Journal of Scientific Computing (2022) 92:64
https://doi.org/10.1007/s10915-022-01899-4

Model Order Reduction Method Based on Machine Learning for Parameterized Time-Dependent Partial Differential Equations

Fangxiong Cheng[1] · **Hui Xu**[2] · **Xinlong Feng**[1]

Received: 13 October 2021 / Revised: 12 April 2022 / Accepted: 5 June 2022 /
Published online: 8 July 2022

Abstract
In this paper, we propose a data-driven model order reduction method to solve parameterized time-dependent partial differential equations. We describe the system with the state variable equations, and represent a class of candidate models with the artificial neural network. The discrete L_2 error between the output of artificial neural network and the high-fidelity solution is minimized with the state variable equations and initial conditions as constraints. Therefore, the model order reduction problem can be described as a kind of optimization problem with constraints, which can be solved by combining Levenberg–Marquardt algorithm and linear search algorithm, followed by sensitivity analysis of the artificial neural network parameters. Finally, by a number of calculating examples, compared to the model-based model order reduction method, data-driven model order reduction method is non-intrusive, is not limited to state variable degrees of freedom. We can find that the data-driven model order reduction method is better than the model-based model order reduction method in both computation time and precision, and has good approximation properties.

Keywords Data-driven model order reduction · Time-dependent partial differential equations · State variable equations · Artificial neural network · Non-intrusive

✉ Hui Xu
dr.hxu@sjtu.edu.cn

Fangxiong Cheng
2785964332@qq.com

Xinlong Feng
fxlmath@xju.edu.cn

[1] College of Mathematics and System Sciences, Xinjiang University, Urumqi 830046, People's Republic of China

[2] School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China

1 Introduction

The numerical solution of parameterized time-dependent partial differential equations (PDEs) has always been a hot topic. Because the computational cost of such problems can be high, especially when high precision solutions are required on finer grids and smaller time steps. Therefore, it is necessary to seek an approximate numerical solution method of parameterized time-dependent PDEs which can save time cost and storage space, so that people can get the fast and reliable approximate solution of time-dependent PDEs in the parameter space of interest.

Model order reduction (MOR) method is a common method in optimal design, optimal control and inverse problem applications. As early in [1], a traditional model-based reduced basis (RB) method for PDEs is introduced, and the methods of generating RB space include proper orthogonal decomposition (POD) and greedy algorithm. As we know, the classical model reduction methods include Pade approximation method, Krylov subspace method, balanced truncation method and POD–Galerkin projection method, etc. Even though the POD–Galerkin method lacks a priori guarantee of stability and convergence for complex nonlinear problems, some attempts are made to use the POD–Galerkin method to solve the Burger's equations and hydrodynamic systems [2, 3].

Due to the black box nature of artificial neural network (ANN), people mainly consider approximating the governing equation through data [4–7]. Another method combines RB method and neural network to solve PDEs by data-driven approximation [8]. In recent years, on the basis of the traditional MOR method, some good results have been obtained by combining ANN. For example, using POD to construct a reduced basis, the mapping between time parameters and RB projection coefficient is approximated to a regression model [9]. A reduction method based on POD and temporal convolutional neural network (CNN) is proposed in [10], where the basis function is obtained through POD, and the coefficient of the basis function is taken as the low-dimensional feature, and the time-domain CNN is used to construct the low-dimensional feature prediction model. In [11], POD is used to construct a basis function with special properties, and radial basis function method is used to estimate the undetermined coefficients of the reduced order model by constructing the reduced order model to meet the initial and boundary value conditions. In addition, the author proposed a machine learning method to identify nonlinear dynamic systems from time series data by combining the multi-step time-stepping format of numerical analysis with the deep neural network in [12]. Of course, there are other non-intrusive MOR approaches for specific problems [13–15].

In recent years, a model reduction method of parameterized time-dependent PDEs based on deep learning algorithm has been proposed, and good results have been achieved. For example, in [16], the author combined CNN and feed forward neural network to propose a MOR method for parameterized time-dependent PDEs, it can not only effectively provide a solution for parameterized cardiac electrophysiological problems [17], and through further research, it can be applied to solve complex elastic dynamics and fluid mechanics complex problems [18]. Recently, physical information based neural networks (PINNs) have been proposed to solve PDEs [19], now it has been successfully used to solve the model reduction problem in [20]. In [21], the convolutional autoencoder is used to deal with the nonlinear reduction of spatial degrees of freedom and the long and short term memory network is used for time evolution.

In this work, we mainly consider data-driven MOR techniques for parameterized time-dependent PDEs. By discretization of space variables, the first order differential equations

of time variables, namely the state variable equations, are obtained. In [22], an investigation of parameterized dynamical systems based on projection MOR method, the necessity of MOR method for parameterized dynamical systems is clarified. The governing equation is approximated by data at the numerical level. And a simplified model of linear system derived by data-driven time-domain Loewner framework is mainly considered in [23]. And in [24], ANN is used to reduce the order of the model of time-dependent differential equations, which is mainly aimed at large-scale ordinary differential equations and partial time-dependent PDEs, and has been successfully applied to the effective simulation of parameterized cardiac electrical problems [25].

We mainly consider the HF model of the time-invariant dynamic system with input $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ and output $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ as follows

$$\begin{cases} \dot{\mathbf{S}}(t) = \mathbf{F}(\mathbf{S}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{S}(0) = \mathbf{S}_0 \\ \mathbf{y}(t) = \mathbf{G}(\mathbf{S}(t)), & t \in (0, T], \end{cases} \tag{1}$$

where $\mathbf{S}(t) \in \mathbb{R}^N$ represents the HF state variable of the system, with $\mathbf{F} : \mathbb{R}^N \times \mathbb{R}^{N_u} \to \mathbb{R}^N$ and $\mathbf{G} : \mathbb{R}^N \to \mathbb{R}^{N_y}$. And $\mathbf{u}(t)$ represent the physical or geometric properties of the system, either time-dependent or time-invariant, such as initial and boundary value conditions, convection term coefficients, diffusion term coefficients, etc. The complexity of the system is represented by the dimension of state variables.

The essence of the MOR method is to approximate the multi-dimensional physical process changing with time by describing it in low dimension, and replace the original high-order model with the generated low-order model, which should satisfy: (a) the approximation error is small enough; (b) the reduced order model has high computational efficiency. So we can get the general form of the reduced order model as follows

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathbf{f}(\mathbf{s}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{s}(0) = \mathbf{s}_0 \\ \widetilde{\mathbf{y}}(t) = \mathbf{g}(\mathbf{s}(t)), & t \in (0, T], \end{cases} \tag{2}$$

where $\mathbf{s}(t) \in \mathbb{R}^n$ ($n \ll N$) represents the low-order state variable of the MOR system, with $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^{N_u} \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{N_y}$, where $\mathbf{f}$ and $\mathbf{g}$ are far less difficult to calculate than $\mathbf{F}$ and $\mathbf{G}$.

From Eqs. (1) to (2), there are two ways to work: one is the model-based MOR method and the other is the data-driven MOR method. When the model of many systems is unknown, and we can only get the relevant data of the system through some detectors or measuring instruments. At this time, model-based MOR method is not available. In addition, although the model-based MOR method can deduce the corresponding error analysis from the model itself, it also has some limitations for strong nonlinear problems. For the above cases, the data-driven MOR method, as a non-intrusive MOR method that relies only on data, is a good choice for approximating HF models.

For the model-based MOR method, the POD–Galerkin projection is an efficient order reduction method, which has been widely used in model reduction in the field of PDEs. We consider approximating our full order state space $\mathbb{R}^N$ with a low-dimensional subspace span$\{\mathbf{V}\}$, where $\mathbf{V} \in \mathbb{R}^{N \times n}$ is a matrix whose columns are the basis for the subspace. Therefore, under the Petrov–Galerkin mapping [1], the reduced model can be expressed as

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathbf{W}^{\mathrm{T}}\mathbf{F}(\mathbf{V}\mathbf{s}(t), \mathbf{u}(t)), & t \in (0, T] \\ \mathbf{s}(0) = \mathbf{W}^{\mathrm{T}}\mathbf{S}_0 \\ \widetilde{\mathbf{y}}_{RB}(t) = \mathbf{G}(\mathbf{V}\mathbf{s}(t)), & t \in (0, T], \end{cases} \tag{3}$$

the choice of $\mathbf{W}$ and $\mathbf{V}$ corresponds to different projection methods.

For the data-driven MOR method, it is a classic problem of function approximation to construct steady nonlinear model from known data, while it is a challenging task to build unsteady nonlinear model. In the identification of nonlinear systems [26], the author proposes an internal dynamics model, which simultaneously estimates the state of the model, and the high complexity of the parameters involved, and realizes the nonlinear state-space model with the form (2). Where $\mathbf{f}$ and $\mathbf{g}$ are parameterized nonlinear mappings. The internal state of the model can be viewed as a manual tool for implementing the required dynamic inputs and outputs.

The paper is organized as follows. In Sect. 2, we will introduce data-driven MOR method about ANN in machine learning. And we give the sensitivity calculation of ANNs for network parameters in Sect. 3. In Sect. 4, some numerical examples are given to show that the proposed method is effective and reliable. Finally, we end with a short conclusion in Sect. 5.

2 Model Reduction Strategy

The main idea of the MOR method is to approximate the high-dimensional physical process that changes with time and replace the original high-order model with the low-order model, so as to reduce the calculation dimension, reduce the calculation amount, save the calculation time and CPU load.

2.1 Data-Driven MOR Method

For the data-driven MOR method, the reduced order model is built from the given data in the early Loewner framework based on the interpolation model reduction technique. Although the classical Loewner framework is suitable for linear models, it is extended to bilinear and quadratic-bilinear systems in [27] and is suitable for mildly nonlinear systems. In [28], the Spare Identification of Nonlinear Dynamics (SIND) technology is proposed to build the dynamics system model based on the measurement of triplet $(\mathbf{S}(t), \mathbf{u}(t), \dot{\mathbf{S}}(t))$ in Eq. (1). However, this method needs to access the full-order state variables and does not reduce the dimension of state variables.

In [26], the author proposes a nonlinear system identification technology called internal dynamics method. This method achieves the required dynamic input and output model with the low-order expression form (2) by reconstructing the information of low-order state variables without accessing the real full-order state variables. Finding a good model complexity is a critical problem for the data-driven reduced order modeling form (2). Therefore, for nonlinear structures $\mathbf{f}$ and $\mathbf{g}$, only some generalized approximators can be used, such as polynomials, neural networks, splines, Fourier series, etc. Here we consider the application of ANN in machine learning [36], i.e., nonlinear maps parameterized by finite real parameters.

