A deep learning enabler for nonintrusive reduced order modeling of fluid flows

Cite as: Phys. Fluids **31**, 085101 (2019); https://doi.org/10.1063/1.5113494
Submitted: 04 June 2019 . Accepted: 11 July 2019 . Published Online: 01 August 2019

S. Pawar, S. M. Rahman, H. Vaddireddy, O. San, A. Rasheed, and P. Vedula

Physics of Fluids ARTICLE scitation.org/journal/phf

A deep learning enabler for nonintrusive reduced order modeling of fluid flows

Cite as: Phys. Fluids 31, 085101 (2019); doi: 10.1063/1.5113494
Submitted: 4 June 2019 • Accepted: 11 July 2019 •
Published Online: 1 August 2019

S. Pawar,[1] S. M. Rahman,[1] H. Vaddireddy,[1] O. San,[1,a)] A. Rasheed,[2] and P. Vedula[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]School of Aerospace and Mechanical Engineering, The University of Oklahoma, Norman, Oklahoma 73019, USA

[a)]**Electronic mail:** osan@okstate.edu

ABSTRACT

In this paper, we introduce a modular deep neural network (DNN) framework for data-driven reduced order modeling of dynamical systems relevant to fluid flows. We propose various DNN architectures which numerically predict evolution of dynamical systems by learning from either using discrete state or slope information of the system. Our approach has been demonstrated using both residual formula and backward difference scheme formulas. However, it can be easily generalized into many different numerical schemes as well. We give a demonstration of our framework for three examples: (i) Kraichnan-Orszag system, an illustrative coupled nonlinear ordinary differential equation, (ii) Lorenz system exhibiting chaotic behavior, and (iii) a nonintrusive model order reduction framework for the two-dimensional Boussinesq equations with a differentially heated cavity flow setup at various Rayleigh numbers. Using only snapshots of state variables at discrete time instances, our data-driven approach can be considered truly nonintrusive since any prior information about the underlying governing equations is not required for generating the reduced order model. Our *a posteriori* analysis shows that the proposed data-driven approach is remarkably accurate and can be used as a robust predictive tool for nonintrusive model order reduction of complex fluid flows.

I. INTRODUCTION

Many realistic transient flows typically involve very wide ranges of spatial and temporal scales, which place an enormous computational burden on direct numerical simulations (DNS) of such flows based on governing equations. Advancement in high-performance computing systems along with the development of consistent, stable, convergent numerical schemes, and efficient parallel algorithms has enabled us to analyze and study complex real world processes. For instance, it is now possible to collect very high-resolution DNS data relevant to selected turbulent flows which cannot be gathered experimentally.[1] However, the computational cost of performing DNS scales roughly as Re^3, where Re is the Reynolds number of the flow.[2] Hence, with the present state of the art computing architectures,[3] such high-resolution simulations of multiphysics flows might require weeks of computations even for simple geometries. The situation worsens when a series of numerical simulations need to be run for any parametric design optimization study. To alleviate this, coarse-graining approaches, as performed, for example, in large eddy simulations (LES), are commonly used to reduce this computational burden.[4–7]

However, the computational cost of full-order simulations (i.e., DNS or even LES) can still be considered extremely prohibitive due to a large number of degrees of freedom needed to resolve all of the flow features, especially in settings where the traditional methods require repeated model evaluations over a large range of parameters. Therefore, many successful model order reduction approaches have been introduced.[8–12] The main purpose of such approaches is to reduce this computational burden and serve as surrogate models for efficient computational analysis of fluid systems. A common objective in such reduced order modeling (ROM) approaches is to determine how well these approaches can reproduce the flow dynamics.

Intrusive finite dimensional low order models routinely arise when we apply Galerkin type projection techniques to infinite dimensional models.[13–15] On the other hand, without prior information on the governing equations, their operator forms, or parameterizations to account for complex physical processes,

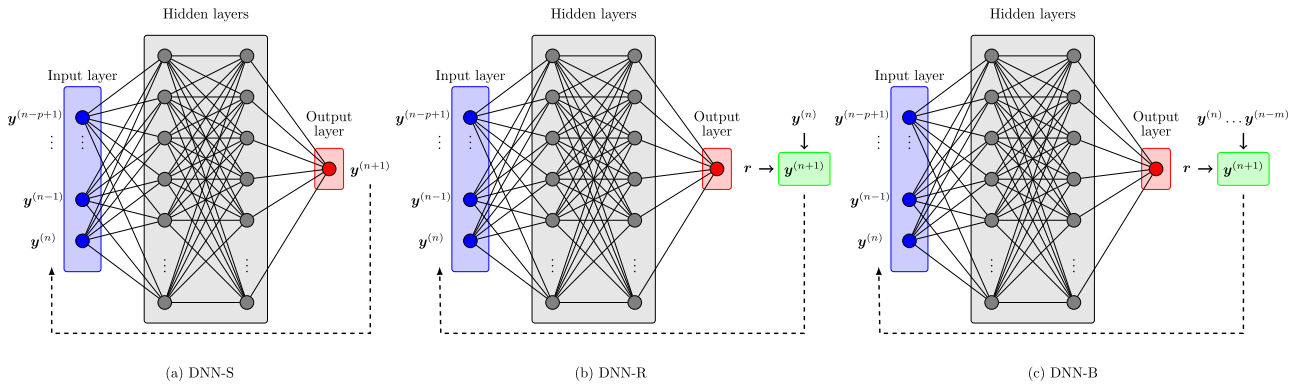

FIG. 1. Iterative prediction using the trained DNN model for different frameworks proposed in this study. The input to the neural network consists of state history for p time steps. For the DNN-R framework, $\boldsymbol{r}$ is the residual predicted by the neural network. For the DNN-B framework, $\boldsymbol{r}$ is the discrete numerical slope computed using any of the numerical method used for training the neural network. The number of inputs to compute numerical slope in DNN-B framework depends upon the numerical scheme applied for calculating the slope (for example, $m = 1$ for the second-order backward difference scheme used in this study).

a nonintrusive ROM approach can be reconstructed to infer such underlying physics from the data itself. Having ability to facilitate dynamic data exchange easier between different components, these nonintrusive models can arguably be more promising and impactful in numerous interdisciplinary fields. Moreover, with the advent of digital twin technologies,[16] the collection of data from sensors has become possible at different stages of product's lifecycle, and model order reduction might be considered a key enabler for this digital twin vision in many emerging cyber-physical systems.[17]

Reduced order models offer promises in many fields such as system identification,[18–20] control,[21–26] optimization,[27–29] and data assimilation[30–32] applications. In these model reduction approaches, we aim at obtaining simplified (but dense) models from high-fidelity numerical simulation data or data collected from the experiment.[12,33] To fulfill their objectives in multiple forward simulations of the problem with different model parameters,[34–39] these models should be sufficiently accurate and computationally much faster than the high-fidelity numerical simulation. Therefore, there has been progress made in recent years to develop such ROM approaches specifically for nonlinear systems.[40–45]

The basic philosophy of projection-based ROM approaches is to reduce the high degrees of freedom of a governing equation through an expansion in a transformed space, traditionally with orthogonal basis. Among the large variety of projection-based ROM strategies, the proper orthogonal decomposition (POD) has emerged as a popular technique for the study of dynamical systems,[9,12,46,47] which targets the most dominant characteristics of the flow considering the largest energy containing modes. The POD technique was first introduced in fluid community in the context of extracting coherent structure from a turbulent flow field.[48] Several methods have also been proposed in the literature aimed at improving the POD modes.[49–52] There are also different variants of POD that have been introduced, such as spatio-temporal biorthogonal decomposition,[53] spectral POD (SPOD),[54] frequency based POD that is also called SPOD,[55] and multiscale POD (MPOD)[56] which splits the correlation matrix into the contribution of different scales.

The evolution equations for the lower order system are then obtained using the Galerkin projection method. For many flows, the POD-Galerkin method provides an efficient and accurate way to generate ROM methodologies.[57–65] Furthermore, several successful closure models have been suggested in order to model the effects of discarded modes.[66–69] The POD-Galerkin intrusive approach can also be stabilized with a nonlinear eddy viscosity model[70] or with proper selection of linear quadratic coefficients.[71] However, the projection-based model reduction approaches have limitations especially for complex systems such as general circulation models since there is a lack of access to the full-order model operators or the complexity of the forward simulation codes that render the need for obtaining the full-order operators.[72–74]

TABLE I. The output of different DNN frameworks learned through training. The trained parameter is then used to update the solution in time, starting from the initial condition.

Neural network framework	Predicted variable	Solution update
DNN-S	$\boldsymbol{r} = \boldsymbol{y}^{(n+1)}$	$\boldsymbol{y}^{(n+1)} = \boldsymbol{r}$
DNN-R	$\boldsymbol{r} = \boldsymbol{y}^{(n+1)} - \boldsymbol{y}^{(n)}$	$\boldsymbol{y}^{(n+1)} = \boldsymbol{y}^{(n)} + \boldsymbol{r}$
DNN-B	$\boldsymbol{r} = \frac{3\boldsymbol{y}^{(n+1)} - 4\boldsymbol{y}^{(n)} + \boldsymbol{y}^{(n-1)}}{2\Delta t}$	$\boldsymbol{y}^{(n+1)} = \frac{4}{3}\boldsymbol{y}^{(n)} - \frac{1}{3}\boldsymbol{y}^{(n-1)} + \frac{2}{3}\boldsymbol{r}\Delta t$

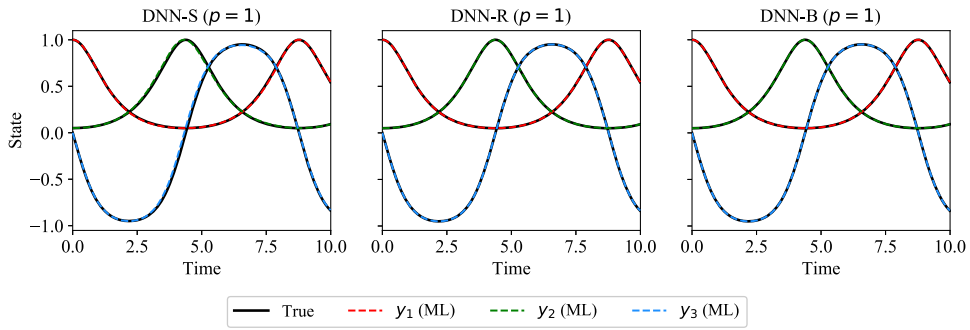

FIG. 2. Prediction of the state of the Kraichnan-Orszag system for different DNN frameworks trained using $p = 1$. The initial condition of the dynamical system is $[y_1\ y_2\ y_3]^T = [1.0\ 0.05\ 0.0]^T$. The solid lines present the true state of the system and the dashed line presents the state of the system predicted using neural network.

One of the challenges in Galerkin projection is the deformation of POD modes. As recently discussed by Reiss *et al.*,[75] transport-dominated phenomena are usually a challenge for modal methods since their dynamics cannot be captured accurately by a few dominant spatial modes. If we include more number of modes to better recover the embedded structures in the underlying system, the computational expense increases and the ROM might not be efficient from practical point of view. Furthermore, the construction of a least-order state space is crucial especially in control since every degree of freedom can amplify noise.[76] A data-driven manifold learning model has been proposed as a general dynamic ROM modeling framework.[45] Ehlert *et al.*[76] have also presented a manifold representation of the transient oscillatory cylinder wake using a locally linear embedding approach as encoder. They found that this representation outperforms a 50-dimensional POD expansion from the same data. Another key dynamic problem is that hyperbolic convection problems are treated with an elliptic Galerkin method.[77] Even though the flow has a certain specific direction, the Galerkin projection assumes that the modes are globally coupled. This mismatch between Navier-Stokes equations and Galerkin dynamics might not be curable. Also, the frequency range of high-dimensional Navier-Stokes solutions is not resolved in the low-dimensional deterministic system.[57,78] Rempfer[79] has demonstrated some implications of projecting the Navier-Stokes equations onto low-dimensional bases and showed how the restriction to a low-dimensional basis as well as improper treatment of boundary conditions might affect the validity of ROM. Due to all these limitations, there is a recent interest in generating fully nonintrusive approaches without the need for access to full-order model operators to establish surrogate models.[73,80–87] Dynamic mode decomposition (DMD) models[88–93] provide this nonintrusive representation directly from data by their nature, and several approaches have been readily available for optimal mode selection.[94–97]

There is a broad range of opportunities for the application of machine learning algorithms to develop nonintrusive reduced order models. A good discussion on the application of data-driven methods for dynamical systems can be found in a book by Brunton and Kutz.[98] We also refer to a recent review article[99] for a comprehensive overview of the machine learning literature in fluid mechanics. A number of studies have been done to apply data-driven techniques to predict the high-dimensional complex dynamical systems.[100–108] San and Maulik[109] proposed a methodology to account for the effects of truncated POD modes using a single layer feed-forward neural network. A multistep neural network was proposed to identify the nonlinear dynamical system from the data by combining classical numerical analysis techniques with the powerful nonlinear approximation capability of neural networks.[102] Xie, Zhang, and Webster[101] used the multistep neural network to approximate the full order model projected on low-dimensional space with a supervised learning task. A deep residual recurrent neural network was introduced as an efficient model reduction technique for nonlinear dynamical systems.[105] Vlachas *et al.*[110] developed the data-driven forecasting method for high-dimensional, chaotic systems using the hybrid approach which combines the mean stochastic model and the recurrent long short-term memory (LSTM) neural network. The LSTM recurrent neural network was used to model the temporal dynamics of turbulence in a ROM framework.[111] Pathak *et al.*[112] proposed a hybrid forecasting model combining the knowledge of the governing equation of the dynamical system and the machine learning technique to predict the long term behavior of chaotic systems.

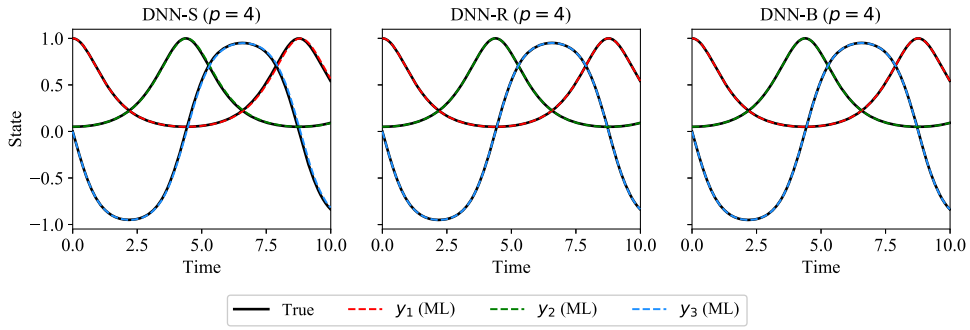

FIG. 3. Prediction of the state of the Kraichnan-Orszag system for different DNN frameworks trained using $p = 4$. The initial condition of the dynamical system is $[y_1\ y_2\ y_3]^T = [1.0\ 0.05\ 0.0]^T$. The solid lines present the true state of the system, and the dashed line presents the state of the system predicted using the neural network.

TABLE II. Quantitative assessment of different DNN frameworks with different numbers of inputs to the neural network for the Kraichnan-Orszag system using the total root mean square error given by Eq. (6).

Framework	RMSE ($p = 1$)	RMSE ($p = 4$)
DNN-S	1.71×10^{-2}	1.90×10^{-2}
DNN-R	1.02×10^{-3}	2.48×10^{-4}
DNN-B	5.67×10^{-4}	5.97×10^{-4}

This hybrid approach was found to be better than either its pure data-driven component or its model-based component.

In our proposed ROM framework, we will bypass the Galerkin projection step of the projection based ROM with our proposed neural network architectures to build a fully nonintrusive approach. This nonintrusive ROM (NIROM) framework can be viewed as a decomposition of the problem into basis representation and forecasting subproblems. We illustrate our NIROM approach using the deep feed-forward neural network architectures. However, it can be easily applied to other types of neural networks (as demonstrated for data-driven forecasting of dynamical systems[110,113]) or more traditional time series forecasting tools.[114] The neural networks are capable of approximating the nonlinear functions and have been successfully used in turbulence modeling,[115–117] solving the differential equation.[118,119] We learn the dynamics of the reduced order model directly from the output of the full order model projected on the low-dimension space using a supervised learning task. The main advantage of this nonintrusive approach is that it does not require information about the equations governing the full order model. Although the proposed approach helps generate a NIROM framework solely from the snapshot data reconstructed onto a POD-spanned space, it may still suffer from fundamental challenges of traditional POD-Galerkin models (e.g., we refer to the work of Zerfas *et al.*[120] for a recent discussion about ways to mitigate their lack of accuracy).

The paper is organized as follows: Sec. II introduces deep neural network (DNN) architecture and implementation of different

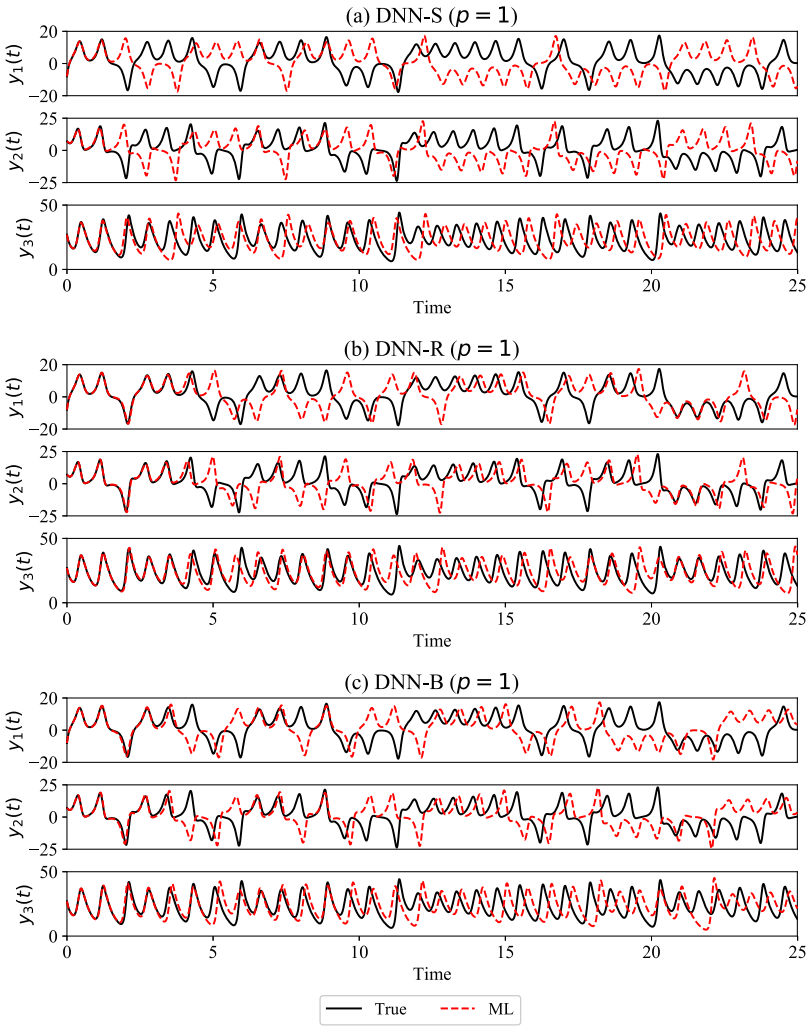

FIG. 4. Time evolution of the Lorenz system trajectories for the initial condition $[y_1\ y_2\ y_3]^T = [-8\ 7\ 27]^T$. The neural network is trained using the data generated from the true solution between $t = 0$ and 25 with $p = 1$.

