Journal of Computational Physics 446 (2021) 110666

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics-informed machine learning for reduced-order modeling of nonlinear problems

Wenqian Chen [a,b], Qian Wang [b,*], Jan S. Hesthaven [b], Chuhua Zhang [a]

[a] Department of Fluid Machinery and Engineering, School of Energy and Power Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China

[b] Chair of Computational Mathematics and Simulation Science, École polytechnique fédérale de Lausanne, 1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Available online 27 August 2021

Keywords:
Physics-informed machine learning
Feedforward neural network
Reduced-order modeling
Nonlinear PDE

ABSTRACT

A reduced basis method based on a physics-informed machine learning framework is developed for efficient reduced-order modeling of parametrized partial differential equations (PDEs). A feedforward neural network is used to approximate the mapping from the time-parameter to the reduced coefficients. During the offline stage, the network is trained by minimizing the weighted sum of the residual loss of the reduced-order equations, and the data loss of the labeled reduced coefficients that are obtained via the projection of high-fidelity snapshots onto the reduced space. Such a network is referred to as physics-reinforced neural network (PRNN). As the number of residual points in time-parameter space can be very large, an accurate network – referred to as physics-informed neural network (PINN) – can be trained by minimizing only the residual loss. However, for complex nonlinear problems, the solution of the reduced-order equation is less accurate than the projection of high-fidelity solution onto the reduced space. Therefore, the PRNN trained with the snapshot data is expected to have higher accuracy than the PINN. Numerical results demonstrate that the PRNN is more accurate than the PINN and a purely data-driven neural network for complex problems. During the reduced basis refinement, the PRNN may obtain higher accuracy than the direct reduced-order model based on a Galerkin projection. The online evaluation of PINN/PRNN is orders of magnitude faster than that of the Galerkin reduced-order model.

1. Introduction

High-fidelity numerical simulation has been widely used in applications modeled by parametrized partial differential equations (PDEs) [1,2], in which the parameters characterize geometric features, boundary conditions, source terms and physical properties, etc. However, for applications like design, control, optimization and uncertainty quantification, all of which require repeated model evaluations over a potentially large range of parameter values [3], high-fidelity simulation remains prohibitively expensive. The need of cost reduction in such applications has led to the development of reduced-order modeling (ROM) [4] that seeks to build low-dimensional models for rapid and reliable solutions.

* Corresponding author.
E-mail addresses: wenqianchen2016@gmail.com (W. Chen), qian.wang@epfl.ch (Q. Wang), Jan.Hesthaven@epfl.ch (J.S. Hesthaven), chzhang@mail.xjtu.edu.cn (C. Zhang).

https://doi.org/10.1016/j.jcp.2021.110666

Reduced basis (RB) methods are a class of well-known and widely-used ROM techniques. RB methods are generally implemented in an offline-online paradigm [5]. During the offline stage, a reduced basis, which represents the main dynamics of the underlying problem, is extracted from a collection of high-fidelity solutions (snapshots). The reduced basis functions can be obtained by a variety of algorithms, such as proper orthogonal decomposition (POD) [1,2,6], the proper generalized decomposition [7], the piece-wise tangential interpolation [8], the matrix interpolation [9] and greedy algorithms [10–12]. Recently, autoencoder [13–15] has been used as a reduction tool on nonlinear manifolds. Once the reduced basis functions are computed, a reduced-order model can be constructed in either an intrusive or a non-intrusive manner. During the online stage, for a new parameter location, the corresponding reduced-basis solution is recovered as a linear combination of the RB functions, with the expansion coefficients (also called the reduced coefficients) computed by evaluation of the reduced-order model.

Intrusive reduced-order model is generally obtained by the projection of the full-order model onto the reduced space spanned by the RB functions. A Galerkin projection [16–18], in which the RB basis is used as the test functions, is the simplest and most popular choice. The popular method that uses POD to generate reduced basis and a Galerkin procedure to build reduced-order model is referred to as POD-G in this paper. During the online stage, the reduced coefficients are determined by solving the reduced-order equations. The POD-G method has been applied successfully to a variety of problems. However, the POD-G lacks *a priori* guarantees of stability, accuracy and convergence for complex nonlinear problems [16,19,20], and the promise of accelerated solution is difficult to realize for general nonlinear problems. Extensive efforts have been made to improve the stability and accuracy of the POD-G method, such as the structure-preservation [21], supremizer enrichment [22,23], basis adaption [24,25], L^1-norm minimization [26], and least-squares Petrov-Galerkin (LSPG) technique [27,28]. Closure modeling based on the Mori-Zwanzig formalism [29–32] is an alternative approach to improve the stability and accuracy of projection-based model order reduction. The short-memory assumption based models, such as the t-model [32] and the τ-model [33,34], have been applied to model order reduction for nonlinear problems. San and Maulik [35,36] used the feedforward neural network to model the closure term of the POD-G reduced-order model. Wang et al. [17] used recurrent neural network (RNN) for memory closure modeling of parametric POD-G reduced-order model.

Non-intrusive reduced-order model generally predicts reduced coefficients for a new time-parameter value by interpolation or regression, based on a data base of reduced-order information [37]. Standard interpolation techniques may fail if only a small number of samples are available [38,39]. Regression-based non-intrusive RB methods [40–44] have recently been developed. Regression models, such as the artificial neural network (ANN) and Gaussian process (GP), are approximate maps between the time-parameter value and the projection coefficients of the high-fidelity solution onto the reduced space, and are trained by high-fidelity data in a supervised learning paradigm [45] during the offline stage. During the online stage, the reduced coefficients are determined by rapid evaluation of the regression map. Recently, non-intrusive reduced-order methods that learn the dynamics of the underlying problems in the reduced space have been developed. Srinivasan et al. [46] used a long short-term memory (LSTM) network to predict temporal dynamics of simple turbulent flows in Fourier space. Gonzalez and Balajewicz [47] and Hasegawa et al. [48] used an autoencoder for nonlinear basis generation and LSTM network to dynamically evolve the reduced solution in time. Swischuk et al. [49] relied on operator inference model for the complex rocket combustion instability problem.

The advantages of the non-intrusive over the intrusive methods are *robustness* and *efficiency*. The robustness is due to the decoupling of the online stage and the high-fidelity scheme in the non-intrusive method. The efficiency is due to the fact that in the online stage, the reduced coefficients of the non-intrusive method are determined by evaluating the regression model, while reduced coefficients of the intrusive method are determined by solving a nonlinear equation system [40]. The major disadvantage of the non-intrusive over intrusive methods is the *offline cost*. Since a big data set, obtained by projecting a large amount of expensive high-fidelity snapshots onto the reduced space, is necessary to train an accurate regression model.

A reduced basis method based on physics-informed machine learning [50–53] is proposed in this work to combine the advantages of intrusive and non-intrusive methods. As aforementioned, the intrusive reduced-order model relies on physics, and the non-intrusive reduced-order model relies on data. In the physics-informed machine learning based RB method, both physics and data are used to train a network that maps the time-parameter value to the reduced coefficients. The loss function for the network training is the weighted sum of the mean squared norm of the residual of the POD-G equation, and the mean squared error of the labeled reduced coefficients that are obtained via the projection of the high-fidelity snapshots onto the reduced space. The network is thus trained to fulfill the POD-G equation and fit the projection data. Such a network is referred to as physics-reinforced neural network (PRNN). In this work, the high-fidelity snapshots are only those used to extract the reduced basis. As the corresponding high-fidelity snapshots on the residual points are not needed, the number of residual points in the time-parameter space is practically unlimited. Therefore, even with a small amount of high-fidelity snapshots, a sufficiently large training data set can be generated to train an accurate PRNN. We also show in this work that an accurate network - referred to as physics-informed neural network (PINN) - can be trained by minimizing only the residual loss. However, for complex nonlinear problems, the projection of the high-fidelity solution onto the reduced space is more accurate than the solution of the reduced-order equation, since the reduced-order equation does not take into account the impact of the unresolved scales (truncated modes) on the resolved scales (reduced basis modes) [17]. Therefore, the use of snapshot data is not mandatory during the offline training, but important for the network accuracy. The PRNN is expected to be more accurate than the PINN for complex nonlinear problems. By using the physics-

informed machine learning technique, we train neural networks to accurately and efficiently predict the reduced-order solution, without requiring extra high-fidelity simulation.

For reduced-order modeling of long-term evolution problems, a significant advantage of the non-intrusive methods over the intrusive methods is the local modeling capability. The modeling can be limited to a time interval of interest, which drastically reduces the modeling complexity. The physics-informed machine learning based RB method inherits the local modeling capability from the non-intrusive RB methods.

Several time-dependent and stationary problems are used to test the proposed reduced-order methods. Numerical results demonstrate that the PRNN can predict reliable reduced-order solutions and is more accurate than the PINN or a purely data-driven neural network. During the reduced basis refinement, the PRNN may obtain higher accuracy than the direct reduced-order model based on the Galerkin projection. It is also demonstrated that the PRNN/PINN is orders of magnitude faster than the POD-G during the online stage.

The remainder of the paper is organized as follows. Section 2 presents the model order reduction framework of parametrized PDEs. Section 3 discusses the physics-informed machine learning of the reduced-order model, while Section 4 presents the numerical results. Section 5 gathers the relevant conclusions.

2. Projection-based reduced basis method

Consider a nonlinear dynamical system governed by the parametrized partial differential equations (PDEs):

$$\begin{aligned} \frac{\partial \phi(\mathbf{x},t)}{\partial t} + \mathcal{N}(\phi(\mathbf{x},t);\boldsymbol{\mu}) &= 0, \qquad (\mathbf{x},t) \in \Omega(\boldsymbol{\mu}) \times \mathcal{T} \\ \mathcal{B}(\phi(\mathbf{x},t);\boldsymbol{\mu}) &= 0, \qquad (\mathbf{x},t) \in \partial\Omega(\boldsymbol{\mu}) \times \mathcal{T} \\ \phi(\mathbf{x},0) &= \phi_0(\mathbf{x};\boldsymbol{\mu}), \qquad \mathbf{x} \in \Omega(\boldsymbol{\mu}) \end{aligned} \tag{1}$$

where $\mathcal{N}$ is a general nonlinear differential operator and $\phi(\mathbf{x},t)$ are field variables. $\mathcal{B}$ is a boundary condition operator defined on the boundary $\partial\Omega$ of the spatial domain Ω. $\mathcal{T}$ is the temporal domain. $\boldsymbol{\mu} \in \mathcal{P}$ are the physical/geometrical parameters and $\mathcal{P} \subset \mathbb{R}^{n_p}$ is the parameter space. n_p is the number of parameters, i.e., the dimension of $\mathcal{P}$.

Snapshots used for generation of the reduced basis functions are obtained from high-fidelity (HF) simulations using the same amount of degree of freedoms (DOFs) on a set of parameter values. Physical coordinates of DOFs are variable when the shape or size of the physical domain Ω is controlled by geometrical parameters. Therefore, to ensure compatibility of the high-fidelity snapshots, we perform the simulations on a computational domain $\widetilde{\Omega}$. The invertible and differentiable problem-dependent mapping $\mathcal{X}: \mathbf{x} \in \Omega(\boldsymbol{\mu}) \rightarrow \boldsymbol{\xi} \in \widetilde{\Omega}$ from physical to computational coordinates is

$$\boldsymbol{\xi} = \mathcal{X}(\mathbf{x};\boldsymbol{\mu}), \tag{2}$$

and (1) becomes

$$\begin{aligned} \frac{\partial \phi(\boldsymbol{\xi},t)}{\partial t} + \widetilde{\mathcal{N}}(\phi(\boldsymbol{\xi},t);\boldsymbol{\mu}) &= 0, \qquad (\boldsymbol{\xi},t) \in \widetilde{\Omega} \times \mathcal{T} \\ \widetilde{\mathcal{B}}(\phi(\boldsymbol{\xi},t);\boldsymbol{\mu}) &= 0, \qquad (\boldsymbol{\xi},t) \in \partial\widetilde{\Omega} \times \mathcal{T} \\ \phi(\boldsymbol{\xi},0) &= \phi_0\left(\mathcal{X}^{-1}(\boldsymbol{\xi};\boldsymbol{\mu});\boldsymbol{\mu}\right), \qquad \boldsymbol{\xi} \in \widetilde{\Omega} \end{aligned} \tag{3}$$

where $\widetilde{\mathcal{N}}$ and $\widetilde{\mathcal{B}}$ are the corresponding operators in the computational space.

2.1. Full-order model

A full-order model is obtained by the discretization of (3) using a high-fidelity scheme on a fine mesh. The number of interior DOFs is denoted as N, the number of boundary DOFs is denoted as N_B. We denote the discrete solutions of (3) as $\boldsymbol{\phi}_h \in \mathbb{R}^N$ and $\boldsymbol{\phi}_h^B \in \mathbb{R}^{N_B}$. The full-order model is expressed as

$$\frac{d\boldsymbol{\phi}_h}{dt} + \mathbf{L}_{\widetilde{\mathcal{N}}}\boldsymbol{\phi}_h + g_{\widetilde{\mathcal{N}}}(\boldsymbol{\phi}_h, \boldsymbol{\phi}_h^B) + \mathbf{C}_{\widetilde{\mathcal{N}}} = 0, \tag{4}$$

$$\mathbf{L}_{\widetilde{\mathcal{B}}}\boldsymbol{\phi}_h + \mathbf{L}_{\widetilde{\mathcal{B}}}^B\boldsymbol{\phi}_h^B + \mathbf{C}_{\widetilde{\mathcal{B}}} = 0, \tag{5}$$

where $\mathbf{L}_{\widetilde{\mathcal{N}}} \in \mathbb{R}^{N\times N}$, $\mathbf{L}_{\widetilde{\mathcal{B}}} \in \mathbb{R}^{N_B\times N}$ and $\mathbf{L}_{\widetilde{\mathcal{B}}}^B \in \mathbb{R}^{N_B\times N_B}$ represent matrices derived from the linear parts of the operators $\widetilde{\mathcal{N}}$ and $\widetilde{\mathcal{B}}$. $\mathbf{C}_{\widetilde{\mathcal{N}}} \in \mathbb{R}^N$ and $\mathbf{C}_{\widetilde{\mathcal{B}}} \in \mathbb{R}^{N_B}$ are constant vectors. $g_{\widetilde{\mathcal{N}}}: \mathbb{R}^N \times \mathbb{R}^{N_B} \mapsto \mathbb{R}^N$ is a nonlinear function derived from the nonlinear part of operator $\widetilde{\mathcal{N}}$. Discretization of the boundary conditions results in a linear equation system (5), as commonly used boundary conditions, e.g., Dirichlet, Neumann and Robin, are linear and thus $\mathcal{B}$ is a linear operator.

A broad range of discretization procedures can be used to build the full-order model, such as finite difference (FD), finite volume (FV), finite element (FE), and spectral methods. In this work, a high-order accurate Chebyshev pseudospectral method [54,55] is used for the spatial discretization, and a third-order backward difference method is used for the temporal discretization.

2.1.1. Chebyshev pseudospectral method

In the Chebyshev pseudospectral method, a d-dimensional physical domain is mapped to the computational domain $\widetilde{\Omega} = [-1, 1]^d$, with each dimension discretized by $N_p + 1$ Chebyshev-Gauss-Lobatto points,

$$\xi_i = \cos\left(\frac{\pi i}{N_p}\right), \quad 0 \le i \le N_p. \tag{6}$$

The solution is approximated by $\phi(\xi) = \sum_{i=0}^{N_p} \phi_i \ell_i(\xi)$, where $\ell_i(\xi)$ is the Lagrange polynomial. Spatial derivatives are discretized as

$$\left[\frac{\partial^s \phi}{\partial \xi^s}\Big|_{\xi_0}, \frac{\partial^s \phi}{\partial \xi^s}\Big|_{\xi_1}, \cdots, \frac{\partial^s \phi}{\partial \xi^s}\Big|_{\xi_{N_p}}\right]^T = \mathbf{D}^s \boldsymbol{\phi}, \tag{7}$$

where $\boldsymbol{\phi} = \{\phi_i\}_{i=0}^{N_p}$, s is the order of the derivative and $\mathbf{D}^s$ is the s-th-order difference matrix of size $(N_p + 1) \times (N_p + 1)$ with elements defined by

$$D^0_{i,j} = \delta_{ij}, \qquad 0 \le i, j \le N_p, \tag{8}$$

$$\begin{cases} D^1_{i,j} = \dfrac{B_i}{B_j} \dfrac{(-1)^{i+n}}{2\sin\left(\frac{(i+j)\pi}{2N_p}\right)\sin\left(\frac{(j-i)\pi}{2N_p}\right)} & 0 \le i, j \le N_p, i \ne j \\ D^1_{i,i} = -\displaystyle\sum_{j=0, j\ne i}^{N_p} D^1_{i,j} \quad 1 \le i \le N_p - 1 \\ D^1_{0,0} = -D^1_{N_p,N_p} = -\dfrac{2{N_p}^2 + 1}{6} \\ B_i = \begin{cases} 2 & i = 0, N_p \\ 1 & 1 \le i \le N_p - 1 \end{cases} \end{cases}, \tag{9}$$

$$D^s_{i,j} = D^1_{i,k} D^{s-1}_{k,j} \qquad 0 \le i, j, k \le N_p. \tag{10}$$

In the flow simulations in this paper, the $IP_N - IP_{N-2}$ method [56] is adopted to prevent spurious pressure mode [57]. In the $IP_N - IP_{N-2}$ method, the pressure derivative is approximated without pressure values on boundary points, thus pressure is approximated with a polynomial of two order lower than other flow variables. The first-order difference matrix for the pressure is $\hat{\mathbf{D}}^1$ defined as

$$\begin{cases} \hat{D}^1_{i,j} = 0 & i = 0, N_p \text{ or } j = 0, N_p \\ \hat{D}^1_{i,i} = \dfrac{3\xi_i}{2(1-\xi_i^2)} & 1 \le i \le N_p - 1 \\ \hat{D}^1_{i,j} = \dfrac{(-1)^{i+j}(1-\xi_j^2)}{2(1-\xi_i^2)(\xi_i - \xi_j)} & 1 \le i \ne j \le N_p - 1 \end{cases}. \tag{11}$$

For more details of the Chebyshev pseudospectral method, we refer the reader to the reference [54,55,57].

