Physics of Fluids

Model fusion with physics-guided machine learning: Projection-based reduced-order modeling

Cite as: Phys. Fluids **33**, 067123 (2021); https://doi.org/10.1063/5.0053349
Submitted: 06 April 2021 • Accepted: 27 May 2021 • Published Online: 29 June 2021

Suraj Pawar, Omer San, Aditya Nair, et al.

Physics of Fluids ARTICLE scitation.org/journal/phf

Model fusion with physics-guided machine learning: Projection-based reduced-order modeling

Cite as: Phys. Fluids 33, 067123 (2021); doi: 10.1063/5.0053349
Submitted: 6 April 2021 · Accepted: 27 May 2021 ·
Published Online: 29 June 2021

View Online Export Citation CrossMark

Suraj Pawar,[1] Omer San,[1,a)] Aditya Nair,[2] Adil Rasheed,[3,4] and Trond Kvamsdal[4,5]

AFFILIATIONS

[1]School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma 74078, USA
[2]Department of Mechanical Engineering, University of Nevada, Reno, Nevada 89557, USA
[3]Department of Engineering Cybernetics, Norwegian University of Science and Technology, N-7465 Trondheim, Norway
[4]Department of Mathematics and Cybernetics, SINTEF Digital, 7034 Trondheim, Norway
[5]Department of Mathematical Sciences, Norwegian University of Science and Technology, 7491 Trondheim, Norway

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

ABSTRACT

The unprecedented amount of data generated from experiments, field observations, and large-scale numerical simulations at a wide range of spatiotemporal scales has enabled the rapid advancement of data-driven and especially deep learning models in the field of fluid mechanics. Although these methods are proven successful for many applications, there is a grand challenge of improving their *generalizability*. This is particularly essential when data-driven models are employed within outer-loop applications like optimization. In this work, we put forth a physics-guided machine learning (PGML) framework that leverages the interpretable physics-based model with a deep learning model. Leveraging a concatenated neural network design from multi-modal data sources, the PGML framework is capable of enhancing the generalizability of data-driven models and effectively protects against or inform about the inaccurate predictions resulting from extrapolation. We apply the PGML framework as a novel model fusion approach combining the physics-based Galerkin projection model and long- to short-term memory (LSTM) network for parametric model order reduction of fluid flows. We demonstrate the improved generalizability of the PGML framework against a purely data-driven approach through the injection of physics features into intermediate LSTM layers. Our quantitative analysis shows that the overall model uncertainty can be reduced through the PGML approach, especially for test data coming from a distribution different than the training data. Moreover, we demonstrate that our approach can be used as an inverse diagnostic tool providing a confidence score associated with models and observations. The proposed framework also allows for multi-fidelity computing by making use of low-fidelity models in the online deployment of quantified data-driven models.

I. INTRODUCTION

Data-driven approaches are emerging as the new paradigm in computational modeling of various problems in different branches of science and engineering, including fluid dynamics.[1,2] The universal approximation capability of neural networks[3] makes them the powerful algorithm for complicated problems like turbulence closure modeling,[4] spatiotemporal super-resolution,[5] state-estimation,[6] and nonlinear model order reduction.[7] One of the key issues with deep learning is that they exhibit poor *generalizability*; that is, they produce inaccurate prediction when the test data are from a distribution far from that of the training data. This adversely affects the trustworthiness of neural networks for scientific applications and embedding the known physics to enhance the generalizability is a challenge and opportunity for data-driven algorithm developments.[8–11]

While many techniques have sought to enforce physics into data-driven models, such as regularizing the neural network with the governing equations or statistical constraints,[12–14] imposing conservation of physical quantities directly into neural network,[15,16] embedding invariance property into neural network architecture,[17,18] and incorporating inductive biases to respect exact conservation laws,[19] they offer many possibilities to fuse domain knowledge to improve the generalizability and explainability of these models.[11] In this work, we propose the physics-guided machine learning (PGML) framework that can answer the question of how to hybridize data-driven (i.e., non-intrusive) and

first-principles (i.e., intrusive) approaches that are often utilized in fluid dynamics applications. The PGML framework is based on a composite modeling strategy where the physics-based model is integrated within the neural network architecture leading to a more generalizable learning engine.

The high-dimensional and multiscale nature of fluid flows makes the numerical discretization methods computationally infeasible for many practical applications. There are several techniques that aim at constructing the lower-order representation of high-dimensional systems that can capture the essential features and are computationally orders of magnitude faster than full order model.[20–22] These reduced-order models (ROMs) are particularly appealing for outer-loop applications, such as data assimilation,[23–25] uncertainty quantification,[26] and model predictive control[27,28] that require multiple evaluations of a model for multiple inputs. Proper orthogonal decomposition (POD)[29] and dynamic mode decomposition[30] enabled ROMs are some of the most widely used methods among the fluid dynamics community. POD is often combined with Galerkin projection (GP) to model the dynamics of the ROM.[31,32] One of the limitations of intrusive approaches like GP is that it requires the complete knowledge about the equations governing the system's dynamics. However, for many physical systems, the exact governing equations are not available due to imperfect knowledge about the system. For example, in many geophysical flows we might not have accurate models for processes such as wind forcing, bottom friction, stratification, and other parameterization.[33]

Lately, equation-free or non-intrusive reduced-order modeling (NIROM) has drawn great attention in the fluid mechanics community due to its flexibility and efficiency for systems with incomplete dynamics and is undergoing rapid development with various algorithms emerging from different perspectives.[34] In most of the NIROMs, the main idea is to employ deep learning methods to construct nonlinear manifold[7,35,36] and to model the temporal dynamics of ROMs.[37–40] Despite the success of NIROMs for many complex nonlinear problems, the naive deployment of NIROMs in multi-query applications is limited because they lack connection with the physics-based model and might suffer from poor performance in extrapolation regime.[41] Therefore, the hybrid modeling approach that can overcome these drawbacks is warranted, and to this end, we apply the PGML framework that exploits the Galerkin projection model for the known physics and long- to short term memory (LSTM) neural network to model the unknown physics. We remark here that the ideas from other approaches like physics-recurrent neural network (RNN)[42] and physics-embedded convolutional autoencoder (PhyCAE)[16] can be easily integrated within the proposed PGML framework. Furthermore, improving model predictions with composite models involving the composition (e.g., addition) of two models has been shown to be effective, for example, in the field of computer experiments using the nonstationary Gaussian processes.[43–46] Such composite models use one model to learn the global trends in the data and another to learn the local volatility, thereby together enhancing predictions. In this case, the second model is trained to predict the residual of the first model. We also refer readers to an interesting discussion on improving predictions of such computer experiments using a small-scale measurement error term called nugget,[47] where the authors highlighted that the nugget term can lead to enhanced surrogate models with better statistical properties.

In this study, we consider a dynamical system whose evolution is governed by partial differential equations (PDEs) such as the Navier–Stokes equations as follows:

$$\dot{\boldsymbol{u}} = \boldsymbol{f}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}), \tag{1}$$

where $\boldsymbol{u}$ is the prognostic variable that depends on a set of parameters, $\boldsymbol{f}$ is the nonlinear function that fully represents the physical processes and conservation laws, and $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ is the parameterization such as Reynolds or Rayleigh numbers, and N_p refers to the number of parameters. Then, we can split $\boldsymbol{f}$ using a notion of known-physics (i.e., $\hat{\boldsymbol{f}}$) and unknown-physics (i.e., $\boldsymbol{\pi}$) parts. More precisely, $\hat{\boldsymbol{f}}$ refers to the model's dynamical core (i.e., a semi-discretized function of PDEs of mass, momentum, energy conservation laws). However, there could be unknown physics that might not be covered within the physics-based dynamical core model $\hat{\boldsymbol{f}}$. For example, $\boldsymbol{\pi}$ might refer to the physical and chemical processes that are either unknown or too complex to be modeled (e.g., precipitation, long and short wave atmospheric radiation, clouds, constituency transport, and chemical reactions, or any forcing function, i.e., not modeled in $\hat{\boldsymbol{f}}$). The augmented representation can then be written at an abstract level as

$$\dot{\boldsymbol{u}} = \hat{\boldsymbol{f}}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}) + \boldsymbol{\pi}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}), \tag{2}$$

where $\boldsymbol{\pi}$ encompasses all the unknown processes and parameterizations. To differentiate the solutions between the full and partial physics, let's assume that we have a physics-based (but incomplete) model

$$\dot{\boldsymbol{u}} = \hat{\boldsymbol{f}}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}). \tag{3}$$

Here, we can define $\delta\boldsymbol{u}$ as a natural way to quantify the unknown physics $\boldsymbol{\pi}$ as follows:

$$\delta\boldsymbol{u} = \sqrt{\frac{||\boldsymbol{u}_1 - \boldsymbol{u}_2||^2}{||\boldsymbol{u}_2||^2}}, \tag{4}$$

where $\boldsymbol{u}_1$ is the solution of Eq. (3) (i.e., without $\boldsymbol{\pi}$), $\boldsymbol{u}_2$ is the solution of Eq. (2) with $\boldsymbol{\pi}$, and $||\cdot||$ is the L_2 norm. One can easily quantify the level of unknown processes using the definition given by Eq. (4), that is, $\delta\boldsymbol{u} \to 0$ when $\boldsymbol{\pi} \to 0$. On the other hand, a higher value of $\delta\boldsymbol{u}$ means that our known physical model $\hat{\boldsymbol{f}}$ is poor.

The chief motivation behind the PGML framework is to exploit domain knowledge to produce generalizable data-driven models. We demonstrate the application of the PGML framework to fuse intrusive and non-intrusive parametric projection-based ROMs for incomplete dynamical systems that can be represented using Eq. (3). Apart from improving the extrapolation performance of data-driven models, the PGML framework can also be considered as an enabler for multi-fidelity computing through a synergistic combination of low-cost/low-fidelity models with the high-fidelity data. The field of multi-fidelity computing is gaining attention in computational science and engineering, as it provides a way to utilize computationally cheap approximate models with high-fidelity observations.[48,49] This is particularly relevant to fluid mechanics applications where there is a wealth of simplified models like potential flow theory, Blasius boundary layer model, and continuum Darcy/pipe flow model. One of the major challenges in multi-fidelity computing is determining the correlation between low and high fidelity models, and the neural networks employed within the PGML framework are capable of discovering this nonlinear relation. This first step in the assessment of the new PGML model shows that the proposed hybrid reduced-order modeling framework could represent a viable tool for emerging digital twin applications,[10] where low-fidelity models are often employed for computational efficiency.

Nonetheless, with available streams of observational data from the real physical asset, PGML can enhance the fidelity of the whole system.

The paper is structured as follows. We introduce the parametric ROM framework in Sec. II. We then discuss the specific formulation of the PGML framework in Sec. III for combining the domain knowledge with data. The numerical results for the two-dimensional vorticity transport equation are detailed in Sec. IV. Finally, Sec. V will present conclusions and ideas for the future work.