DNN frameworks for dynamical systems. Section III gives numerical results using our DNN frameworks for two dynamical systems: Kraichnan-Orszag system and Lorenz system. We present the generalized nonintrusive ROM framework in Sec. IV A. In Sec. IV B, we present the Boussinesq equation problem and its POD analysis. The nonintrusive ROM framework for the Boussinesq equation is described in Sec. IV C. The numerical results in Sec. V demonstrate the effectiveness of our nonintrusive approach for reduced order modeling of a differentially heated cavity problem at two Rayleigh numbers. We give some concluding remarks and suggestions for future work in Sec. VI.

II. LEARNING FRAMEWORK

The deep neural network is an artificial neural network composed of several layers made up of the predefined number of nodes. These nodes are also called neurons. A node combines the input from the data with a set of coefficients called weights. These weights either amplify or dampen the input and thereby assign the significance to the input with respect to the output that the DNN is trying to learn. In addition to the weights, these nodes have a bias for each input to the node. The input-weight product and the bias are summed and the sum is passed through a node's activation function. The above process can be described using the matrix operation as given by[121]

$$S^l = \mathbf{W}^l X^{l-1}, \tag{1}$$

where X^{l-1} is the output of the $(l-1)$th layer and $\mathbf{W}^l$ is the matrix of weights for the lth layer. The output of the lth layer is given by

$$X^l = \zeta(S^l + B^l), \tag{2}$$

where B^l is the vector of biasing parameters for the lth layer and ζ is the activation function. If there are L layers between the input and the output, then the mapping of the input to the output can be derived as follows:

$$Y = \zeta_L(\mathbf{W}^L, B^L, \ldots, \zeta_2(\mathbf{W}^2, B^2, \zeta_1(\mathbf{W}^1, B^1, X))), \tag{3}$$

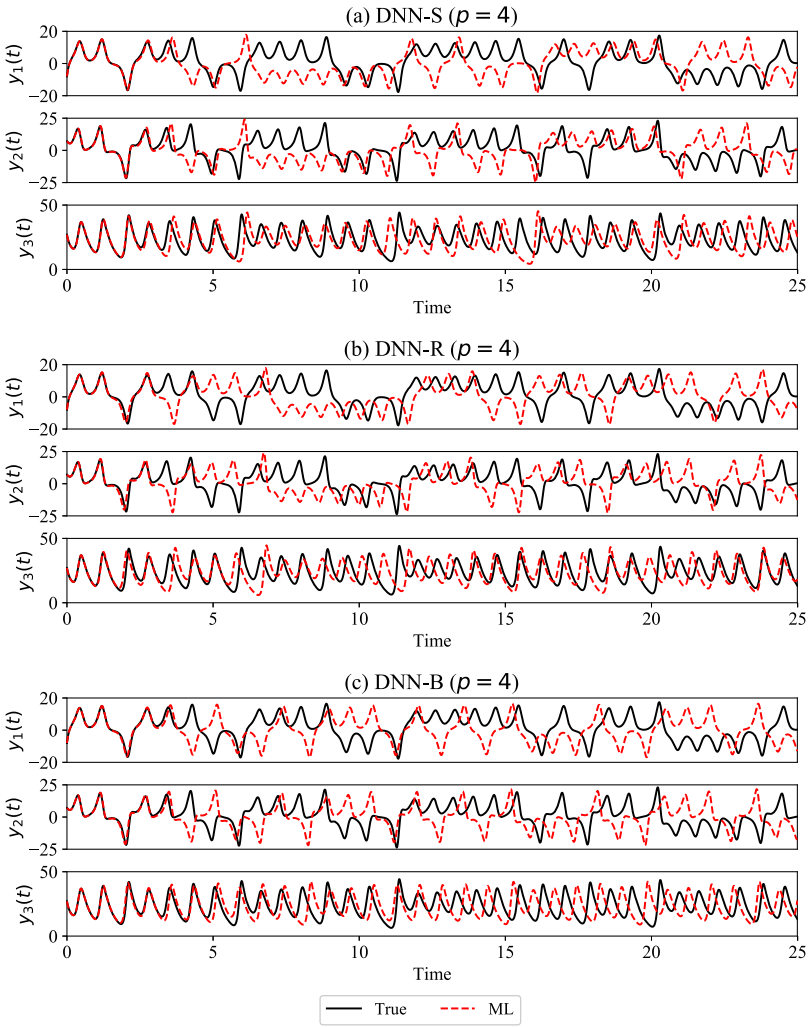

FIG. 5. Time evolution of the Lorenz system trajectories for the initial condition $[y_1\ y_2\ y_3]^T = [-8\ 7\ 27]^T$. The neural network is trained using the data generated from the true solution between $t = 0$ and 25 with $p = 4$.

where X and Y are the input and output of the deep neural network, respectively.

The input layer usually takes the raw data from the training dataset and transfers them to the second layer. It does not have any biasing or activation through an activation function. The output layer usually has the linear activation function and some bias associated with the inputs. The linear activation function simply takes the summation of inputs received from the previous hidden layer and the associated bias of the output layer. In this study, we use the ReLU activation function for all hidden layers and the linear activation function for the output layer in all DNN frameworks. The ReLU activation function can be expressed as

$$\zeta(\chi) = \max(0, \chi), \tag{4}$$

where ζ is the activation function and χ is the input to the node.

Each entry of the matrices $\mathbf{W}$ and B is learned through backpropagation and some optimization algorithm. The backpropagation algorithm provides a way to compute the gradient of the objective function efficiently, and the optimization algorithm gives a rapid way to learn optimal weights. The objective of the neural networks in this study is to learn the weights associated with each node in such a way that the root mean square error between the true labels Y_0 and the output of the neural network Y is minimized. The backpropagation proceeds as follows: (i) the input and output of the neural network are specified along with initial weights, (ii) the training data are run through the network to produce output Y whose true value is Y_0, (iii) the derivative of the objective function with each of the training weight is computed using the chain rule, and (iv) the weights are updated based on the learning rate and then we go to step (ii). We continue to iterate through this procedure until convergence, or the maximum number of iterations is reached. The Adams optimization algorithm[122] is used in this study for learning optimal weights to minimize the objective function.

Deep neural networks are capable of approximating nonlinear dynamical systems as shown in many studies.[102,103,106,123] The general nonlinear dynamical system can be presented by an equation of the form

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{F}(\boldsymbol{y}, t), \tag{5}$$

where $\boldsymbol{y}(t)$ is the state variable at time t and $\boldsymbol{F}(\boldsymbol{y}, t)$ is the nonlinear function evaluated for each component of state variable $\boldsymbol{y}(t)$.

Figure 1 shows three different DNN frameworks used in this study to predict the dynamical system. Although their characteristics on the stability and well-posedness are beyond the scope of the present work, we refer to the study of Chang *et al.*[124] for several deep neural networks and their stability issues.

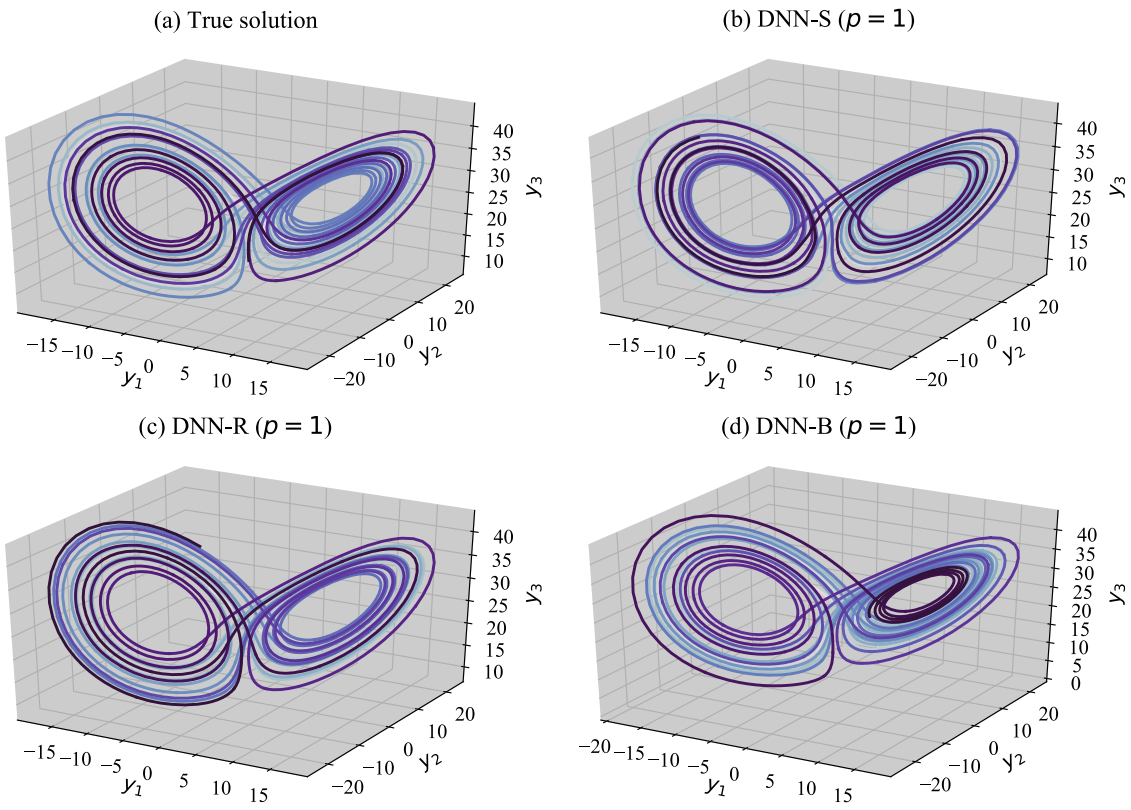

FIG. 6. The true phase portrait of the Lorenz system (a) compared with the phase portrait predicted by different DNN frameworks [(b)-(d)] for time integration from $t = 0$ to $t = 25$. Note that there is a good qualitative argument between the phase portrait predicted by the DNN-S framework (b), DNN-R framework (c), and DNN-B framework (d) with the true phase portrait (a). All DNN frameworks are trained with $p = 1$.

Our motivation for different choices of DNNs comes from recent development of approximation and discovery of dynamical systems using deep learning techniques.[102,107] At this point, it may be inconclusive to say which DNN framework is superior as compared to others. However, our empirical evidence suggests that learning residual information or numerical slope helps in more accurate prediction of dynamical systems. In the case of the DNN-S framework, we train the neural network to learn an update formula which advances the state of the system from $\boldsymbol{y}^{(n)}$ to $\boldsymbol{y}^{(n+1)}$ directly, where (n) denotes the state of the system at time t_n. The S in the DNN-S stands for sequential. The past history of the state of the system can be utilized to predict the system's future state by incorporating it in the input features of the neural network. If we want to include the state history for p time steps, then the input of the neural network consists of $\boldsymbol{y}^{(n)}, \boldsymbol{y}^{(n-1)}, \ldots, \boldsymbol{y}^{(n-p+1)}$. Hence, the neural network will have $R \times p$ input features and R output labels (i.e., $\mathfrak{M} : \mathbb{R}^{R\times p} \Rightarrow \mathbb{R}^{R}$), where $\mathfrak{M}$ refers to the DNN model, and R is the number of components of the dynamical system. Once the neural network is trained and the weights are learned, the neural network is used to predict the state of the system starting with an initial condition $\boldsymbol{y}^{(0)}$ and proceeding in time iteratively. The prediction of the system in iterative fashion is shown in Fig. 1(a). If the neural network is trained using the state of the system for p time steps, then the previous p time steps should be stored in the prediction routine. The future state of the system $\boldsymbol{y}^{(n+1)}$ is predicted using the true state of the system for first p time steps. After p time steps, only the predicted values by the neural network are used in the input. The predicted variable of the neural network and the solution update formula during an iterative prediction are given in Table I.

For the DNN-R framework, we learn the difference between the state of the system at time step t_n and next time step t_{n+1}. The residual between two time steps is then applied to update the current state during prediction. The learning of the residual information instead of sequential update formula helps in stabilizing the neural network[123] and also improves the accuracy of neural network prediction.[113] The DNN-R framework can also be implemented using the history of the system's state, similar to the DNN-S framework. The related framework was employed for the model order reduction of the parametric viscous Burgers equation problem[125] with one temporal leg history (i.e., $p = 1$). They call it the POD-ANN-RN framework, and this framework was found to give better results for interpolatory and extrapolatory ROM than sequential learning. The iterative prediction of the dynamical system using the DNN-R framework is shown in Fig. 1(b). The future state of the system is computed using the solution update formula mentioned in Table I.

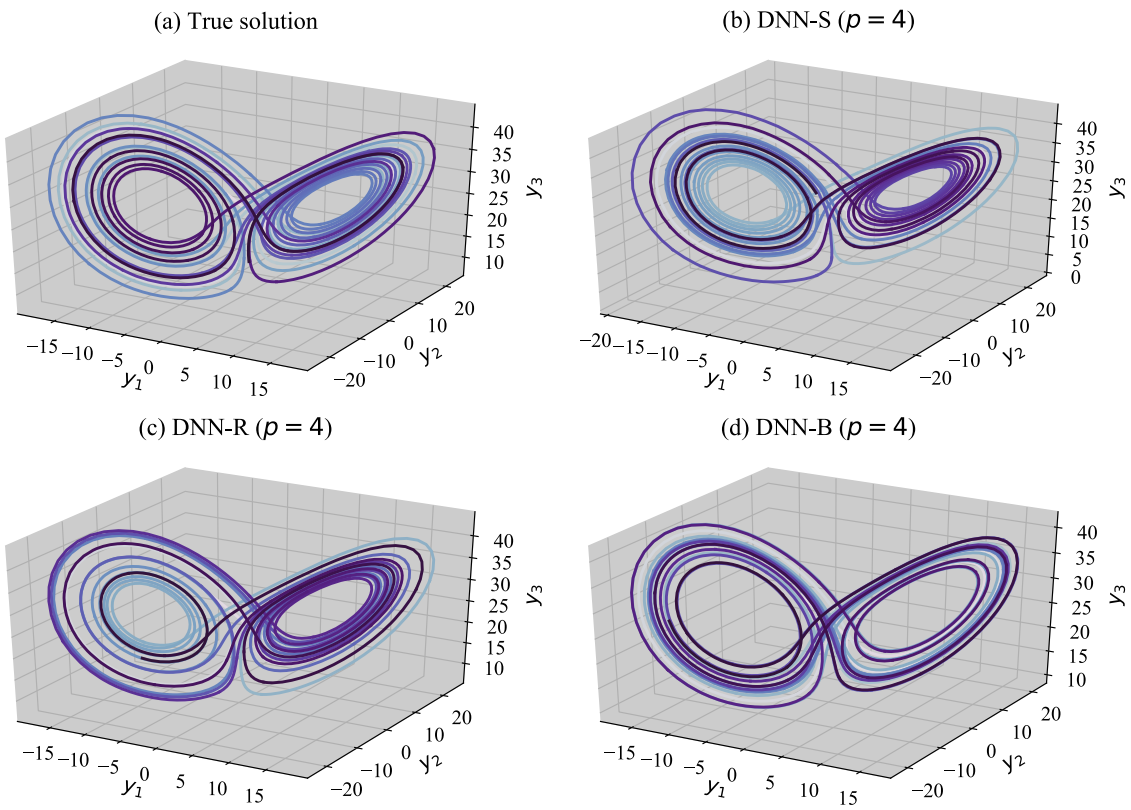

FIG. 7. The true phase portrait of the Lorenz system (a) compared with the phase portrait predicted by different DNN frameworks [(b)-(d)] for time integration from $t = 0$ to $t = 25$. Note that there is a good qualitative argument between the phase portrait predicted by the DNN-S framework (b), DNN-R framework (c), and DNN-B framework (d) with the true phase portrait (a). All DNN frameworks are trained with $p = 4$.

We also introduce an additional framework called a DNN-B framework, as shown in Fig. 1(c). The B in the name stands for the backward difference. In this framework, we use the second order backward difference numerical scheme to compute the slope at time step t_n. The numerical slope information is then used to update the state of the system during prediction. The similar approach is also implemented in other data-driven methods for dynamical systems. One such work is the multistep neural network[102] used for the data-driven discovery of nonlinear dynamical systems from experimental measurements of the state of the system. In their work, the evolution of system $\boldsymbol{F}$ is learned using the neural network by incorporating the multistep Adams-Moulton numerical scheme in computing the loss function of the neural network. In our work, we use the numerical slope as the predicted variable and use mean squared error as the loss function. One of the advantages of directly learning the numerical slope is that standard loss functions available in Keras library can be directly applied without any modification. We apply the second order backward difference scheme in the DNN-B framework to compute the discrete numerical slope. However, the framework can be implemented with any family of numerical schemes such as the central difference or forward difference family. The equation used to determine the numerical slope and the solution update formula during iterative prediction is provided in Table I.

The quantitative performance of each DNN framework is measured by a quantity root mean square error (RMSE). The root mean square is determined for each component k between the true state and the state predicted by the neural network from the initial time to final time. The root mean square of each component is added to get the total root mean square error. The root mean square error is defined as

$$\text{RMSE} = \sum_{k=1}^{R} \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_k^{(i)} - \tilde{y}_k^{(i)}\right)^2}, \tag{6}$$

where R is the total number of components of the dynamical system, N is the total number of time steps in the evolution of the dynamical system, $\boldsymbol{y}$ is the true solution, and $\tilde{\boldsymbol{y}}$ is the solution predicted by the neural network.

III. TIME SERIES PREDICTION FOR DYNAMICAL SYSTEMS

Before implementing our proposed DNN frameworks within the nonintrusive ROM setup, we demonstrate the capability of our DNN frameworks to model nonlinear dynamical systems using two examples. Section III A provides numerical results for the three mode Kraichnan-Orszag problem, and Sec. III B gives results for the chaotic Lorenz system.

A. Kraichnan-Orszag system

We start by considering the nonlinear dynamical system Kraichnan-Orszag[126] of order $R = 3$ as the first test problem.

