Journal of Computational Physics 446 (2021) 110660

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Deep learning nonlinear multiscale dynamic problems using Koopman operator

Mengnan Li[a], Lijian Jiang[b,*]

[a] School of Mathematics, Hunan University, Changsha 410082, China
[b] School of Mathematical Sciences, Tongji University, Shanghai 200092, China

ARTICLE INFO

Article history:
Available online 24 August 2021

Keywords:
Nonlinear multiscale dynamic systems
Deep learning
Koopman operator
Multiscale basis functions

ABSTRACT

In this paper, a deep learning method using Koopman operator is presented for modeling nonlinear multiscale dynamical problems. Koopman operator is able to transform a nonlinear dynamical system into a linear system in a Koopman invariant subspace. However, it is usually very challenging to choose a set of suitable observation functions spanning the Koopman invariant subspace when only data is available for the model. It is practically important for us to predict the evolution of the state of the dynamical system from the Koopman invariant subspace. To this end, we introduce a reconstruction operator that maps the observation function space to the model's state space. Incorporating measurement data, a set of neural networks are constructed to learn the Koopman invariant subspace and the reconstruction operator. The loss function not only considers the properties of Koopman invariant subspace, but also reflects the prediction of future state, which makes the proposed method can realize the prediction of future state for a relatively long time. It may be experimentally expensive to collect the fine-scale data. It will be challenging to use limited computational resources to generate sufficient fine-scale data for neural network training. To overcome this difficulty, we use the data in a coarse-scale and learn effective coarse models for the nonlinear multiscale dynamical problems. In order to make the learned coarse model effectively capture fine-scale information, the loss functions for the neural networks are constructed using a set of multiscale basis functions, which are assumed to be given as a prior. In this case, an accurate fine-scale model can be derived by downscaling the learned coarse model. The deep learning multiscale models using Koopman operator can achieve a relatively long-time prediction for the evolution of the state of the nonlinear multiscale dynamical problems. A few numerical examples are presented to show that the effectiveness of learning multiscale models and the long-time prediction. The numerical results also demonstrate the advantage of the proposed learning method over some other similar learning methods.

1. Introduction

Many scientific and engineering problems involve multiple scales and strong nonlinearity. For example, the properties of composite materials, the flow of fluid in porous media and so on. Due to the interaction between multiple scales and nonlinearity, it may be very challenging to use classical finite element and finite difference methods to solve the nonlinear

* Corresponding author.
E-mail addresses: mengnanli@hnu.edu.cn (M. Li), ljjiang@tongji.edu.cn (L. Jiang).

https://doi.org/10.1016/j.jcp.2021.110660
0021-9991/© 2021 Elsevier Inc. All rights reserved.

multiscale problems. Resolving all scales may be infeasible for practical multiscale dynamical systems. Effective macroscopic property is critical for multiscale problems. Many multiscale methods can effectively capture coarse-scale and some fine-scale effect on a coarse grid, but do not need to be solved on fine grids. These multiscale methods include homogenization approach [3,10], multiscale finite element method [14,15], heterogeneous multiscale method [2,11], multiscale finite volume method [17,25], generalized multiscale finite element method [5,12], constraint energy minimizing generalized multiscale finite element method (CEM-GMsFEM) [6], non-local multicontinuum approach [7,8] and so on.

Although these multiscale reduction methods represent the fine-scale solution with coarse degrees of freedom and significantly reduce the computational cost in linear problems, the evaluation of nonlinear term in nonlinear problems still substantially depends on the high-dimensional fine-scale space. In order to further reduce the nonlinear models, discrete empirical interpolation method (DEIM) [4,9] is widely used in the nonlinear problems. DEIM combines projection with interpolation to estimate the nonlinear terms. It needs to only update the value of the interpolation points during online calculation. The number of interpolation points is much less than the degrees of freedom in the fine-scale space. Although DEIM can significantly reduce nonlinear models, there are still a couple of challenges. Firstly for nonlinear problems, the reduced-order model constructed by DEIM is still nonlinear and needs to recall nonlinear solvers (e.g., Piccard iteration and Newton method). This may take a large amount of computing resources. Secondly, DEIM not only uses snapshot data to find interpolation points, but also needs to know a specific form of the nonlinear model during online computation.

The above-mentioned model reduction methods for multiscale models are based on the fact that the model is known and described by a partial differential equation, but these methods are not applicable when the model is unknown and some data are available. Combining CEM-GMsFEM and dynamic mode decomposition (DMD) [30], CEM-DMD [16] method is proposed. It is a method driven by both model and data, which aims at the problem that the linear term of the model is known and the nonlinear term is unknown, but there are some observation data of the nonlinear term. It takes advantage of multiscale model reduction and data-driven method to construct a linear surrogate model for these nonlinear multiscale problems. According to the prior knowledge of the model and the characteristics of multiscale basis functions, the CEM-DMD model can be constructed by using fine-scale observation or coarse-scale observation. In this paper, we focus on a completely data-driven method, which only uses the data at some time instants to predict the future state when the model is unknown.

For multiscale models, it may be hard to get all fine-scale data and only the coarse-scale data is available at many occasions. In the work, the measurement data is related to the coarse-scale moment information or the coarse-scale degrees of freedom of a multiscale finite element method. Then a fine-scale model can be derived by downscaling the learned coarse model.

As data and scientific computation are playing more and more important roles in sciences and engineering, Koopman operator [18,22,27] has received wide attention as an effective data-driven tool. It can lift a finite-dimensional nonlinear dynamics to an infinite-dimensional linear dynamics by acting on an infinite-dimensional function space. In order to be numerically solvable, the Koopman operator can be restricted on a finite dimensional subspace that spanned by a set of observed functions. Then we can obtain a finite dimensional linear system in a Koopman invariant subspace. However, it is very challenging to find these functions that make up the Koopman invariant subspace even knowing the active terms of nonlinear dynamical systems. Dynamic mode decomposition (DMD) [30] method originates from the field of fluid dynamics, which is used to construct the best fitting linear dynamical system in the sense of least squares for a data sequence. Rowley et al. [29] proved that under certain conditions, DMD method gives the estimation of eigenvalues and modes of Koopman operator. The future state in the observation space can be given by the eigenvalues and modes of the Koopman operator. Therefore, selecting the observation functions from invariant subspace is critical for building effective DMD models. In the standard DMD algorithm, the identity function is used as the observation function. The standard DMD is equivalent to constructing the best-fitting linear system for a nonlinear dynamical system, and does not capture the effect of Koopman operator very well. It should be noted that in general, the identity function can not span an effective Koopman invariant subspace. If we have expert knowledge of the system, we can adopt a broader set of observation functions. Both extended DMD method [21,23,35] and kernel DMD method [20,24] obtain approximate finite-dimensional Koopman operators in a larger observation function space. But it is very trick to choose the observation functions in these methods. However, if we do not have the prior knowledge of nonlinear dynamical systems or the nonlinear dynamical systems are very complex, these methods are not able to work well for the nonlinear problems. Due to the complexity of practical problems and the rapid development of deep learning, it becomes popular to use deep neural networks to learn Koopman operator and its eigenfunctions [26,36].