Because of the approximation by superpositions of a sigmoidal function [29, 30], the neural network with single hidden layer can approximate any continuous function on the compact set with arbitrary precision. Therefore, the continuous functions $\mathbf{f}$ and $\mathbf{g}$ in Eq. (2) are represented by single hidden layer ANN functions, which is

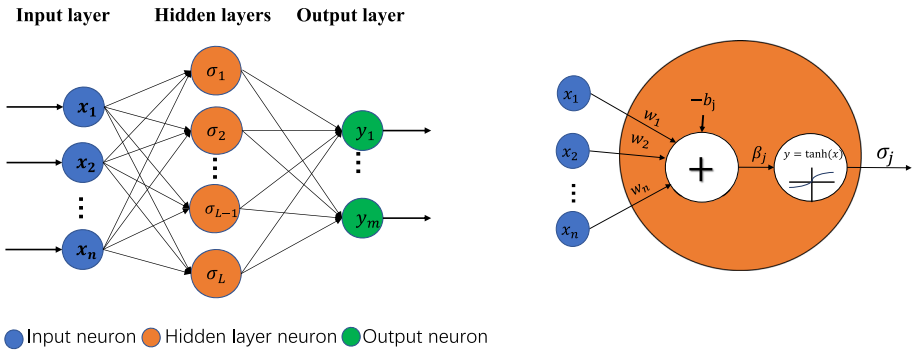

Fig. 1 Left: the structure of ANNs; Right: the inner structure of the i-th neuron

$$\begin{cases} \dot{\mathbf{s}}(t) = \widetilde{\mathbf{f}}(\mathbf{s}(t), \mathbf{u}(t); \mu), & t \in (0, T] \\ \mathbf{s}(0) = \mathbf{s}_0 \end{cases} \tag{4}$$

$$\widetilde{\mathbf{y}}_{ANN}(t) = \widetilde{\mathbf{g}}(\mathbf{s}(t); \nu), \qquad t \in (0, T],$$

where μ and ν are parameters, representing the weight and bias of neural network function $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ respectively. The hypothesis space of $\widetilde{\mathbf{f}}$ is $\widehat{\mathcal{F}} := \mathcal{C}^0(\mathbb{R}^n \times \mathbb{R}^{N_u}; \mathbb{R}^n)$, which is $\widetilde{\mathbf{f}} \in \widehat{\mathcal{F}}$, therefore $\widetilde{\mathbf{g}} \in \widehat{\mathcal{G}} := \mathcal{C}^0(\mathbb{R}^n; \mathbb{R}^{N_y})$ and the initial vector value $\mathbf{s}_0 \in \mathcal{S}_n \equiv \mathbb{R}^n$. Because of the reduced state variables are only auxiliary variables for the evolution of the dynamic system over time without any physical explanation. Therefore, the dimension of the reduced state variable can be chosen with great freedom.

2.2 Artificial Neural Network

A neural network is an operational model consisting of a large number of neurons connected to each other, usually approximating a logical strategy or function. The output of the network depends on the network structure, connection mode, weight and activation function [31]. As shown in Fig. 1, neural network is generally composed of input layer, hidden layer and output layer. The hidden layer can be multi-layer or single-layer. Each neuron receives input signals from the previous layer of neurons, and these input signals are transmitted through weighted connections. The total information received by the neuron is compared with the threshold value of the neuron, and the output of the neuron is processed by activation function.

Neural network is a mathematical model containing many parameters, which is derived from several functions such as $\sigma_j = f_{act}(\sum_i w_i x_i - b_j)$ nested with each other. In this work, we use the fully connected single-hidden layer neural network structure, and use the sigmoidal activation function, which is $f_{act}(x) = \tanh(x)$, this provides a mathematical theory to guarantee for solving the sensitivity of neural network parameters.

2.2.1 Data Preprocessing

In the training process of ANN, we generally use data including training set and test set. Sometimes, in order to select appropriate hyper-parameters, such as the number of hidden neurons, we use a validation set. Sometimes, we use 20% of the training set as a validation

set to update the hyper-parameters in time by evaluating the generalization error in the neural network training process. And the training set and the test set use the same dataset generation process, that is to say, they are equally distributed and independent of each other. In some practical problems, the sample data we get is multi-dimensional, and a sample may contain multiple-feature information. In order to eliminate the influence of different dimensions and orders of magnitude, we normalise the data, i.e. applying the idea of coordinate change to make all the data value in the interval $[-1, 1]$. In this work, the initial weight and bias are Gaussian random distributions with mean of zero and variance of $1/N_s$ (where N_s is the number of training sets).

2.3 Optimizing Strategy

First we consider the case where the input and output data are continuous in time, the collection of N_s input-output pairs $\{\widehat{\mathbf{u}}_j(t), \widehat{\mathbf{y}}_j(t)\}$, $j = 1, \cdots, N_s$, where $\widehat{\mathbf{u}}_j(t) \in \mathcal{U} := \mathcal{C}^0([0, T]; \mathbb{R}^{N_u})$ and $\widehat{\mathbf{y}}_j(t) \in \mathcal{Y} := \mathcal{C}^0([0, T]; \mathbb{R}^{N_y})$. For the reconstructed low-order model (4), where the model quality is measured by the error function between the low-order model output and the HF model output, namely, the loss function is obtained as follows

$$J = \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \left| \widehat{\mathbf{y}}_j(t) - \widetilde{\mathbf{g}}\left(\mathbf{s}_j(t); \nu\right) \right|^2 dt. \tag{5}$$

Here, (5) is taken as the loss function and the reduced state variable equations and initial conditions in (4) as constraints. Therefore, the following constrained optimization problem is obtained

$$\begin{cases} \min\limits_{\widetilde{\mathbf{f}} \in \widehat{\mathcal{F}}, \widetilde{\mathbf{g}} \in \widehat{\mathcal{G}}, \mathbf{s}_0 \in \mathcal{S}_n} & \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \left| \widehat{\mathbf{y}}_j(t) - \widetilde{\mathbf{g}}\left(\mathbf{s}_j(t); \nu\right) \right|^2 dt \\ \text{s.t.} & \dot{\mathbf{s}}_j(t) = \widetilde{\mathbf{f}}\left(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t); \mu\right), \quad t \in (0, T], \quad j = 1, \ldots, N_s \\ & \mathbf{s}_j(0) = \mathbf{s}_0, \qquad\qquad\qquad\quad j = 1, \ldots, N_s. \end{cases} \tag{6}$$

We need to find the optimal $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ in the hypothesis space $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$, respectively, and look for an appropriate representation of the initial value of $\mathbf{s}_0$ in $\mathcal{S}_n$.

Since we use ANN to implicitly build the mapping relationship between input set $\mathcal{U}$ and output set $\mathcal{Y}$, we have mapping set $\Phi = \{\varphi : \mathcal{U} \to \mathcal{Y}\}$, where input-output mapping φ is related to the choice of $\widetilde{\mathbf{f}}$, $\widetilde{\mathbf{g}}$ and $\mathbf{s}_0$, and we write it as $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0}$. Because the representation of the low-order initial state variable $\mathbf{s}_0$ in $\mathcal{S}_n$ is not unique, the representation of $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0}$ is not unique either. For example, for any reversible and regular mapping $\mathbf{q} : \mathbb{R}^n \mapsto \mathbb{R}^n$, when $\widehat{\mathbf{s}} = \mathbf{q}(\mathbf{s})$, and have

$$\widehat{f}(\widehat{\mathbf{s}}, \mathbf{u}) = (\nabla \mathbf{q} \circ \mathbf{q}^{-1}) \widetilde{f}(\mathbf{q}^{-1}(\widehat{\mathbf{s}}), \mathbf{u}), \qquad \widehat{\mathbf{g}}(\widehat{\mathbf{s}}) = \widetilde{\mathbf{g}}(\mathbf{q}^{-1}(\widehat{\mathbf{s}})), \qquad \widehat{\mathbf{s}_0} = \mathbf{q}(\mathbf{s}_0). \tag{7}$$

Then we have $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0} = \varphi_{\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{s}_0}}$, i.e. the input-output mappings $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0}$ and $\varphi_{\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{s}_0}}$ are equivalent, the input-output mapping $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0}$ is ill-posed. In particular, if the hypothesis space is large enough to contain both $\varphi_{\widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}, \mathbf{s}_0}$ and $\varphi_{\widehat{\mathbf{f}}, \widehat{\mathbf{g}}, \widehat{\mathbf{s}_0}}$, optimization problem (6) loses its representation uniqueness, which will greatly reduce the performance of the optimization algorithm. Therefore, we can make $\mathbf{s}_0$ and $\widetilde{\mathbf{g}}$ representational unique by limiting the size of their hypothesis space by imposing certain constraints on them.

2.3.1 Specific Constraints on $\mathbf{s}_0$

For the input-output mapping $\varphi_{\widetilde{\mathbf{f}},\widetilde{\mathbf{g}},\mathbf{s}_0}$, we consider the invertible transformation $\widehat{\mathbf{s}} = \mathbf{q}(\mathbf{s}) = \mathbf{s} - \mathbf{s}_0$, and by transforming (7), we obtain its equivalent model $\varphi_{\widehat{\mathbf{f}},\widehat{\mathbf{g}},\widehat{\mathbf{s}}_0}$, such that $\widehat{f}(\widehat{\mathbf{s}}, \mathbf{u}) = (\nabla \mathbf{q} \circ \mathbf{q}^{-1})\widetilde{f}(\mathbf{q}^{-1}(\widehat{\mathbf{s}}), \mathbf{u})$, $\widehat{\mathbf{g}}(\widehat{\mathbf{s}}) = \widetilde{\mathbf{g}}(\mathbf{q}^{-1}(\widehat{\mathbf{s}}))$, and $\widehat{\mathbf{s}}_0 = \mathbf{q}(\mathbf{s}_0) = \mathbf{0}$. The corresponding reduced state variable equation will have the following form

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathbf{f}(\mathbf{s}(t), \mathbf{u}(t)), \quad t \in (0, T], \\ \mathbf{s}(0) = \mathbf{0} \end{cases} \tag{8}$$

and the corresponding optimization problem is obtained as follows

$$\begin{cases} \min\limits_{\widetilde{\mathbf{f}} \in \widehat{\mathcal{F}}, \widetilde{\mathbf{g}} \in \widehat{\mathcal{G}}} \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \left| \widehat{\mathbf{y}}_j(t) - \widetilde{\mathbf{g}}\left(\mathbf{s}_j(t); \nu\right) \right|^2 dt \\ \quad \text{s.t.} \quad \dot{\mathbf{s}_j}(t) = \widetilde{\mathbf{f}}\left(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t); \mu\right), \quad t \in (0, T], \quad j = 1, \ldots, N_s \\ \qquad\quad \widehat{\mathbf{s}_j}(0) = \mathbf{0}, \qquad\qquad\qquad j = 1, \ldots, N_s. \end{cases} \tag{9}$$

Compared with (6), the reduced initial state variable $\mathbf{s}_0$ is fixed to the zero vector by the translation operator, which is $\mathcal{S}_n = \{\mathbf{0}\}$, and we refer to as output-outside-the state (OOS) (the output is a function of the reduced state variables). The optimization variables are only neural network functions $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ and we only needs to find the optimal $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ in the hypothesis space $\widehat{\mathcal{F}}$ and $\widehat{\mathcal{G}}$, respectively. This reduces the scope of the hypothesis space to some extent.

2.3.2 Specific Constraints on $\widetilde{\mathbf{g}}$ and $\mathbf{s}_0$

Since there is a great degree of freedom in the selection of the reduced state variables $\mathbf{s}(t)$, when $n \geq N_y$, we assume that the first N_y reduced state variables provide all information about the relevant output, and then we force the former N_y reduced state variable $\mathbf{s}_1$, $\mathbf{s}_2$, …, $\mathbf{s}_{N_y}$ to coincide with the output $\mathbf{y}_1$, $\mathbf{y}_2$, …, $\mathbf{y}_{N_y}$, that is, there exists a self-mapping $I_{N_y}(\mathbf{x})$, such that $I_{N_y}(\mathbf{x}) = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{N_y})^{\mathrm{T}}$.