II. PARAMETRIC ROM FRAMEWORK

The first step in parametric ROMs is the dimensionality reduction. We choose proper orthogonal decomposition (POD) to compute the linear orthogonal basis functions. POD is one of the most popular technique for dimensionality reduction in the fluid mechanics community.[50,51] POD provides the linear basis functions that minimize the error between the true data and its projection in the L_2 sense compared to all linear basis functions of the same dimension. Given the N_s snapshots of the data for a variable of interest in the physical system, we can form the matrix $\boldsymbol{A}$ as follows:

$$\boldsymbol{A} = \left[\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(N_s)}\right] \in \mathbb{R}^{N \times N_s}, \tag{5}$$

where $\boldsymbol{u}^{(k)} \in \mathbb{R}^N$ corresponds to an individual snapshot of the solution in time. Then, we perform the singular value decomposition (SVD) as follows:

$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{k=1}^{N_s} \sigma_k \boldsymbol{w}_k \boldsymbol{v}_k^T, \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^{N \times N_s}$ is a matrix with orthonormal columns, which represent the left-singular vectors, $\boldsymbol{V} \in \mathbb{R}^{N_s \times N_s}$ is matrix with orthonormal columns representing the right-singular vectors, and $\boldsymbol{\Sigma} \in \mathbb{R}^{N_s \times N_s}$ is a matrix with non-negative diagonal entries, called singular values, arranged such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{N_s} \geq 0$. In dimensionality reduction, only the first N_r columns of $\boldsymbol{W}$ and $\boldsymbol{V}$ are retained along with the upper-left $N_r \times N_r$ sub-matrix of $\boldsymbol{\Sigma}$. The singular values σ_i give a measure of the quality of information, that is, retailed in N_r-order approximation of the matrix $\boldsymbol{A}$. The N_r columns of $\boldsymbol{W}$ are referred to as the POD modes or basis functions, denoted as $\boldsymbol{\Phi}_{\boldsymbol{\mu}} = \{\phi_k\}_{k=1}^{N_r}$. The amount of the energy retained by POD modes can be computed using the quantity called the relative information content (RIC) as follows:

$$\mathrm{RIC}(N_r) = \frac{\sum_{j=1}^{N_r} \sigma_j^2}{\sum_{j=1}^{N_s} \sigma_j^2}. \tag{7}$$

We utilize the Grassmann manifold interpolation[52,53] to compute the basis functions for a new set of parameters from available POD basis function computed in the offline phase. In the offline phase, we calculate a set of POD basis functions $\{\boldsymbol{\Phi}_i\}_{i=1}^{N_c}$ corresponding to a set of parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{N_c}$, where N_c is the number of parameterized datasets (e.g., $N_c = 4$ in this study since we utilize data snapshots from $\mathrm{Re} = \{200, 400, 600, 800\}$ in our training). The Grassmann manifold interpolation consists of choosing a reference point S_0 corresponding to the basis set $\boldsymbol{\Phi}_0$ and then mapping each point S_i to a matrix $\boldsymbol{\Gamma}_i$, which represents a point on the tangent space at S_0 using logarithmic map Log_{S_0} as follows:

$$(\boldsymbol{\Phi}_i - \boldsymbol{\Phi}_0\boldsymbol{\Phi}_0^T\boldsymbol{\Phi}_i)(\boldsymbol{\Phi}_0^T\boldsymbol{\Phi}_i)^{-1} = \boldsymbol{W}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T, \tag{8}$$

$$\boldsymbol{\Gamma}_i = \boldsymbol{W}_i \tan^{-1}(\boldsymbol{\Sigma}_i)\boldsymbol{V}_i^T. \tag{9}$$

The matrix $\boldsymbol{\Gamma}_t$ corresponding to target operating point $\boldsymbol{\mu}_t$ is computed by interpolating the corresponding entries of matrices $\{\boldsymbol{\Gamma}_i\}_{i=1}^{N_c}$. When $N_p = 1$ (e.g., $\boldsymbol{\mu} \in \{\mathrm{Re}\}$ in this work), typically Lagrange-type interpolation method is applied as follows:

$$\boldsymbol{\Gamma}_t = \sum_{i=1}^{N_c} \left(\prod_{\substack{j=1 \\ j \neq i}}^{N_c} \frac{\mu_t - \mu_j}{\mu_i - \mu_j} \right) \boldsymbol{\Gamma}_i. \tag{10}$$

The POD basis function $\boldsymbol{\Phi}_t$ corresponding to the test parameter $\boldsymbol{\mu}_t$ is computed using the exponential mapping as follows:

$$\boldsymbol{\Gamma}_t = \boldsymbol{W}_t\boldsymbol{\Sigma}_t\boldsymbol{V}_t^T, \tag{11}$$

$$\boldsymbol{\Phi}_t = [\boldsymbol{\Phi}_0\boldsymbol{V}_t \cos(\boldsymbol{\Sigma}_t) + \boldsymbol{W}_t \sin(\boldsymbol{\Sigma}_t)]\boldsymbol{V}_t^T, \tag{12}$$

where the trigonometric operators apply only to diagonal elements.

Once the basis functions corresponding to the test operating point are computed, the reduced-order representation of the high-dimensional state $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu})$ is as follows:

$$\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \boldsymbol{u}_r(\boldsymbol{x}, t; \boldsymbol{\mu}) = \boldsymbol{\Phi}_{\boldsymbol{\mu}}\boldsymbol{a}(t; \boldsymbol{\mu}), \tag{13}$$

where $\boldsymbol{a}(t; \boldsymbol{\mu}) \in \mathbb{R}^{N_r}$ is the reduced state on the basis functions and is also called as the modal coefficients and $\boldsymbol{\Phi}_{\boldsymbol{\mu}}$ is the POD basis function for a set of parameters $\boldsymbol{\mu}$. The ROM is obtained by substituting the low-rank approximation given in Eq. (13) into the full order model defined in Eq. (2) and then taking the inner product with test basis functions to yield a system of N_r ordinary differential equations (ODEs). For many fluid mechanics problems, the mechanistic description of some variables or processes is not available or insufficient for the desired task [i.e., the term $\boldsymbol{\pi}(\cdot)$ is unknown]. Therefore, the physics-based intrusive approaches like Galerkin projection ROM (GROM) provide us

$$\dot{\boldsymbol{a}} = \boldsymbol{\Phi}_{\boldsymbol{\mu}}^T \boldsymbol{f}(\boldsymbol{\Phi}_{\boldsymbol{\mu}}\boldsymbol{a}; \boldsymbol{x}, t; \boldsymbol{\mu}). \tag{14}$$

We note here that in GROM, the test basis functions are the same as the trial basis, which allows us to make use of the orthonormality, that is, $\boldsymbol{\Phi}_{\boldsymbol{\mu}}^T\boldsymbol{\Phi}_{\boldsymbol{\mu}} = \boldsymbol{I}_{N_r}$.

One of the limitations with intrusive ROMs like GROM is that the governing equations of the system have to be known exactly. If the governing equations are not known exactly, the effect of unknown dynamics is truncated in GROMs and accurate prediction is not achieved. This issue is mitigated in NIROMs that exploit machine learning (ML) algorithms to learn the reduced-order dynamics from the observed data.[34] The training data for the ML algorithm can be generated by using the same data used for computing the POD basis functions as follows:

$$\boldsymbol{\alpha}(t; \boldsymbol{\mu}) = \langle \boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}), \boldsymbol{\Phi}_{\boldsymbol{\mu}} \rangle, \tag{15}$$

where $\boldsymbol{\alpha}$ is the data projection coefficient, and the angle parentheses refer to the Euclidean inner product defined as $\langle \boldsymbol{p}, \boldsymbol{q} \rangle = \boldsymbol{p}^T\boldsymbol{q}$. The neural network is one of the most successful algorithms for learning the

dynamics on reduced-order basis functions. However, as the complexity of the problems increases, the number of trainable parameters quickly explodes and neural networks suffer from high epistemic uncertainty associated with limited training data. We address this problem by hybridizing intrusive and non-intrusive approaches through our PGML framework. More specifically, in Sec. III we present a PGML approach to combine a data-driven neural network model and a projection-based ROM for improved model generalizability.

III. PHYSICS-GUIDED MACHINE LEARNING (PGML) FRAMEWORK

In this section, we detail different components of the proposed PGML framework for modeling the reduced-order representation of the high-dimensional nonlinear dynamical system. In a broader sense, the PGML is a modular approach to leverage the interpretable physics-based model with a deep learning model. Thus, the PGML makes it possible to inject known physics during the training and deployment processes to reduce uncertainty, and consequently improve the trade-off between efficiency and accuracy. Since it relies on injecting additional features into the internal layers of neural networks as shown in Fig. 1, it can also be viewed as a highly generic approach to embed simplified theories, physics-based kernels or features directly into deep neural network models. For example, in our recent study on aerodynamics load calculations,[54] we show that embedding a panel method significantly improves the deep learning-based surrogate models (e.g., achieved approximately 75% reduction in uncertainty). In this paper, our aim is to demonstrate how the PGML combines physical information with POD-based GROM, and explore numerically how these physics-based features assist the neural network models in constraining the output to a manifold of the physically more realizable solution and lead to improved generalizability for the extrapolation regime beyond the training datasets. Moreover, we study the sensitivity of the layer-wise location of the physics injection. Here, we highlight that there are more advanced hyperparameter tuning methods on designing automated neural network architectures (e.g., using genetic algorithms[55]) and these automated machine learning approaches can be utilized to find the optimal layer(s) to inject the physics in the PGML architectures, a topic we would like to explore in our future studies.

Different components of the PGML framework for ROM are illustrated in Fig. 2. First, we present the formulation of the LSTM neural network for time series modeling and then discuss its application within the PGML framework. In a typical ML algorithm for time series modeling, the model is trained on a time series of observable $\{\mathbf{o}^{(1)}, \ldots, \mathbf{o}^{(T)}\}$, where $\mathbf{o} \in \mathbb{R}^{d_o}$, sampled at a fixed time interval. The LSTM neural network is one of the most popular ML algorithms for time-series modeling due to its ability to model long-term temporal dependencies without suffering from the vanishing gradient problem of recurrent neural network.[56] Recently, LSTM has also been very successful in modeling high-dimensional spatiotemporal chaotic systems and for ROMs.[57–61] The general functional form of the models used for time series forecasting can be written as

$$\mathbf{h}^{(t)} = f_h^h(\mathbf{o}^{(t)}, \mathbf{h}^{(t-1)}), \tag{16}$$

$$\tilde{\mathbf{o}}^{(t+1)} = f_h^o(\mathbf{h}^{(t)}), \tag{17}$$

where $\mathbf{o}^{(t)}$ is the current observable, and $\tilde{\mathbf{o}}^{(t+1)}$ is the forecast for the observable at the next time step $\mathbf{o}^{(t+1)}$ and $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state. Here, f_h^h is the hidden-to-hidden mapping and f_h^o is the hidden-to-output mapping.