In this paper, we present a complete data-driven method for nonlinear multiscale dynamical systems by learning Koopman invariant subspaces. The proposed method can be used to predict the future state of the multiscale dynamical systems. With the development of science and technology, we can obtain a lot of measurement data for practical models. We suppose that the multiscale basis functions are given by model's prior and the measurement data is obtained at some time instants. Neural networks with a single hidden layer and sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function [13]. The nonlinear coarse grid model can be treated as a multilayer neural network. But for time-dependent nonlinear problems, the time level $n-1$ is used as the input to the network and the time level n is the output. In order to get the state at time level n, it is necessary to repeatedly applying the network n times from the initial time. This is because the solution at time level n depends on the time level $n-1$ and the associated input parameters. The Koopman operator lifts the nonlinear problem to a linear problem in the observed function space. Spectral theory and eigenfunction decomposition result in an efficient prediction of the future state of the linear system. Only the eigenvalues, eigenvectors and initial values extracted from data are used to predict the state of n-th step without iteration. The challenge of using Koopman operator is the construction of invariant subspace. The Koopman invariant subspace is spanned by a set of observation functions with possibly strong nonlinearity. Neural networks can be used to accurately approximate these observation functions. Takeishi et al. [33] proposed a completely data-driven approach to learning Koopman invariant subspaces using deep learning. It learns a set of observation functions from the data and does not require to know the dynamical system itself. The loss function constructed by Takeishi et al. measures the discrepancy between the data and the estimated regression model to learn the Koopman invariant subspace. In order to reconstruct the original state space, an additional term is added to the loss function, which measures the distance between the original measurement data and the approximate measurement data obtained by the reconstruction operator. The Koopman invariant subspace learned in this way can achieve data fitting well. In this article, we are more concerned with the prediction of the future state. Therefore, we construct the loss function by measuring the distance between the observation data and the DMD approximation using the observation data. For the multiscale problems, the metric distance incorporates the matrix of the given multiscale basis functions. Thus, we can accurately downscale to fine-scale information from coarse-scale learning. Furthermore, we also use the neural network to learn the reconstruction operator, which maps the observation function space to the original solution space of the multiscale problem. DMD is used for evaluating the observed functions. In order to obtain the original state space, the observed functions are required to be invertible in some sense. Hence, we need to consider the effect of DMD to map the future observation back to the original state space. Based on the above motivations, we use two parts to learn the reconstruction operator, one is to reconstruct the original state, and the other is to test the effect of returning to the original state through DMD prediction. Once the Koopman invariant subspace and reconstruction operator are learned, we can predict the future state without iterating for a new set of measurement data. The new measurement data and the observation functions of Koopman invariant subspace are as the inputs in DMD algorithm. The DMD method produces the observation at any time in the future, and then by the reconstruction operator, the future state can be obtained immediately. There exist some variants for the Koopman based data-driven methods. We make comparison between these different methods through a few numerical examples.

This article is structured as follows. In Section 2, we give a short introduction of the nonlinear multiscale problems. In Section 3, we present the data-driven method with Koopman operator. In Section 4, we address learning the Koopman invariant subspace and the reconstruction of original state with neural networks. In Section 5, a few numerical results are presented to illustrate the long-term prediction of the proposed method for nonlinear multiscale dynamical problems. Finally, some conclusions and comments are given.

2. Preliminaries

Let Ω be a bounded domain in R^d $(d = 2, 3)$ and $T > 0$ be a fixed time. We consider the following general dynamical system

$$\frac{\partial u}{\partial t} = F(\kappa(x), u, \nabla u, I), \quad (x,t) \in \Omega \times (0, T], \tag{2.1}$$

where I represents model inputs, such as source terms, coefficients, initial and boundary value conditions. $\kappa(x)$ is a spatially multiscale field (e.g., permeability field in porous media), the ratio $\frac{\max(\kappa(x))}{\min(\kappa(x))}$ may be large. The solution $u(x,t)$ solves equation (2.1) subject input I.

It is usually computationally challenging to resolve all scales for multiscale problems. To overcome this difficulty, we usually solve this problem numerically in a N_c-dimensional space V_{ms}, where N_c is the number of coarse degrees of freedom, which is much smaller than the number of degrees of freedom on the fine scale. The solution can be expressed by

$$u(x, t_n) \approx u_h^n = \sum_{i=1}^{N_c} z_i^n \psi_i(x),$$

where $t_n = n\Delta t$, Δt is the time-step size and $\{\psi_i(x)\}_{i=1}^{N_c}$ is the multiscale basis functions in V_{ms}, we place them in the following basis matrix,

$$R = [\psi_1, \psi_2, \cdots, \psi_{N_c}].$$

The selection of multiscale basis functions $\{\psi_i(x)\}_{i=1}^{N_c}$ may be flexible, such as multiscale finite element basis function [14], constraint energy minimizing generalized multiscale finite element basis function [6], non-local multicontinuum basis function [7] and so on. The coefficient $\boldsymbol{z}^n = (z_1^n, z_2^n, \cdots, z_{N_c}^n)^T$ may be related a coarse-scale moment information for the solution $u(x, t^n)$.

