

Constructing reduced model for complex physical systems via interpolation and neural networks

Xuefang Lai(赖学方), Xiaolong Wang(王晓龙), and Yufeng Nie(聂玉峰)
Citation: Chin. Phys. B, 2021, 30 (**3**): 030204. DOI: 10.1088/1674-1056/abd92e
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Design of passive filters for time-delay neural networks with quantized output

Jing Han(韩静), Zhi Zhang(章枝), Xuefeng Zhang(张学锋), and Jianping Zhou(周建平)
Chin. Phys. B, 2020, 29 (**11**): 110201. DOI: 10.1088/1674-1056/aba602

Finite-time Mittag-Leffler synchronization of fractional-order delayed memristive neural networks with parameters uncertainty and discontinuous activation functions

Chong Chen(陈冲), Zhixia Ding(丁芝侠), Sai Li(李赛), Liheng Wang(王利恒)
Chin. Phys. B, 2020, 29 (**4**): 040202. DOI: 10.1088/1674-1056/ab7803

A novel stable value iteration-based approximate dynamic programming algorithm for discrete-time nonlinear systems

Yan-Hua Qu(曲延华), An-Na Wang(王安娜), Sheng Lin(林盛)
Chin. Phys. B, 2018, 27 (**1**): 010203. DOI: 10.1088/1674-1056/27/1/010203

Finite-time robust control of uncertain fractional-order Hopfield neural networks via sliding mode control

Yangui Xi(喜彦贵), Yongguang Yu(于永光), Shuo Zhang(张硕), Xudong Hai(海旭东)
Chin. Phys. B, 2018, 27 (**1**): 010202. DOI: 10.1088/1674-1056/27/1/010202

Augmented Lyapunov approach to H_∞ state estimation of static neural networks with discrete and distributed time-varying delays

M. Syed Ali, R. Saravanakumar
Chin. Phys. B, 2015, 24 (**5**): 050201. DOI: 10.1088/1674-1056/24/5/050201

RAPID COMMUNICATION

Constructing reduced model for complex physical systems via interpolation and neural networks*

Xuefang Lai(赖学方), Xiaolong Wang(王晓龙)†, and Yufeng Nie(聂玉峰)

Research Center for Computational Science, School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710129, *China*

(Received 12 November 2020; revised manuscript received 17 December 2020; accepted manuscript online 7 January 2021)

The work studies model reduction method for nonlinear systems based on proper orthogonal decomposition (POD) and discrete empirical interpolation method (DEIM). Instead of using the classical DEIM to directly approximate the nonlinear term of a system, our approach extracts the main part of the nonlinear term with a linear approximation before approximating the residual with the DEIM. We construct the linear term by Taylor series expansion and dynamic mode decomposition (DMD), respectively, so as to obtain a more accurate reconstruction of the nonlinear term. In addition, a novel error prediction model is devised for the POD-DEIM reduced systems by employing neural networks with the aid of error data. The error model is cheaply computable and can be adopted as a remedy model to enhance the reduction accuracy. Finally, numerical experiments are performed on two nonlinear problems to show the performance of the proposed method.

Keywords: model reduction, discrete empirical interpolation method, dynamic mode decomposition, neural networks

PACS: 02.30.Jr, 02.60.Cb, 02.60.Gf **DOI:** 10.1088/1674-1056/abd92e

1. Introduction

Many physical systems and processes in scientific and engineering fields are modeled mathematically by higher-order differential equations.[1,2] Usually, it is time-consuming or even computationally intractable to directly perform simulations on these models. The computational challenge has motivated the development of model reduction. Model reduction is a powerful technique that seeks to approximate the higher-order complex system with a low-order one, aiming at reducing the computational cost with a controlled loss of accuracy.

In this article, we concentrate on the nonlinear systems which are encountered in a wide range of physical phenomena.[3] For nonlinear systems, the existing approaches for model reduction cover quadratic expansion,[4] trajectory piecewise-linear (TPWL),[5,6] piecewise-polynomial representations,[7] proper orthogonal decomposition (POD),[8,9] and so on.[10] Among them, POD is the most popular method. Up to date, a variety of reduced-order models (ROMs) based on the POD method have been developed, including the intrusive ROMs and non-intrusive ROMs. Usually, the non-intrusive ROMs predict the projection coefficients of the POD basis by applying interpolation or machine learning methods, such as neural networks,[11] and Gaussian process regression.[12] In contrast, the intrusive ROMs are constructed by implementing the Galerkin projection,[13] and often are more effective to use the physical knowledge of the governing equations. Unfortunately, the POD-Galerkin method often suffers from instability and nonlinear inefficiency problems.

An efficient method for obviating this problem is sparse sampling. The pioneer works include gappy POD,[14] empirical interpolation method (EIM),[15] best points (missing points) estimation (MPE),[16] discrete empirical interpolation method (DEIM),[17] and so on. The gappy POD was firstly introduced in Ref. [14] for dealing with the "gappy" or incomplete data vector and later extended to the reconstruction of unsteady flow problems in Ref. [18]. EIM[15] and MPE[16] are another two methods developed for efficiently managing the computation of the nonlinearity. MPE is involved with a least-squares minimization problem, and its computational complexity is much higher than that of the EIM architecture to some extent. Especially, the DEIM,[17] as a discrete variant of EIM, shows superior performance in approximating the nonlinearity by using a small, discrete sampling of spatial points.

Although the interpolation points selected by the DEIM method provide a good approximation for the nonlinear function, some information is inevitably lost. To fill this gap, we propose an improved DEIM method by using the technique of linear truncation, denoted by LT-DEIM. More specifically, instead of applying the classical DEIM approach to the nonlinear function directly, our approach computes a linear approximation for the nonlinear function and evaluates the residual via the DEIM. Here, the purpose of the linear approximation term is to extract some information of the nonlinear function. By integrating the linear truncation method into the DEIM, a

*Project supported by the National Natural Science Foundation of China (Grant Nos. 11871400 and 11971386) and the Natural Science Foundation of Shaanxi Province, China (Grant No. 2017JM1019).

†Corresponding author. E-mail: xlwang@nwpu.edu.cn

© 2021 Chinese Physical Society and IOP Publishing Ltd

more accurate approximation of the nonlinear function can be achieved.

The linear approximation term plays a crucial role in the LT-DEIM model, and we construct it by using Taylor's expansion and dynamic mode decomposition (DMD),[19] respectively. Taylor's expansion is a powerful technique for the nonlinear model reduction.[4–6] For example, a reduced model of the Navier–Stokes equations was generated in Ref. [20] by combining the quadratic expansion with the DEIM. However, Taylor's expansion has some limitations when the nonlinear term is unknown or non-differentiable. Moreover, the nonlinear term is a high-dimensional nonlinear vector function, making it hard to compute the Jacobian matrix and select proper expansion points. Thus, the DMD method is also employed to construct the linear approximation term in this work. DMD is an equation-free and data-driven modal decomposition method that has successfully been used for nonlinear model reduction.[21,22] Moreover, DMD shows the marked improvement in the computational speed for evaluating the nonlinearity.

After the ROM has been constructed, we expect to know how accurate this ROM performs. Some assumptions and approximations were used to construct the ROM, which inevitably leads to an error between the ROM and the original model. These errors are crucial for the final resulting analysis. Hence, some pioneer work has been invested into quantifying the errors introduced by ROMs. Reference [23] provided error bounds for the state approximations from POD-DEIM reduced systems. Similarly, reference [24] presented a posteriori error estimation for POD-DEIM reduced systems. However, DEIM is a data-driven method, making it hard to efficiently estimate the error of DEIM-related ROMs through theoretical analysis. For this reason, by employing the radial basis function (RBF) neural networks,[25] this work presents a data-driven error model for the POD-DEIM reduced systems. Actually, some error models have been devised by applying machine learning methods such as Gaussian-process regression,[26,27] random forests,[28] neural networks, and so on.[29] These error models often utilize machine learning methods to build a mapping from some features (such as model parameters) to the error in a quantity of interest and have shown promising results. However, these error models often were in lack of stability and online computational efficiency.

