Physics-informed machine learning of reduced-order model

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 basis functions that represent the main dynamics of the full-order model. A physics informed neural network (PINN), which approximates the mapping from the parameters to the reduced coefficients, is trained by minimizing the loss function as a sum of the residual of the reduced-order model and the matching error with the projection of high-fidelity snapshots onto the reduced space. In the online stage, given a new parameter location, the reduced coefficients are predicted by the physics-informed neural network. Numerical results demonstrate that the PINN can provide fast and accurate solutions to nonlinear problems.

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

1. Introduction

Why we need reduced-order modeling.

Intrusive reduced-order model. POD-G is a classical choice. Pros and cons.

Intrusive reduced-order model. POD-NN is a recently developed one. Pros and cons. Especially the need of large amount of data for training.

We propose to use physics-informed machine learning to improve the reduced-order modeling by taking advantage of both physics and data. Then explain the idea.

The advantage of PINN over POD-G and POD-NN. Numerical results demonstrate the advantages.

The remainder of this paper is organized as follows.

2. Projection-based reduced basis method

Consider a nonlinear dynamic system governed by 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

*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)

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 domains will be mapped into a fixed computational domain with the same collation of degree of freedoms. This can be implemented by a invertible problem-dependent mapping $\mathcal{X} : \mathbf{x} \in \Omega(\boldsymbol{\mu}) \rightarrow \boldsymbol{\xi} \in \widetilde{\Omega}$, which reads

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

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

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

where $\widetilde{\Omega}$ is the computational domain, $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{N}}$ are derived operators in computational space resulting from 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 pseudospectral method, spectral difference method, 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 $\boldsymbol{\mu}$. $g_{\widetilde{\mathcal{N}}} : \mathbf{R}^N \times \mathbf{R}^{N_B} \mapsto \mathbf{R}^N$ is a nonlinear function derived from nonlinear part of operator $\widetilde{\mathcal{N}}$.

For the treatment of boundary conditions, it can classified into two categories, namely weakly enforced and strongly enforced boundary conditions. As for weakly enforced boundary conditions, no DOFs are 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 is adopted to prevent spurious pressure mode from containing the flow fields. In the $IP_N - IP_{N-2}$ method, pressure derivative is approximated without 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.

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 degree 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 Eq. (**??**) can be well approximated with 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 $\mathcal{V}$

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

where $\mathbf{V} \in \mathbf{R}^{N\times m}$ is a matrix with the basis vectors as its columns, $\boldsymbol{\alpha}$ is the coefficients of the basis, $\boldsymbol{\epsilon}$ is the projection error, $\mathbf{I} \in \mathbf{R}^N$ is filled with 1. $\tilde{\phi} \in \mathbf{R}$ is independent of $\boldsymbol{\mu}$, set for filtering a constant value to avoid its domination on the basis, and usually it can be set as the average. The boundary conditions 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

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

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 $g : \mathbf{R}^N \mapsto \mathbf{R}^N$ is the 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 a 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 restrict ourself to 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 basis. In the literature, there exists many methods to achieve it. Such popular methods includes, but not limited to, the Proper Orthogonal Decomposition (POD), the Proper Generalized Decomposition, the Piecewise Tangential Interpolation, the Matrix Interpolation, the Loewner framework and several greedy-based approaches for the identification of sub-optimal subspaces. Among these methods, POD, perhaps the most widely used technique, 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 $\mathbf{\Phi}_{\mathcal{P}_M}$ can represent the underlying physics dynamics of the problem. The idea behind POD technique is to find a suitable orthogonal basis matrix to minimise the problem defined as follows:

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

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

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

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

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

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 $\mathbf{\Phi}_{\mathcal{P}_M}$ at an arbitrary accuracy. Luckily, most problems exhibit an exponentially decaying series of singular values, and thus we can chose a very small value of m to approximate the problems with a good accuracy.

2.2.2. Further reduction in online cost

Although we have the reduced-order model defined in Eq. (17), the computational cost is still very expensive, scaling with the original size of teh full-order model. For online stage, a further reduction of computational cost 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 online 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), its discrete variant (DEIM), Gappy-POD and Missing Point Estimation (MPE). In the present work, we restrict ourselves to quadratic nonlinearity, which can be exactly transformed to quadratic form, and thus hyper-reduction is avoid. 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 element-wise multiplication operator. Substituting Eq. (22) in to Eq. (17), the nonlinear part can be transformed as

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

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

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

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 parameter space to high-fidelity solution. Based on the reduced basis, we jsut need to find the mapping from parameter space to reduced basis coefficient, namely the projection of high-fidelity solution 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 ??, the underlying idea is an interpolation, such as cubic spline method based on a tensor-product grid in parameter space and building an artificial neural network (ANN) with a labeled data. These non-intrusive methods can achieve a good approximation and very little online cost, but the drawback is their highly relying on the size of labeled data, namely the number of evaluations of full-order model. According to Jan, artificial neural network 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 is prohibitively larger than the size of snapshots for building a satisfactory reduced basis. A good alternate is the intrusive method, where the reduced-order model is solved 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 m. To this end, we combine ANN and the reduced-order model together together.

