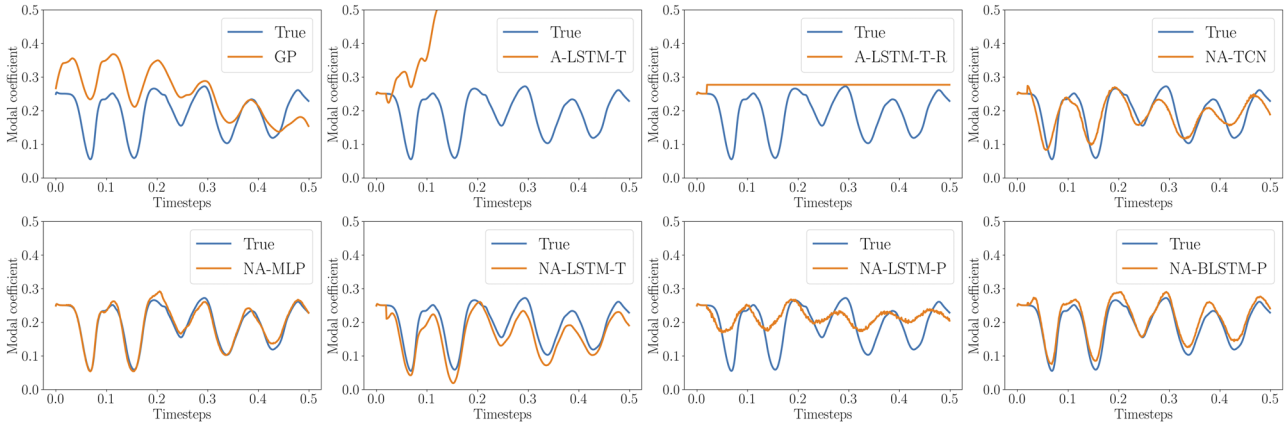

FIG. 4. Predictive ability for the assessed frameworks for PCA component 1. Non-autoregressive methods are seen to be better than their autoregressive counterparts. Note that in order to build a model, GP requires the solution of a partial differential equation in addition to greater observations from the true system.

(3.98×10^{-3}), NA-TCN (4.55×10^{-3}), and NA-MLP (4.77×10^{-3}). As expected, non-autoregressive methods exhibit lower testing errors than their autoregressive counterparts.

We also note that all the non-autoregressive methods display lower testing errors than GP does (which has a testing error of 2.85×10^{-3}). The comparison is made more favorable by the fact that GP requires an equation-based evolution of all variables in the system, whereas our data-driven methods are built solely for $\rho\eta$. In addition, GP requires the PCA of the other variables to identify the latent space for evolution. These results are displayed in Fig. 4, which shows testing predictions for the normalized coefficients of the first PCA coefficient for $\rho\eta$. Similar results are obtained for higher-order coefficients, as shown in Figs. 5–7.

We plot final time fields of $\rho\eta$ in Fig. 8. Field reconstructions, compared to the truth, show that final time predictions of non-autoregressive frameworks are more successful in stable predictions. Observe that GP proves less accurate than the NAT methods in addition to requiring an equation-based evolution of *all* variables via a set of coupled ODEs.

D. Further validation for different datasets

In this section, we compare the NA-BLSTM-P and A-LSTM-T methods for the non-intrusive model-order reduction of the viscous Burgers equation (as introduced and explained in Ref. 43). The purpose of this assessment is to determine if the non-autoregressive methods can outperform the standard LSTM for problems on which the latter have been deployed successfully. The reader is directed to the aforementioned reference for a thorough explanation of the data generation and model reduction process. In the following, we provide a high-level explanation of the training and testing datasets and the subsequent assessment.

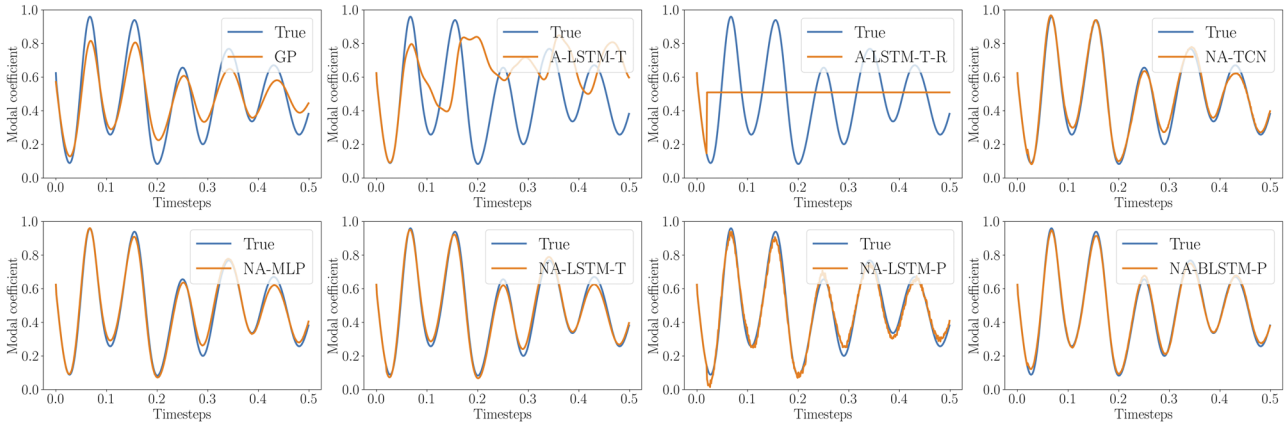

FIG. 5. Predictive ability for the assessed frameworks for PCA component 2. Non-autoregressive methods are seen to be better than their autoregressive counterparts. Note that in order to build a model, GP requires the solution of a partial differential equation in addition to greater observations from the true system.

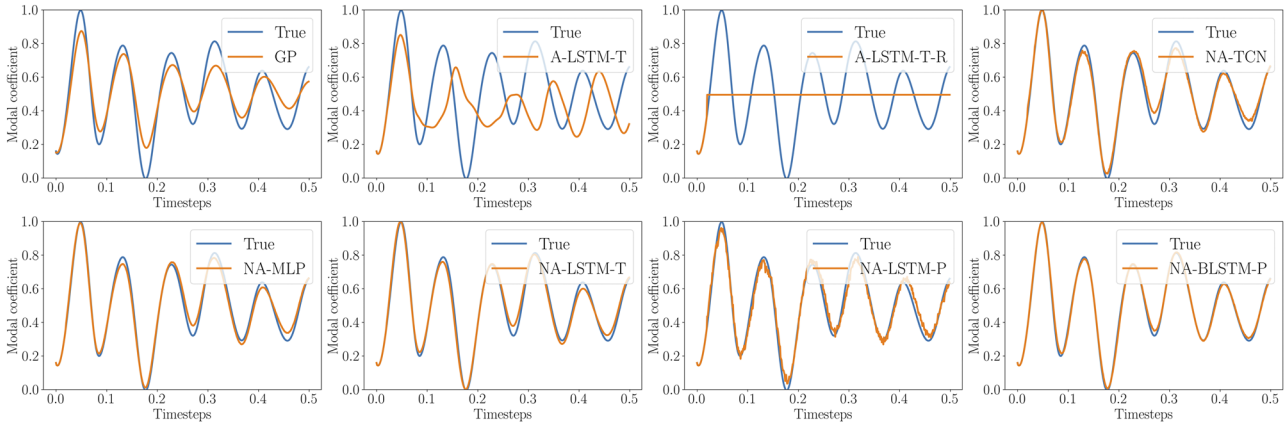

FIG. 6. Predictive ability for the assessed frameworks for PCA component 3. Non-autoregressive methods are seen to be better than their autoregressive counterparts. Note that in order to build a model, GP requires the solution of a partial differential equation in addition to greater observations from the true system.