The continuous-time dynamic system (2.1) can be rewritten by the following discrete-time dynamic system

$$\boldsymbol{z}^{n+1} = \boldsymbol{f}(\boldsymbol{z}^n). \tag{2.2}$$

We use the following nonlinear multiscale parabolic problem as an example and derive its discrete dynamic system in coarse-scale. Let us consider the model

$$u_t - \nabla \cdot \big(\kappa(x) b(u) \nabla u\big) = h(x) \ \text{ in } \Omega \times (0, T], \tag{2.3}$$

where $b(u)$ is a nonlinear function of u. Eq. (2.3) can model the diffusion process of flows in heterogeneous porous media. For examples, Richard's equation, p-Laplacian equation and porous media equation can be rewritten in this form.

If we use Galerkin finite element method to solve this problem and use explicit Euler temporal discretization, then this problem is transformed into: find $u_h^{n+1} \in V_{ms}$ such that

$$(u_h^{n+1} - u_h^n, v) + \Delta t a(u_h^{n+1}, v) = \Delta t(h, v) \quad \forall v \in V_{ms}$$

where

$$a(u_h^{n+1}, v) = \int_\Omega \kappa(x) b(u_h^n) \nabla u_h^{n+1} \cdot \nabla v, \quad (h, v) = \int_\Omega h(x) v.$$

Due to $u_h^n = \sum_{i=1}^{N_c} z_i^n \psi_i(x)$, the coefficient $\boldsymbol{z}^n = (z_1^n, z_2^n, \cdots, z_{N_c}^n)^T$ satisfies the recurrence relation

$$\boldsymbol{z}^{n+1} = \boldsymbol{f}(\boldsymbol{z}^n) = (\boldsymbol{M} + \Delta t \boldsymbol{N}(\boldsymbol{z}^n))^{-1}(\Delta t \boldsymbol{H} + \boldsymbol{M}\boldsymbol{z}^n),$$

where

$$\begin{aligned}
\boldsymbol{M}_{ij} &= \int_\Omega \psi_i(x)\psi_j(x), \\
\boldsymbol{N}(\boldsymbol{z}^n)_{ij} &= \int_\Omega \kappa(x) b(\sum_{i=1}^{N_c} z_i^n \psi_i(x)) \nabla \psi_i(x) \cdot \nabla \psi_j(x), \\
\boldsymbol{H}_i &= \int_\Omega h(x))\psi_i(x).
\end{aligned}$$

We note that implicit Euler can also be used to discretize the temporal space and derive a discrete-time dynamic system similarly. But in this case, $\boldsymbol{f}(\boldsymbol{z}^n)$ may not have an explicit expression.

Traditional multiscale numerical methods are used to simulate dynamic multiscale problems when the model (2.1) or (2.2) is known. Using equation (2.2), $\boldsymbol{z}^n$ can be obtained recursively, thus we can get the approximate solution u_h^n. However, when the recurrence relationship (2.2) is very complicated or even unknown, the traditional numerical methods will not work. We will learn the dynamic model using a data-driven approach. For multiscale problems, it is very expensive to directly collect information in a fine scale. Coarse-scale data is desirable instead. In this article, we suppose that the truth multiscale model is unknown, but multiscale basis functions $\{\psi_i(x)\}_{i=1}^{N_c}$ and some data $\{\boldsymbol{z}^j\}_{j=1}^m$ are available. We will develop a data-driven, equation-free approach to accurately predict states of future without knowing the model (2.1) or (2.2).

3. Data-driven method using Koopman operator

In this section, we present a data-driven method using Koopman operator. The Koopman operator is a linear, infinite-dimensional operator. It does not linearize the dynamical system, but instead transforms the finite-dimensional nonlinear problem into an linear system by acting on the observation function space. DMD is an equation-free, data-driven method capable of providing approximate modes of Koopman operator. DMD model is constructed through calculating the eigenvalues and eigenvectors of the finite dimensional linear model from the observation data. In this paper, we call the data $\boldsymbol{z}$ as the measurement data, $\boldsymbol{g}(\boldsymbol{z})$ as the observation data, where $\boldsymbol{g}$ is a vector-valued observation function. If the selected observation functions span a Koopman invariant subspace, then DMD method can provide accurate modes of Koopman operators provided that there are enough data. However, when the model is completely unknown, it is very challenging to choose the observation functions to span the Koopman invariant subspace. Even if the prior information of the model is known, it is still a difficult problem. In this work, we only use the measurement data to learn a Koopman invariant subspace by neural networks.

3.1. Koopman operator

We focus on a (possibly nonlinear) discrete-time dynamical system

$$\boldsymbol{z}^{n+1} = \boldsymbol{f}(\boldsymbol{z}^n), \quad \boldsymbol{z} \in \mathcal{M}, \quad n \in T := 0 \cup \mathbb{N}, \tag{3.4}$$

where $\mathcal{M}$ denotes the state space and $\boldsymbol{f}$ is a map from $\mathcal{M}$ to itself. The Koopman operator is a linear infinite-dimensional operator $\mathcal{K}$ that acts on scalar-valued functions on $\mathcal{M}$ given by the following manner: for any scalar-valued observation function $g:\mathcal{M}\to\mathbb{C}$,

$$\mathcal{K}g(\boldsymbol{z}) := g(\boldsymbol{f}(\boldsymbol{z})). \tag{3.5}$$

So the nonlinear dynamical system (3.4) can be lifted to the linear (but infinite-dimensional) problem (3.5). However, numerical computation of an infinite dimensional operator $\mathcal{K}$ is infeasible. To fulfil the numerical computation, we restrict the infinite-dimensional system to a finite-dimensional invariant subspace.