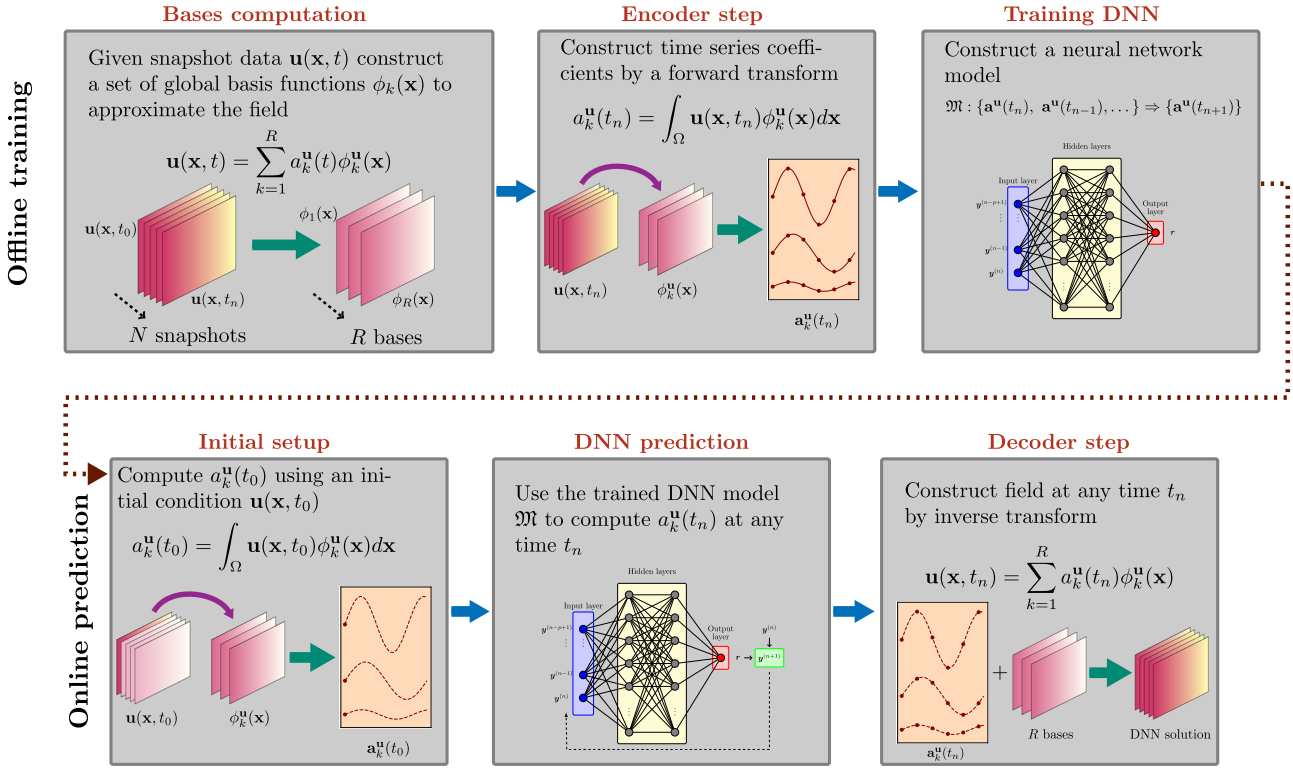

FIG. 8. Generalized nonintrusive reduced order modeling (NIROM) framework.

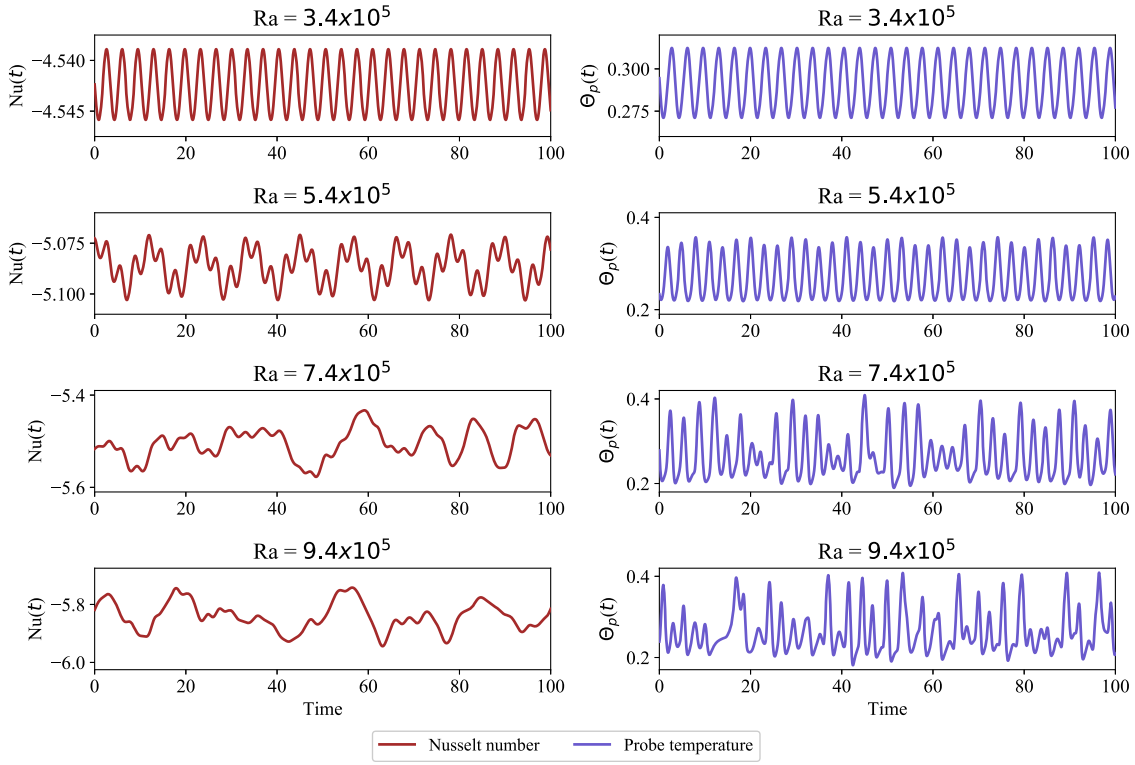

FIG. 9. Evolution of the Nusselt number Nu(t) at x = 0 (left) and designated probe temperature θ_p. The temperature θ_p is probed at the location x = 0.125 and y = 7.0. Note that the temperature variation at the probed location is periodic at low Rayleigh number. While at higher Rayleigh number the probed temperature varies in a chaotic manner.

The Kraichnan-Orszag system is defined as

$$\frac{dy_1}{dt} = y_1 y_3, \quad \frac{dy_2}{dt} = -y_2 y_3, \quad \frac{dy_3}{dt} = -y_1^2 + y_2^2, \tag{7}$$

with an initial condition $y_1(0) = 1$, $y_2(0) = 0.1\xi$, and $y_3(0) = 0$. The input ξ lies between the interval $[-1, 1]$. The modeling of this system is particularly challenging due to the discontinuity at planes $y_1(0) = 0$ and $y_2(0) = 0$.

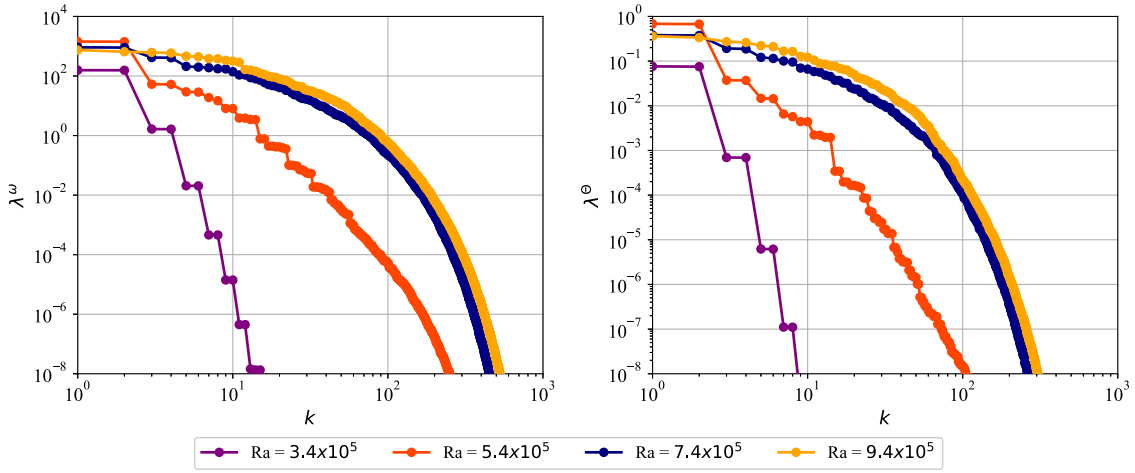

FIG. 10. Eigenvalues of the correlation matrix C using M = 1000 snapshots for different Rayleigh numbers.

The system is solved numerically using the SciPy function odeint for time integration between the time interval [0, 10] and $N = 1000$ time steps ($\Delta t = 0.01$). The initial condition considered for the system is $\xi = 0.5$ (i.e., $[y_1\ y_2\ y_3]^T = [1.0\ 0.05\ 0.0]^T$). The true solution is used for training the neural network. We apply the same hyperparameters for all DNN frameworks. We use three hidden layers with 128 neurons each. The maximum number of iterations is set to 600 and 10% of the data are used for validation to avoid overfitting.

The true state of the system and the state predicted by different DNN frameworks are shown in Fig. 2 for $p = 1$. All DNN frameworks are correctly able to predict all three states of the dynamical system. We also see that DNN-R and DNN-B frameworks perform better than the DNN-S framework. For DNN-S framework, the state predicted by the neural network is very slightly shifted from the original state. Figure 3 provides the numerical results for DNN frameworks for $p = 4$. The state predicted by all DNN frameworks is almost the same as the true state. Table II compares the RMSE calculated using Eq. (6) for all three DNN frameworks with different numbers of inputs to the neural network.

B. Lorenz system

The Lorenz system[127] can be described by the following equations:

$$\begin{aligned} \frac{dy_1}{dt} &= \alpha(y_2 - y_1), \\ \frac{dy_2}{dt} &= y_1(\rho - y_3) - y_2, \\ \frac{dy_3}{dt} &= y_1 y_2 - \beta y_3. \end{aligned} \tag{8}$$

For the Lorenz system, we use $\alpha = 10$, $\rho = 28$, and $\beta = 8/3$. The modeling of the dynamics of the Lorenz system is a challenging problem

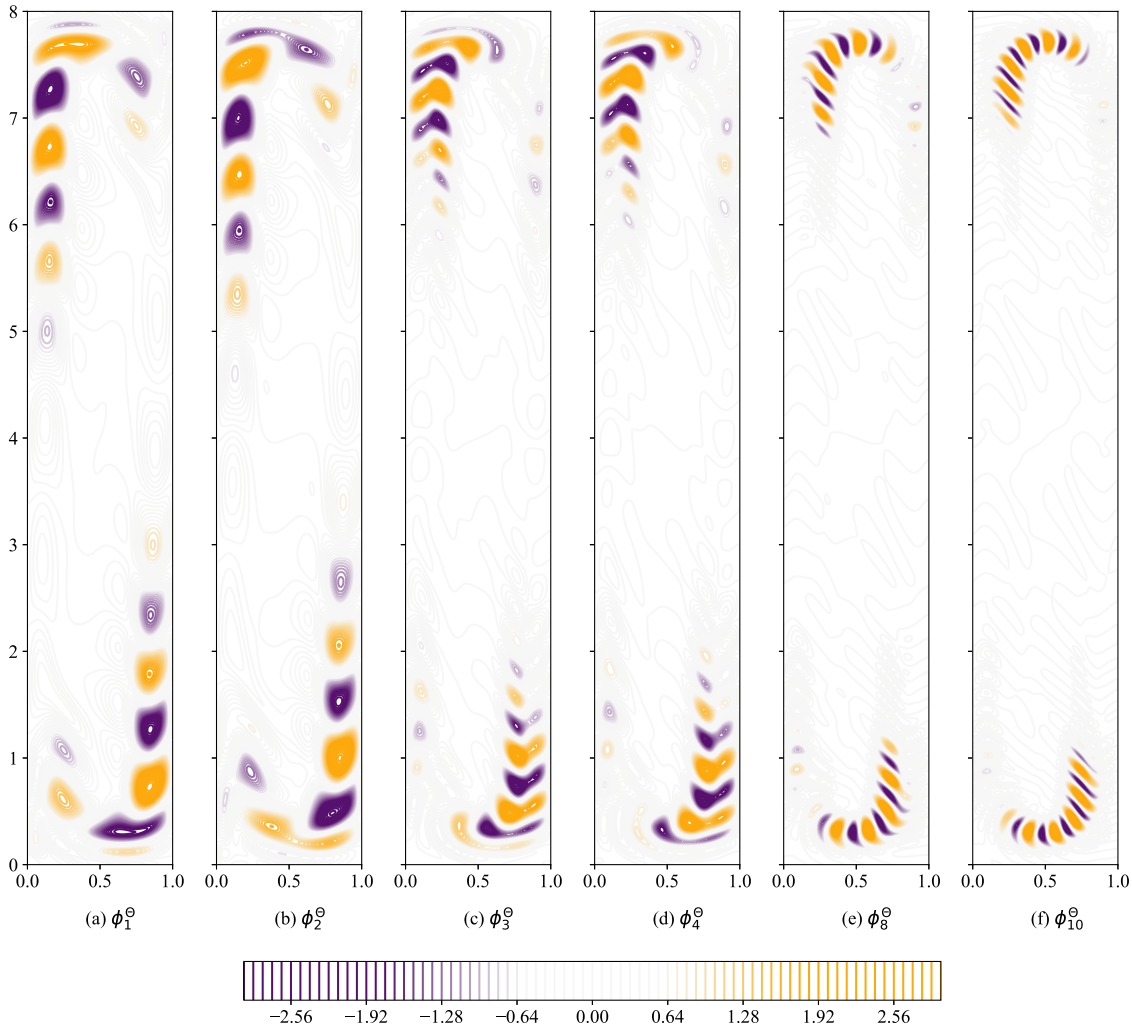

FIG. 11. Illustrative contour plots for some of the POD basis functions for the temperature field at Ra = 3.4×10^5 for the differentially heated cavity problem.

due to its highly nonlinear and chaotic behavior. This system arises in many simplified models for physical processes.[128,129]

The training data for the neural network are generated using the true solution to the Lorenz system. We use the initial condition $[y_1\ y_2\ y_3]^T = [-8\ 7\ 27]^T$ and generate the true solution numerically using the SciPy function odeint. The true solution is generated between the time interval [0, 25] with the time step $\Delta t = 0.01$ ($N = 2500$ time steps). We apply the same hyperparameters for all DNN frameworks. We use five hidden layers with 128 neurons each. The maximum number of iterations is set to 1000, and 10% of the training data are used for validation to avoid overfitting.

Figure 4 shows the true trajectory of the Lorenz system and the trajectory predicted by different DNN frameworks using $p = 1$ in the input training data. The time period for which the predicted trajectory is the same as the true trajectory varies for different DNN frameworks. The Lorenz system has a chaotic behavior, and a smaller error in the predicted state of the system can lead to a larger error in the forecasted state of the system. The time period for which the predicted trajectory follows the true trajectory is longer for the DNN-R and DNN-B frameworks than the DNN-S framework. Figure 5 shows similar results for $p = 4$ in input training data. If we compare Figs. 4(a) and 5(a), we see that the predicted trajectory follows the true trajectory longer as we include the temporal history of the state of the system in the input to the neural network. We do not see a similar behavior for DNN-R and DNN-B frameworks.

Even though the neural network is not able to predict the true trajectory of the individual state after some time, we should check the ability of the neural network to capture the overall dynamics of the Lorenz attractor. This is due to the fact that the Lorenz system has a positive Lyapunov exponent, and a small perturbation in the initial condition can cause the system to diverge exponentially. Hence, a number of studies check for the dynamics of the Lorenz attractor rather than the individual state of the system.[102,130] Figures 6 and 7 show the exact dynamics of the Lorenz attractor and

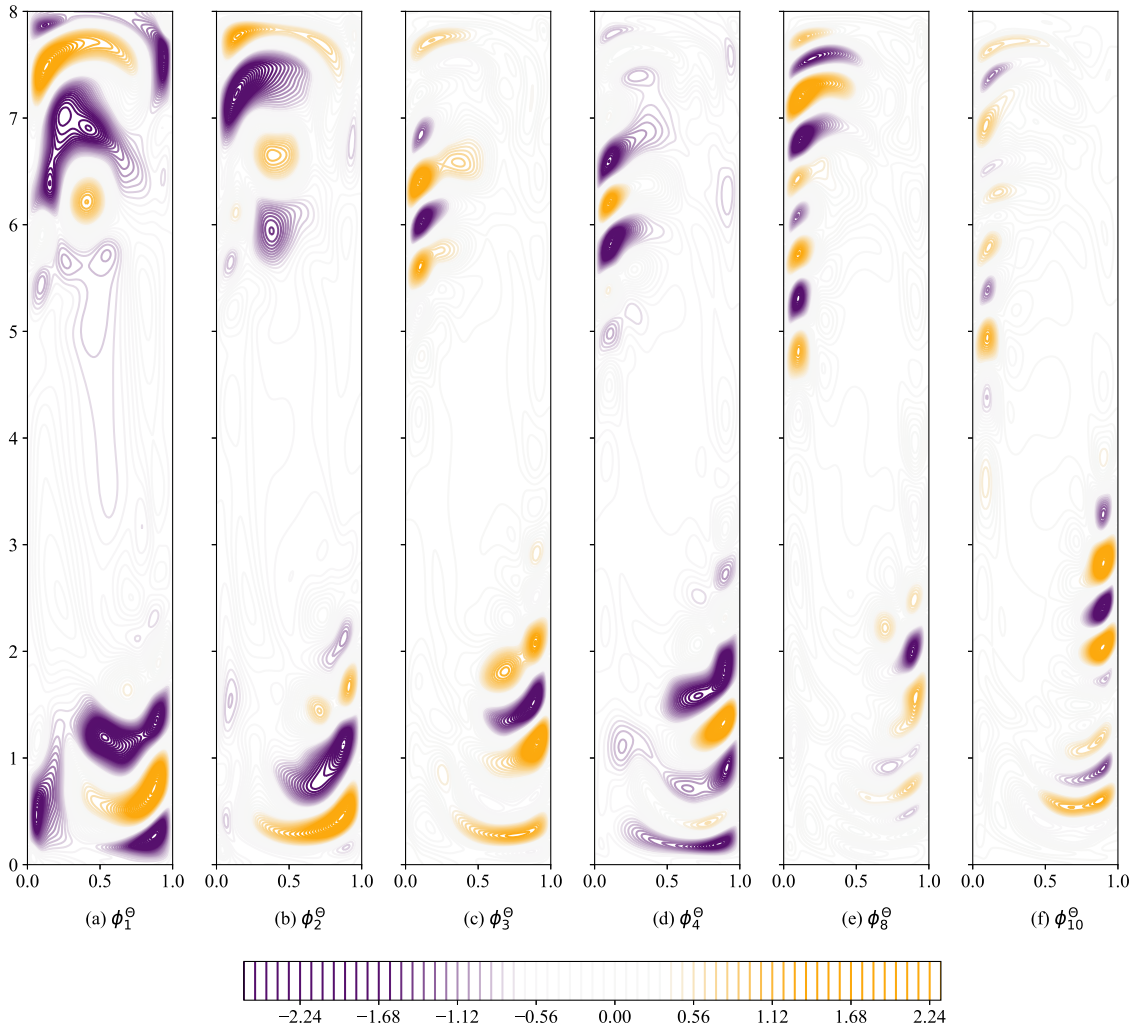

FIG. 12. Illustrative contour plots for some of the POD basis functions for the temperature field at Ra = 9.4×10^5 for the differentially heated cavity problem.

predicted dynamics for different DNN frameworks with $p = 1$ and $p = 4$, respectively. All DNN frameworks are capable of capturing the dynamics of the Lorenz system and accurately predict the correct shape of the Lorenz attractor.