The viscous Burgers equations described by the solution given in Eq. (36) of Maulik *et al.*[43] may be utilized to generate multiple trajectories at different viscosities. Following this, we may perform a POD-based data compression to obtain coefficients that are conditioned on time as well as the control parameter (in this case, solely the Reynolds number $Re = 1/\nu$, where ν is the viscosity). The task of our forecasting techniques would be to predict coefficient trajectories given an initial burn-in of **k** inputs and to predict for **N**–**k** time steps. For simplicity, we utilize the best architectures obtained from the deterministic hyperparameter search in Sec. IV B and use the previously defined values for **N**, **k**, and **r**. We use 18 different viscosities obtained by sampling evenly in reciprocal space (given by values of 0.000 526 32, 0.000 555 56, 0.000 588 24, 0.000 625, 0.000 666 67, 0.000 714 29, 0.000 769 23, 0.000 833 33, 0.000 909 09, 0.001, 0.001 111 11, 0.001 25, 0.001 428 57, 0.001 666 67, 0.002, 0.0025, 0.003 333 33, 0.005) for generating our training data and use an identical criterion for training termination as introduced previously. We assess our trained A-LSTM-T and NA-BLSTM-P networks on test parameters given by viscosities of 0.0004 and 0.004. The results for the former are shown in Fig. 9 in terms of the time evolution of the first four POD modal coefficients. From a visual assessment, it is seen that the NA-BLSTM-P method provides a closer agreement with the true trajectory. This is also seen in Fig. 10 for a viscosity of 0.004. The final time fields for both these testing simulations are shown in Fig. 11 where the NA-BLSTM-P is able to capture the true solution profile accurately. The A-LSTM-T approach does obtain a qualitative agreement but has a poorer accuracy. Reconstruction metrics over time are L_2 errors of 6.56×10^{-5}, 3.972×10^{-5} for A-LSTM-T and 8.08×10^{-7}, 2.82×10^{-6} for NA-BLSTM-P for the low and high viscosity cases, respectively.

E. Stability analysis

Analyzing the stability of ROM models is a critical step in the development and deployment of such models. To that end, we perturb the model through the incorporation of dropout[67] and analyze its stability with respect to these perturbations. Dropout is a

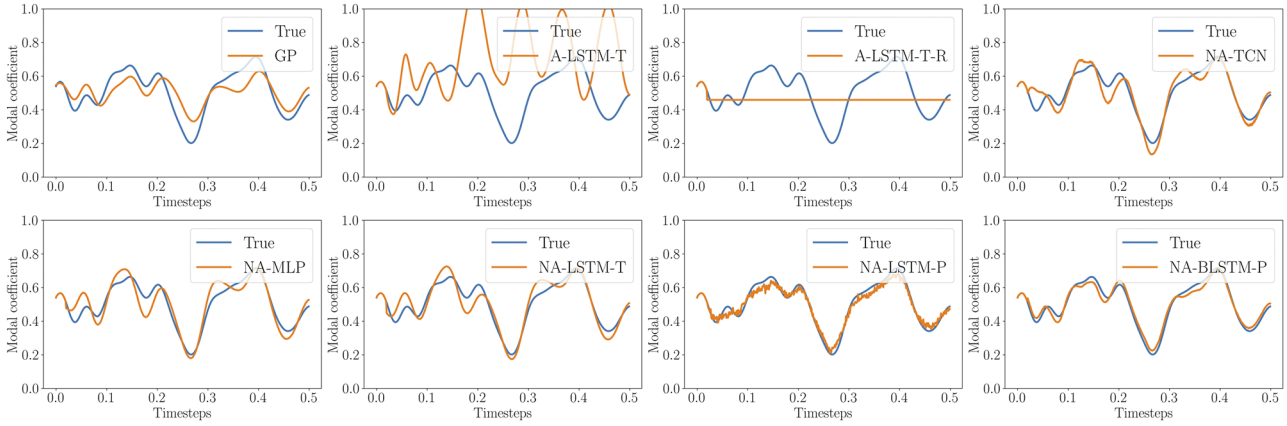

FIG. 7. Predictive ability for the assessed frameworks for PCA component 4. Non-autoregressive methods are seen to be better than their autoregressive counterparts. Note that in order to build a model, GP requires the solution of a partial differential equation in addition to greater observations from the true system.