Suppose that there exists an invariant subspace $\mathbf{G}$ of $\mathcal{K}$, i.e.,

$$\mathcal{K}g\in\mathbf{G}, \forall g\in\mathbf{G}.$$

Let a set of observation functions $\{g_1,\dots,g_q\}(q<\infty)$ span $\mathbf{G}$. We consider the restriction of $\mathcal{K}$ to $\mathbf{G}$ and denote it by $\mathcal{K}|_{\mathbf{G}}$. Thus $\mathcal{K}|_{\mathbf{G}}$ is a finite-dimensional linear operator. Let $\boldsymbol{g}=[g_1,\dots,g_q]^T$ be a vector-valued observation. Then $\mathcal{K}|_{\mathbf{G}}$ has a matrix-form representation $\boldsymbol{K}$ with respect to $\{g_1,\dots,g_q\}$, i.e.,

$$\begin{aligned}\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{z})) &= \begin{bmatrix} g_1(\boldsymbol{f}(\boldsymbol{z}))\\ g_2(\boldsymbol{f}(\boldsymbol{z}))\\ \vdots\\ g_q(\boldsymbol{f}(\boldsymbol{z}))\end{bmatrix} = \begin{bmatrix}\mathcal{K}g_1(\boldsymbol{z})\\ \mathcal{K}g_2(\boldsymbol{z})\\ \vdots\\ \mathcal{K}g_q(\boldsymbol{z})\end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1q}\\ k_{21} & k_{22} & \dots & k_{2q}\\ \vdots & \vdots & & \vdots\\ k_{q1} & k_{q2} & \dots & k_{qq}\end{bmatrix}\begin{bmatrix} g_1(\boldsymbol{z})\\ g_2(\boldsymbol{z})\\ \vdots\\ g_q(\boldsymbol{z})\end{bmatrix}\\ &= \boldsymbol{K}\boldsymbol{g}(\boldsymbol{z}).\end{aligned}$$

The following theorem gives a representation of matrix $\boldsymbol{K}$, which is the matrix form of Koopman operator under the basis $\{g_1, g_2\dots,g_q\}$.

Theorem 3.1. *[33]: Let $\{g_1,g_2\dots,g_q\}$ be a set of square-integrable observation functions and the matrix $\boldsymbol{K}:=(\int_{\mathcal{M}}(\boldsymbol{g}\circ\boldsymbol{f})d\mu)(\int_{\mathcal{M}}(\boldsymbol{g}\boldsymbol{g}^T)d\mu)^{\dagger}$. Then $\forall\boldsymbol{z}\in\mathcal{M}$, $\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{z}))=\boldsymbol{K}\boldsymbol{g}(\boldsymbol{z})$ if and only if $\{g_1,g_2\dots,g_q\}$ spans a Koopman invariant subspace.*

The spectral decomposition theory of Koopman operator can give an expression for the observation functions. To this end, we consider the eigendecomposition of the matrix $\boldsymbol{K}$. Let $\boldsymbol{\phi}_j$ and $\boldsymbol{\omega}_j$ be the right eigenvector and left eigenvector of $\boldsymbol{K}$ corresponding to the eigenvalue λ_j, respectively. Then

$$\boldsymbol{K}\boldsymbol{\phi}_j=\lambda_j\boldsymbol{\phi}_j,\ \ \boldsymbol{\omega}_j^T\boldsymbol{K}=\lambda_j\boldsymbol{\omega}_j^T.$$

Without loss of generality, we may assume that $\boldsymbol{\omega}_i^T\boldsymbol{\phi}_j=\delta_{ij}$. If the matrix $\boldsymbol{K}\in\mathbb{C}^{q\times q}$ has q linearly independent eigenvectors, then any $\boldsymbol{g}\in\mathbf{G}$ can be expressed by a linear combination of the eigenvectors, i.e.,

$$\boldsymbol{g}(\boldsymbol{z})=\sum_{j=1}^{q}v_j(\boldsymbol{z})\boldsymbol{\phi}_j, \tag{3.6}$$

where $v_j(\boldsymbol{z})=\boldsymbol{\omega}_j^T\boldsymbol{g}(\boldsymbol{z})$. We note that $\{v_j(\boldsymbol{z})\}_{j=1}^q$ are the eigenfunctions of Koopman operator $\mathcal{K}$. In fact,

$$\begin{aligned}\mathcal{K}v_j(\boldsymbol{z}) &= \boldsymbol{\omega}_j^T\mathcal{K}\boldsymbol{g}(\boldsymbol{z})=\boldsymbol{\omega}_j^T\boldsymbol{K}\boldsymbol{g}(\boldsymbol{z})\\ &=\boldsymbol{\omega}_j^T\sum_{i=1}^{q}v_i(\boldsymbol{z})\boldsymbol{K}\boldsymbol{\phi}_i\\ &=\lambda_jv_j(\boldsymbol{z}).\end{aligned}$$

Thus Eq. (3.6) implies that $\boldsymbol{g}(\boldsymbol{z})$ is a linear combination the eigenvectors $\{\boldsymbol{\phi}_j\}_{j=1}^q$ of the matrix $\boldsymbol{K}$, or the eigenfunctions $\{v_j(\boldsymbol{z})\}_{j=1}^q$ of the Koopman operator $\mathcal{K}$. For a sequential time series, repeatedly applying Koopman operator to Eq. (3.6) gives

$$\boldsymbol{g}(\boldsymbol{z}^n)=\sum_{j=1}^{q}\lambda_j^nv_j(\boldsymbol{z}^0)\boldsymbol{\phi}_j. \tag{3.7}$$

This sequence of triples $\{\lambda_j,v_j(\boldsymbol{z}),\boldsymbol{\phi}_j\}$ are called Koopman eigenvalues, eigenfunctions and modes. The Koopman triples enable us to evaluate the function $\boldsymbol{g}(\boldsymbol{z})$ at any time. So far we have assumed that the set of observation functions in a Koopman invariant subspace are available. Next we present DMD to estimate the eigenvalues and modes of Koopman from a set of measurement data and a set of observation functions, which span a Koopman invariant subspace.

3.2. Dynamic mode decomposition

In this subsection, we assume that the set of observation functions $\boldsymbol{g}$ in a Koopman invariant subspace are available. If the dynamic system is known, these observation functions $\boldsymbol{g}$ can be obtained through a careful artificial selection, which is usually very challenging. The DMD method proposed by Schmid [30] corresponds to identity observation function selection. Williams et al. [35] have used a dictionary of predefined functions as the observation functions $\boldsymbol{g}$. The widely used dictionary functions are Hermite polynomials and radial basis functions. The selection of dictionary functions is based on the prior knowledge of the dynamic system of interest. However, these are artificial selection of some functions as an approximation of the Koopman invariant subspace. For complex dynamic systems, these artificially selected functions may not span a Koopman invariant subspace. This may lead to an inaccurate approximation of the DMD methods. In this article, we will use neural network to learn Koopman invariant subspace. We will describe the learning method in Section 4. In this subsection, we first review the construction of Koopman's eigenvalues and modes by DMD method using measurement data under the assumption: the set of observation functions $\boldsymbol{g}$ in a Koopman invariant subspace have been given.