For the input-output mapping $\varphi_{\widetilde{\mathbf{f}},\widetilde{\mathbf{g}},\mathbf{s}_0}$, we consider the invertible transformation $\mathbf{q}_1 : \mathbb{R}^n \mapsto \mathbb{R}^{n-N_y}$. Therefore, we can obtain an invertible and regular mapping $\mathbf{q} = (I_{N_y}^{\mathrm{T}}(s), \mathbf{q}_1^{\mathrm{T}}(s))^{\mathrm{T}}$, and apply $\mathbf{q}$ to the transformation formula (7) to obtain the corresponding equivalent mapping $\varphi_{\widehat{\mathbf{f}},\widehat{\mathbf{g}},\widehat{\mathbf{s}}_0}$, where

$$\widehat{f}(\widehat{\mathbf{s}}, \mathbf{u}) = (\nabla \mathbf{q} \circ \mathbf{q}^{-1})\widetilde{f}(\mathbf{q}^{-1}(\widehat{\mathbf{s}}), \mathbf{u}), \qquad \widehat{\mathbf{g}}(\widehat{\mathbf{s}}) = I_{N_y}(\widehat{\mathbf{s}}), \qquad \widehat{\mathbf{s}}_0 = (I_{N_y}^{\mathrm{T}}(\mathbf{s}_0), \mathbf{q}_1^{\mathrm{T}}(\mathbf{s}_0))^{\mathrm{T}}, \tag{10}$$

where $I_{N_y}^{\mathrm{T}}(\mathbf{s}_0) = \mathbf{y}_0$. We can again take the transformation (7), where the transformation is mapped to $\mathbf{q}_2(\mathbf{s}) = \mathbf{q}(\mathbf{s}) - (\mathbf{0}^{\mathrm{T}}, \mathbf{q}_1^{\mathrm{T}}(\mathbf{s}_0))$. Finally, we get the reduced model in the following form

$$\begin{cases} \dot{\mathbf{s}}(t) = \mathbf{f}(\mathbf{s}(t), \mathbf{u}(t)), \quad t \in (0, T] \\ \mathbf{s}(0) = (\mathbf{y}_0^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}}), \end{cases} \tag{11}$$

and then the corresponding optimization problem is obtained as follows

$$\begin{cases} \min\limits_{\widetilde{\mathbf{f}} \in \widehat{\mathcal{F}}} \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \left| \widehat{\mathbf{y}}_j(t) - I_{N_y}\left(\mathbf{s}_j(t)\right) \right|^2 dt \\ \text{s.t.} \quad \dot{\mathbf{s}}_j(t) = \widetilde{\mathbf{f}}\left(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t); \mu\right), \quad t \in (0, T], \quad j = 1, \ldots, N_s \\ \qquad \mathbf{s}_j(0) = \mathbf{s}_0, \qquad\qquad j = 1, \ldots, N_s, \end{cases} \tag{12}$$

where $\mathbf{s}_0 = (\mathbf{y}_0^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$. Compared with optimization problems (9) and (12), by setting the neural network function $\widetilde{\mathbf{g}}$ as a self-mapping, which is $\widehat{\mathcal{G}} = \{I_{N_y}(\mathbf{x})\}$ and $\mathcal{S}_n = \{(\mathbf{y}_0^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}\}$, we refer to as output-inside-the state (OIS) (the output is part of the reduced state variables). The optimized variable is only the neural network function $\widetilde{\mathbf{g}}$. We therefore need to find the optimal $\widetilde{\mathbf{f}}$ in the hypothesis space $\widehat{\mathcal{F}}$. This greatly reduces the scope of the hypothesis space, but there is one premise that satisfies $n \geq N_y$.

Remark 2.1 Since the optimization problem s(9) and (12) are to some extent simple versions of basic optimization problem (6), we take the optimization problem (6) as an example to carry out the optimization solution calculation.

The Lagrange multiplier method is used to solve the optimization problem (6) with constraints, and the Lagrange functional as follow

$$\begin{aligned}\mathcal{J}\left(\mu, \nu, \left\{\mathbf{s}_j(t)\right\}, \left\{\mathbf{w}_j(t)\right\}\right) =& \frac{1}{2}\sum_{j=1}^{N_s}\int_0^T \left|\widehat{\mathbf{y}}_j(t) - \widetilde{\mathbf{g}}\left(\mathbf{s}_j(t); \nu\right)\right|^2 dt \\ &- \sum_{j=1}^{N_s}\int_0^T \mathbf{w}_j(t)\cdot\left(\dot{\mathbf{s}}_j(t) - \widetilde{\mathbf{f}}\left(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t); \mu\right)\right) dt \\ &- \sum_{j=1}^{N_s}\mathbf{w}_j(0)\cdot\left(\mathbf{s}_j(0) - \mathbf{s}_0\right).\end{aligned} \tag{13}$$

By setting the derivative of $\mathcal{J}$ with respect to the dual variable to zero, we get the equation for the Lagrange multiplier as follows

$$\begin{cases} -\dot{\mathbf{w}}_j(t) = \nabla_{\mathbf{s}}\widetilde{\mathbf{g}}(\mathbf{s}_j(t); \nu)^{\mathrm{T}}(\nabla_{\mathbf{s}}(\widetilde{\mathbf{g}}(\mathbf{s}_j(t); \nu) - \widehat{\mathbf{y}}_j(t)) + \nabla_{\mathbf{s}}\widetilde{\mathbf{f}}(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t), \mu)^{\mathrm{T}}\mathbf{w}_j(t), \quad t \in [0, T) \\ \mathbf{w}_j(T) = \mathbf{0}. \end{cases} \tag{14}$$

Once the Lagrangian multiplier information is available, the optimization of the neural network functions $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ requires the first-order sensitivity of the Lagrangian function (13) with respect to the neural network parameters μ and ν, i.e.,

$$\begin{cases} \nabla_{\mu}\mathcal{J} = \sum_{j=1}^{N_s}\int_0^T \nabla_{\mu}\widetilde{\mathbf{f}}(\mathbf{s}_j(t), \widehat{\mathbf{u}}_j(t); \mu)^{\mathrm{T}}\mathbf{w}_j(t)dt, \\ \nabla_{\nu}\mathcal{J} = \sum_{j=1}^{N_s}\int_0^T \nabla_{\nu}\widetilde{\mathbf{g}}(\mathbf{s}_j(t); \nu)^{\mathrm{T}}(\widetilde{\mathbf{g}}(\mathbf{s}_j(t); \nu) - \widehat{\mathbf{y}}_j(t))dt, \end{cases} \tag{15}$$

where $\mathbf{w}_j(t)$ satisfies Eqs. (14). Here we have obtained the solution to the optimization problem of time continuity, and then we are required to derive the discrete equivalent form of (14) and (15) so that we can numerically solve problem (6).

2.4 Discrete Solution

In order to solve the above problem numerically, we obtain the corresponding discrete problem for the time-discrete equation of the reduced state variable (4) and loss function (5). Let's consider the constant time step Δt, such that $t_k = k\Delta t$, $k = 0, 1, \cdots, M_j$, then we will collect time instants $0 = t_0 < t_1 < \cdots < t_{M_j} = T$. We use $u_j^k = \hat{u}_j(t_k)$, $s_j^k = s_j(t_k)$

and $y_j^k = \hat{y}_j(t_k)$ to represent the corresponding input values, the reduced state variables and output values at the discrete time. The time discrete form corresponding to the loss function (5) is

$$J = \tfrac{1}{2}\Delta t \sum_{j=1}^{N_s} \sum_{k=0}^{M_j-1} |y_j^k - \widetilde{\mathbf{g}}(s_j^k; \nu)|^2, \tag{16}$$

and the corresponding time discrete optimization problem, namely, the time discrete form of (6) is

$$\begin{cases} \min\limits_{(\mu,\nu)\in\mathbb{R}^{N_{\widetilde{\mathbf{f}}}+N_{\widetilde{\mathbf{g}}}}} & \frac{1}{2}\sum_{j=1}^{N_s}\sum_{k=0}^{M_j-1}\left|\mathbf{y}_j^k - \widetilde{\mathbf{g}}\left(\mathbf{s}_j^k; \nu\right)\right|^2 \\ \text{s.t.} & \mathbf{s}_j^{k+1} = \mathbf{s}_j^k + \Delta t\widetilde{\mathbf{f}}\left(\mathbf{s}_j^k, \mathbf{u}_j^k; \mu\right), \quad j = 1, \cdots, N_s, k = 1, \ldots, M_j - 1 \\ & \mathbf{s}_j^0 = \mathbf{s}_0, \qquad\qquad j = 1, \ldots, N_s. \end{cases} \tag{17}$$

The only difference is that, for the discrete problem (17), we need to find the optimal network parameter information, that is, we need to find the optimal μ^{T}, ν^{T} in the parameter space $\mathbb{R}^{N_{\widetilde{\mathbf{f}}}}$ and $\mathbb{R}^{N_{\widetilde{\mathbf{g}}}}$ of the network respectively, so as to minimize the loss function (16).

For the time continuity problem, Lagrange multiplier method is adopted. Therefore, we discretize the Eq. (13) in time and obtain the fully discretized Lagrange equation as follows

$$\begin{aligned} \mathcal{J}\left(\mu, \nu, \left\{\mathbf{s}_j^k\right\}, \left\{\mathbf{w}_j^k\right\}\right) =& \frac{1}{2}\Delta t \sum_{j=1}^{N_s}\sum_{k=0}^{M_j-1}\left|\mathbf{y}_j^k - \widetilde{\mathbf{g}}\left(\mathbf{s}_j^k; \nu\right)\right|^2 \\ &- \sum_{j=1}^{N_s}\sum_{k=1}^{M_j}\mathbf{w}_j^k \cdot \left(\mathbf{s}_j^k - \mathbf{s}_j^{k-1} - \Delta t\widetilde{\mathbf{f}}\left(\mathbf{s}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu\right)\right) \\ &- \sum_{j=1}^{N_s}\mathbf{w}_j^0 \cdot \left(\mathbf{s}_j^0 - \mathbf{s}_0\right). \end{aligned} \tag{18}$$

And the parameter sensitivity calculation of the corresponding discrete form, namely the time discrete form of (14)

$$\begin{aligned} \nabla_\mu \mathcal{J} &= \Delta t \sum_{j=1}^{N_s}\sum_{k=1}^{M_j}\nabla_\mu \widetilde{\mathbf{f}}(\mathbf{s}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^{\mathrm{T}}\mathbf{w}_j^k, \\ \nabla_\nu \mathcal{J} &= \Delta t \sum_{j=1}^{N_s}\sum_{k=0}^{M_j-1}\nabla_{\mathbf{s}}\widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu)^{\mathrm{T}}(\widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu) - \mathbf{y}_j^k). \end{aligned} \tag{19}$$