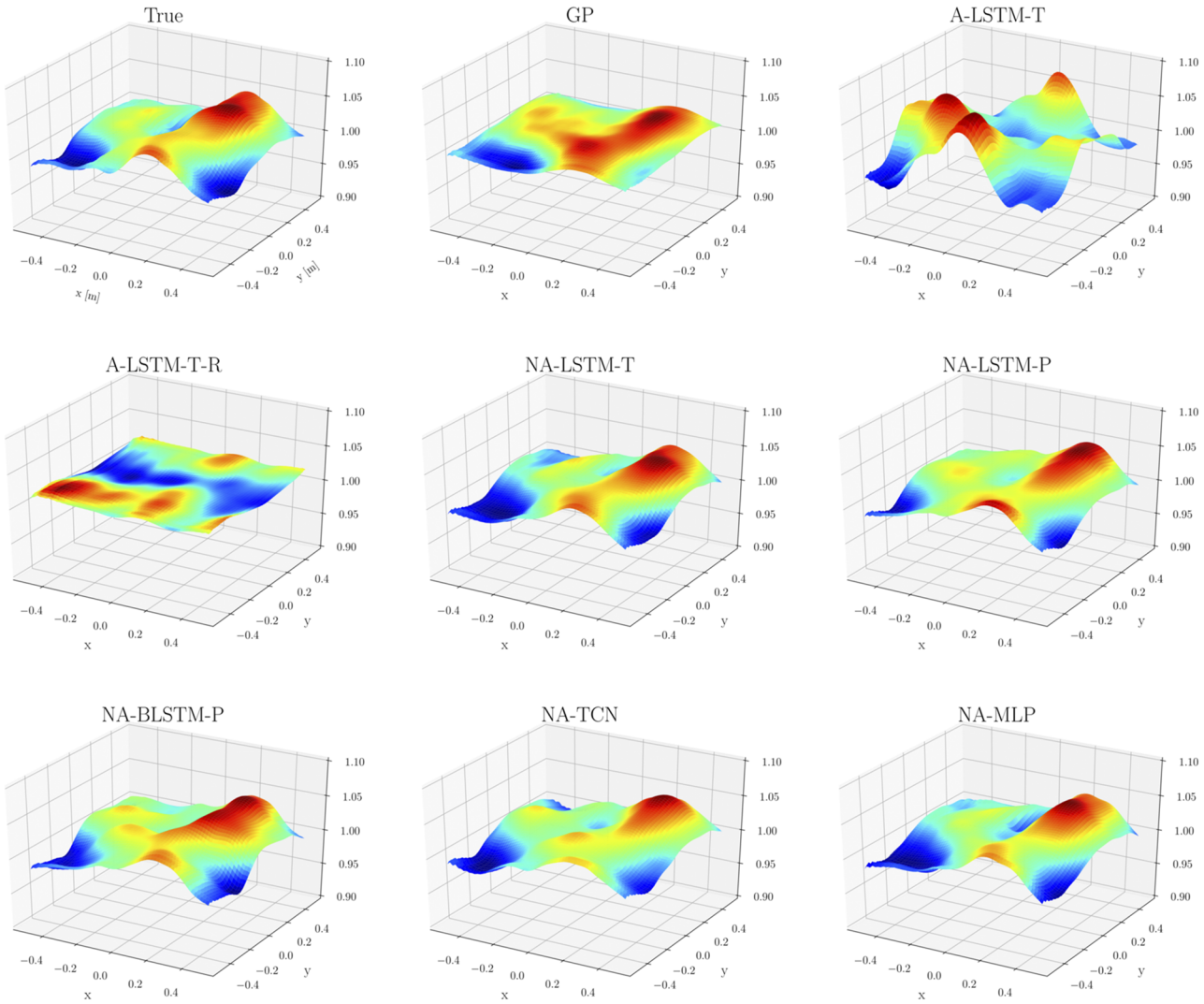

FIG. 8. Predicted fields at final time. We remind the reader that GP requires the solution of a partial differential equation in addition to greater observations from the true system to build a model.

well-known machine learning regularization technique where randomly selected neurons in a neural network are set to zero during training. This prevents a neural network from overfitting. Recently, dropout has also been used to represent model uncertainty in deep learning,[68] where random neurons are switched off during several predictions. Moments may then be generated from these multiple predictions. In this section, we shall use dropout during training to ascertain the effect of regularization on our forecast methods. Dropout during inference will be utilized to analyze the sensitivity of the surrogate models to error accumulation. Our hypothesis is that greater sensitivity to perturbations of the machine-learned models will lead to greater forecasting error. This analysis is also motivated by the fact that few reduced-order modeling techniques include notions of model-form uncertainty or sensitivity analyses during forecasting.

We perform the stability analysis with the incorporation of dropout during both training and inference with the former aimed at avoiding overfitting. The purpose of incorporating dropout during inference is to assess the effect of perturbations on the trained frameworks. This assessment can provide further validation of the conclusions made in Sec. IV C, where low training losses for autoregressive methods were not correlated with good testing performance. We believe that the error growth over a long-term prediction horizon was the cause of model inaccuracy in testing. Predictions obtained by using dropout at inference time allow for multiple time-series predictions by a trained network. A slight perturbation to the model (by

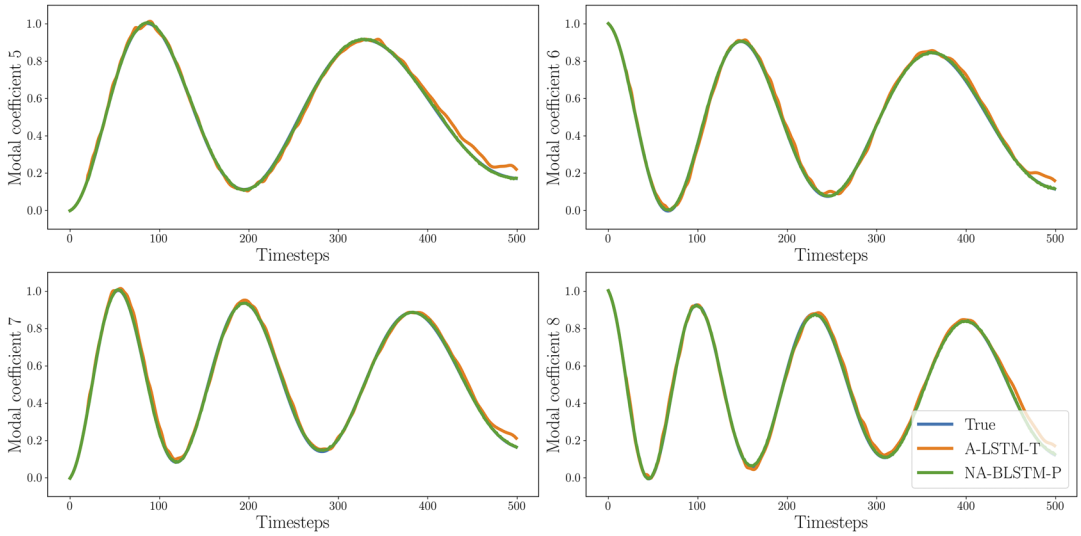

FIG. 9. Representative modal coefficient evolution for the viscous Burgers equation at a testing viscosity of 0.0004. NA-BLSTM-P (closely superimposed on the true trajectory) is seen to be superior to standard A-LSTM-T demonstrated in previous literature.[43]

switching off a subcomponent) would lead to error accumulation for autoregressive methods, whereas the non-autoregressive methods would bypass this effectively. We therefore use this technique to perform a stability analysis of the trained networks.

We run a hyperparameter search for all the learning methods with dropout for each hidden layer. As a default, we used a dropout probability of 0.2. The results are shown in Table III. Once trained, each method is tasked with predicting on the test dataset 1000 times. Then, the mean of all these predictions and their variance are calculated. The mean of the predictions is then compared with the true data (which is deterministic) to obtain testing errors. Figure 12 shows the predicted mean and variance of these 1000 inferences for all assessed methods on the first principal component. Results for this analysis on other higher-order PCA components are shown in Figs. 13–15. Note that the GP method is equation-based and *deterministic*. The A-LSTM-T method saturates as predictions evolve in time. However, the A-LSTM-T-R cannot match the right phase or frequency of the oscillations for any PCA coefficients and

