Physics-informed machine learning for reduced-order modelling

Wenqian Chen[a], 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*

Abstract

A physics-informed machine learning framework is developed for the reduced-order modeling of parametirzed steady-state partial differential equations (PDEs). In the offline stage, a reduced-order model is constructed by projecting the high-fidelity numerical model onto a reduced space, spanned by a set of reduced basis that represents the main dynamics of the full-order model. A physics-informed neural network (PINN) and a physics-reinforced neural network (PRNN) are developed to approximate the mapping from the parameters to the projection coefficients. The former is trained by minimizing the residual of the reduced-order model, while the latter is trained by minimizing the residual in combination with the matching error between the projection coefficients and network outputs. In the online stage, given a new parameter location, the projection coefficients are predicted by the neural networks. Numerical results demonstrate that the PINN and PRNN can provide more accurate solutions to nonlinear problems with fewer high-fidelity solutions, compared with a non-intrusive reduced-order method. Besides, the PRNN are validated more accurate and robust than the PINN for complex nonlinear problems.

Keywords: physics-informed machine learning, Neural network, reduced-order modeling, nonlinear PDE

1. Introduction

In recent years, although high-fidelity numerical method has witnessed substantial developments for modeling complex phenomena, it is still challenging for many applications, such as design, optimization, control and uncertainty quantification problems. Such applications are governed by parameterized partial differential equations(PDEs) [1, 2], where parameters characterize underlying geometry, boundary conditions, source terms and physical properties, etc. High-fidelity numerical method are prohibitively expensive to such applications in multi-entry and real-time contexts, due to the required lots of degrees of freedom(DOFs). The need to largely reduce computational cost gives birth to reduced-order modeling methods [3], aiming at replacing the full-order system with a cheap reduced-order system with an affordable loss of accuracy.

Reduced basis (RB) method is one of the well-known and widely-used reduced-order modeling methods, and always follows the offline-online paradigm [4]. The offline stage comprises preparing high-fidelity solutions(snapshots), extracting the reduced basis and building a cheap evaluation model. The reduced space, spanned by the reduced basis extracted from the snapshots, is expected to represent the main dynamics of the full-order model. There exist many methods to extract the

*Corresponding author.
Email addresses: wenqianchen2016@gmail.com (Wenqian Chen), qian.wang@epfl.ch (Qian Wang), Jan.Hesthaven@epfl.ch (Jan S. Hesthaven), chzhang@mail.xjtu.edu.cn (Chuhua Zhang)

reduced basis, such as the Proper Orthogonal Decomposition (POD) [1, 2, 5], the Proper Generalized Decomposition [6], the Piecewise Tangential Interpolation [7], the Matrix Interpolation [8], greedy algorithms[9–11]. Among them, the POD perhaps is the most widely-known and widely-used one, where the reduced basis is generated by a singular value decomposition (SVD) algorithm for some given snapshots with respect to a chosen set of parameters. After the reduced basis has been properly set up, an approximation to high-fidelity solution is recovered from projection coefficients of the reduced space. Then a suitable model will be built offline, aiming to provide online evaluation for a new given parameter. For building such a model, both 'white-box' and 'black-box' strategies are applied in the literature.

Galerkin method [12–14] is one of widely-used white-box strategies. In the offline stage, a reduced-order model determining projection coefficients is derived from Galerkin projection of the full-order model. In the online stage, the reduced-order model is solved by a suitable numerical solver for each parameter. As projection coefficients are directly solved by a deterministic method, the achieved approximation accuracy is always very good. What's more, no more extra snapshots is required besides the snapshots for building RB. However, the online solving of the reduced-order model is not trivial, which remains an open problem with respect to its accuracy, stability and convergence [12, 15, 16]. Besides, the online solving of the reduced-order model is still time consuming especially for complex nonlinear problems, even if it is much cheaper than the solving of full-order model.

The black-box strategy is also termed as an non-intrusive procedure, in which the building and also the solving of reduced-order model is avoided. Usually, the high-fidelity simulations producing snapshots are treated as a black box and recovered with a machine learning model. During the offline stage, a map between the parameter values and the projection coefficients is built by a data-driven way. During the online stage, the projection coefficients for new parameter values can be recovered as outputs of the machine learning model. For example, non-intrusive approaches using radial basis function (RBF) [17–19], artificial neural network(ANN) [20] and Gaussion process regression(GPR) [21] have been applied successively for parameterized partial differential equations, even for flow problems of strong nonlinearity. These approaches benefit from their high established effectiveness and efficient online evaluation. However, the drawback, common for data-driven method, is their requirement for an excessive number of observed data to guarantee accuracy. That is to say, many high-fidelity solutions have to be precomputed, sometimes leading to an unacceptable cost even if it is performed in the offline stage [20].

The white-box Galerkin method targets the full-order model, while the black-box machine learning models target high-fidelity solutions. These two strategies approximate the parameterized PDEs in different directions. In this work, we try to develop a physics-informed machine learning framework using a grey-box strategy to take the advantages of the above two strategies. Inspired by the well-know physics-informed neural network (PINN) [22–24], we can directly solve the reduced-order model by optimizing a neural network. Thus, we keep on the intrusive approach and utilize the reduced-order information to build a neural network. For the PINN approach, there is no need for labeled data, thus no need for extra snapshots, and what we need to do is to train a network to satisfy the reduced-order model in some sense. Besides this approach, the projection coefficients of snapshots can also be added into the training of network, although they are not the solutions of the reduced-order model. we term this approach as the physics-reinforced neural network (PRNN). For both PINN and PRNN approaches, they enjoy a higher accuracy and independence of extra snapshots inherited from Galerkin method, and also the effectiveness inherited from machine learning models. To test this assertion, the PINN and PRNN are applied to one-dimensional Burges' equation, two-dimensional lid-driven flow and natural convection in enclosure. A detailed comparison with the white-box POD-G method and the black-box POD-NN method proposed by Jan et al.[20] is analysed for the three test cases.

The remainder of this paper is organized as follows. Section 2 presents the framework from parameterized PDEs to the reduced-order model. Section 3 briefly introduce the POD-NN method

(referred to PDNN in this work) and presents the derivation of PINN and PRNN method based on the reduced-order information. Section 4 details some numerical setups for these methods, and offers some numerical results to show the superiority of the proposed PINN and PRNN methods, In Section 5, some conclusions are given.

2. Projection-based reduced basis method

Consider a nonlinear dynamic system governed by the following parameterized partial differential equations(PDEs):

$$\begin{aligned} \mathcal{N}\left(\phi\left(\mathbf{x}\right);\boldsymbol{\mu}\right) &= 0, \qquad \mathbf{x} \in \Omega\left(\boldsymbol{\mu}\right), \\ \mathcal{B}\left(\phi\left(\mathbf{x}\right);\boldsymbol{\mu}\right) &= 0, \qquad \mathbf{x} \in \partial\Omega\left(\boldsymbol{\mu}\right), \end{aligned} \tag{1}$$

where $\mathcal{N}$ is a general nonlinear differential operator, $\phi\left(\mathbf{x}\right)$ are field variables to be solved on Cartesian coordinates $\mathbf{x}$. $\mathcal{B}$ are operators defining boundary conditions for the boundary $\partial\Omega$ of the physical domain Ω. $\boldsymbol{\mu} \in \mathcal{P}$ are specific parameters characterizing the nonlinear dynamic system as well as the physical domain Ω, where $\mathcal{P}$ is the parameter space.

As is always the case for reduced-order methods, a dynamic system is approximated by finding the its projection in a space spanned by a few well-chosen basis vectors. These basis are generated from a suitable linear combination of some precomputed high-fidelity approximations, termed as snapshots. To ensure the compatibility among snapshots, the variable physical domain has to be addressed. To this end, the variable physical domain will be mapped into a fixed computational domain with the same collation of degrees of freedom. This can be implemented by an invertible problem-dependent mapping $\mathcal{X} : \mathbf{x} \in \Omega\left(\boldsymbol{\mu}\right) \rightarrow \boldsymbol{\xi} \in \widetilde{\Omega}$, which reads

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

and thus the governing equations (1) are recast as follows:

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

where $\widetilde{\Omega}$ is the computational domain, $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{N}}$ are the derived operators in computational space resulting from the transformation Eq. (2).

2.1. Full-order model

To simulate the nonlinear dynamic system in Eqs. (3), the including spatial derivatives are always first discretized by a suitable high-fidelity (HF) method (such as the Chebyshev pseudospectral method [25, 26], spectral difference method [27], etc.). To achieve a satisfactory numerical accuracy, a fine mesh is employed with a large number of degrees of freedom (DOFs), generally including N interior DOFs and N_B boundary DOFs. Finally, the resulting discretized equations for Eqs. (3) are solved with a suitable explicit/implicit solver.

Here we denote the discrete solutions of Eq (3) in vector form $\boldsymbol{\phi}_h \in \mathbf{R}^N$ and $\boldsymbol{\phi}_h^B \in \mathbf{R}^{N_B}$ for parameter $\boldsymbol{\mu} \in \mathcal{P}$. Given a suitable HF method, we have the governing Eq. (3) in discrete form

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

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

where $\mathbf{L}_{\widetilde{\mathcal{N}}} \in \mathbf{R}^{N\times N}$, $\mathbf{L}_{\widetilde{\mathcal{B}}} \in \mathbf{R}^{N_B\times N}$ and $\mathbf{L}_{\widetilde{\mathcal{B}}}^B \in \mathbf{R}^{N_B\times N_B}$ represent matrixes derived from linear parts of operators $\widetilde{\mathcal{N}}$ and $\widetilde{\mathcal{B}}$. $\mathbf{C}_{\widetilde{\mathcal{N}}} \in \mathbf{R}^N$ and $\mathbf{C}_{\widetilde{\mathcal{B}}} \in \mathbf{R}^{N_B}$ are constant vectors independent of $\boldsymbol{\phi}_h$ and $\boldsymbol{\phi}_h^B$. $g_{\widetilde{\mathcal{N}}} : \mathbf{R}^N \times \mathbf{R}^{N_B} \mapsto \mathbf{R}^N$ is a nonlinear function derived from the nonlinear part of operator $\widetilde{\mathcal{N}}$.

For the treatment of boundary conditions, it can be classified into two categories, namely weakly enforced and strongly enforced boundary conditions. As for weakly enforced boundary conditions, no DOFs is required to deploy on the boundaries, i.e. $N_B = 0$, implying a vanishing of $\boldsymbol{\phi}_h^B$ and also Eq. (5). As for strongly enforced boundary conditions, a suitable collocation of DOFs on boundaries is necessary. Note that the commonly used boundary conditions always linear ones, such as Dirictlet, Neumann and Robin conditions, and thus $\mathcal{B}$ are always linear operators. Without loss of generality, we restrict ourselves that the mapping in Eq. (2) will retain the linearity of boundary condition operator $\widetilde{\mathcal{B}}$ in computational space. Therefore, discretization of the boundary conditions result in a linear system of equations, i.e., Eq. (5).