The LSTM mitigates the issue with vanishing (or exploding) gradient by employing the gating mechanism that allows information to be forgotten. The equations that implicitly define the hidden-to-hidden

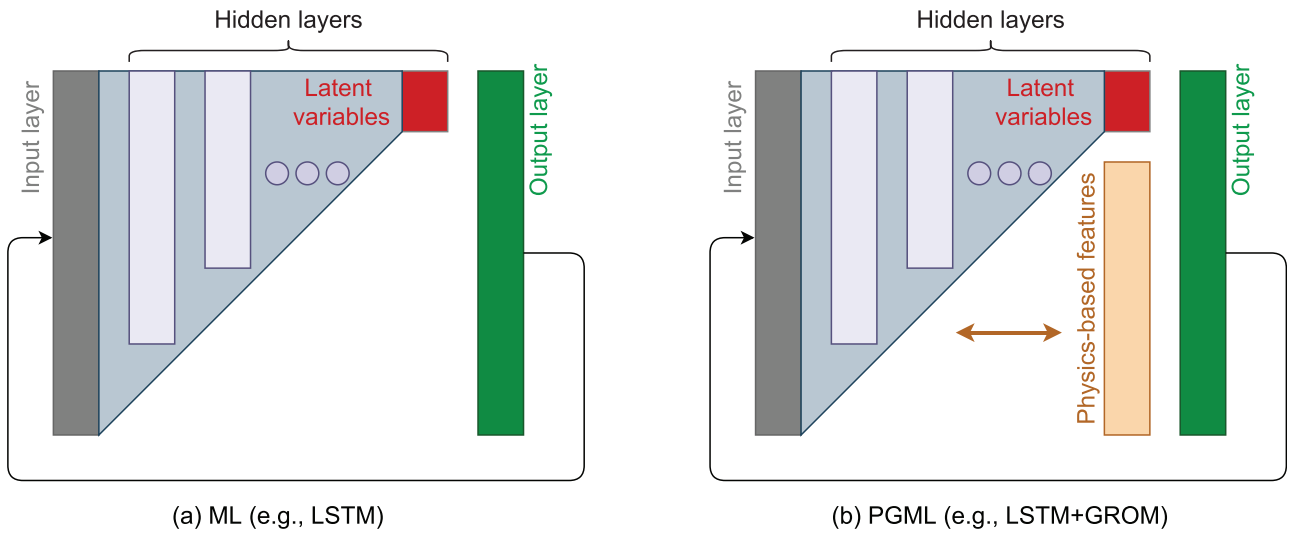

FIG. 1. General schematic of the PGML framework. The PGML is a modular deep neural network architecture that makes it possible to inject known physics during the training and deployment processes to reduce uncertainty, and consequently improve the trade-off between efficiency and accuracy. Of particular interest, in this paper we consider (a) an end-to-end non-intrusive model reduction approach using a recurrent neural network architecture based on the LSTM, and (b) its PGML version with the GROM features embedded into the LSTM layers. As one of the main features of the PGML framework, we can concatenate additional features into the middle layers of neural networks without breaking the neural network training process.

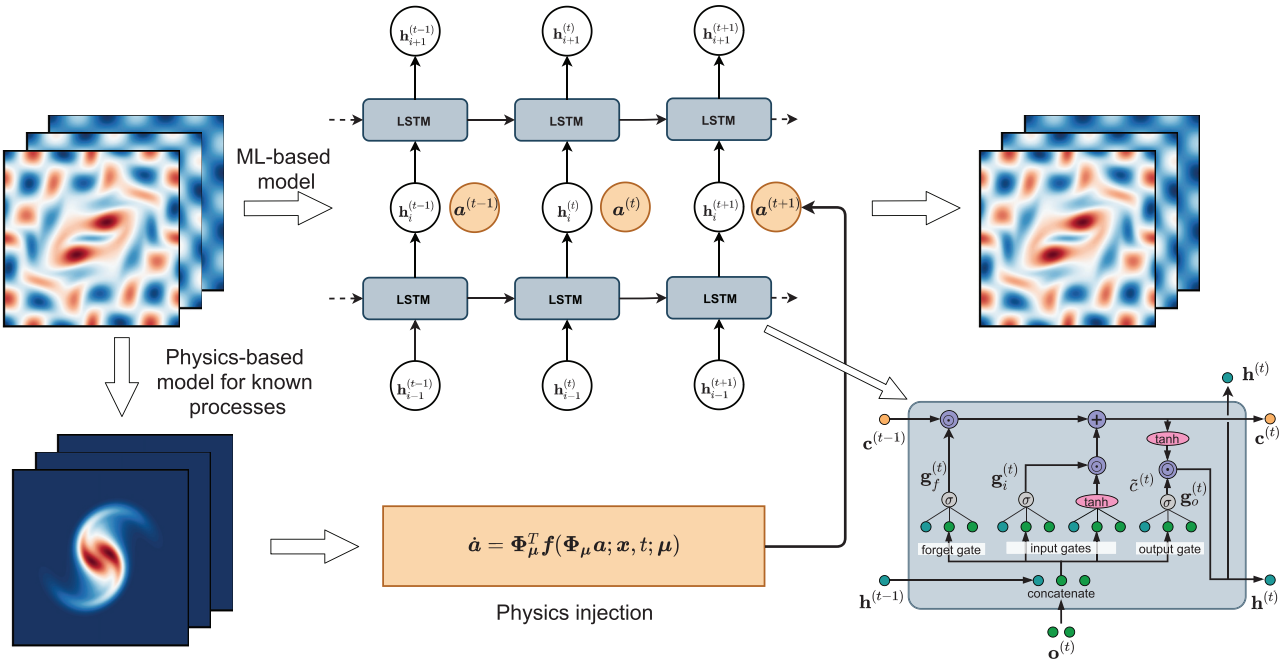

FIG. 2. Proposed physics-guided machine learning (PGML) framework for reduced-order modeling, where the physics-based features are directly embedded into hidden layers of the neural network along with learned latent variables. Here, the physics-based features obtained by GROM assist the LSTM-based neural network models in constraining the output to a manifold of the physically more realizable solution, and lead to improved generalizability for the extrapolation regime beyond the training datasets.

mapping from hidden state from the previous time step [i.e., $\mathbf{h}^{(t-1)}$] and input vector at the current time step [i.e., $\mathbf{o}^{(t)}$] to the forecast hidden state [i.e., $\mathbf{h}^{(t)}$] can be written as

$$\mathbf{g}_f^{(t)} = \sigma_f(\mathbf{W}_f[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_f), \tag{18}$$

$$\mathbf{g}_i^{(t)} = \sigma_i(\mathbf{W}_i[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_i), \tag{19}$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_c), \tag{20}$$

$$\mathbf{c}^{(t)} = \mathbf{g}_f^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{g}_i^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \tag{21}$$

$$\mathbf{g}_o^{(t)} = \sigma_o(\mathbf{W}_o[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_o), \tag{22}$$

$$\mathbf{h}^{(t)} = \mathbf{g}_o^{(t)} \odot \tanh(\mathbf{c}^{(t)}), \tag{23}$$

where $\mathbf{g}_f^{(t)}$, $\mathbf{g}_i^{(t)}$, $\mathbf{g}_o^{(t)} \in \mathbb{R}^{d_h}$ are the forget gate, input gate, and output gate, respectively. The $\mathbf{o}^{(t)} \in \mathbb{R}^{d_o}$ is the input vector at time t, $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state, $\mathbf{c}^{(t)} \in \mathbb{R}^{d_h}$ is the cell state, $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_c$, $\mathbf{W}_o \in \mathbb{R}^{d_h \times (d_h + d_o)}$ are the weight matrices, and $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_c$, $\mathbf{b}_o \in \mathbb{R}^{d_h}$ are the bias vectors. The symbol $\odot$ denotes the element-wise multiplication, and σ is the sigmoid activation function. The activation functions $\sigma_f, \sigma_i, \sigma_o$ are sigmoids. The hidden-to-output mapping is given by a fully connected layer with a linear activation function as follows:

$$\tilde{\mathbf{o}}^{(t+1)} = \mathbf{W}_o \mathbf{h}^t, \tag{24}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_o \times d_h}$.

When we apply the LSTM neural network for building NIROMs, the reduced-order state of the system at a future time step, that is, $\boldsymbol{\alpha}^{(t+1)}$, is learned as the function of a short history of d past temporally consecutive reduced-order states as follows:

$$\boldsymbol{\alpha}^{(t+1)} = \mathcal{F}(\underbrace{\boldsymbol{z}^{(t)}, \boldsymbol{z}^{(t-1)}, \ldots, \boldsymbol{z}^{(t-d+1)}}_{\boldsymbol{z}^{(t):(t-d+1)}}; \theta), \tag{25}$$

where $\mathcal{F}(\cdot\,; \theta)$ is the nonlinear function parameterized by a set of parameters θ, and $\boldsymbol{z}$ refers to input features consisting of the POD modal coefficients and a set of parameters governing the system, that is, $\boldsymbol{z} \in \mathbb{R}^{N_r + N_p}$. The LSTM is trained using the backpropagation through time (BPTT) algorithm, and the parameters are optimized as

$$\theta^* = \arg\min_\theta \frac{1}{N - d + 1} \sum_{n=d}^{N} ||\mathcal{F}(\underbrace{\boldsymbol{z}^{(t):(t-d+1)}}_{\zeta}; \theta) - \boldsymbol{\alpha}^{(t+1)}||_2^2. \tag{26}$$

We employ $(N_l - 1)$ LSTM layers and a single dense layer with a linear activation function as the last layer. Therefore, the ML model can be written as

$$\mathcal{F}_{\mathrm{ML}}(\boldsymbol{\zeta}; \theta) = \boldsymbol{h}_{N_l}(\cdot; \Theta_{N_l}) \circ \cdots \circ \boldsymbol{h}_2(\cdot; \Theta_2) \circ \boldsymbol{h}_1(\boldsymbol{\zeta}; \Theta_1), \tag{27}$$

where the output of each LSTM layer $(i = 1, \ldots, N_l - 1)$ is $\boldsymbol{h}_i(\cdot; \Theta_i) \in \mathbb{R}^{d \times N_h}$ and the last dense layer maps the final hidden state to the output, that is, $\boldsymbol{h}_{N_l}(\cdot; \Theta_{N_l}) : \mathbb{R}^{N_h} \to \mathbb{R}^{N_r}$. Here, N_h is the number of hidden units in the LSTM cell and we use the constant number of hidden units across all LSTM layers.