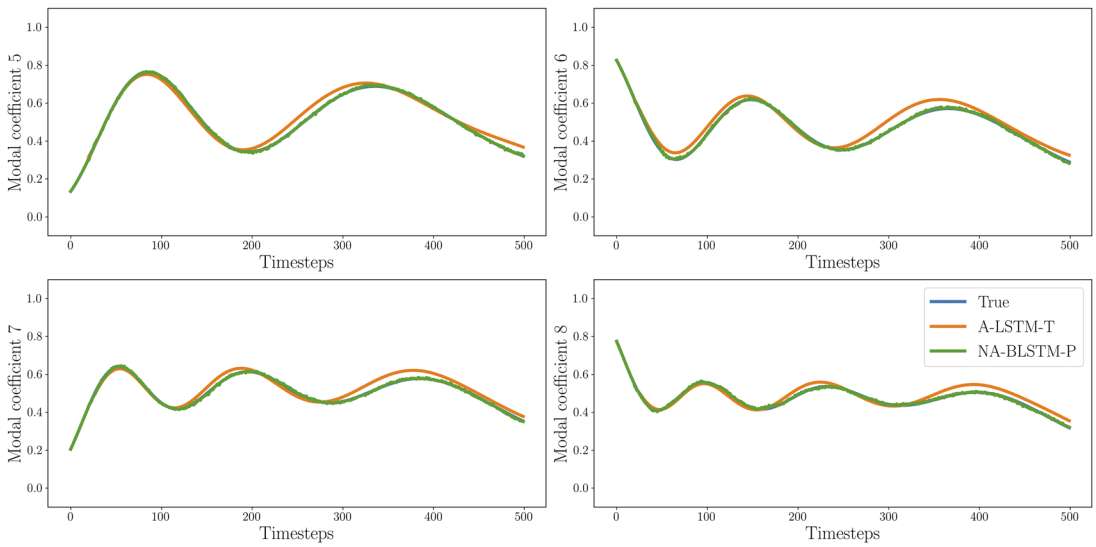

FIG. 10. Representative modal coefficient evolution for the viscous Burgers equation at a testing viscosity of 0.004. NA-BLSTM-P (closely superimposed on the true trajectory) is seen to be superior to standard A-LSTM-T demonstrated in previous literature.[43]

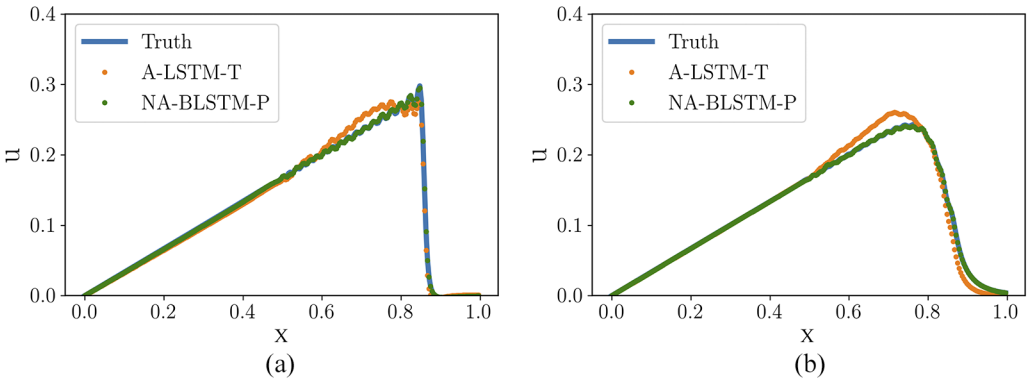

FIG. 11. Final time solution fields obtained by emulators using A-LSTM-T and NA-BLSTM-P for two testing viscosities. It is observed that the non-autoregressive method is superior and closely matches the true final time solution field.

TABLE III. Optimal hyperparameters obtained by DeepHyper with dropout during training and inference. Non-autoregressive methods are seen to perform better than their autoregressive counterparts. In addition, the proposed method (NA-BLSTM-P) can be seen to provide the lowest mean squared error (MSE) for this problem.

Method	Neurons	Batch size	Learning rate	Training loss MSE	Testing loss MSE/variance	Trainable parameters
GP	N/A	N/A	N/A	N/A	$2.85 \times 10^{-3}/6.15 \times 10^{-5}$	N/A
A-LSTM-T	197	186	2.68×10^{-2}	2.11×10^{-1}	$1.43 \times 10^{-2}/1.42 \times 10^{-3}$	508 300
A-LSTM-T-R	101	217	7.67×10^{-1}	1.86×10^{-2}	$1.39 \times 10^{-1}/9.24 \times 10^{-3}$	144 472
NA-LSTM-T	145	6	5.23×10^{-3}	4.23×10^{-3}	$1.32 \times 10^{-3}/1.06 \times 10^{-5}$	3 081 020
NA-LSTM-P	88	9	2.58×10^{-4}	3.23×10^{-3}	$1.96 \times 10^{-3}/2.06 \times 10^{-5}$	143 392
NA-BLSTM-P	145	6	5.23×10^{-3}	2.37×10^{-3}	$\mathbf{8.38 \times 10^{-4}/5.71 \times 10^{-6}}$	838 000
NA-TCN	88	9	2.58×10^{-4}	4.94×10^{-3}	$1.86 \times 10^{-3}/1.26 \times 10^{-5}$	504 544
NA-MLP	119	4	9.62×10^{-4}	4.73×10^{-3}	$1.53 \times 10^{-3}/1.44 \times 10^{-5}$	2 413 837

generates a random noisy signal. In contrast, the NA methods are able to match the phase of the predictions appropriately. These trends are repeated for the other coefficients as well. Quantitative comparisons across all coefficients are given in Table III and show that the proposed methods NA-BLSTM-P and NA-LSTM-P outperform their counterparts. We also show the number of trainable parameters for each architecture in this table. One can note that the proposed methods NA-BLSTM-P and NA-LSTM-P require a

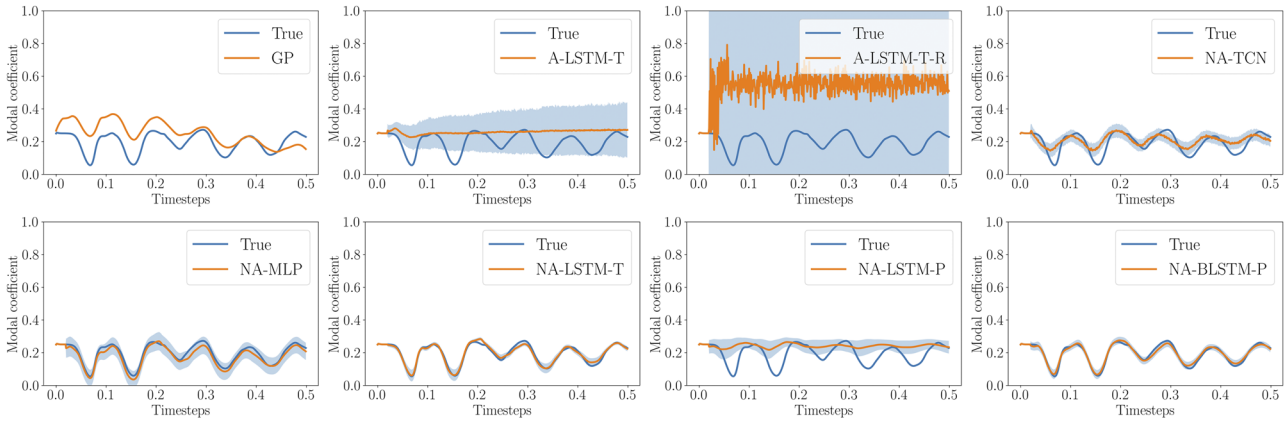

FIG. 12. Predictive ability for the assessed frameworks for PCA component 1. Note that GP requires the solution of a partial differential equation in addition to complete observations from the true system to build a model. In addition, GP is deterministic.