In what remains, the Chebyshev pseudospectral method, characterized by its outstanding accuracy, is chosen for discretizing Eqs. (3), which will be detailed in the following subsection.

2.1.1. Chebyshev pseudospectral method

In the Chebyshev pseudospectral method, the d-dimensional variable physical domain of will be mapped into a unit regular computational domain $\widetilde{\Omega} = [-1, 1]^d$, each dimension discretized by $N_p + 1$ Chebyshev-Gauss-Lobatto points,

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

Spatial derivatives in operators $\widetilde{\mathcal{N}}$ and $\widetilde{\mathcal{B}}$ are approximated with a matrix-vector multiplication in each dimension. In each dimension, the derivatives are approximated as follows:

$$\left.\frac{\partial^s \phi}{\partial \xi^s}\right|_{\xi_i} = \mathbf{D}^s \boldsymbol{\phi}, \tag{7}$$

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

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

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

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

For the following simulations of flow problems, the well known $IP_N - IP_{N-2}$ method [28] is adopted to prevent spurious pressure mode from contaminating the flow fields [29]. In the $IP_N - IP_{N-2}$ method, pressure derivative is approximated without pressure values at boundary points, and thus pressure is approximated with polynomial of two order lower than all other field variables, such as velocity and temperature. All other variables are discretized with Eqs. (8)-(10)

except that the first-order difference matrix $\mathbf{D}^1$ of pressure is replaced with $\hat{\mathbf{D}}^1$ as follows

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

For more details of the Chebyshev pseudospectral method, we refer the reader to the reference [25, 26, 29].

2.2. Reduced-order model

As for solving the problem in Eq. (3), full-order model is always suffering from large computational cost since the required degrees of freedom N is always very large. Therefore, we are interested in replacing the full-order model with a well-posed reduced-order model to represent main dynamic of the system. The full-order model is termed as reducible when high-fidelity solution $\boldsymbol{\phi}_h$ can be well approximated in m-dimensional subspace $\mathcal{V} = \mathbf{R}^{N\times m}$. The subspace is spanned by a suitable set of basis vectors $\{\mathbf{V}_i \in \mathbf{R}^N\}_{i=1}^m$. Note that the the size of problem will be largely reduced, only if $m \ll N$. The full-order solution of the interior DOFs $\boldsymbol{\phi}_h$ can be projected into the subspace $\mathcal{V}$ by

$$\boldsymbol{\phi}_h = \mathbf{V}\boldsymbol{\alpha} + \tilde{\phi} + \boldsymbol{\epsilon} \quad \approx \mathbf{V}\boldsymbol{\alpha} + \tilde{\phi}, \tag{12}$$

where $\mathbf{V} \in \mathbf{R}^{N\times m}$ is a matrix with the basis vectors as its columns, $\boldsymbol{\alpha}$ is the projection coefficients, $\boldsymbol{\epsilon}$ is the projection error. $\tilde{\phi} \in \mathbf{R}$ is independent of $\boldsymbol{\mu}$, intended for filtering a constant value to avoid its domination on the basis, and usually it can be set as the average. The boundary conditions in Eq. (5) say that the interior DOFs have a linear relationship with interior DOFs. Thus, we can enforce the boundary conditions without any reduction. The procedure keeps the reduced-order model of high accuracy at bounaries and avoid shifting of boundary values. According to Eq.(5), the boundary DOFs can be denoted as follows

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

Substituting Eq. (12) and (13) into Eq. (4), we have the overdetermined system for interior DOFs

$$\mathbf{L}_{\widetilde{\mathcal{N}}}\mathbf{V}\boldsymbol{\alpha} + \mathbf{g}_{\widetilde{\mathcal{N}}}\left(\mathbf{V}\boldsymbol{\alpha} + \tilde{\phi} + \boldsymbol{\epsilon}, -\left(\mathbf{L}_{\widetilde{\mathcal{B}}}^B\right)^{-1}\left(\mathbf{L}_{\widetilde{\mathcal{B}}}\mathbf{V}\boldsymbol{\alpha} + \mathbf{C}_{\widetilde{\mathcal{B}}} + \mathbf{L}_{\widetilde{\mathcal{B}}}\mathbf{I}\tilde{\phi} + \mathbf{L}_{\widetilde{\mathcal{B}}}\boldsymbol{\epsilon}\right)\right) + \mathbf{L}_{\widetilde{\mathcal{N}}}\mathbf{I}\tilde{\phi} + \mathbf{L}_{\widetilde{\mathcal{N}}}\boldsymbol{\epsilon} + \mathbf{C}_{\widetilde{\mathcal{N}}} = 0, \tag{14}$$

where $\mathbf{I} \in \mathbf{R}^N$ is an vector filled with 1. For the sake of clarity, Eq. (14) can be rearranged in the simplified form

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

where $\mathbf{L} \in \mathbf{R}^{N\times N}$, $\mathbf{C} \in \mathbf{R}^N$ and $g : \mathbf{R}^N \mapsto \mathbf{R}^N$ are a matrix, vector and function derived from Eq. (14), respectively. The quantity $\tilde{\boldsymbol{\epsilon}}$ represent the residual resulting from the projection error $\boldsymbol{\epsilon}$. In the framework of Petrov-Galerkin projection, if we consider a basis $\mathbf{W} \in \mathbf{R}^{N\times m}$ that is orthogonal to the residual, the overdetermined system is reduced to a system of m equations

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

In this work, we employ the Galerkin framework, namely $\mathbf{W}$=$\mathbf{V}$. Thus, we have the reduced-order model as follows:

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

As discussed above, the key point of the reduced-order model is to get a suitable reduced basis. In the literature, there exist many methods to achieve it. Such methods includes, but not limited to, the Proper Orthogonal Decomposition (POD) [1, 2, 5], the Proper Generalized Decomposition [6], the Piecewise Tangential Interpolation [7], the Matrix Interpolation [8], greedy algorithms[9–11]. Among these methods, the POD, perhaps the most widely used method, is described and employed in the following.

2.2.1. POD reduced basis

The Proper Orthogonal Decomposition (POD) is one of the most widely-used techniques to compress data and extract fundamental information by building a series of orthogonal basis with decreasing energy distribution. Let $\mathcal{P}_M = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_M\} \subset \mathcal{P}$ be a discrete and finite set of M parameters generated with a suitable sampling method, and $\boldsymbol{\Phi}_{\mathcal{P}_M} = \{\boldsymbol{\phi}_h(\boldsymbol{\mu}_1), \boldsymbol{\phi}_h(\boldsymbol{\mu}_2), ..., \boldsymbol{\phi}_h(\boldsymbol{\mu}_M)\} \in \mathbf{R}^{N\times M}$ be the corresponding snapshot matrix obtained by full-order method. Consider that snapshot matrix $\boldsymbol{\Phi}_{\mathcal{P}_M}$ can represent the underlying physics dynamics of the problem. The idea behind POD technique is to find a suitable orthogonal basis matrix $\mathbf{V}$ to minimise the problem defined as follows:

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

where $\mathbf{E}$ is the identify matrix of size m and $\|\cdot\|_F$ is the Frobenius norm. According to the Eckart-Young theorem [30], the solution of Eq. (18) is exactly the first mth left singular vectors of the matrix $\boldsymbol{\Phi}_{\mathcal{P}_M}$, derived from singular value decomposition (SVD) which reads

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

where the singular values σ_i, $1 \le i \le \min(N, M)$ are sorted in a decreasing order. Choosing the first mth columns of $\mathbf{U}_S$, we have an error estimation of Eq. (18) as

$$\min_{\mathbf{V}\in\mathbf{R}^{N\times m}} \left\|\boldsymbol{\Phi}_{\mathcal{P}_M} - \mathbf{V}\mathbf{V}^T\boldsymbol{\Phi}_{\mathcal{P}_M}\right\|_F^2 = \sum_{i=m+1}^{\min(N,M)} \sigma_i^2. \tag{20}$$

In Eq. (20), it is shown that the error is exactly made up with the squares of the neglected singular values. That is to say, with a suitable m, we can approximate the snapshot matrix $\boldsymbol{\Phi}_{\mathcal{P}_M}$ at an arbitrary accuracy. Luckily, most problems exhibit an exponentially decaying series of singular values, and thus we can choose a very small value of m to approximate the problems with a good accuracy.

2.2.2. Further reduction in the cost of reduced-order model

Although we have the reduced-order model defined in Eq. (17), the computational cost is still very expensive, scaling with the original size of full-order model. Thus, a further reduction of computational cost of reduced-order model is required. For linear parts in Eq. (17), they can be treated as follows

$$\begin{aligned} \mathbf{V}^T\mathbf{L}\mathbf{V}\boldsymbol{\alpha} &= \tilde{\mathbf{L}}\boldsymbol{\alpha}, \\ \mathbf{V}^T\mathbf{C} &= \tilde{\mathbf{C}}, \end{aligned} \tag{21}$$

where $\tilde{\mathbf{L}} \in \mathbf{R}^{m\times m}$ and $\tilde{\mathbf{C}} \in \mathbf{R}^m$ can only be computed once with the knowledge of basis matrix $\mathbf{V}$. Therefore, the calculation of linear parts scales with m. However, the drawback of the reduced-order model is that the cost of exact evaluation of nonlinear part $\mathbf{V}^T g\left(\mathbf{V}\boldsymbol{\alpha}\right)$ in Eq. (17) scales with N, namely the size of full-order model. To tackle this defect, several hyper-reduction methods have been developed in the last decades to enable significant speedups for nonlinear part. These methods tries to find the optimal tradeoff between accuracy and efficiency. Such methods include but not limited

to the Empirical Interpolation method (EIM) [31], its discrete variant (DEIM) [32], Gappy-POD [33]and Missing Point Estimation (MPE) [34]. In the present work, we restrict ourselves to quadratic nonlinearity, which can be exactly transformed to a quadratic form, and thus hyper-reduction is avoided. Consider a common quadratic nonlinearity

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

where $\otimes$ denotes the element-wise multiplication operator. Substituting Eq. (22) into Eq. (17), the nonlinear part can be transformed as

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

where $\mathbf{E}_k \in \mathbf{R}^m$ is kth column of the unit matrix $\mathbf{E}$, $\mathbf{A}^k \in \mathbf{R}^{m\times m}$ is calculated as

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

where $\mathbf{V}_i$, $\mathbf{V}_j$ and $\mathbf{V}_k$ are the ith, jth and kth columns of $\mathbf{V}$, respectively.