2.2. Reduced-order model

The solution of the full-order model (3) is often expensive due to the large amount of DOFs. Therefore, we are interested in replacing the full-order model with a low-dimensional reduced-order model to efficiently resolve the main dynamics of the underlying problem. The full-order model is said to be reducible if the high-fidelity solution $\boldsymbol{\phi}_h$ can be well approximated in an m-dimensional subspace $\mathcal{V}$. The reducibility can be quantified by the Kolmogorov m-width [58], which represents the smallest approximation error that can be obtained with an m-dimensional subspace. The subspace is spanned by a suitable set of basis vectors $\{\mathbf{v}_i \in \mathbb{R}^N\}_{i=1}^m$. The size of the problem is significantly reduced if $m \ll N$. The full-order solution of the interior DOFs $\boldsymbol{\phi}_h$ can be approximated by its projection onto the subspace $\mathcal{V}$ as

$$\boldsymbol{\phi}_h \approx \widetilde{\boldsymbol{\phi}} + \mathbf{V}\boldsymbol{\alpha}, \tag{12}$$

where $\mathbf{V} \in \mathbb{R}^{N \times m}$ is a matrix with the basis vectors as columns and $\boldsymbol{\alpha}$ are the projection coefficients. The projection error is $\boldsymbol{\epsilon} = \boldsymbol{\phi}_h - \widetilde{\boldsymbol{\phi}} - \mathbf{V}\boldsymbol{\alpha}$. $\widetilde{\boldsymbol{\phi}} \in \mathbb{R}^N$ is a time-independent reference value used to avoid a possible dominant constant basis, and is usually set as the mean of the snapshots or the initial solution. As shown in (5), the boundary DOFs are linearly related to the interior DOFs. Therefore, the boundary DOFs $\boldsymbol{\phi}_h^B$ can be treated as a function of the interior DOFs $\boldsymbol{\phi}_h$ as

$$\boldsymbol{\phi}_h^B = -(\mathbf{L}_{\widetilde{\mathcal{B}}}^B)^{-1}\left(\mathbf{L}_{\widetilde{\mathcal{B}}}\boldsymbol{\phi}_h + \mathbf{C}_{\widetilde{\mathcal{B}}}\right). \tag{13}$$

Substituting (12) and (13) into (4), we obtain the following over-determined system for the reduced coefficients $\boldsymbol{\alpha}$

$$\begin{aligned}\mathbf{V}\frac{d\boldsymbol{\alpha}}{dt} &+ \mathbf{L}_{\widetilde{\mathcal{N}}}\left(\widetilde{\boldsymbol{\phi}} + \mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{L}_{\widetilde{\mathcal{N}}}\boldsymbol{\epsilon} + \mathbf{C}_{\widetilde{\mathcal{N}}} \\ &+ \mathbf{g}_{\widetilde{\mathcal{N}}}\left(\widetilde{\boldsymbol{\phi}} + \mathbf{V}\boldsymbol{\alpha} + \boldsymbol{\epsilon}, -\left(\mathbf{L}_{\widetilde{\mathcal{B}}}^B\right)^{-1}\left(\mathbf{L}_{\widetilde{\mathcal{B}}}\left(\widetilde{\boldsymbol{\phi}} + \mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{C}_{\widetilde{\mathcal{B}}} + \mathbf{L}_{\widetilde{\mathcal{B}}}\boldsymbol{\epsilon}\right)\right) = 0,\end{aligned} \tag{14}$$

which can be simplified into a compact from

$$\mathbf{V}\frac{d\boldsymbol{\alpha}}{dt} + \mathbf{L}\mathbf{V}\boldsymbol{\alpha} + g\left(\mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{C} = \tilde{\boldsymbol{\epsilon}}, \tag{15}$$

where $\mathbf{L} \in \mathbb{R}^{N\times N}$, $\mathbf{C} \in \mathbb{R}^N$ and $g : \mathbb{R}^N \mapsto \mathbb{R}^N$ are the matrix, vector and nonlinear function derived from (14), respectively. The residual error $\tilde{\boldsymbol{\epsilon}}$ results from the projection error $\boldsymbol{\epsilon}$.

By performing the procedure of (12)–(15), we obtain a reduced-order model for the interior DOFs with enforcement of the boundary conditions.

In the framework of Petrov-Galerkin, by projecting the over-determined system (15) onto a subspace $\mathbf{W} \in \mathbb{R}^{N\times m}$ and requiring that the residual is orthogonal to the test space, we recover a system of m equations

$$\mathbf{W}^T\left(\mathbf{V}\frac{d\boldsymbol{\alpha}}{dt} + \mathbf{L}\mathbf{V}\boldsymbol{\alpha} + \mathbf{g}\left(\mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{C}\right) = \mathbf{0}. \tag{16}$$

In this work, we restrict ourselves to the Galerkin framework, namely $\mathbf{W} = \mathbf{V}$, reducing (16) to

$$\mathbf{V}^T\left(\mathbf{V}\frac{d\boldsymbol{\alpha}}{dt} + \mathbf{L}\mathbf{V}\boldsymbol{\alpha} + g\left(\mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{C}\right) = \mathbf{0}. \tag{17}$$

Once the reduced coefficients $\boldsymbol{\alpha}$ are obtained by solving the reduced-order equation system (17), the interior solution $\boldsymbol{\phi}_h$ and boundary solution $\boldsymbol{\phi}_h^B$ can be recovered by (12) and (13), respectively.

A key point of the reduced-order model is to get a suitable set of reduced basis functions. There exist several methods for reduced basis generation, such as the proper orthogonal decomposition [1,2,6], the proper generalized decomposition [7], the piece-wise tangential interpolation [8], the matrix interpolation [9], greedy algorithms [10–12] and an autoencoder [13–15]. The most widely-used reduced basis generation method, the proper orthogonal decomposition, is described and used in the rest of this paper.

2.2.1. Proper orthogonal decomposition

The proper orthogonal decomposition (POD) is one of the most widely-used techniques to compress data and extract an optimal set of orthonormal basis in the least-squares sense. Let $\mathcal{P}_h = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_{N_s}\} \subset \mathcal{P}$ be a discrete and finite set of N_s parameter values generated with a suitable sampling, $\mathcal{T}_h = \{t_1, t_2, \cdots, t_{N_t}\} \subset \mathcal{T}$ be a discrete and finite set of N_t time values, and $\boldsymbol{\Phi}_h = \{\boldsymbol{\phi}_h(t_1;\boldsymbol{\mu}_1) - \widetilde{\boldsymbol{\phi}}, \boldsymbol{\phi}_h(t_2;\boldsymbol{\mu}_1) - \widetilde{\boldsymbol{\phi}}, \cdots, \boldsymbol{\phi}_h(t_{N_t};\boldsymbol{\mu}_{N_s}) - \widetilde{\boldsymbol{\phi}}\} \in \mathbb{R}^{N\times N_s N_t}$ be the corresponding snapshot matrix. The POD basis of size m is the solution to the minimization problem

$$\begin{aligned}&\min_{\mathbf{V}\in\mathbb{R}^{N\times m}} \left\|\boldsymbol{\Phi}_h - \mathbf{V}\mathbf{V}^T\boldsymbol{\Phi}_h\right\|_F, \\ &s.t. \qquad \mathbf{V}^T\mathbf{V} = \mathbf{I},\end{aligned} \tag{18}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbf{I} \in \mathbb{R}^{m\times m}$ is the identity matrix. According to the Eckart-Young theorem [59], the solution of (18) is the first m left singular vectors of the matrix $\boldsymbol{\Phi}_h$, which can be obtained from the singular value decomposition (SVD)

$$\boldsymbol{\Phi}_h = \mathbf{U}_S\boldsymbol{\Sigma}_S\mathbf{V}_S, \qquad \boldsymbol{\Sigma}_S = \mathrm{diag}\left(\sigma_i\right), \tag{19}$$

where the singular values σ_i, $1 \le i \le \min(N, N_s N_t)$ are sorted in descending order. Choosing the first m columns of $\mathbf{U}_S$ as the POD basis functions, we have an error estimation for (18) as

$$\min_{\mathbf{V}\in\mathbb{R}^{N\times m}} \left\|\boldsymbol{\Phi}_h - \mathbf{V}\mathbf{V}^T\boldsymbol{\Phi}_h\right\|_F^2 = \sum_{i=m+1}^{\min(N,N_s N_t)} \sigma_i^2. \tag{20}$$

For (20), it is clear that the error in the POD basis is equal to the sum of the squares of the neglected singular values, i.e., by controlling the size m, we can approximate the snapshot matrix $\boldsymbol{\Phi}_h$ with arbitrary accuracy. Many problems exhibit an exponential decay of the singular values [2], allowing the use of a low-dimensional reduced space to approximate the high-fidelity solution with satisfactory accuracy.

For the time-dependent problems with many time steps, such as the unsteady natural convection problem in Section 4.2.3, the snapshot matrix is large, leading to an expensive SVD. The randomized SVD algorithm [60] which only needs to perform SVD of small matrices, can be used for efficient reduced basis generation of time-dependent problems with large snapshot matrices.

2.2.2. Treatment of nonlinear term

The cost of the reduced-order model (17) depends on the computational complexities of the four terms. The cost of the temporal term and the two linear terms scales with m, since they can be simplified as

$$\mathbf{V}^T\mathbf{V}\frac{d\boldsymbol{\alpha}}{dt} = \frac{d\boldsymbol{\alpha}}{dt},\ \mathbf{V}^T\mathbf{L}\mathbf{V}\boldsymbol{\alpha} = \tilde{\mathbf{L}}\boldsymbol{\alpha},\ \mathbf{V}^T\mathbf{C} = \tilde{\mathbf{C}}, \tag{21}$$

where the matrix $\tilde{\mathbf{L}} \in \mathbb{R}^{m\times m}$ and the vector $\tilde{\mathbf{C}} \in \mathbb{R}^m$ need to be computed only once.

The computational bottleneck is the computation of the nonlinear term $\mathbf{V}^T g(\mathbf{V}\boldsymbol{\alpha})$ that scales with the full-order size N, even though the dimension of the equation system has been reduced to $m \ll N$. A further cost reduction of the nonlinear term is necessary to obtain an efficient reduced-order model. Several such hyper-reduction methods have been developed to enable significant speedups for the nonlinear term computation, e.g., the empirical interpolation method (EIM) [61], its discrete variant (DEIM) [62], gappy-POD [63] and missing point estimation (MPE) [64]. These methods seek a trade-off between accuracy and efficiency. In the present work, we restrict ourselves to the case of quadratic nonlinearity, for which hyper-reduction is not needed since the nonlinear term can be transformed exactly into a quadratic form [65]. Consider a quadratic nonlinearity

$$g(\mathbf{V}\boldsymbol{\alpha}) = (\mathbf{V}\boldsymbol{\alpha}) \otimes (\mathbf{V}\boldsymbol{\alpha}) \tag{22}$$

where $\otimes$ denotes the element-wise multiplication operator. Substituting (22) into (17), the nonlinear term becomes

$$\mathbf{V}^T g(\mathbf{V}\boldsymbol{\alpha}) = \mathbf{V}^T\left((\mathbf{V}\boldsymbol{\alpha}) \otimes (\mathbf{V}\boldsymbol{\alpha})\right) \quad = \sum_{k=0}^{m}\left(\boldsymbol{\alpha}^T\mathbf{A}^k\boldsymbol{\alpha}\right)\mathbf{I}_k, \qquad 0 \le k \le m, \tag{23}$$

where $\mathbf{I}_k \in \mathbb{R}^m$ is k-th column of the identity matrix $\mathbf{I}$. $\mathbf{A}^k \in \mathbb{R}^{m\times m}$ is defined as

$$\mathbf{A}^k_{i,j} = \mathbf{v}_k^T\left(\mathbf{v}_i \otimes \mathbf{v}_j\right), \qquad 0 \le i, j, k \le m, \tag{24}$$

where $\mathbf{v}_l$ is the l-th column of $\mathbf{V}$.

Once the matrices $\mathbf{A}^k$ are computed and saved during the offline stage, the online evaluation of the nonlinear term has a computational complexity of $\mathcal{O}(m^3)$.

3. Physics-informed machine learning of reduced-order model

This section presents the physics-informed machine learning for reduced-order modeling. A POD-Galerkin reduced-order model can be built following the procedures in Section 2. However, the intrusive POD-G model suffers from stability, accuracy and convergence issues for complex nonlinear problems [16,19,20]. An alternative approach is to perform the non-intrusive reduced-order modeling based on regression [40]. The regression model approximates the mapping from the time-parameter to the projection coefficients of the underlying high-fidelity solution onto the reduced space. The neural network used as the regression model is referred to as projection driven neural network (PDNN) in this paper. The drawback of the non-intrusive reduced-order modeling is the high cost of the offline stage, since a large number of high-fidelity simulations are needed to generate a sufficiently large data set to train an accurate regression model.

A reduced-order modeling framework based on physics-informed machine learning [50–53] is proposed in this section to combine the advantages of both intrusive and non-intrusive methods. In this framework, the reduced coefficients are predicted by a feedforward neural network. The network can be trained by minimizing the mean squared norm of the residual of the reduced-order equation (17) on sampled training points in time-parameter space. By taking the residual of the reduced-order equation as the loss function, no labeled data is needed for the training. Such a network is referred to as the physics-informed neural network (PINN). However, the accuracy of PINN for complex nonlinear problem is often limited. To address this, the labeled data obtained by the projection of the high-fidelity snapshots that are used for basis generation, can be used to improve the accuracy. The usage of this data set does not require extra high-fidelity simulations. By using the labeled data, the loss function of the network becomes the weighted sum of the mean squared norm of the residual of the reduced-order equation and the mean squared error between the network output and the label. The network trained by this strategy is referred to as physics-reinforced neural network (PRNN). The framework of the physics-informed machine learning for reduced-order modeling is sketched in Fig. 1.

In the remainder of this section, the physics-informed machine learning framework, including the network construction, data preparation and training procedure will be presented in detail.

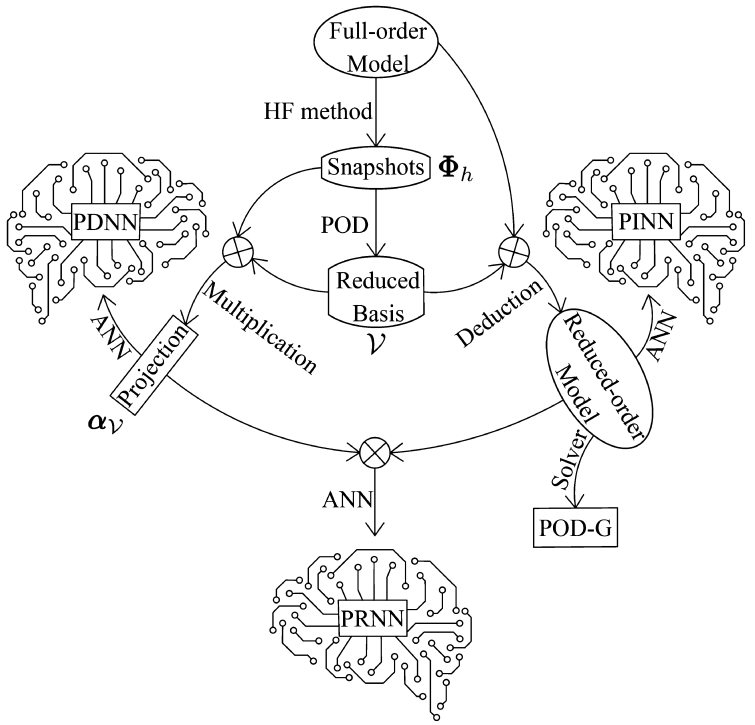

Fig. 1. Framework of physics-informed machine learning for reduced-order modeling.

3.1. Feedforward neural network

One of the most important issues for the design of an artificial neural network (ANN) is the architecture of the network. Several network architectures have been proposed, among which the feedforward neural network (FNN) is often preferred for function regression tasks.

A FNN maps the input $\mathbf{x}$ to the output $\mathbf{y}$. Generally, a FNN consists of an input layer, L hidden layers and an output layer. The l-th layer has n_l neurons, where $l = 0, 1, ..L, L+1$ denotes the input layer, first hidden layer,..., L-th hidden layer and the output layer, respectively. The forward propagation, i.e., the function $\mathbf{y} = f_{\text{nn}}(\mathbf{x})$ is

$$\begin{cases} \mathbf{y}^{(0)} = \mathbf{x} \\ \mathbf{y}^{(l)} = f_{\text{act}}\left(\mathbf{W}^{(l)}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \cdots, L, \\ \mathbf{y} = \mathbf{y}^{(L+1)} = \mathbf{W}^{(L+1)}\mathbf{y}^{(L)} + \mathbf{b}^{(L+1)} \end{cases} \tag{25}$$

where $\mathbf{y}^{(l)} \in \mathbb{R}^{n_l}$ is the output of l-th layer, n_0 is the input dimension and n_{L+1} is the output dimension, $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ are the weight matrices, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ are the biases and f_{act} is the nonlinear element-wise activation function.