To enhance the computational efficiency, our error model is characterized by the following two steps. Firstly, we utilize the DEIM method to extract the input variables of the error model. With DEIM algorithm, the dimension of input variables can be reduced significantly. Secondly, the RBF neural network uses the global and the local POD bases of the dynamical system as the center vectors. The center vectors exert an important impact on both the efficiency and accuracy of the RBF neural networks. Usually, it is difficult for users to make a reasonable choice of the center vectors from a large number of training data. The center vector selection via the POD technique was proposed in Ref. [30] and was extended to nonlinear aerodynamic ROM in Ref. [34]. However, they only used the local POD basis as the center vector, not making full use of the global information of snapshot data. In this work, both the global and the local POD bases are chosen to enhance the stability and the efficiency of the proposed error model. Finally, note that RBF neural networks have the capability of universal approximation and fast learning speed. With the neural networks, we can approximate the error of POD-DEIM reduced systems effectively.

Our presentation is organized as follows. In Section 2, we formulate the problem and provide a brief overview of the POD-DEIM reduced systems. In Section 3, we construct the LT-DEIM model by using Taylor series expansion and DMD, respectively. In Section 4, a data-driven error model is introduced for the POD-DEIM reduced systems based on the RBF neural networks. Section 5 tests the performance of the proposed method with two complex physical systems. Conclusions are offered in Section 6.

2. Problem formulation

Let us consider the following nonlinear input-output state-space system:

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{B}\boldsymbol{\mu}(t), \\ \boldsymbol{y}(t) = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{x}(t), \end{cases} \tag{1}$$

where $\boldsymbol{x}(t) \in R^{n\times 1}$ is the state variable with $\boldsymbol{x}(0) = \boldsymbol{x}_0$, number n is the order of the system. $\boldsymbol{f}(\boldsymbol{x}(t))$ is the nonlinear vector function, and $\boldsymbol{\mu}(t) \in R^p$ is the input of the system with $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$. Variable $\boldsymbol{y}(t) \in R^q$ is the system output, and $\boldsymbol{B} \in R^{n\times p}$ and $\boldsymbol{C} \in R^{n\times q}$ are the input and output matrices, respectively. In particular, these ordinary differential equations (ODEs) could arise from the discretization of partial differential equations (PDEs) based on numerical methods such as finite difference method (FDM) or finite element method (FEM).

2.1. Proper orthogonal decomposition based reduced model

We use the POD method to construct a reduced model for system (1). POD is a widely used data-driven modal decomposition method for obtaining low-complexity models.[8] Suppose we have collected a snapshot matrix $\boldsymbol{X}^{\mathrm{snap}} = [\boldsymbol{x}(t_1), \ldots, \boldsymbol{x}(t_N)] \in R^{n\times N}$, where $\boldsymbol{x}(t_i)$ represents the solution snapshot of system (1) at time t_i. Then the k-dimensional $(k \ll n)$ POD basis $\boldsymbol{V}$ for approximating state variable can be

determined by taking the singular value decomposition (SVD) of snapshot matrix $\boldsymbol{X}^{\text{snap}}$,

$$\boldsymbol{X}^{\text{snap}} \approx \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^{\text{T}}, \tag{2}$$

where $\boldsymbol{V} \in R^{n\times k}$ and $\boldsymbol{W}^{\text{T}} \in R^{k\times N}$ are real orthonormal matrices, and $\boldsymbol{\Sigma} \in R^{k\times k}$ is a diagonal matrix. Superscript symbol T represents the transpose of a matrix or a vector. With the constructed POD basis $\boldsymbol{V}$, the state variable $\boldsymbol{x}(t)$ of system (1) can be approximated by

$$\boldsymbol{x}(t) \approx \boldsymbol{V}\boldsymbol{a}(t), \tag{3}$$

where $\boldsymbol{a}(t) \in R^k$ is a time-dependent coefficient vector. Substituting Eq. (3) into system (1) and then applying the Galerkin projection yield

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{V}^{\text{T}}\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t)) + \hat{\boldsymbol{B}}\boldsymbol{\mu}(t), \\ \boldsymbol{y}(t) = \hat{\boldsymbol{C}}^{\text{T}}\boldsymbol{a}(t), \end{cases} \tag{4}$$

where $\hat{\boldsymbol{B}} = \boldsymbol{V}^{\text{T}}\boldsymbol{B} \in R^{k\times p}$, $\hat{\boldsymbol{C}}^{\text{T}} = \boldsymbol{C}^{\text{T}}\boldsymbol{V} \in R^{q\times k}$. Obviously, the dimension of system (4) is much smaller than that of system (1). We call system (4) the POD-Galerkin reduced system.

Unfortunately, the reduced system (4) usually is not able to evaluate $\boldsymbol{V}^{\text{T}}\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t))$ efficiently. DEIM is an effective method to overcome the issue.[17] The DEIM approximates the nonlinear term of the system through the technique of interpolation. With the POD method, the $\boldsymbol{f}(\boldsymbol{x}(t))$ in system (1) can be represented by

$$\boldsymbol{f}(\boldsymbol{x}(t)) \approx \boldsymbol{U}\boldsymbol{c}(t), \tag{5}$$

where $\boldsymbol{U} \in R^{n\times m}$ is the m-dimensional POD basis obtained by taking the SVD of the snapshot matrix of the nonlinear term, $\boldsymbol{c}(t) \in R^{m\times 1}$ is the time-dependent coefficient vector. Multiplying both sides of Eq. (5) by matrix $\boldsymbol{P}^{\text{T}}$ and then solving for the coefficients vector $\boldsymbol{c}(t)$, we obtain the DEIM approximation

$$\boldsymbol{f}(\boldsymbol{x}(t)) \approx \boldsymbol{U}(\boldsymbol{P}^{\text{T}}\boldsymbol{U})^{-1}\boldsymbol{P}^{\text{T}}\boldsymbol{f}(\boldsymbol{x}(t)), \tag{6}$$

where $\boldsymbol{P} = [\boldsymbol{e}_{r_1}, \ldots, \boldsymbol{e}_{r_m}] \in R^{n\times m}$, $\boldsymbol{e}_{r_i} \in R^n$ is the r_i-th column of identity matrix $\boldsymbol{I}_n \in R^{n\times n}$, and r_i is the interpolation index. In this expression, interpolation matrix $\boldsymbol{P}$ plays a crucial role for the approximation error and can be obtained by executing the greedy search algorithm provided in Ref. [17].

Substituting Eq. (6) into Eq. (4), we obtain

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{V}^{\text{T}}\boldsymbol{U}(\boldsymbol{P}^{\text{T}}\boldsymbol{U})^{-1}\boldsymbol{P}^{\text{T}}\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t)) + \hat{\boldsymbol{B}}\boldsymbol{\mu}(t), \\ \boldsymbol{y}(t) = \hat{\boldsymbol{C}}^{\text{T}}\boldsymbol{a}(t). \end{cases} \tag{7}$$

We call Eq. (7) the POD-DEIM reduced system. In (7), nonlinear term $\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t))$ is only evaluated at the elements corresponding to the interpolation indices, and thereby the efficiency can be greatly improved. After obtaining the POD basis $\boldsymbol{V}$, $\boldsymbol{U}$ and interpolation matrix $\boldsymbol{P}$, matrix $\boldsymbol{V}^{\text{T}}\boldsymbol{U}(\boldsymbol{P}^{\text{T}}\boldsymbol{U})^{-1}$ can be precomputed on the off-line stage. Here, matrix $\boldsymbol{P}^{\text{T}}\boldsymbol{U} \in R^{m\times m}$ is non-singular.

3. An improved interpolation method with linear approximation

As described above, for the DEIM method, only a few entries corresponding to the interpolation indices are selected to reconstruct the nonlinear function $\boldsymbol{f}(\boldsymbol{x}(t))$. This will inevitably lose the information contained in the unselected part. For this reason, an improved DEIM is presented by using linear truncation. We attempt to extract the information of the nonlinear function as much as possible before applying the DEIM approach, so as to reduce the error caused by interpolation.