Where the Lagrange multiplier satisfies the time-discrete version of Eq. (14), it is given by

$$\begin{cases} \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t\nabla_{\mathbf{s}}\widetilde{\mathbf{f}}(\mathbf{s}_j^k, \mathbf{u}_j^k; \mu)^{\mathrm{T}}\mathbf{w}_j^{k+1} + \Delta t\nabla_{\mathbf{s}}\widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu)^{\mathrm{T}}(\widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu) - \mathbf{y}_j^k), \\ \mathbf{w}_j^{M_j} = 0. \end{cases} \tag{20}$$

2.5 Selection of ANN Structure

The choice of network structure can adjust the capacity of neural network model, that is, the ability of neural network to fit various functions. The selection of model capacity can

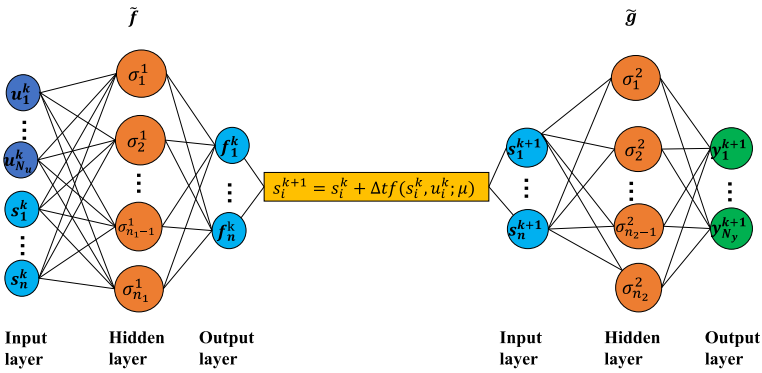

Fig. 2 The ANN structure of the OOS neural network

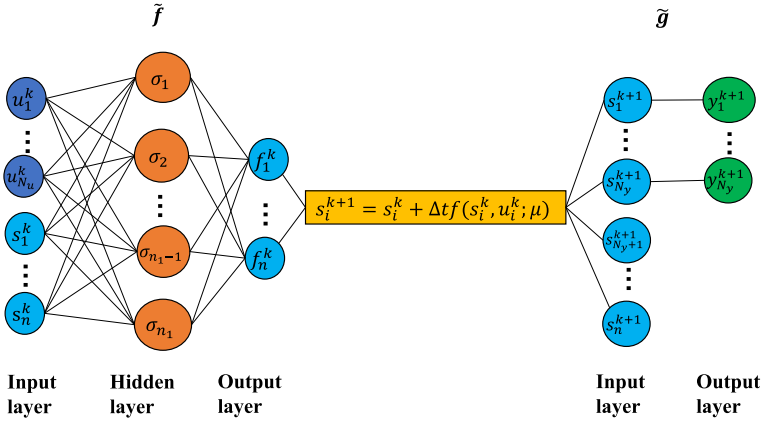

Fig. 3 The ANN structure of the OIS neural network

control whether the model is biased to over-fitting or under-fitting. Models with low capacity may find it difficult to fit the training set, while models with high capacity may over-fitting [32]. One way to control model capacity is to choose the appropriate hypothesis space, which depends on the selection of ANN structure.

First, for OOS, since the output is a function of the reduced order state variables, the overall network structure is shown in Fig. 2, which is also called OOS neural network. The output of this network is not limited by the dimension of the reduced state variable, and it is suitable for model reduction of dynamic system with more output information. And by forcing the output $\widetilde{\mathbf{y}}(t) \in \mathbb{R}^{N_y}$ to be consistent with the first N_y reduced state variables, it indicates OIS. Since the neural network function $\widetilde{\mathbf{g}}$ is the mapping of the reduced state variable to the network output itself, the overall network structure is shown in Fig. 3, which is also called OIS neural network.

Through the comparison of the two network structures, we can find that the structure of the OOS neural network is more complex, that is, its neural network model has more capacity. In general, as the neural network model capacity increases for the same problem, the training error will decrease until it asymptotes to its smallest possible error (assuming there is a minimum for the error measure).

2.6 Optimizing Algorithm

The nonlinear least squares problem is of the following form

$$\min_{\mathbf{x}\in\mathbb{R}^{N_{\widetilde{\mathbf{f}}}+N_{\widetilde{\mathbf{g}}}}} f(\mathbf{x}) = \tfrac{1}{2} \parallel \mathbf{r}(\mathbf{x}) \parallel^2, \tag{21}$$

where $\mathbf{x} = (\mu^{\mathrm{T}}, \nu^{\mathrm{T}})^{\mathrm{T}}$ represents the goals that we need to optimize, and $\mathbf{r}(\mathbf{x})$ is the residual vector. Because the second-order information item of the derivative in the Hessian matrix is usually difficult to be calculated, we're approximating the Hessian matrix with low cost calculations. The two most popular algorithms are Gauss–Newton algorithm and Levenberg–Marquardt (LM) algorithm [26]. Because of possibility of vanishing gradient, we employ the LM algorithm, combined with the linear search algorithm and have

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k(\nabla\mathbf{r}(\mathbf{x}_k)^{\mathrm{T}}\nabla\mathbf{r}(\mathbf{x}_k) + \lambda_k\mathbf{I})^{-1}\nabla\mathbf{r}(\mathbf{x}_k)^{\mathrm{T}}\mathbf{r}(\mathbf{x}_k). \tag{22}$$

And the update of vector $\mathbf{x}$ follows rule $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$, where α_k is the descent step length, which is satisfied Wolfe and Armijo criterion [33].

LM algorithm achieves the advantages of combining Gauss–Newton algorithm and gradient descent method by modifying parameters λ_k, when $\lambda_k \gg 1$, it's close to the gradient descent method, and when $\lambda_k = 0$, it can be considered as Gauss–Newton algorithm [31]. In order to ensure the superlinear convergence of the LM algorithm, we choose $\lambda_k = min\{|r(\mathbf{x}_0)|^2, |\nabla\mathbf{r}(\mathbf{x}_k)^{\mathrm{T}}\nabla\mathbf{r}(\mathbf{x}_k)|\}$.

To evaluate the performance of the proposed data-driven MOR method, the average relative error indicator $\epsilon_{rel} \in \mathbb{R}$ is used and given by

$$\epsilon_{rel}\left(\mathbf{y}_j^k, \widetilde{\mathbf{g}}\left(\mathbf{s}_j^k; \nu\right)\right) = \frac{1}{N_{\mathrm{s}}}\sum_{j=0}^{N_s}\left(\frac{\sqrt{\sum_{k=0}^{M_{j-1}} \parallel \mathbf{y}_j^k - \widetilde{\mathbf{g}}\left(\mathbf{s}_j^k; \nu\right) \parallel^2}}{\sqrt{\sum_{k=0}^{M_{j-1}} \parallel \mathbf{y}_j^k \parallel^2}}\right), \tag{23}$$

and the effectiveness and feasibility of the proposed method are evaluated by the performance of ϵ_{rel} on the test set.

3 Sensitivity Calculation

Since neural network uses weighted staggered connections between neurons, the degree of influence of input variables on output variables cannot be intuitively obtained. The purpose of sensitivity analysis is to calculate the influence factors of network parameters on network output. We can know from Eq. (17) that the relationship between parameter μ and the reduced state variable $\mathbf{s}_j^k$ is implicitly expressed by the reduced state variable equation, so here we calculate the sensitivity with the help of Lagrange function. If we want to calculate the sensitivity of $\Gamma = \Gamma(\mathbf{s}_j^k; \mu, \nu)$ with respect to parameter μ and ν, then the corresponding Lagrange equation is

$$\begin{aligned}\mathcal{J}(\mu, \nu, \{\mathbf{s}_j^k\}, \{\mathbf{w}_j^k\}) =& \Gamma(\mathbf{s}_j^k; \mu, \nu) - \sum_{j=1}^{N_s}\sum_{k=1}^{M_j}\mathbf{w}_j^k \cdot \left(\mathbf{s}_j^k - \mathbf{s}_j^{k-1} - \Delta t\widetilde{\mathbf{f}}\left(\mathbf{s}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu\right)\right) \\ &- \sum_{j=1}^{N_s}\mathbf{w}_j^0 \cdot \left(\mathbf{s}_j^0 - \mathbf{s}_0\right).\end{aligned} \tag{24}$$

By setting the derivative of $\mathcal{J}$ with respect to the dual variables to be zero, we finally obtain the relevant information of Lagrange multipliers $\mathbf{w}_j^k$, for $j = 1, \cdots, N_s$,

$$\begin{cases} \mathbf{w}_j^{M_j} = \dfrac{\partial \Gamma}{\partial \mathbf{s}_j^{M_j}}, \\ \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{s}} \widetilde{\mathbf{f}}(\mathbf{s}_j^k, \mathbf{u}_j^k; \mu) \mathbf{w}_j^{k+1} + \dfrac{\partial \Gamma}{\partial \mathbf{s}_j^k}, \quad k = 1, \cdots, M_j - 1. \end{cases} \tag{25}$$

If our purpose is to calculate the sensitivity of the residual vector $\mathbf{r}$ to parameters μ and ν, since each component of the residual vector function is

$$r^{j,k,h} = \sqrt{\Delta t}(\widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu) - \mathbf{y}_j^k) \cdot \mathbf{e}_h, \qquad j = 1, \ldots, N_s, k = 0, \ldots, M_j - 1, h = 1, \ldots, N_y,$$

where $\mathbf{e}_h$ represents the h-th component of the standard regular basis for $\mathbb{R}^{N_y}$. By setting $\Gamma = r^{j,m,h}$, then we have, for $j = 1, \cdots, N_s, m = 1, \cdots, M_j - 1$,

$$\nabla_\mu r^{j,m,h} = \Delta t \sum_{k=1}^{m} \nabla_\mu \widetilde{\mathbf{f}}(\mathbf{s}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^{\mathrm{T}} \mathbf{w}_j^k, \quad \nabla_\nu r^{j,m,h} = \sqrt{\Delta t} \nabla_\nu \widetilde{\mathbf{g}}(\mathbf{s}_j^m; \nu)^{\mathrm{T}} \mathbf{e}_h, \tag{26}$$

where

$$\begin{cases} \mathbf{w}_j^k = \mathbf{w}_j^{k+1} + \Delta t \nabla_{\mathbf{s}} \widetilde{\mathbf{f}}(\mathbf{s}_j^k, \mathbf{u}_j^k; \mu)^{\mathrm{T}} \mathbf{w}_j^{k+1}, \qquad k = 0, \ldots, m-1 \\ \mathbf{w}_j^m = \sqrt{\Delta t} \nabla_\nu \widetilde{\mathbf{g}}(\mathbf{s}_j^k; \nu)^{\mathrm{T}} \mathbf{e}_h. \end{cases}$$

In the process of neural network optimization, it is necessary to calculate the gradient of the discrete loss function $J = \frac{1}{2}\mathbf{r}^{\mathrm{T}}\mathbf{r}$ with respect to parameters μ and ν. If we have calculated $\nabla_\mu \mathbf{r}$ and $\nabla_\nu \mathbf{r}$, then the gradient of J can be obtained by $\nabla \mathbf{r}^{\mathrm{T}}\mathbf{r}$, i.e.

$$\begin{aligned} \nabla_\mu \mathcal{J} &= \nabla \mathbf{r}_\mu^{\mathrm{T}} \mathbf{r}_\mu + \Delta t \textstyle\sum_{j=1}^{N_s} \sum_{k=1}^{M_j} \nabla_\mu \widetilde{\mathbf{f}}(\mathbf{s}_j^{k-1}, \mathbf{u}_j^{k-1}; \mu)^{\mathrm{T}} \mathbf{w}_j^k, \\ \nabla_\nu \mathcal{J} &= \nabla \mathbf{r}_\nu^{\mathrm{T}} \mathbf{r}_\nu. \end{aligned} \tag{27}$$

We can also calculate $\nabla_\mu \mathcal{J}$ and $\nabla_\nu \mathcal{J}$ from Eqs. (19) and (20). In order to save storage space and computing time, we generally choose the Eqs. (27).