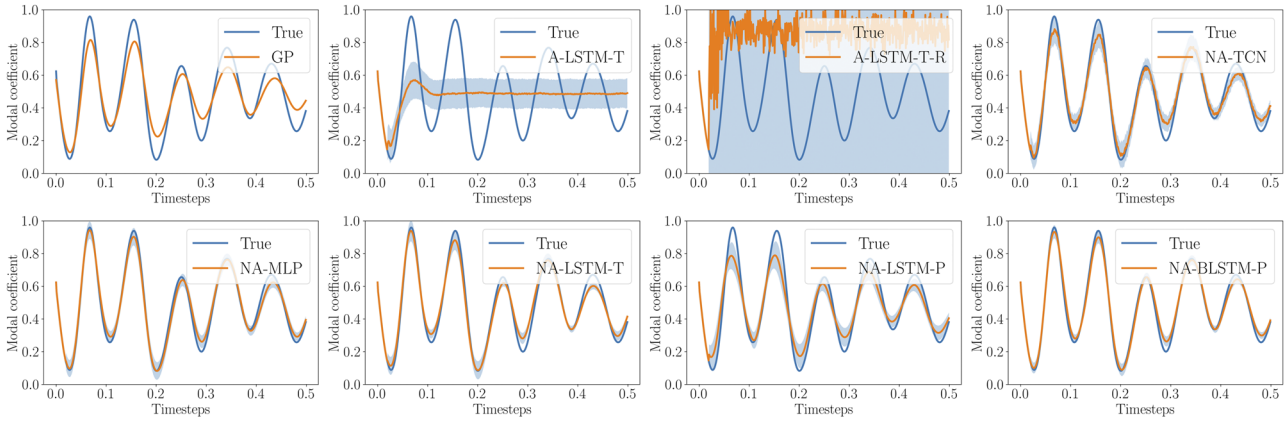

FIG. 13. Predictive ability for the assessed frameworks for PCA component 2. Note that GP requires the solution of a partial differential equation in addition to complete observations from the true system to build a model. In addition, GP is deterministic.

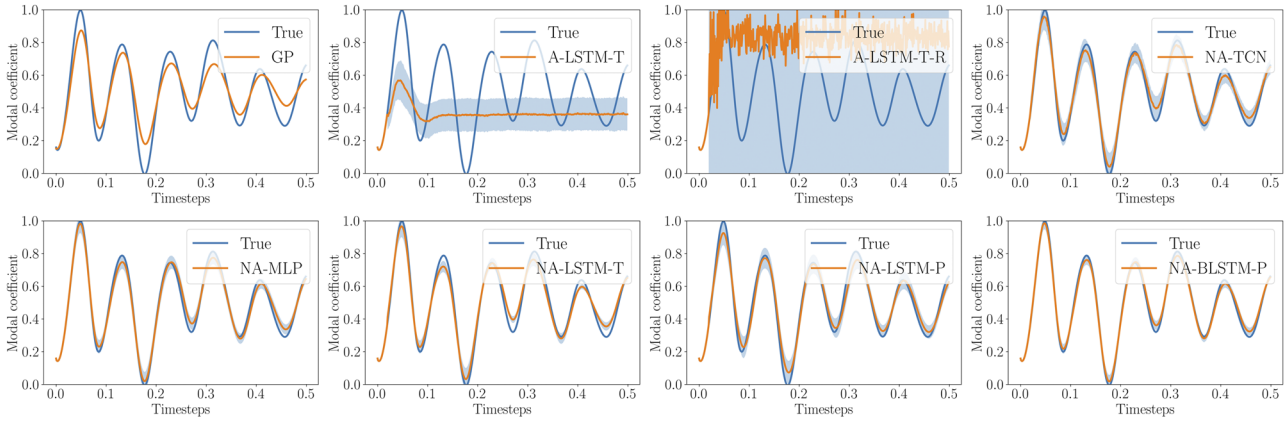

FIG. 14. Predictive ability for the assessed frameworks for PCA component 3. Note that GP requires the solution of a partial differential equation in addition to complete observations from the true system to build a model. In addition, GP is deterministic.

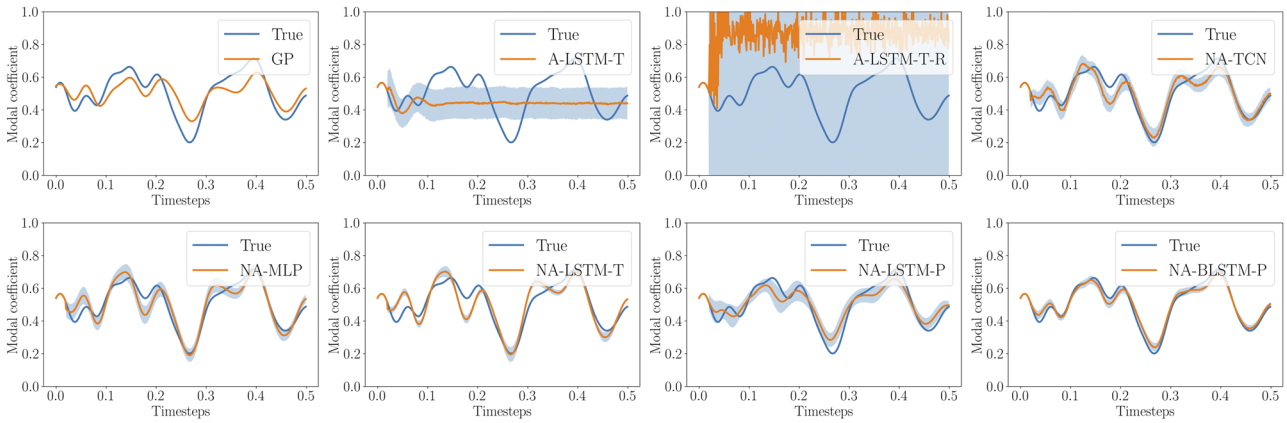

FIG. 15. Predictive ability for the assessed frameworks for PCA component 4. Note that GP requires the solution of a partial differential equation in addition to complete observations from the true system to build a model. In addition, GP is deterministic.