IV. NONINTRUSIVE REDUCED ORDER MODELING (NIROM)

We observe a successful predictive performance for a simple nonlinear dynamical system (governed by a set of 3 coupled ordinary differential equations) using different DNN frameworks in Sec. III A. We also saw the ability of the neural network to capture the overall dynamics of the chaotic nonlinear Lorenz system. This has motivated us to test the proposed DNN frameworks for the model order reduction of a real-world test problem.

Indeed, we transform a partial differential equation system into a set of ordinary differential equations in order to get the reduced order model. Our main motivation of this study is to convey the message that the neural network architectures can be used in developing a robust and efficient nonintrusive reduced order model. With a goal to recover the reduced order model dynamics of the underlying flow phenomena, in this section, we implement all DNN frameworks introduced in Sec. II in model order reduction of a differentially heated cavity problem. This test problem setup is well-established as a model validation test case due to the problem's simplicity in terms of problem definition as well as the wide variety of applications such as nuclear reactor core isolation, solar energy storage, and so on.[131–133] At first, we will describe our nonintrusive ROM framework. Then, we will define our test problem setup along with the governing equations, and then we will briefly demonstrate our nonintrusive ROM framework for the Boussinesq equation problem. Finally, we will demonstrate the comparative performance of the aforementioned DNN frameworks in terms of the temporal evolution of vorticity and temperature field and contours of temperature field for a given initial condition in Sec. V.

A. NIROM framework

To develop our NIROM framework, we define a generalized partial differential equation (PDE) system as follows:

$$\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} = \mathfrak{R}(\mathbf{u}(\mathbf{x},t);\mathbf{x},t), \tag{9}$$

where $\mathbf{u}$ refers to the problems of interest (e.g., velocity, pressure, and temperature, etc) and $\mathfrak{R}$ converts the physical process possibly with linear, nonlinear, and forcing terms.

In our NIROM framework, we assume that we do not have access to the $\mathfrak{R}(\mathbf{u}(\mathbf{x},t);\mathbf{x},t)$ operator. We formulate the proposed NIROM framework assuming that we have discrete snapshots of $\mathbf{u}(\mathbf{x}, t)$. The detailed steps of our NIROM framework are outlined in Algorithm 1. We highlight that this physics-agnostic modeling approach is quite modular and decomposes the problem into the basis representation and forecasting problems. Figure 8 depicts

ALGORITHM 1. NIROM framework.

Offline training

1: We pick or construct a set of orthonormal basis functions $\phi_k^{\mathbf{u}}(\mathbf{x})$ over domain Ω (e.g., Fourier bases, POD bases, etc)

$$\mathbf{u}(\mathbf{x},t) = \sum_{k=1}^{R} a_k^{\mathbf{u}}(t)\phi_k^{\mathbf{u}}(\mathbf{x}), \tag{10}$$

such that

$$\int_\Omega \phi_i^{\mathbf{u}}(\mathbf{x})\phi_j^{\mathbf{u}}(\mathbf{x})d\mathbf{x} = \delta_{ij}, \tag{11}$$

where δ_{ij} is the Kronecker delta operator, and $\mathbf{u}(\mathbf{x}, t)$ is approximated from the span of these bases.

2: Encoder step: construct time series coefficients by a forward transform

$$a_k^{\mathbf{u}}(t_n) = \int_\Omega \mathbf{u}(\mathbf{x},t_n)\phi_k^{\mathbf{u}}(\mathbf{x})d\mathbf{x}. \tag{12}$$

3: Train a time series forecasting model (e.g., a DNN model discussed in Sec. II)

$$\mathfrak{M} : \{\mathbf{a}^{\mathbf{u}}(t_n), \mathbf{a}^{\mathbf{u}}(t_{n-1}), \ldots \mathbf{a}^{\mathbf{u}}(t_{n-p+1})\} \Rightarrow \mathbf{a}^{\mathbf{u}}(t_{n+1}). \tag{13}$$

Online prediction

4: Given an initial condition $\mathbf{u}(\mathbf{x}, t_0)$, compute $a_k^{\mathbf{u}}(t_0)$ using below relation,

$$a_k^{\mathbf{u}}(t_0) = \int_\Omega \mathbf{u}(\mathbf{x},t_0)\phi_k^{\mathbf{u}}(\mathbf{x})d\mathbf{x}. \tag{14}$$

5: Use the trained DNN model $\mathfrak{M}$ to predict $a_k^{\mathbf{u}}(t_n)$ at any time t_n.

6: Decoder step: construct the field at any time t_n by inverse transform,

$$\mathbf{u}(\mathbf{x},t_n) = \sum_{k=1}^{R} a_k^{\mathbf{u}}(t_n)\phi_k^{\mathbf{u}}(\mathbf{x}). \tag{15}$$

various stages of the NIROM framework for any generalized PDE system.

B. Problem definition: Boussinesq equations

For this numerical experiment, we investigate the convective flow behavior in a two-dimensional buoyancy-driven flow in a differentially heated tall cavity.[134] We consider the following dimensionless form of the two-dimensional incompressible Boussinesq equations[65,135–137] on our computational domain, Ω:

$$\nabla \cdot \mathbf{u} = 0, \tag{16}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{u} + \mathrm{Ri}\,\theta\,\hat{e}_j, \tag{17}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u} \cdot \nabla)\theta = \frac{1}{\mathrm{Re\,Pr}}\nabla^2 \theta, \tag{18}$$

where $\mathbf{u}$ refers to the velocity vector in two dimensions, p and θ denote the pressure and temperature fields, respectively, and $\hat{e}_j$ is the unit vector in the y direction. Here, ∇ and ∇^2 are the standard two-dimensional differential and Laplacian operators, respectively. In general, the Boussinesq equations are characterized by three dimensionless numbers: Reynolds number (Re), Prandtl number (Pr), and Richardson number (Ri). However, other relevant dimensionless numbers can be introduced into the equation as a control parameter based on the physics of the system. For example, we introduce the Rayleigh number (Ra) in our study due to the natural convection heat transfer which can be expressed as

$$\mathrm{Ra} = \mathrm{Ri\,Re}^2\,\mathrm{Pr}. \tag{19}$$

In our problem setup, we fix Pr = 0.71, Ri = 1. In our two-dimensional full order model (FOM) simulation, we utilize the vorticity-streamfunction formulation to avoid numerical complexity associated with the primitive variable formulation.[137] Therefore, we introduce the vorticity ($\omega = \nabla \times \mathbf{u}$) and streamfunction ($\psi$) for Eq. (16) specified by the following coupled equations:[65,133]

$$\frac{\partial \omega}{\partial t} + \mathrm{J}(\omega, \psi) = \frac{1}{\mathrm{Re}}\nabla^2 \omega + \mathrm{Ri}\frac{\partial \theta}{\partial x}, \tag{20}$$

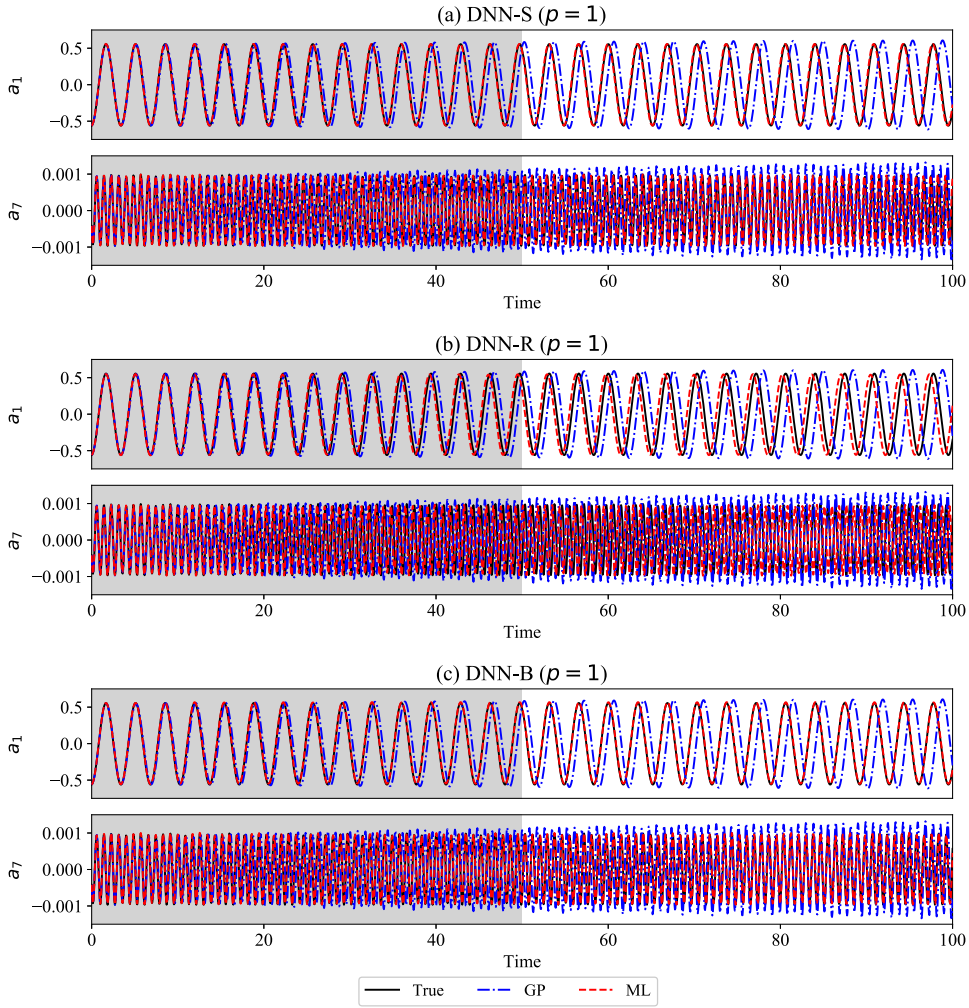

FIG. 13. Evolution of temporal coefficients for the vorticity transport equation at Ra = 3.4 × 10^5 for different frameworks with $p = 1$. The neural network is trained using the data highlighted in light gray in the figure.

$$\frac{\partial \theta}{\partial t} + \mathrm{J}(\theta, \psi) = \frac{1}{\mathrm{Re\,Pr}} \nabla^2 \theta, \tag{21}$$

where Jacobian J accounts for the nonlinear advection term, which is defined as

$$\mathrm{J}(f, g) = \frac{\partial g}{\partial y}\frac{\partial f}{\partial x} - \frac{\partial g}{\partial x}\frac{\partial f}{\partial y}. \tag{22}$$

The flow velocity components can be found from the streamfunction, ψ, using the following definitions:

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{23}$$

The kinematic equation connecting the vorticity and streamfunction can be found by substituting the velocity components in terms of streamfunction, which form the following divergence-free constraint satisfying the Poisson equation:

$$\nabla^2 \psi = -\omega. \tag{24}$$

Our Cartesian computational domain is $(x, y) \in [0, 1] \times [0, 8]$. We utilize wall boundary conditions for all four sides of our computational domain with an adiabatic condition on the top and bottom walls. We imply Dirichlet conditions on left ($\theta = 0.5$) and right ($\theta = -0.5$) walls. At the cavity walls, we enforce no-slip boundary conditions by setting zero values for streamfunction and vorticity values calculated by Briley's formula.[138] A detailed derivation and discussion on our test problem setup along with the numerical schemes for generating snapshots through FOM simulation can be found in a recent study conducted by San and Maulik.[137] Note that, even though the simulation is performed for a maximum time of $t = 1000$, we perform all our statistical analysis from $t = 900$ (after an initial transient period) at 128×1024 grid resolutions, collect snapshot data only between $t = 900$ and $t = 950$, and perform our quantitative analyses for the prediction of both in-sample data zone (between $t = 900$ and $t = 950$) and the out-of-sample data zone (between $t = 950$ and $t = 1000$). For the rest of this paper, we shall refer to this initial state (i.e., time $t = 900$ in our physical simulation) as $t = 0$ for prescribing initial conditions for ROMs. Therefore, our

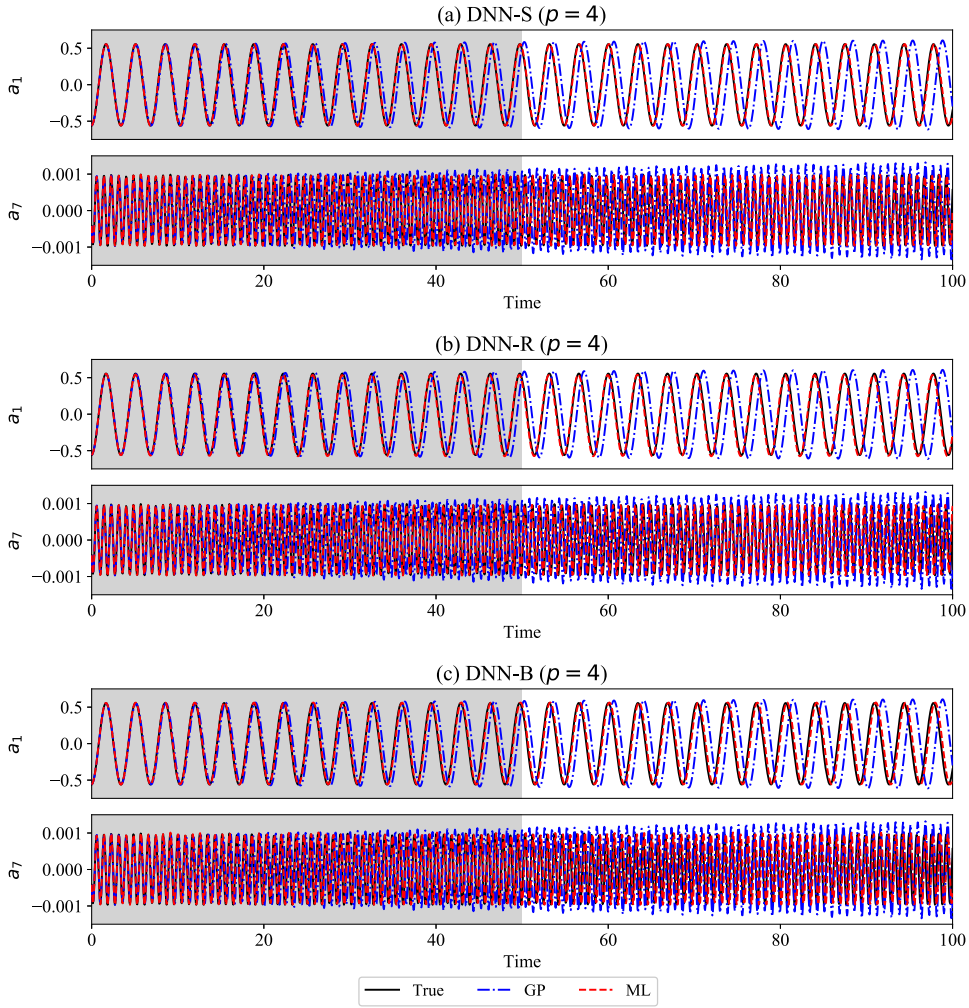

FIG. 14. Evolution of temporal coefficients for vorticity transport equation at $\mathrm{Ra} = 3.4 \times 10^5$ for different frameworks with $p = 4$. The neural network is trained using the data highlighted in light gray in the figure.

in-sample data zone spans between $t = 0$ and $t = 50$ (where we store 1000 snapshots for the training purposes), and our out-of-sample data zone spans between $t = 50$ and $t = 100$ (where we store additional 1000 snapshots for the testing purposes). We use the data in the in-sample-zone for training the neural network.

The DNS technique used for performing the FOM simulation is validated by recording the simulation statistics at a designed probe point and comparing it with the studies in the literature.[139] We simulate the differentially heated cavity problem for four different Rayleigh numbers. The flow is smooth and orderly (i.e., time-periodic) for lower Rayleigh numbers. As we increase the Rayleigh number, the flow starts getting chaotic and turbulent. We calculate the Nusselt number along the vertical left wall ($x = 0$) using the following formula:

$$\mathrm{Nu}(t) = \frac{1}{H}\int_0^H \frac{\partial\theta}{\partial x}\Big|_{x=0} dy, \tag{25}$$

where $H = 8$. Figure 9 shows the statistics of the Nusselt number along the left wall and the designated probe temperature ($x = 0.125$ and $y = 7.0$). It can be seen that the flow behavior is periodic for both the Nusselt number and the temperature history at the probed location at lower Rayleigh numbers. As the Rayleigh number increases, we see that the Nusselt number and probed temperature variation are not periodic due to the turbulent nature of flow. We test our nonintrusive framework for $\mathrm{Ra} = 3.4 \times 10^5$ where the flow is periodic and for $\mathrm{Ra} = 9.4 \times 10^5$ where the flow is chaotic.