4 Numerical Experiments

In this section, we give some numerical examples of solving PDEs with MOR methods. In the first example we consider a linear system, the heat equation, in which the input parameters depend on its geometric properties. Then we consider nonlinear Burgers equations, a one-dimensional example is given to illustrate the influence of network structure on the effect of the data-driven MOR method. Then, the effects of the data-driven MOR method and the model-based MOR method in a system with high-dimensional output are compared in a two-dimensional example. Finally, we consider a multi-input multi-output (MIMO) system, that is, using some known information of the system to establish a low-order approximation of the related input and output. All the results of this paper can be obtained from online repository https://gitee.com/cfx-one/data-driven-mor.git.

4.1 Linear System

We firstly consider the heat equation in the one-dimensional region $x \in [0, 1]$, with the following form

$$\begin{cases} \dfrac{\partial}{\partial t}\psi(x,t) - \dfrac{\partial^2}{\partial^2 x}\psi(x,t) = 0 & (x,t) \in (0,1) \times (0,T] \\ \psi(0,t) = u_1(t) & t \in (0,T] \\ \psi(1,t) = u_2(t) & t \in (0,T] \\ \psi(x,0) = 0 & x \in (0,1), \end{cases} \tag{28}$$

with $T = 1$, and where the 2-dimensional parameters $\mathbf{u}(t) = [u_1(t), u_1(t)]$ and $\mathbf{u}(t) \in [0, 2]^2$ are Dirichlet boundary conditions at both ends of the region, the output $\mathbf{y}(t) \in \mathbb{R}^2$ is defined as the value of the solution at $x = 0.3$ and $x = 0.8$ that evolves over time. For the HF numerical solution, we adopt forward Euler scheme in time and P_2 finite element in space, with the spatial discretization step of $h = 10^{-2}$ and the time discretization step of $\Delta t = 10^{-2}$.

We selected 200 sets of training sets and 20 sets of testing sets, as well as 50 sets of validation sets. The initial network weight and bias are randomly initialized by Gaussian with mean of 0 and variance of $1/200$. We evaluate the generalization error of neural network training through validation set. Considering that the loss function (13) is non-convex and has many local minima, we employ early stop strategy to avoid over-training. When the error indicator ϵ_{rel} on the validation sets stops as soon as it rises, we may miss out on better choices, so it stops when ϵ_{rel} does not rise for a period of time. In the initial phase, we could iterate 10 times, gradually choosing more iterations, such as the error indicator ϵ_{rel} not improving after 100 iterations, etc.

For such a linear problem, we consider using the OIS neural network, where the dimension of the reduced state variables is $n = 2, 4, 6, 8$ respectively, and the number of neurons in the hidden layer of $\widetilde{\mathbf{f}}$ is $F_{layer} = 2n$, which can approximately double the number of parameters

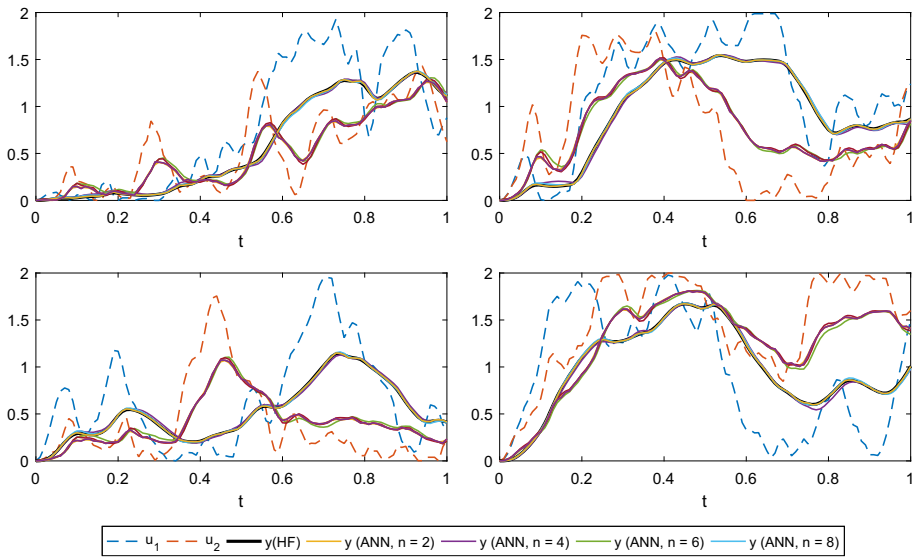

Fig. 4 The results of four different test cases on different ANNs

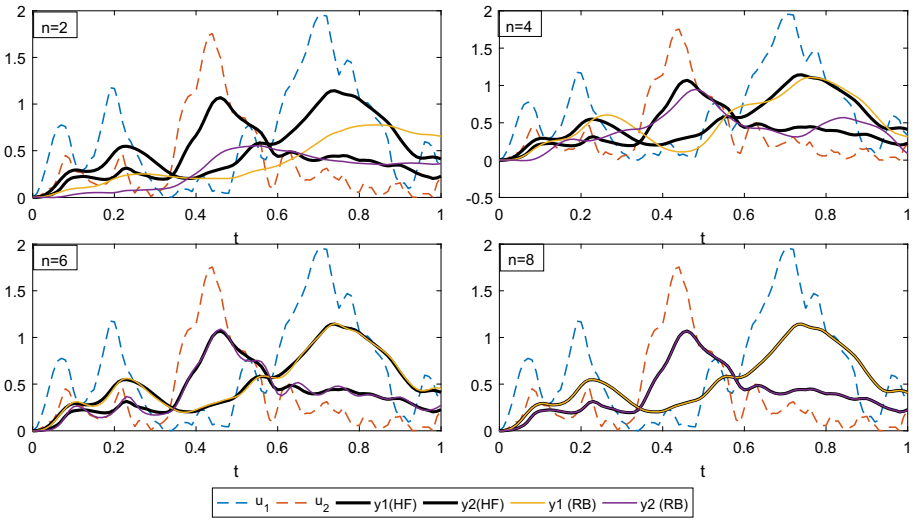

Fig. 5 Test results of a test case in model-based MOR method

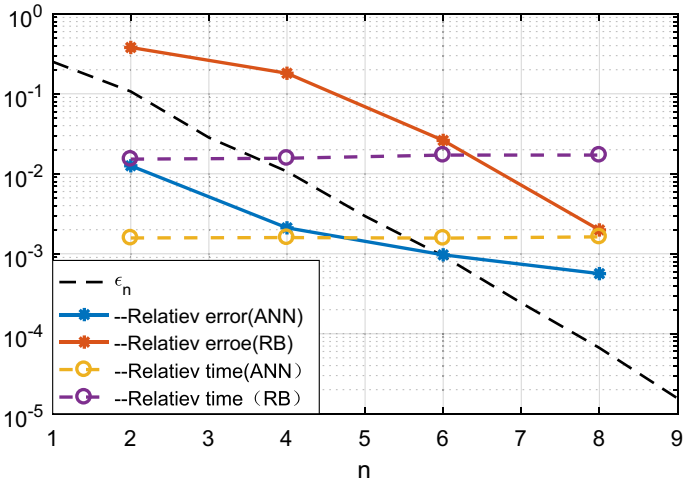

Fig. 6 Relative error and relative time comparison of different MOR methods

of the neural network. We can see the time evolution results of four different ANN models on the four test sets as shown in Fig. 4. Moreover, in Fig. 5, we respectively verified the model-based MOR method on a test set, and it can be found that when the dimension of the reduced state variable is $n = 2, 4$, the output of the model-based MOR method is quite different from the HF output and with the further increase of n, the approximation effect gradually improved.

Finally, we compare the relative error and relative time of the data-driven MOR method and the model-based MOR method on the test set in Fig. 6. Where ε_n represents the proportion of uncaptured energy of the first n bases of POD, and it is referred to as the relative information content of the POD basis. In terms of relative error, the data-driven MOR method always

Table 1 The comparison between the OIS neural network and the OOS neural network

case	the OIS neural network				the OOS neural network			
F_{layer}	2	3	4	5	2	3	4	5
train error	2.17e–02	2.20e–02	2.21e–02	2.24e–02	9.00e–03	2.13e–04	2.09e–04	1.89e–04
test error	2.21e–02	2.24e–02	2.24e–02	2.28e–02	9.30e–03	3.27e–04	2.83e–04	2.57e–04

outperforms the model-based MOR method in the same dimension of state variables. In terms of relative time, the data-driven MOR method always take an order of magnitude less time than the model-based MOR method. So in terms of accuracy and timeliness, the data-driven MOR method has certain advantages.

4.2 Nonlinear System

The Burger's equation, as a nonlinear problem in fluid dynamics, is an important and basic parabolic PDEs. It is of great practical significance to study the MOR problem by the Burger's equation.

Case 1: *The* 1*D Burger's equation*

We consider the Burger's equation described by

$$\begin{cases} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial^2 x} = 0 & (x,t) \in (0,L) \times (0,T] \\ u(0,t) = 0 & t \in (0,T] \\ u(L,t) = 0 & t \in (0,T] \\ u(x,0) = \sin(\pi x) & x \in (0,L), \end{cases} \tag{29}$$

with $T = 1$, $L = 1$ and where the viscosity coefficient $\nu \in [0.01, 0.1]$ is a parameter that affects behaviour of the system (29), it can be either a time-dependent function or a constant. The HF numerical solution is obtained by the traditional finite element method with time step $\Delta t = 10^{-2}$ and space step $h = 3.90 \times 10^{-3}$.

We first deliberately consider the case of the OIS neural network and the OOS neural network on a small training set, with 200 sets of training data and 100 sets of test data. The relevant training results show as Table 1. From the table, we know that when the number of neurons in the hidden layer of $\widetilde{\mathbf{f}}$ is $F_{layer} = 2, 3, 4, 5$, relatively, the relative error of the OOS neural network is nearly 2 orders of magnitude lower than that of the OIS neural network. That is, the OIS neural network we chose is too small to represent the change of reduced state variables, and we call the under-fitting situation. Then we can know that compared with increasing the number of hidden layer neuron, the use of the OOS neural network can be more effective for neural network learning.

In the following, sensitivity analysis is performed on the performance of the OIS neural network, and the complexity of the network is mainly reflected in the number of hidden layer neurons. Therefore, we target the number of neurons of different hidden layers $F_{layer} = 2, 3, 4, 5$. Considering that the reduced state variables are both inputs to the OIS neural network and reduced order variables, without loss of generality, we also consider the dependence of the proposed OIS neural network on the number of hidden layer neurons when the number of the reduced state variables is $n = 1, 2, 3$, respectively. We take any five groups of data from the training set, test set and verification set to test their relative error indicators for different network structures respectively, as shown in Fig. 7.