The measurement data may be obtained either by numerical simulation or by experiment. Suppose that we have a snapshot sequence of data $\{\boldsymbol{z}^0, \boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^m\}$, DMD [30] is used to extract the features of data. Let $\boldsymbol{g}$ be the given observation function. We define the data matrices of observables $\mathbf{Y}_0$ and $\mathbf{Y}_1$ as follows:

$$\mathbf{Y}_0 = \begin{bmatrix} | & | & & | \\ \boldsymbol{g}(\boldsymbol{z}^0) & \boldsymbol{g}(\boldsymbol{z}^1) & \ldots & \boldsymbol{g}(\boldsymbol{z}^{m-1}) \\ | & | & & | \end{bmatrix}$$

$$\mathbf{Y}_1 = \begin{bmatrix} | & | & & | \\ \boldsymbol{g}(\boldsymbol{z}^1) & \boldsymbol{g}(\boldsymbol{z}^2) & \ldots & \boldsymbol{g}(\boldsymbol{z}^{m}) \\ | & | & & | \end{bmatrix}.$$

Let the DMD matrix

$$\boldsymbol{A} := \mathbf{Y}_1 \mathbf{Y}_0^{\dagger}, \tag{3.8}$$

where $\mathbf{Y}_0^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{Y}_0$. The eigenvectors and eigenvalues of $\boldsymbol{A}$ are the DMD modes and eigenvalues.

Remark 3.1. If the matrix $\mathbf{Y}_0\mathbf{Y}_0^T$ is invertible, the matrix $\boldsymbol{A}$ in (3.8) is the least-squares solution. That is,

$$\boldsymbol{A} = \underset{\widehat{\boldsymbol{A}}}{\operatorname{argmin}} \|\mathbf{Y}_1 - \widehat{\boldsymbol{A}}\mathbf{Y}_0\|_F^2.$$

It is worth noting that if $\mathbf{Y}_0$ is not a full rank, then the minimization problem does not have a unique solution. In this case, Tikhonov-Phillips [32] regularization makes the minimization problem become

$$\begin{aligned} \boldsymbol{A} &= \underset{\widehat{\boldsymbol{A}}}{\operatorname{argmin}} \|\mathbf{Y}_1 - \widehat{\boldsymbol{A}}\mathbf{Y}_0\|_F^2 + \lambda\|\widehat{\boldsymbol{A}}\|_F^2. \\ &= \mathbf{Y}_1\mathbf{Y}_\lambda^{\dagger}, \end{aligned}$$

where $\mathbf{Y}_\lambda^{\dagger} := \mathbf{Y}_0^T[\mathbf{Y}_0\mathbf{Y}_0^T + \lambda I]^{-1}$.

We recall that $\boldsymbol{K}$ is the matrix form of Koopman operator under the basis function. The following theorem gives the relationship between matrix $\boldsymbol{K}$ and DMD matrix $\boldsymbol{A}$.

Theorem 3.2. *[33]: If $\{g_1, \ldots, g_q\}$ spans a Koopman invariant subspace and assume that the time average of the observation function converges to its spatial average (i.e., $\lim_{m\to\infty} \frac{1}{m}\sum_{j=0}^{m} g(\boldsymbol{z}^{\boldsymbol{j}}) = \int_{\mathcal{M}} g(\boldsymbol{z})d\mu$), then almost surely*

$$\lim_{m\to\infty} \|\boldsymbol{K} - \boldsymbol{A}\| = 0.$$

In order to accurately learn the state of the nonlinear dynamical system, the Koopman invariant subspace is usually selected with a high dimension q. Thus it may be very expensive to do eigendecomposition directly on the matrix $\boldsymbol{A}$. Tu et al. [34] proposed exact DMD, which reconstructs the nonzero eigenvalues and eigenvectors of $\boldsymbol{A}$ by calculating the eigendecomposition of a low-dimensional projection of $\boldsymbol{A}$.

Here, we give the main steps of exact DMD.

- First, take the SVD of $\mathbf{Y}_0$:

$$\mathbf{Y}_0 = \boldsymbol{U\Sigma V^T},$$

where $\boldsymbol{U} \in \mathbb{C}^{q\times r}$, $\boldsymbol{\Sigma} \in \mathbb{C}^{r\times r}$ and $\boldsymbol{V} \in \mathbb{C}^{m\times r}$. Here r is the truncated rank chosen by the hard threshold technique [31]. The left singular vectors in $\boldsymbol{U}$ are POD modes.
- Next, compute the $r \times r$ projection $\widetilde{\boldsymbol{A}}$ of the full matrix $\boldsymbol{A}$ onto POD modes by

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U^T A U} = \boldsymbol{U^T}\mathbf{Y_1}\boldsymbol{V\Sigma^{-1}}.$$

- Compute eigenvalues and right-left eigenvectors of $\widetilde{\boldsymbol{A}}$, respectively, by

$$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{\Phi}} = \widetilde{\boldsymbol{\Phi}}\boldsymbol{\Lambda}, \quad \widetilde{\boldsymbol{W}}^T\widetilde{\boldsymbol{A}} = \boldsymbol{\Lambda}\widetilde{\boldsymbol{W}}^T,$$

where the columns of $\widetilde{\boldsymbol{\Phi}}$ and $\widetilde{\boldsymbol{W}}$ are the right-left eigenvectors, respectively, and $\boldsymbol{\Lambda}$ is the diagonal matrix consisting of the corresponding eigenvalues $\{\lambda_i\}_{i=1}^r$.
- Reconstruct eigendecomposition of $\boldsymbol{A}$ from $\widetilde{\boldsymbol{\Phi}}$, $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{\Lambda}$. In particular, $\{\lambda_i\}_{i=1}^r$ are also the eigenvalues of $\boldsymbol{A}$, and the eigenvectors of $\boldsymbol{A}$ given by

$$\boldsymbol{\Phi} = \mathbf{Y_1}\boldsymbol{V\Sigma^{-1}}\widetilde{\boldsymbol{\Phi}}, \quad \boldsymbol{W} = \boldsymbol{U}\widetilde{\boldsymbol{W}},$$

where $\boldsymbol{\Phi}, \boldsymbol{W} \in \mathbb{C}^{q\times r}$.
- Finally, the approximate solution for prediction is given by

$$\boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^n) = \boldsymbol{\Phi\Lambda}^n\boldsymbol{v}(\boldsymbol{z}^0), \;\; n = 0, 1\cdots, \tag{3.9}$$

where $\boldsymbol{v}(\boldsymbol{z}^0) = \boldsymbol{W}^T\boldsymbol{g}(\boldsymbol{z}^0)$.