In the PGML framework, the features extracted from the physics-based model are embedded into the ith intermediate hidden layer along with the latent variables as follows:

$$\mathcal{F}_{\text{PGML}}(\boldsymbol{\zeta};\theta) = \boldsymbol{h}_{N_l}(\cdot;\Theta_{N_l}) \circ \cdots \circ \underbrace{\times\, \mathcal{C}(\boldsymbol{h}_i(\cdot;\Theta_i), \boldsymbol{a}^{(t):(t-d+1)})}_{\text{Physics injection}} \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{\zeta};\Theta_1), \quad (28)$$

where $\mathcal{C}(\cdot,\cdot)$ represents the concatenation operation and $\mathbf{a}$ are the physics-based features (i.e., the GROM modal coefficients). Therefore, the output of ith layer will be in $\mathbb{R}^{d\times(N_r+N_h)}$. We highlight here that the physics-based features are embedded only at a single layer of the neural network. Therefore, at which layer shall the physics-based features be embedded becomes another hyperparameter of the PGML framework. While some guidelines or rule of thumb can be established based on the interplay between known and unknown physics, a thorough systematic investigation is needed. Methods like layer-wise relevance propagation can be adopted to bring interpretability to complex neural networks in the PGML framework and we consider it as part of our future work. The physics-based features assist the LSTM network in constraining the output to a manifold of the physically realizable solution and leads to improved generalizability for the extrapolation regime. Even though we demonstrate the PGML framework for POD-ROM, we emphasize here that the PGML framework is highly modular and can also be applied to nonlinear, cluster-based, and network-based ROMs. For example, we do not need to restrict to only linear basis construction methods like POD and the PGML framework can be easily extended to nonlinear manifold generation methods like convolutional autoencoders.[7,35,62]

IV. VORTEX MERGING EXPERIMENTS

We apply the PGML framework to the two-dimensional vorticity transport equation for the vortex merger problem as a prototypical test case. This problem involves the study of the merging of two co-rotating vortices. The two-dimensional vorticity transport equation can be written as

$$\frac{\partial\omega}{\partial t} + J(\omega,\psi) = \frac{1}{\text{Re}}\nabla^2\omega + \boldsymbol{\pi}(\boldsymbol{x},t;\boldsymbol{\mu}), \quad (29)$$

$$\nabla^2\psi = -\omega, \quad (30)$$

where ω is the vorticity defined as $\omega = \nabla\times\mathbf{u}$, $\mathbf{u} = [u,v]^T$ is the velocity vector, ψ is the streamfunction, Re is the Reynolds number parameterizing the system, and $\boldsymbol{\pi}(\cdot)$ represents the unknown physics. Using the notion introduced in Eq. (3), we can rewrite Eq. (29) as

$$\frac{\partial\omega}{\partial t} = \hat{\boldsymbol{f}}(\boldsymbol{x},t;\boldsymbol{\mu}) + \boldsymbol{\pi}(\boldsymbol{x},t;\boldsymbol{\mu}), \quad (31)$$

where

$$\hat{\boldsymbol{f}}(\boldsymbol{x},t;\boldsymbol{\mu}) = \frac{1}{\text{Re}}\nabla^2\omega - J(\omega,\psi). \quad (32)$$

The two terms on the right hand side of Eq. (32), that is, the linear Laplacian term and the nonlinear Jacobian term, represent the known physics of the system and are defined as follows:

$$\nabla^2\omega = \frac{\partial^2\omega}{\partial x^2} + \frac{\partial^2\omega}{\partial y^2}, \quad (33)$$

$$J(\omega,\psi) = \frac{\partial\psi}{\partial y}\frac{\partial\omega}{\partial x} - \frac{\partial\psi}{\partial x}\frac{\partial\omega}{\partial y}. \quad (34)$$

To demonstrate the proposed PGML framework for ROMs, we assume that $\boldsymbol{\pi}$ is unknown. Therefore, for the best-case scenario, the known processes can be represented by

$$\frac{\partial\omega}{\partial t} + J(\omega,\psi) = \frac{1}{\text{Re}}\nabla^2\omega \quad (35)$$

together with the kinematic relationship given by Eq. (30). As discussed in Sec. II, the reduced-order approximation of the vorticity and streamfunction fields can be written as

$$\omega(x,y,t) \approx \sum_{k=1}^{N_r} a_k(t)\phi_k^{\omega}(x,y), \quad (36)$$

$$\psi(x,y,t) \approx \sum_{k=1}^{N_r} a_k(t)\phi_k^{\psi}(x,y). \quad (37)$$

The vorticity and streamfunction share the same time-dependent modal coefficients as they are both related by the kinematic relationship given in Eq. (30). Furthermore, the POD basis functions of the streamfunction can be obtained by solving the below Poisson equation

$$\nabla^2\phi_k^{\psi} = -\phi_k^{\omega}. \quad (38)$$

The GROM for the vorticity transport equations considering only the known physics can be written as

$$\dot{\boldsymbol{a}} = \mathfrak{L}\boldsymbol{a} + \boldsymbol{a}^T\mathfrak{N}\boldsymbol{a} \quad (39)$$

or more explicitly

$$\frac{\mathrm{d}a_k}{\mathrm{d}t} = \sum_{i=1}^{N_r}\mathfrak{L}_k^i a_i + \sum_{i=1}^{N_r}\sum_{j=1}^{N_r}\mathfrak{N}_k^{ij}a_i a_j, \quad (40)$$

where $\mathfrak{L}$ and $\mathfrak{N}$ are the linear and nonlinear operators. We highlight that the model provided by Eq. (40) does not have access to the unknown processes $\boldsymbol{\pi}$. Since it has access to the transport operators given by Eq. (32), it is thus referred to a physics-based model (i.e., an intrusive reduced-order model) in this work, even though its basis functions are generated from data. Instead of the POD basis function expansion provided by Eqs. (36) and (37), one can generalize the model using a standard Fourier Galerkin approach,[63] completely eliminating the dependency on a prerecorded dataset on building a physics-based ROM. The linear and nonlinear operators for the vorticity transport equation are as follows:

$$\mathfrak{L}_{ik} = \left\langle \frac{1}{\text{Re}}\nabla^2\phi_i^{\omega}, \phi_k^{\omega}\right\rangle, \quad (41)$$

$$\mathfrak{N}_{ijk} = \langle -J(\phi_i^{\omega},\phi_j^{\psi}), \phi_k^{\omega}\rangle, \quad (42)$$

where the angle parentheses refer to the Euclidean inner product defined as $\langle\mathbf{x},\mathbf{y}\rangle = \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{N} x_i y_i$. We apply the third-order Adams–Bashforth (AB3) numerical scheme to solve the system of ODEs given in Eq. (40). In discrete sense, the update formula can be written as

$$a_k^{(n+1)} = a_k^{(n)} + \Delta t\sum_{q=0}^{s}\beta_q G(a_k^{(n-q)}), \quad (43)$$

where s and β_q are the constants corresponding to AB3 scheme, which are $s = 2$, $\beta_0 = 23/12$, $\beta_1 = -16/12$, and $\beta_2 = 5/12$. Here, the operator $G(a_k)$ is the right hand side of Eq. (40) as follows:

$$G(a_k) = \sum_{i=1}^{N_r} \mathfrak{L}_k^i a_i + \sum_{i=1}^{N_r}\sum_{j=1}^{N_r} \mathfrak{N}_k^{ij} a_i a_j. \tag{44}$$

We start with an initial vorticity field of two Gaussian-distributed vortices

$$\omega(x,y,0) = \exp\left(-\pi\left[(x-x_1)^2 + (y-y_1)^2\right]\right) + \exp\left(-\pi\left[(x-x_2)^2 + (y-y_2)^2\right]\right), \tag{45}$$

where their centers are initially located at $(x_1, y_1) = (3\pi/4, \pi)$ and $(x_2, y_2) = (5\pi/4, \pi)$. We utilize an arbitrary array of decaying Taylor–Green vortices as the unknown processes parameterized by $\mu = \{\mathrm{Re}\}$, i.e., we have

$$\boldsymbol{\pi}(x,y,t;\mathrm{Re}) = -\gamma\, e^{-t/\mathrm{Re}} \cos(3x)\cos(3y). \tag{46}$$

The parameter γ controls the strength of unknown physics compared to the known physics. The synthetic data for building the ROM are generated by solving Eq. (29) on the computational domain $(x, y) \in [0, 2\pi]$ with periodic boundary conditions and 256^2 spatial discretization. The system is evolved for 20 time units with a time step size of $\Delta t = 1 \times 10^{-3}$. The data snapshots for the training are obtained for $\mathrm{Re} = \{200, 400, 600, 800\}$, and the trained models are evaluated for $\mathrm{Re} = \{1000, 1500, 2000, 3000\}$. Specifically, the physical variable of interest is the vorticity field, that is, $\boldsymbol{u} = \{\omega\}$, and the unknown source term $\boldsymbol{\pi}(\cdot)$ is given in Eq. (46). We retain $N_r = 8$ POD modes for all Reynolds numbers and these POD modes captures more than 99% of the energy of the system as illustrated in Fig. 3. Instantaneous snapshots of the vorticity field for different Reynolds numbers at different time instances are displayed in Fig. 4. We note here that the vorticity field shown in Fig. 4 is obtained by solving the vorticity transport equation assuming that there is no unknown physics, that is, setting $\gamma = 0.0$.

The LSTM network in the ML model consists of $N_l = 5$ layers with the first four layers as the LSTM layers and the dense layer with a linear activation function as the output layer. Each LSTM layer has $N_h = 80$ hidden units, and tanh activation function is applied. From our experimentation with the hyperparameters, we found that the results were not improving by employing deeper network architecture, and in some cases, deeper networks led to poor performance due to overfitting. We utilize $d = 3$; that is, the state of the system for three previous time steps is used to predict the future state. Since we are using the information of only d past temporally consecutive states as the input, the LSTM can capture dependencies up to d previous time steps. During the online deployment, the initial condition for the first d time steps is provided. This information is used to predict the forecast state at $(d+1)$th time step. Then, the state of the system from $2-(d+1)$ is used to predict the forecast state at $(d+2)$th time step. This procedure is continued until the final time step. The PGML framework also uses the same LSTM network architecture, and the features from the Galerkin projection model (i.e., GROM modal coefficients) are embedded into the third LSTM layer. We reiterate here that the layer at which the physics-based features should be embedded is another hyperparameter of PGML framework and methods like neural architecture search can be used for complex problems.[64] The online deployment of the PGML framework is similar to the ML framework. We characterize the generalizability of the ML and PGML model through uncertainty estimate computed using the deep ensembles method where ensemble of neural networks are trained with different initialization of parameters.[65–67] In deep ensembles method, the prediction from the ensemble of neural networks is interpreted as the epistemic uncertainty. Despite the simplicity of this method, it is appealing due to its scalability and strong empirical results, showing that the uncertainty estimate is as good as the Bayesian neural networks.[67] The quantitative performance for ML and PGML framework is measured through the statistics of the weighted root mean squared error (WRMSE) defined as

$$\epsilon(i) = \sum_{k=1}^{N_r} \lambda_k \left(\frac{1}{N_t} \sum_{n=1}^{N_t} (\alpha_k^{(n)} - \tilde{\alpha}_k^{(n)}(i))^2 \right)^{1/2}, \tag{47}$$

where $\lambda_k = \sigma_k^2 / (\sum_{j=1}^{N_r} \sigma_j^2)$ represents the energy in each POD mode, and $\tilde{\alpha}_k(i)$ is the prediction of the modal coefficients for ith ensemble. Each neural network in the ensemble set is initiated by using a different initial seed. The statistics of the WRMSE, that is, the mean and the standard deviation (SD), is computed as follows:

$$\bar{\epsilon} = \frac{1}{N_e} \sum_{i=1}^{N_e} \epsilon(i), \quad s_\epsilon = \left(\frac{\sum_{i=1}^{N_e} (\epsilon(i) - \bar{\epsilon})^2}{N_e - 1} \right)^{1/2}, \tag{48}$$

where N_e is the size of ensemble of the neural network. We set $N_e = 30$ for both ML and PGML models in our study.