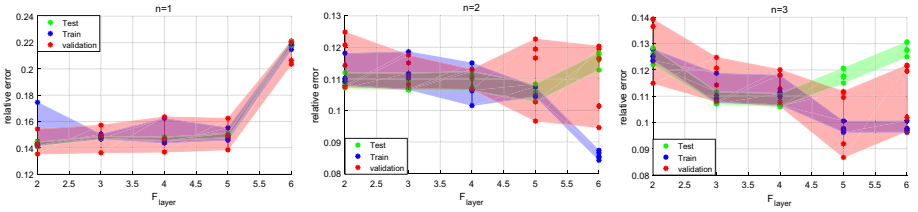

Fig. 7 Comparison of the effect of data-driven MOR method on different dimensional state variables in training set, test set and validation set

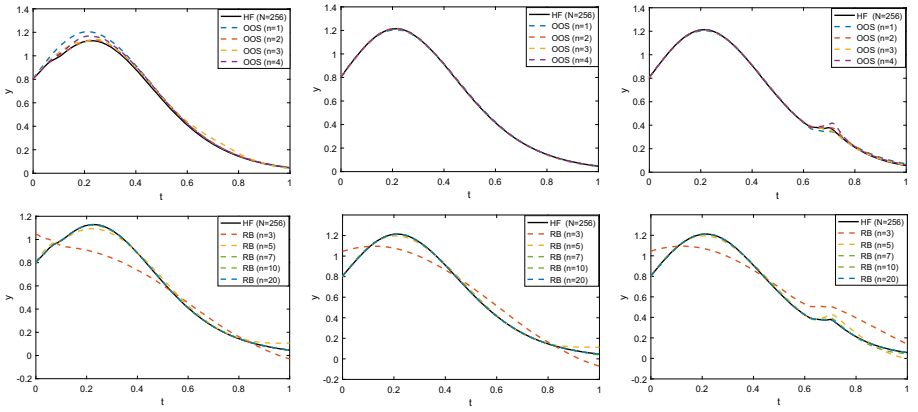

Fig. 8 Comparison of the result of the HF model with those of the different reduced models for three random tests. above: the OOS neural network vs HF model; below: model-based MOR method vs HF model

First of all, when $n = 1$, the whole network structure is insufficient to express the relationship between network input and network output due to the insufficient representation ability of neural network. When $n = 2$ and $n = 3$, as the number of hidden layer neurons increases, the training error gradually decreases. However, by further increasing the number of hidden layer neurons, it is found that even though the training error ϵ_{train} decreases, the test error ϵ_{test} begins to increase, which can lead to over-fitting phenomenon. In this case, we can use ratio $\epsilon_{test}/\epsilon_{train}$ to select the early stop strategy. At the same time, for $F_{layer} = 5$ in the case $n = 2$ and for $F_{layer} = 4$ in the case $n = 3$, the result of OIS neural network represents the mapping between input and output as the optimal network, which is related to the non-uniqueness of network representation mentioned earlier, and also to the multiple local minima of the objective function of our training neural network. In short, the optimal neural network depends not only on the initialization of the neural network itself but also on the selection of the number of hidden layer neurons and the selection of network structure.

According to the prior knowledge obtained above, we consider training a more complex network in a large training data (340 training sets and 160 test sets). Among them, the number of hidden layer neurons of $\widetilde{\mathbf{f}}$ is $F_{layer} = 12$, and the number of hidden layer neurons of $\widetilde{\mathbf{g}}$ is $G_{layer} = 9$. When the number of the reduced state variables are $n = 1, 2, 3, 4$, relatively, good training results can be obtained. Compared with model-based MOR method, data-driven MOR method can get more reliable low-order system. In Fig. 8, we compare the changes of function values with time of the data-driven MOR method and model-based MOR method at $x = 0.3$, that is, $y(t) = u(0.3, t)$.

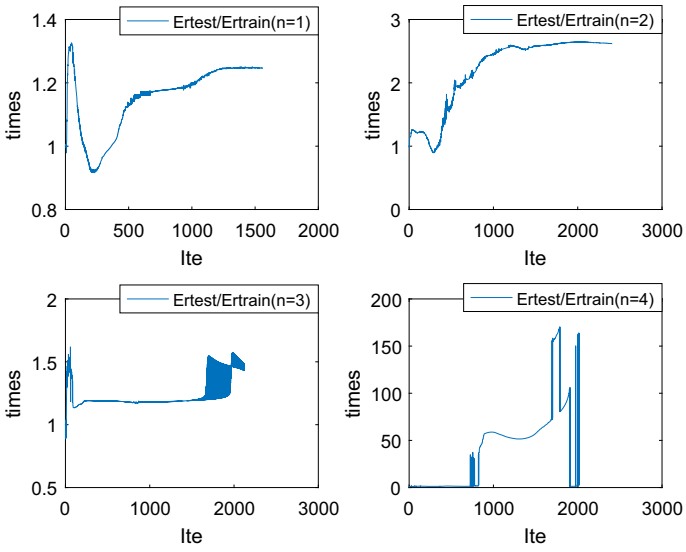

Fig. 9 Comparison of the result of the training set and test set errors

The factors to judge the effect of the reduced order model include: (1) reducing the training error; (2) narrowing the gap between the training error and the test error, that is, the training error and the test error should be kept at the same order of magnitude. Failure to obtain a small enough error on the training set will result in under-fitting phenomenon. If the error on the training set differs greatly from the error on the test set, over-fitting phenomenon will occur.

In Fig. 9, we compare the ratio of training set and test set of neural network with the reduced state variable dimensions $n = 1, 2, 3, 4$, respectively. It can be found that when the reduced state variable dimension $n = 1, 2, 3$, the ratio of training set and test set is at the same order of magnitude. When $n = 4$, with the increase of iteration times, the error between the training set and the test set differs by at least one order of magnitude, or even two orders of magnitude. That is to say, when $n = 4$, the data-driven MOR method appears over-fitting phenomenon, here we can use the early stop strategy to avoid over-fitting.

Case 2: *2D Burger's equation*

We now consider a two-dimensional unsteady Burger's equation, which is a problem with no true solution, has been considered in [34]. The spatial domain is $\Omega = [0, 1]^2$, the equation and its initial and boundary conditions are written as

$$\begin{cases} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial^2 x} + \nu\frac{\partial^2 u}{\partial^2 y} & (x, y, t) \in \Omega \times (0, T] \\ u(x, y, t) = 0 & (x, y, t) \in \partial\Omega \times (0, T] \\ u(x, y, 0) = \sin(2\pi x)\cos(2\pi y) & (x, y) \in \Omega, \end{cases} \tag{30}$$

where the viscosity coefficient $\nu \in [0.01, 0.1]$ is used, and we use the traditional finite element method to obtain the HF numerical solutions at moments $t = 0.3$, $t = 0.6$, and $t = 1$, as shown in Fig. 10.

Since the output of this model is $N_y = 441$, in order to achieve the purpose of order reduction, we can only use the OOS neural network. In order to avoid large parameter space, we only take $G_{layer} = 1$ and $G_{layer} = 2$ as the hidden neurons of $\widetilde{\mathbf{g}}$ network, and

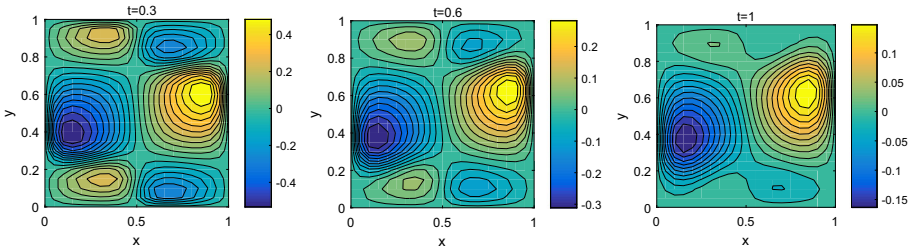

Fig. 10 A random test set of HF solutions at different times

Table 2 Parameter setting and numerical results of the data-driven MOR methods

case	$G_{layer} = 1$ (OOS1)				$G_{layer} = 2$ (OOS2)			
n	1	2	3	4	1	2	3	4
parameters	897	911	929	951	1340	1355	1374	1397
LM time	3.08s	3.17s	3.25s	3.43s	5.88s	6.17s	6.34s	6.59s
train errors	5.48e–02	5.48e–02	5.48e–02	5.48e–02	9.50e–03	8.90e–03	8.80e–03	8.80e–03
test errors	5.80e–02	5.79e–02	5.79e–02	5.79e–02	1.07e–02	1.00e–02	9.90e–03	9.90e–03

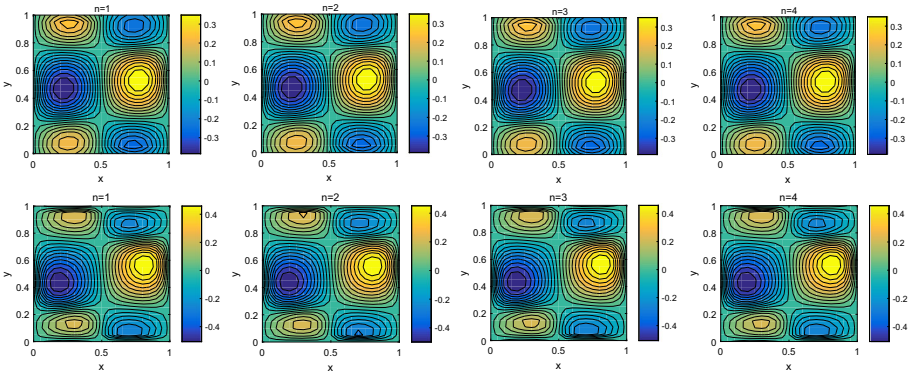

Fig. 11 The results of the data-driven MOR method based on $G_{layer} = 1$ (OOS1) and $G_{layer} = 2$ (OOS1) at moment $t = 0.3$. above: the results of OOS1 neural network; below: the results of OOS2 neural network

$F_{layer} = n + 2$ as the hidden neurons of $\widetilde{\mathbf{f}}$ network. At this time, the relevant parameters of the network are shown in Table 2.

In Table 2, we compared the computation time of single iteration step of the LM algorithm in different neural network parameter spaces. The computation time changes with size of the parameter spaces. And the number of network parameters with $G_{layer} = 2$ is 1.5 times that with $G_{layer} = 1$. The number of parameters of the network does not change much with the different dimensions of the reduced state variables, we found that the training error and test error are gradual minimum possible error (it is assumed that when $G_{layer} = 1$, the minimum possible errors of train set and test set are $5.48e-02$ and $5.79e-02$, respectively, when $G_{layer} = 2$, the minimum possible errors of train set and test set are $8.80e-03$ and $9.90e-03$, respectively).