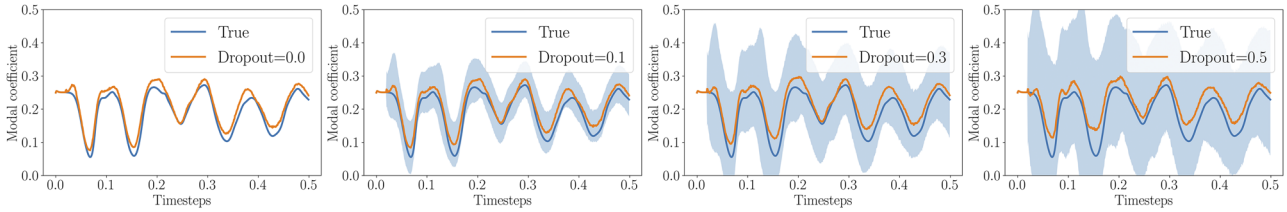

FIG. 16. Ablation study focusing on the effect of dropout probability on the NA-BLSTM-P method. Testing mean-squared errors/error variance for the four models were (from left to right) $8.40 \times 10^{-5}/5.23 \times 10^{-6}$, $8.53 \times 10^{-4}/5.08 \times 10^{-6}$, $1.01 \times 10^{-3}/5.44 \times 10^{-6}$, and $1.50 \times 10^{-3}/8.59 \times 10^{-6}$.

comparable number of trainable parameters compared to NA-TCN and far fewer than the NA-LSTM-T and NA-MLP methods. We note here that the latter two algorithms do not interpret the data to be sequential in PCA space.

Non-autoregressive methods, in general, provide several orders of magnitude acceleration over GP and the autoregressive methods, thereby proving suitable for their ultimate application in cost reduction. For instance, the average time-to-solution for a non-autoregressive method was around 0.01 s, whereas the autoregressive methods required ~3 s for forecasting. The equation-based GP model required around 20 s for a complete simulation. The readers may note that the large cost of the GP model stems from the need to retain 40 principal components, which causes greater computational demands due to the complexity of the nonlinear term computation.

For another validation, we take the best hyperparameters obtained for NA-BLSTM-P and perform an ablation study for the magnitude of the dropout probability, as shown in Fig. 16.

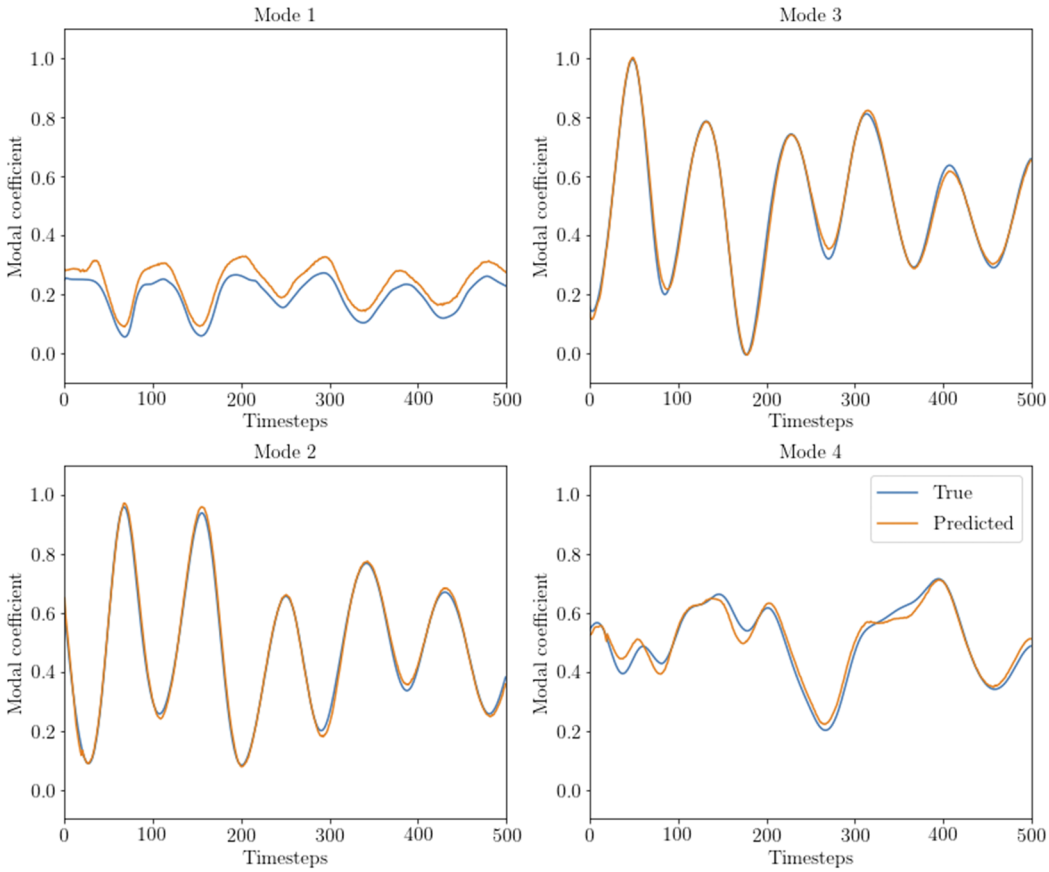

FIG. 17. Predictive ability for the NA-BLSTM-P method equipped with a MLP-based meta-model to predict the burn-in sequence given w.

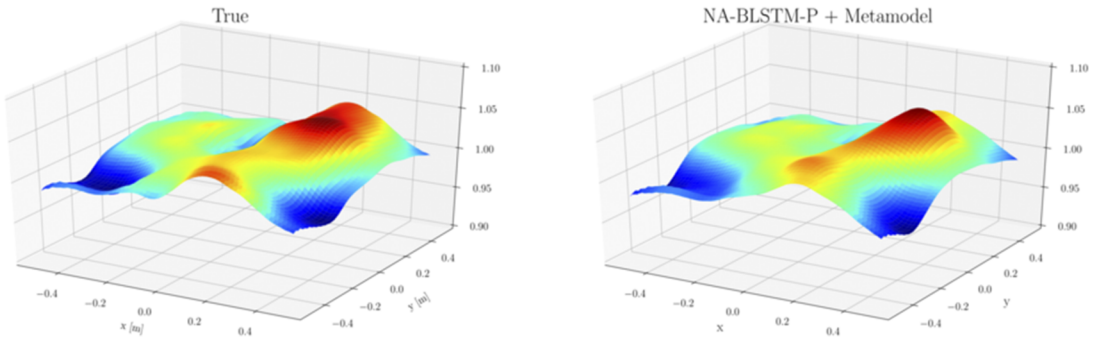

FIG. 18. Predictive ability for the NA-BLSTM-P method equipped with a MLP-based meta-model. The meta-model predicts the burn-in sequence given w, which may then be used as an input to the non-autoregressive surrogate.

The network was deployed four times with dropout probability values of 0.0, 0.1, 0.3, and 0.5; 1000 inferences were made on the test dataset for each value of the dropout probability. As the dropout probabilities of neurons increase during inference, the network is seen to display increased variability. However, mean predictions for the ensemble show good agreement with the underlying truth. We note that this model was trained without dropping out neurons and the role of dropout during inference is to simulate error accumulation. The viable performance for different dropout magnitudes indicates that the model is robust to error forcing at different magnitudes for the entire length of the forecast, thereby displaying robustness for long-term forecasts.