We decompose the $\boldsymbol{f}(\boldsymbol{x}(t))$ of system (1) as

$$\boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t)) + \boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t)), \tag{8}$$

where $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))$ and $\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t))$ represent the linear approximation term and its residual term, respectively. Instead of applying the DEIM approach to the nonlinear term $\boldsymbol{f}(\boldsymbol{x}(t))$ directly, we only compute the DEIM approximation of the residual term $\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t))$,

$$\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t)) \approx \boldsymbol{\Phi}(\boldsymbol{S}^{\text{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\text{T}}\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t)), \tag{9}$$

where $\boldsymbol{\Phi} \in R^{n\times m}$ and $\boldsymbol{S} \in R^{n\times m}$ are the POD basis and the interpolation matrix that are calculated through the sampled data of the residual term $\{\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_1)), \ldots, \boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_N))\}$. Note that to calculate $\boldsymbol{\Phi}$ and $\boldsymbol{S}$, we need to first collect the snapshots of nonlinear function $\boldsymbol{f}(\boldsymbol{x}(t_i))$ and compute $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t_i))$, and then obtain the sampled data $\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_i)) = \boldsymbol{f}(\boldsymbol{x}(t_i)) - \boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t_i))$, $i = 1, 2, \ldots, N$.

With Eqs. (8) and (9), we have

$$\boldsymbol{f}(\boldsymbol{x}(t)) \approx \boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t)) + \boldsymbol{\Phi}(\boldsymbol{S}^{\text{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\text{T}}\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t)). \tag{10}$$

Through retaining the linear approximation of the original nonlinear function, model (10) has more advantage than the classical DEIM method in obtaining an accurate approximation of $\boldsymbol{f}(\boldsymbol{x}(t))$, which has been shown by the error analysis in Subsection 3.3. In addition, considering the efficiency of the reduced system, $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))$ should allow for a low-cost storage and a fast evaluation.

3.1. Linear approximation based on Taylor series expansion

Obviously, the linear approximation term $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))$ can be constructed by using Taylor series expansion. Based on Taylor series expansion, we have

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}(t)) &= \boldsymbol{f}(\boldsymbol{x}^*) + \boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{x}(t) - \boldsymbol{x}^*) + \boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t)) \\ &= \boldsymbol{f}_{\text{lin}}^{\text{Taylor}}(\boldsymbol{x}(t)) + \boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t)), \end{aligned} \tag{11}$$

where $\boldsymbol{J}(\boldsymbol{x}^*) \in R^{n\times n}$ denotes the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x}(t))$ at expansion point $\boldsymbol{x}^*$, and $\boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t))$ is the corresponding residual term. Here, we assume that $\boldsymbol{f}(\boldsymbol{x}(t))$ is a known and continuously differentiable nonlinear vector function. When $\boldsymbol{f}(\boldsymbol{x}(t))$ is unknown or non-differentiable, the Taylor series expansion fails and we will construct the linear approximation term with a data-driven method.

Taking the DEIM to approximate the residual term $\boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t))$ and substituting it into Eq. (11), we have

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{x}(t)) \approx\ & \boldsymbol{f}(\boldsymbol{x}^*)+\boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{x}(t)-\boldsymbol{x}^*)\\ &+\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t)),\end{aligned} \tag{12}$$

where $\boldsymbol{\Phi}$ and $\boldsymbol{S}$ are the POD basis and the interpolation matrix of the residual term $\boldsymbol{f}_{\text{res}}^{\text{Taylor}}(\boldsymbol{x}(t))$. In order to perform model reduction, taking the approximation $\boldsymbol{x}(t) \approx \boldsymbol{V}\boldsymbol{a}(t)$ and substituting Eq. (12) into Eq. (7), we obtain the corresponding reduced system

$$\begin{cases}\dfrac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{V}^{\mathrm{T}}[\boldsymbol{f}(\boldsymbol{x}^*)+\boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{V}\boldsymbol{a}(t)-\boldsymbol{x}^*)]\\ \qquad\qquad +\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\mathrm{T}}[\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t))-\boldsymbol{f}(\boldsymbol{x}^*)\\ \qquad\qquad -\boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{V}\boldsymbol{a}(t)-\boldsymbol{x}^*)]+\hat{\boldsymbol{B}}\mu(t),\\ \boldsymbol{y}(t) = \hat{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a}(t),\end{cases} \tag{13}$$

where all $\boldsymbol{V} \in R^{n\times k}$, $\boldsymbol{f}(\boldsymbol{x}^*) \in R^{n\times 1}$, $\boldsymbol{J}(\boldsymbol{x}^*) \in R^{n\times n}$, and $\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1} \in R^{k\times m}$ are constant matrices and can be pre-computed on the off-line stage. Hence, compared with the reduced system (7), system (13) also is efficient. For simplicity, we call model (12) Taylor-DEIM and system (13) POD-Taylor-DEIM reduced system.

Note that the performances of models (12) and (13) are associated with the expansion points $\boldsymbol{x}^*$. Considering that the error between $\boldsymbol{f}(\boldsymbol{x}(t))$ and $\boldsymbol{f}_{\text{lin}}^{\text{Taylor}}(\boldsymbol{x}(t))$ is smaller when $\boldsymbol{x}(t)$ is close to expansion points $\boldsymbol{x}^*$, we use the arithmetic mean of snapshots of state variable as expansion points, i.e., $\boldsymbol{x}^* = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}(t_i)$. Here, it is assumed that the expansion points are located in the trajectory of system. When this assumption is not valid for theory analysis, the state variable that is closest to the arithmetic mean can be used as the expansion points. In addition, compared with the first order Taylor series expansion, the quadratic expansion can provide a more accurate approximation of the nonlinear function. In Ref. [20], a hybrid model based on quadratic expansion and DEIM algorithm was presented to construct the ROM of Navier–Stokes equations. However, the memory and computational costs required by quadratic expansion are much larger, especially for higher-order complex systems (the cost for computing and storing Jacobian matrix and Hessian tensor are $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, respectively).

3.2. Linear approximation based on dynamic mode decomposition

As discussed above, we can construct the linear approximation term $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))$ with Taylor series expansion. However, this method has some limitations when the function $\boldsymbol{f}(\boldsymbol{x}(t))$ is unknown or non-differentiable. Moreover, $\boldsymbol{f}(\boldsymbol{x}(t))$ is a high-dimensional nonlinear vector function, making it hard to compute the Jacobian matrix $\boldsymbol{J}(\boldsymbol{x}^*)$ and select proper expansion points. Considering DEIM is a data-driven method, we construct another version of $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))$ by employing DMD.[19] DMD is a data-driven and equation-free modal decomposition method. This method has recently gained increasing attention owing to its ability to reveal and quantify the dynamics of systems. In particular, DMD shows the marked improvement in the computational speed for evaluating the nonlinearity.[21,22]

We assume that $\{\boldsymbol{f}(\boldsymbol{x}(t_1)),\ldots,\boldsymbol{f}(\boldsymbol{x}(t_N))\}$ is a data sequence of the nonlinear term, where each $\boldsymbol{f}(\boldsymbol{x}(t_i)) = \boldsymbol{f}(\boldsymbol{x}(i\Delta t))$, and Δt is a fixed sampling rate, N is the number of sampling. Because the DEIM method is itself a data-driven method, our method does not require additional sampling data. Let $\boldsymbol{F}_0 = [\boldsymbol{f}(\boldsymbol{x}(t_1)),\ldots,\boldsymbol{f}(\boldsymbol{x}(t_{N-1}))]$, $\boldsymbol{F}_1 = [\boldsymbol{f}(\boldsymbol{x}(t_2)),\ldots,\boldsymbol{f}(\boldsymbol{x}(t_N))]$, the linear operator can be approximated by

$$\boldsymbol{A}_F = \boldsymbol{F}_1\boldsymbol{F}_0^{\dagger}, \tag{14}$$

where $\boldsymbol{F}_0^{\dagger}$ is the pseudoinverse of $\boldsymbol{F}_0$ and we can obtain it by taking the reduced SVD of $\boldsymbol{F}_0$. In particular, for large-scale problems, we can replace the reduced SVD with randomized SVD method[31] to further reduce the computational cost. When the data is collected from a nonlinear dynamical system, $\boldsymbol{A}_F$ is considered to be a linear approximation of those dynamics. Once we have got $\boldsymbol{A}_F$, by taking an eigendecomposition of $\boldsymbol{A}_F$, we can obtain the DMD modes and the corresponding eigenvalues. Note that for some complex dynamical systems, the dimension n can be very large, it is extremely time and storage consuming to take an eigendecomposition of $\boldsymbol{A}_F$ and we usually projection it to a low-dimensional space.[19] In this work, our approach uses the uniformly sampled data and the standard DMD model for theoretical analysis and the numerical experiments in Section 5. For non-uniformly sampled data, we can employ the improved DMD model introduced by Ref. [32].