C. NIROM framework for Boussinesq equations

In this section, we present our nonintrusive ROM setup for the unsteady, incompressible Boussinesq equations given by Eqs. (20) and (21). In our ROM settings, we first compute the desired set of orthogonal spatial basis functions from stored high-fidelity data snapshots using proper orthogonal decomposition (POD). We obtain the data snapshots from a high-resolution FOM simulation. Using the precomputed basis functions, we develop the nonintrusive framework in an encoder-decoder approach using different DNN frameworks proposed in Sec. II. To compare the predictive performance of the nonintrusive ROMs with respect to the standard

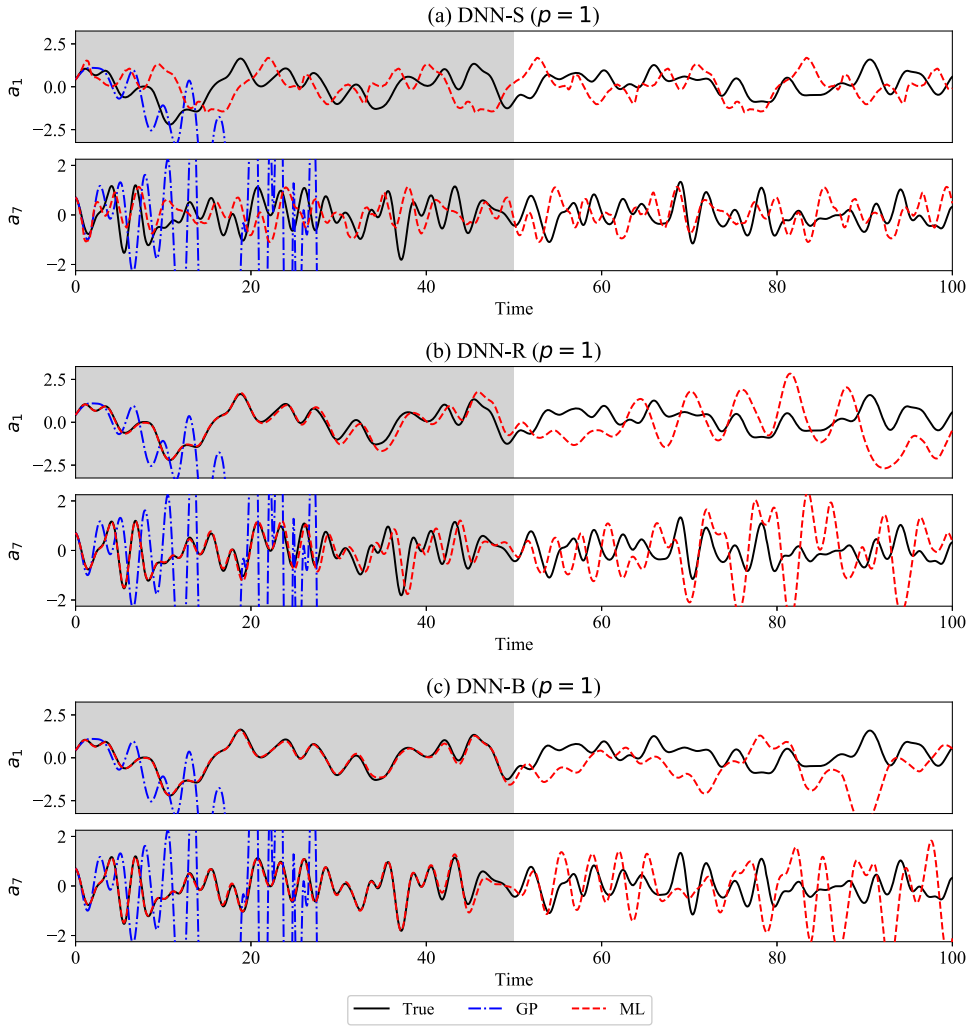

FIG. 15. Evolution of temporal coefficients for the vorticity transport equation at $\mathrm{Ra} = 9.4 \times 10^5$ for different frameworks with $p = 1$. The neural network is trained using the data highlighted in light gray in the figure.

intrusive ROM, we develop our intrusive ROM framework (ROM-G) using the Galerkin projection to derive the dynamical model for the POD coefficients.[8,58,140] The implementation of Galerkin projection for the underlying test problem is detailed in the recent work by San and Maulik[137] and, hence, is not discussed in the present work. Here, we demonstrate the development of the nonintrusive ROM methodologies briefly before proceeding to the numerical results for analyses.

The POD bases for the vorticity field can be constructed from the field variable $\omega(x, y)$ at different time steps which we denote as snapshots, i.e., for M number of snapshots, $\omega(\mathbf{x}, t_n)$ are the stored snapshots for $n = 1, 2, \ldots, M$. The time-averaged field can be computed as

$$\bar{\omega}(\mathbf{x}) = \frac{1}{M}\sum_{n=1}^{M}\omega(\mathbf{x}, t_n). \tag{26}$$

To map the snapshot data to its origin, we then compute the mean-subtracted snapshots or the fluctuating fields by

$$\omega'(\mathbf{x}, t_n) = \omega(\mathbf{x}, t_n) - \bar{\omega}(\mathbf{x}). \tag{27}$$

This subtraction guarantees that the ROM solution would satisfy the same boundary conditions as the full order model.[141] To simplify the eigenvalue problem necessary for POD bases calculation, we utilize the standard method of snapshots proposed by Sirovich[142] which reduces the larger dimension problem to a much smaller dimension problem. To do so, we construct an $M \times M$ correlation matrix of the fluctuating part $C = [c_{ij}]$ which is computed from the inner product of the mean-subtracted snapshots,

$$c_{ij} = \langle \omega'(\mathbf{x}, t_i), \omega'(\mathbf{x}, t_j) \rangle, \tag{28}$$

where i and j refer to the snapshot indices. The definition of the inner product of any two arbitrary fields f and g can be expressed as

$$\langle f, g \rangle = \int_{\Omega} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}. \tag{29}$$

In the current study, we use the well-known Simpson's integration rule for a numerical computation of the inner products. Next, an eigendecomposition of the C matrix is performed by solving

$$CW = W\mathbf{\Lambda}, \tag{30}$$

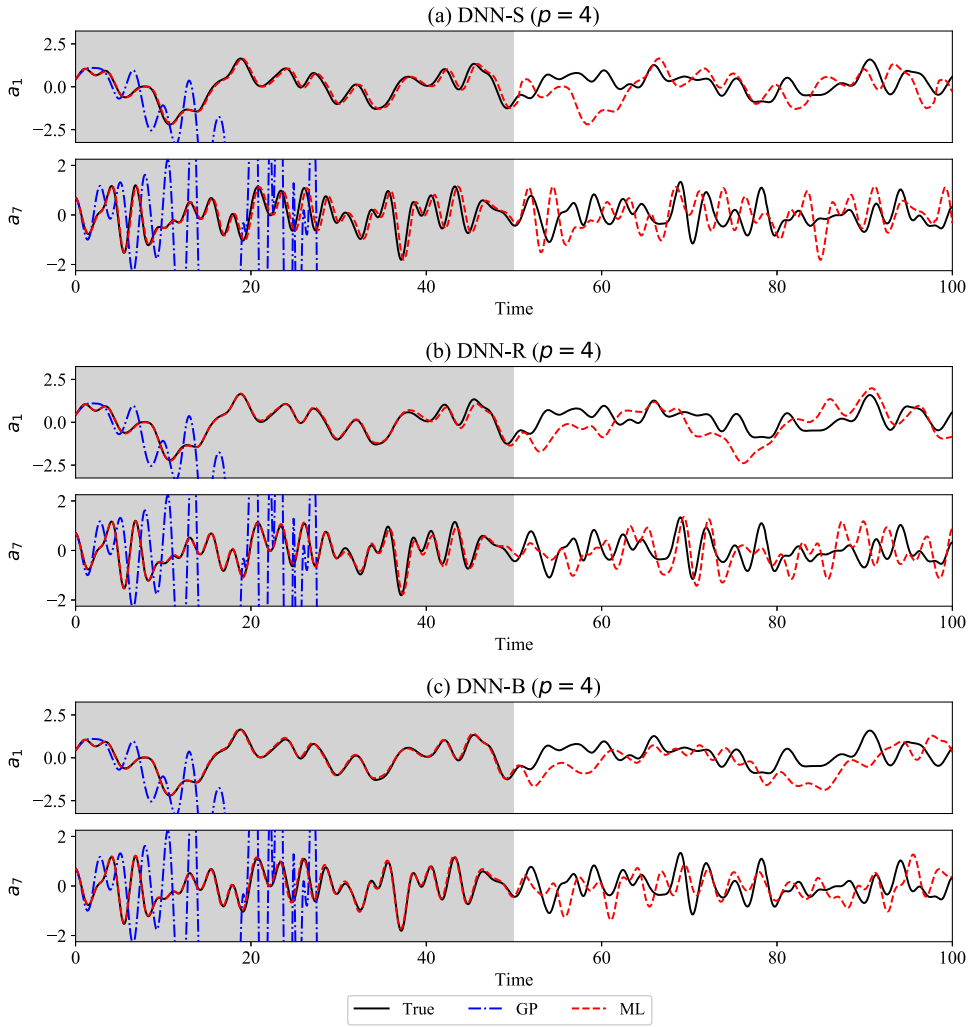

FIG. 16. Evolution of temporal coefficients for the vorticity transport equation at Ra = 9.4 × 10^5 for different frameworks with $p = 4$. The neural network is trained using the data highlighted in light gray in the figure.

where $\mathbf{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues λ_k^ω of C, and W is a matrix whose columns w_k are the corresponding eigenvectors. This has been shown in detail in various POD literature (see, e.g., Refs. 63 and 142). It should be noted that eigenvalues need to be arranged in a descending order (i.e., $\lambda_1^\omega \geq \lambda_2^\omega \geq \cdots \geq \lambda_M^\omega$), for proper selection of the POD modes. The POD modes of vorticity field ϕ_k^ω are then computed as

$$\phi_k^\omega(\mathbf{x}) = \frac{1}{\sqrt{\lambda_k^\omega}} \sum_{n=1}^{M} w_k^n \omega'(\mathbf{x}, t_n), \tag{31}$$

where w_k^n is the nth component of the eigenvector W. The scaling factor, $\left(\frac{1}{\sqrt{\lambda_k^\omega}}\right)$, is to guarantee the orthonormality of POD modes, i.e., $\langle \phi_i^\omega, \phi_j^\omega \rangle = \delta_{ij}$. Here, δ_{ij} is the Kronecker delta defined by

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}. \tag{32}$$

Similarly, we can compute the POD bases for the temperature field which is $\phi_k^\theta(\mathbf{x})$.

Figure 10 shows the decay of eigenvalues of the correlation matrix for vorticity and temperature field. We can observe that the first 10 modes are able to capture more than 99% of the total energy for both vorticity and temperature at lower Rayleigh numbers ($\mathrm{Ra} = 3.4 \times 10^5$ and $\mathrm{Ra} = 5.4 \times 10^5$). Also, there is a faster decay of eigenvalues for lower Rayleigh numbers than for higher Rayleigh numbers. For higher Rayleigh numbers, the first 10 modes are not sufficient to capture a major portion of the energy. For example, the first 10 modes of vorticity field capture only 76% and 69% of the total energy for vorticity at Rayleigh numbers $\mathrm{Ra} = 7.4 \times 10^5$ and $\mathrm{Ra} = 9.4 \times 10^5$, respectively. In order to capture more than 95% of the energy, we will need to consider more than 40 modes. If we include 40 modes, then the Galerkin projection becomes computationally expensive and we lose the benefit of ROM framework. If we include only 10 modes, then the Galerkin projection is unbounded, as we will see in Sec. V, and it gives a physically wrong solution. We demonstrate that our nonintrusive framework gives sufficiently accurate results comparable to the true projection of the FOM solution on reduced order space even with less number of POD modes. It is evident that if we include a higher number of modes to capture the total energy, the true projection of the FOM solution on lower dimensional bases will approximate the FOM solution. However, the computational burden will also go up with an increased number of modes. In Fig. 11, we provide the contour plots for a few POD basis for the temperature field θ to indicate the structure of the solution at $\mathrm{Ra} = 3.4 \times 10^5$. We can observe that the solution is smooth and periodic for the lower Rayleigh number case and only small structures are truncated after 10 POD modes. On the other hand, we observe from Fig. 12 that there are still some of the large structures remaining in the 10th mode for the higher Rayleigh number case. This means that some of the important flow features are truncated due to consideration of only the first 10 dominant POD modes.

To obtain the nonintrusive ROM, we utilize an encoder-decoder approach to transfer data from full order space to reduced order space and vice versa. During the encoder stage, we transform data from the full order space to the reduced order space by using the following projection for both field parameters:

$$a_k(t) = \langle \omega(\mathbf{x}, t) - \bar{\omega}(\mathbf{x}), \phi_k^\omega(\mathbf{x}) \rangle, \tag{33}$$

$$b_k(t) = \langle \theta(\mathbf{x}, t) - \bar{\theta}(\mathbf{x}), \phi_k^\theta(\mathbf{x}) \rangle. \tag{34}$$

So, at initial time $t = 0$, we can compute the initial conditions by

$$a_k(0) = \left\langle \omega(\mathbf{x}, 0) - \bar{\omega}(\mathbf{x}), \phi_k^\omega(\mathbf{x}) \right\rangle, \tag{35}$$

$$b_k(0) = \left\langle \theta(\mathbf{x}, 0) - \bar{\theta}(\mathbf{x}), \phi_k^\theta(\mathbf{x}) \right\rangle, \tag{36}$$

where $\omega(\mathbf{x}, 0)$ and $\theta(\mathbf{x}, 0)$ are the vorticity and temperature field, respectively, specified at initial time. Here, $a_k(t)$ and $b_k(t)$ are the time-dependent modal coefficients for the vorticity and temperature, respectively. This will form initial conditions for the underlying ordinary differential equations similar to the problem in Sec. III where we require solving da_k/dt and db_k/dt until the final time. As discussed earlier, we utilize the different DNN frameworks and time histories as input to predict the temporal evolution of $a_k(t)$ and $b_k(t)$. For the prediction with sequential time history data, we predict the next time step using the DNN-S framework with one ($p = 1$) and four ($p = 4$) leg temporal history in the input. We also employ DNN-R and DNN-B frameworks which predicts the residual and the numerical slope, respectively, based on one ($p = 1$) and four ($p = 4$) leg temporal history in the input to the neural network. In the decoder stage, we reconstruct the reduced order solution to the full order solution by using the following definition:

$$\omega(\mathbf{x}, t) = \bar{\omega}(\mathbf{x}) + \sum_{k=1}^{R} a_k(t) \phi_k^\omega(\mathbf{x}), \tag{37}$$

TABLE III. Quantitative assessment of Galerkin projection and different DNN frameworks for vorticity and temperature modal coefficients for $\mathrm{Ra} = 3.4 \times 10^5$ and $\mathrm{Ra} = 9.4 \times 10^5$ using the total root mean square error given by Eq. (6).

Framework	RMSE (a)	RMSE (b)
$\mathrm{Ra} = 3.4 \times 10^5$		
ROM-G	9.8×10^{-1}	2.1×10^{-2}
DNN-S ($p = 1$)	9.9×10^{-2}	6.3×10^{-3}
DNN-R ($p = 1$)	4.8×10^{-1}	1.3×10^{-2}
DNN-B ($p = 1$)	2.9×10^{-2}	5.3×10^{-3}
DNN-S ($p = 4$)	9.1×10^{-2}	8.8×10^{-4}
DNN-R ($p = 4$)	1.2×10^{-1}	2.2×10^{-4}
DNN-B ($p = 4$)	2.0×10^{-1}	5.9×10^{-4}
$\mathrm{Ra} = 9.4 \times 10^5$		
ROM-G	3.1×10^{2}	1.3×10^{0}
DNN-S ($p = 1$)	8.9×10^{0}	1.6×10^{-1}
DNN-R ($p = 1$)	8.3×10^{0}	1.3×10^{-1}
DNN-B ($p = 1$)	7.7×10^{0}	1.7×10^{-1}
DNN-S ($p = 4$)	4.9×10^{0}	1.8×10^{-1}
DNN-R ($p = 4$)	5.6×10^{0}	1.0×10^{-1}
DNN-B ($p = 4$)	4.6×10^{0}	1.0×10^{-1}

$$\theta(\mathbf{x},t) = \bar{\theta}(\mathbf{x}) + \sum_{k=1}^{R} b_k(t)\phi_k^{\theta}(\mathbf{x}), \tag{38}$$

where R ($\ll M$) is the retained most energetic POD modes. Since we do not use any physical equations for the time integration of the solution field, we can say our ROM setup using DNN is fully nonintrusive.

V. NUMERICAL RESULTS FOR BOUSSINESQ EQUATIONS

To evaluate the performance of our DNN frameworks within the underlying nonintrusive ROM setup, we present the time series evolution for the vorticity transport equation. In addition, we compare the temperature field at the final time (i.e., $t = 100$) predicted using nonintrusive ROM framework with FOM simulation and its projection on reduced order space (true projection). We further evaluate the performance of DNN frameworks in predicting engineering quantities of interest such as the time-averaged Nusselt number. We also perform the quantitative assessments of different model's predictive performance in terms of the RMSE calculated using Eq. (6). We present our analysis for two Rayleigh numbers: Ra = 3.4×10^5 where the flow is smooth and periodic and Ra = 9.4×10^5 where the flow is turbulent and chaotic.

For all DNN frameworks, we use six hidden layers with 120 neurons each. The maximum number of iterations is set to 900, and 10% of the data are used for validation to avoid overfitting. The root mean square error used for evaluating the quantitative performance of DNN frameworks measures the difference between the true data and the predicted data for each mode. Hence, we present the true and predicted trajectories by the neural network for only two modes a_1 and a_7 for conciseness. Additionally, we compare the modal coefficient trajectory with the POD Galerkin projection (GP) trajectories

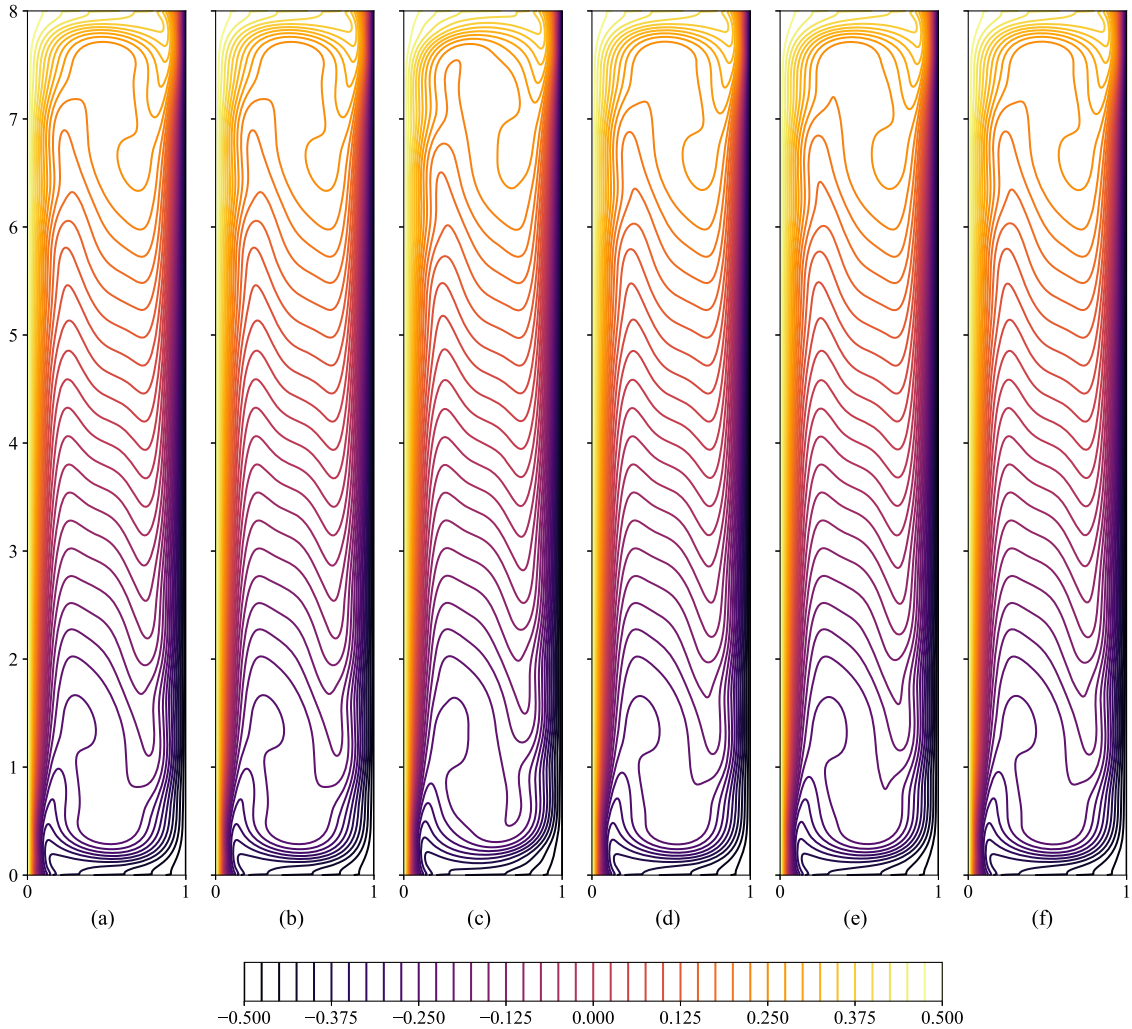

FIG. 17. Contours of instantaneous temperature at Ra = 3.4×10^5. The neural network is trained for different frameworks with $p = 1$. (a) FOM, (b) true projection, (c) ROM-G, (d) DNN-S framework, (e) DNN-R framework, (f) DNN-B framework.