Figure 5 shows the temporal evolution of selected modal coefficients predicted by true projection, GP, ML, and PGML models at $\mathrm{Re} = 1500$ for $\gamma = 0.01$. The ensemble-averaged modal coefficients predicted by the PGML model matches very accurately with the true projection compared to the GP and ML model. We also observe that the ensemble-average prediction by the ML model is also improved compared to the GP model. However, the prediction of the ML model is marred by high uncertainty compared to the PGML model

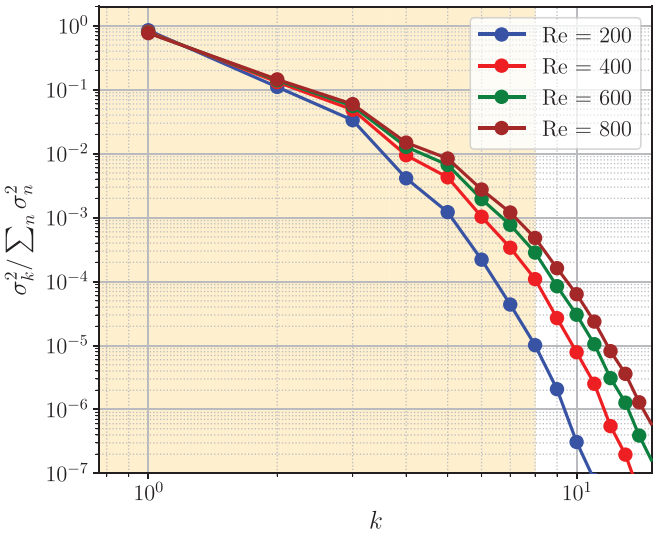

FIG. 3. Amount of the kinetic energy captured by each POD modes.

FIG. 4. Temporal evolution of the vorticity field at different time instances for several Reynolds number investigated in this study. We note here that the vorticity field is presented for $\gamma = 0.0$.

prediction. From Fig. 4, we can notice that the flow is topologically very similar at different Reynolds numbers and the main difference is in the amplitude of the vorticity field. Therefore, the task of extrapolation for this problem is not very challenging for neural networks. However, in this study, we demonstrate the proof of concept to show the benefit of PGML over a purely data-driven ML model, and evaluating the PGML framework for topologically different datasets is a part of our future work. Figure 6 depicts the evolution of selected modal coefficients predicted by true projection, GP, ML, and PGML models at $\mathrm{Re} = 3000$ for $\gamma = 0.01$. The ensemble-averaged modal coefficient predicted by the ML model is very inaccurate with high uncertainty as the test Reynolds number is significantly different from the training Reynolds numbers. The magnitude of the unknown source term is relatively small for $\gamma = 0.01$, and the GP model can predict the modal coefficient with sufficient accuracy. The PGML model achieves the most accurate prediction with less uncertainty as shown in Fig. 6. Here, we would like to emphasize that the accuracy of any supervised ML approach mostly depends on the quality and relevance of the training dataset. Even though all ML models are trained from the same dataset, the underpinning optimization problem yields different predictor model (i.e., a different local minimum) at each realization due to random initialization of the weights. One of our chief goals in developing the PGML approach is to reduce the uncertainty of these predictor models through an information or knowledge injection process. Our numerical results support that the uncertainty of the ML models could be improved significantly through injection of the GROM features that are derived from the known part of the governing equations.

In Fig. 7, the temporal evolution of the last mode (i.e., a_8) predicted by true projection, GP, ML, and PGML model is displayed for different magnitudes of the unknown source term. Overall, we can observe that the PGML framework can predict the modal coefficients with high confidence up to $\gamma = 0.04$ and the prediction for $\gamma = 0.1$ is less certain. One of the reasons behind inaccurate prediction at

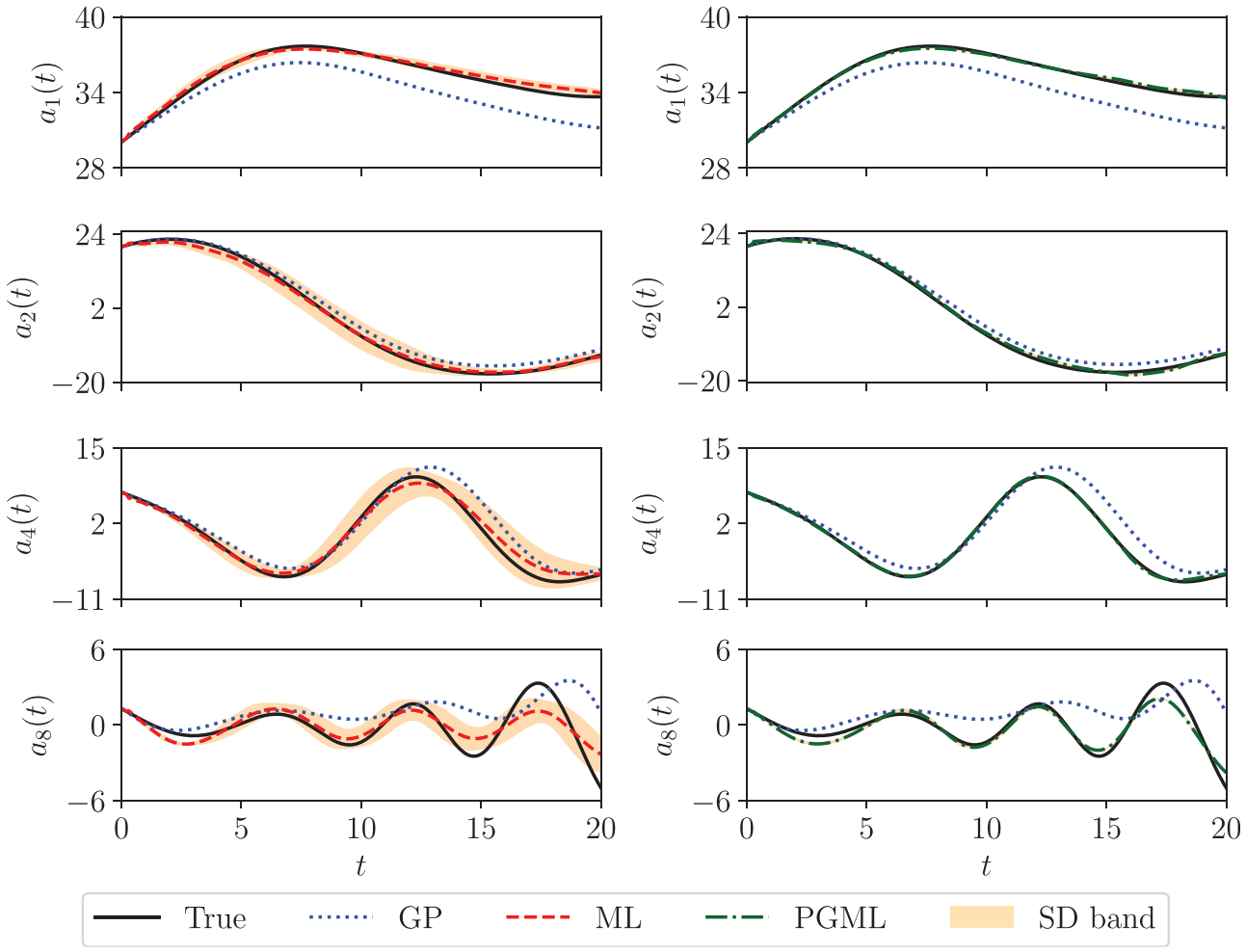

FIG. 5. Temporal evolution of selected modal coefficients for the vortex-merger test case at $\mathrm{Re} = 1500$ and $\gamma = 0.01$. The ML (left) and PGML (right) represent the average of the modal coefficients predicted by all neural networks trained using MSE loss function with different seeds for initialization of the parameters. Although the ensemble-averaged ML model provides more accurate predictions than the GP model, there is a large standard deviation (SD) band over all ML model predictions.

$\gamma = 0.1$ can be due to a very poor correlation between the known physics in the GP model and the actual physics of the system.

We also evaluate the capability of GP, ML, and PGML framework in reconstructing the vorticity field using the RMSE between the true vorticity field and the predicted vorticity field. The RMSE is defined as

$$\mathrm{RMSE}(t) = \left(\frac{1}{N} \sum_{i=1}^{N} (\omega_T(\boldsymbol{x}_i, t) - \omega_R(\boldsymbol{x}_i, t))^2 \right)^{1/2}, \qquad (49)$$

where N is the spatial resolution (i.e., $N = N_x \times N_y$), ω_T is the true projection of the vorticity field, and ω_R is the vorticity field predicted by ROM [computed using Eq. (36)]. The ω_R is computed using the ensemble-averaged modal coefficients predicted by ML and PGML models. Figure 8 shows the time evolution of RMSE for different magnitudes of the source term at $\mathrm{Re} = 1000$ and $\mathrm{Re} = 1500$. We can observe that when the complete physics of the vortex-merging process is known (i.e., $\gamma = 0.0$), the RMSE for both ML and PGML framework is less than the GP model at $\mathrm{Re} = 1000$. However, at $\mathrm{Re} = 1500$, the RMSE for the ML model is higher than the GP model. This is due to the poor extrapolation capability of a purely data-driven ML model. On the other hand, the PGML model can successfully use the physics-based information from the GP model leading to less RMSE compared to both GP and ML models. The vorticity field predicted by the GP model quickly diverges from the true vorticity field in the presence of unknown physics and overall the PGML prediction is better than the ML model as illustrated in Fig. 8. The difference between the vorticity field at the final time $t = 20$ predicted by true projection modal coefficients and GP, ML, and PGML models is depicted in Fig. 9.

Table I reports the quantitative analysis of GP, ML, and PGML models at $\mathrm{Re} = \{1000, 1500\}$ for different magnitudes of the unknown source term. For the ML and PGML models, the WRMSE mentioned in Table I is calculated using Eq. (50) and its statistics is computed with Eq. (48). For the GP model, we report WRMSE computed by using