In this work, the network is used to predict the reduced coefficients $\boldsymbol{\alpha}$, given the time-parameter value $(t, \boldsymbol{\mu})$. Therefore, the input dimension is $n_0 = n_p + 1$ and output dimension is $n_{L+1} = m$. The network structure is shown in Fig. 2. We put the same number of neurons in each hidden layer, namely $n_1 = n_2 = \cdots = n_L = n_H$. The nonlinear activation function is the Swish function $f_{\text{act}}(x) = x \cdot \text{sigmoid}(x)$ [66].

The network is trained, i.e., the weights and biases are optimized by minimizing a loss function. In the next two subsections, we will define several machine learning strategies by using different loss functions.

The training of the FNN is implemented in PyTorch [67]. The Adam optimizer [68] and the L-BFGS optimizer [69] are used in training. The Adam optimizer converges faster, while the L-BFGS optimizer generalizes better. Towards an efficient and accurate training procedure, the Adam optimizer is used in the first 90% of the epochs, and the L-BFGS optimizer is used for the remaining 10% of the epochs. Within a single epoch, the Adam optimizer uses mini-batches, while the L-BFGS optimizer uses full-batch of the training data. The training is performed for a sufficient number of epochs to obtain a converged network. The convergence speed of the Adam optimizer is controlled by the learning rate η.

3.2. Data preparation

In this subsection, we will present the procedure to generate the projection data set for the training process. The projection data is collected from the high-fidelity snapshots used to generate the reduced basis. As mentioned in Section 2, the

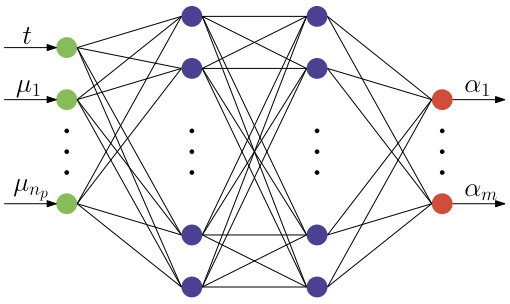

Fig. 2. A sample feedforward neural network structure with two hidden layers.

snapshots $\mathbf{\Phi}_h = \{\boldsymbol{\phi}_h(t_1;\boldsymbol{\mu}_1) - \widetilde{\boldsymbol{\phi}}, \boldsymbol{\phi}_h(t_2;\boldsymbol{\mu}_1) - \widetilde{\boldsymbol{\phi}}, ..., \boldsymbol{\phi}_h(t_{N_t};\boldsymbol{\mu}_{N_s}) - \widetilde{\boldsymbol{\phi}}\} \in \mathbb{R}^{N\times N_s N_t}$ are obtained on the sampled parameter set $\mathcal{P}_h = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_{N_s}\} \subset \mathcal{P}$ for N_t time steps.

The projection coefficients of a snapshot $\boldsymbol{\phi}_h(t;\boldsymbol{\mu})$ onto the reduced space $\mathcal{V}$ are

$$\boldsymbol{\alpha}_{\mathcal{V}}(t;\boldsymbol{\mu}) = \mathbf{V}^T\left(\boldsymbol{\phi}_h(t;\boldsymbol{\mu}) - \widetilde{\boldsymbol{\phi}}\right). \tag{26}$$

The projection is the best approximation of the high-fidelity solution in the reduced space. Therefore, for a time-parameter value $(t,\boldsymbol{\mu})$, the "best" output of the network is $\boldsymbol{\alpha}_{\mathcal{V}}(t;\boldsymbol{\mu})$. A data set $\mathcal{D}_{Pr} = \{(t_i,\boldsymbol{\mu}_i), \boldsymbol{\alpha}_{\mathcal{V}}(t_i;\boldsymbol{\mu}_i)\}_{i=1}^{N_s N_t}$ can be collected from the high-fidelity solutions.

The training speed of neural networks is affected by the range of the input/output [70], thus feature scaling needs to be performed before feeding the training data into the network. For the input $(t,\boldsymbol{\mu}) \in \mathcal{T}\times\mathcal{P}$, as the boundary of the time-parameter space is predefined, the minimum and maximum of the time and parameters can be used to scale the input to $[-1,1]^{n_p+1}$ as

$$\tilde{t} = \frac{t - (t_{\max}+t_{\min})/2}{(t_{\max}-t_{\min})/2}, \qquad \tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu} - (\boldsymbol{\mu}_{\max}+\boldsymbol{\mu}_{\min})/2}{(\boldsymbol{\mu}_{\max}-\boldsymbol{\mu}_{\min})/2}. \tag{27}$$

For the output $\boldsymbol{\alpha}$, there is no a predefined range. We turn to a statistical method to do the normalization using the standard score. To this end, the mean $\overline{\boldsymbol{\alpha}}$ and standard derivation $\boldsymbol{\sigma}_{\boldsymbol{\alpha}}$, computed from the data set $\mathcal{D}_{Pr}$, are used to normalize the output as

$$\tilde{\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha} - \overline{\boldsymbol{\alpha}}}{\boldsymbol{\sigma}_{\boldsymbol{\alpha}}}. \tag{28}$$

Feature scaling of the input/output can be viewed as a pre-processing, thus a corresponding post-processing is necessary to compute the final prediction

$$\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) = f_{\mathrm{nn}}\left(\tilde{t},\tilde{\boldsymbol{\mu}}\right)\otimes\boldsymbol{\sigma}_{\boldsymbol{\alpha}} + \overline{\boldsymbol{\alpha}}. \tag{29}$$

In this paper, the projection data set $\mathcal{D}_{Pr}$ is used as the training data set for the PDNN and part of the training data set for PRNN. Besides the training data set, a validation data set $\mathcal{D}_{val}$ and a test data set $\mathcal{D}_{te}$ are also required. The validation and test data sets are obtained following the same procedure as $\mathcal{D}_{Pr}$, on two independently sampled parameter sets.

3.3. Physics-informed machine learning

Once the data preparation is completed, we train a network to predict reduced coefficients using the data set $\mathcal{D}_{Pr}$, following the method proposed in [40,41]. As the network is trained on the data set collected by the projection of high-fidelity snapshots onto the reduced space, we refer to the network as projection driven neural network (PDNN). In the training procedure, $\mathcal{D}_{Pr}$ is employed as the training data set. The loss function is defined as the mean squared error (MSE)

$$\mathcal{L}_{\mathrm{PDNN}} = \frac{1}{N_{\mathcal{D}}}\sum_{(t,\boldsymbol{\mu}),\boldsymbol{\alpha}\in\mathcal{D}} \left\| f_{\mathrm{nn}}\left(\tilde{t},\tilde{\boldsymbol{\mu}}\right) - \tilde{\boldsymbol{\alpha}}_{\mathcal{V}}(t;\boldsymbol{\mu})\right\|_2^2, \tag{30}$$

where $\mathcal{D}$ is the chosen data set of size $N_{\mathcal{D}}$, which can be the training set $\mathcal{D}_{Pr}$ or the validation set $\mathcal{D}_{val}$.

In real applications, we do not expect to generate too many snapshots, since the high-fidelity simulation is expensive. Thus the data set $\mathcal{D}_{Pr}$ is typically small. To prevent over-fitting, L_2 weight regularization is employed as

$$\widetilde{\mathcal{L}}_{\mathrm{PDNN}} = \mathcal{L}_{\mathrm{PDNN}} + \lambda\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{31}$$

where λ is the parameter used to control the amount of regularization.

Algorithm 1 Training process of the PINN.

function $[\mathbf{W}_{tr}, \mathbf{b}_{tr}]$=PINN_TRAINING($\mathcal{D}_{Resi}, N_{epo}, N_{batch}, \eta_0$)
  $\mathbf{W}, \mathbf{b} \leftarrow$ INIT($\mathbf{W}, \mathbf{b}$) ▷ initialize weights and biases
  $N_b \leftarrow N_{Resi}/N_{batch}$ ▷ Mini-batch size
  for $epoch \leftarrow 1, 0.9 * N_{epo}$ **do** ▷ Adam training epoch loop
    $\eta \leftarrow$ LEARNING_RATE_DECAY($\eta_0, epoch$) ▷ learning rate decay
    $\mathcal{D}_{Resi} \leftarrow$ SHUFFLE($\mathcal{D}_{Resi}$) ▷ shuffle the training data set
    for $s \leftarrow 1, N_{batch}$ **do** ▷ mini-batch loop
      $\mathcal{D}_{Resi,s} \leftarrow \mathcal{D}_{Resi}[(s-1) * N_b + 1 : s * N_b]$ ▷ the s-th mini-batch
      $\mathcal{L} \leftarrow \frac{1}{N_b} \sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}_{Resi,s}} \left\| \frac{d\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})}{dt} + \mathbf{V}^T \left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) + g\left(\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})\right) + \mathbf{C}\right) \right\|_2^2$ ▷ compute loss
      $\Delta\mathbf{W} \leftarrow -\eta \mathrm{G}_{\mathrm{ADAM}}(\nabla_{\mathbf{W}}\mathcal{L}), \Delta\mathbf{b} \leftarrow -\eta \mathrm{G}_{\mathrm{ADAM}}(\nabla_{\mathbf{b}}\mathcal{L})$ ▷ Adam optimizer step
      $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}, \mathbf{b} \leftarrow \mathbf{b} + \Delta\mathbf{b}$ ▷ update weights and biases
    end for
  end for
  for $epoch \leftarrow 0.9 * N_{epo} + 1, N_{epo}$ **do** ▷ L-BFGS training epoch loop
    $\mathcal{L} \leftarrow \frac{1}{N_{Resi}} \sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}_{Resi}} \left\| \frac{d\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})}{dt} + \mathbf{V}^T \left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) + g\left(\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})\right) + \mathbf{C}\right) \right\|_2^2$ ▷ compute loss
    $\Delta\mathbf{W} \leftarrow -\mathrm{G}_{\mathrm{L-BFGS}}(\nabla_{\mathbf{W}}\mathcal{L}, \nabla^2_{\mathbf{W}}\mathcal{L}), \Delta\mathbf{b} \leftarrow -\mathrm{G}_{\mathrm{L-BFGS}}(\nabla_{\mathbf{b}}\mathcal{L}, \nabla^2_{\mathbf{b}}\mathcal{L})$ ▷ L-BFGS optimizer step
    $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}, \mathbf{b} \leftarrow \mathbf{b} + \Delta\mathbf{b}$ ▷ update weights and biases
  end for
  $\mathbf{W}_{tr} \leftarrow \mathbf{W}, \mathbf{b}_{tr} \leftarrow \mathbf{b}$ ▷ save optimized weights and biases
end function

3.3.1. Physics-informed neural network (PINN)

Due to the limited amount of snapshots, it is difficult to train an accurate PDNN. A physics-informed machine learning for reduced-order modeling is considered for more accurate network prediction of the reduced-order solution without performing extra high-fidelity simulations.

Recently, physics-informed neural network (PINN) has been developed and applied to PDEs [50–52]. The idea is to use the residual of the PDE as part of the loss function, to make the network solution fulfill the physics. As there is no labeled data involved in the residual loss, unlimited training data points can be sampled in the space-time domain, which significantly reduces the amount of labeled data points needed for the training. It is also possible to train a network to predict the solution of a PDE without any labeled data [71]. Inspired by these observations, we propose to train a PINN to predict $\boldsymbol{\alpha}$, which is the solution of the reduced-order equation (17), without any high-fidelity snapshot data.

The PINN for reduced-order modeling is trained to satisfy the POD-G equation (17), by minimizing the residual loss of the equation on a set of sampled points in the time-parameter space. As we only need to evaluate the residual of (17) on each sampled time-parameter point $(t, \boldsymbol{\mu})$, the labeled output $\boldsymbol{\alpha}_{\mathcal{V}}(t;\boldsymbol{\mu})$ is not needed. Therefore, high-fidelity snapshots on the residual points are not required, which means that a large training data set can be generated at a small cost, independent of the full-order model size. Using Latin hypercube sampling in time-parameter space, we generate the data set $\mathcal{D}_{Resi} = \{t_i, \boldsymbol{\mu}_i\}_{i=1}^{N_{Resi}}$, containing N_{Resi} residual points. The loss function is defined as

$$\mathcal{L}_{\mathrm{PINN}} = \frac{1}{N_{\mathcal{D}}} \sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}} \left\| \frac{d\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})}{dt} + \mathbf{V}^T \left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) + g\left(\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})\right) + \mathbf{C}\right) \right\|_2^2. \tag{32}$$

The time derivative of the network is computed by using the automatic difference package of PyTorch. The training procedure of the PINN is summarized in Algorithm 1.

During the training of the PINN, the residual of the POD-G model equation needs to be computed on a set of points in the time-parameter space at each epoch. Therefore, the complexity of the nonlinear term of the reduced-order model affects the training speed of the PINN. In the present work, we restrict ourselves to the case of a quadratic nonlinearity, for which hyper-reduction is not needed. For more general nonlinearities, a proper hyper-reduction technique, such as the empirical interpolation method (EIM) [61], its discrete variant (DEIM) [62], gappy-POD [63] and missing point estimation (MPE) [64], should be used to efficiently evaluate the nonlinear term of the reduced-order model. Another technical difficulty that affects the training speed is the non-affine dependence of the linear PDE operators on the parameters. The computation of the linear term in (21) is a matrix-vector multiplication. The matrix is not of high-dimension, and the number of residual points does not need to be very large in real applications, as shown in Section 4. Therefore, we can pre-compute and store the matrices for all parameter values, to make the training efficient.

As the PINN is trained to approximate the reduced-order model, the accuracy limit of the PINN is the POD-G.

3.3.2. Physics-reinforced neural network (PRNN)

For complex nonlinear problems, the projection of the high-fidelity solution onto the reduced space is more accurate than the solution of the reduced-order equation, since the reduced-order equation does not take into account of the impact of the unresolved scales (truncated modes) on the resolved scales (reduced basis modes). Therefore, the projection coefficients can be used to provide more physical information during the network training and thus improve the accuracy. For this reason,

Algorithm 2 Training process of the PRNN.

function $[\mathbf{W}_{tr}, \mathbf{b}_{tr}]$=PRNN_TRAINING($\mathcal{D}_{Resi}, \mathcal{D}_{Pr}, N_{epo}, N_{batch}, \eta_0$)
  $\mathbf{W}, \mathbf{b} \leftarrow$ INIT($\mathbf{W}, \mathbf{b}$) ▷ initialize weights and biases
  $N_b^{Resi} \leftarrow N_{Resi}/N_{batch}, N_b^{Pr} \leftarrow N_s N_t/N_{batch}$ ▷ Mini-batch size
  for $epoch \leftarrow 1, 0.9 * N_{epo}$ **do** ▷ Adam training epoch loop
    $\eta \leftarrow$ LEARNING_RATE_DECAY($\eta_0, epoch$) ▷ learning rate decay
    $\mathcal{D}_{Resi} \leftarrow$ SHUFFLE($\mathcal{D}_{Resi}$), $\mathcal{D}_{Pr} \leftarrow$ SHUFFLE($\mathcal{D}_{Pr}$) ▷ shuffle the training data set
    for $s \leftarrow 1, N_{batch}$ **do** ▷ mini-batch loop
      $\mathcal{D}_{Resi,s} \leftarrow \mathcal{D}_{Resi}[(s-1) * N_b^{Resi} + 1 : s * N_b^{Resi}]$
      $\mathcal{D}_{Pr,s} \leftarrow \mathcal{D}_{Pr}[(s-1) * N_b^{Pr} + 1 : s * N_b^{Pr}]$ ▷ the s-th mini-batch
      $\mathcal{L} \leftarrow \frac{\omega}{N_b^{Resi}} \sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}_{Resi,s}} \left\| \frac{d\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})}{dt} + \mathbf{V}^T \left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) + g\left(\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})\right) + \mathbf{C}\right) \right\|_2^2 +$
      $\frac{1}{N_b^{Pr}} \sum_{(t,\boldsymbol{\mu}),\boldsymbol{\alpha} \in \mathcal{D}_{Pr,s}} \left\| f_{\mathrm{nn}}\left(\tilde{t}, \tilde{\boldsymbol{\mu}}\right) - \tilde{\boldsymbol{\alpha}}_{\mathcal{V}}(t;\boldsymbol{\mu}) \right\|_2^2$ ▷ compute loss
      $\Delta\mathbf{W} \leftarrow -\eta \mathrm{G}_{\mathrm{ADAM}}(\nabla_{\mathbf{W}}\mathcal{L}), \Delta\mathbf{b} \leftarrow -\eta \mathrm{G}_{\mathrm{ADAM}}(\nabla_{\mathbf{b}}\mathcal{L})$ ▷ Adam optimizer step
      $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}, \mathbf{b} \leftarrow \mathbf{b} + \Delta\mathbf{b}$ ▷ update weights and biases
    end for
  end for
  for $epoch \leftarrow 0.9 * N_{epo} + 1, N_{epo}$ **do** ▷ L-BFGS training epoch loop
    $\mathcal{L} \leftarrow \frac{\omega}{N_{Resi}} \sum_{(t,\boldsymbol{\mu}) \in \mathcal{D}_{Resi}} \left\| \frac{d\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})}{dt} + \mathbf{V}^T \left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu}) + g\left(\mathbf{V}\boldsymbol{\alpha}_{\mathrm{nn}}(t;\boldsymbol{\mu})\right) + \mathbf{C}\right) \right\|_2^2 +$
    $\frac{1}{N_s N_t} \sum_{(t,\boldsymbol{\mu}),\boldsymbol{\alpha} \in \mathcal{D}_{Pr}} \left\| f_{\mathrm{nn}}\left(\tilde{t}, \tilde{\boldsymbol{\mu}}\right) - \tilde{\boldsymbol{\alpha}}_{\mathcal{V}}(t;\boldsymbol{\mu}) \right\|_2^2$ ▷ compute loss
    $\Delta\mathbf{W} \leftarrow -\mathrm{G}_{\mathrm{L-BFGS}}(\nabla_{\mathbf{W}}\mathcal{L}, \nabla_{\mathbf{W}}^2\mathcal{L}), \Delta\mathbf{b} \leftarrow -\mathrm{G}_{\mathrm{L-BFGS}}(\nabla_{\mathbf{b}}\mathcal{L}, \nabla_{\mathbf{b}}^2\mathcal{L})$ ▷ L-BFGS optimizer step
    $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}, \mathbf{b} \leftarrow \mathbf{b} + \Delta\mathbf{b}$ ▷ update weights and biases
  end for
  $\mathbf{W}_{tr} \leftarrow \mathbf{W}, \mathbf{b}_{tr} \leftarrow \mathbf{b}$ ▷ save optimized weights and biases
end function