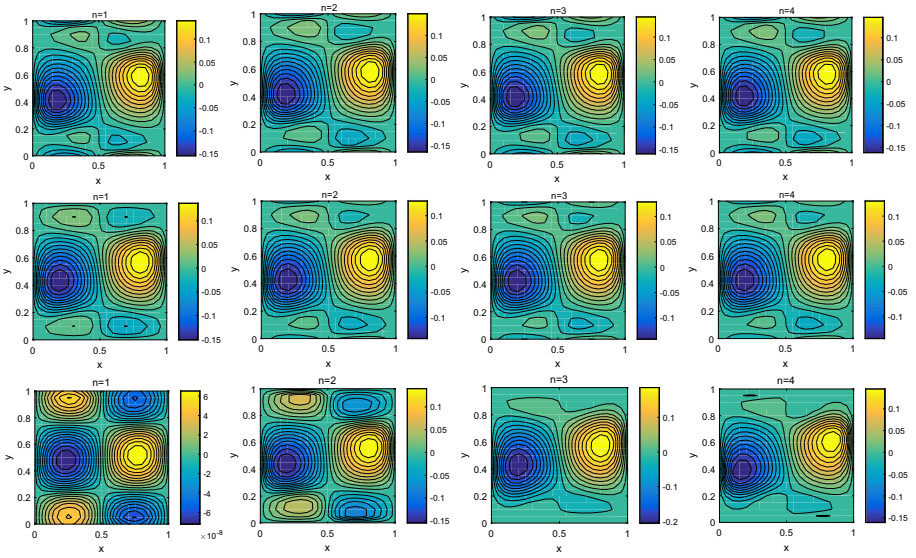

Fig. 12 Comparison of model-based MOR method and two data-driven MOR methods ($G_{layer} = 1$ and $G_{layer} = 2$) at time $t = 1$. above: data-driven MOR method with $G_{layer} = 1$; middle: data-driven MOR method with $G_{layer} = 2$; below: model-based MOR method

Then, in a random test set, we respectively tested the data-driven MOR effect of the network corresponding to $G_{layer} = 1$ and the network corresponding to $G_{layer} = 2$ at $t = 0.3$. As shown in Fig. 11, which can be compared with Fig. 10 (left), it can be found that the network result is closer to its HF numerical solution when $G_{layer} = 2$. In addition, we also compared the model-based MOR method in which nonlinear terms are linearized to the form (3) by explicit and implicit schemes and data-driven MOR methods at $t = 1$, as shown in Fig. 12. By comparing with Fig. 10 (right), it is found that the results of data-driven MOR method with the reduced state variables $n = 1$ and $n = 2$ are much better than that the model-based MOR method.

Finally, we compare the relative test errors and relative computation times of the model-based MOR method and data-driven MOR methods relative to the HF model in Fig. 13. In the left figure, it can be found that with the reduction of relative information ε_n, the relative test error of model-based MOR method decreases accordingly. The relative test error of data-driven MOR methods reaches the convergence error, which is lower than the model-based MOR method under the same dimension of the reduced state variables. In the middle figure, we compare the calculation time of the model-based MOR method and data-driven MOR methods relative to the HF model. The computation time of data-driven MOR methods are not much different, but it differs by three orders of magnitude compared to model-based MOR method, so the data-driven MOR method can save a lot of computation time. It can be found from the right figure that the data-driven MOR method corresponding to $G_{layer} = 2$ is the best choice for order reduction of this model, no matter in terms of test error or calculation time.

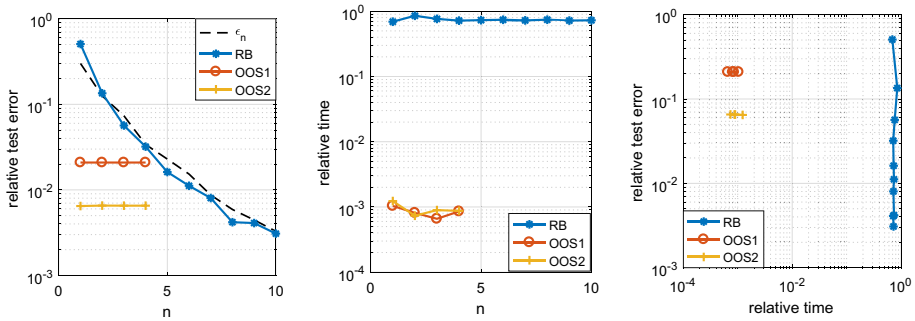

Fig. 13 Comparison of error and computation time between model-based MOR method and data-driven MOR method

4.3 Multi-Input Multi-Output System

The last example we consider a three-dimensional heat equation problem in the cell cube $\Omega = [0, 1]^3$, and this is a generalization of the problem considered in [35], whose boundary is divided into three parts, the top boundary $\Gamma_t : \psi(\mathbf{x}, 0) = 0$ is Dirichlet boundary, the bottom boundary Γ_b with a constant inward heat flow $\varphi = 50$, and with a flux-free boundary Γ_w. Then the spatial temperature distribution function $\psi(\mathbf{x}, t)$ can be described as the following heat equation form

$$\begin{cases} \frac{\partial}{\partial t}\psi(\mathbf{x}, t) - \operatorname{div}(k(\mathbf{x}, \mathbf{u}(t))\nabla\psi(\mathbf{x}, t)) = 0 & (\mathbf{x}, t) \in \Omega \times (0, T] \\ k(\mathbf{x}, \mathbf{u}(t))\nabla\psi(\mathbf{x}, t) \cdot \mathbf{n} = 0 & (\mathbf{x}, t) \in \Gamma_w \times (0, T] \\ k(\mathbf{x}, \mathbf{u}(t))\nabla\psi(\mathbf{x}, t) \cdot \mathbf{n} = \varphi & (\mathbf{x}, t) \in \Gamma_b \times (0, T] \\ \psi(\mathbf{x}, t) = 0 & (\mathbf{x}, t) \in \Gamma_t \times (0, T] \\ \psi(\mathbf{x}, 0) = 0 & \mathbf{x} \in \Omega. \end{cases} \tag{31}$$

We divide spatial domain Ω into eight equal subdomains Ω_i, for $i = 1, 2, \cdots, 8$, as in the left of Fig. 14. Consider the constant diffusion coefficient k, which depends on the 8-dimensional input $u(t) \in [10, 60]^8$, defined as follow

$$k(\mathbf{x}, \mathbf{u}(t)) = \sum_{i=1}^{8} u_i \amalg_{\Omega_i}(\mathbf{x})$$

where $\amalg_{\Omega_i}$ is an indicator function on domain Ω_i. The five detection points we considered are located in domain $\Omega_1, \Omega_3, \Omega_6, \Omega_8$ and their intersection respectively, so output $y(t) \in R^5$ is the temperature values of these probe points.

We used the traditional finite element method to obtain the snapshot matrix as shown in the right of Fig. 14, and we presented the slice figures of four random training sets as shown in Fig. 15.

In Fig. 16, we mainly present the composition of the training set, including the steady-state input and output with a response time of $T = 0.4$ and the unsteady-state input and output with a response time of $T = 1$. The training set mainly includes 20 steady-state training data and 180 unsteady-state training data.

For such a MIMO system, we train the reduced order model by using the OIS neural network and the OOS neural network respectively. For the OIS neural network, due to the dimensions of the reduced state variable is greater than or equal to the dimensions of output, i.e. $n \geq N_y$. Therefore, the dimensions of state variables are $n = 5, 6, 7$, respectively, the

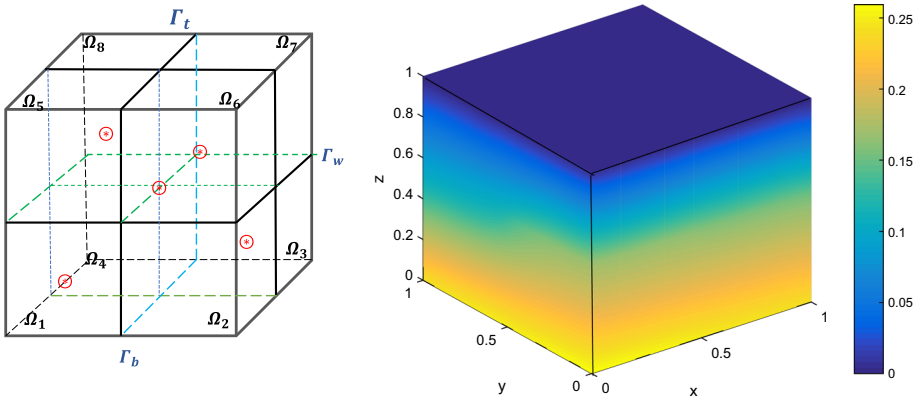

Fig. 14 Left: domain and boundary conditions; Right: a random snapshot

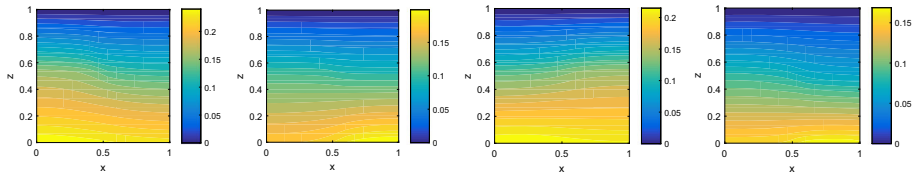

Fig. 15 Examples of snapshots y = 0.5 cross section obtained at different times with different inputs

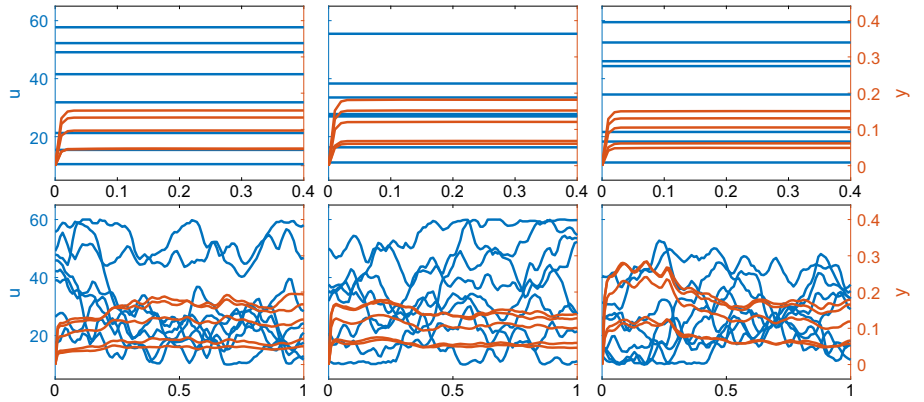

Fig. 16 A subset of the training set

OIS neural network on two random test sets of results as shown in Fig. 17 (above). And for the OOS neural network, the dimensions of the reduced state variables are $n = 1, 3, 5$, respectively, obtained the result of the data-driven MOR method as shown in Fig. 17 (middle). Finally, we applied the model-based MOR method, namely RB method, to obtain the low-order system with state variable $n = 1, 3, 5, 10$, respectively. The test results on the random test set are shown in Fig. 17 (below).