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

This section presents several methods, aiming to find the mapping from parameters to high-fidelity solutions. Based on the reduced basis, we just need to find the mapping from parameters to projection coefficients, namely the projection of high-fidelity solutions onto reduced basis. These methods are classified into intrusive and non-intrusive methods. We term it as a non-intrusive method if the reduced-order model won't need to be built, otherwise it is an intrusive method. As for non-intrusive methods in reference [20], the underlying idea is an interpolation, such as the projection-driven neural network (PDNN) proposed (referred to POD-NN in [20]) and the traditional cubic spline method. These non-intrusive methods can achieve a good approximation with very little online cost, but the drawback is their highly relying on the size of labeled data set, namely the number of evaluations of full-order model. For example, the traditional cubic spline method requires high-fidelity solutions with a tensor-product grid in parameter space, and the POD-NN are trained with a labeled data set. According to Jan [20], the PDNN has a good reduction in the eagerness of more labeled data compared with the cubic spline method. Even so, the required size of labeled data for the PDNN is prohibitively larger than the size of snapshots for building a satisfactory reduced basis.

A good alternate is the intrusive method, feathering a much cheaper computational cost of the reduced-order model. One can directly solve the reduced-order model with a suitable nonlinear solver, such as Newton-like method. However, it is also a tough work to find the right solution especially for a larger number of reduced-order model, as a reasonable initial solution is not always available. To this end, we combine artificial network (ANN) and the reduced-order model together together, benefiting from the cheap cost of reduced-order mode and powerful approximating capability of ANN. We first propose to solve the reduced-order model with ANN, referred to physics-informed network (PINN). Then we try to add more prior knowledge into reinforce the PINN, referred to physics-reinforced neural network (PRNN). For the sake of clarity, the relationships of these methods are illustrated in Fig. 1.

In the next subsections, we first briefly introduce the basic idea of artificial neural networks. Then we will start from the projection-driven neural networks proposed by Jan [20]. After that, the proposed physics-inform neural network and physics-reinforced neural network will be detailed.

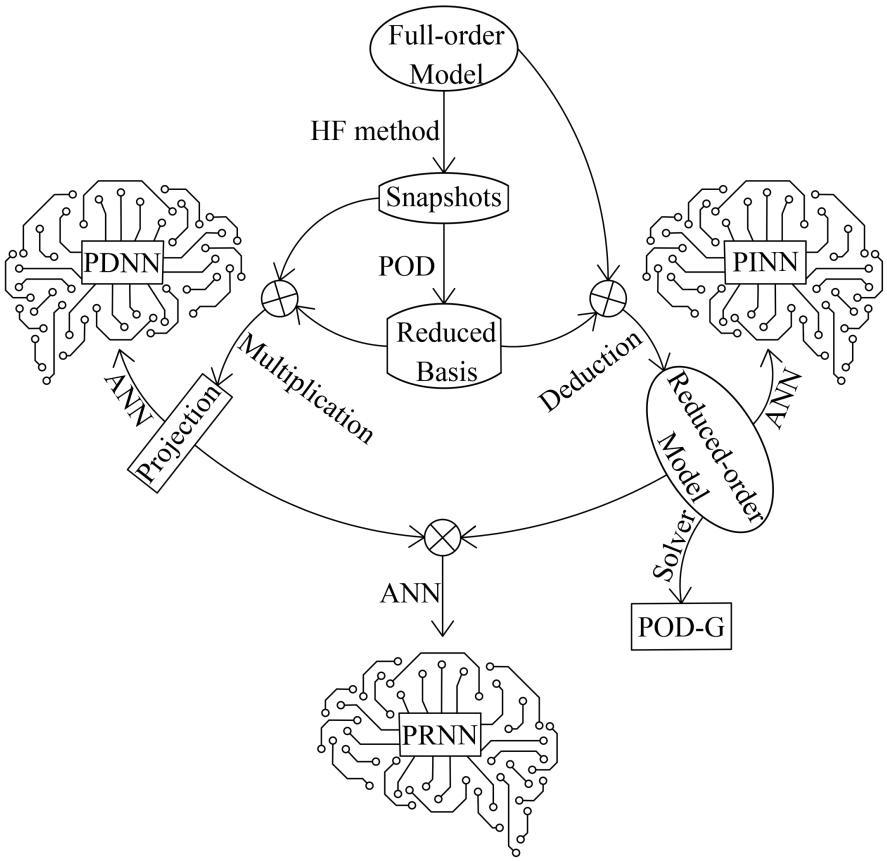

Figure 1: *Relationship among the projection-driven neural network (PDNN), physics-informed network (PINN) and physics-reinforced neural network (PRNN).*

3.1. Artificial neural network (ANN)

For an arbitrary target function, ANN is a powerful nonlinear approximator constituted of a linear combination of some nonlinear functions. Generally, ANN contains L hidden layers besides input and output layers, the ith layer equipped with n_l neurons, where $l = 0, 1, ..L, L+1$ denotes the input layer, 1st hidden layer,..., Lth hidden layer and the output layer, respectively. ANN can be expressed as a nonlinear function $\mathcal{O} : \mathbf{R}^{n_0} \mapsto \mathbf{R}^{n_{L+1}}$ of inputs $\mathbf{x}$, comprising a recursion process

$$\begin{cases} \mathcal{O}^0 = \mathbf{x} \\ \mathcal{O}^l = \sigma^l \left(\mathbf{W}^l \mathcal{O}^{l-1} + \mathbf{b}^l \right) \qquad 1 \leq l \leq L+1 \end{cases}, \tag{25}$$

where $\mathcal{O}^l \in \mathbf{R}^{n_l}$ is the output of lth layer, n_0 is the input dimension and n_{L+1} is the output dimension, $\mathbf{W}^l \in \mathbf{R}^{n_l \times n_{l-1}}$ is the weights, $\mathbf{b}^l \in \mathbf{R}^{n_l}$ is the biases and σ^l is an element-wise activation function. Usually, the activation function of the output layer is just set as the identification function, namely $\sigma^{L+1}(\mathbf{x}) = \mathbf{x}$, while the activation function of all hidden layers are set as a same nonlinear function.

In this work, we simply set the activation function as Swish function [35] $\text{Swish}(x) = x \times \text{sigmoid}(x)$ and use the same number of neurons in each hidden layer, namely $n_1 = n_2 =, ..., = n_L = n_H$, if not stated otherwise. For a specific ANN architecture (L the network depth and n_H the network width), the weights and biases of ANN are trained to minimize the discrepancy between ANN outputs and

targets for the given inputs, where the discrepancy is metricized by a scalar loss function. Thus the training of ANN is essentially an single-objective minimization problem

$$\arg\min_{\mathbf{W},\mathbf{b}} \; loss\left(\mathcal{O}\left(\mathbf{x}\right),\mathcal{O}_{\text{target}}\left(\mathbf{x}\right)\right). \tag{26}$$

However, optimizing the minimization problem is not trivial. As for approximating a complex nonlinear target function with high dimensional inputs/outputs, an ANN of large depth and width is always necessary, resulting in a much larger size of independent variables in $\mathbf{W},\mathbf{b}$, where $\mathbf{W}=\{\mathbf{W}_i\}_{i=1}^{L+1}$ and $\mathbf{b}=\{\mathbf{b}_i\}_{i=1}^{L+1}$. Thus many numerical issues will emerge, among which local minima and overfitting traps are the most common cases. Luckily, many training techniques have been developed in last decades, such as Mini-Batch method to relieve local minima trap and regulation method to relieve overfitting trap.

3.2. Customized networks

Some preparation need to be made before introducing the customized networks. Recall that the snapshot set $\mathbf{\Phi}_{\mathcal{P}_M}=\{\boldsymbol{\phi}_h(\boldsymbol{\mu}_1),\boldsymbol{\phi}_h(\boldsymbol{\mu}_2),...,\boldsymbol{\phi}_h(\boldsymbol{\mu}_M)\}\in\mathbf{R}^{N\times M}$ corresponding with the parameter set $\mathcal{P}_M=\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,...,\boldsymbol{\mu}_M\}\subset\mathcal{P}$ is employed to extract the reduced basis $\mathbf{V}$, according to Section 2.2. Due to the orthogonality of $\mathbf{V}$, the projection coefficient of each snapshot on the reduced space is derived by left multiplying Eq. (12) with $\mathbf{V}^T$, namely

$$\begin{aligned}\boldsymbol{\alpha}&=\mathbf{V}^T(\boldsymbol{\phi}_h-\tilde{\phi}-\boldsymbol{\epsilon})\\&\approx\mathbf{V}^T(\boldsymbol{\phi}_h-\tilde{\phi}).\end{aligned} \tag{27}$$

All the parameters in $\mathcal{P}_M$ along with the projections of their corresponding snapshots, denoted by $\mathcal{D}_{Pr}=\left\{\left(\boldsymbol{\mu}_i,\boldsymbol{\alpha}|_{\boldsymbol{\mu}=\boldsymbol{\mu}_i}\right)\right\}_{i=1}^{M}$, is collected as an input-output data set. As the projection is the best approximation to the snapshot with only the projection error, the data set $\mathcal{D}_{Pr}$ is the best choice for training an ANN.

As ANN is sensitive to the difference in dimensional scale of inputs/outputs [36], the data set $\mathcal{D}_{Pr}$ needs to be further normalized before feeding it into the ANN. For inputs, namely $\boldsymbol{\mu}\in\mathcal{P}$, as the range of the parameter space is always acknowledged, the minimal and maximal of the parameter space can be utilized to scale the inputs to $[-1,1]^{n_0}$, which is formulated as follows

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{\mu})=\frac{\boldsymbol{\mu}-(\boldsymbol{\mu}_{\max}+\boldsymbol{\mu}_{\min})/2}{(\boldsymbol{\mu}_{\max}-\boldsymbol{\mu}_{\min})/2}. \tag{28}$$

We note that the division in Eq. (28) denotes an element-wise operation, and it also applies in the following. As for the outputs, there is not an prior range of the output, however. We turn to a statistical method to do the normalization with an assumption that the outputs satisfy the Gaussian distribution. To this end, the standard derivation $\boldsymbol{\Theta}$ and mean $\overline{\boldsymbol{\alpha}}$ of outputs are calculated dimension by dimesion, and then the outputs are normalized as follows

$$\tilde{\boldsymbol{\alpha}}(\boldsymbol{\alpha})=\frac{\boldsymbol{\alpha}-\overline{\boldsymbol{\alpha}}}{\boldsymbol{\Theta}}. \tag{29}$$

We wrap the normalisation of inputs/ouputs into the ANN as follows

$$\tilde{\mathcal{O}}\left(\boldsymbol{\mu}\right)=\mathcal{O}\left(\tilde{\boldsymbol{\mu}}(\boldsymbol{\mu})\right)\otimes\boldsymbol{\Theta}+\overline{\boldsymbol{\alpha}}. \tag{30}$$

Thus, the inputs and outputs of the wrapped network $\tilde{\mathcal{O}}(\bullet)$ will be physical variables, while the inputs and outputs of the original network $\mathcal{O}(\bullet)$ will be normalized variables.