we refer to the trained network as physics-reinforced neural network (PRNN). The loss function is defined as the weighted sum of the mean squared norm of the residual of the reduced-order equation and the mean squared error between the network output and the projection coefficients of the high-fidelity solutions. The detailed form of the loss function is

$$\mathcal{L}_{\mathrm{PRNN}} = \mathcal{L}_{\mathrm{PDNN}}|_{\mathcal{D}=\mathcal{D}_{Pr}} + \omega\, \mathcal{L}_{\mathrm{PINN}}|_{\mathcal{D}=\mathcal{D}_{Resi}}\,. \tag{33}$$

where the hyper-parameter ω is the residual loss weight of which the quasi-optimal value is determined by grid-search in the numerical tests. The training procedure of the PRNN is summarized in Algorithm 2. We note that the projection data set for a steady-state problem is typically small, and we just use one mini-batch for the projection data set. For unsteady problems, during the training, the projection and residual data sets are divided into the same number of mini-batches.

The PRNN is trained using the information of physics and projection data, and an ideal PRNN is supposed to achieve accuracy that is higher than the POD-G and lower than the projection. However, in the numerical results in Section 4, we observe that for the PDNN, PINN and PRNN, the accuracy reaches a certain limit during the reduced basis refinement. This is caused by the prediction error of the networks, and is impossible to avoid since the generalization accuracy of an artificial neural network is limited.

A great advantage of a non-intrusive RB method over its intrusive counterpart is the local modeling capability [41]. Inheriting the local modeling capability from the non-intrusive RB method, the physics-informed machine learning based reduced-order models can be built on a local time range of interest, which reduces the modeling complexity.

4. Numerical results

4.1. Steady-state problems

In this subsection, we use several steady-state problems to evaluate the proposed methods. Although the methods are described for unsteady problems in Sections 2 and 3, it is straightforward to obtain the steady-state formulas by setting $N_t = 1$ and removing the time derivatives and initial conditions. This subsection presents the numerical results of the reduced basis methods based on PDNN, PINN and PRNN for the one-dimensional Burgers' equation, the two-dimensional lid-driven cavity flow and the two-dimensional natural convection. The results of the following two methods are also presented for comparison:

(1) **Projection**: Projection of high-fidelity solution onto the reduced basis space;
(2) **POD-G**: Intrusive POD-Galerkin reduced-order model.

Table 1
Network training hyper-parameters for steady-state problems.

	Number of mini-batches				
Network	$\mathcal{D}_{Resi}$	$\mathcal{D}_{Pr}$	Epochs	Learning rate for epoch i	λ
PDNN		1	20,000	$0.01 \times 0.95^{\text{int}(\frac{i-1}{200})}$	10^{-10}
PINN	10		20,000	$0.01 \times 0.95^{\text{int}(\frac{i-1}{200})}$	
PRNN	10	1	20,000	$0.01 \times 0.95^{\text{int}(\frac{i-1}{200})}$	

The mean relative error of the reduced-order solution is used to measure the accuracy of the methods:

$$\varepsilon_{\bullet} = \frac{1}{N_{\mathcal{D}_{te}}} \sum_{\boldsymbol{\mu}, \boldsymbol{\phi}_h \in \mathcal{D}_{te}} \frac{\left\| \boldsymbol{\phi}_h - \widetilde{\boldsymbol{\phi}} - \mathbf{V}\boldsymbol{\alpha}_{\bullet}(\boldsymbol{\mu}) \right\|_2}{\|\boldsymbol{\phi}_h\|_2}, \tag{34}$$

where $\mathcal{D}_{te} = \{\boldsymbol{\mu}_i, \boldsymbol{\phi}_h(\boldsymbol{\mu}_i)\}_{i=1}^{N_{\mathcal{D}_{te}}}$ is the test data set of size $N_{\mathcal{D}_{te}}$, the subscript "$\bullet$" is used to take the place of "Projection", "POD-G","PDNN","PINN" and "PRNN", respectively. For steady-state problems, the reference value is simply set as zero, namely $\widetilde{\boldsymbol{\phi}} = 0$.

The sampling methods in the parameter space $\mathcal{P}$ in the numerical experiments are:

(1) high-fidelity snapshots: Sobol sequence;
(2) residual points: Latin hypercube sampling;
(3) validation data points: Random sampling;
(4) test data points: Uniform sampling;

The network training hyper-parameters for steady-state problems are set as the values listed in Table 1, if not stated otherwise. The quasi-optimal residual loss weight ω (defined in (33)) for PRNN is sought among five values $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$, if not stated otherwise. To avoid the influence of random initialization of weights and biases on network training, each ANN is trained with 5 restarts. The trained network that has the smallest mean relative error (34) on the validation data set is selected as the final model.

We refer the reader to the repository[1] for details of the code and data.

4.1.1. Burgers' equation

To validate the proposed methods, we first consider the one-dimensional parameterized Burgers' equation

$$\begin{cases} \phi(x;\boldsymbol{\mu}) \dfrac{\partial \phi(x;\boldsymbol{\mu})}{\partial x} - \dfrac{\partial \phi^2(x;\boldsymbol{\mu})}{\partial x^2} = s(x;\boldsymbol{\mu}) & -1 \le x \le 1 \\ \phi(x = \pm 1;\boldsymbol{\mu}) = 0 \end{cases}, \tag{35}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2) \in [1, 10] \times [1, 10]$ are the parameters in the manufactured solution

$$\phi(x;\boldsymbol{\mu}) = (1 + \mu_1 x)\sin(-\mu_2 x/3)(x^2 - 1). \tag{36}$$

The source term $s(x;\boldsymbol{\mu})$ can be derived analytically by substituting the solution (36) into (35). High-fidelity snapshots are obtained using the Chebyshev pseudospectral method with 129 Chebyshev Gauss-Lobatto points.

A baseline reduced-order model is constructed to validate the proposed methods. The reduced basis functions are extracted using $N_s = 80$ high-fidelity snapshots. The singular value decay is shown in Fig. 3(a). The singular value decays quickly, implying that a small number of reduced basis functions is enough to recover the main dynamics of the solution. Meanwhile, the number of high-fidelity snapshots can be also reduced. The PDNN, PINN and PRNN have the same network architecture with $L = 3$ hidden layers and $n_H = 30$ neurons in each hidden layer. The number of residual points for the PINN and PRNN is $N_{Resi} = 1024$. The accuracy of the physics-informed machine learning based reduced-order model depends on the sizes of the projection data set, the residual data set and the feedforward network. The impacts of these three factors on the accuracy will be investigated separately, e.g., we train networks of different sizes on the same projection and residual data sets, to study the effect of the network size. A data set with 40 randomly sampled points in parameter space is employed as the validation data set. A data set with $101 \times 101 = 10201$ uniformly sampled points in parameter space is used to test the accuracy of the models.

Reduced-order models built with $N_s = 10$, 80 and 320 snapshots are tested to investigate the impact of the size of the projection data on the accuracy of the methods. The prediction error is shown in Fig. 4, and the corresponding quasi-optimal residual loss weight for PRNN is shown in Fig. 5. The POD-G is accurate as its error is only slightly larger than

[1] available at https://github.com/cwq2016/POD-PINN.

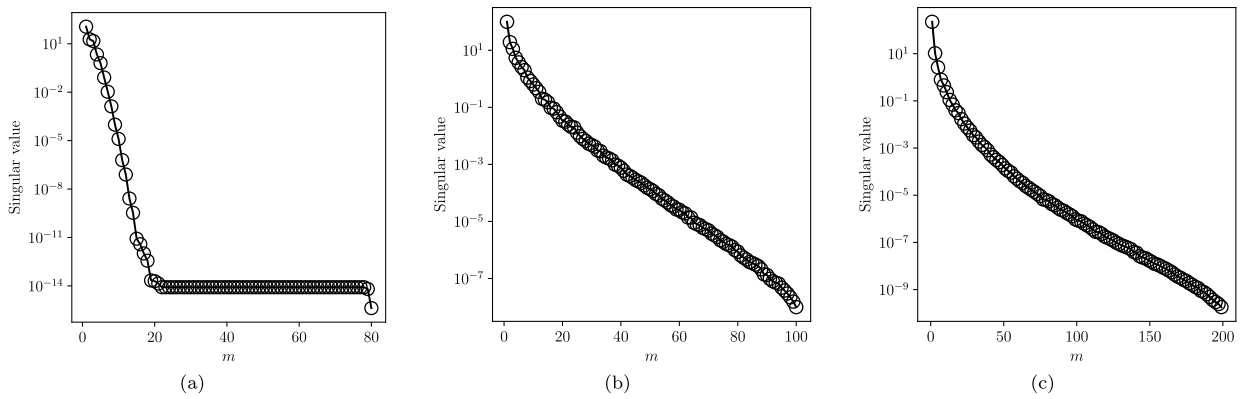

Fig. 3. Singular value decay of the steady-state problems: (a) the Burgers' equation with 80 snapshots, (b) the lid-driven cavity flow problem with 100 snapshots and (c) the natural convection problem with 200 snapshots.

the projection, indicating that the impact of the unresolved scales on the resolved scales can be neglected. The PINN is as accurate as PRNN, and both follow the POD-G closely until they reach the accuracy limit of $\mathcal{O}(10^{-5})$ at $m = 8$. The results show that adding the projection data to the physics-informed learning is not beneficial in this case, since the POD-G model is already very accurate so that an accurate PINN can be obtained by learning the POD-G model on the residual data set. The PINN and PRNN are much more accurate than the PDNN. With $N_s = 10$ and $m \geq 8$, the error of the PINN/PRNN is up to three orders of magnitude smaller than that of the PDNN. As the dynamics of the problem can be recovered accurately on a small set of reduced basis, significant accuracy improvement due to the refinement of the high-fidelity snapshots is not observed in Fig. 4, except for the purely data-driven PDNN.

An important observation in Fig. 4 is that during the reduced basis refinement for the network-based reduced-order methods, the accuracy reaches a plateau when the basis size becomes larger than a threshold. This is caused by the reduced coefficients prediction error, and is impossible to avoid since the generalization accuracy of an artificial neural network is limited. As lower-order modes play more important roles than higher-order modes, prediction error of the reduced coefficients becomes dominant when the set of reduced basis is sufficiently large and the accuracy of the reduced-order solution saturates.

Reduced-order models built with $N_{Resi} = 256$, 512 and 1024 are tested to investigate the impact of the size of the residual data set on the accuracy of PINN and PRNN. In Fig. 6(a), it is shown that the accuracy improves with the residual data set refinement, which indicates that the PINN and PRNN can learn the reduced-order equation more accurately with more residual points.

Reduced-order models based on networks with different hidden layer widths and depths are tested to investigate the impact of the network size on the accuracy. The errors of the PDNN, PINN and PRNN with various hidden layer widths are shown in Fig. 6(b), where the network depth is fixed as $L = 3$. The errors of the PDNN, PINN and PRNN with various network depths are shown in Fig. 6(c), where the network hidden layer width is fixed as $n_H = 30$. The results show that the larger PDNN, PINN and PRNN are more accurate, since larger networks have stronger ability to learn the projection data or the reduced-order equation. Furthermore, it is shown that the hidden layer width has larger impact on the prediction accuracy of networks than the depth.

Reduced-order models with $\lambda = 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 0$ are tested to investigate the impact of L_2 regularization on the accuracy of PDNN. In Fig. 6(d), it is shown that the accuracy reaches the maximum at $\lambda = 10^{-10}$, which is taken as the default value of λ in Table 1.

4.1.2. Lid-driven cavity flow

In this subsection, the lid-driven cavity flow problem is used to test the proposed reduced-order methods. The viscous flow in a parallelogram-shaped cavity is described by the following non-dimensionalized incompressible Navier-Stokes equations

$$\begin{cases} \nabla_{\mathbf{x}} \cdot \mathbf{u} = 0 & \mathbf{x} \in \Omega \\ \mathbf{u} \cdot \nabla_{\mathbf{x}} \mathbf{u} = -\nabla_{\mathbf{x}} p + \dfrac{1}{Re} \nabla_{\mathbf{x}}^2 \mathbf{u} & \mathbf{x} \in \Omega \\ u = u_w, v = 0 & \mathbf{x} \in \Gamma_1 \\ u = v = 0 & \mathbf{x} \in \Gamma_2 \end{cases}, \tag{37}$$

where $\mathbf{u} = (u, v)$ is the velocity in the Cartesian coordinate system $\mathbf{x} = (x, y)$, p is the pressure and Re is the Reynolds number. The geometry of the problem is depicted in Fig. 7 (a), and is affected by the angle θ between the oblique sides

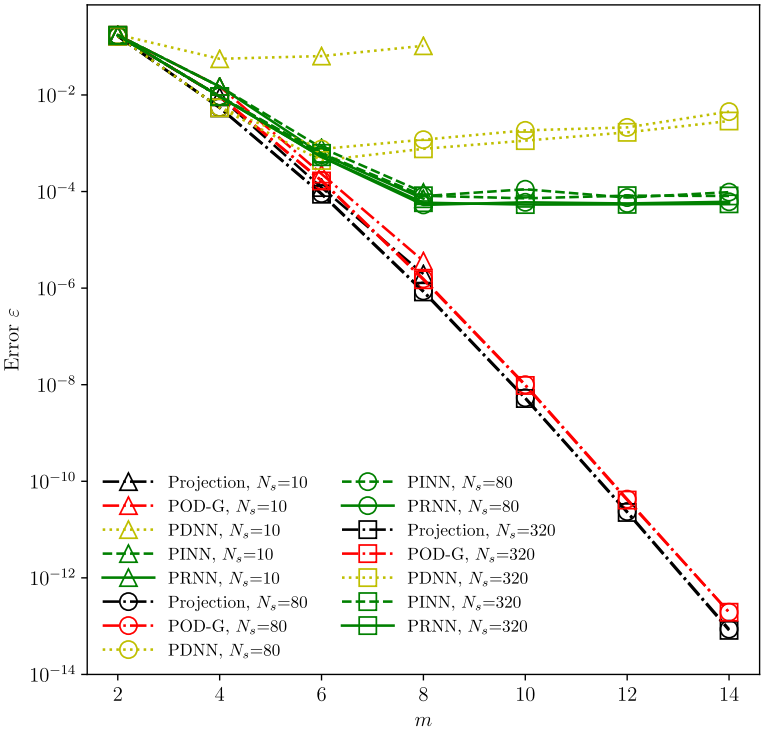

Fig. 4. Prediction error for the steady-state Burgers' equation with different number of snapshots. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

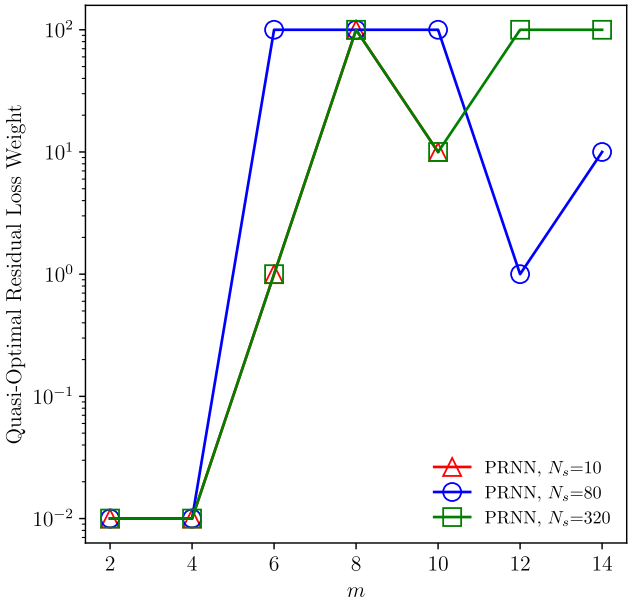

Fig. 5. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$) of the PRNN for the steady-state Burgers' equation.

and the positive x-semiaxis. The boundary is $\partial\Omega = \Gamma_1 \cup \Gamma_2$, where Γ_1 represents the top moving wall and Γ_2 represents the other three static non-slip walls. $\boldsymbol{\mu} = (Re, \theta)$ are the parameters for this problem.