3.3. The prediction accuracy of DMD in Koopman invariant subspace

Once the eigenvalues and modes of Koopman are available, Eq. (3.9) can be used to predict the future state of the observed function $\boldsymbol{g}(\boldsymbol{z})$. In this subsection, we analyze the accuracy of DMD prediction. The error analysis allows one to quantify the prediction effect of DMD.

We rewrite the prediction equation (3.9) of Exact DMD by

$$\begin{aligned}\boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^{n+1}) &= \sum_{j=1}^r \lambda_j^{n+1} v_j(\boldsymbol{z}^0)\boldsymbol{\phi}_j \\ &= \boldsymbol{\Phi\Lambda}^{n+1}\boldsymbol{v}(\boldsymbol{z}^0) \\ &= \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^n),\end{aligned}$$

where $\boldsymbol{W}^T\boldsymbol{\Phi} = \boldsymbol{I}$, $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_r]$, and $\boldsymbol{W} = [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_r]$.

Theorem 3.3. *Assume that $\{g_1, \ldots, g_q\}$ spans a Koopman invariant subspace and $\boldsymbol{g}(\boldsymbol{z}^{n+1}) = \boldsymbol{K}\boldsymbol{g}(\boldsymbol{z}^n)$, where the matrix $\boldsymbol{K} \in \mathbb{C}^{q\times q}$ has q linearly independent eigenvectors. Let m be the number of snapshots and r the truncated rank. Define the local error and global error, respectively, by*

$$e_{local}^{n+1} := \boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}(\boldsymbol{z}^n), \quad e_{global}^{n+1} := \boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{g}_{DMD}(\boldsymbol{z}^{n+1}).$$

Then

$$\|e_{global}^{n+1}\| \le \big(\|\boldsymbol{K} - \boldsymbol{A}\| + \|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\|\big)\|\boldsymbol{g}(\boldsymbol{z}^0)\|\sum_{j=0}^n \sigma(\boldsymbol{K})^j\|\boldsymbol{\Phi\Lambda}^j\boldsymbol{W}^T\|.$$

In particular, the $\|e_{local}^{n+1}\|$ and $\|e_{global}^{n+1}\|$ decrease as m and r increase.

Proof. Although $\boldsymbol{\Phi}, \boldsymbol{W}$ and $\boldsymbol{\Lambda}$ are the exact right-left eigenvectors and eigenvalues of $\boldsymbol{A}$ (in $\mathbb{C}^{q\times q}$), they are all induced by the lower dimensional matrix $\widetilde{\boldsymbol{A}} \in \mathbb{C}^{r\times r}$ in DMD algorithm. Thus $\boldsymbol{\Phi}, \boldsymbol{W}$ and $\boldsymbol{\Lambda}$ can only determine r eigenvectors and eigenvalues of $\boldsymbol{A}$, and

$$\lim_{r\to q}\|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\| = 0.$$

Since the local error assumes that the n-th input $\boldsymbol{g}(\boldsymbol{z}^n)$ is accurate, $n > m$, then

$$\begin{aligned}
\|e_{local}^{n+1}\|_2 &= \|\boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}(\boldsymbol{z}^n)\|_2 \\
&= \|(\boldsymbol{K} - \boldsymbol{\Phi\Lambda W}^T)\boldsymbol{g}(\boldsymbol{z}^n)\|_2 \\
&= \|(\boldsymbol{K} - \boldsymbol{A} + \boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T)\boldsymbol{g}(\boldsymbol{z}^n)\|_2 \\
&\le (\|\boldsymbol{K} - \boldsymbol{A}\| + \|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\|)\|\boldsymbol{g}(\boldsymbol{z}^n)\| \\
&\le (\|\boldsymbol{K} - \boldsymbol{A}\| + \|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\|)\|\boldsymbol{K}^n\boldsymbol{g}(\boldsymbol{z}^0)\| \\
&\le (\|\boldsymbol{K} - \boldsymbol{A}\| + \|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\|)\sigma(\boldsymbol{K})^n\|\boldsymbol{g}(\boldsymbol{z}^0)\|,
\end{aligned}$$

where $\sigma(\boldsymbol{K})$ represents the largest singular value of $\boldsymbol{K}$.

The global error is measured by the difference between the true value of the $(n+1)$-th step and the value of the $(n+1)$-th step of the prediction iteration. Thus

$$\begin{aligned}
e_{global}^{n+1} &= \boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^{n+1}) \\
&= \boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^n) \\
&= \boldsymbol{g}(\boldsymbol{z}^{n+1}) - \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}(\boldsymbol{z}^n) + \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}(\boldsymbol{z}^n) - \boldsymbol{\Phi\Lambda W}^T\boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^n) \\
&= e_{local}^{n+1} + \boldsymbol{\Phi\Lambda W}^T e_{global}^n \\
&= e_{local}^{n+1} + \boldsymbol{\Phi\Lambda W}^T(e_{local}^n + \boldsymbol{\Phi\Lambda W}^T e_{global}^{n-1}) \\
&= \cdots \\
&= e_{local}^{n+1} + \boldsymbol{\Phi\Lambda W}^T e_{local}^n + \cdots + \boldsymbol{\Phi\Lambda}^{n-1}\boldsymbol{W}^T e_{local}^2 + \boldsymbol{\Phi\Lambda}^n\boldsymbol{W}^T e_{global}^1 \\
&= \sum_{j=0}^{n} \boldsymbol{\Phi\Lambda}^j\boldsymbol{W}^T e_{local}^{n+1-j}.
\end{aligned}$$

The last equation holds because

$$e_{global}^1 = \boldsymbol{g}(\boldsymbol{z}^1) - \boldsymbol{g}_{\mathrm{DMD}}(\boldsymbol{z}^0) = \boldsymbol{g}(\boldsymbol{z}^1) - \boldsymbol{\Phi\Lambda W}^T(\boldsymbol{z}^0) = e_{local}^1.$$