By means of the DMD procedure, the nonlinear function $\boldsymbol{f}(\boldsymbol{x}(t))$ is approximated by

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{x}(t)) \approx \boldsymbol{f}_{\text{lin}}^{\text{DMD}}(\boldsymbol{x}(t)) &= \sum_{j=1}^{l}\psi_j\alpha_j\lambda_j^{(t/\Delta t-1)}\\ &= \boldsymbol{\Psi}\boldsymbol{D}_{\alpha}\boldsymbol{V}_{\text{and}}^{(t/\Delta t-1)},\end{aligned}$$

$$\boldsymbol{\Psi} = [\psi_1,\psi_2,\ldots,\psi_l], \quad \boldsymbol{D}_{\alpha} = \begin{bmatrix}\alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_l\end{bmatrix},$$

$$\boldsymbol{V}_{\text{and}}^{(t/\Delta t-1)}=\begin{bmatrix}(\lambda_1)^{(t/\Delta t-1)}\\(\lambda_2)^{(t/\Delta t-1)}\\\vdots\\(\lambda_l)^{(t/\Delta t-1)}\end{bmatrix}, \tag{15}$$

where vector $\boldsymbol{\psi}_j \in C^n$ is the DMD mode, l is the number of DMD modes involved in the reconstruction. $\lambda_j \in C$ is the Ritz eigenvalue, which is associated with the growth rate and the frequency of the DMD mode. Coefficient $\alpha_j \in C$ denotes the amplitude of the jth mode, and the initial amplitude vector $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_l]^{\mathrm{T}}$ is given by $\boldsymbol{\alpha}=\boldsymbol{\Psi}^{\dagger}\boldsymbol{f}(\boldsymbol{x}(t_1))$ for the standard model, where $\boldsymbol{\Psi}^{\dagger}$ is the pseudoinverse of $\boldsymbol{\Psi}$. Note that the DMD approximation $\boldsymbol{f}_{\text{lin}}^{\text{DMD}}(\boldsymbol{x}(t))$ in Eq. (15) usually is a complex vector. To obtain a real approximation of the nonlinear function, we just use the real part of this DMD approximation in the actual calculations. This is a commonly used technique for obtaining a real approximation of the DMD model.[19]

With Eq. (15), we decompose $\boldsymbol{f}(\boldsymbol{x}(t))$ as

$$\boldsymbol{f}(\boldsymbol{x}(t))=\boldsymbol{f}_{\text{lin}}^{\text{DMD}}(\boldsymbol{x}(t))+\boldsymbol{f}_{\text{res}}^{\text{DMD}}(\boldsymbol{x}(t)), \tag{16}$$

where $\boldsymbol{f}_{\text{res}}^{\text{DMD}}(\boldsymbol{x}(t))$ is the corresponding residual term. Similarly, by applying DEIM method to $\boldsymbol{f}_{\text{res}}^{\text{DMD}}(\boldsymbol{x}(t))$ and taking the approximation $\boldsymbol{x}(t)\approx\boldsymbol{V}\boldsymbol{a}(t)$, and then substituting Eq. (16) into Eq. (7), we obtain the following reduced system:

$$\begin{cases}\dfrac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t}=\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Psi}\boldsymbol{D}_{\alpha}\boldsymbol{V}_{\text{and}}^{(t/\Delta t-1)}+\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}\\ \qquad\qquad[\boldsymbol{f}(\boldsymbol{V}\boldsymbol{a}(t))-\boldsymbol{\Psi}\boldsymbol{D}_{\alpha}\boldsymbol{V}_{\text{and}}^{(t/\Delta t-1)}]+\hat{\boldsymbol{B}}\mu(t),\\ \boldsymbol{y}(t)=\hat{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{a}(t),\end{cases} \tag{17}$$

where the POD basis $\boldsymbol{\Phi}$ and the interpolation matrix $\boldsymbol{S}$ are computed by using the sampling data of the residual term $\boldsymbol{f}_{\text{res}}^{\text{DMD}}(\boldsymbol{x}(t))$. We call model (16) DMD-DEIM and system (17) POD-DMD-DEIM reduced system.

For DMD method, how to extract the dominant DMD modes for characterizing the high-order dynamical systems still is an open problem and often needs priori knowledge on dynamical systems. To solve this problem, Lai and Wang *et al.*[33] presented an adaptive sparse-promoting DMD by combining tools from convex optimization with the compressive sensing. For simplicity, this work adopts a cross-validation strategy to determine the number of DMD modes in model (15). After constructing the DMD model, we use some new sampled data to test its performance. The optimal mode number is determined by minimizing the error between $\boldsymbol{f}_{\text{lin}}^{\text{DMD}}(\boldsymbol{x}(t))$ and $\boldsymbol{f}(\boldsymbol{x}(t))$.

3.3. Error analysis

Next, we present an error analysis to show that model (10) has more advantage than model (6) in approximating the vector function $\boldsymbol{f}(\boldsymbol{x}(t))$. For simplicity, we sign $\boldsymbol{f}(\boldsymbol{x}(t))=\boldsymbol{f}$, $\boldsymbol{f}_{\text{lin}}(\boldsymbol{x}(t))=\boldsymbol{f}_{\text{lin}}$, $\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t))=\boldsymbol{f}_{\text{res}}$, $\boldsymbol{U}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{-1}\boldsymbol{P}^{\mathrm{T}}=\mathbb{P}$, and $\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\mathrm{T}}=\mathbb{S}$. Let

$$\begin{aligned}\boldsymbol{E}_0(t)&=\boldsymbol{f}-\boldsymbol{U}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{-1}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{f}=\boldsymbol{f}-\mathbb{P}\boldsymbol{f},\\ \boldsymbol{E}_1(t)&=\boldsymbol{f}-(\boldsymbol{f}_{\text{lin}}+\boldsymbol{\Phi}(\boldsymbol{S}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{f}_{\text{res}})\\ &=\boldsymbol{f}-(\boldsymbol{f}_{\text{lin}}+\mathbb{S}\boldsymbol{f}_{\text{res}}).\end{aligned} \tag{18}$$

To begin, consider a set of snapshots of residual term $\{\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_1)),\ldots,\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_{ns}))\}\in R^n$. $\boldsymbol{\Phi}$ is the POD basis of the residual term $\boldsymbol{f}_{\text{res}}$, and we obtain $\boldsymbol{\Phi}$ by solving minimization problem[8]

$$\begin{cases}\min\limits_{\boldsymbol{\Phi}\in R^{n\times m}}\sum\limits_{i=1}^{ns}\|(\boldsymbol{I}_n-\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}(\boldsymbol{x}(t_i))\|_2,\\ \text{s.t. }\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}=\boldsymbol{I}_m.\end{cases}$$

where $\boldsymbol{I}_m\in R^{m\times m}$ and $\boldsymbol{I}_n\in R^{n\times n}$ are the identity matrices. Then, we have

$$\|(\boldsymbol{I}_n-\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}\|_2\le\|(\boldsymbol{I}_n-\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}\|_2. \tag{19}$$

Without loss of generality, assume that $\|(\boldsymbol{I}_n-\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}\|_2=h$, $\|(\boldsymbol{I}_n-\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}\|_2=h+\varepsilon$, where h,ε are non-negative constants, and $\varepsilon=0$ only for $\boldsymbol{f}_{\text{lin}}=0$.