In the next subsections, we first briefly introduce the basis idea of artificial neural networks. Then we will start from the projection-driven neural networks proposed by Jan ??. After that two methods, namely the physics-inform neural network and physics-reinforced neural network are proposed, pursue a higher prediction accuracy with fewer snapshots.

3.1. Artificial neural network (ANN)

For an arbitrary target function, ANN is a powerful nonlinear approximators 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 is a nonlinear function $\mathcal{O} : \mathbf{R}^{n_0} \mapsto \mathbf{R}^{n_{L+1}}$ of inputs $\mathbf{x}$, comprising a recursion

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

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 Swish(x) = xsigmoid(x) and use the same number of neurons in each hidden layers $n_1 = n_2 =, ..., = n_L = n_H$, 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 metricate by a scalar loss function. Thus the training of ANN is essentially an single-objective minimization problem

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

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 $\mathbf{W} = \{\mathbf{W_i}\}_{\mathbf{i=1}}^{\mathbf{L+1}}$ and $\mathbf{b} = \{\mathbf{b_i}\}_{\mathbf{i=1}}^{\mathbf{L+1}}$. 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 relive local minima trap and regulation method to relieve overfitting trap.

3.2. Training the networks

Some preparation need to be made before training the 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 $\mathbf{V}^T$

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

All the parameters in $\mathcal{P}_M$ along with the projections of their corresponding snapshots, denoted by $\mathcal{D}_{Pr} = \left\{\left(\boldsymbol{\mu}_i, \boldsymbol{\alpha}|_{\boldsymbol{\mu}=\boldsymbol{\mu}_i}\right)\right\}_{i=1}^{M}$, will be a good collection of input-output data set for training an ANN. 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, the data set $\mathcal{D}_{Pr}$ need 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 do the normalization with an assumption that the outputs satisfy the Gaussian distribution. To this end, the standard derivation $\mathbf{\Theta}$ and mean $\overline{\boldsymbol{\alpha}}$ are calculated, and then the outputs are normalized as follows

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

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

$$\tilde{\mathcal{O}}\left(\boldsymbol{\mu}; \mathbf{W}, \mathbf{b}\right) = \mathcal{O}\left(\tilde{\boldsymbol{\mu}}(\boldsymbol{\mu})\right) \otimes \mathbf{\Theta} + \overline{\boldsymbol{\alpha}} \tag{30}$$

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)

According to Jan ??, 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 $\mathcal{D}_{Pr}^{te}$. The famous Adam optimizer is employed to train the network. The ratio

between the size of $\mathcal{D}_{\mathrm{POD}}^{tr}$ 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

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

where $\mathcal{D}$ is the chosen data set of size $N_{\mathcal{D}}$, which can be the train set or the test set.

In real applications, we are not expected to generate too many snapshots, since the full-order model is always prohibitively expensive. Thus the data set size $\mathcal{D}_{Pr}$ is always to 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 $\mathbf{W}$

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

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 $K_{\mathrm{early}} = 6$ 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 don't explicitly approximate some given targets, but just do 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 $\mathcal{D}_{Resi} = \{\boldsymbol{\mu}_i\}_{i=1}^{M_{Resi}}$, constraining M_{Resi} residual points in parameter space. The loss function is defined as follows

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

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 in to 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 a great number of applications.

3.2.3. physics-reinforced neural network(PRNN)

Physics-informed neural network(PINN) seems an perfect potential in prediction reduced basis coefficients. But the 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 projection RB coefficients into the loss function of PINN. Although the projection RB coefficients are always not the solutions of the reduced-order model,

the projections are more close to the snapshots and thus full-order model. Besides, the projection RB coefficients can be calculated directly according to Eq. 27, and it has no problem of domination for different modes. Therefore, the projection RB 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

In this section, the following notations will be used repeatedly in the text and plots to distinguish results of different methods: we will discuss the application of PDNN, PINN and PRNN methods to three parameterized PDEs, namely the one-dimensional Burges' equation, two-dimensional lid driven flow and two-dimensional natural convection flow. 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.