Consequently, we have

$$\|e_{global}^{n+1}\| \le (\|\boldsymbol{K} - \boldsymbol{A}\| + \|\boldsymbol{A} - \boldsymbol{\Phi\Lambda W}^T\|)\|\boldsymbol{g}(\boldsymbol{z}^0)\|\sum_{j=0}^{n}\sigma(\boldsymbol{K})^j\|\boldsymbol{\Phi\Lambda}^j\boldsymbol{W}^T\|.$$

By Theorem 3.2,

$$\lim_{m\to\infty}\|\boldsymbol{K} - \boldsymbol{A}\| = 0,$$

where m is the number of snapshots. Therefore, with the increase of the number of snapshots m and the truncated rank r, $\|e_{local}^{n+1}\|$ and $\|e_{global}^{n+1}\|$ decrease. □

In this section, we have presented the DMD method to predict the future state of the observation function in the Koopman invariant subspace, and analyzed its prediction accuracy. It should be emphasized that it may be difficult to find a set of observed functions spanning the Koopman invariant subspace. Furthermore, we want to estimate the state of nonlinear dynamical system and have to reconstruct the original state from the observation functions. DMD itself can not fulfill this goal. Next, we will use deep learning to learn a nontrivial Koopman invariant subspace and the reconstruction of original state.

4. Learning Koopman invariant subspace and reconstruction of original state

Choosing the observation functions $\boldsymbol{g}$ that span a Koopman invariant subspace plays a very important role. It significantly effects on the finite dimensional approximation of the Koopman invariant subspace. The artificial selections of observation functions in DMD methods may not span a Koopman invariant subspace, and render inaccurate models. In this section, we use measurement data to learn the observation functions spanning a Koopman invariant subspace through neural networks.

A neural network is usually used to learn a map from existing data [13]. Fig. 4.1 depicts a generic architecture of a multi-layer neural network. It has input layer, hidden layers and output layer. Given enough hidden units, the neural network can be used to approximate any Borel measurable function [13]. If the neural network $\mathcal{N}(\cdot)$ has a feed-forward fully-connected structure, then

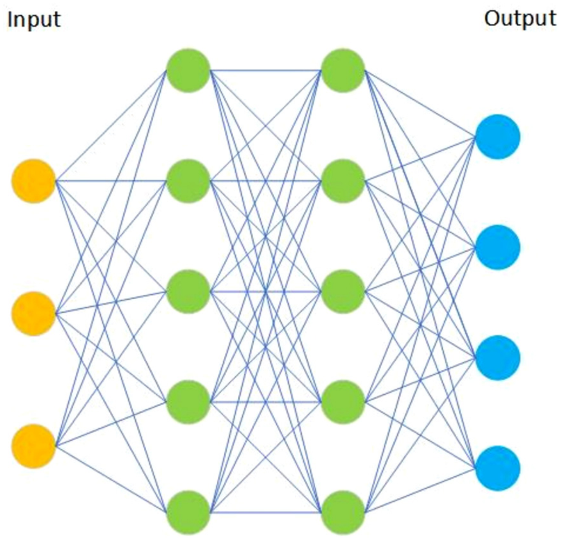

Fig. 4.1. Illustration of a neural network architecture mapping an input layer to an output layer. The middle layers are hidden layers. The number of layers and the map between layers are selected by users.

$$\mathcal{N}(\boldsymbol{z};\theta) = W_n\sigma(\cdots\sigma(W_2\sigma(W_1\boldsymbol{z}+b_1)+b_2)\cdots)+b_n.$$

Here, $\boldsymbol{z}$ is the input data, $\sigma(W_1\boldsymbol{z}+b_1)$ represents the first hidden layer, $\mathcal{N}(\boldsymbol{z};\theta)$ is the output, $\sigma(\cdot)$ is nonlinear activation function and $\theta=\{W_1,b_1,\cdots,W_n,b_n\}$ represents tuneable parameters in the neural network. We use $\mathcal{N}(\boldsymbol{z};\theta)$ to approximate the desired output and use a loss function to evaluate the output of the neural network. By minimizing the loss function, the optimal parameters θ^* are solved for the neural network. This optimization problem can be solved by gradient descent type methods, such as stochastic gradient descent method (SGD) [1] or Adam optimizer [19]. In order to minimize the loss function, the parameters θ are updated iteratively. This process is called training. Once the optimal parameters θ^* are estimated, the neural network architecture is constructed. For a new input $\widehat{\boldsymbol{z}}$, $\mathcal{N}(\widehat{\boldsymbol{z}};\theta^*)$ gives the corresponding output.

We need to find a set of observation functions $\{g_1,g_2,\cdots,g_q\}$ spanning a Koopman invariant subspace, which plays an important role in transforming the nonlinear problem into a finite dimensional linear system. In this section, we will use neural networks to learn these observation functions from data. We use the fully-connected network structure to represent the observation functions $\boldsymbol{g}=[g_1,\ldots,g_q]^T$ in the Koopman invariant subspace, that is, $\boldsymbol{g}(\boldsymbol{z})=\mathcal{N}^1(\boldsymbol{z};\theta^1)$. The superscript 1 represents the first network. To this end, we need to construct a suitable loss function. In our problem, we use multiple trajectories $\{\boldsymbol{z}^{0,i},\boldsymbol{z}^{1,i},\cdots,\boldsymbol{z}^{m,i}\}$ to train the network. For $\boldsymbol{z}^{m,i}$, the superscript m represents the m-th moment, and the superscript i represents the i-th sample. For simplicity of presentation, we take a trajectory $\{\boldsymbol{z}^0,\boldsymbol{z}^1,\cdots,\boldsymbol{z}^m\}$ as an example to introduce the construction of the loss function. For multiple trajectories, the loss function is the average of multiple trajectories. Since we use the eigenvalues and modes of the Koopman operator to predict the future observation state, we do not explicitly learn the matrix $\boldsymbol{K}$. Therefore, we construct the loss function by measuring the distance between the observed states and the estimated observed states by the spectral decomposition of the linear operator $\mathcal{K}$. Thus, the loss function of learning Koopman invariant subspace is defined by