Next, we will analyze the error bounds of $\boldsymbol{E}_0(t)$ and $\boldsymbol{E}_1(t)$. From Eq. (18), we have

$$\boldsymbol{E}_0(t)=(\boldsymbol{I}_n-\mathbb{P})(\boldsymbol{f}_{\text{lin}}+\boldsymbol{f}_{\text{res}}).$$

Then, we have

$$\|\boldsymbol{E}_0(t)\|_2\le\|(\boldsymbol{I}_n-\mathbb{P})\boldsymbol{f}_{\text{lin}}\|_2+\|(\boldsymbol{I}_n-\mathbb{P})\boldsymbol{f}_{\text{res}}\|_2. \tag{20}$$

According to Ref. [17], we have

$$\begin{aligned}\|(\boldsymbol{I}_n-\mathbb{P})\boldsymbol{f}_{\text{res}}\|_2&\le\|(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{U})^{-1}\|_2\|(\boldsymbol{I}_n-\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}})\boldsymbol{f}_{\text{res}}\|_2\\ &\le(1+\sqrt{2n})^{m-1}\|\boldsymbol{u}_1\|_{\infty}^{-1}(h+\varepsilon),\end{aligned} \tag{21}$$

where $\boldsymbol{u}_1$ is the first column of POD basis $\boldsymbol{U}$, $\|\cdot\|_{\infty}$ represents the infinite norm of a vector. With inequalities (20) and (21), the error upper bound of $\boldsymbol{E}_0(t)$ becomes

$$\begin{aligned}&\|\boldsymbol{E}_0(t)\|_2\\ &\le\|(\boldsymbol{I}_n-\mathbb{P})\boldsymbol{f}_{\text{lin}}\|_2+(1+\sqrt{2n})^{m-1}\|\boldsymbol{u}_1\|_{\infty}^{-1}(h+\varepsilon).\end{aligned} \tag{22}$$

Similarly, from Eq. (18), we also have

$$\|\boldsymbol{E}_1(t)\|_2=\|\boldsymbol{f}-(\boldsymbol{f}_{\text{lin}}+\mathbb{S}\boldsymbol{f}_{\text{res}})\|_2=\|(\boldsymbol{I}_n-\mathbb{S})\boldsymbol{f}_{\text{res}}\|_2.$$

Then, we have

$$\|(\boldsymbol{I}_n-\mathbb{S})\boldsymbol{f}_{\text{res}}\|_2\le(1+\sqrt{2n})^{m-1}\|\boldsymbol{\varphi}_1\|_{\infty}^{-1}h,$$

where $\boldsymbol{\varphi}_1$ is the first column of POD basis $\boldsymbol{\Phi}$. Then the error upper bound of $\boldsymbol{E}_1(t)$ can be obtained by

$$\|\boldsymbol{E}_1(t)\|_2\le(1+\sqrt{2n})^{m-1}\|\boldsymbol{\varphi}_1\|_{\infty}^{-1}h. \tag{23}$$

Referring to inequalities (22) and (23), we can notice that the error upper bounds of $\boldsymbol{E}_0(t)$ and $\boldsymbol{E}_1(t)$ are related to $\|\boldsymbol{u}_1\|_\infty$ and $\|\boldsymbol{\varphi}_1\|_\infty$. Because POD is a data-driven method, we are not optimistic that the rigorous estimations of $\|\boldsymbol{u}_1\|_\infty$ or $\|\boldsymbol{\varphi}_1\|_\infty$ can be obtained mathematically. However, by comparing inequalities (22) and (23), it is evident that $\boldsymbol{E}_1(t)$ has an advantage over $\boldsymbol{E}_0(t)$ in obtaining a smaller error bound.

4. Modeling the error of the reduced system with neural networks

During constructing ROMs, some assumptions and approximations were employed, which inevitably leads to an error between the ROM and the original system. The errors of ROMs are crucial for the final resulting analysis. However, as the error analysis discussed in Subsection 3.3, DEIM is a data-driven method, and it is hard to estimate the error of DEIM-related ROMs through theoretical analysis. To eliminate this drawback, we introduce a data-driven model to quantify the error of the POD-DEIM reduced system. The error model is constructed by using RBF neural networks[25] and can be computed cheaply. Once the error model has been built, we can correct the ROM with the prediction of the system error.

The state-space error of the reduced system can be expressed as

$$\Delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \tilde{\boldsymbol{x}}(t) \in R^n, \tag{24}$$

where $\boldsymbol{x}(t)$ and $\tilde{\boldsymbol{x}}(t)$ are state variables obtained by solving the original full system and the reduced system, respectively. We apply the RBF neural networks[25] to build a mapping from $\tilde{\boldsymbol{x}}$ to the error $\Delta\boldsymbol{x}(t)$

$$\Delta\boldsymbol{x}(t) \approx \boldsymbol{w}_0 + \sum_{j=1}^{d} \boldsymbol{w}_j \cdot \exp\left(- \frac{\|Q(\tilde{\boldsymbol{x}}(t)) - \boldsymbol{g}_j\|_2}{\sigma_j^2} \right), \tag{25}$$

where $Q(\tilde{\boldsymbol{x}}(t))$ and $\Delta\boldsymbol{x}(t)$ are the model input and output, and $Q : R^n \to R^{nq}$ is an operator that extracts the elements of the state variable for the model input. $\boldsymbol{w}_0 \in R^n$ and $\boldsymbol{w}_j \in R^n$ are the basis vector and weight vector, respectively. d is the number of hidden neurons, σ_j is the width value, and $\boldsymbol{g}_j$ is the center vector. With Eq. (25), the new approximation of state variable $\boldsymbol{x}(t)$ is given by $\boldsymbol{x}(t) \approx \tilde{\boldsymbol{x}}(t) + \Delta\boldsymbol{x}(t)$.

4.1. Optimization of the error model

Two techniques are applied to enhance the efficiency of the error model (25). Firstly, the DEIM method is applied to select the model input,

$$Q(\tilde{\boldsymbol{x}}(t)) = \boldsymbol{P}_1^{\mathrm{T}} \tilde{\boldsymbol{x}}(t),$$

where $\boldsymbol{P}_1 \in R^{n\times nq}$ is an interpolation matrix obtained by DEIM. Since $nq \ll n$, the dimension of input variables of the error model can be reduced significantly.

Secondly, we choose the global and the local POD bases of state variable as the center vectors $\boldsymbol{g}_j$ in Eq. (25). How to choose the center vectors is a crucial issue for RBF neural networks. Usually, it is difficult to make a reasonable choice of the center vectors from a large number of training samples. To overcome this issue, the POD technique was employed for the center vector selection in Ref. [30]. Specifically, the training samples were divided into many segments, and then the first-order POD basis of each segment was chosen as one center vector. Because the number of segments is smaller than the number of training samples, the POD technique maintains a balance between the accuracy and the complexity of the RBF neural networks. However, they only use the local POD basis as the center vectors, not making use of the global information of training samples. Thus, in this work, both the global and the local POD bases of state variable are employed as the center vectors of RBF neural networks.

Suppose we have N training samples, then model (25) can be written in matrix form as

$$\Delta\boldsymbol{X} = \boldsymbol{X} - \tilde{\boldsymbol{X}} = \boldsymbol{W}\boldsymbol{H}, \tag{26}$$

where

$$\begin{aligned}
&\boldsymbol{X} = [\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots, \boldsymbol{x}(t_N)] \in R^{n\times N},\\
&\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}(t_1), \tilde{\boldsymbol{x}}(t_2), \ldots, \tilde{\boldsymbol{x}}(t_N)] \in R^{n\times N},\\
&\Delta\boldsymbol{X} = [\Delta\boldsymbol{x}(t_1), \Delta\boldsymbol{x}(t_2), \ldots, \Delta\boldsymbol{x}(t_N)] \in R^{n\times N},\\
&\boldsymbol{W} = [\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_d] \in R^{n\times(d+1)},\\
&\boldsymbol{H} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ h(\tilde{\boldsymbol{x}}_1, \boldsymbol{g}_1) & h(\tilde{\boldsymbol{x}}_2, \boldsymbol{g}_1) & \cdots & h(\tilde{\boldsymbol{x}}_N, \boldsymbol{g}_1) \\ \cdots & \cdots & \cdots & \cdots \\ h(\tilde{\boldsymbol{x}}_1, \boldsymbol{g}_d) & h(\tilde{\boldsymbol{x}}_2, \boldsymbol{g}_d) & & h(\tilde{\boldsymbol{x}}_N, \boldsymbol{g}_d) \end{bmatrix} \in R^{(d+1)\times N},
\end{aligned} \tag{27}$$

$$h(\tilde{\boldsymbol{x}}_i, \boldsymbol{g}_j) = \exp\left(- \frac{\|Q(\tilde{\boldsymbol{x}}(t_i)) - \boldsymbol{g}_j\|_2}{\sigma_j^2} \right). \tag{28}$$

Note that equation (28) is a radial basis function, which is the activation function of the neural networks. Here, a combination of validation samples and heuristic algorithms can be used to determine the optimal structure and parameters of the neural networks, including the width value σ_j and the number of hidden neurons d.[25]

The procedure for calculating the weight matrix $\boldsymbol{W}$ is summarized as Algorithm 1. At the first step of this algorithm, we run the original system and the reduced system at the sampling interval to obtain training samples $\boldsymbol{X} = [\boldsymbol{x}(t_1), \ldots, \boldsymbol{x}(t_N)]$ and $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}(t_1), \ldots, \tilde{\boldsymbol{x}}(t_N)]$. After obtaining the training samples, we compute the weight matrix $\boldsymbol{W}$ and interpolation matrix $\boldsymbol{P}_1$ according to the proposed algorithm. Note that the processes above are executed on the off-line stage. Obviously, the computational cost of constructing the error remedy model is associated with the number of samples and is usually small. Once the error remedy model has been constructed, the error remedy model is executed on the on-line stage to predict of the state-space error of the reduced system. By using the DEIM method to select the input variable and the POD-based center vectors, the error remedy model is efficient on the on-line stage.