The high-fidelity simulations are performed in the computational domain $[-1, 1]^2$. The mapping from the physical coordinate $\mathbf{x} = (x, y)$ to the computational coordinate $\boldsymbol{\xi} = (\xi_1, \xi_2) \in [-1, 1]^2$ is defined as

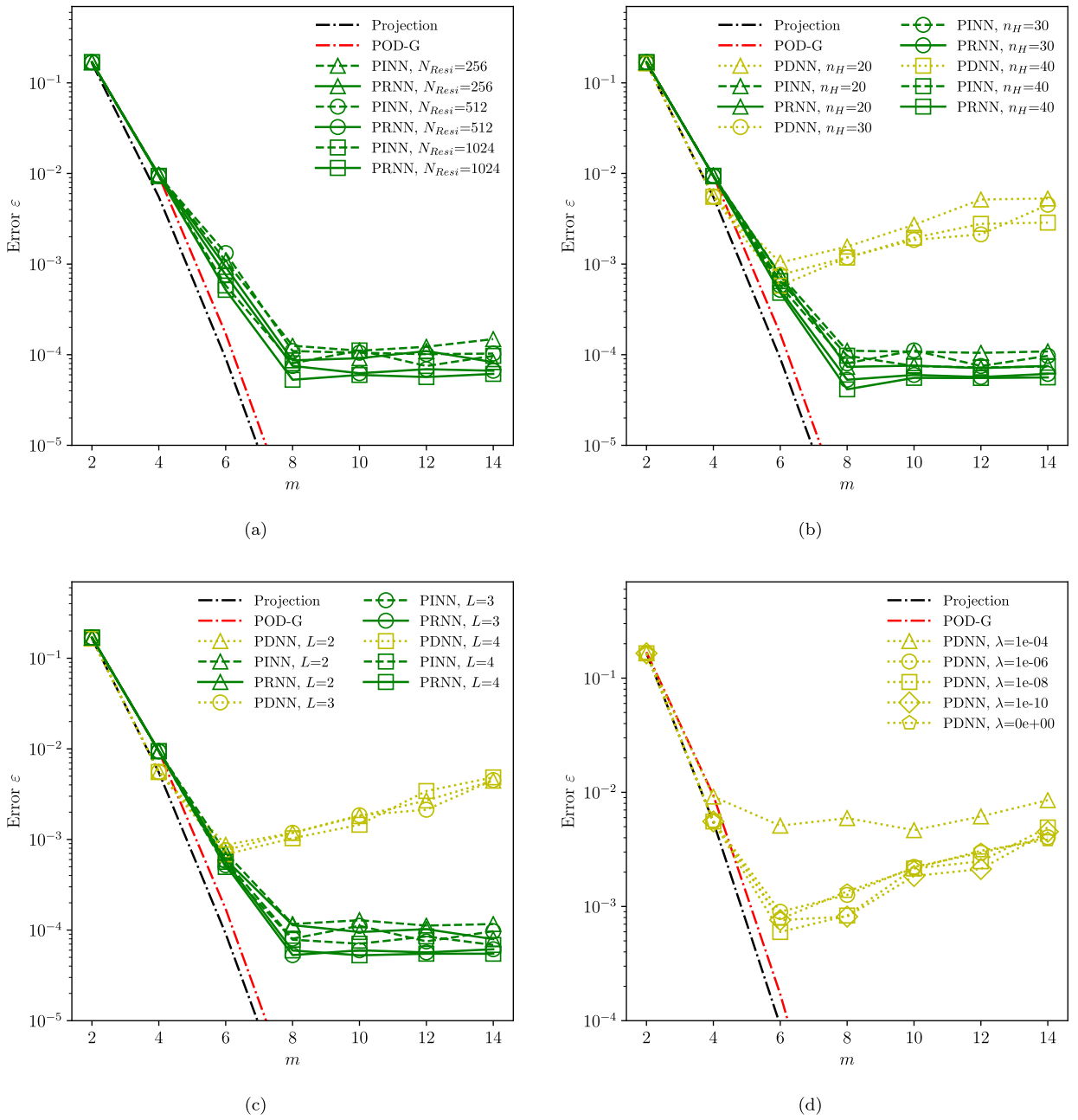

Fig. 6. Prediction errors for the steady-state Burgers' equation with different (a) number of residual points N_{Resi}, (b) network hidden layer width n_H, (c) network depth L and (d) L_2 regularization parameter λ.

$$\mathcal{X}: \begin{cases} x = 1/2\xi_1 + 1/2\xi_2 \cos(\theta) \\ y = 1/2\xi_2 \sin(\theta) \end{cases} \quad \rightleftharpoons \quad \mathcal{X}^{-1}: \begin{cases} \xi_1 = 2(x - y\cot(\theta)) \\ \xi_2 = 2y/\sin(\theta) \end{cases}. \tag{38}$$

The Jacobian matrix of the mapping is

$$\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 2 & 0 \\ -2\cot(\theta) & 2/\sin(\theta) \end{bmatrix}. \tag{39}$$

The governing equations in computational space are

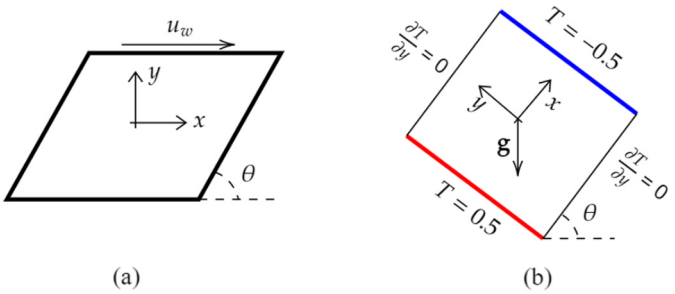

Fig. 7. The geometry and boundary conditions for (a) Lid-driven cavity flow and (b) Natural convection.

$$\begin{cases} \left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)\cdot\mathbf{u}=0 \\ \left(\mathbf{u}\cdot\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)\right)\mathbf{u}=-\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)p+\dfrac{1}{Re}\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)^2\mathbf{u} \end{cases}. \tag{40}$$

The flow is driven by the top wall moving with a velocity

$$u_w=(1+\xi_1)^2(1-\xi_1)^2. \tag{41}$$

The governing equations (40) are discretized by the Chebyshev pseudospectral method on a tensor-product mesh of 49×49 points. The IP_N-IP_{N-2} method [56,57] is adopted to remove the spurious modes of pressure. The artificial compressibility method [72] is used to address the velocity-pressure coupling issue by adding pseudo time derivatives to the steady-state equation (40). The discretized equations are advanced in pseudo time using the explicit four-stage fourth-order Runge-Kutta scheme.

The parameter space of interest is $\boldsymbol{\mu}=(Re,\theta)\in[100,500]\times[\pi/3,2\pi/3]$. The singular values for $N_s=100$ snapshots are shown in Fig. 3(b). It shows that the singular values decay slowly, implying that a large number of basis functions are required to recover the main dynamics of the problem accurately.

The POD-G solver diverges for many parameter values if a proper initial reduced-order solution can not be provided. To avoid the divergence issue, for a parameter $\boldsymbol{\mu}$, we use the projection of the corresponding high-fidelity solution onto the reduced space as the initial solution for the POD-G solver. This of course comes at high cost.

The PDNN, PINN and PRNN have the same network architecture with $L=5$ hidden layers and $n_H=30$ neurons in each hidden layer. The number of residual points for the PINN and PRNN is $N_{Resi}=2048$.

Reduced-order models are built with $N_s=30$, 60 and 100 snapshots. A data set with 50 random sampled points in parameter space is employed as the validation data set. A data set with $11\times 11=121$ uniformly sampled points in parameter space is used to test the accuracy of the reduced-order models. The prediction errors of the reduced-order models are shown in Fig. 8, and the corresponding quasi-optimal residual loss weight for PRNN is shown in Fig. 9. It is shown that the POD-G error is larger than the projection error, due to the difficulty of recovering the main dynamics of the problem accurately using a small set of basis functions. As the projection of the high-fidelity solution onto the reduced space is more accurate than the POD-G solution, the use of the projection data in the training makes the PRNN more accurate than the PINN. It is shown in Fig. 8 that the error curve of the PRNN lies in between that of the POD-G and projection when $m\leq 20$, which demonstrates that the PRNN can reach higher accuracy than the POD-G in a low-dimensional reduced space. PRNN is more accurate than the purely data-driven PDNN, which suffers from the low-accuracy and over-fitting issues on small data sets. The error of the PRNN is around one order of magnitude smaller than that of the PDNN for $m\geq 20$ and $N_s=30$.

Predictions by the PRNN for several parameter values are compared with the corresponding high-fidelity solutions in Fig. 10 to intuitively show the accuracy of the proposed reduced-order method. The PRNN reduced-order solutions agree well with the high-fidelity solutions.

4.1.3. Natural convection

The natural convection problem is used to further test the proposed reduced-order methods. The unit square cavity is filled with an incompressible Newtonian fluid. The cavity is heated and cooled on the left and right walls, while the other two boundaries are adiabatic walls. The geometry and boundary conditions are shown in Fig. 7(b). The flow is driven by the vertical buoyancy force, which is modeled by the Boussinesq approximation. The orientation of the cavity is controlled by an angle θ. The physical coordinate system $x-y$ is attached to the cavity, by setting the origin as the center of the cavity and the y-axis parallel to the heated/cooled walls. The natural convection can be modeled by the non-dimensionalized incompressible Navier-Stokes equations plus an energy equation

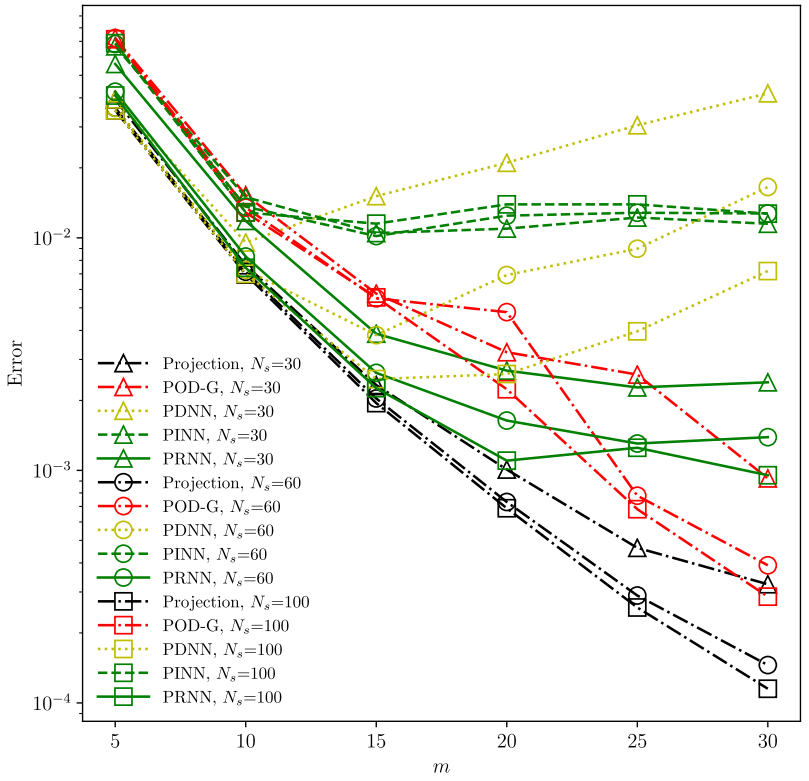

Fig. 8. Prediction error for the lid-driven cavity flow problem with different number of snapshots.

$$\begin{cases} \nabla_{\mathbf{x}} \cdot \mathbf{u} = 0 & \mathbf{x} \in \Omega \\ \mathbf{u} \cdot \nabla_{\mathbf{x}} \mathbf{u} = -\nabla_{\mathbf{x}} p + \sqrt{\left(\frac{Pr}{Ra}\right)} \nabla_{\mathbf{x}}^2 \mathbf{u} - T \mathbf{n}_g & \mathbf{x} \in \Omega \\ \mathbf{u} \cdot \nabla_{\mathbf{x}} T = \frac{1}{\sqrt{(Pr \times Ra)}} \nabla_{\mathbf{x}}^2 T & \mathbf{x} \in \Omega \\ \mathbf{u} = 0, T = +0.5 & \mathbf{x} \in \Gamma_1 \\ \mathbf{u} = 0, T = -0.5 & \mathbf{x} \in \Gamma_2 \\ \mathbf{u} = 0, \partial T / \partial \mathbf{n} = 0 & \mathbf{x} \in \Gamma_3 \end{cases}, \tag{42}$$

where $\mathbf{u} = (u, v)$ is the velocity in the Cartesian coordinate system $\mathbf{x} = (x, y)$, p is the pressure, Ra is the Rayleigh number, Pr is the Prandtl number and $\mathbf{n}_g = (-\sin\theta, -\cos\theta)$ is the direction of gravity. Γ_1 represents the left hot wall, Γ_2 represents the right cold wall and Γ_3 represents the other two walls. Similar to the lid-driven cavity problem in Section 4.1.2, the high-fidelity simulations are obtained using the Chebyshev pseudospectral method on a mesh with 49×49 points. As the physical domain is a unit square, the physical coordinate can be linearly scaled to the computational coordinate. The governing equations in computational space can be easily obtained from (42) and are thus omitted.

The parameter space of interest is $\boldsymbol{\mu} = (Ra, Pr, \theta) \in [10^4, 10^5] \times [0.6, 0.8] \times [\pi/4, \pi/2]$. The singular values for $N_s = 200$ snapshots are shown in Fig. 3(c). It shows that the singular values decay slowly, suggesting that a large number of basis functions are needed to accurately recover the main dynamics of the problem.

Similar to the lid-driven cavity problem, to avoid the divergence issue, we use the projection of the corresponding high-fidelity solution onto the reduced space as the initial solution for the POD-G solver.

The PDNN, PINN and PRNN have the same network architecture with $L = 5$ hidden layers and $n_H = 30$ neurons in each hidden layer. The number of residual points for the PINN and PRNN is $N_{Resi} = 2048$.

Reduced-order models are built with $N_s =$ 30, 100 and 200 snapshots. A data set with 30 random sampled points in parameter space is employed as the validation data set. A data set with $6 \times 6 \times 6 = 216$ uniformly sampled points in parameter space is used to test the accuracy of the reduced-order models. The prediction errors of the reduced-order models are shown in Fig. 11, and the corresponding quasi-optimal residual loss weight for PRNN is shown in Fig. 12. It is shown that the POD-G error is larger than the projection error, due to the difficulty of recovering the main dynamics of the

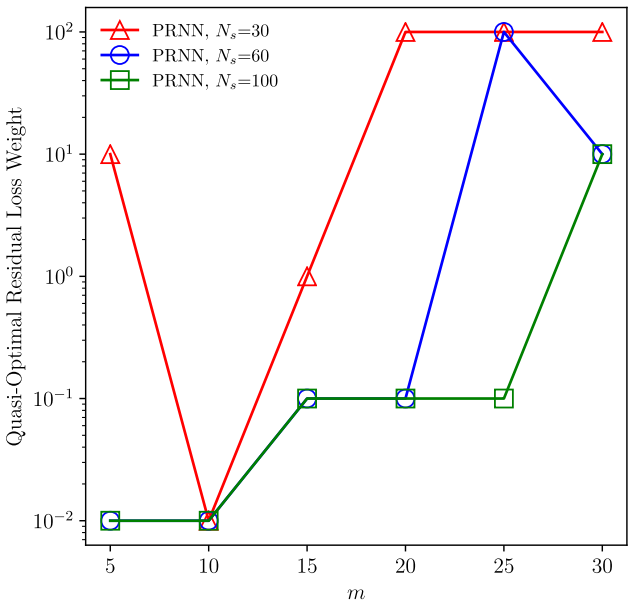

Fig. 9. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$) of the PRNN for the lid-driven cavity flow problem.