3.2.1. Projection-driven neural network(PDNN)

The data set $\mathcal{D}_{Pr}$ is used directly to train the network to avoid building the reduced-order model. The data set $\mathcal{D}_{Pr}$ is randomly split into two parts: train set $\mathcal{D}_{Pr}^{tr}$ and validation set $\mathcal{D}_{Pr}^{va}$. The famous Adam stochastic optimizer [37] is employed to train the network. The ratio between the size of $\mathcal{D}_{\text{POD}}^{tr}$ and $\mathcal{D}_{Pr}$ is set as 0.7. The loss function is defined as the mean square error between network outputs and targets scaled with the standard derivation Θ, namely

$$loss_{\text{PDNN}} = \frac{1}{N_{\mathcal{D}}} \sum_{\boldsymbol{\mu},\boldsymbol{\alpha}\in\mathcal{D}} \left\| \frac{\left(\widetilde{\mathcal{O}}\left(\boldsymbol{\mu};\mathbf{W},\mathbf{b}\right) - \boldsymbol{\alpha}\right)}{\boldsymbol{\Theta}} \right\|^2, \tag{31}$$

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

In real applications, we are not expected to generate too many snapshots, since the full-order model is always expensive. Thus the data set size $\mathcal{D}_{Pr}$ is always very small, so the training of the network will get trapped into overfitting problem. To prevent overfitting, first the L_2 regulation technique is employed by penalizing the loss of the train set with the weights $\mathbf{W}$

$$\widetilde{loss}_{\text{PDNN}} = loss_{\text{PDNN}} + \eta\|\mathbf{W}\|^2, \tag{32}$$

where η is the decay weight. Besides, we adopt the early stopping criterion: training will be immediately stopped only if the validation loss keeps increasing over K_{early} epoches. In what remains, $\eta = 10^{-4}$ and $K_{\text{early}} = 6$ are employed.

3.2.2. Physics-informed neural network(PINN)

Due to the limited number of available snapshots, it is always difficult to train a good network with Projection-driven neural network. Luckily, we have the reduced-order model, an another choice for training an ANN. Given the reduced-order model, ANN outputs need not explicitly approximate some given targets, but just satisfy the reduced-order model. In this way, no target is needed, and thus we have unlimited training data. By Latin hypercube sampling in parameter space, we generate the data set $\mathcal{D}_{Resi} = \{\boldsymbol{\mu}_i\}_{i=1}^{M_{Resi}}$, containing M_{Resi} residual points in parameter space. The loss function is defined as follows

$$loss_{\text{PINN}} = \frac{1}{N_{\mathcal{D}}} \sum_{\boldsymbol{\mu}\in\mathcal{D}} \left\| ROM\left(\widetilde{\mathcal{O}}\left(\boldsymbol{\mu};\mathbf{W},\mathbf{b}\right)\right)\right\|^2, \tag{33}$$

where $ROM(\boldsymbol{\alpha}) = \mathbf{V}^T\left(\mathbf{L}\mathbf{V}\boldsymbol{\alpha} + g\left(\mathbf{V}\boldsymbol{\alpha}\right) + \mathbf{C}\right)$ is the reduced-order model defined in Eq. (17). As we have unlimited training data, there is no problem of overfitting and thus no need of validation data set. The bottleneck of training the physics-informed neural network is how to handle the local minima trap. Here, we adopt the mini-batch training method, where the data set are shuffled and randomly into serval non-overlap subsets in each epoch. With each subset, the weights and biases will be updated by optimizer. The mini-batch training method can effectively avoid local minima trap and has enjoyed great success in lots of applications.

3.2.3. Physics-reinforced neural network(PRNN)

Physics-informed neural network(PINN) seems an perfect potential in predicting projection coefficients. But numerical experiments show that the accuracy of the physics-informed neural network sometimes is not that good, even if the the corresponding loss drops down to a very low level. The underlying reason comes from that the large scale difference of different RB modes. First, the RB mode with smaller index will be more dominant to the residual of the reduced-order model. Second, the training of network cannot work like a deterministic nonlinear solver, and it can only reduce the loss to a moderate level. Thus, the reduced-order model cannot be well-solved thorough

training network. As a result, the optimizer tends to neglect the relatively unimportant modes, namely the high-index modes.

To address this problem, we add the acknowledged data set $\mathcal{D}_{Pr}$ into the loss function of PINN. Although the projection coefficients in $\mathcal{D}_{Pr}$ are always not the solutions of the reduced-order model, the corresponding projections are closer to the snapshots and thus full-order model. Besides, the projection coefficients is calculated directly according to Eq. 27, and it has no problem of domination for different modes. Therefore, the projection coefficients can be used to provide more physical information into the network training. Thus we name this method as physics-reinforced neural network. The loss function is defined as the weighted sum of those of PDNN and PINN, namely

$$loss_{\mathrm{PRNN}} = loss_{\mathrm{PDNN}}|_{\mathcal{D}=\mathcal{D}_{Pr}} + w \times loss_{\mathrm{PINN}}|_{\mathcal{D}=\mathcal{D}_{Resi}}\,, \tag{34}$$

where w is a specific weight balancing projection and reduced-order model. In this work, we find that simply setting $w = 1$ will produces a good result. When training the network, the two data sets are treated separately. As for projection data set $\mathcal{D}_{Pr}$, it is more accurate but its size is very small, so we continue to use full batch. As for residual data set $\mathcal{D}_{Pr}$, it is less accurate but its size is very large, and we use mini-batch method for each updating of the network.

4. Numerical results and discussion

In this section, we will discuss the application of PDNN, PINN and PRNN methods to the following parameterized PDEs, namely the one-dimensional Burges' equation, two-dimensional lid driven flow and natural convection flow. The one-dimensional case are designed with an artificial solution with a zero-value boundary condition, intended for testing the prediction accuracy of the three networks and also study the influence of several factors. Their prediction accuracy are further discussed the two realistic flow problems, where geometry parameters and common boundary conditions (Dirichlet and Neumann) are included.

Besides, for comparison of accuracy, the results of two analytical methods are also considered:

(1) **Projection**: Projection of high-fidelity solution onto reduced basis;

(2) **POD-G**: The solution of the reduced-order model.

As the PDNN is trained with the projection coefficients of high-fidelity solutions, the PDNN targets the accuracy of the Projection. The PINN is trained to meet the reduced-order model, the PINN targets the accuracy of the POD-G. The PRNN can be viewed as a blending of the PDNN and PINN, and thus its accuracy is expected to fall somewhere between those of the Projection and POD-G.

To assess the online accuracy of these methods, the following metrics are defined in the sense of mean relative Euclidean distance.

(1) the average relative error of projections

$$\varepsilon_{\mathrm{Proj}} = \frac{1}{N_{\mathcal{D}_{te}}} \sum_{\boldsymbol{\mu},\boldsymbol{\phi}_h \in \mathcal{D}_{te}} \frac{\left\| \boldsymbol{\phi}_h - \tilde{\phi} - \mathbf{V}\mathbf{V}^T \left(\boldsymbol{\phi}_h - \tilde{\phi} \right) \right\|}{\left\| \boldsymbol{\phi}_h - \tilde{\phi} \right\|}; \tag{35}$$

(2) the average relative error of POD-G solutions

$$\varepsilon_{\bullet} = \frac{1}{N_{\mathcal{D}_{te}}} \sum_{\boldsymbol{\mu},\boldsymbol{\phi}_h \in \mathcal{D}_{te}} \frac{\left\| \boldsymbol{\phi}_h - \tilde{\phi} - \mathbf{V}\boldsymbol{\alpha}_{\bullet}\left(\boldsymbol{\mu}\right) \right\|}{\left\| \boldsymbol{\phi}_h - \tilde{\phi} \right\|}. \tag{36}$$

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 $\mathcal{D}_{te}$, the subscript "$\bullet$" is used to take the place of "POD-G","PDNN","PINN" and "PRNN", respectively. Thus, $\boldsymbol{\alpha}_\bullet(\boldsymbol{\mu})$ is the solution calculated from the corresponding method.

Note that the Projection, POD-G, PDNN, PINN and PRNN methods are all coded in Python. The networks built are all under the framework of Pytorch. The training of networks are all implemented on two GUPs (NVIDIA Quadro GP100) of a server class station. To solve the reduced-order model directly for the POD-G method, the nonlinear solver "linalg.solve" in Numpy library is employed. We refer the reader to the repository [1] for further details of the code. For different data set as discussed above, we employ the specific sampling methods in parameter space: for building the RB, we adopt the Sobol sequences, a quasi-random low-discrepancy sequences; For choosing residual points, we use the Latin hypercube sampling; For generating test data set, we employ uniform tensor-product grid.

4.1. Burgers' equation

In this subsection, to validate the proposed methods, we consider the following one-dimensional parameterized Burges' equation

$$\begin{cases} \phi(x;\boldsymbol{\mu}) \cdot \nabla\phi(x;\boldsymbol{\mu}) - \Delta\phi(x;\boldsymbol{\mu}) = s(x;\boldsymbol{\mu}) & -1 \le x \le 1 \\ \phi\left(x = \pm 1;\boldsymbol{\mu}\right) = 0 \end{cases}, \tag{37}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2) \in [1, 10] \times [1, 10]$ and $s(x;\boldsymbol{\mu})$ is the source term defined so that the exact solution satisfies

$$\phi(x;\boldsymbol{\mu}) = (1 + \mu_1 x)\sin(-\mu_2 x/3)(x^2 - 1). \tag{38}$$

The solution at the two end points are simply set zero to avoid the influence of boundary conditions. The problem is solved with Chebyshev pseudospectral (PS) method with $N_p + 1$=128+1 Chebyshev Gauss-Lobatto points.

We are interested in the prediction accuracy of the Projection, POD-G, PDNN, PINN and PRNN methods. The accuracy of Projection and POD-G is only determined by two factors, namely the sample size N_s for building reduced basis (RB) and and the number of chosen modes m, provided that the nonlinear solver for the POD-G is accurate enough. Here we term the two factors as essential factors. Besides the two factors, the neural networks, namely the PDNN, PINN and PRNN, are also determined by network architecture, the number of residual points, and so on. To study the influence of these factors, we resort to the control variates method. Some base values are set for these factors, namely sample size is $N_s = 80$, the number of chosen modes is picked from the set $\{2, 3, ..., 9\}$, network architecture is $L = 3$ and $n_H = 20$, the number of residual points is $N_{Resi} = 5000$. To test the prediction accuracy, a test data set of size $101 \times 101 = 10201$ is generated on a tensor-product parameter grid $\{1 + 0.09i\}_{i=0}^{i=100} \times \{1 + 0.09i\}_{i=0}^{i=100}$. The dense test grid will guarantee the reliability of the prediction accuracy. As for the training of networks, we use the same following hyperparameters: 20000 epoches, 10 batches with each batch of size $N_{Resi}/10$, leaning rate initialized as 0.01 and decaying by 96% every 200 epoches.