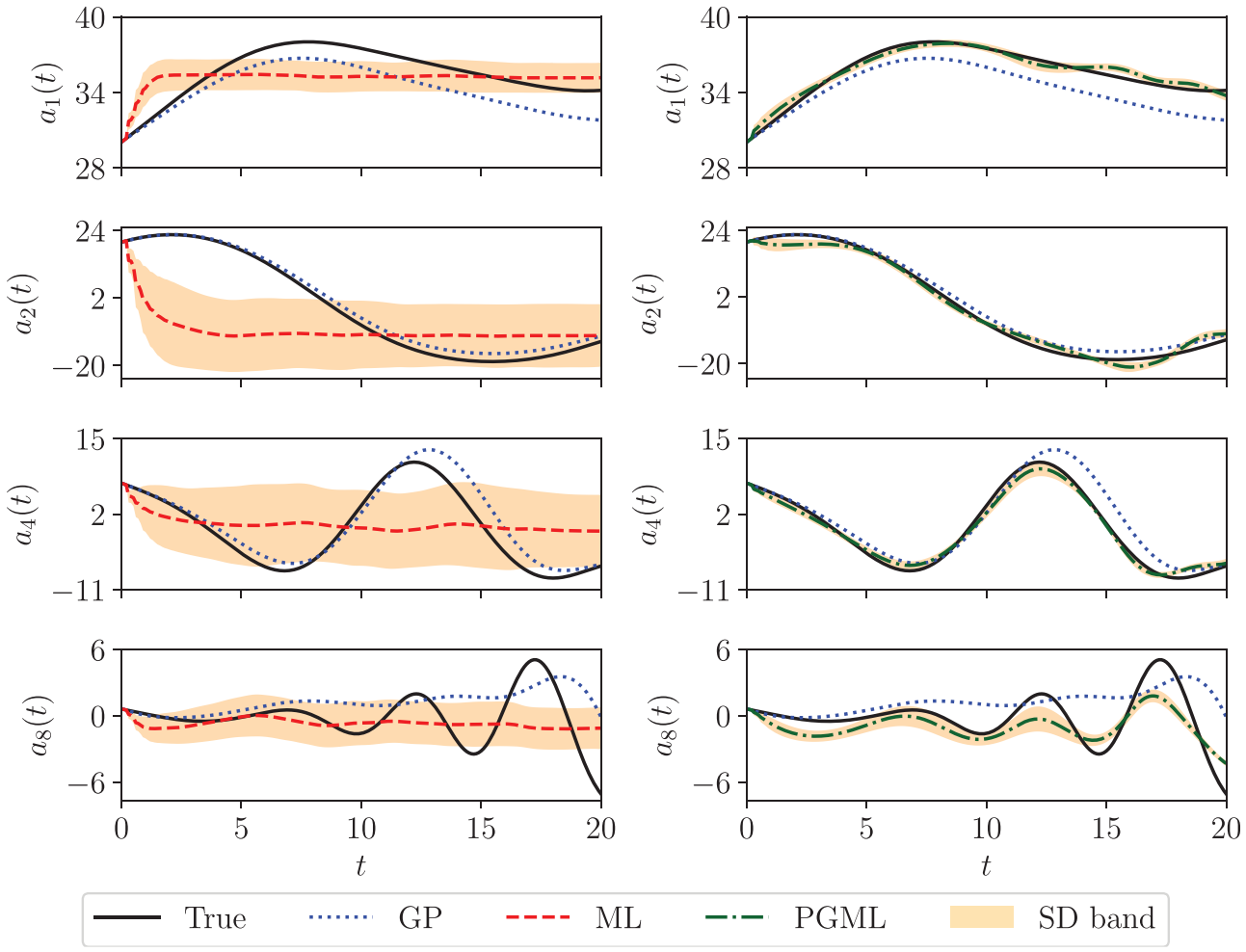

FIG. 6. Temporal evolution of selected modal coefficients for the vortex-merger test case at $\mathrm{Re} = 3000$ and $\gamma = 0.01$. The ML (left) and PGML (right) represent the average of the modal coefficients predicted by all neural networks trained using MSE loss function with different seeds for the initialization of the parameters. The ensemble-averaged ML model provides very inaccurate predictions than the GP model, and there is a large standard deviation (SD) band over all ML model predictions.

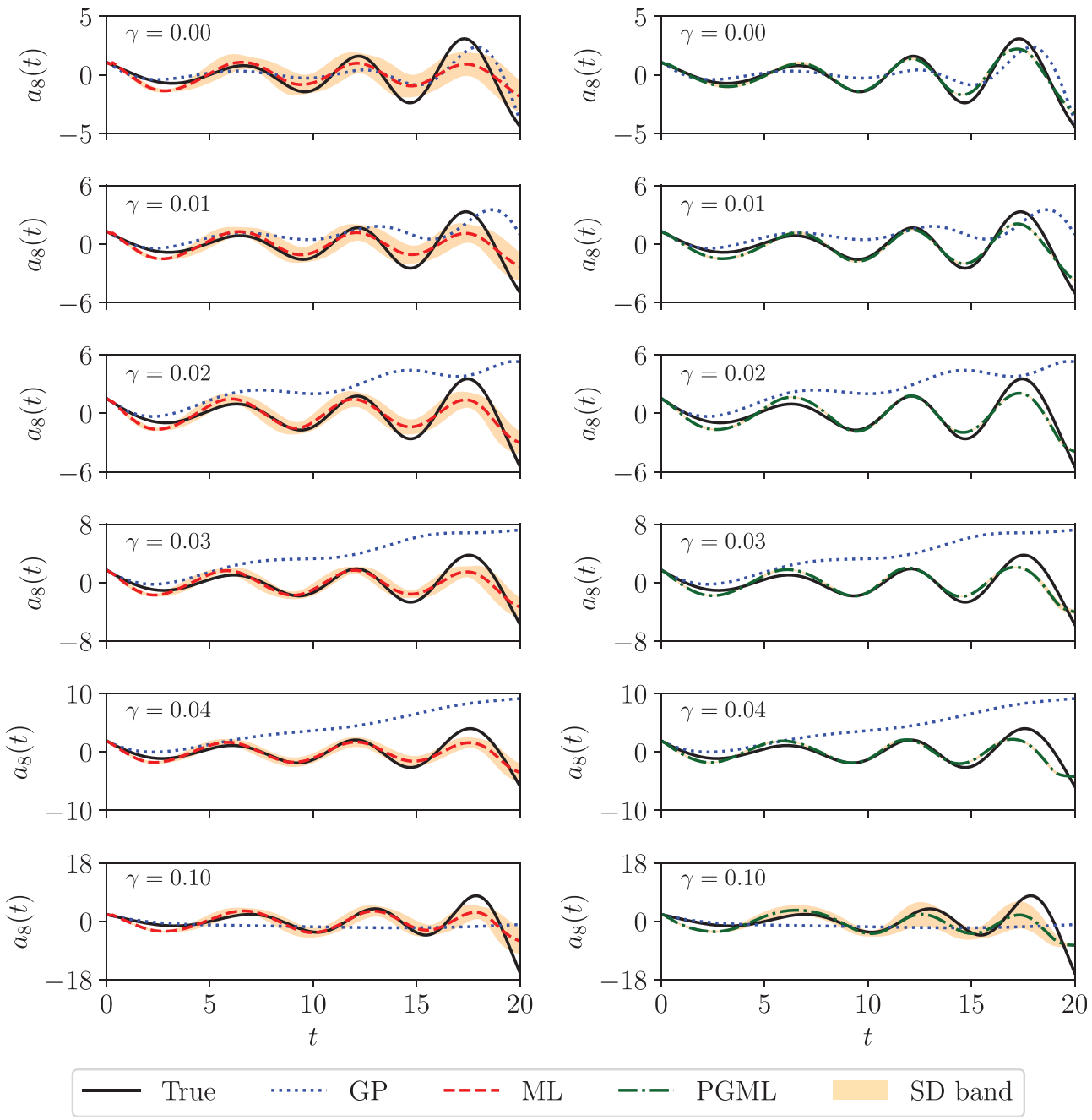

FIG. 7. Temporal evolution of the last modal coefficient for the vortex-merger test case at Re = 1500 and for different magnitudes of the source term. The dashed red curve represents the average of the modal coefficients predicted by ML model (left) and the PGML model (right).

$$\epsilon = \sum_{k=1}^{N_r} \lambda_k \left(\frac{1}{N_t} \sum_{n=1}^{N_t} (\alpha_k^{(n)} - a_k^{(n)})^2 \right)^{1/2}, \tag{50}$$

where α_k is the set of true projection coefficients, and a_k refers to the GROM coefficients. In Table I, $\gamma = 0.0$ corresponds to the special case where the physics is completely available to us. In that case, if the closure error is also negligible there is no need to build such a stochastic learning machine (i.e., a neural network derived from true projection coefficients), so a physics-based model (e.g., GP) will provide an accurate prediction. Moreover, as verified by Table I, we would expect that the GP approach should indeed yield more accurate results than the ML (or PGML) approach drawn from true projection coefficients,

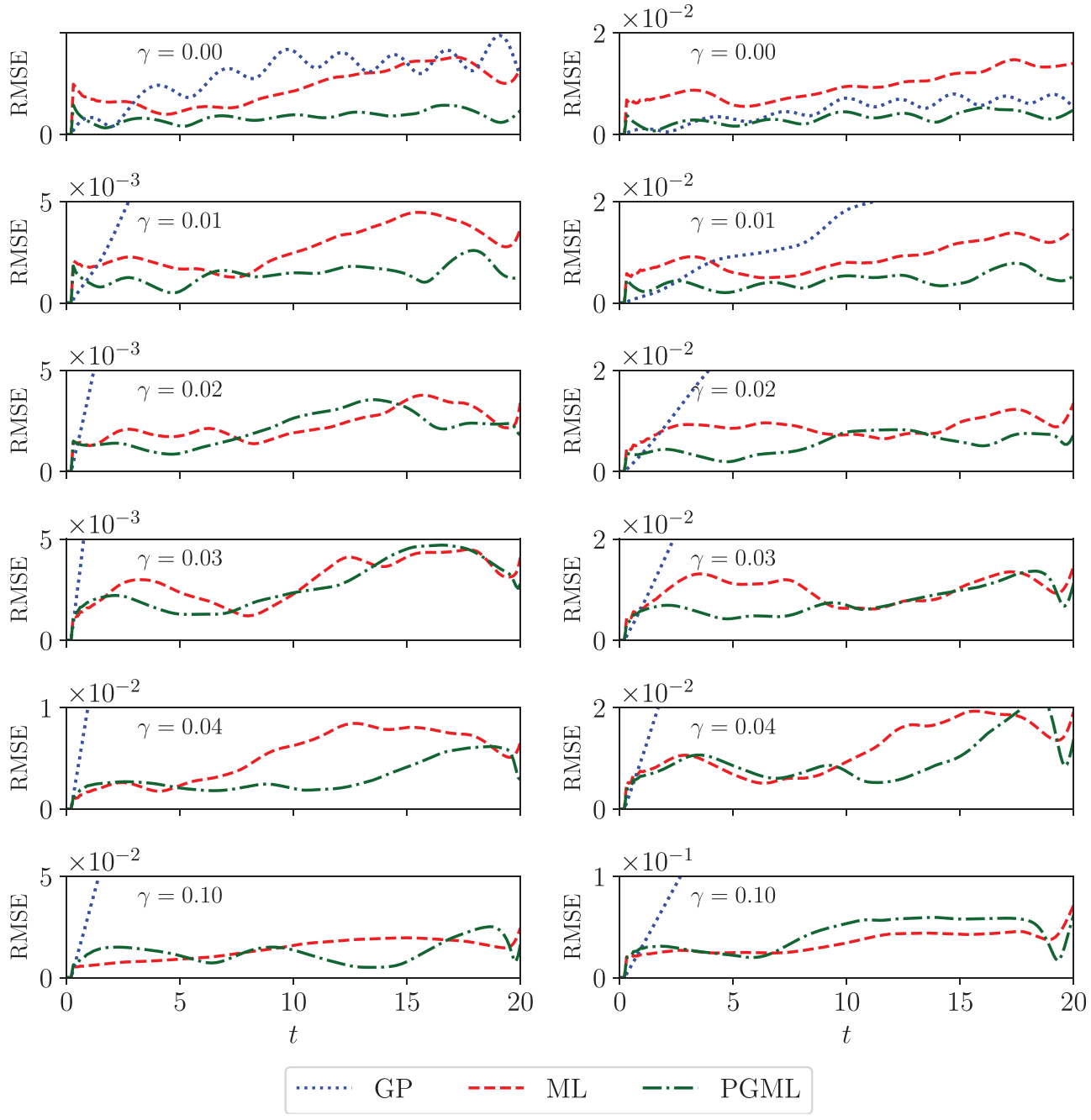

FIG. 8. Temporal evolution of the RMSE of the vorticity field [defined in Eq. (49)] for the vortex-merger test case at Re = 1000 (left) and Re = 1500 (right) for different magnitudes of the source term. The dashed red curve represents the average of the modal coefficients predicted by ML model (left) and the PGML model (right).