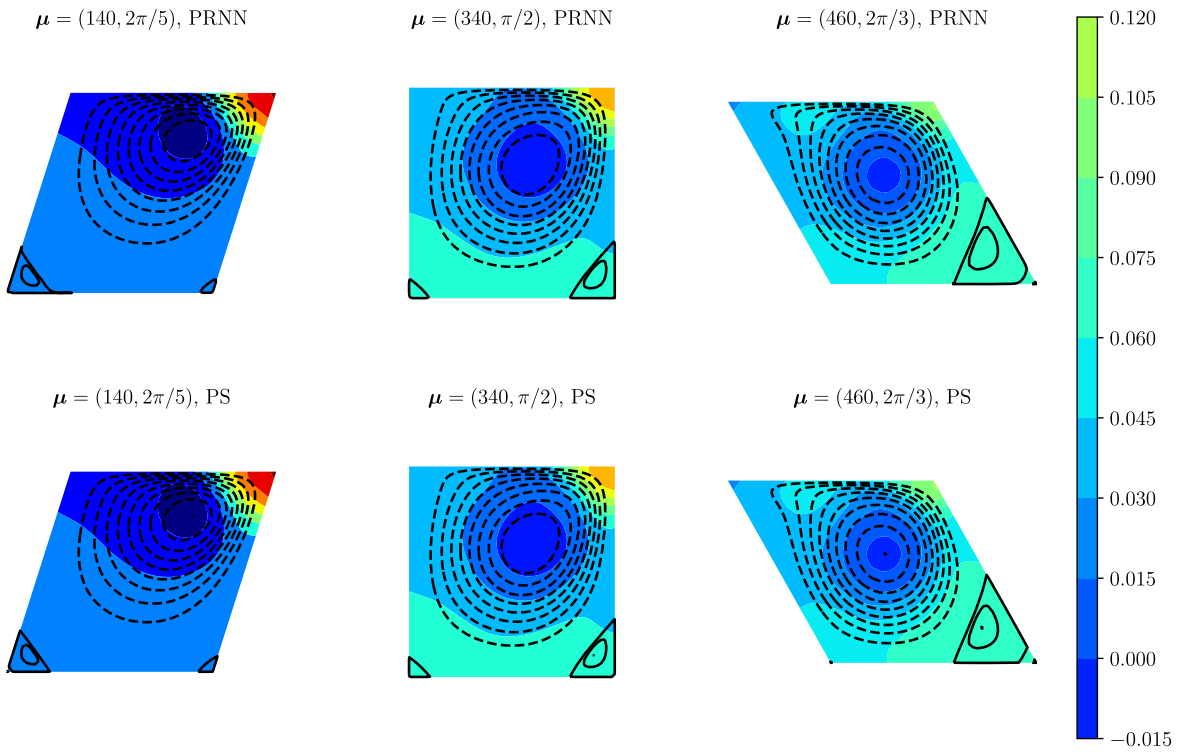

Fig. 10. Pressure contours and streamlines of the lid-driven cavity flow computed by the high-fidelity solver and the PRNN for three parameter values. The reduced basis is of size $m = 25$ and extracted from 100 snapshots. The solid streamline indicates an anticlockwise vortex and the dashed streamline indicates an clockwise vortex.

problem accurately using a small set of basis functions. The PRNN is more accurate than the PINN because of the use of the projection data in the training. It is shown in Fig. 11 that the error curve of the PRNN lies between that of the POD-G and projection when $m \leq 15$, which demonstrates that the PRNN reaches higher accuracy than the POD-G in low-dimensional reduced space. PRNN is more accurate than the purely data-driven PDNN.

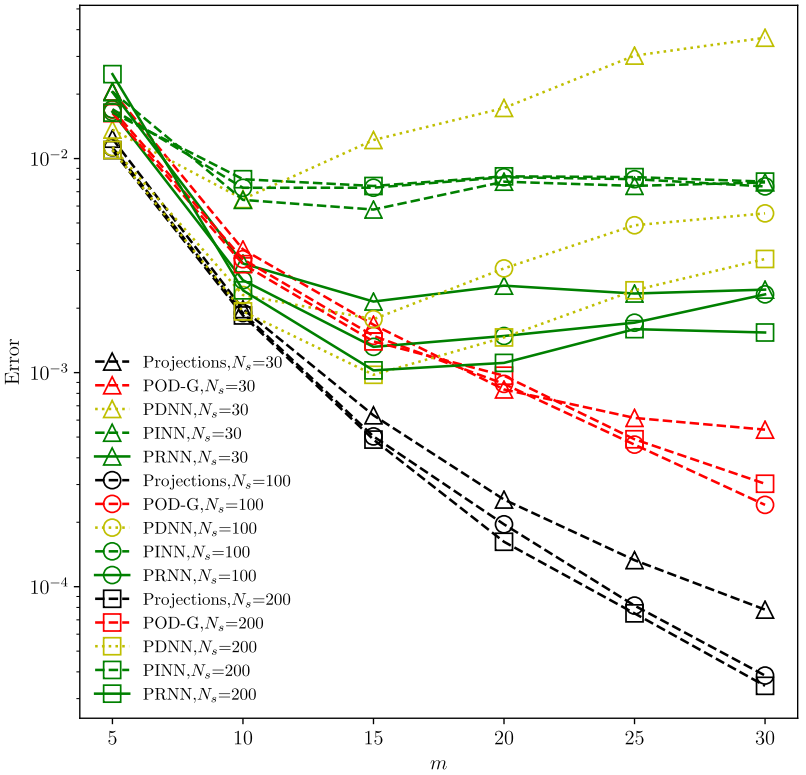

Fig. 11. Prediction error for the steady-state natural convection problem with different number of snapshots.

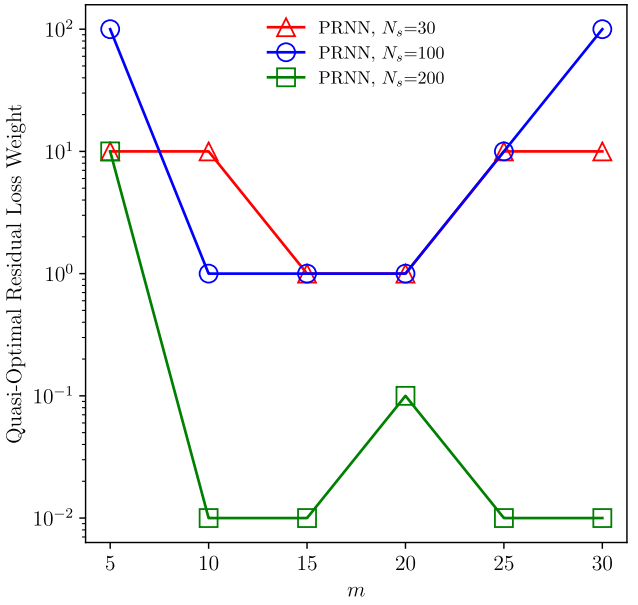

Fig. 12. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$) of the PRNN for the steady-state natural convection problem.

Predictions by the PRNN for several parameter values are compared with the corresponding high-fidelity solutions in Fig. 13 to show the accuracy of the proposed reduced-order method. The PRNN reduced-order solutions agree well with the high-fidelity solutions.

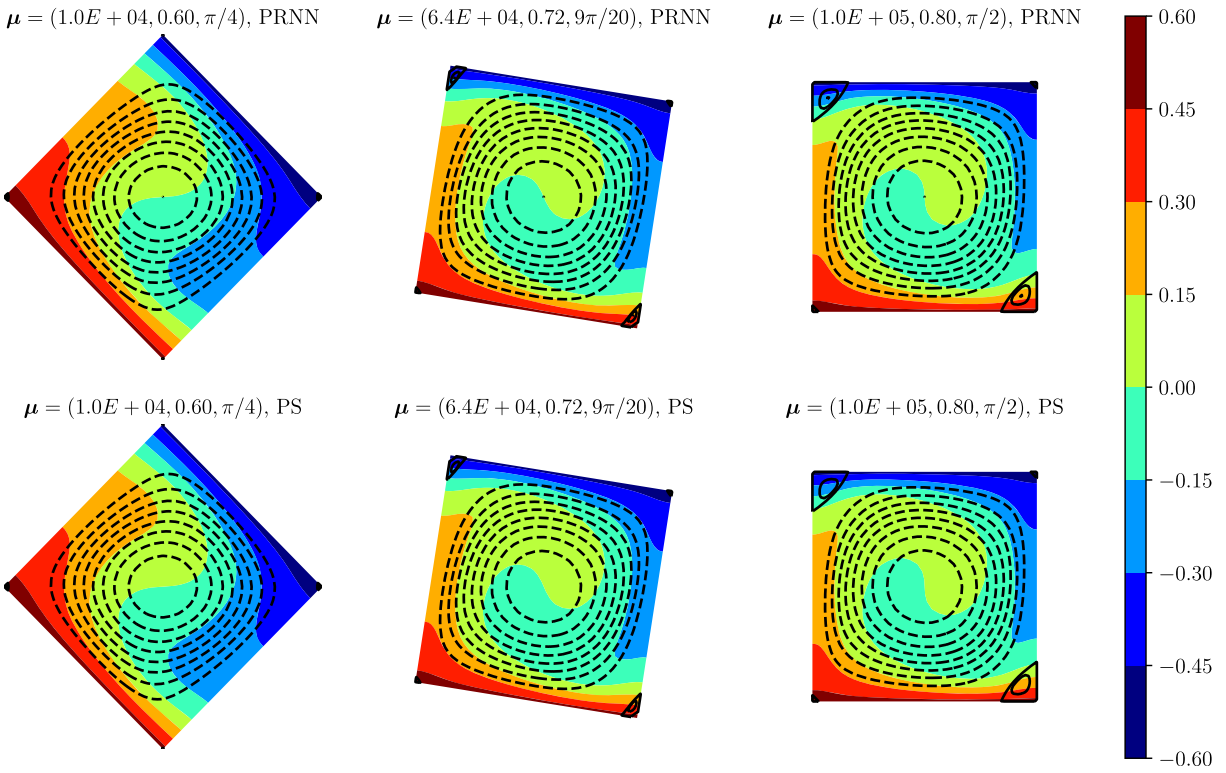

Fig. 13. Temperature contours and streamlines of the natural convection computed by the high-fidelity solver and the PRNN for three parameter values. The reduced basis is of size $m = 20$ and extracted from 200 snapshots. The solid streamline indicates an anticlockwise vortex and the dashed streamline indicates an clockwise vortex.

4.2. Unsteady problems

In this subsection, we use several unsteady problems to further evaluate the proposed methods. It is shown in Section 4.1 that the PINN is less accurate than the PRNN, as the projection data is added to the training of PRNN. Therefore, only two networks, namely the PDNN and PRNN, are considered in this subsection. The test cases are the one-dimensional Burgers' equation, the two-dimensional advection diffusion reaction flow and the two-dimensional natural convection flow. The results of the following three methods are also presented for comparison:

(1) **Projection**: Projection of high-fidelity solution onto the reduced basis space;
(2) **POD-G**: Intrusive POD-Galerkin reduced-order model.
(3) **POD-G + closure**: Intrusive POD-Galerkin reduced-order model with closure.

The closure term for the POD-G model represents the impact of the unresolved scales on the resolved scales. The stability and accuracy of the POD-G model can be significantly improved by adding the closure term. For the introduction of closure modeling for reduced-order modeling, we refer the reader to [17]. As we do not intend to do prediction beyond the time range of the training data, a Radial Basis Function (RBF) interpolation is utilized to compute the closure term of the POD-G model, instead of the recurrent neural network used in [17]. The RBF interpolation model is trained on the projection data set. The input for the RBF interpolation is the time-parameter $(t, \boldsymbol{\mu})$, and the labeled output is the closure term which is the difference between the projection of the full-order model onto the reduced space and the POD-G model. The POD-G and the "POD-G + closure" are advanced in time using the backward Euler method.

The sampling methods in the time-parameter space $\mathcal{T} \times \mathcal{P}$ in the numerical experiments are:

(1) high-fidelity snapshots: $\mathcal{T}_h \times$ Sobol sequence in parameter space $\mathcal{P}$;
(2) residual points: Latin hypercube sampling in the time-parameter space $\mathcal{T} \times \mathcal{P}$;
(3) validation data points: $\mathcal{T}_h \times$ Random sampling in parameter space;
(4) test data points: $\mathcal{T}_h \times$ Uniform sampling in parameter space $\mathcal{P}$;

Table 2
Network training hyper-parameters for unsteady problems.

	Number of mini-batches				
Network	$\mathcal{D}_{Resi}$	$\mathcal{D}_{Pr}$	Epochs	Learning rate for epoch i	λ
PDNN	10	10	20,000	$0.01 \times 0.95^{\text{int}(\frac{i-1}{200})}$	10^{-10}
PRNN	10	10	20,000	$0.01 \times 0.95^{\text{int}(\frac{i-1}{200})}$	

The following metric is used to measure the accuracy for unsteady problems:

$$\varepsilon_{\bullet} = \frac{1}{N_{\mathcal{D}_{te}}} \sum_{\boldsymbol{\mu} \in \mathcal{D}_{te}} \sqrt{\frac{1}{t_e - t_s} \int_{t_s}^{t_e} \left\| \boldsymbol{\phi}_h - \widetilde{\boldsymbol{\phi}} - \mathbf{V}\boldsymbol{\alpha}_{\bullet}(t;\boldsymbol{\mu}) \right\|_2^2 dt}, \tag{43}$$

where $[t_s, t_e]$ is the time range to be tested on, $\mathcal{D}_{te} = \{\boldsymbol{\mu}_i\}_{i=1}^{N_{\mathcal{D}_{te}}}$ is the test data set of size $N_{\mathcal{D}_{te}}$ in parameter space, the subscript "$\bullet$" is used to take the place of "Projection", "POD-G", "POD-G + closure", "PDNN", and "PRNN", respectively.

The network training hyper-parameters for unsteady problems are set as the values listed in Table 2, if not stated otherwise. The quasi-optimal residual loss weight ω for PRNN is considered among the five values $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$, if not stated otherwise. To avoid the influence of random initialization of weights and biases on network training, each ANN is trained with 5 restarts. The trained network that has the smallest mean relative error (43) on the validation data set is selected as the final model.

4.2.1. Burgers' equation

The following time-dependent one-dimensional Burgers' equation is used to test the proposed methods

$$\begin{cases} \dfrac{\partial \phi}{\partial t} + \phi \dfrac{\partial \phi}{\partial x} = \dfrac{1}{Re} \dfrac{\partial^2 \phi}{\partial x^2} & (x,t) \in \Omega \times \mathcal{T} \\ \phi(x,t) = 0 & (x,t) \in \partial\Omega \times \mathcal{T} \\ \phi(x,0) = \dfrac{x}{1 + \sqrt{\frac{1}{t_0}} \exp\left(Re \frac{x^2}{4}\right)} & x \in \Omega \end{cases}. \tag{44}$$

The spatial domain is $\Omega = [0, 1]$ and the temporal domain is $\mathcal{T} = [0, 2]$. The exact solution is

$$\phi(x,t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}} \exp\left(Re \frac{x^2}{4t+4}\right)}, \tag{45}$$

where $t_0 = \exp(Re/8)$. The only parameter for this problem is $\boldsymbol{\mu} = Re \in [200, 800]$. High-fidelity snapshots are computed using (45), with 129 Chebyshev Gauss-Lobatto points in space and 101 equally spaced points in time.

The reduced basis is extracted from the snapshots computed using $N_s = 10$ parameter instances and the reference value $\widetilde{\boldsymbol{\phi}}$ is set as the initial solution. The singular value decay is shown in Fig. 14(a). The singular value decays quickly, suggesting that a small number of reduced basis functions is enough to recover the main dynamics of the solution. The PDNN and PRNN have the same network architecture with $L = 2$ hidden layers and $n_H = 100$ neurons in each hidden layer. The number of residual points for the PRNN is $N_{Resi} = 2048$. A data set with 10 randomly sample parameter points and 101 equally spaced time points is employed as the validation data set. A data set with uniformly sampled 100 parameter points and 101 equally spaced time points is used to test the accuracy of the models.

Reduced-order models built with $N_s = 5$, 10 and 20 parameter instances are tested to investigate the impact of the size of the projection data on the accuracy of the methods. The prediction error is shown in Fig. 15, and the corresponding quasi-optimal residual loss weight for PRNN is shown in Fig. 16. The POD-G is accurate as its error is only slightly larger than the projection, which indicates that the impact of the unresolved scales on the resolved scales can be neglected and there is no need for closure modeling. The PRNN follows the POD-G closely until it reach the accuracy limit of $\mathcal{O}(10^{-4})$ at $m = 12$. The PRNN is much more accurate than the PDNN. With $N_s = 5$ and $m \geq 12$, the error of the PRNN is up to two orders of magnitude smaller than that of the PDNN. It is observed in Fig. 15 that the projection data set size has much larger impact on the accuracy of PDNN than that of PRNN.

4.2.2. Advection diffusion reaction

To validate the improvement by using the local modeling strategy of the non-intrusive methods, we consider a linear advection-diffusion-reaction problem

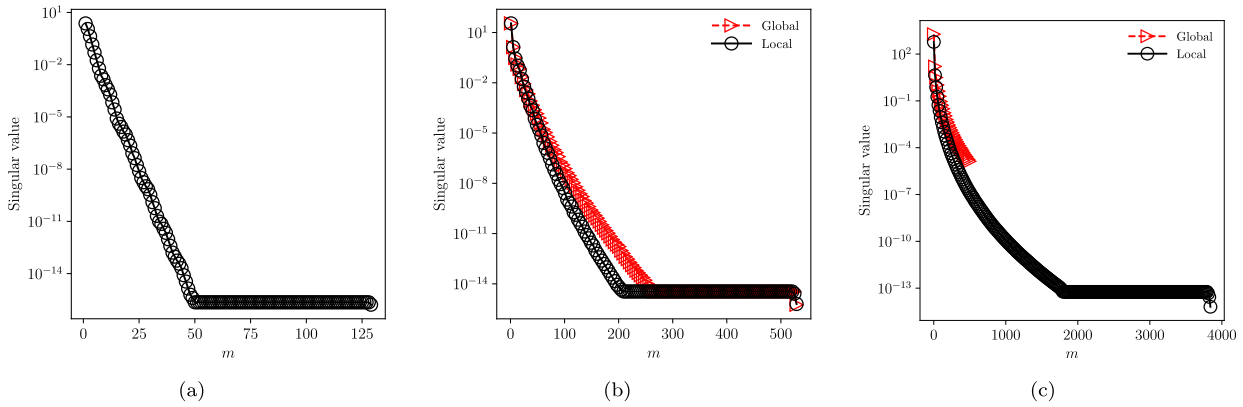

Fig. 14. Singular value decay of the unsteady problems: (a) the Burgers' equation with 10 parameter instances, (b) the advection diffusion reaction problem with 10 parameter instances and (c) the natural convection problem with 60 parameter instances. The basis functions of the unsteady natural convection problem are generated using the truncated randomized SVD, in which only 500 singular values and vectors are extracted.