We first study the influence of the sample size. Fig. 2(a), presents the singular value distribution for the sample number $N_s = 80$. It is shown that the singular value decays quickly, implying that a small number of RB modes is enough for representing the dynamic of the problem. Meanwhile, the number of samples for building the necessary RB modes can also be largely reduced. Thus, for comparison, $N_s = 10$ and 320 random parameter samples are independently generated with Sobol sequences, and also their corresponding snapshots are calculated. The five methods are applied to the problem after the RB is built for $N_s = 10$, 80 and 320, respectively. The prediction accuracy

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

is shown in Fig. 3. It is shown that the Projection enjoys a better accuracy compared with the POD-G, but their discrepancy is very small. The pojection/POD-G improves just a little for sample number N_s increasing from 10 to 80, and almost stagnates from $N_s = 80$ to 320. Thus, sample number $N_s = 10$ is enough for capture the main dynamic of the problem. The networks perform quite differently. The accuracy of PDNN is far lower than that of PINN and PRNN. Even with the increase of sample number, the accuracy of PDNN can improve by one order of magnitude, but the error curves fluctuate intensively especially for larger m. On the contrary, both PINN and PRNN perform much better, they almost keep pace with POD-G for smaller m. As $m \geq 6$, the PINN performs a little better than the PRNN, but their error level both come to a saturation and cannot further drop down. The reason for the saturation is that both the PINN and PRNN can only reduce their loss function down to a level of 10^{-6} for this problem rather than zero or machine zero level. Thus the high-index modes of little importance are neglected by the networks. The PINN performs a little better than the PRNN, which is opponent with our inference in Section 3.2.3. The reason is that two-fold: first, the PINN performs near the POD-G; second, the POD-G is very close to the Projection. And the priority of the PRNN will be revealed when their discrepancy increases, as shown for the more complex problems in next two subsections.

To further study the influence of network architecture on the prediction accuracy of the three networks, we change network width to $n_H = 10$ and $n_H = 30$, respectively, while keeping network depth $L = 3$ fixed. The results are shown in Fig. 4. It is shown that the PDNN can not gain an obvious improvement of accuracy with the increase of n_H, or even get rid of the fluctuation, resulting from the overfitting trap. On the contrary, the PINN and PRNN show a much better tendency. Their prediction accuracy increases with n_H but tends to saturate for a larger n_H. This is in accordance with the situation that the approximation capability of the network increases and gradually saturate with the increase of network size.

As there are unlimited residual points available for training both the PINN and PRNN, we are interested in how the number of residual points influence the prediction accuracy. In addition to $N_{Resi} = 5000$, the numbers of residual points $N_{Resi} = 1250$, 2500 and 10000 are considered, and the results are shown in Fig. 5. The influence of the number of residual points is similar with that of network architecture. That is to say, the prediction accuracy increases with the number of residual points N_{Resi}, but it will reach a saturation after some critical N_{Resi}.

Besides, the further confirmation of the reliability of PRNN is shown in Fig. 6, offering a good agreement between the Chebyshev pseudospectral method and PRNN for some specific parameters.

4.2. Lid-driven flow

The lid-driven flow is considered for testing the prediction accuracy of the proposed methods in handling realistic problem. The flow is governed by the following non-dimensional incompressible Navier-Stokes equations

$$\begin{cases} \nabla_{\mathbf{x}} \cdot \mathbf{u} = 0 \\ (\mathbf{u} \cdot \nabla_{\mathbf{x}})\, \mathbf{u} = -\nabla_{\mathbf{x}} p + \dfrac{1}{Re} \nabla_{\mathbf{x}}^2 \mathbf{u} \end{cases}, \tag{39}$$

where $\mathbf{u} = (u, v)$ is the dimensionless velocity in Cartesian coordinate $\mathbf{x} = (x, y)$, p is the pressure, ∇ is the Hamiltonian operator, Re is the Reynolds number. The geometry of the problem is dipicted in Fig. 7(a), which is affected by the inclining angle θ of the left and right side walls. The Reynolds number and the inclining angle are the parameter $\boldsymbol{\mu}$ controlling the problem. In order to address the variable geometry, the physical domain is mapped to an square domain $[-1, 1]^2$. The mapping $\mathcal{X}$ from the physical domain $\mathbf{x} \in \Omega(\boldsymbol{\mu})$ to the computation domain $\boldsymbol{\xi} = (\xi_1, \xi_2) \in [-1, 1]^2$ and its inverse $\mathcal{X}^{-1}$ are defined as follows:

$$\mathcal{X} : \begin{cases} x = 1/2\xi_1 + 1/2\xi_2 \cos(\theta) \\ y = 1/2\xi_2 \sin(\theta) \end{cases} \quad \rightleftharpoons \quad \mathcal{X}^{-1} : \begin{cases} \xi_1 = 2(x - y\cot(\theta)) \\ \xi_2 = 2y/\sin(\theta) \end{cases}. \tag{40}$$

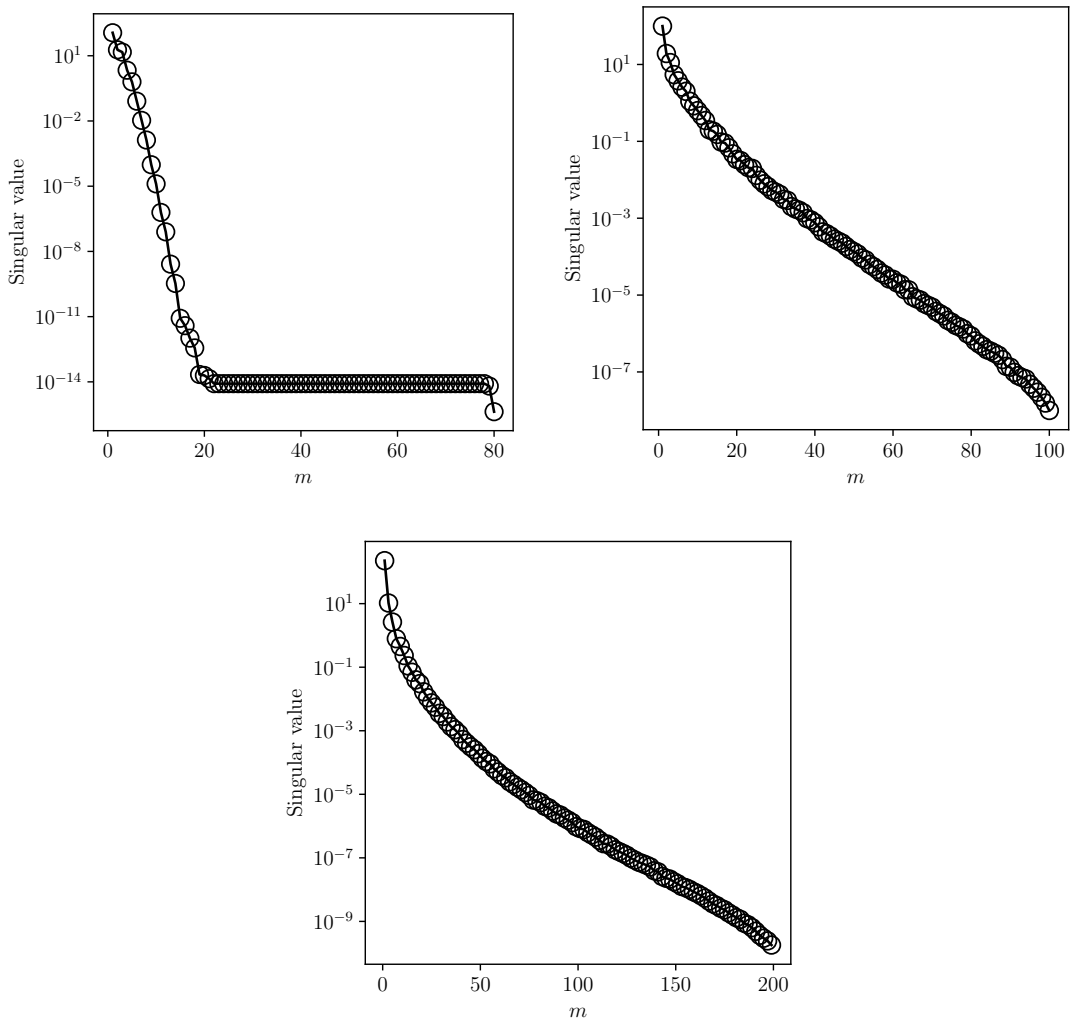

Figure 2: *Singular value distribution for (a) 1D Burges' equation with 80 snapshots, (b) 2D lid -driven flow with 100 snapshots and (c) 2D Natural convection with 200 snapshots.*

Thus, we have the Jacobin matrix of the mapping

$$\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 2 & 0 \\ -2\cot(\theta) & 2/\sin(\theta) \end{bmatrix}. \tag{41}$$

Substituting Eq. 41 into Eq. 39, we have the governing equation in computational space:

$$\begin{cases} \left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)\cdot\mathbf{u} = 0 \\ \left(\mathbf{u}\cdot\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)\right)\mathbf{u} = -\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)p + \dfrac{1}{Re}\left(\mathbf{J}^{-1}\nabla_{\boldsymbol{\xi}}\right)^2\mathbf{u} \end{cases}. \tag{42}$$

The flow is driven by the top moving wall. All the other three sides are all no-slip walls. Eq. (42) is discritized by the Chebyshev pseudospectral method with a tensor-product grid of 49×49. For this problem, it is crucial to address the well-known difficulty of the corner singularity, resulting