(i.e., time dependent amplitude coefficients of ROM-G). It should be noted that our criteria for selection of hyperparameters are not necessarily optimal but are based on heuristics (involving different activation functions, number of layers/neurons/iterations, etc.) that enable our neural networks to accurately predict time evolution of dynamical systems. We also highlight that there are statistical methods available to select hyperparameters such as Bayesian optimization.[143]

As illustrated in Fig. 13 and for the rest of our analysis in this section, we display the true solution as a black solid line and regular POD-Galerkin projection based ROM, i.e., ROM-G solution, as a blue dashed-dotted line. The solution predicted by the neural network is shown by the dashed red line. The training data for the neural network are taken from the time series of modal coefficients obtained by projecting the FOM solution on the reduced order space between $t = 0$ and $t = 50$. This is consistent with the snapshot data used for generating the POD bases. The training data for the neural network are highlighted using the light gray in all time series plots. When we test the neural network, we start with an initial condition at $t = 0$ and proceed in an iterative fashion, as discussed in Sec. II. Therefore, the data between $t = 0$ and $t = 50$ are in-sample data and $t = 50$ and $t = 100$ are the out-of-sample data. The neural network has seen the in-sample data during training and hence is expected to give a good prediction for that time period. The question to ask is how does a neural network predict for the out-of-sample data.

Figure 13 shows the time evolution of two modal coefficients of the vorticity transport equation for all DNN frameworks using $p = 1$ in the input training data for $\mathrm{Ra} = 3.4 \times 10^5$. It can be clearly seen that there is a phase difference between the true projection and the Galerkin projection for the first modal coefficient a_1. Also, the Galerkin projection predicts slight amplification in the

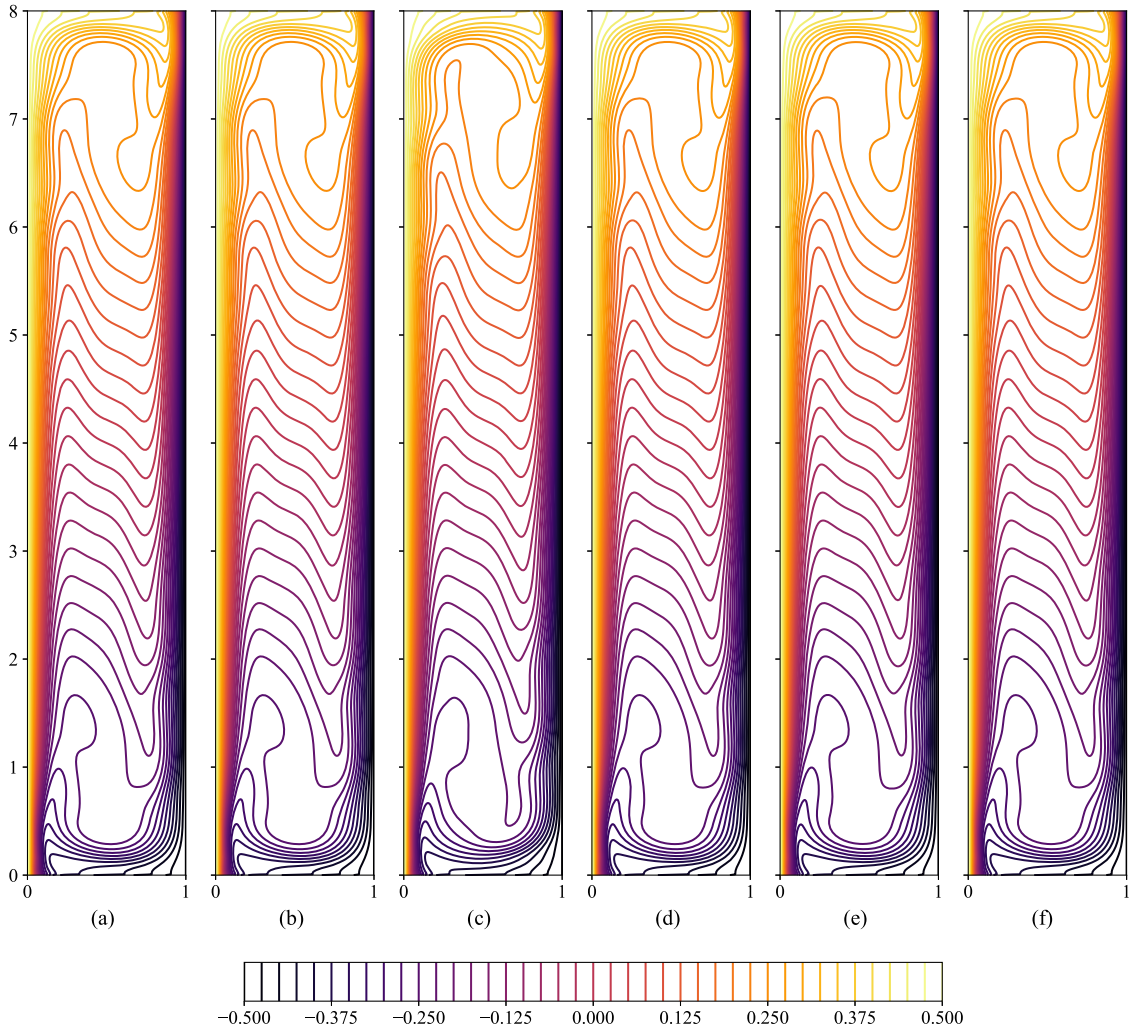

FIG. 18. Contours of instantaneous temperature at $\mathrm{Ra} = 3.4 \times 10^5$. The neural network is trained for different frameworks with $p = 4$. (a) FOM, (b) true projection, (c) ROM-G, (d) DNN-S framework, (e) DNN-R framework, (f) DNN-B framework.

amplitude of a_1, especially near the final time. The similar amplification in magnitude is also observed for a_7 with the Galerkin projection. The prediction by all DNN frameworks is better than the Galerkin projection, and the prediction is very close to the true projection of the FOM solution on reduced order space. For the DNN-R framework, there is a slight phase shift in DNN prediction with respect to the true projection as the time proceeds. The prediction for the DNN-R framework can be improved by including the past history of the modal coefficients in the input training data. Figure 14 shows the DNN prediction with $p = 4$ in the input training data. We can see that the prediction is improved for the DNN-R framework using $p = 4$. The prediction for DNN-S and DNN-B frameworks was already good using $p = 1$ and remains the same with $p = 4$ also. We see a similar prediction for other modal coefficients also.

The results presented in Figs. 13 and 14 were for Ra = 3.4×10^5. At a lower Rayleigh number, the flow is smooth and orderly, and hence, the evolution of modal coefficients was periodic. This is a simple problem with stationary time series and can be solved using simple methods like extreme learning machine.[100,144] The DNN will be beneficial for the higher Rayleigh number case after the onset of turbulence. Figures 15 and 16 show similar results for Ra = 9.4×10^5. The modal coefficients are not periodic due to the chaotic and turbulent nature of flow taking place in the cavity at such a higher Rayleigh number. There is a considerable variation in the evolution of the modal coefficients as the time proceeds. The Galerkin projection is unbounded with less number of modes and gives nonphysical results for such complex flows. In order to recover the correct physics using Galerkin projection, we will have to use an increased number of modes and this will lead to increased computational cost. However, we are interested in recovering the accurate physics as much as possible with less computational cost.

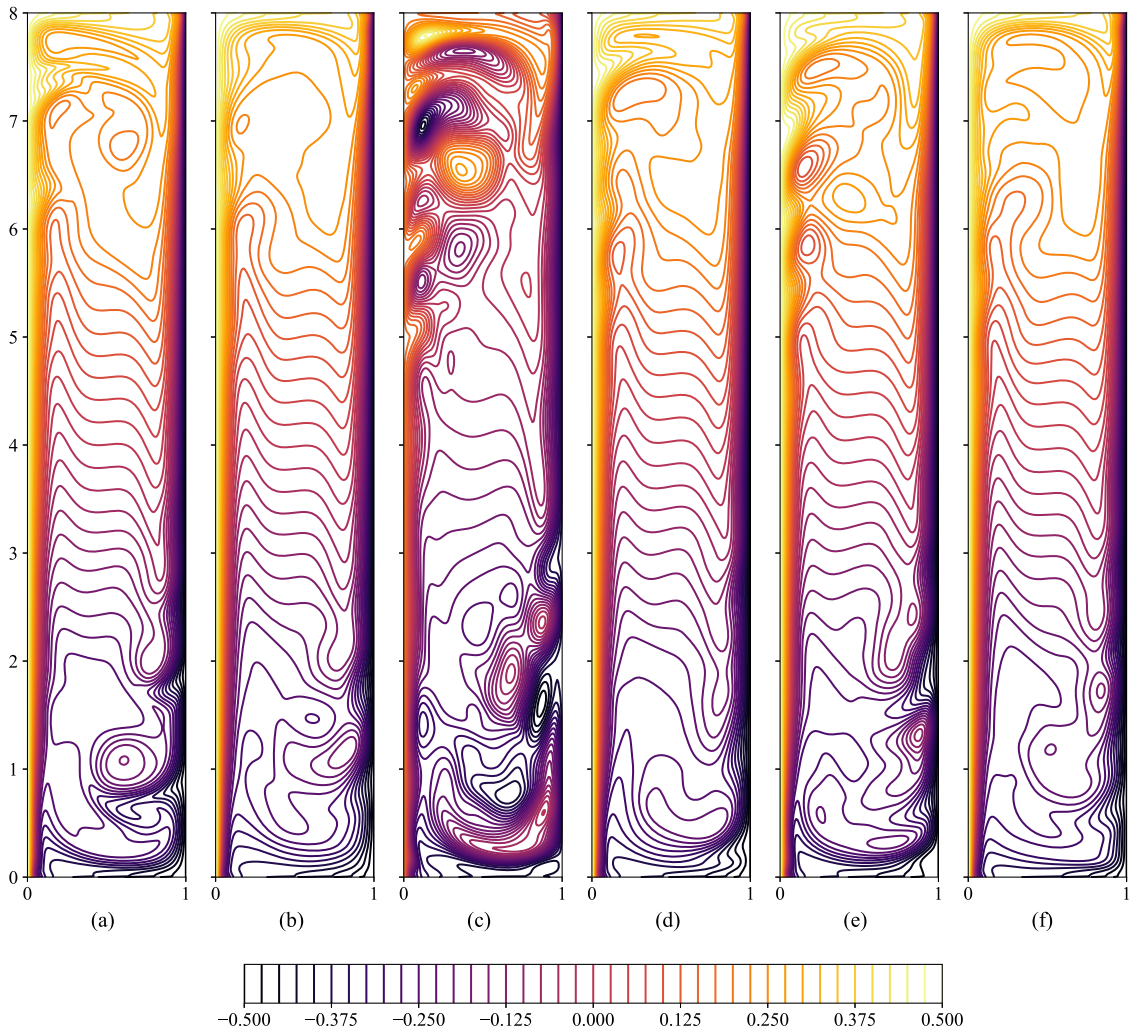

FIG. 19. Contours of instantaneous temperature at Ra = 9.4×10^5. The neural network is trained for different frameworks with $p = 1$. (a) FOM, (b) true projection, (c) ROM-G, (d) DNN-S framework, (e) DNN-R framework, (f) DNN-B framework.

Figure 15 presents results for all DNN frameworks with $p = 1$ in the input training data for Ra = 9.4×10^5. We can see that the DNN prediction is bounded for all DNN frameworks. Although there exist necessary and sufficient conditions for global boundedness[71] for Galerkin systems, we have not performed rigorous analysis about boundedness of the proposed frameworks. A deeper discussion of bounded-input-bounded-output systems can be found elsewhere.[145] We observe that the DNN-R and DNN-B frameworks perform better than the DNN-S framework especially in the in-sample zone (i.e., $t = 0$ to $t = 50$). However, we see that the DNN-R and DNN-B frameworks overpredict both modal coefficients a_1 and a_7 for the out-of-sample zone than the true projection of the FOM solution on reduced order space. Figure 16 shows the similar results for all DNN frameworks with $p = 4$ in the input training data. We notice that the prediction has improved for all DNN frameworks when we use the modal coefficient history in the input training data. All DNN frameworks are almost able to predict the true projection of the FOM solution accurately in the in-sample zone with $p = 4$. The problem of overprediction is also reduced for DNN-R and DNN-B frameworks by including the time history in the input data.

Interestingly, including time history in the input training data seems to have helped neural networks to predict more accurate results for both cases (Ra = 3.4×10^5 and Ra = 9.4×10^5). One explanation for this behavior is that the short-term past history of the system helps the neural network to learn the state of the system and predict the future state more accurately. Our numerical experiments show that increasing the past history of the system might not help beyond some point. In some cases, including long-term history might give adverse results due to overfitting or DNN trying to find an unrelated pattern between the input and the output.

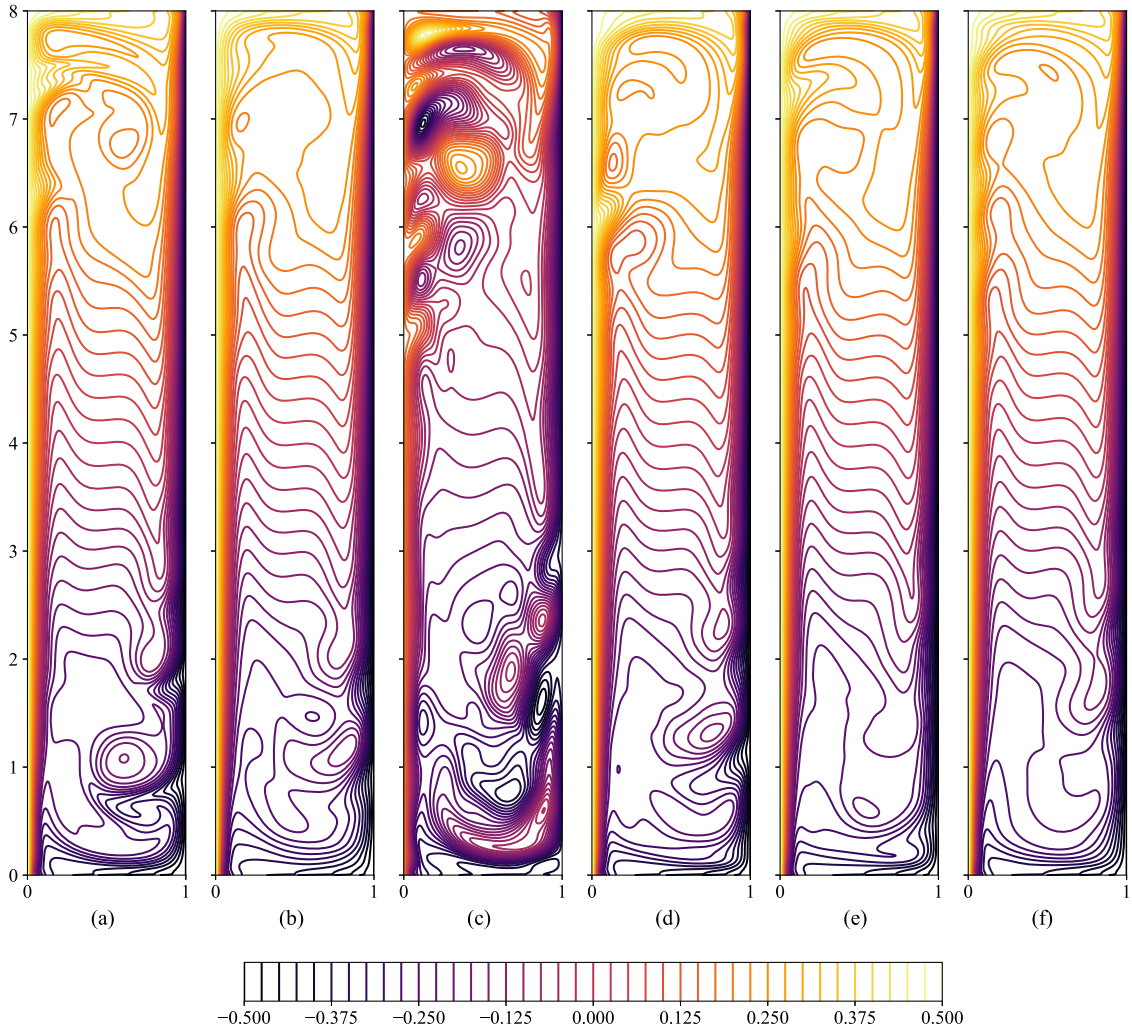

FIG. 20. Contours of instantaneous temperature at Ra = 9.4×10^5. The neural network is trained for different frameworks with $p = 4$. (a) FOM, (b) true projection, (c) ROM-G, (d) DNN-S framework, (e) DNN-R framework, (f) DNN-B framework.

Figures 13–16 show the vorticity modal coefficient only for two modes a_1 and a_7. We use the total RMSE given by Eq. (6) to measure the quantitative performance for all DNN frameworks. The total RMSE measures the deviation between the true projection of the FOM solution and the prediction by the neural network for all modes. In Table III, we report the total RMSE for all DNN frameworks investigated in this study for vorticity and temperature modal coefficients for both Rayleigh numbers Ra = 3.4×10^5 and Ra = 9.4×10^5. Table III also reports the root mean square error for the ROM-G framework. It can be easily seen that all DNN frameworks perform better than the ROM-G framework for both cases. At a higher Rayleigh number, the ROM-G framework is unbounded, and hence, the error is very large. For both cases, we see an improvement in prediction in terms of RMSE for all DNN frameworks as we increase p from 1 to 4.

After comparing the time evolution of vorticity, and temperature modal coefficient for all DNN frameworks, we proceed to compare the performance of proposed DNN frameworks in predicting the temperature field in the cavity at final time $t = 100$. We compare our results with the true projection of the FOM solution on the reduced order space. The FOM solution is obtained by DNS and is discussed briefly in Sec. IV B. For the lower Rayleigh number case, first 10 modes capture more than 99% of the total energy. Hence, the FOM solution and its true projection will be close to each other. However, for the higher Rayleigh number case, first 10 modes capture only 69% of the total energy and we expect to see some discrepancy between the FOM solution and its true projection on reduced order space. We train the neural network using the evolution of modal coefficients for the true projection of the FOM solution, and hence, we can recover at most true projection of the FOM solution.