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.

To assess the online accuracy of these methods, the following metrics are defined in the sense of mean squared error.

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

Note that Projection, POD-G, PDNN, PINN and PRNN are all coded in Python. The networks built are 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 POD-G, the solver "linalg.solve" in Numpy library is employed. We refer the reader to the repository [1] for further details of the code method. For more information of the code, we refe 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 uniformly tensor-product grid.

To train the netwroks

4.1. Burgers' equation

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

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

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

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

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 are interested in the prediction accuracy of 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 for building RB and and the number of chosen modes, given that the nonlinear solver for POD-G is accurate enough. Here we term the two factors as essential factors. Besides the two factors, neural network like 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 factor, 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 $N_{Resi} = 5000$. 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 grantee the reliability of the prediction accuracy.

We first study the influence of the sample size. Fig. ??, 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 sample number can also be largely reduced. Thus, for comparison, $N_s = 10$ and 320 random parameter samples are independently generated with Sobol sequences. The five methods are applied to the problem after the RB are built for $N_s = 10$, 80 and 320, respectively. The prediction accuracy is shown in Fig. ??. 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

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

PINN performs a little better than the PRNN, but their error level both come to a flat and cannot further drop down. The reason for the flat level is that both PINN and PRNN can only reduce their loss function down to a level of 10^{-6} for this problem rather zero or machine zero level. Thus the high-index modes are neglected by the networks. The PINN performs a little better than the PRNN, it is opponent with our inference in Section 3.2.3. The reason is two-fold: first, the problem and the target function is relatively easy for networks; second, the accuracy of POD-G is very close to projection. Therefore, There is no priority for the PRNN, as its loss function is a hybrid.

To further study the influence of network architecture on the prediction accuracy of networks, we change the network width to $n_H = 10$ and $n_H = 30$,respectively, while keeping the network depth $L = 3$ fixed. The results are shown in Fig. ??. It is shown that the PDNN can not gain an 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 network size.

As there is unlimited residual points for training PINN and PRNN, we are interested in how the number of residual points influence the prediction accuracy. Apart from $N_{Resi} = 5000$, the numbers of residual points $N_{Resi} = 1250$, 2500 and 10000 are considered, and their results are shown in Fig. ??. The influence of the numbers of residual points is similar with that of network architecture.

4.2. Lid-driven cavity

The lid-driven cavity flow is considered for testing the prediction accuracy of the 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 illustrated in Fig. ??, 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, as shown in Fig. ??. 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:

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

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. The flow is solved by the Chebyshev pseudospectral method. To solve the flow, the Chebyshev pseudospectral

method is employed with a grid of 49×49. For this problem, it is crucial to address the well-known difficulty the corner singularity, resulting from the discontinuous horizontal velocity at the two top corners. 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 ??. Artificial compressibility method is employed to address the coupling 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 Sobol sequence are calculated and collected for building the reduced basis, and the corresponding singular value distribution is plotted in Fig. ??. It is shown that the singular value decays slowly, implying that more modes are required to represent the underlying dynamic. We set $m = 5, 10, 15, ..., 30$ to test the prediction accuracy of PDNN, PINN and PRNN. To test the prediction accuracy, a test data set of size $11 \times 11 = 121$ is generated on a tensor-product parameter grid $\boldsymbol{\mu} = \{100 + 40i\}_{i=0}^{i=10} \times \{\pi/3 + i\pi/15\}_{i=0}^{i=10}$. For comparison, we pick the first 30, 60 snapshots from the 100 snapshots to do the same test. The prediction accuracy is shown in Fig. ??. It is shown that both PINN and PRNN are much better than

4.3. Natural convection in enclosure

To further test the prediction accuracy of the networks for more complex problems, natural convection in enclosure problem is considered. The geometry and boundary is shown in Fig. ??. The square cavity of unit size 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:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ (\mathbf{u} \cdot \nabla)\, \mathbf{u} &= -\nabla p + \sqrt{\left(\frac{Pr}{Ra}\right)} \nabla^2 \mathbf{u} + T \mathbf{n}_g \\ (\mathbf{u} \cdot \nabla)\, T &= \frac{1}{\sqrt{(Pr \times Ra)}} \nabla^2 T, \end{aligned} \tag{44}$$

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. Similarly, the flow solved with the Chebyshev pseudospectral method iwth a grid of 49×49.

5. Conclusions

Acknowledgments

The first author is financially supported by Xi'an Jiaotong University xxx.