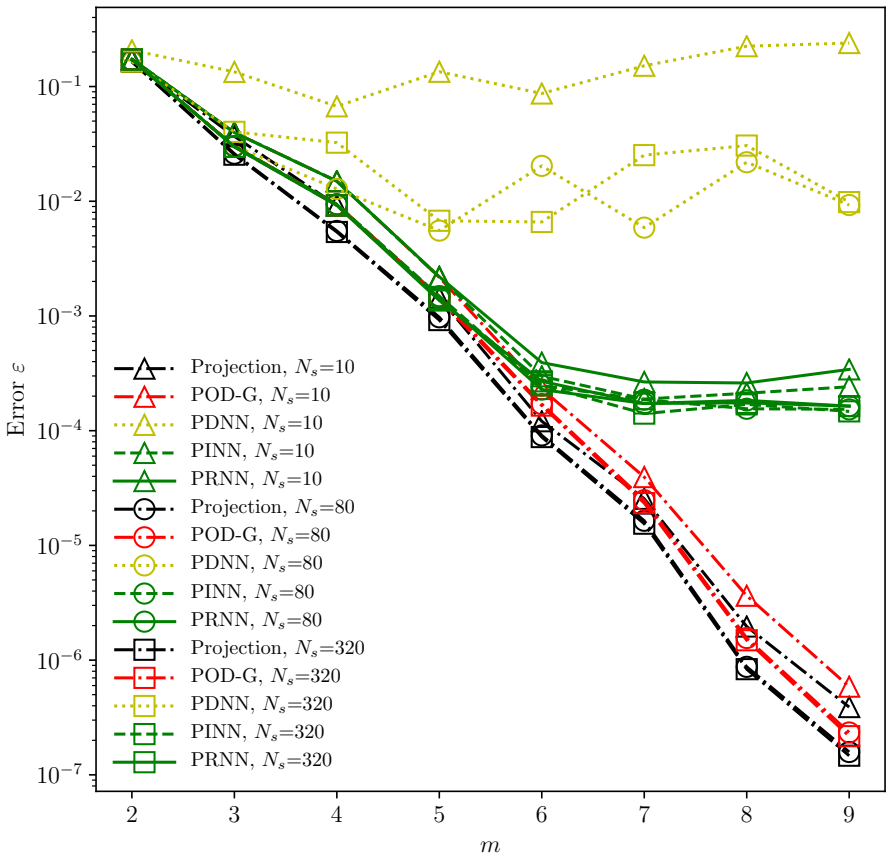

Figure 3: *The prediction accuracy of the Projection, POD-G, PDNN, PINN and PRNN methods for the number of snapshots N_s=10, 80 and 320, respectively.*

from the discontinuous horizontal velocity at the two top corners. In this work, to remove the singularity, the velocity of top wall is specified as

$$u_W = (1+\xi_1)^2(1-\xi_1)^2. \tag{43}$$

as adopted in references [38–40]. The $IP_N - IP_{N-2}$ method [28, 29] is employed to remove the spurious modes of pressure. Besides, the artificial compressibility method [41] is employed to address the coupling of velocity field and pressure by transforming steady Eq. (42) into an unsteady one in pseudo time. The derived discrete equation is then solved using an explicit fourth-order four-stage Runge-Kutta integrator in pseudo time.

The parameter space of interest is $\boldsymbol{\mu} = (Re, \theta) \in [100, 500] \times [\pi/3, 2\pi/3]$. 100 snapshots of high-fidelity solutions for parameters generated with Sobol sequence are calculated and collected for building the reduced basis, and the corresponding singular value distribution is plotted in Fig. 2(b). It is shown that the singular value decays slowly, implying that more modes are required to represent the underlying dynamic. Thus we choose the first $m = 5, 10, ..., 30$ modes, respectively. The prediction accuracy is assessed on a test data set of size $11 \times 11 = 121$, generated on a tensor-product parameter grid $\boldsymbol{\mu} = \{100 + 40i\}_{i=0}^{10} \times \{\pi/3 + j\pi/15\}_{j=0}^{10}$. For the POD-G method solving the reduced-order model, the solving always blows up if it starts from an improper initialization. In this work, for testing the prediction accuracy of the POD-G method, the solving of the reduced-order model for a parameter $\boldsymbol{\mu}$ starts from the projection coefficients of the full-order solution $\boldsymbol{\phi}_h(\boldsymbol{\mu})$ on

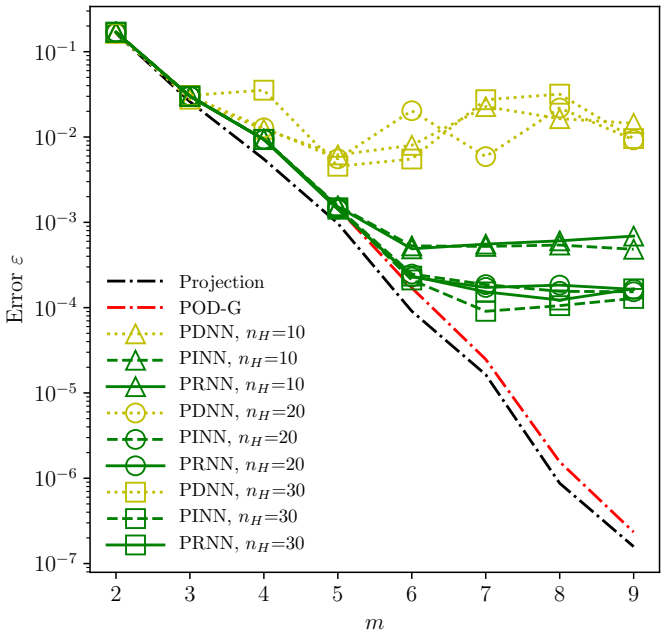

Figure 4: *The prediction accuracy of the Projection, POD-G, PDNN, PINN and PRNN methods. The network width is set $n_H = 10$, 20 and 30, respectively.*

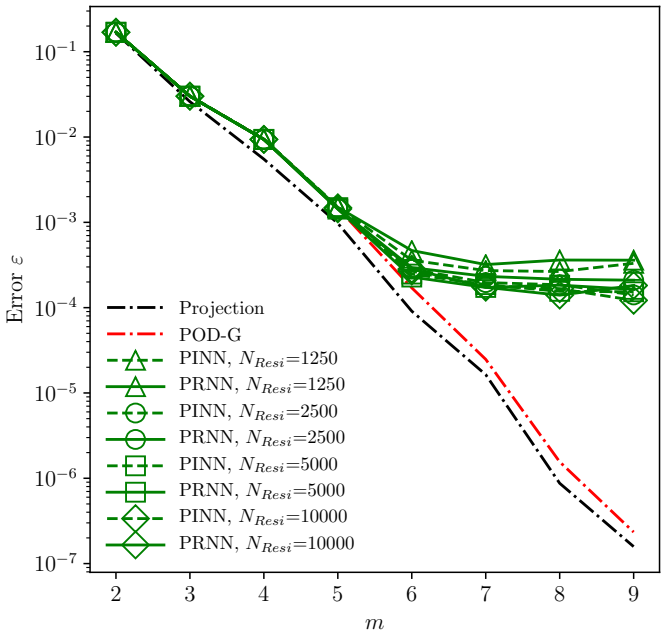

Figure 5: *The prediction accuracy of the Projection, POD-G, PINN and PRNN methods. N_{Resi}=1250, 2500, 5000, 10000 residual points are chosen for training the PINN and PRNN.*

the reduced basis. Besides, for the sake of comparison, we pick the first 30, 60 snapshots from

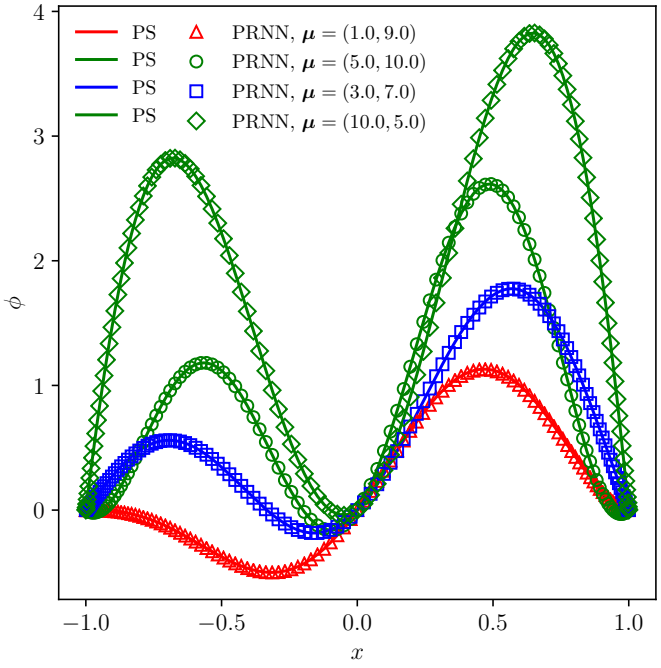

Figure 6: *The comparison between the Chebyshev pseudospectral (PS) and the PRNN solutions for four specific parameters. The results of the PRNN are obtained via employing the m=9 modes extracted from 10 snapshot and a network of L=3 and n_H=20 architecture trained with 5000 residual points.*

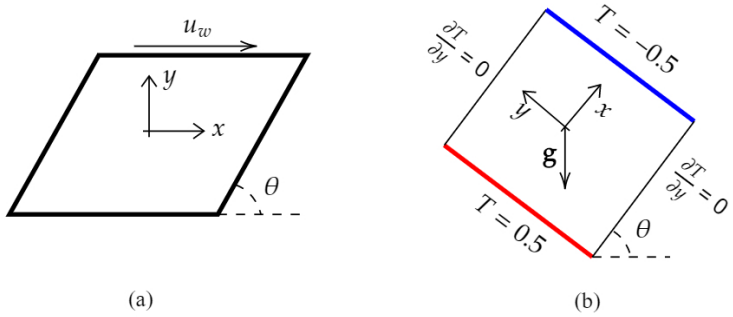

Figure 7: *The geometry and boundary conditions for (a) Lid-Driven flow and (b) Natural convection in enclosure.*

the 100 snapshots to do the same test. As for the training of networks, we use the same following hyperparameters: network architecture L=5 and n_H=30, 40000 epoches, batch size 1000, leaning rate initialized as 0.01 and decaying by 96% every 200 epoches.

The prediction accuracy is shown in Fig. 8. It is shown that the POD-G is far away from the Projection, compared with Fig. 3 for the Burges' equation. Similar convergence tendency features the PINN and PRNN, i.e. the accuracy increases with smaller m but saturates after a critical value of m. However, the priority of the PRNN over PINN is demonstrated. The PINN cannot work well, and the PINN will saturate for $m \geq 15$, even if more snapshots are employed for building the reduced basis. The PRNN performs much better, showing a satisfactory properties. On one way, the

PRNN can achieve an accuracy between the POD-G an Projection for smaller m, conforming the blending construction of the loss function in Eq. (34) of the PRNN; on the other way, the PRNN performs better than PINN, and shows an expected increase of accuracy with the snapshot number N_S. The PDNN enjoys an agreement with the Projection for a smaller m, and also an improvement with the increase of m. However, it quickly deteriorates after a critical value of m. What's more, the accuracy of the PDNN using 100 snapshots is lower than only both the PINN and PRNN using 30 snapshots.