We compute the instantaneous temperature field using Eq. (38). The coefficient b_k is considered at the final time step $t = 100$. Figure 17 displays the temperature field at the final time predicted by FOM simulation, true projection of FOM on reduced order space, ROM-G framework, and all DNN frameworks. We observe that the FOM solution and its true projection are almost identical. We see that there is some deviation in the temperature field predicted using the ROM-G framework and the true projection. The temperature field predicted using DNN-S and DNN-B frameworks is very close to the true projection solution. We see some variation in the solution predicted by the DNN-R framework, which can be attributed to the phase shift in the modal coefficient predicted by the DNN-R framework with $p = 1$ (as can be seen in Fig. 13). Figure 18 shows similar results for the lower Rayleigh number case with $p = 4$. We notice an improvement in prediction by the DNN-R framework and the predicted temperature field is close to the true projection solution.

Figure 19 illustrates similar results for the temperature field at final time $t = 100$ for the higher Rayleigh number case. The neural network is trained using $p = 1$ in the input training data. We see some of the discrepancies between the FOM solution and its projection on reduced order space. The discrepancy is due to less amount of energy in the first 10 POD modes, and hence, some of the flow features get neglected. The deviation is mainly observed at the bottom and top of the heated cavity due to the vortical structures formed in these regions at a higher Rayleigh number. From Fig. 19, we see that the solution predicted by the ROM-G framework is very different from the true projection for the higher Rayleigh number case. The solution predicted by all DNN frameworks is not identical to the true projection solution. The difference is primarily seen at the bottom and top regions. However, we see overall good qualitative agreement between the temperature field predicted by DNN and true projection solution. Figure 20 shows similar results for the higher Rayleigh number case when the neural network is trained using $p = 4$ in the input training data. We observe a slight improvement in the temperature field prediction. This is consistent with an improvement in the prediction of the modal coefficient (refer to Fig. 16) with an increase in the solution history in the input data for the neural network. The prediction of modal coefficients is very close to the modal coefficients for the true projection of the FOM solution in the in-sample zone ($t = 0$ to $t = 50$). We do not get similar accuracy for the out-of-sample zone ($t = 50$ to $t = 100$). Therefore, we are not able to recover the true projection solution exactly. Despite its limitations, it is clear that our data-driven nonintrusive framework is robust and accurate compared to the intrusive ROM-G framework and can be used for challenging flow problems where the flow is turbulent and not uniform.

Next, we calculate the time-averaged Nusselt number along the left wall ($x = 0$). This quantity can be of interest in many engineering applications. The instantaneous Nusselt number is calculated using Eq. (25). The temperature in Eq. (25) can be obtained using Eq. (38). The gradient of the temperature is computed using a right-sided second-order finite difference scheme. The integration of the temperature gradient is computed using Simpson's integration rule. After calculating the instantaneous temperature at every time

TABLE IV. Statistics of the Nusselt number computed on the left wall ($x = 0$) for Ra = 3.4×10^5 and Ra = 9.4×10^5. The mean Nusselt number and its standard deviation are computed from the instantaneous Nusselt number from time period $t = 0$ to $t = 100$.

Framework	μ	σ
Ra = 3.4×10^5		
FOM	−4.5425	2.46×10^{-3}
True	−4.5494	2.48×10^{-3}
ROM-G	−4.5492	2.55×10^{-3}
DNN-S ($p = 1$)	−4.5494	2.48×10^{-3}
DNN-R ($p = 1$)	−4.5494	2.50×10^{-3}
DNN-B ($p = 1$)	−4.5494	2.49×10^{-3}
DNN-S ($p = 4$)	−4.5494	2.48×10^{-3}
DNN-R ($p = 4$)	−4.5494	2.48×10^{-3}
DNN-B ($p = 4$)	−4.5494	2.48×10^{-3}
Ra = 9.4×10^5		
FOM	−5.8411	4.81×10^{-2}
True	−5.8603	2.29×10^{-2}
ROM-G	−6.2530	2.43×10^{-1}
DNN-S ($p = 1$)	−5.8572	3.45×10^{-2}
DNN-R ($p = 1$)	−5.8643	2.14×10^{-2}
DNN-B ($p = 1$)	−5.8639	2.19×10^{-2}
DNN-S ($p = 4$)	−5.8634	2.87×10^{-2}
DNN-R ($p = 4$)	−5.8637	2.50×10^{-2}
DNN-B ($p = 4$)	−5.8642	2.29×10^{-2}

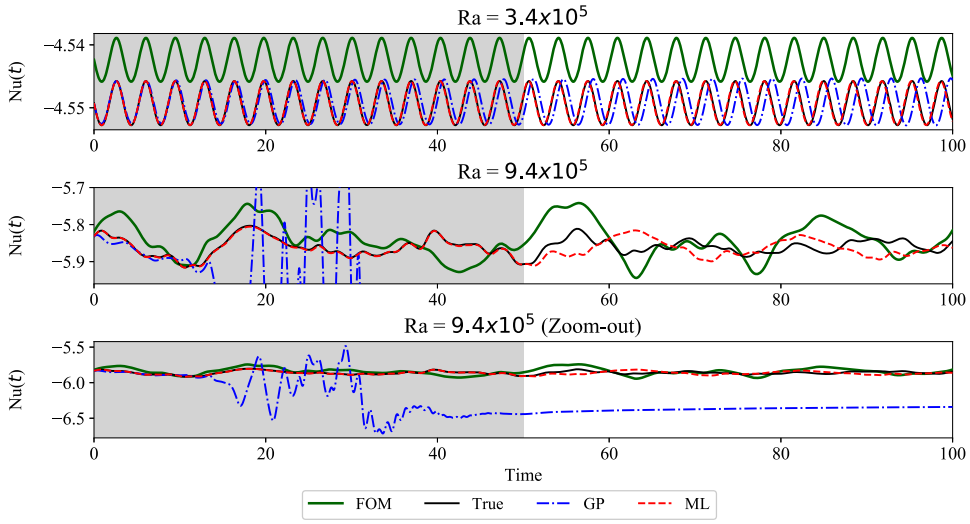

FIG. 21. Evolution of the instantaneous Nusselt number for two different Rayleigh numbers for FOM, true projection, ROM-G, and DNN-R framework with $p = 4$. The neural network is trained using the data highlighted in light gray in the figure. The bottom figure shows the zoom-out plot for the middle figure to show the large range of variation in Nusselt number prediction.

step, we take its average to get the time-averaged Nusselt number. Detailed formulas for calculating the instantaneous Nusselt number are provided in the Appendix.

In Table IV, we list the statistics of the time-averaged Nusselt number for the FOM solution, true projection of the FOM solution, ROM-G framework, and all DNN frameworks investigated in this study. We can notice all DNN frameworks, and the intrusive ROM-G framework gives an accurate prediction of the time-averaged Nusselt number. The standard deviation of Nusselt number is also predicted correctly by all DNN frameworks and the ROM-G framework for the lower Rayleigh number case. We do not recover similar results for the higher Rayleigh number case. The time-averaged Nusselt number for the true projection solution is slightly different from the FOM solution. The ROM-G framework overpredicts the time-averaged Nusselt number, and the difference between the true projection solution and ROM-G prediction is significant. On the other hand, all DNN frameworks predicted the time-averaged Nusselt number close to the true projection solution. The standard deviation of the Nusselt number is also predicted with sufficient accuracy for all DNN frameworks.

To illustrate the temporal variation of Nusselt number, we show the evolution of Nusselt number for FOM, true projection, ROM-G, and DNN-R framework for both Rayleigh numbers in Fig. 21. It can be clearly seen that the ROM-G framework fails to predict the temporal behavior of the Nusselt number correctly especially at $\mathrm{Ra} = 9.4 \times 10^5$. At a lower Rayleigh number, $\mathrm{Ra} = 3.4 \times 10^5$, there is a phase shift in the Nusselt number prediction by the ROM-G framework. At this Rayleigh number, with limit cycle oscillations, the Nusselt number is slightly underpredicted (up to second digit accurate) by the true projection of the FOM solution on reduced order bases. All DNN frameworks predict the Nusselt number accurately close to the true projection results. We would like to again emphasize that the neural network is trained using the true projection of the FOM solution, and hence, we can at most recover the true projection results and not the FOM results. We also note that much simpler methods, like autocorrelation analysis (instead of the heavy DNN), can be used in predicting this stationary time series problem.[114] For a higher Rayleigh number, the Nusselt number prediction for the ROM-G framework becomes unbounded in the beginning and then it calculates the overpredicted value of instantaneous Nusselt number similar to modal coefficients for vorticity and temperature field. The DNN-R framework correctly predicts the instantaneous Nusselt number close to the true projection of the FOM solution in the in-sample zone and sufficiently accurate results for the out-of-sample zone. To avoid redundancy, we do not present results for other DNN-frameworks since we get a similar prediction.

To summarize, we demonstrated the capability of our DNN frameworks within the nonintrusive ROM setup for the differentially heated cavity problem at two Rayleigh numbers. We do a systematic analysis of our DNN frameworks in terms of prediction of the time evolution of modal coefficients, instantaneous temperature field prediction at the final time, and prediction of time-averaged quantities. Our nonintrusive ROM framework gives sufficiently accurate results for simple as well as complex flows.

VI. CONCLUDING REMARKS

In this work, we put forward a nonintrusive reduced order modeling framework which uses a deep neural network to predict the dynamics of ROM for complex flow problems. Deep neural networks are capable of approximating a complex nonlinear relationship between the input and the output, and we achieve this via a supervised learning task. The key difference between the proposed DNN frameworks is the output variable that is learned by the neural network. The nonintrusive ROM is devised using an encoder-decoder approach, and DNN is used for predicting the modal coefficients in an iterative fashion. We use our DNN frameworks with multiple temporal legs (short term history of the state of the system) in the input data. This enables the neural network to take the memory effect into account for predicting the future state of the system. We leverage the classical numerical schemes

(backward-difference) in our proposed DNN-B framework, and this framework can also be implemented with other numerical schemes such as Adams-Bashforth and Adams-Moulton families. First, we use all DNN frameworks for two benchmark problems: Kraichnan-Orszag system and Lorenz system. All DNN frameworks are able to correctly predict each state of the Kraichnan-Orszag system. Even though all DNN frameworks fail to predict correct trajectories for each state of the Lorenz system for a longer duration of time, it predicts the correct dynamics of the Lorenz system in terms of the Lorenz attractor.

After evaluating all DNN frameworks for predicting the dynamics of nonlinear dynamical systems, we proceed to reduce order modeling of the differentially heated cavity problem. We extract 1000 snapshots from DNS simulation after the steady state has been reached. The POD bases are constructed using these 1000 snapshots. Based on the existing literature and our findings from DNS simulation, we see the change from periodic flow to turbulent flow with an increase in Rayleigh number. For this reason, we test our proposed nonintrusive ROM framework for two cases: lower Rayleigh number case (Ra = 3.4×10^5) and higher Rayleigh number case (Ra = 9.4×10^5). We use 10 POD modes for our analysis. Due to the turbulent nature of flow at a higher Rayleigh number, the first 10 modes capture only 69% of the energy. We sacrifice the advantage of ROM if we increase the number of modes (R = 40 for 95% of the energy), and hence, we attempt to get results close to the true projection of the FOM solution on reduced order space with the proposed nonintrusive ROM framework. We assess the performance of the nonintrusive ROM framework using different parameters such as prediction capability of modal coefficients, prediction of instantaneous temperature at the final time, and prediction of engineering quantities of interest such as the time-averaged Nusselt number. The nonintrusive ROM frameworks perform exceptionally well for all these parameters for the lower Rayleigh number case. We see some deviation with the prediction of modal coefficients for the higher Rayleigh number case (especially for the out-of-sample zone). Despite this deviation, the proposed framework is able to predict the instantaneous temperature field and time-averaged Nusselt number with sufficient accuracy.

We also compared our results with the results for Galerkin projection (ROM-G framework). The ROM-G framework gives a good prediction for the low Rayleigh number case. However, the ROM-G framework is unbounded for the higher Rayleigh number case and produces the wrong prediction. Our analysis for the differentially heated cavity problem at two Rayleigh numbers indicates that the nonintrusive ROM setup equipped with the DNN frameworks yields satisfactorily accurate results and has a potential for reduced order modeling of complex flow problems. In this paper, we used snapshot POD to represent the high-dimensional data onto low-dimensional space. At this point, we can ask the following question: is the POD preconditioning for the model identification step necessary? Given the analogy between the POD and the shallow autoencoder, it is possible to use deep neural networks to provide a more compact representation of high-dimensional data.[99] We might arguably assume that a DNN could also learn the potentially much lower dimensional manifold from snaphsot data.[45] Indeed, this is an exciting future direction and the work is in progress on this topic.

ACKNOWLEDGMENTS

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 their support. Direct numerical simulations for this project were performed using resources of the Oklahoma State University High Performance Computing Center. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce Titan Xp GPU for our research.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

APPENDIX: NUSSELT NUMBER CALCULATION

The instantaneous Nusselt number of the flow along the left wall ($x = 0$) is given by

$$\mathrm{Nu}(t) = \frac{1}{H}\int_0^H \left.\frac{\partial\theta}{\partial x}\right|_{x=0} dy, \tag{A1}$$

where H is the height of the cavity.

For the FOM simulation, we have the temperature data available for each discrete point at each time step. The gradient of the temperature is computed using the right-sided finite difference scheme along the left wall ($x = 0$) as given by the following equation:

$$\left.\frac{\partial\theta}{\partial x}\right|_{x=0} = \frac{-3\theta_{0,j} + 4\theta_{1,j} - \theta_{2,j}}{2h}, \tag{A2}$$

where j represents the discrete spatial location in the y-direction and h is the grid spacing in the x-direction. The numerical integration is performed using the fourth-order accurate Simpson's rule. The mean and the standard deviation of the Nusselt number are then computed from the series of instantaneous Nusselt number evaluated at each time step between $t = 0$ and $t = 100$ using the following formulas:

$$\mu = \frac{1}{N}\sum_{n=1}^{N} \mathrm{Nu}(t_n), \tag{A3}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N} (\mathrm{Nu}(t_n) - \mu)^2}, \tag{A4}$$

where N is the total number of time steps between $t = 0$ and $t = 100$.

In the ROM framework, the full order solution can be recovered using the modal coefficient using Eq. (38) and is also given as follows:

$$\theta(\mathbf{x},t) = \tilde{\theta}(\mathbf{x}) + \sum_{k=1}^{R} b_k(t)\phi_k^{\theta}(\mathbf{x}), \tag{A5}$$

where R is the number of modes retained for POD ($R = 10$ in this study).

We can compute the temperature field at each time step using Eq. (A5) and then follow the same procedure similar to the FOM solution. Another faster method to determine the Nusselt number is to calculate the gradient of the mean temperature and each basis function and store it in the memory. The stored gradient can be used to find the instantaneous temperature gradient using the following equation:

$$\left.\frac{\partial\theta}{\partial x}\right|_{x=0} = \left.\frac{\partial\tilde{\theta}}{\partial x}\right|_{x=0} + \sum_{k=1}^{R} b_k(t)\left.\frac{\partial\phi_k^{\theta}}{\partial x}\right|_{x=0}. \tag{A6}$$

If we substitute Eq. (A6) in Eq. (A1), we get

$$\mathrm{Nu}(t) = \kappa + \sum_{k=1}^{R} b_k(t)\gamma_k, \tag{A7}$$

where the predetermined coefficients are

$$\kappa = \frac{1}{H}\int_0^H \left.\frac{\partial\tilde{\theta}}{\partial x}\right|_{x=0} dy, \tag{A8}$$

$$\gamma_k = \frac{1}{H}\int_0^H \left.\frac{\partial\phi_k^{\theta}}{\partial x}\right|_{x=0} dy. \tag{A9}$$

The gradient for the mean temperature and basis function is evaluated using Eq. (A2). The temporal value of temperature modal coefficient is determined using Galerkin projection for the ROM-G framework and using DNN frameworks for the nonintrusive ROM setup. The numerical integration of the instantaneous temperature gradient along the left wall is computed using the composite Simpson's rule with SciPy function integrate.simps for all ROMs. The mean and standard deviation of the Nusselt number are then computed using Eqs. (A3) and (A4), respectively.

REFERENCES

[1] T. Ishihara, T. Gotoh, and Y. Kaneda, "Study of high–Reynolds number isotropic turbulence by direct numerical simulation," Annu. Rev. Fluid Mech. **41**, 165–180 (2009).

[2] G. E. Karniadakis and S. A. Orszag, "Nodes, modes and flow codes," Phys. Today **46**(3), 34–42 (1993).

[3] D. A. Reed and J. Dongarra, "Exascale computing and big data," Commun. ACM **58**, 56–68 (2015).

[4] P. J. Mason, "Large-eddy simulation: A critical review of the technique," Q. J. R. Metereol. Soc. **120**, 1–26 (1994).

[5] U. Piomelli, "Large-eddy simulation: Achievements and challenges," Prog. Aerosp. Sci. **35**, 335–362 (1999).

[6] C. Meneveau and J. Katz, "Scale-invariance and turbulence models for large-eddy simulation," Annu. Rev. Fluid Mech. **32**, 1–32 (2000).

[7] P. Moin, "Advances in large eddy simulation methodology for complex flows," Int. J. Heat Fluid Flow **23**, 710–720 (2002).

[8] D. J. Lucia, P. S. Beran, and W. A. Silva, "Reduced-order modeling: New approaches for computational physics," Prog. Aerosp. Sci. **40**, 51–117 (2004).

[9] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," SIAM Rev. **57**, 483–531 (2015).

[10] S. L. Brunton and B. R. Noack, "Closed-loop turbulence control: Progress and challenges," Appl. Mech. Rev. **67**, 050801 (2015).

[11] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal analysis of fluid flows: An overview," AIAA J. **55**, 4013–4041 (2017).

[12] C. W. Rowley and S. T. Dawson, "Model reduction for flow analysis and control," Annu. Rev. Fluid Mech. **49**, 387–417 (2017).

[13] S. Lakshmivarahan and Y. Wang, "On the relation between energy-conserving low-order models and a system of coupled generalized Volterra gyrostats with nonlinear feedback," J. Nonlinear Sci. **18**, 75–97 (2008).

[14] S. Lakshmivarahan and Y. Wang, "On the structure of the energy conserving low-order models and their relation to Volterra gyrostat," Nonlinear Anal.: Real World Appl. **9**, 1573–1589 (2008).

[15] Y. Wang and S. Lakshmivarahan, "On the relation between energy conserving low-order models and Hamiltonian systems," Nonlinear Anal.: Theory, Methods Appl. **71**, e351–e358 (2009).

[16] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," IEEE Trans. Ind. Inf. **15**, 2405–2415 (2019).

[17] D. Hartmann, M. Herz, and U. Wever, "Model order reduction a key technology for digital twins," in *Reduced-Order Modeling (ROM) for Simulation and Optimization* (Springer, Cham, Switzerland, 2018), pp. 167–179.

[18] J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," J. Guid., Control, Dyn. **8**, 620–627 (1985).