Calculation of weight matrix $\boldsymbol{W}$

1: Collect training samples $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ from the full system and the reduced system, respectively.
2: Compute the error matrix: $\Delta\boldsymbol{X} = \boldsymbol{X} - \tilde{\boldsymbol{X}}$.
3: Perform the DEIM method on the training samples $\boldsymbol{X}$: $\boldsymbol{X} \approx \boldsymbol{V}(\boldsymbol{P}_1\boldsymbol{V})^{-}\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{X}$, where $\boldsymbol{P}_1 \in R^{n\times nq}$; construct the input variable $Q(\tilde{\boldsymbol{x}}(t)) = \boldsymbol{P}_1^{\mathrm{T}}\tilde{\boldsymbol{x}}(t)$.
4: Compute the first N_{glo} POD basis of $\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{X}$ as the center vector.
5: Divide training samples $\boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{X}$ into N_{loc} segments; choose the first-order POD basis of each segment as the center vector. Here we have $d = N_{\mathrm{glo}} + N_{\mathrm{loc}}$.
6: Compute matrix $\boldsymbol{H}$ using Eqs. (27) and (28).
7: Compute weight matrix $\boldsymbol{W}$ using Eq. (26): $\boldsymbol{W} = (\Delta\boldsymbol{X})\boldsymbol{H}^{\dagger}$, where $\boldsymbol{H}^{\dagger}$ is the pseudoinverse of $\boldsymbol{H}$.

5. Results and discussion

Two examples are used to illustrate the effectiveness of the proposed model. The function ode15s in Matlab software is used to solve the full systems and the reduced systems. All the numerical simulations are performed on a desktop with Intel Core i7, 3.4 Ghz, 8 GB RAM, and software Matlab.

To check the accuracy of the model, the following relative errors are defined for nonlinear term $\boldsymbol{f}(\boldsymbol{x}(t))$ and system output $\boldsymbol{y}(t)$:

$$e_{\mathrm{fro}}(\boldsymbol{f}) = \left\|\boldsymbol{F} - \tilde{\boldsymbol{F}}\right\|_{\mathrm{fro}}^2 / \left\|\boldsymbol{F}\right\|_{\mathrm{fro}}^2,$$
$$e_1(\boldsymbol{y}) = \left\|\boldsymbol{y}(t) - \tilde{\boldsymbol{y}}(t)\right\|_1 / \left\|\boldsymbol{y}(t)\right\|_1, \tag{29}$$

where $\boldsymbol{F} = [\boldsymbol{f}(\boldsymbol{x}(t_1)), \ldots, \boldsymbol{f}(\boldsymbol{x}(t_{N_t}))] \in R^{n\times N_t}$ and $\boldsymbol{y}(t)$ are obtained from the full system, $\tilde{\boldsymbol{F}}$ and $\tilde{\boldsymbol{y}}(t)$ represent the approximations of $\boldsymbol{F}$ and $\boldsymbol{y}(t)$, respectively. $\|\cdot\|_{\mathrm{fro}}$ is the Frobenius norm (F-norm), and $\|\cdot\|_1$ is the L1-norm of a vector.

5.1. Example 1: nonlinear transmission line circuit

We consider the nonlinear transmission line circuit system introduced in Refs. [4,5]. The circuit includes resistors, capacitors, and diodes. The resistors and capacitors have unit resistance and capacitance. The diode I–V characteristic is $i_{\mathrm{c}}(v) = \exp(40v) - 1$. Let $\boldsymbol{x}(t) = [x_1(t), \ldots, x_n(t)]^{\mathrm{T}}$, and x_i describes the voltage at node i at time t. The system input and output are the current source and the voltage at node 1, respectively, i.e., the input and output matrices are $\boldsymbol{B} = \boldsymbol{C} = [1, 0, \ldots, 0]^{\mathrm{T}} \in R^{n\times 1}$. The system input $\mu(t) = \exp(-t)$, $t \in [0, T]$.

The number of nodes of the original system is $n = 100$. The total number of time steps is $N_t = 10000$ ($T = 10$, $\Delta t = 0.001$). In this example, 600 snapshots are uniformly selected from the original system in the interval $[0, 6]$ to construct the reduced systems. For the DMD-DEIM model, the number of DMD modes is fixed to 10. Figure 1 displays the relative error for approximating the nonlinear term of the system versus the dimension of the POD basis. The relative errors of Taylor-DEIM and DMD-DEIM are obviously smaller than those of DEIM method, suggesting that the linear approximation method in Section 3 can improve the accuracy of approximating the nonlinear term. Moreover, DMD-DEIM almost has the same accuracy as Taylor-DEIM when the dimension of POD basis $m \leq 5$. However, the DMD-DEIM gradually approaches the DEIM as the dimension of POD basis increases. The reason is that the performance of DMD-DEIM is influenced by the linear approximation term $\boldsymbol{f}_{\mathrm{lin}}(\boldsymbol{x}(t))$ and the residual term $\boldsymbol{f}_{\mathrm{res}}(\boldsymbol{x}(t))$, and the POD basis and interpolation matrix of the residual term are different from those of the nonlinear term. For comparison, figure 1 also gives the results of Taylor-POD and DMD-POD. Here, Taylor-POD and DMD-POD represent the models that we replace DEIM with the POD method to approximate the residual term. Because the DEIM method almost has the same performance as the POD method in approximating the residual term, the relative errors of Taylor-DEIM and DMD-DEIM are close to those of Taylor-POD and DMD-POD, respectively.

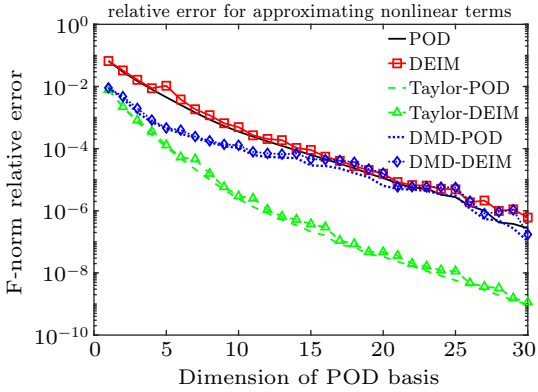


Fig. 1. Relative error for approximating nonlinear term in Example 1.

Figure 2 presents the results of various reduced systems of order $k = 5$, where the system outputs are shown in Fig. 2(a) and the corresponding L1-norm relative errors computed by Eq. (29) are given in Fig. 2(b). Without loss of generality, for all the reduced systems, we set the dimension of POD basis for state variable (k) and nonlinear term (m) the same, i.e., $k = m$. In Fig. 2(a), all the outputs of the reduced systems match the output of the full system well. However, it is clear that the relative errors of POD-Taylor-DEIM and POD-DMD-DEIM are much smaller than that of POD-DEIM in Fig. 2(b).

Figure 3 compares the relative errors of the outputs between POD-DEIM reduced system and POD-DEIM-RBF reduced system for order $k = 5$. Here, POD-DEIM-RBF represents the POD-DEIM reduced system with the error prediction model constructed by RBF neural networks. For error prediction model, the center vectors consist of 10 global POD bases and 20 local POD bases, the dimension of model input is 30, and width value $\sigma_j = 1$. The fact that POD-DEIM-RBF performs better than POD-DEIM confirms the effectiveness of the error prediction model introduced in Section 4.