All in all, it can be concluded that the PRNN is much better tan the PINN and PDNN. The further confirmation of the reliability of PRNN is shown in Fig. 9, offering a good agreement between the Chebyshev pseudospectral method and PRNN for some specific parameters. We note that the streamlines displayed in Fig. 9 are derived by solving the following Poisson equation for streamfunction with an acknowledged velocity field:

$$\begin{cases} \nabla^2_{\mathbf{x}}\psi = \dfrac{\partial u}{\partial y} - \dfrac{\partial v}{\partial x} & x \in \Omega \\ \psi = 0 & x \in \partial\Omega \end{cases}. \tag{44}$$

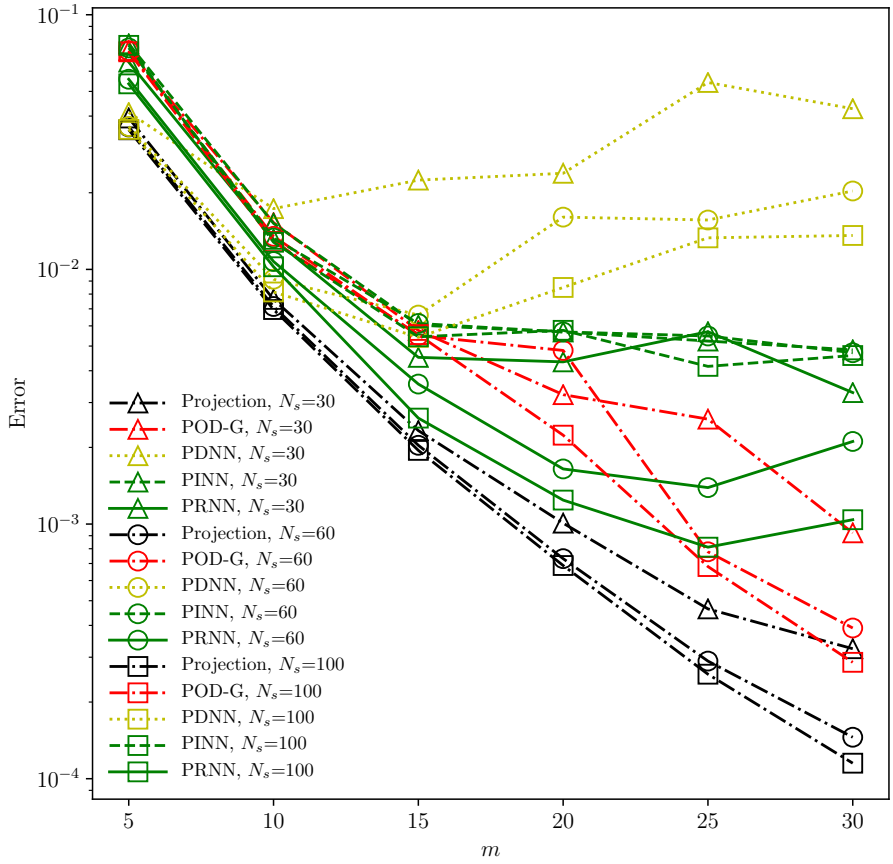

Figure 8: *The prediction accuracy of the Projection, POD-G, PDNN, PINN and PRNN methods for 2D lid-driven flow with the number of snapshots N_s=30, 60 and 100, respectively.*

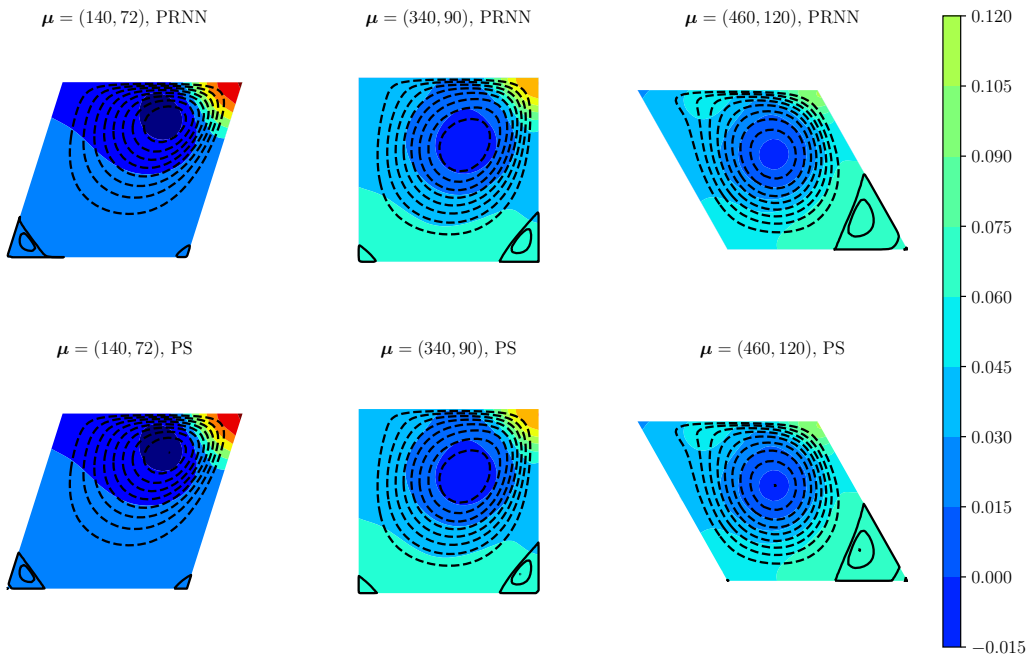

Figure 9: *The comparison of pressure contour and streamlines of 2D lid-driven flow between the Chebyshev pseudospectral (PS) and the PRNN solutions for four specific parameters. The results of the PRNN are obtained via employing the m=25 modes extracted from 100 snapshots and a network of L=5 and n_H=30 architecture trained with 20000 residual points. The solid streamline indicates an anticlockwise vortex, the dashed streamline an clockwise vortex.*

4.3. Natural convection in enclosure

To further test the prediction accuracy of the networks for more complex problems, natural convection in enclosure is considered. The geometry and boundary conditions are shown in Fig. 7(b). The unit square cavity is filled with incompressible Newtonian fluid. The cavity is heated and cooled differentially on the two opponent sides, with the other two sides insulated. The gravity acts in the vertical direction. The flow is driven by the vertical buoyancy force, which is modeled according to Boussinesq approximation. The cavity is deployed with a rotation angle θ. To address the variable geometry, a variable coordinate system is applied with x-axis perpendicular to the heated/cooled sides. Thus, the flow is governed by the following two-dimensional incompressible Navier-Stokes equation and energy equation:

$$\begin{cases} \nabla_{\mathbf{x}} \cdot \mathbf{u} = 0 \\ (\mathbf{u} \cdot \nabla_{\mathbf{x}})\, \mathbf{u} = -\nabla_{\mathbf{x}} p + \sqrt{\left(\dfrac{Pr}{Ra}\right)} \nabla^2_{\mathbf{x}} \mathbf{u} + T \mathbf{n}_g \\ (\mathbf{u} \cdot \nabla_{\mathbf{x}})\, T = \dfrac{1}{\sqrt{(Pr \times Ra)}} \nabla^2_{\mathbf{x}} T, \end{cases} \tag{45}$$

where $\mathbf{u} = (u, v)$ is the dimensionless velocity in Cartesian coordinate $\mathbf{x} = (x, y)$, p is the pressure, ∇ is the Hamiltonian operator, Ra is the Rayleigh number, Pr is the Prandtl number, $\mathbf{n}_g = (\sin(\theta), \cos(\theta))$ is the unit vertical vector. All the sides are no-slip walls. The heated and cooled sides are enforced with the dimensionless temperature $T = 0.5$ and $T = -0.5$, respectively. The other two sides are

enforced with adiabatic boundary condition $\partial T/\partial y = 0$. Similarly, the flow is solved with the Chebyshev pseudospectral method accompanied with the $IP_N - IP_{N-2}$ scheme on a grid of 49×49.

The parameter space of interest is $\boldsymbol{\mu} = (Ra, Pr, \theta) \in [10^4, 10^5] \times [0.6, 0.8] \times [\pi/4, \pi/2]$. 200 snapshots of high-fidelity solutions for parameters generated with Sobol sequence are calculated and collected for building the reduced basis, and the corresponding singular value distribution is plotted in Fig. 2(c). It is shown that the singular value decays slowly, implying that more modes are required to represent the underlying dynamic. We choose the first $m = 5, 10, 15, ..., 30$ modes, respectively. To test the prediction accuracy, a test data set of size $6 \times 6 \times 6 = 216$ is generated on a tensor-product parameter grid $\boldsymbol{\mu} = \{(1 + 1.8i) \times 10^4\}_{i=0}^{5} \times \{0.6 + 0.04j\}_{j=0}^{5} \times \{\pi/4 + k\pi/20\}_{k=0}^{5}$. Similarly, for testing the prediction accuracy of the POD-G method, the solving of the reduced-order model for a parameter $\boldsymbol{\mu}$ starts from the projection coefficients of the full-order solution $\boldsymbol{\phi}_h(\boldsymbol{\mu})$ on the reduced basis. For comparison, we pick the first 30, 100 snapshots from the 200 snapshots to do the same test. As for the training of networks, we use the same following hyperparameters: network architecture L=5 and n_H=30, 40000 epoches, batch size 1000, the leaning rate initialized as 0.01 and decaying by 96% every 200 epoches.

The prediction accuracy is shown in Fig. 10. The tendency of PDNN, PINN and PRNN is similar with that in Fig. 8. Both the PINN and PRNN are much better than the PDNN. Both the PINN and PRNN can achieve a higher accuracy with 30 snapshots than the PDNN even with 200 snapshots. The PRNN is the best choice, as it not only enjoys a high accuracy, but also shows an improvement with the increase of the snapshots. Besides, the test error of the PRNN is about one order of magnitude smaller than the PDNN for a larger m. The further confirmation of the reliability of PRNN is shown in Fig. 11, offering a good agreement between the Chebyshev pseudospectral method and the PRNN for some specific parameters. Also, the displayed streamlines are derived from solving Eq. 44.