F. Parametric meta-model for latent space initialization

In this section, we assess further possibilities for accelerating our reduced-order model by removing the dependence on the previously introduced burn-in duration. We remind the reader that our experiments, thus far, assume the availability of a slice of the trajectory for a test initial condition (from a numerical solver) to initialize the deployment of the data-driven forecast methods. This represents a potential bottleneck for physical systems where large lead-times may be necessary for the effect of control parameters to cause learnable differences in trajectories. To that end, we propose the use of a meta-modeling strategy that predicts the initial burn-in trajectory from w alone. Essentially, our meta-model learns to predict the trajectory for the first $\mathbf{k}$ time steps by observing w; following this, our time-series method predicts the trajectory for the $\mathbf{N}$–$\mathbf{k}$ time steps.

For simplicity, we select a 3-layer perceptron with 20, 40, and 20 neurons in hidden layers 1, 2, and 3, respectively. Each hidden layer is also equipped with a rectified linear activation function. This architecture is used to map from w to $\mathbf{z_1}$, $\mathbf{z_2}$, …, $\mathbf{z_k}$. Our dataset remains unchanged, with 20 training points being used to train this map. Note that the paucity of training data also motivates the use of such a simple framework. We train our framework with an L_1-regularization strategy where the objective function of our training is penalized by the sum of absolute values of trainable parameters.

PCA coefficient predictions from a deployment of the NA-BLSTM-P model trained in Sec. IV B are shown in Fig. 17. In contrast to the results obtained previously, the initial burn-in window is not available from a numerical simulation but is predicted by our meta-model. One can observe that the combination of the two data-driven maps allows for the direct prediction of a trajectory, given solely initial conditions from w. Contours for the associated trajectory are shown in Fig. 18 where the final time flow-field is reconstructed accurately.

V. CONCLUSIONS AND FUTURE WORK

We introduce a novel methodology for the accurate prediction of nonlinear dynamical systems solely from data. The dataset is interpreted to be sequential in PCA space due to the ordered nature of the PCA bases. Consequently, this allows for gating in the PCA coefficient dimension. Through this interpretation of the nonlinear dynamics, the introduced methods address challenges related to the stability of traditional time-series learning methods such as the LSTM for advection-dominated problems.

Our results show that the incorporation of a bidirectional gating in the PCA coefficient dimension leads to the lowest testing and reconstruction errors for the nonlinear dynamics of the shallow water equations. In addition, the choice of the bidirectional gating is *physics-informed* since the PCA coefficients are ordered according to decreasing spectral content capture. In physics, this variance corresponds to the amount of spectral content captured by the different bases. The bidirectional LSTM allows for the output to be influenced by relevant spectral content from multiple PCA bases and may be the cause of the best performance among the methods studied here. We compare these results with other non-autoregressive methods such as the non-autoregressive temporal convolutional network and the non-autoregressive multilayered perceptron where superior performance is observed. The novel interpretation of nonlinear dynamics also allows for rapid inference with orders of magnitude reduction in inference times in comparison with the benchmark GP and autoregressive methods. We also note that the model size of the proposed non-autoregressive bidirectional LSTM is comparatively lower than that of the other non-autoregressive methods. These features are useful for lightweight deployments of the proposed methods for applications in control, data assimilation, and parametric forecasting.

A few limitations of the non-autoregressive methods need to be explored. While these methods aid in bypassing issues of stability and slow inference times, their efficacy for forecasting (for unseen t) remains to be seen. Our current problem is purely transient in nature, and precise prediction beyond 500 time steps is not desired to avoid extrapolation. However, these methods must be assessed for datasets where self-similar temporal information can be leveraged to make forecasts. Another limitation may arise from the nature of training these non-autoregressive methods. Since each nonlinear dynamical system solve is considered a sample, much larger datasets may need to be generated for effective learning. Training times (given in Table III) show that these methods may be costly to train on large datasets. Our future work is aimed at addressing these issues. We should also caution the reader that when a considerable amount of underlying knowledge about a system is available, such as through a thorough understanding of the governing equations and the availability of exceptional observational data, projection-based ROMs may still be very competitive. Indeed, machine-learned ROMs may also suffer from the presence of data with observational biases (such as non-Gaussian noise) for which equation-based techniques may prove superior. We advocate for computational workflows that utilize the strengths of several ROM methodologies for mitigating their individual weaknesses.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract No. DE-AC02-06CH11357. This research was funded by, in part, and used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract No. DE-AC02-06CH11357. R.M. acknowledges support from the Margaret Butler Fellowship at the Argonne Leadership Computing Facility. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in this paper do not necessarily represent the views of the U.S. DOE or the United States Government.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

DATA AVAILABILITY

All the source codes for this study, including training and testing data, is available at https://github.com/rmjcs2020/NATSurrogates.

REFERENCES

[1] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-net: Learning PDEs from data," in *Proceedings of the 35th International Conference on Machine Learning* (PMLR, 2018), Vol. 80, pp. 3208–3216.
[2] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," J. Mach. Learn. Res. **19**(1), 932–955 (2018).
[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[4] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," J. Comput. Phys. **375**, 1339–1364 (2018).
[5] M. Yang and Z. Xiao, "Improving the k–ω–γ–Ar transition model by the field inversion and machine learning framework," Phys. Fluids **32**(6), 064101 (2020).
[6] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, "Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning," Phys. Fluids **32**(5), 053605 (2020).
[7] H. Vaddireddy, A. Rasheed, A. E. Staples, and O. San, "Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data," Phys. Fluids **32**(1), 015113 (2020).
[8] O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowlishwaran, "CFDNet: A deep learning-based accelerator for fluid simulations," arXiv:2005.04485 (2020).
[9] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).
[10] K. Fukami, K. Fukagata, and K. Taira, "Assessment of supervised machine learning methods for fluid flows," Theor. Comput. Fluid Dyn. **34**, 1–23 (2020).
[11] B. N. Hanna, N. T. Dinh, R. W. Youngblood, and I. A. Bolotnov, "Machine-learning based error prediction approach for coarse-grid computational fluid dynamics (CG-CFD)," Prog. Nucl. Energy **118**, 103140 (2020).
[12] R. W. C. P. Verstappen and A. E. P. Veldman, "Direct numerical simulation of turbulence at lower costs," J. Eng. Math. **32**(2-3), 143–159 (1997).
[13] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," arXiv:1911.08655 (2019).
[14] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks **18**(5-6), 602–610 (2005).
[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**(8), 1735–1780 (1997).
[16] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," arXiv:1609.03499 (2016).
[17] C. W. Rowley, T. Colonius, and R. M. Murray, "Model reduction for compressible flows using POD and Galerkin projection," Physica D **189**(1-2), 115–129 (2004).
[18] W. W. Hsieh and B. Tang, "Applying neural network models to prediction and data analysis in meteorology and oceanography," Bull. Am. Meteorol. Soc. **79**(9), 1855–1870 (1998).
[19] A. Mannarino and P. Mantegazza, "Nonlinear aeroelastic reduced order modeling by recurrent neural networks," J. Fluids Struct. **48**, 103–121 (2014).
[20] W. Zhang, J. Kou, and Z. Wang, "Nonlinear aerodynamic reduced-order model for limit-cycle oscillation and flutter," AIAA J. **54**, 3304–3311 (2016).
[21] J. Kou and W. Zhang, "Layered reduced-order models for nonlinear aerodynamics and aeroelasticity," J. Fluids Struct. **68**, 174–193 (2017).
[22] O. San and R. Maulik, "Neural network closures for nonlinear model order reduction," Adv. Comput. Math. **44**(6), 1717–1750 (2018).
[23] J. S. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks," J. Comput. Phys. **363**, 55–78 (2018).
[24] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows," AIAA J. **58**, 25–36 (2019).
[25] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," in *Computer Graphics Forum* (Wiley Online Library, 2019), Vol. 38, pp. 59–70.