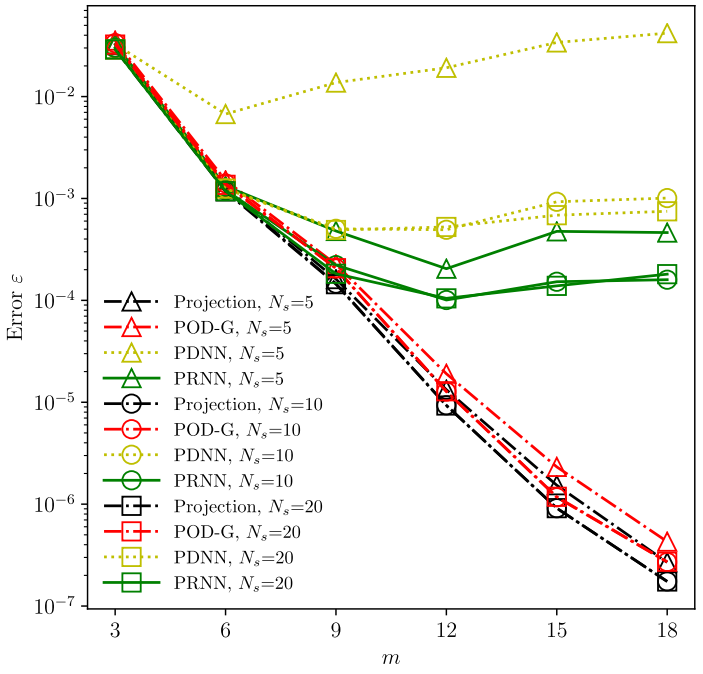

Fig. 15. Prediction error for the unsteady Burgers' equation with different number of parameter instances.

$$\begin{cases} \dfrac{\partial \phi}{\partial t} - K\nabla_{\mathbf{x}}^2\phi + \mathbf{u}\cdot\nabla_{\mathbf{x}}\phi + a_0 u = F & (\mathbf{x},t)\in\Omega\times\mathcal{T} \\ \dfrac{\partial \phi}{\partial \mathbf{n}} = 0 & (\mathbf{x},t)\in\partial\Omega\times\mathcal{T} \\ \phi(\mathbf{x},0) = 0 & \mathbf{x}\in\Omega \end{cases}, \tag{46}$$

in the two-dimensional spatial domain $\Omega = [0,1]^2$ and the temporal domain $\mathcal{T} = [0, 2\pi]$, where $a_0 = 0.01$, $\mathbf{u} = (\cos(t), \sin(t))^T$ and $F = \exp\left(-\frac{(x-0.5)^2+(y-0.5)^2}{0.07^2}\right)$. The parameter of interest is $\boldsymbol{\mu} = K \in [0.05, 0.5]$. High-fidelity snapshots are obtained using the Chebyshev pseudospectral method with 25×25 Chebyshev Gauss-Lobatto points for spatial discretization and a third-order backward difference method with 201 equally spaced points for temporal discretization.

To validate the effectiveness of the local modeling feature of non-intrusive methods, we test the PDNN and PRNN on both the global temporal domain $\mathcal{T} = [0, 2\pi]$ and a local temporal domain $\mathcal{T}_{loc} = [0.2\pi, 2\pi]$.

The reduced basis is extracted from the snapshots computed using $N_s = 10$ parameter instances and the reference value $\widetilde{\boldsymbol{\phi}} = \mathbf{0}$. The singular values decay for both the global and local time domains as shown in Fig. 14(b). The singular values decay slowly for the global time domain, implying that a large number of reduced basis functions is required to recover

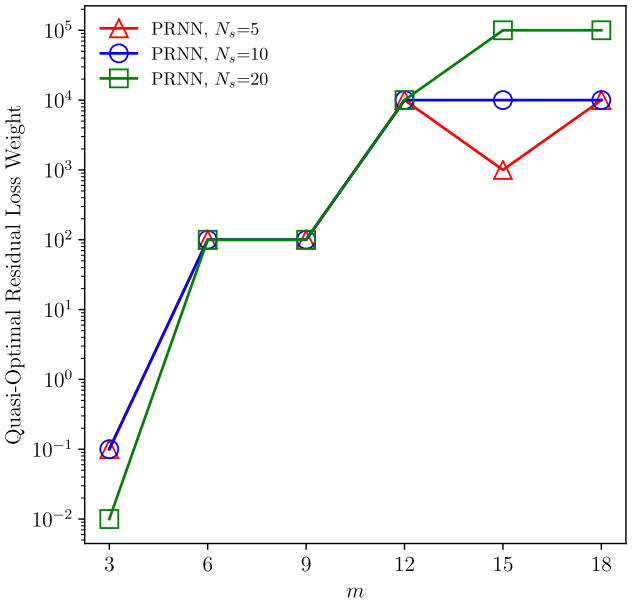

Fig. 16. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}\}$) of the PRNN for the unsteady Burgers' equation.

the main dynamics of the solution. The singular values of the local time domain decay faster than that of the global time domain, featuring the advantage of local modeling. The PDNN and PRNN have the same network architecture with $L = 2$ hidden layers and $n_H = 100$ neurons in each hidden layer. The number of residual points for the PRNN is $N_{Resi} = 2048$. A data set with 10 randomly sampled parameter points and 201 equally spaced time points is employed as the validation data set. A data set with 100 uniformly sampled parameter points and 201 equally spaced time points is used to test the accuracy of the models.

The test error is shown in Fig. 17, and the quasi-optimal residual loss weight for PRNN is shown in Fig. 18. Without closure, the error curve of POD-G lies far from the projection error curve, implying a poor accuracy of POD-G. By adding the closure term, the accuracy of the POD-G is significantly improved, which indicates that the impact of the unresolved scales on the resolved scales can not be neglected. The PRNN follows the "POD-G + closure" closely and is much more accurate than the original POD-G. The PRNN is also more accurate than the PDNN. It is observed in Fig. 17 that both the PDNN and the PRNN achieve much higher accuracy on the local time range than the global time range, which demonstrates the effectiveness of the local modeling strategy.

Predictions by the PRNN for several time-parameter values are compared with the corresponding high-fidelity solutions in Fig. 19 to show the accuracy of the proposed reduced-order method. The PRNN reduced-order solutions agree well with the high-fidelity solutions.

4.2.3. Natural convection

To further validate the capability of the proposed method in handling complex nonlinear problems, we consider the unsteady natural convection enclosed in a cavity, described by the following equation

$$\begin{cases} \nabla_{\mathbf{x}} \cdot \mathbf{u} = 0 & \mathbf{x} \in \Omega \times \mathcal{T} \\ \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla_{\mathbf{x}} \mathbf{u} = -\nabla_{\mathbf{x}} p + \sqrt{\left(\dfrac{Pr}{Ra}\right)} \nabla_{\mathbf{x}}^2 \mathbf{u} - T \mathbf{n}_g & \mathbf{x} \in \Omega \times \mathcal{T} \\ \dfrac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla_{\mathbf{x}} T = \dfrac{1}{\sqrt{(Pr \times Ra)}} \nabla_{\mathbf{x}}^2 T & \mathbf{x} \in \Omega \times \mathcal{T} \\ \mathbf{u} = 0, T = f(t) & \mathbf{x} \in \Gamma_1 \times \mathcal{T} \\ \mathbf{u} = 0, T = -0.5 & \mathbf{x} \in \Gamma_2 \times \mathcal{T} \\ \mathbf{u} = 0, \partial T / \partial \mathbf{n} = 0 & \mathbf{x} \in \Gamma_3 \times \mathcal{T} \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0, p(\mathbf{x}, 0) = p_0, T(\mathbf{x}, 0) = T_0 & \mathbf{x} \in \Omega \end{cases}, \tag{47}$$

where $Pr = 0.71$ is the Prandtl number, $\mathbf{n}_g = (-\sin\theta, -\cos\theta)$ is the direction of gravity. The geometry and boundary conditions are the same as for the steady natural convection problem in Section 4.1.3, except that the heated wall is subjected to a sinusoidal time-dependent temperature

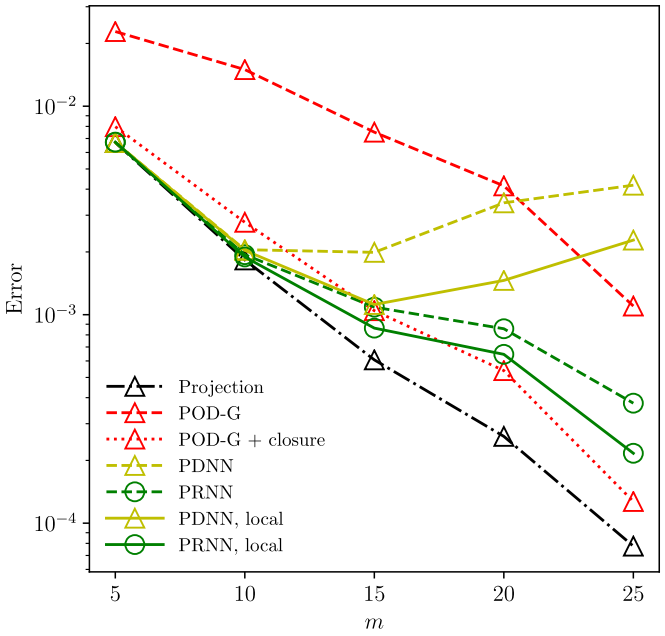

Fig. 17. Prediction error for the advection diffusion reaction problem. The mark "local" denotes the test on the local temporal domain $[0.2\pi, 2\pi]$.

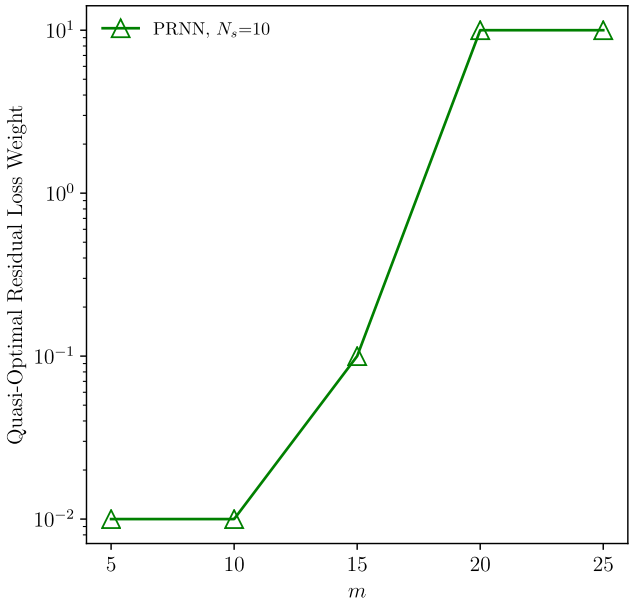

Fig. 18. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$) of the PRNN for the advection diffusion reaction problem.

$$f(t) = 0.5 + a \sin\left(\frac{2\pi t}{P}\right), \tag{48}$$

where the amplitude and the period of the fluctuating temperature are set as $a = 0.8$ and $P = 10$, respectively. The parameter of interest is $\boldsymbol{\mu} = (Ra, \theta) \in [10^4, 3 \times 10^4] \times [5\pi/12, \pi/2]$, and the time interval is $\mathcal{T} = [0, 100]$, corresponding to 10 time periods.

High-fidelity snapshots are obtained using the Chebyshev pseudospectral method with 33×33 Chebyshev Gauss-Lobatto points for the spatial discretization and a third-order backward difference method with 1001 equally spaced points in time for the temporal discretization. To make the problem reducible, the steady-state solution of a special case $(Ra, \theta) = (10^4, \pi/2)$, is employed as the initial condition.

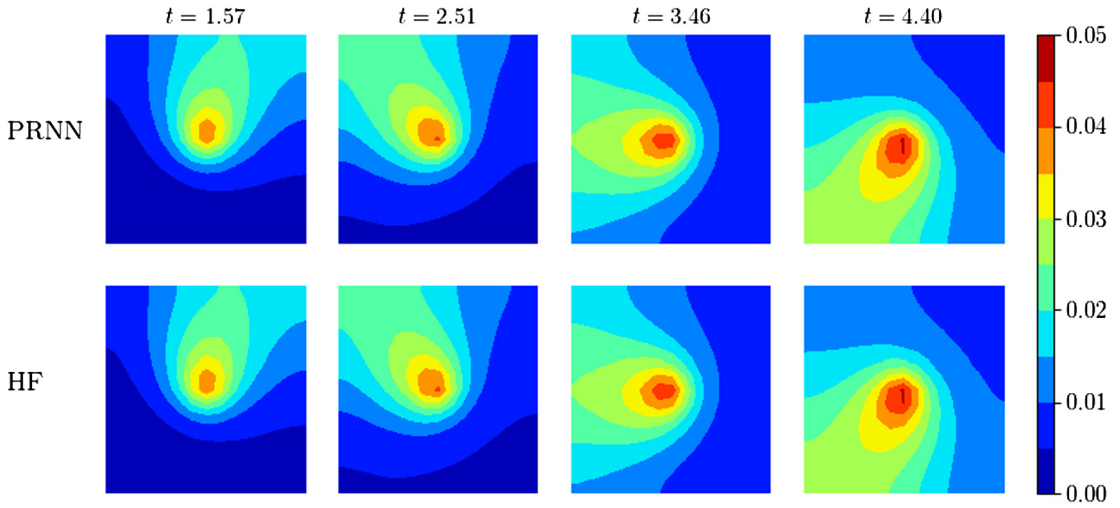

Fig. 19. Contours of the advection diffusion reaction problem computed by the high-fidelity solver and the PRNN for $K = 0.10$. The reduced basis is of size $m = 25$ and extracted from 10 parameter instances.

To validate the effectiveness of the local modeling feature of non-intrusive methods for complex nonlinear problems, we test the PDNN and PRNN on both the global time range (10 time periods) $\mathcal{T} = [0, 100]$ and a local time range (the last time period) $\mathcal{T}_{loc} = [90, 100]$.

The reduced basis is extracted from the snapshots computed using $N_s = 60$ parameter instances. During the reduced basis generation for the global time domain, the reference value $\widetilde{\boldsymbol{\phi}}$ is set as the initial solution. During the reduced basis generation for the local time domain, the reference value $\widetilde{\boldsymbol{\phi}}$ is set as the average of the snapshots. The singular values decay for both the global and local time domains as shown in Fig. 14(c). However, the singular values decay slowly for the global time domain, implying that a large number of reduced basis functions is required to recover the dynamics of the solution. The singular values of the local time domain decay faster than that of the global time domain, featuring the advantage of local modeling. The PDNN and PRNN have the same network architecture with $L = 2$ hidden layers and $n_H = 100$ neurons in each hidden layer. The number of residual points for the PRNN is $N_{Resi} = 2048$. A data set with 30 randomly sampled parameter points and 1001 equally spaced time points is employed as the validation data set. A data set with uniformly sampled $7 \times 7 = 49$ parameter points and 1001 equally spaced time points is used to test the accuracy of the models.

On the global temporal domain, the POD-G is found to diverge eventually for some parameter instances. Therefore, only the results for local modeling are presented in Fig. 20. The corresponding quasi-optimal residual loss weight for PRNN is shown in Fig. 21. For the evolution on the local temporal domain, the POD-G uses the projections of the high-fidelity snapshots onto the reduced space as initial conditions, which is not feasible in real applications as the high-fidelity snapshot at an arbitrary time-parameter instance is not available.

It is shown in Fig. 20 that, without closure, the error of POD-G is large and does not decrease with basis refinement, which indicates that the POD-G fails. By adding the closure term, the accuracy of the POD-G is significantly improved, which indicates that this test case is complex and the closure term plays an important role. The PRNN is slightly more accurate than the "POD-G + closure" and is much more accurate than the PDNN.

Predictions by the PRNN for several time-parameter values are compared with the corresponding high-fidelity solutions in Fig. 22 to show the accuracy of the proposed reduced-order method. The PRNN reduced-order solutions agree well with the high-fidelity solutions.

4.3. Computational cost

The offline and online computational cost for reduced-order modeling of both the steady-state and unsteady natural convection problems, using $m = 30$ reduced basis functions, is listed in Table 3. The high-fidelity simulations are performed serially on the Intel® Xeon® E5 CPU nodes of the Tianhe-2 (China) cluster. All other computations are performed on the IZAR cluster (EPFL, Switzerland), which has 64 nodes and each node has two 20-core Intel® Xeon®-Gold CPUs and two NVIDIA V100 PCIe 32 GB GPUs. Each ANN is trained on IZAR using a single GPU during the offline stage. As the number of restarts and the number of hyper-parameter values in grid-search during the training of PDNN/PINN/PRNN can be set arbitrarily, we only list the offline training cost for one restart in a certain hyper-parameter setting in Table 3 to avoid confusion.

As shown in Table 3, although the offline training of PRNN is more expensive than that of PDNN, the training cost is far less than the cost for high-fidelity snapshots generation. Compared to POD-G, reduced basis method based on PRNN is characterized by its extremely fast online evaluation, demonstrating its efficiency.