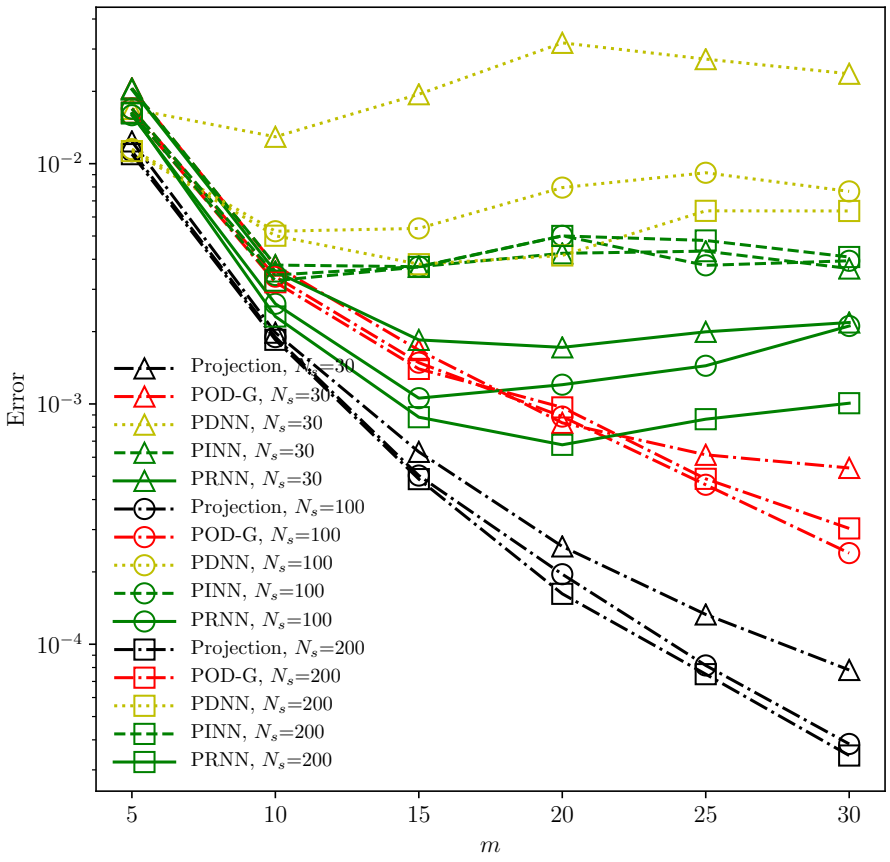

Figure 10: *The prediction accuracy of the Projection, POD-G, PDNN, PINN and PRNN methods for 2D natural convection with the number of snapshots N_s=30, 100 and 200, respectively.*

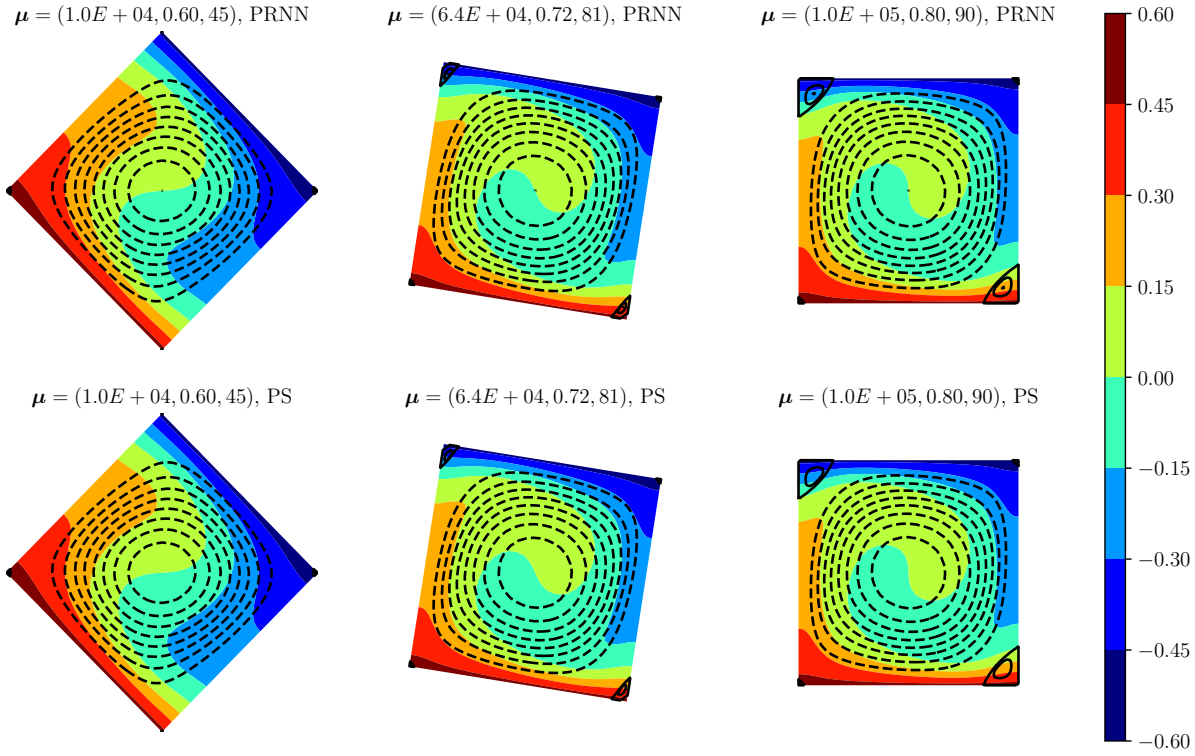

Figure 11: *The comparison of temperature contour and streamlines of 2D natural convection between the Chebyshev pseudospectral (PS) and the PRNN solutions for four specific parameters. The results of the PRNN are obtained via employing the m=25 modes extracted from 100 snapshots and a network of L=5 and n_H=30 architecture trained with 20000 residual points. The solid streamline indicates an anticlockwise vortex, the dashed streamline an clockwise vortex.*

5. Conclusions

This paper proposed a physics-informed machine learning framework for reduced-order modeling of parameterized PDEs. A reduced space, spanned by the reduced basis extracted by high-fidelity solutions (snapshots), is built for representing the main dynamics of the problem. The full-order model is projected onto the reduced space according to Galerkin method, leading to the reduce-order model. Combining reduced-order information and marching learning, a physics-informed neural network (PINN) and a physics-reinforced neural network are proposed and trained for approximating the map from parameters to projection coefficients. The PINN are intended for resolving the reduced-order model directly, while the PRNN are intended for finding somewhere between the reduced-order model and high-fidelity solutions. The proposed PINN and PRNN are compared with the Projection method, the POD-G method and a non-intrusive reduced basis method (referred to PDNN), by testing on three problems: one-dimensional Burges' equation, two-dimensional lid-driven flow and natural convection in enclosure.

The proposed PINN and PRNN both can achieve a higher accuracy with fewer snapshots than the PDNN. For complex nonlinear problems, the PINN sometimes will be trapped with local minima problem, thus limiting its accuracy. On the contrary, the PRNN has no such problems, and its accuracy increases with the number of available snapshots. In particular, once the snapshots are enough for building reduced basis, the PRNN method will promise a satisfactory accuracy.

Acknowledgments

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] D. J. Lucia, P. S. Beran, W. A. Silva, Reduced-order modeling: new approaches for computational physics, Progress in aerospace sciences 40 (1-2) (2004) 51–117.

[4] Y. Maday, Reduced basis method for the rapid and reliable solution of partial differential equations.

[5] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications-part i: Theory, Journal of Sound and vibration 252 (3) (2002) 527–544.

[6] F. Chinesta, P. Ladeveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Archives of Computational Methods in Engineering 18 (4) (2011) 395.

[7] 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.

[8] H. Panzer, J. Mohring, R. Eid, B. Lohmann, Parametric model order reduction by matrix interpolation, at-Automatisierungstechnik 58 (8) (2010) 475–484.

[9] M. Billaud-Friess, A. Nouy, Dynamical model reduction method for solving parameter-dependent dynamical systems, SIAM Journal on Scientific Computing 39 (4) (2017) A1766–A1792.

[10] 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.

[11] J. S. Hesthaven, B. Stamm, S. Zhang, Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methodsâĹŮ, ESAIM: Mathematical Modelling and Numerical Analysis 48 (1) (2014) 259–283.

[12] C. W. Rowley, T. Colonius, R. M. Murray, Model reduction for compressible flows using pod and galerkin projection, Physica D: Nonlinear Phenomena 189 (1-2) (2004) 115–129.

[13] Q. Wang, N. Ripamonti, J. S. Hesthaven, Recurrent neural network closure of parametric pod-galerkin reduced-order models based on the mori-zwanzig formalism, Journal of Computational Physics (2020) 109402.

[14] A. Deane, I. Kevrekidis, G. E. Karniadakis, S. Orszag, Low-dimensional models for complex geometry flows: application to grooved channels and circular cylinders, Physics of Fluids A: Fluid Dynamics 3 (10) (1991) 2337–2354.

[15] C. Huang, K. Duraisamy, C. Merkle, Challenges in reduced order modeling of reacting flows, in: 2018 Joint Propulsion Conference, 2018, p. 4675.

[16] A. Iollo, S. Lanteri, J.-A. Désidéri, Stability properties of pod–galerkin approximations for the compressible navier–stokes equations, Theoretical and Computational Fluid Dynamics 13 (6) (2000) 377–396.

[17] W. Chen, J. S. Hesthaven, B. Junqiang, Y. Qiu, Z. Yang, Y. Tihao, Greedy nonintrusive reduced order model for fluid dynamics, AIAA Journal 56 (12) (2018) 4927–4943.

[18] S. Walton, O. Hassan, K. Morgan, Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions, Applied Mathematical Modelling 37 (20-21) (2013) 8930–8945.

[19] D. Xiao, F. Fang, C. Pain, G. Hu, Non-intrusive reduced-order modelling of the navier–stokes equations based on rbf interpolation, International Journal for Numerical Methods in Fluids 79 (11) (2015) 580–595.

[20] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55–78.

[21] M. Guo, J. S. Hesthaven, Data-driven reduced order modeling for time-dependent problems, Computer methods in applied mechanics and engineering 345 (2019) 75–99.

[22] 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, Journal of Computational Physics 378 (2019) 686–707.

[23] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.

[24] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows, Computer Methods in Applied Mechanics and Engineering 360 (2020) 112789.

[25] W. Chen, Y. Ju, C. Zhang, A multidomain multigrid pseudospectral method for incompressible flows, Numerical Heat Transfer, Part B: Fundamentals 74 (1) (2018) 415–431.

[26] W. Chen, Y. Ju, C. Zhang, A parallel inverted dual time stepping method for unsteady incompressible fluid flow and heat transfer problems, Computer Physics Communications (2020) 107325.

[27] W. Chen, Y. Ju, C. Zhang, A collocated-grid spectral difference method for compressible flows, Computers & Fluids 196 (2020) 104341.

[28] W. Zhang, C. Zhang, G. Xi, An explicit chebyshev pseudospectral multigrid method for incompressible navier–stokes equations, Computers & Fluids 39 (1) (2010) 178–188.

[29] R. Peyret, Spectral methods for incompressible viscous flow, Vol. 148, Springer Science & Business Media, 2013.

[30] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.

[31] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An âĂŸempirical interpolationâĂŹmethod: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathematique 339 (9) (2004) 667–672.

[32] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764.

[33] R. Everson, L. Sirovich, Karhunen–loeve procedure for gappy data, JOSA A 12 (8) (1995) 1657–1664.

[34] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Transactions on Automatic Control 53 (10) (2008) 2237–2251.

[35] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941.

[36] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[37] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[38] J. Shen, Hopf bifurcation of the unsteady regularized driven cavity flow, Journal of Computational Physics 95 (1) (1991) 228–245.

[39] A. Pinelli, A. Vacca, Chebyshev collocation method and multidomain decomposition for the incompressible navier-stokes equations, International journal for numerical methods in fluids 18 (8) (1994) 781–799.

[40] T. N. Phillips, G. W. Roberts, The treatment of spurious pressure modes in spectral incompressible flow calculations, Journal of Computational Physics 105 (1) (1993) 150–164.

[41] J. R. Clausen, Entropically damped form of artificial compressibility for explicit simulation of incompressible flow, Physical Review E 87 (1) (2013) 013309.