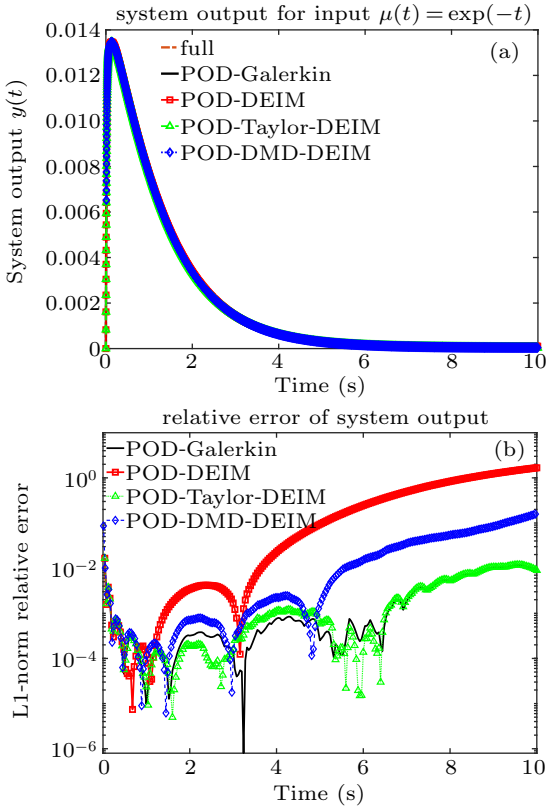


Fig. 2. Example 1: (a) System outputs of various reduced systems of order $k=5$. (b) Relative error of the system output.

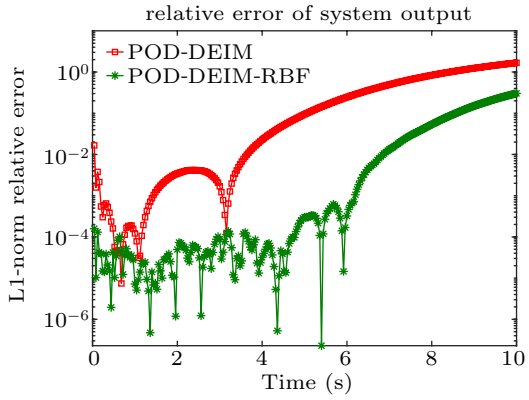


Fig. 3. The relative error of the system output comparison between POD-DEIM and POD-DEIM-RBF for Example 1, the order of reduced system $k=5$.

5.2. Example 2: Burgers' equation

Next, we consider one-dimensional Burgers' equation with the following initial-boundary conditions:[6]

$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t}+\dfrac{\partial f(U(x,t))}{\partial x}=g(x), \\ U(x,0)=1, \quad U(0,t)=\mu(t), \\ x\in[0,L], \quad t\in(0,T], \end{cases} \tag{30}$$

where $\mu(t)=\sqrt{5}$, $f(U)=0.5U^2$, $g(x)=0.02\exp(0.02x)$. Discretizing U with respect to x and using U_i to approximate U at points x_i yield

$$\frac{\mathrm{d}\bar{U}}{\mathrm{d}t}=\boldsymbol{F}(\bar{U})+\boldsymbol{G}+\boldsymbol{B}\mu^2, \tag{31}$$

where

$$\bar{U}=[U_1,\ldots,U_n]^{\mathrm{T}}, \quad \boldsymbol{B}=\left[\frac{1}{2\Delta x},0,\ldots,0\right]^{\mathrm{T}},$$

$$\boldsymbol{F}(\bar{U})=\frac{1}{\Delta x}[-0.5U_1^2,0.5(U_1^2-U_2^2),\ldots,0.5(U_{n-1}^2-U_n^2)]^{\mathrm{T}},$$

$$\boldsymbol{G}=0.02[\exp(0.02x_1),\ldots,\exp(0.02x_n)]^{\mathrm{T}}.$$

The dimension of the full model $n=1000$, $L=100$, $x_i=i\Delta x$, $\Delta x=L/n$, $T=60$, $\Delta t=0.01$. For this test case, the POD basis is constructed from 600 snapshots obtained from the solutions of the full system at equally spaced time steps in the interval $[0,T]$.

Figure 4 compares the abilities of different methods to approximate the nonlinear term of the system. It is apparent that Taylor-DEIM and DMD-DEIM are more accurate than DEIM. Similarly, Taylor-POD and DMD-POD are more accurate than POD. For DMD-POD and DMD-DEIM, the number of DMD modes is fixed to 100. Figure 5 shows the reconstructions of state variables when the orders of the reduced systems are $k=50$. Generally, the results in Fig. 5 are consistent with those in Fig. 4. As shown in Fig. 5, the curves of POD-Galerkin, POD-Taylor-DEIM, and POD-DMD-DEIM match the curves of the full system better than those of POD-DEIM.

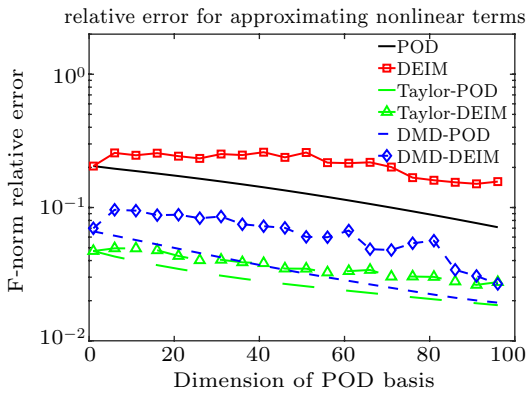


Fig. 4. Relative error for approximating nonlinear term in Example 2.

Figure 6 reports the results of POD-DEIM-RBF reduced system. For POD-DEIM-RBF, the center vectors of the error prediction model consist of 50 global POD bases and 10 local POD bases, the dimension of model input is 50, and width value $\sigma_j=10$. As shown in Fig. 6(a), the curves of POD-DEIM-RBF match the curves of the full system better than those of POD-DEIM. The corresponding relative error of the system output is shown in Fig. 6(b), where the relative error of POD-DEIM-RBF is smaller than that of POD-DEIM. Here, we assume that the system output $\boldsymbol{y}(t)=\boldsymbol{C}^{\mathrm{T}}\bar{U}(x,t)$, where $\boldsymbol{C}=[1,\ldots,1]^{\mathrm{T}}$. Moreover, it is worth mentioning that the error prediction model can be extended to other low-fidelity reduced systems. For example, figure 6(b) also provides the simulation results of POD-Galerkin-RBF reduced system (using POD-Galerkin to replace the POD-DEIM in POD-DEIM-RBF). Similarly, POD-Galerkin-RBF has superior performance to POD-Galerkin.

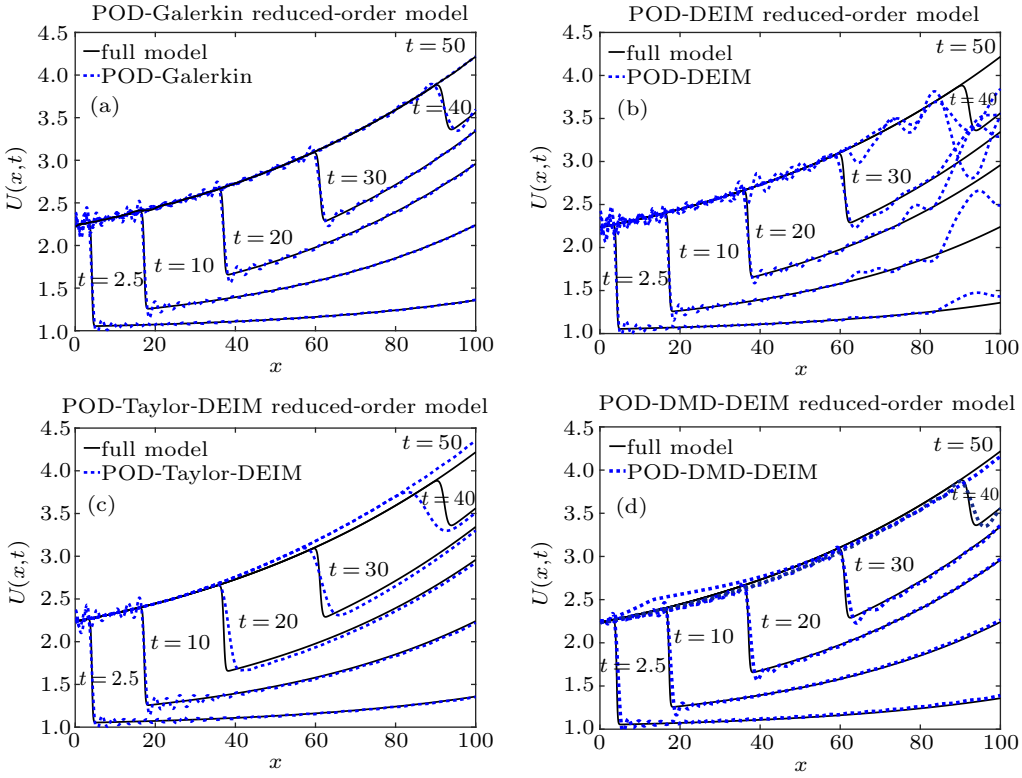


Fig. 5. Example 2: reduced systems of order $k = 50$. (a) POD-Galerkin. (b) POD-DEIM. (c) POD-Taylor-DEIM. (d) POD-DMD-DEIM.