[26]R. Han, Y. Wang, Y. Zhang, and G. Chen, "A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network," Phys. Fluids **31**(12), 127101 (2019).
[27]T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," J. Fluid Mech. **882**, A13 (2020).
[28]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).
[29]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proc. Natl. Acad. Sci. U. S. A. **113**(15), 3932–3937 (2016).
[30]K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," Proc. Natl. Acad. Sci. U. S. A. **116**(45), 22445–22451 (2019).
[31]K. K. Chen, J. H. Tu, and C. W. Rowley, "Variants of dynamic mode decomposition: Boundary condition, Koopman, and Fourier analyses," J. Nonlinear Sci. **22**(6), 887–915 (2012).
[32]J. N. Kutz, X. Fu, and S. L. Brunton, "Multiresolution dynamic mode decomposition," SIAM J. Appl. Dyn. Syst. **15**(2), 713–735 (2016).
[33]Q. Wang, J. S. Hesthaven, and D. Ray, "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem," J. Comput. Phys. **384**, 289–307 (2019).
[34]S. E. Ahmed, S. M. Rahman, O. San, A. Rasheed, and I. M. Navon, "Memory embedded non-intrusive reduced order modeling of non-ergodic flows," Phys. Fluids **31**(12), 126602 (2019).
[35]M. Guo and J. S. Hesthaven, "Reduced order modeling for nonlinear structural analysis using Gaussian process regression," Comput. Methods Appl. Mech. Eng. **341**, 807–826 (2018).
[36]M. Kast, M. Guo, and J. S. Hesthaven, "A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems," Comput. Methods Appl. Mech. Eng. **364**, 112947 (2020).
[37]Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo, "Model identification of reduced order fluid dynamics systems using deep learning," Int. J. Numer. Methods Fluids **86**(4), 255–268 (2018).
[38]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).
[39]A. T. Mohan and D. V. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks," arXiv:1804.09269 (2018).
[40]S. M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu, "Nonintrusive reduced order modeling framework for quasigeostrophic turbulence," Phys. Rev. E **100**(5), 053306 (2019).
[41]Z. Deng, Y. Chen, Y. Liu, and K. C. Kim, "Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework," Phys. Fluids **31**(7), 075108 (2019).
[42]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**(3), 036602 (2020).
[43]R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu, "Time-series learning of latent-space dynamics for reduced-order model closure," Physica D **405**, 132368 (2020).
[44]R. Maulik, R. Egele, B. Lusch, and P. Balaprakash, "Recurrent neural network architecture search for geophysical emulation," arXiv:2004.10928 (2020).
[45]B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," Neural Comput. **1**(2), 263–269 (1989).
[46]S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems* (NeurIPS, 2019), pp. 15353–15363.
[47]P. Jin, A. Zhu, G. E. Karniadakis, and Y. Tang, "Symplectic networks: Intrinsic structure-preserving networks for identifying Hamiltonian systems," arXiv:2001.03750 (2020).
[48]J. W. Burby, Q. Tang, and R. Maulik, "Fast neural poincaré maps for toroidal magnetic fields," arXiv:2007.04496 (2020).
[49]J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," J. Guid., Control, Dyn. **8**(5), 620–627 (1985).
[50]K. C. Hall, J. P. Thomas, and E. H. Dowell, "Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows," AIAA J. **38**(10), 1853–1862 (2000).
[51]R. Ştefănescu and I. M. Navon, "POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model," J. Comput. Phys. **237**, 95–114 (2013).
[52]S. Hijazi, G. Stabile, A. Mola, and G. Rozza, "Data-driven POD-Galerkin reduced order model for turbulent flows," J. Comp. Phys. **416**, 109513 (2020).
[53]S. Chaturantabut and D. C. Sorensen, "Nonlinear model reduction via discrete empirical interpolation," SIAM J. Sci. Comput. **32**(5), 2737–2764 (2010).
[54]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).
[55]J. R. Dormand and P. J. Prince, "A family of embedded Runge–Kutta formulae," J. Comput. Appl. Math. **6**(1), 19–26 (1980).
[56]R. Engelken, F. Wolf, and L. F. Abbott, "Lyapunov spectra of chaotic recurrent neural networks," arXiv:2006.02427 (2020).
[57]B. Hamzi and H. Owhadi, "Learning dynamical systems from data: A simple cross-validation perspective," arXiv:2007.05074 (2020).
[58]B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," Nat. Commun. **9**(1), 4950 (2018).
[59]Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu, "Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison," Comput. Methods Appl. Mech. Eng. **237-240**, 10–26 (2012).
[60]J. Xu and K. Duraisamy, "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics," arXiv:1912.11114 (2019).
[61]G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," Annu. Rev. Fluid Mech. **25**(1), 539–575 (1993).
[62]A. Mendible, S. L. Brunton, A. Y. Aravkin, W. Lowrie, and J. N. Kutz, "Dimensionality reduction and reduced order modeling for traveling wave physics," Theor. Comp. Fluid Dyn. **34**, 1–16 (2020).
[63]A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding hard physical constraints in convolutional neural networks for 3D turbulence," in ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.
[64]B. Liu, J. Tang, H. Huang, and X.-Y. Lu, "Deep learning methods for super-resolution reconstruction of turbulent flows," Phys. Fluids **32**(2), 025105 (2020).
[65]P. Balaprakash, M. Salim, T. Uram, V. Vishwanath, and S. Wild, "DeepHyper: Asynchronous hyperparameter search for deep neural networks," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)* (IEEE, 2018), pp. 42–51.
[66]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res. **12**(85), 2825–2830 (2011).
[67]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res. **15**(1), 1929–1958 (2014).
[68]Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning* (ACM, 2016), pp. 1050–1059.