[19] V. Harish and A. Kumar, "Reduced order modeling and parameter identification of a building energy system model through an optimization routine," Appl. Energy **162**, 1010–1023 (2016).

[20] S. W. Fung and L. Ruthotto, "A multiscale method for model order reduction in PDE parameter estimation," J. Comput. Appl. Math. **350**, 19–34 (2019).

[21] K. Ito and S. Ravindran, "A reduced-order method for simulation and control of fluid flows," J. Comput. Phys. **143**, 403–425 (1998).

[22] K. Kunisch and S. Volkwein, "Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition," J. Optim. Theory Appl. **102**, 345–371 (1999).

[23] S. Hovland, J. T. Gravdahl, and K. E. Willcox, "Explicit model predictive control for large-scale systems via model reduction," J. Guid., Control, Dyn. **31**, 918–926 (2008).

[24] M. Bergmann and L. Cordier, "Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models," J. Comput. Phys. **227**, 7813–7840 (2008).

[25] B. R. Noack, M. Morzynski, and G. Tadmor, *Reduced-Order Modelling for Flow Control* (Springer Science & Business Media, New York, 2011).

[26] M. Kärcher, Z. Tokoutsi, M. A. Grepl, and K. Veroy, "Certified reduced basis methods for parametrized elliptic optimal control problems with distributed controls," J. Sci. Comput. **75**, 276–307 (2018).

[27] T. Lassila and G. Rozza, "Parametric free-form shape design with PDE models and reduced basis method," Comput. Methods Appl. Mech. Eng. **199**, 1583–1592 (2010).

[28] P. Benner, E. Sachs, and S. Volkwein, "Model order reduction for PDE constrained optimization," in *Trends in PDE Constrained Optimization* (Springer, Berlin, 2014), pp. 303–326.

[29] M. Heinkenschloss and D. Jando, "Reduced order modeling for time-dependent optimization problems with initial value controls," SIAM J. Sci. Comput. **40**, A22–A51 (2018).

[30] 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).

[31] D. Daescu and I. Navon, "A dual-weighted approach to order reduction in 4DVAR data assimilation," Mon. Weather Rev. **136**, 1026–1041 (2008).

[32] 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).

[33]I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," Philos. Trans. R. Soc., A **374**, 20150202 (2016).
[34]M. Allen, G. Weickum, and K. Maute, "Application of reduced order models for the stochastic design optimization of dynamic systems," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (AIAA, 2004), p. 4614.
[35]M. Frangos, Y. Marzouk, K. Willcox, and B. van Bloemen Waanders, "Surrogate and reduced-order modeling: A comparison of approaches for large-scale statistical inverse problems," in *Large-Scale Inverse Problems and Quantification of Uncertainty* (John Wiley & Sons, 2010), Chap. 7.
[36]J. Degroote, J. Vierendeels, and K. Willcox, "Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis," Int. J. Numer. Methods Fluids **63**, 207–230 (2010).
[37]G. Weickum, M. Eldred, and K. Maute, "Multi-point extended reduced order modeling for design optimization and uncertainty analysis," in *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th* (AIAA, 2006), p. 2145.
[38]J. Shenefelt, R. Luck, R. Taylor, and J. Berry, "Solution to inverse heat conduction problems employing singular value decomposition and model-reduction," Int. J. Heat Mass Transfer **45**, 67–74 (2002).
[39]C. Lieberman, K. Willcox, and O. Ghattas, "Parameter and state model reduction for large-scale statistical inverse problems," SIAM J. Sci. Comput. **32**, 2523–2542 (2010).
[40]R. Everson and L. Sirovich, "Karhunen–Loeve procedure for gappy data," J. Opt. Soc. Am. A **12**, 1657–1664 (1995).
[41]S. Chaturantabut and D. C. Sorensen, "Nonlinear model reduction via discrete empirical interpolation," SIAM J. Sci. Comput. **32**, 2737–2764 (2010).
[42]B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele, "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake," J. Fluid Mech. **497**, 335–363 (2003).
[43]K. Carlberg, C. Bou-Mosleh, and C. Farhat, "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," Int. J. Numer. Methods Eng. **86**, 155–181 (2011).
[44]A. Alla and J. N. Kutz, "Nonlinear model order reduction via dynamic mode decomposition," SIAM J. Sci. Comput. **39**, B778–B796 (2017).
[45]J.-C. Loiseau, B. R. Noack, and S. L. Brunton, "Sparse reduced-order modelling: Sensor-based dynamics to full-state estimation," J. Fluid Mech. **844**, 459–490 (2018).
[46]G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," Annu. Rev. Fluid Mech. **25**, 539–575 (1993).
[47]A. C. Antoulas, D. C. Sorensen, and S. Gugercin, "A survey of model reduction methods for large-scale systems," Contemp. Math. **280**, 193–220 (2000).
[48]J. L. Lumley, "The structure of inhomogeneous turbulent flows," in *Atmospheric Turbulence and Radio Wave Propagation* (Nauka, Moscow, 1967), pp. 166–176.
[49]T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders, "Goal-oriented, model-constrained optimization for reduction of large-scale systems," J. Comput. Phys. **224**, 880–896 (2007).
[50]A. Hay, J. T. Borggaard, and D. Pelletier, "Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition," J. Fluid Mech. **629**, 41–72 (2009).
[51]K. Kunisch and S. Volkwein, "Optimal snapshot location for computing POD basis functions," ESAIM: Math. Modell. Numer. Anal. **44**, 509–529 (2010).
[52]K. Carlberg and C. Farhat, "A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems," Int. J. Numer. Methods Eng. **86**, 381–402 (2011).
[53]N. Aubry, R. Guyonnet, and R. Lima, "Spatiotemporal analysis of complex signals: Theory and applications," J. Stat. Phys. **64**, 683–739 (1991).
[54]M. Sieber, C. O. Paschereit, and K. Oberleithner, "Spectral proper orthogonal decomposition," J. Fluid Mech. **792**, 798–828 (2016).
[55]A. Towne, O. T. Schmidt, and T. Colonius, "Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis," J. Fluid Mech. **847**, 821–867 (2018).
[56]M. Mendez, M. Balabane, and J.-M. Buchlin, "Multi-scale proper orthogonal decomposition of complex fluid flows," J. Fluid Mech. **870**, 988–1036 (2019).
[57]N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, "The dynamics of coherent structures in the wall region of a turbulent boundary layer," J. Fluid Mech. **192**, 115–173 (1988).
[58]C. W. Rowley, T. Colonius, and R. M. Murray, "Model reduction for compressible flows using POD and Galerkin projection," Physica D **189**, 115–129 (2004).
[59]M. F. Barone, I. Kalashnikova, D. J. Segalman, and H. K. Thornquist, "Stable Galerkin reduced order models for linearized compressible flow," J. Comput. Phys. **228**, 1932–1946 (2009).
[60]I. Akhtar, A. H. Nayfeh, and C. J. Ribbens, "On the stability and extension of reduced-order Galerkin models in incompressible flows," Theor. Comput. Fluid Dyn. **23**, 213–237 (2009).
[61]V. L. Kalb and A. E. Deane, "An intrinsic stabilization scheme for proper orthogonal decomposition based low-dimensional models," Phys. Fluids **19**, 054106 (2007).
[62]M. Bergmann, C.-H. Bruneau, and A. Iollo, "Enablers for robust POD models," J. Comput. Phys. **228**, 516–538 (2009).
[63]S. S. Ravindran, "A reduced-order approach for optimal control of fluids using proper orthogonal decomposition," Int. J. Numer. Methods Fluids **34**, 425–448 (2000).
[64]D. Amsallem, J. Cortial, K. Carlberg, and C. Farhat, "A method for interpolating on manifolds structural dynamics reduced-order models," Int. J. Numer. Methods Eng. **80**, 1241–1258 (2009).
[65]O. San and J. Borggaard, "Principal interval decomposition framework for POD reduced-order modeling of convective Boussinesq flows," Int. J. Numer. Methods Fluids **78**, 37–62 (2015).
[66]J. Borggaard, T. Iliescu, and Z. Wang, "Artificial viscosity proper orthogonal decomposition," Math. Comput. Modell. **53**, 269–279 (2011).
[67]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**, 10–26 (2012).
[68]O. San and T. Iliescu, "Proper orthogonal decomposition closure models for fluid flows: Burgers equation," Int. J. Numer. Anal. Modell., Ser. B **5**, 217–237 (2014).
[69]J. Östh, B. R. Noack, S. Krajnović, D. Barros, and J. Borée, "On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body," J. Fluid Mech. **747**, 518–544 (2014).
[70]L. Cordier, B. R. Noack, G. Tissot, G. Lehnasch, J. Delville, M. Balajewicz, G. Daviller, and R. K. Niven, "Identification strategies for model-based control," Exp. Fluids **54**, 1580 (2013).
[71]M. Schlegel and B. R. Noack, "On long-term boundedness of Galerkin models," J. Fluid Mech. **765**, 325–352 (2015).
[72]T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza, "Model order reduction in fluid dynamics: Challenges and perspectives," in *Reduced Order Methods for Modeling and Computational Reduction* (Springer, New York, 2014), pp. 235–273.
[73]B. Peherstorfer and K. Willcox, "Data-driven operator inference for nonintrusive projection-based model reduction," Comput. Methods Appl. Mech. Eng. **306**, 196–215 (2016).
[74]M. S. Siddiqui, E. Fonn, T. Kvamsdal, and A. Rasheed, "Finite-volume high-fidelity simulation combined with finite-element-based reduced-order modeling of incompressible flow problems," Energies **12**, 1271 (2019).
[75]J. Reiss, P. Schulze, J. Sesterhenn, and V. Mehrmann, "The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena," SIAM J. Sci. Comput. **40**, A1322–A1344 (2018).
[76]A. Ehlert, C. N. Nayeri, M. Morzynski, and B. R. Noack, "Locally linear embedding for transient cylinder wakes," preprint arXiv:1906.07822 (2019).
[77]B. R. Noack, "From snapshots to modal expansions–bridging low residuals and pure frequencies," J. Fluid Mech. **802**, 1–4 (2016).
[78]J. Bourgeois, B. Noack, and R. Martinuzzi, "Generalized phase average with applications to sensor-based flow estimation of the wall-mounted square cylinder wake," J. Fluid Mech. **736**, 316–350 (2013).
[79]D. Rempfer, "On low-dimensional Galerkin models for fluid flow," Theor. Comput. Fluid Dyn. **14**, 75–88 (2000).

[80]C. Audouze, F. De Vuyst, and P. B. Nair, "Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations," Numer. Methods Partial Differ. Equations **29**, 1587–1628 (2013).
[81]M. P. Mignolet, A. Przekop, S. A. Rizzi, and S. M. Spottswood, "A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures," J. Sound Vib. **332**, 2437–2460 (2013).
[82]D. Xiao, F. Fang, C. Pain, and G. Hu, "Non-intrusive reduced-order modelling of the Navier–Stokes equations based on RBF interpolation," Int. J. Numer. Methods Fluids **79**, 580–595 (2015).
[83]J. S. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks," J. Comput. Phys. **363**, 55–78 (2018).
[84]J. Hampton, H. R. Fairbanks, A. Narayan, and A. Doostan, "Practical error bounds for a non-intrusive bi-fidelity approach to parametric/stochastic model reduction," J. Comput. Phys. **368**, 315–332 (2018).
[85]W. Chen, J. S. Hesthaven, B. Junqiang, Y. Qiu, Z. Yang, and Y. Tihao, "Greedy nonintrusive reduced order model for fluid dynamics," AIAA J. **56**, 4927–4943 (2018).
[86]D. Xiao, C. Heaney, F. Fang, L. Mottet, R. Hu, D. Bistrian, E. Aristodemou, I. Navon, and C. Pain, "A domain decomposition non-intrusive reduced order model for turbulent flows," Comput. Fluids **182**, 15–27 (2019).
[87]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).
[88]C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," J. Fluid Mech. **641**, 115–127 (2009).
[89]P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," J. Fluid Mech. **656**, 5–28 (2010).
[90]M. R. Jovanović, P. J. Schmid, and J. W. Nichols, "Sparsity-promoting dynamic mode decomposition," Phys. Fluids **26**, 024103 (2014).
[91]M. S. Hemati, M. O. Williams, and C. W. Rowley, "Dynamic mode decomposition for large and streaming datasets," Phys. Fluids **26**, 111701 (2014).
[92]J. L. Proctor and P. A. Eckhoff, "Discovering dynamic patterns from infectious disease data using dynamic mode decomposition," Int. Health **7**, 139–145 (2015).
[93]B. R. Noack, W. Stankiewicz, M. Morzyński, and P. J. Schmid, "Recursive dynamic mode decomposition of transient and post-transient wake flows," J. Fluid Mech. **809**, 843–872 (2016).
[94]D. A. Bistrian and I. M. Navon, "An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs POD," Int. J. Numer. Methods Fluids **78**, 552–580 (2015).
[95]A. Alekseev, D. Bistrian, A. Bondarev, and I. Navon, "On linear and nonlinear aspects of dynamic mode decomposition," Int. J. Numer. Methods Fluids **82**, 348–371 (2016).
[96]D. A. Bistrian and I. M. Navon, "Randomized dynamic mode decomposition for nonintrusive reduced order modelling," Int. J. Numer. Methods Eng. **112**, 3–25 (2017).
[97]G. Pascarella, M. Fossati, and G. Barrenechea, "Adaptive reduced basis method for the reconstruction of unsteady vortex-dominated flows," Comput. Fluids **190**, 382–397 (2019).
[98]S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, New York, 2019).
[99]S. Brunton, B. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," preprint arXiv:1905.11075 (2019).
[100]O. San and R. Maulik, "Extreme learning machine for reduced order modeling of turbulent geophysical flows," Phys. Rev. E **97**, 042322 (2018).
[101]X. Xie, G. Zhang, and C. G. Webster, "Data driven reduced order modeling of fluid dynamics using linear multistep network," preprint arXiv:1809.07820 (2018).
[102]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," preprint arXiv:1801.01236 (2018).
[103]T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Montréal, Canada, 2018), pp. 6571–6583.
[104]S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, "Data-driven identification of parametric partial differential equations," SIAM J. Appl. Dyn. Syst. **18**, 643–660 (2019).
[105]J. N. Kani and A. H. Elsheikh, "DR-RNN: A deep residual recurrent neural network for model reduction," preprint arXiv:1709.00939 (2017).
[106]K. Yeo and I. Melnyk, "Deep learning algorithm for data-driven simulation of noisy dynamical system," J. Comput. Phys. **376**, 1212–1231 (2019).
[107]T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," J. Comput. Phys. **395**, 620 (2019).
[108]S. Rahman, A. Rasheed, and O. San, "A hybrid analytics paradigm combining physics-based modeling and data-driven modeling to accelerate incompressible flow solvers," Fluids **3**, 50 (2018).
[109]O. San and R. Maulik, "Neural network closures for nonlinear model order reduction," Adv. Comput. Math. **44**, 1717–1750 (2018).
[110]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).
[111]A. T. Mohan and D. V. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks," preprint arXiv:1804.09269 (2018).
[112]J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos **28**, 041101 (2018).
[113]Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *International Conference on Machine Learning* (Proceedings of Machine Learning Research, Stockholm, Sweden, 2018), pp. 3282–3291.
[114]R. H. Shumway and D. S. Stoffer, *Time Series Analysis and its Applications: With R Examples* (Springer, New York, 2017).
[115]J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," J. Fluid Mech. **807**, 155–166 (2016).
[116]M. Milano and P. Koumoutsakos, "Neural network modeling for near wall turbulent flow," J. Comput. Phys. **182**, 1–26 (2002).
[117]M. Gamahara and Y. Hattori, "Searching for turbulence models by artificial neural network," Phys. Rev. Fluids **2**, 054604 (2017).
[118]M. Raissi, P. Perdikaris, and G. 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).
[119]C. Michoski, M. Milosavljevic, T. Oliver, and D. Hatch, "Solving irregular and data-enriched differential equations using deep neural networks," preprint arXiv:1905.04351 (2019).
[120]C. Zerfas, L. G. Rebholz, M. Schneier, and T. Iliescu, "Continuous data assimilation reduced order models of fluid flow," preprint arXiv:1903.04029 (2019).
[121]M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design* (PWS Publishing, Boston, 1996), Vol. 20.
[122]B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," Proc. IEEE **104**, 148–175 (2015).
[123]E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," Inverse Probl. **34**, 014004 (2017).
[124]B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[125]O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," Commun. Nonlinear Sci. Numer. Simul. **77**, 271–287 (2019).
[126]X. Wan and G. E. Karniadakis, "An adaptive multi-element generalized polynomial chaos method for stochastic differential equations," J. Comput. Phys. **209**, 617–642 (2005).
[127]E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[128]P. Berg, Y. Pomeau, and C. Vidal, *Order within Chaos: Towards a Deterministic Approach to Turbulence* (Wiley, New York, 1986).

[129]D. Poland, "Cooperative catalysis and chemical chaos: A chemical model for the Lorenz equations," Physica D **65**, 86–99 (1993).
[130]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**, 3932–3937 (2016).
[131]S. Paolucci, "The differentially heated cavity," Sadhana **19**, 619–647 (1994).
[132]B. Podvin and P. Le Quéré, "Low-order models for the flow in a differentially heated cavity," Phys. Fluids **13**, 3204 (2001).
[133]H. Johnston and R. Krasny, "Fourth-order finite difference simulation of a differentially heated cavity," Int. J. Numer. Methods Fluids **40**, 1031–1037 (2002).
[134]J.-G. Liu, C. Wang, and H. Johnston, "A fourth order scheme for incompressible Boussinesq equations," J. Sci. Comput. **18**, 253–285 (2003).
[135]A. E. Deane and L. Sirovich, "A computational study of Rayleigh-Bénard convection. Part 1. Rayleigh-number scaling," J. Fluid Mech. **222**, 231–250 (1991).
[136]L. Sirovich and A. E. Deane, "A computational study of Rayleigh–Bénard convection. Part 2. Dimension considerations," J. Fluid Mech. **222**, 251–265 (1991).
[137]O. San and R. Maulik, "Machine learning closures for model order reduction of thermal fluids," Appl. Math. Modell. **60**, 681–710 (2018).
[138]R. Maulik and O. San, "A dynamic subgrid-scale modeling framework for Boussinesq turbulence," Int. J. Heat Mass Transfer **108**, 1656–1675 (2017).
[139]A. Tyliszczak, "High-order compact difference algorithm on half-staggered meshes for low Mach number flows," Comput. Fluids **127**, 131–145 (2016).
[140]A. Iollo, S. Lanteri, and J.-A. Désidéri, "Stability properties of POD–Galerkin approximations for the compressible Navier–Stokes equations," Theor. Comput. Fluid Dyn. **13**, 377–396 (2000).
[141]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**, 887–915 (2012).
[142]L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent structures," Q. Appl. Math. **45**, 561–571 (1987).
[143]J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Montréal, Canada, 2012), pp. 2951–2959.
[144]G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing **70**, 489–501 (2006).
[145]Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," IEEE Trans. Neural Networks **18**, 359–372 (2007).