since the GP captures almost 100% of RIC at $\gamma = 0.0$. Furthermore, from Table I we can see that the WRMSE for the ML model is considerably higher for Re = 1500 at $\gamma = 0.0$ due to the poor extrapolation ability of pure data-driven models. However, the WRMSE for the PGML model is significantly lower than the ML. Yet, the WRMSE for the PGML model is slightly higher than the GP model even though the reconstructed vorticity field is more accurate as illustrated in Fig. 8. The WRMSE assigns more weightage to the first few modes, and therefore, the inaccurate prediction for dominant modes leads to a higher value of WRMSE for the PGML model. Being the primary

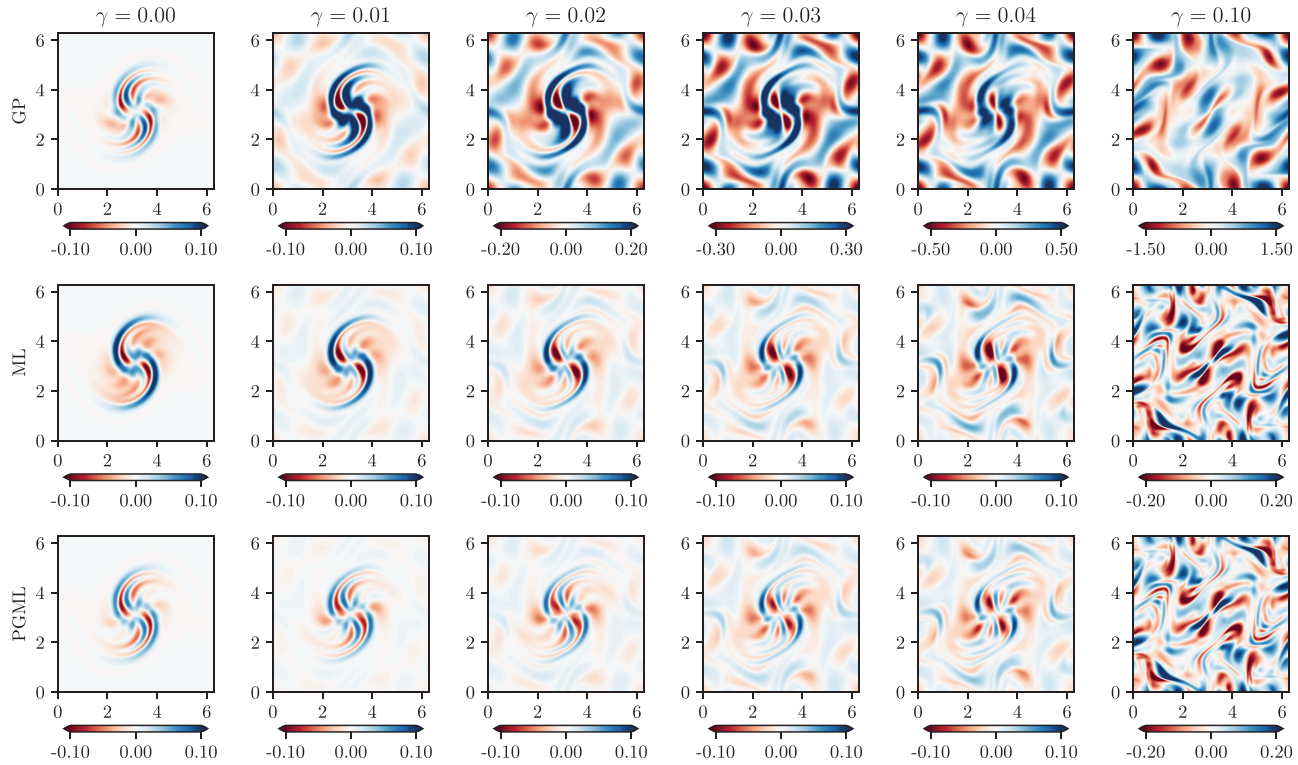

FIG. 9. The error in the prediction of the vorticity field at the final time $t = 20$ for the GP model (top), ML model (middle), and PGML model (bottom). The error corresponds to the vortex-merger test case at $\mathrm{Re} = 1500$.

focus of this study, for higher values of γ, the dynamics of the system is not known exactly and it gives us an opportunity to introduce a stochastic neural network model as a predictive engine. As demonstrated in Table I, the trustworthiness of the pure data-driven ML model is less due to the high value of the uncertainty in the prediction (i.e., high standard deviation as shown in parenthesis). However, once we incorporate physics-based features from the GP model in the PGML framework, the uncertainty in the prediction is reduced by almost one order magnitude.

We also consider the hypothetical situation where $\delta \boldsymbol{u} = 0.94$ (so the physics is almost completely unknown and as shown in Fig. 9, and unknown physics severely dominates the vortex merging processes), and then, it is not surprising that the PGML framework would not be in favor of the standard ML because of the injection of the unrepresentative physics. Considering a false-negative analogy, this further verifies the success of the PGML methodology and will automatically indicate that the injected physics is not representative of the underlying processes. In fact, this feature of the PGML framework could act as a diagnostic approach in many digital twin applications to physical systems. There are numerous simulation packages specifically designed for modeling and computing $\hat{\boldsymbol{f}}$ in these systems. On the other hand, exhaustive datasets become more and more available to

TABLE I. The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. (48) in predicting the modal coefficients for the vortex merging flows at $\mathrm{Re} = \{1000, 1500\}$. The training has been performed using data for $\mathrm{Re} = \{200, 400, 600, 800\}$. Note that the control parameter γ adjusts the relative strength between known physics and unknown physics (i.e., $\gamma = 0$ refers to the special case when physics is fully known), and $\delta \boldsymbol{u}$ quantifies the level of unknown processes.

	Re = 1000				Re = 1500			
γ	$\delta \boldsymbol{u}$	GP	ML	PGML	$\delta \boldsymbol{u}$	GP	ML	PGML
0.00	0.00	0.075	0.262 (0.190)	0.064 (0.012)	0.00	0.138	0.959 (0.632)	0.192 (0.024)
0.01	0.25	1.564	0.258 (0.194)	0.110 (0.012)	0.26	1.563	0.852 (0.521)	0.300 (0.029)
0.02	0.46	6.033	0.316 (0.113)	0.181 (0.026)	0.47	6.061	0.998 (0.458)	0.498 (0.040)
0.03	0.61	12.809	0.465 (0.387)	0.259 (0.040)	0.62	12.905	1.352 (0.582)	0.784 (0.092)
0.04	0.72	20.843	0.726 (0.591)	0.312 (0.047)	0.72	21.082	1.828 (0.992)	1.116 (0.149)
0.10	0.94	70.935	1.910 (0.804)	1.424 (0.356)	0.94	71.923	5.252 (2.387)	5.842 (6.227)

TABLE II. The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. (48) in predicting the modal coefficients for the vortex merging flows at $\mathrm{Re} = \{2000, 3000\}$. The training has been performed using data for $\mathrm{Re} = \{200, 400, 600, 800\}$.

	Re = 2000				Re = 3000			
γ	$\delta\boldsymbol{u}$	GP	ML	PGML	$\delta\boldsymbol{u}$	GP	ML	PGML
0.00	0.00	0.221	4.087 (2.179)	0.331 (0.062)	0.00	0.360	6.017 (0.833)	0.595 (0.147)
0.01	0.26	1.564	3.542 (2.003)	0.503 (0.086)	0.27	1.563	5.636 (0.904)	0.867 (0.219)
0.02	0.48	6.033	4.412 (2.297)	0.809 (0.095)	0.48	6.061	7.309 (1.036)	1.542 (0.474)
0.03	0.63	12.809	5.046 (2.364)	1.441 (0.342)	0.64	12.905	10.813 (2.811)	3.345 (1.071)
0.04	0.73	20.843	6.402 (3.409)	2.384 (0.589)	0.74	21.082	15.439 (3.931)	5.813 (1.519)
0.10	0.94	70.935	17.672 (14.194)	11.353 (6.816)	0.95	71.923	45.863 (13.155)	31.468 (9.623)

generate versatile machine learning-based predictive models. To this end, a confidence score $\delta\boldsymbol{u}$ defined in Eq. (4) can be utilized to assess the relative modeling strength of principled and data-driven models, where lower $\delta\boldsymbol{u}$ might indicate that the principled model is a reliable tool and the physics-based features from the principled model should be incorporated into data-driven models. This diagnostic aspect and the investigation of such an adaptive composite model will be the subject of our future studies.

To further investigate the limits of the proposed PGML approach, we carry out the quantitative analysis for $\mathrm{Re} = 2000$ and $\mathrm{Re} = 3000$. Table II lists the WRMSE and its statistics defined by Eq. (48) for higher Reynolds numbers. It can be observed that the WRMSE and the uncertainty in the prediction are significantly lower for the PGML model compared to the ML model for both Reynolds numbers. For all the numerical experiments conducted at different $\delta\boldsymbol{u}$ levels, it can be concluded that the PGML approach leads to significantly more accurate results compared to the ML approach, which clearly demonstrates the feasibility of the proposed physics injection approach for the projection-based reduced-order models.

Furthermore, we reiterate here that GROM features can be embedded at any layer of the neural network or perhaps at all hidden layers. We investigate the effect of embedding GROM features at the specific layer of the neural network and present the quantitative results in Table III for $\gamma = 0.01$ at different Reynolds numbers. We notice that the ensemble-averaged (i.e., $N_e = 30$ ensembles from different initialization) WRMSE and the uncertainty estimate is minimum at all Reynolds numbers when the GROM features are embedded at each of the hidden layers of the neural network. Although our systematic comparison in Table III highlights the underlying trends, the optimal neural network architecture for the PGML framework can be discovered using techniques like AutoML[68] and we consider this as part of our future work. Moreover, it is also worth mentioning that we intentionally use the same architecture in our comparisons between the ML and PGML models throughout our study. Intuitively, a PGML model might result in a simpler (fewer hyperparameters) model than a physics-agnostic ML model.

TABLE III. The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. (48) in predicting the modal coefficients for the vortex merging flows in the case of $\gamma = 0.01$ at different Reynolds numbers. The training has been performed using data for $\mathrm{Re} = \{200, 400, 600, 800\}$, and the physics-based features are embedded at different intermediate layer of the neural network. For example, in PGML-1 model, the GROM features are embedded at the first hidden layer. For the PGML-A model, GROM features are embedded at each hidden layer of the neural network.