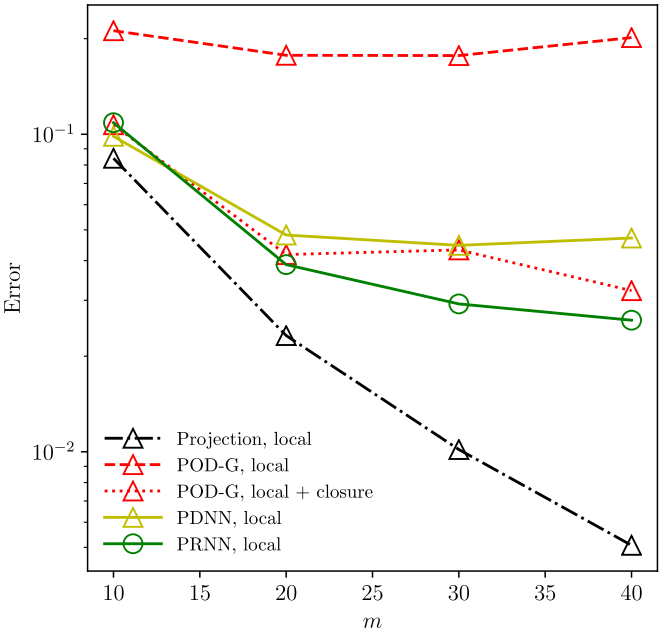

Fig. 20. Prediction error for the unsteady natural convection problem.

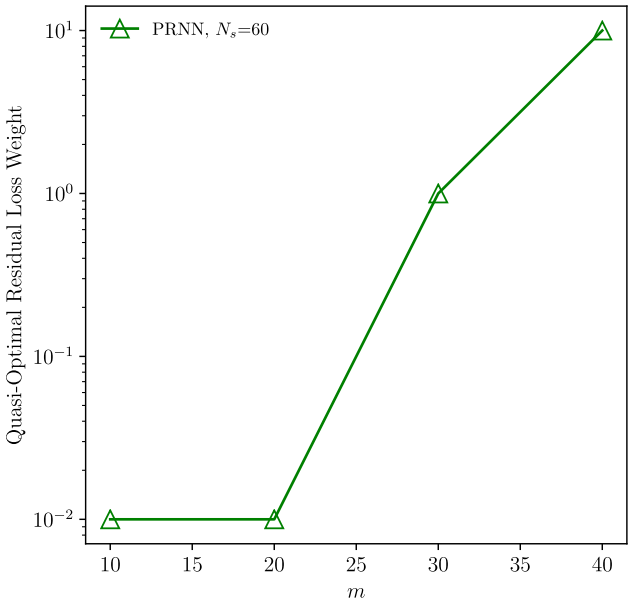

Fig. 21. Quasi-optimal residual loss weight (chosen from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$) of the PRNN for the unsteady natural convection problem.

5. Conclusions

This paper proposes a physics-informed machine learning for reduced-order modeling. The reduced coefficients are predicted by a feedforward neural network. A physics-informed neural network (PINN) can be used to learn the POD-G reduced-order model, by setting the mean squared norm of the residual of the reduced-order equation as the loss function in the training procedure. The PINN is an approximation of the reduced-order model that is obtained without any labeled data. For a more accurate network, the projection data, collected from the high-fidelity snapshots that are used in the reduced basis generation, are introduced into the training to provide more physical information. The physics-reinforced neural network (PRNN) is trained by minimizing the weighted sum of the residual loss of the reduced-order equation and the matching error loss on the projection data set. The physics-informed machine learning technique makes it possible to train an accurate network to predict reduced-order solution, without the requirement of extra high-fidelity simulations.

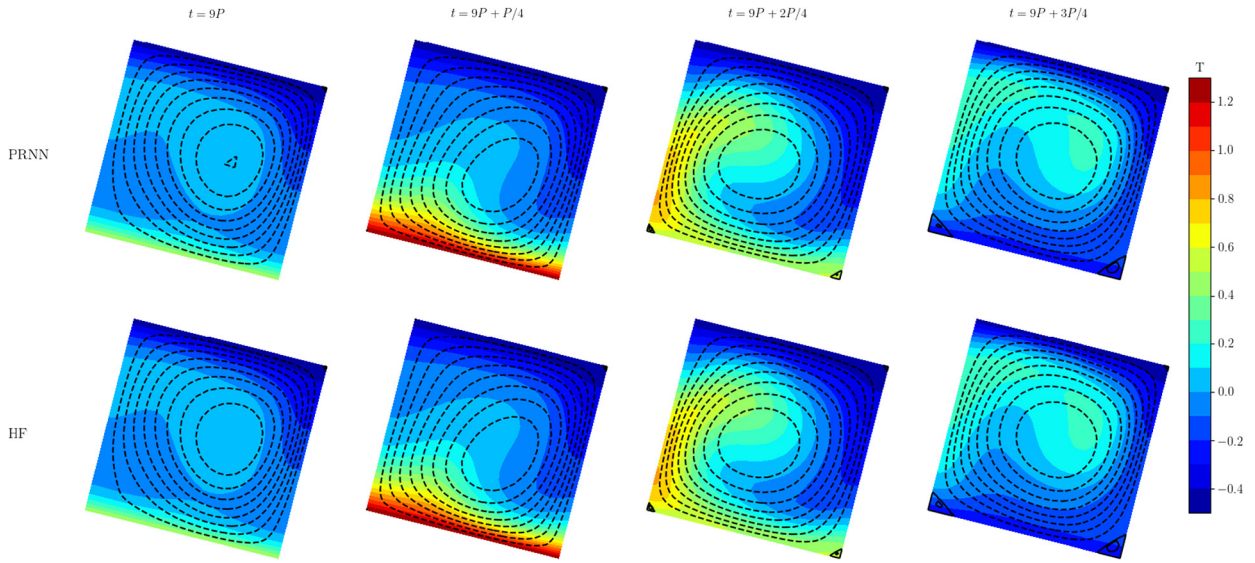

Fig. 22. Temperature contours and streamlines within one period of the unsteady natural convection flow computed by the high-fidelity solver and the PRNN for $\boldsymbol{\mu} = (2.86 \times 10^4, 1.32)$. The reduced basis is of size $m = 30$ and extracted from 60 physical parameters. The solid streamline indicates an anticlockwise vortex and the dashed streamline indicates an clockwise vortex.

Table 3
Computational time cost. The unit of the time cost is second(s). The PDNN, PINN and PRNN have the same network architecture and activation function, thus have the same online cost. Therefore, "NN" is used to represent these networks in terms of online runtime.

			Steady-state	Unsteady
Offline	Snapshots		7.35E4	6.68E5
	POD		1.23E1	1.81E2
	Training	PDNN	1.20E2	1.24E3
		PINN	1.13E3	–
		PRNN	1.37E3	8.95E3
Online	POD-G		2.27E-4	3.78E-1
	POD-G+Closure		–	3.785E-1
	NN		5.06E-6	7.32E-6

Numerical results demonstrate that the PRNN achieves higher accuracy than the POD-G when the reduced basis set is small. The online stage of the physics-informed learning based reduced-order method is efficient as it just requires the evaluation of the network. Therefore, the PRNN is an accurate and efficient reduced-order modeling tool for general nonlinear problems.

CRediT authorship contribution statement

Wenqian Chen: Conceptualization, Investigation, Methodology, Software, Writing – original draft. **Qian Wang:** Conceptualization, Methodology, Supervision, Writing – original draft. **Jan S. Hesthaven:** Resources, Supervision, Writing – review & editing. **Chuhua Zhang:** Funding acquisition, Resources, Writing – review & editing.

Declaration of competing interest

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

Acknowledgements

The first author is financially supported by Xi'an Jiaotong University Graduate Short-term Academic Visiting Program, National Key Research and Development Project of China [Grant number 2016YFB0200901], National Science and Technology Major Project of China [Grant number 2017-II-0006-0020].

References

[1] J.S. Hesthaven, G. Rozza, B. Stamm, et al., Certified reduced basis methods for parametrized partial differential equations, vol. 590, Springer, 2016.
[2] A. Quarteroni, A. Manzoni, F. Negri, Reduced basis methods for partial differential equations: an introduction, vol. 92, Springer, 2015.
[3] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (4) (2015) 483–531.
[4] D.J. Lucia, P.S. Beran, W.A. Silva, Reduced-order modeling: new approaches for computational physics, Prog. Aerosp. Sci. 40 (1–2) (2004) 51–117.
[5] Y. Maday, Reduced basis method for the rapid and reliable solution of partial differential equations, in: Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006, 2007, pp. 1255–1270.
[6] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications – part I: theory, J. Sound Vib. 252 (3) (2002) 527–544.
[7] F. Chinesta, P. Ladeveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Arch. Comput. Methods Eng. 18 (4) (2011) 395.
[8] K. Gallivan, A. Vandendorpe, P. Van Dooren, Model reduction via tangential interpolation, in: MTNS 2002 (15th Symp. on the Mathematical Theory of Networks and Systems), 2002, p. 6.
[9] H. Panzer, J. Mohring, R. Eid, B. Lohmann, Parametric model order reduction by matrix interpolation, Automatisierungstechnik 58 (8) (2010) 475–484.
[10] M. Billaud-Friess, A. Nouy, Dynamical model reduction method for solving parameter-dependent dynamical systems, SIAM J. Sci. Comput. 39 (4) (2017) A1766–A1792.
[11] E. Lappano, F. Naets, W. Desmet, D. Mundo, E. Nijman, A greedy sampling approach for the projection basis construction in parametric model order reduction for structural dynamics models, in: Proceedings of ISMA, 2016, pp. 19–21.
[12] J.S. Hesthaven, B. Stamm, S. Zhang, Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods, Modél. Math. Anal. Numér. 48 (1) (2014) 259–283.
[13] M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. 182 (1) (2002) 1–26.
[14] T. Murata, K. Fukami, K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, J. Fluid Mech. 882 (2020).
[15] K. Lee, K.T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.
[16] C.W. Rowley, T. Colonius, R.M. Murray, Model reduction for compressible flows using pod and Galerkin projection, Physica D: Nonlinear Phenom. 189 (1–2) (2004) 115–129.
[17] Q. Wang, N. Ripamonti, J.S. Hesthaven, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, J. Comput. Phys. (2020) 109402.
[18] A. Deane, I. Kevrekidis, G.E. Karniadakis, S. Orszag, Low-dimensional models for complex geometry flows: application to grooved channels and circular cylinders, Phys. Fluids A, Fluid Dyn. 3 (10) (1991) 2337–2354.
[19] C. Huang, K. Duraisamy, C. Merkle, Challenges in reduced order modeling of reacting flows, in: 2018 Joint Propulsion Conference, 2018, p. 4675.
[20] A. Iollo, S. Lanteri, J.-A. Désidéri, Stability properties of pod–Galerkin approximations for the compressible Navier–Stokes equations, Theor. Comput. Fluid Dyn. 13 (6) (2000) 377–396.
[21] B.M. Afkham, N. Ripamonti, Q. Wang, J.S. Hesthaven, Conservative model order reduction for fluid flow, in: Quantification of Uncertainty: Improving Efficiency and Technology, Springer, 2020, pp. 67–99.
[22] G. Rozza, K. Veroy, On the stability of the reduced basis method for Stokes equations in parametrized domains, Comput. Methods Appl. Mech. Eng. 196 (7) (2007) 1244–1260.
[23] F. Ballarin, A. Manzoni, A. Quarteroni, G. Rozza, Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations, Int. J. Numer. Methods Eng. 102 (5) (2015) 1136–1161.
[24] K. Carlberg, Adaptive h-refinement for reduced-order models, Int. J. Numer. Methods Eng. 102 (5) (2015) 1192–1210.
[25] B. Peherstorfer, K. Willcox, Online adaptive model reduction for nonlinear systems via low-rank updates, SIAM J. Sci. Comput. 37 (4) (2015) A2123–A2150.
[26] R. Abgrall, R. Crisovan, Model reduction using L 1-norm minimization as an application to nonlinear hyperbolic problems, Int. J. Numer. Methods Fluids 87 (12) (2018) 628–651.
[27] K. Carlberg, M. Barone, H. Antil, Galerkin, V: least-squares Petrov–Galerkin projection in nonlinear model reduction, J. Comput. Phys. 330 (2017) 693–734.
[28] S. Grimberg, C. Farhat, N. Youkilis, On the stability of projection-based model order reduction for convection-dominated laminar and turbulent flows, J. Comput. Phys. 419 (2020) 109681.
[29] H. Mori, Transport, collective motion, and Brownian motion, Prog. Theor. Phys. 33 (3) (1965) 423–455.
[30] R. Zwanzig, Nonlinear generalized Langevin equations, J. Stat. Phys. 9 (3) (1973) 215–220.
[31] A.J. Chorin, O.H. Hald, R. Kupferman, Optimal prediction with memory, Physica D, Nonlinear Phenom. 166 (3–4) (2002) 239–257.
[32] A. Chorin, P. Stinis, Problem reduction, renormalization, and memory, Commun. Appl. Math. Comput. Sci. 1 (1) (2007) 1–27.
[33] E.J. Parish, K. Duraisamy, A dynamic subgrid scale model for large eddy simulations based on the Mori–Zwanzig formalism, J. Comput. Phys. 349 (2017) 154–175.
[34] P. Stinis, Renormalized Mori–Zwanzig-reduced models for systems without scale separation, Proc. R. Soc. A, Math. Phys. Eng. Sci. 471 (2176) (2015) 20140446.
[35] O. San, R. Maulik, Extreme learning machine for reduced order modeling of turbulent geophysical flows, Phys. Rev. E 97 (4) (2018) 042322.
[36] O. San, R. Maulik, Neural network closures for nonlinear model order reduction, Adv. Comput. Math. 44 (6) (2018) 1717–1750.
[37] F. Casenave, A. Ern, T. Lelièvre, A nonintrusive reduced basis method applied to aeroacoustic simulations, Adv. Comput. Math. 41 (5) (2015) 961–986.
[38] D. Amsallem, Interpolation on manifolds of CFD-based fluid and finite element-based structural reduced-order models for on-line aeroelastic predictions, Ph.D. thesis, Stanford University, 2010.
[39] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, Adv. Comput. Math. 12 (4) (2000) 273–288.
[40] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.
[41] Q. Wang, J.S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, J. Comput. Phys. 384 (2019) 289–307.
[42] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression, Comput. Methods Appl. Mech. Eng. 341 (2018) 807–826.
[43] M. Guo, J.S. Hesthaven, Data-driven reduced order modeling for time-dependent problems, Comput. Methods Appl. Mech. Eng. 345 (2019) 75–99.
[44] J. Xu, K. Duraisamy, Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, Comput. Methods Appl. Mech. Eng. 372 (2020) 113379.
[45] M. Riedmiller, Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms, Comput. Stand. Interfaces 16 (3) (1994) 265–278.

[46] P. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, R. Vinuesa, Predictions of turbulent shear flows using deep neural networks, Phys. Rev. Fluids 4 (5) (2019) 054603.
[47] F.J. Gonzalez, M. Balajewicz, Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems, preprint, arXiv:1808.01346, 2018.
[48] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, Theor. Comput. Fluid Dyn. 34 (4) (2020) 367–383.
[49] R. Swischuk, B. Kramer, C. Huang, K. Willcox, Learning physics-based reduced-order models for a single-injector combustion process, AIAA J. 58 (6) (2020) 2658–2672.
[50] M. Raissi, P. Perdikaris, 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 (2019) 686–707.
[51] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[52] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Eng. 360 (2020) 112789.
[53] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. (2021) 1–19.
[54] W. Chen, Y. Ju, C. Zhang, A multidomain multigrid pseudospectral method for incompressible flows, Numer. Heat Transf., Part B, Fundam. 74 (1) (2018) 415–431.
[55] W. Chen, Y. Ju, C. Zhang, A parallel inverted dual time stepping method for unsteady incompressible fluid flow and heat transfer problems, Comput. Phys. Commun. (2020) 107325.
[56] W. Zhang, C. Zhang, G. Xi, An explicit Chebyshev pseudospectral multigrid method for incompressible Navier–Stokes equations, Comput. Fluids 39 (1) (2010) 178–188.
[57] R. Peyret, Spectral methods for incompressible viscous flow, vol. 148, Springer Science & Business Media, 2013.
[58] B. Unger, S. Gugercin, Kolmogorov n-widths for linear dynamical systems, Adv. Comput. Math. 45 (5) (2019) 2273–2286.
[59] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[60] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[61] M. Barrault, Y. Maday, N.C. Nguyen, A.T. Patera, An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations, C. R. Math. 339 (9) (2004) 667–672.
[62] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764.
[63] R. Everson, L. Sirovich, Karhunen–Loeve procedure for gappy data, JOSA A 12 (8) (1995) 1657–1664.
[64] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Trans. Autom. Control 53 (10) (2008) 2237–2251.
[65] W. Cazemier, R. Verstappen, A. Veldman, Proper orthogonal decomposition and low-dimensional models for driven cavity flows, Phys. Fluids 10 (7) (1998) 1685–1699.
[66] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, preprint, arXiv:1710.05941, 2017.
[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[68] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, preprint, arXiv:1412.6980, 2014.
[69] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1) (1989) 503–528.
[70] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, preprint, arXiv:1502.03167, 2015.
[71] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Eng. 361 (2020) 112732.
[72] J.R. Clausen, Entropically damped form of artificial compressibility for explicit simulation of incompressible flow, Phys. Rev. E 87 (1) (2013) 013309.