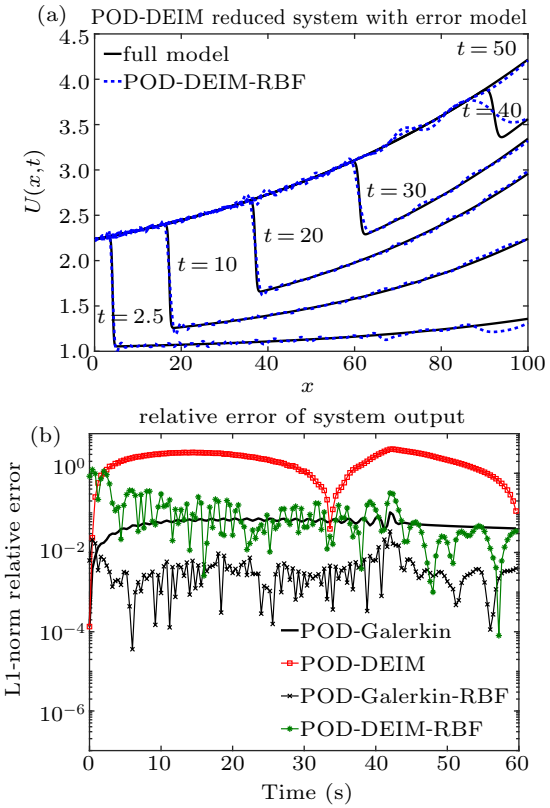


Fig. 6. Example 2: reduced systems of order $k = 50$. (a) The POD-DEIM reduced system with error prediction model. (b) Relative error of system output.

Table 1 lists the average CPU time for constructing (off-line) and solving (on-line) the reduced systems. The POD-Taylor-DEIM almost has the same off-line and on-line CPU time as the POD-DEIM. However, compared with POD-DEIM, POD-DMD-DEIM takes more CPU time both on the off-line and the on-line stages. This is because the POD-DMD-DEIM needs to compute the DMD approximation of the nonlinear term. In addition, the POD-DEIM-RBF costs more CPU time than the POD-DEIM both for the off-line stage and the on-line stage. The reason is that the POD-DEIM-RBF needs to construct the error remedy model on the off-line stage. For the on-line stage, POD-DEIM-RBF executes the error remedy model to predict the state-space error of POD-DEIM reduced system. However, in terms of accuracy, POD-DMD-DEIM and POD-DEIM-RBF are competitive with POD-DEIM in this example.

Table 1. The average off-line and on-line CPU time (second) of various reduced systems.

Model	Off-line	On-line	Total
Full model	0	1.452	1.452
POD-Galerkin	0.037	0.372	0.409
POD-DEIM	0.051	0.121	0.172
POD-Taylor-DEIM	0.066	0.123	0.189
POD-DMD-DEIM	0.114	0.207	0.321
POD-DEIM-RBF	0.178	0.264	0.442
POD-Galerkin-RBF	0.157	0.512	0.669

6. Conclusions

This work introduced an improved DEIM method for nonlinear model reduction. The improved DEIM first constructed a linear approximation for the nonlinear term of the

system, and then evaluated the residual between the linear approximation and the nonlinear term by using the standard DEIM. The line approximation term was constructed by the linear Taylor's expansion and the data-driven method, respectively. The error analysis was also provided to show the effectiveness of the proposed method.

Moreover, an error prediction model was introduced for the low-fidelity reduced models based on the RBF neural networks. To enhance the computational efficiency, the DEIM was applied to extract the input of the neural networks, and the global and the local POD bases of state variables were chosen as the center vectors of the neural networks. At last, our proposed method was tested on nonlinear transmission line circuit system and one-dimensional Burgers' equation. The example results showed that our approach is efficient and can significantly improve the accuracy of the low-fidelity reduced system.

References

[1] Wang X L and Jiang Y L 2013 *Math. Comp. Model. Dyn.* **19** 575

[2] Liu X Z, Yu J, Lou Z M and Qian X M 2019 *Chin. Phys. B* **28** 010201

[3] Qu Y H, Wang A N and Lin S 2018 *Chin. Phys. B* **27** 010203

[4] Chen Y and White J 2000 *Proc. Int. Conf. Modeling and Simulation of Microsystems* **2000** 477

[5] Rewienski M and White J 2001 *IEEE. T. Comput. Aid. D* **22** 155

[6] Rewieński M 2003 *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems* (Ph.D. Dissertation) (Boston: Massachusetts Institute of Technology)

[7] Ning D and Roychowdhury J 2008 *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.* **27** 249

[8] Holmes P, Lumley J L, Berkooz G and Rowley C W 2012 *Turbulence, coherent structures, dynamical systems and symmetry, 2nd edition* (England: Cambridge University Press) pp. 68–105

[9] Li S, Jiang N, Yang S Q, Huang Y X and Wu Y H 2018 *Chin. Phys. B* **27** 104701

[10] Xu W, Li S and Li R H 2007 *Chin. Phys. B* **16** 1591

[11] Hesthaven J S and Ubbiali S 2018 *J. Comput. Phys.* **363** 55

[12] Guo M W and Hesthaven J S 2019 *Comput. Method Appl. M.* **345** 75

[13] Kunisch K and Volkwein S 2002 *SIAM J. Numer. Anal.* **40** 492

[14] Everson R and Sirovich L 1995 *J. Opt. Soc. Am. A* **12** 1657

[15] Barrault M, Maday Y and Nguyen N C 2004 *C. R. Math.* **339** 667

[16] Nguyen N C, Patera A T and Peraire J 2008 *Int. J. Numer. Meth. Eng.* **73** 521

[17] Chaturantabut S and Sorensen D C 2011 *SIAM J. Sci. Comput.* **32** 2737

[18] Willcox K 2006 *Comput. Fluids* **35** 208

[19] Kutz J N, Brunton S L, Brunton B W and Proctor J L 2016 *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (Philadelphia: SIAM)

[20] Xiao D, Fang F, Buchan A G and Ionel M N 2014 *J. Comput. Phys.* **8** 1

[21] Bistrian D A and Navon I M 2017 *Int. J. Numer. Meth. Eng.* **112** 3

[22] Alla A and Kutz J N 2017 *SIAM J. Sci. Comput.* **39** *B*778

[23] Chaturantabut S and Sorensen D C 2012 *SIAM J. Numer. Anal.* **50** 46

[24] Wirtz D, Sorensen D C and Haasdonk B 2014 *SIAM J. Sci. Comput.* **36** *A*311

[25] Zhao Z Q and Huang D S 2007 *Appl. Math. Model.* **31** 1271

[26] Drohmann M and Carlberg K 2015 *SIAM/ASA J. Uncertain.* **3** 116

[27] Xiao D 2019 *Comput. Method. Appl. M.* **355** 513

[28] Lipponen A, Kolehmainen V, Romakkaniemi S and Kokkola H 2013 *Geosci. Model Dev.* **6** 2087

[29] Freno B A and Carlberg K T 2019 *Comput. Method. Appl. M.* **348** 250

[30] Zhang W B, Guo X C, Wang C Y and Wu C G 2007 *Adaptive and Natural Computing Algorithms, Lecture Notes in Computer Science* (Berlin: Springer) pp. 189–198

[31] Liberty E, Woolfe F, Martinsson P G, Rokhlin V and Tygert M 2007 *Proc. Natl. Acad. Sci. USA* **51** 104

[32] Guéniat F, Mathelin L and Pastur L R 2015 *Phys. Fluids* **27** 025113

[33] Lai X F, Wang X L, Nie Y F and Li Q 2020 *Int. J. Numer. Meth. Fluids* **92** 587

[34] Zhang W W, Kou J Q and Wang Z Y 2016 *AIAA J.* **54** 3302