Model	Re = 1000	Re = 1500	Re = 2000	Re = 3000
PGML-1	0.108 (0.018)	0.259 (0.030)	0.446 (0.071)	1.119 (0.507)
PGML-2	0.114 (0.015)	0.286 (0.023)	0.450 (0.055)	0.774 (0.157)
PGML-3	0.110 (0.012)	0.300 (0.029)	0.503 (0.086)	0.867 (0.219)
PGML-A	0.108 (0.012)	0.267 (0.033)	0.397 (0.057)	0.596 (0.093)

V. DISCUSSION AND CONCLUSION

Although advanced machine learning (ML) models like deep learning networks are powerful tools for finding patterns in complicated datasets, the number of trainable parameters (weights) quickly explodes as these neural network models become complex, adversely affecting their interpretability and hence their trustworthiness. Our chief aim in this study is to illustrate how physics can be injected into these neural networks to improve the trustworthiness of the overall system. With this in mind, we introduce a physics-guided ML (PGML) framework that fuses first-principles with neural-network models, and explore complementary physics vs statistics issues that arise in the development of the PGML framework. The PGML framework puts particular emphasis on the physics-based features and embeds them at an intermediate layer of the neural network. The robustness of the PGML framework is successfully demonstrated for the vortex-merging process in the presence of an array of Taylor–Green vortices as the unknown source term. Our results indicate that the PGML framework can give considerably accurate prediction in the extrapolation regime compared to its counterpart, that is, pure data-driven ML model. The physics-based features from the GP model also ensure that the model uncertainty in the neural network prediction is reduced. The PGML framework is achievable very accurate prediction with a very small uncertainty at Reynolds number substantially different from the training. Finally, we also emphasize that the PGML framework is highly modular, and it naturally allows for the multi-fidelity model fusion in different branches of science and engineering.

The natural extension of the present study is to assess the PGML framework for much more complex flows, where there is a significant variation between the train and test data. Other unanswered questions are understanding how neural network assigns importance to

physics-based features, at which layer shall the physics-based features be embedded, and methods from machine learning community can be utilized for this (e.g. Refs. 69 and 70). Another direction for future work will be to use other uncertainty quantification methods like Monte Carlo dropouts and Bayesian neural networks to quantify model uncertainty.

Moreover, the PGML approach might lead to modular data-driven workflows across multiple scientific domains. Such design of a hierarchically sequential learning algorithm allows the embedding of simplified theories, possible symmetries, space–time correlations, scaling laws, and other physics-based kernels, multi-modal data sources or features directly into deep neural network models. These physics-based features often assist the deep neural network models in constraining the output to a manifold of the physically realizable solution. Consequently, this concatenated neural network design can lead to improved generalizability for the extrapolation regime beyond the training datasets, as illustrated in this paper in the context of projection-based model reduction. In our future studies, we will also exploit how to develop an automatic-PGML framework to determine the optimal architecture for multifidelity physics injection.

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. 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.

DATA AVAILABILITY

The data that support the findings of this study are available within the article. The open-source implementation is available at https://github.com/surajp92/PGML_ROM.

REFERENCES

[1]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).
[2]M. Brenner, J. Eldredge, and J. Freund, "Perspective on machine learning for advancing fluid mechanics," Phys. Rev. Fluids **4**, 100501 (2019).
[3]K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks **2**, 359–366 (1989).
[4]K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357–377 (2019).
[5]K. Fukami, K. Fukagata, and K. Taira, "Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows," J. Fluid Mech. **909**, A9 (2021).
[6]N. J. Nair and A. Goza, "Leveraging reduced-order models for state estimation using deep learning," J. Fluid Mech. **897**, R1 (2020).
[7]K. Lee and K. T. Carlberg, "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," J. Comput. Phys. **404**, 108973 (2020).
[8]A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, "Theory-guided data science: A new paradigm for scientific discovery from data," IEEE Trans. Knowl. Data Eng. **29**, 2318–2331 (2017).
[9]M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais *et al.*, "Deep learning and process understanding for data-driven Earth system science," Nature **566**, 195–204 (2019).
[10]A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," IEEE Access **8**, 21980–22012 (2020).
[11]K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh *et al.*, "Physics-informed machine learning: Case studies for weather and climate modelling," Philos. Trans. R. Soc. A **379**, 20200093 (2021).
[12]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[13]J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao *et al.*, "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," J. Comput. Phys. **406**, 109209 (2020).
[14]N. Geneva and N. Zabaras, "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks," J. Comput. Phys. **403**, 109056 (2020).
[15]T. Beucler, S. Rasp, M. Pritchard, and P. Gentine, "Achieving conservation of energy in neural network emulators for climate modeling," arXiv:1906.06622 (2019).
[16]A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding hard physical constraints in neural network coarse-graining of 3d turbulence," arXiv:2002.00021 (2020).
[17]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).
[18]L. Zanna and T. Bolton, "Data-driven equation discovery of ocean mesoscale closures," Geophys. Res. Lett. **47**, e2020GL088376 (2020), https://doi.org/10.1029/2020GL088376.
[19]S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," arXiv:1906.01563 (2019).
[20]C. W. Rowley and S. T. Dawson, "Model reduction for flow analysis and control," Annu. Rev. Fluid Mech. **49**, 387–417 (2017).
[21]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).
[22]K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. Dawson, and C.-A. Yeh, "Modal analysis of fluid flows: Applications and outlook," AIAA J. **58**, 998 (2020).
[23]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).
[24]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).
[25]M. Meldi and A. Poux, "A reduced order model based on Kalman filtering for sequential data assimilation of turbulent flows," J. Comput. Phys. **347**, 207–234 (2017).
[26]S. Guzzetti, L. M. Alvarez, P. Blanco, K. T. Carlberg, and A. Veneziani, "Propagating uncertainties in large-scale hemodynamics models via network uncertainty quantification and reduced-order modeling," Comput. Methods Appl. Mech. Eng. **358**, 112626 (2020).
[27]C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," Automatica **25**, 335–348 (1989).
[28]S. Hovland, J. T. Gravdahl, and K. E. Willcox, "Explicit model predictive control for large-scale systems via model reduction," J. Guidance Control Dyn. **31**, 918–926 (2008).

[29]P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry* (Cambridge University Press, New York, 1998).
[30]J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).
[31]D. Rempfer, "On low-dimensional Galerkin models for fluid flow," Theor. Comput. Fluid Dyn. **14**, 75–88 (2000).
[32]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).
[33]V. M. Krasnopolsky and M. S. Fox-Rabinovitz, "Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction," Neural Networks **19**, 122–134 (2006).
[34]J. Yu, C. Yan, and M. Guo, "Non-intrusive reduced-order modeling for fluid problems: A brief review," Proc. Inst. Mech. Eng., Part G: J. Aerosp. Eng. **233**, 5896–5912 (2019).
[35]F. J. Gonzalez and M. Balajewicz, "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems," arXiv:1808.01346 (2018).
[36]R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders," Phys. Fluids **33**, 037106 (2021).
[37]J. Xu and K. Duraisamy, "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics," Comput. Methods Appl. Mech. Eng. **372**, 113379 (2020).
[38]S. M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu, "Nonintrusive reduced order modeling framework for quasigeostrophic turbulence," Phys. Rev. E **100**, 053306 (2019).
[39]Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo, "Model identification of reduced order fluid dynamics systems using deep learning," Int. J. Numer. Methods Fluids **86**, 255–268 (2018).
[40]M. Guo and J. S. Hesthaven, "Data-driven reduced order modeling for time-dependent problems," Comput. Methods Appl. Mech. Eng. **345**, 75–99 (2019).
[41]H. F. S. Lui and W. R. Wolf, "Construction of reduced-order models for fluid flows using deep feedforward neural networks," J. Fluid Mech. **872**, 963–994 (2019).
[42]Q. Wang, N. Ripamonti, and J. S. Hesthaven, "Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism," *Journal of Computational Physics* (Elsevier, 2020), pp. 109402.
[43]S. Ba and V. R. Joseph, "Composite Gaussian process models for emulating expensive functions," Ann. Appl. Stat. **6**, 1838–1860 (2012).
[44]X. Deng, C. D. Lin, K.-W. Liu, and R. Rowe, "Additive Gaussian process for computer models with qualitative and quantitative factors," Technometrics **59**, 283–292 (2017).
[45]C. Liu, T. Liu, J. Du, Y. Zhang, X. Lai, and J. Shi, "Hybrid nonlinear variation modeling of compliant metal plate assemblies considering welding shrinkage and angular distortion," J. Manuf. Sci. Eng. **142**, 041003 (2020).
[46]C. B. Davis, C. M. Hans, and T. J. Santner, "Prediction of non-stationary response functions using a Bayesian composite Gaussian process," Comput. Stat. Data Anal. **154**, 107083 (2021).
[47]R. B. Gramacy and H. K. Lee, "Cases for the nugget in modeling computer experiments," Stat. Comput. **22**, 713–722 (2012).
[48]B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," SIAM Rev. **60**, 550–591 (2018).
[49]P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis, "Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling," Proc. R. Soc. A: Math. Phys. Eng. Sci. **473**, 20160751 (2017).
[50]L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent structures," Q. Appl. Math. **45**, 561–571 (1987).
[51]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).
[52]D. Amsallem and C. Farhat, "Interpolation method for adapting reduced-order models and application to aeroelasticity," AIAA J. **46**, 1803–1813 (2008).
[53]R. Zimmermann, B. Peherstorfer, and K. Willcox, "Geometric subspace updates with applications to online adaptive nonlinear model reduction," SIAM J. Matrix Anal. Appl. **39**, 234–261 (2018).
[54]S. Pawar, O. San, B. Aksoylu, A. Rasheed, and T. Kvamsdal, "Physics guided machine learning using simplified theories," Phys. Fluids **33**, 011701 (2021).
[55]A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," in *Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)* (IEEE, 2019), pp. 1471–1479.
[56]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).
[57]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
[58]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: Math. Phys. Eng. Sci. **474**, 20170844 (2018).
[59]R. Maulik, B. Lusch, and P. Balaprakash, "Non-autoregressive time-series methods for stable parametric reduced-order models," Phys. Fluids **32**, 087115 (2020).
[60]P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Networks **126**, 191 (2020).
[61]K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata, "CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers," Fluid Dyn. Res. **52**, 065501 (2020).
[62]H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," Phys. Fluids **32**, 105104 (2020).
[63]O. San and R. Maulik, "Neural network closures for nonlinear model order reduction," Adv. Comput. Math. **44**, 1717–1750 (2018).
[64]R. Maulik, R. Egele, B. Lusch, and P. Balaprakash, "Recurrent neural network architecture search for geophysical emulation," arXiv:2004.10928 (2020).
[65]R. Tibshirani, "A comparison of some error estimates for neural network models," Neural Comput. **8**, 152–163 (1996).
[66]T. Heskes, "Practical confidence and prediction intervals," in *Advances in Neural Information Processing Systems* (NIPS, 1997), pp. 176–182.
[67]B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in Proceedings of the 31st International Conference on Neural Information Processing Systems (*NIPS*) (Curran Associates, Inc., Red Hook, NY, 2017), pp. 6405–6416.
[68]X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," Knowl.-Based Syst. **212**, 106622 (2021).
[69]H. Robinson, A. Rasheed, and O. San, "Dissecting deep neural networks," arXiv:1910.03879 (2019).
[70]G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Springer, Berlin, 2019), pp. 193–209.