Finally, in Fig. 18 (left), we compared the errors of the data-driven MOR methods and model-based MOR method on the test set. With the increase of the reduced state variable dimension, the OOS neural network error decreasing, the OIS neural network error is less than

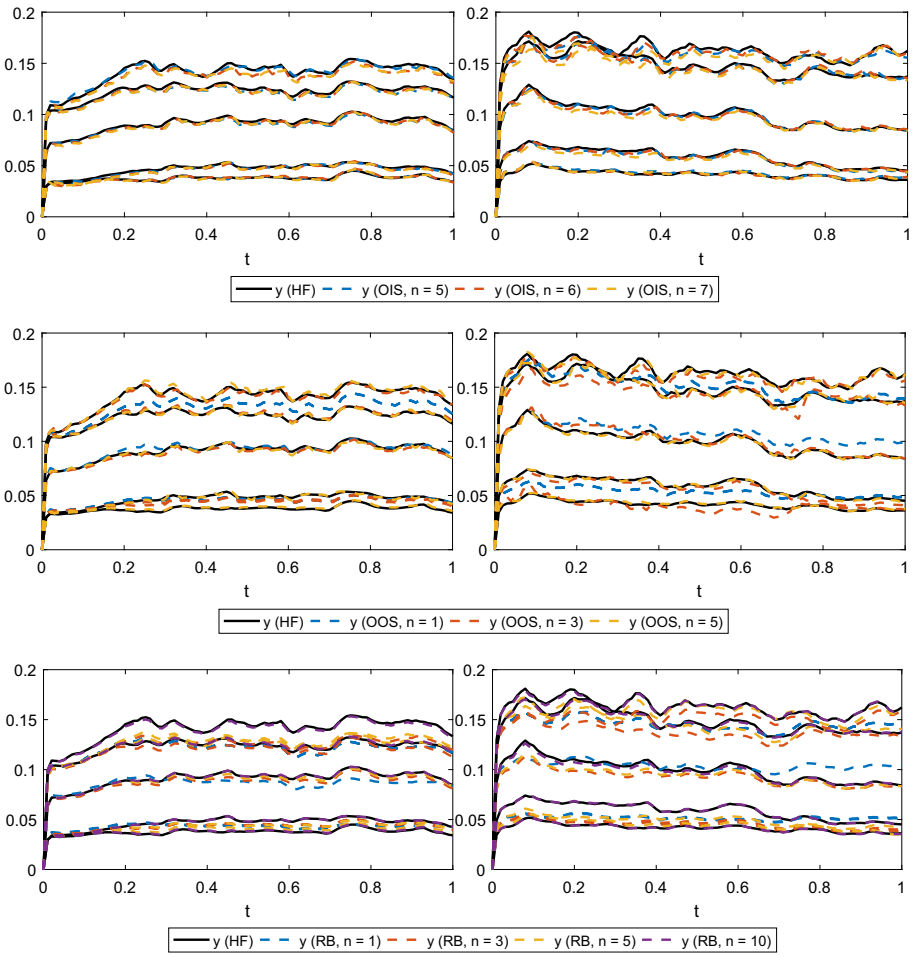

Fig. 17 Comparison of the results of the HF model with the different reduced models for two tests. Top: the OIS neural network MOR method vs HF model; middle: the OOS neural network MOR method vs HF model; bottom: model-based MOR method vs HF model

the error of the model-based MOR method, and the error of the model-based MOR method is also in decline, this can be guaranteed by the ε_n. In Fig. 18 (middle) shows that compared with HF model, the calculation time of model-based MOR method increases with the increase of the dimension of the reduced state variables. For the data-driven MOR methods, once the network structure is determined, its computation time tends to be stable relative to the HF model, and the computation speed of the OIS neural network is faster than that the OOS neural network. Considering the test error and relative calculation time as shown in Fig. 18 (right), it can be found that the OIS neural network is superior to the OOS neural network, and both of them are superior to the model-based MOR method.

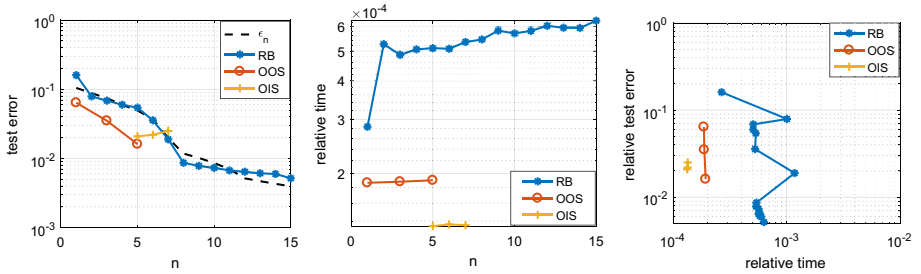

Fig. 18 Comparison of error and computation time between model-based MOR method and data-driven MOR method

5 Conclusions and Perspectives

By considering a data-driven MOR method for parameterized time-dependent PDEs, which approximates a time-evolving multidimensional physical system with a low order system. Using known input and output data to reconstruct the reduced state variable equation, with the reduced state variable equation as the constraint condition, minimizes the error function of the reduced model output and the HF output. The reduced model is represented by ANN and the MOR problem is transformed into a class of optimization problems with constraints. The optimization algorithm is employed to find the optimal ANN representation.

Through numerical examples, we have found that for parameterized time-dependent PDEs, it is efficient to use ANN to establish candidate model with system parameters as input. The choice of network capacity is very important for ANN training. Under-fitting is easy to occur in networks with low capacity, while over-fitting occurs in networks with high capacity. In order to control the model capacity, we generally have used the OOS neural network for multi-output system. For MIMO system, we have build ANN candidate model based on the known information of part of the system. It can be found that this non-invasive modeling method can effectively use the known information to build an efficient substitute model.

By comparing the performance of data-driven MOR method with that of traditional RB method, we find that the data-driven MOR method is superior to the RB model reduction method popular in the PDEs field in terms of online computing time and accuracy, although the cost of off-line training time is high. There still exist possible improvements which can be made in future, for example, the structure of deep neural network (DNN) can be considered in the parameterization of $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{g}}$ in the low-order approximation system. When the number of neural network parameters is too large, we can consider Batch Gradient Descent, Random Gradient Descent, L-BFGS, and Adam algorithms widely used in the field of machine learning. From the data-driven perspective, the POD prior dimension reduction technology, pre-training on small batch sets and updating parameters each time over several time steps can be considered to accelerate the off-line training stage. Low-order state variable equations with larger value of n can also be considered to explore the potential of the proposed method for low-order approximation of more complex dynamic systems.

Acknowledgements This work is supported by the Research Fund from Key Laboratory of Xinjiang Province (No. 2020D04002), the Natural Science foundation of China (No. U19A2079, 11671345, 91630205, 91852106, 92152301).

Funding The authors have not disclosed any funding.

Data Availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Competing Interests The authors have no conflicts to disclose.

References

1. Quarteroni, A., Manzoni, A., Negri, F.: Reduced basis methods for partial differential equations: an introduction. Springer, Berlin (2015)
2. Kunisch, K., Volkwein, S.: Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. Soc. Ind. Appl. Math. **40**(2), 492–515 (2002)
3. Abbasi, F., Velni, J.M.: Nonlinear model order reduction of Burgers' equation using proper orthogonal decomposition, American Control Conferen, pp. 583–588 (2015)
4. Berg, J., Nyström, K.: A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing **317**, 28–41 (2018)
5. Rudy, S.H., Brunton, S.L., Proctor, J.L., Kutz, J.N.: Data-driven discovery of partial differential equations. Sci. Adv. **3**(4), 1602614 (2017)
6. Raissi, M., Karniadakis, G.E.: Hidden physics models: Machine learning of nonlinear partial differential equations. J. Comput. Phys. **357**, 125–141 (2018)
7. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Machine learning of linear differential equations using Gaussian processes. J. Comput. Phys. **348**, 683–693 (2017)
8. Santo, N., Deparis, S., Pegolotti, L.: Data driven approximation of parametrized PDEs by reduced basis and neural networks. J. Comput. Phys. **416**, 109550 (2020)
9. Guo, M., Hesthaven, J.S.: Data-driven reduced order modeling for time-dependent problems. Comput. Methods Appl. Mech. Eng. **345**, 75–99 (2019)
10. Wu, P., Sun, J., Chang, X., Zhang, W., Arcucci, R., Guo, Y., Pain, C.C.: Data-driven reduced order model with temporal convolutional neural network. Comput. Methods Appl. Mech. Eng. **360**, 112766 (2020)
11. Audouze, C., Vuyst, F.D., Nair, P.B.: Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. Numer. Methods Partial Differ. Equ. **29**(5), 1587–1628 (2013)
12. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Multistep neural networks for data-driven discovery of nonlinear dynamical systems, (2018), arXiv:1801.01236v1
13. San, O., Maulik, R.: Machine learning closures for model order reduction of thermal fluids. Appl. Math. Model. **60**, 681–710 (2018)
14. Wang, Q., Hesthaven, J.S., Ray, D.: Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to acombustion problem. J. Comput. Phys. **384**, 289–307 (2019)
15. Hesthaven, J.S., Ubbiali, S.: Non-intrusive reduced order modeling of nonlinear problems using neural networks. Article in Journal of Computational. Phys. **363**, 55–78 (2018)
16. Frescaa, S., Dedè, L., Manzonia, A.: A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J. Sci. Comput. **87**, 61 (2021)
17. Frescaa, S., Manzonia, A.: POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition, (2021), arXiv:2101.11845v1
18. Fresca, S., Manzoni, A., Dedè, L., Quarteroni, A.: Deep learning-based reduced order models in cardiac electrophysiology. PLoS ONE **15**(10), 0239416 (2020)
19. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–70 (2019)
20. Chen, W., Wang, Q., Hesthaven, J.S., Zhang, C.: Physics-informed machine learning for reduced-order modeling of nonlinear problems. J. Comput. Phys. **446**, 110666 (2021)
21. Mücke, N.T., Bohté, S.M., Oosterlee, C.W.: Reduced order modeling for parameterized time-dependent PDEs using spatially and memory aware deep learning. J. Comput. Sci. **53**, 101408 (2021)
22. Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. Soc. Ind. Appl. Math. **57**(4), 483–531 (2015)
23. Peherstorfer, B., Gugercin, S., Willcox, K.: Data-driven reduced model construction with time-domain Loewner models. SIAM J. Sci. Comput. **39**(5), A2152–A2178 (2017)

24. Regazzoni, F., Dedè, L., Quarteroni, A.: Machine learning for fast and reliable solution of time-dependent differential equations. J. Comput. Phys. **397**, 108852 (2019)
25. Regazzoni, F., Dedè, L., Quarteroni, A.: Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics. Comput. Methods Appl. Mech. Eng. **370**, 113268 (2020)
26. Nelles, O.: Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and Gaussian processes. Springer, Berlin (2021)
27. Gosea, I.V.: Model order reduction of linear and nonlinear systems in the Loewner framework, (2018)http://urn-resolving.de/urn:nbn:de:gbv:579-opus-1007899
28. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**(15), 3932–3937 (2016)
29. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems **2**(4), 303–314 (1989)
30. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. Inf. Theory **39**(3), 930–945 (1993)
31. Hagan, M.T., Demuth, H.B.: Neural network design. China Machine Press, Beijing, China (2002)
32. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning, the MIT Press, (2016), http://www.deeplearningbook.org
33. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. Soc. Ind. Appl. Math. **60**(2), 223–311 (2018)
34. Zhang, J., Yan, G.: Lattice Boltzmann method for one and two-dimensional Burgers' equation. Phys. A **387**, 4771–4786 (2008)
35. Manzoni, A., Pagani, S., Lassila, T.: Accurate solution of Bayesian inverse uncertainty quantification problems combining reduced basis methods. SIAM/ASA J. Uncertain. Quant. **4**(1), 380–412 (2016)
36. Tiumentsev, Y.V., Egorchev, M.V.: Neural network modeling and identification of dynamical systems. Academic Press, London, UK (2019)