$$\begin{aligned}\mathcal{L}_{LI}(\boldsymbol{g};(\boldsymbol{z}^0,\boldsymbol{z}^1,\cdots,\boldsymbol{z}^{m_1})) &= \sum_{i=0}^{m_1}\|\boldsymbol{g}(\boldsymbol{z}^i)-\boldsymbol{g}_{\text{DMD}}(\boldsymbol{z}^i)\|^2\\ &= \sum_{i=0}^{m_1}\|\boldsymbol{g}(\boldsymbol{z}^i)-\boldsymbol{\Phi}\boldsymbol{\Lambda}^i\boldsymbol{v}(\boldsymbol{z}^0)\|^2,\end{aligned}$$

where $\boldsymbol{\Phi}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{v}(\boldsymbol{z}^0)$ can be obtained from $\{\boldsymbol{g}(\boldsymbol{z}^i)\}_{i=0}^{m_1}$ by exact DMD, $\boldsymbol{g}_{\text{DMD}}(\boldsymbol{z}^i)=\boldsymbol{\Phi}\boldsymbol{\Lambda}^i\boldsymbol{v}(\boldsymbol{z}^0)$.

Once we know the invariant subspace, we can simulate the observation function $\boldsymbol{g}(\boldsymbol{z})$ at any time by Eq. (3.9). However, our goal is to predict the evolution of the state $\boldsymbol{z}$. Thus we need to reconstruct the original state by using another neural network. Let $\boldsymbol{h}=\mathcal{N}^2(\boldsymbol{y};\theta^2)$ be the reconstruction operator. We want $\boldsymbol{z}\approx\boldsymbol{h}(\boldsymbol{g}(\boldsymbol{z}))$. We expect the reconstruction operator $\boldsymbol{h}$ not only to recover the state space from the Koopman invariant subspace, but also recover the original state for DMD prediction. Therefore, the training includes two parts. The data of the first m_1 moments is used for learning the map $\boldsymbol{g}(\boldsymbol{z})\to\boldsymbol{z}$. The rest of data $(\boldsymbol{z}^{m_1+1},\cdots,\boldsymbol{z}^m)$ is used to identify $\boldsymbol{z}$ by composting $\boldsymbol{h}$ and DMD prediction. At the same time, we also hope that the composition of $\boldsymbol{h}$ and the DMD model obtained by using the data $(\boldsymbol{z}^0,\cdots,\boldsymbol{z}^{m_1})$ is able to well approximate the state of the first m_1 moments. Therefore we define the loss function of learning reconstruction by

$$\mathcal{L}_{LR}(\boldsymbol{g},\boldsymbol{h};(\boldsymbol{z}^0,\boldsymbol{z}^1,\cdots,\boldsymbol{z}^{m_1},\cdots,\boldsymbol{z}^m))=\alpha_1\sum_{i=0}^{m_1}\|\boldsymbol{z}^i-\boldsymbol{h}(\boldsymbol{g}(\boldsymbol{z}^i))\|^2 + \alpha_2\sum_{i=0}^{m}\|\boldsymbol{z}^i-\boldsymbol{h}(\boldsymbol{g}_{\text{DMD}}(\boldsymbol{z}^i))\|^2.$$

Since the reconstruction operator $\boldsymbol{h}$ strongly depends on the observation function $\boldsymbol{g}$, we need to design a loss function to learn $\boldsymbol{g}$ and $\boldsymbol{h}$ simultaneously. Then the final loss function should be

$$\mathcal{L}(\boldsymbol{g},\boldsymbol{h};(\boldsymbol{z}^0,\boldsymbol{z}^1,\cdots,\boldsymbol{z}^{m_1},\boldsymbol{z}^{m_1+1},\cdots,\boldsymbol{z}^m))=\alpha_0\mathcal{L}_{LI}+\mathcal{L}_{LR}. \tag{4.10}$$

Here α_0, α_1 and α_2 are hyperparameters. α_2 controls the effect of predicting the future state, α_0 and α_1 control the effect of data fitting. If we are more concerned about the prediction effect, then the corresponding need α_2 is larger than α_0 and α_1. If we only care about the data fitting effect, α_0, α_1 and α_2 can be with the same weights. The neural network is constructed by seeking θ^* such that

$$\theta^*=\underset{\theta}{\operatorname{argmin}}\,\mathcal{L}(\boldsymbol{g},\boldsymbol{h};(\boldsymbol{z}^0,\boldsymbol{z}^1,\cdots,\boldsymbol{z}^m)),$$

where $\theta=(\theta^1,\theta^2)$ is from $\mathcal{N}^1(\boldsymbol{z};\theta^1)$ and $\mathcal{N}^2(\boldsymbol{y};\theta^2)$.

For multiscale problems, in order to downscale the coarse-scale degrees of freedom (DOF) to the fine-scale degrees of freedom, we use multiscale basis functions and define a weighted l^2 norm by

$$\|\boldsymbol{z}\|_R=\sqrt{\boldsymbol{z}^T R^T R\boldsymbol{z}}$$

where $\boldsymbol{z}$ is the coarse-scale DOF and R is the multiscale basis matrix. Therefore, when we recover the fine-scale solution, the loss function for the multiscale problems is defined by the weighted l^2 norm, i.e.,

$$\begin{aligned}&\mathcal{L}(\boldsymbol{g},\boldsymbol{h};(\boldsymbol{z}^0,\cdots,\boldsymbol{z}^{m_1},\cdots,\boldsymbol{z}^m))\\&=\alpha_0\sum_{i=0}^{m_1}\|\boldsymbol{g}(\boldsymbol{z}^i)-\boldsymbol{\Phi}\boldsymbol{\Lambda}^i\boldsymbol{v}(\boldsymbol{z}^0)\|_R^2+\alpha_1\sum_{i=0}^{m_1}\|\boldsymbol{z}^i-\boldsymbol{h}(\boldsymbol{g}(\boldsymbol{z}^i))\|_R^2\\&+\alpha_2\sum_{i=0}^{m}\|\boldsymbol{z}^i-\boldsymbol{h}(\boldsymbol{g}_{\text{DMD}}(\boldsymbol{z}^i))\|_R^2.\end{aligned} \tag{4.11}$$

In a multiscale dynamical problem, we usually only get some coarse-scale data, but we are also interested in the fine-scale solution. The fine-scale solution can be approximated by using the multiscale basis function. Therefore, for multiscale problems, we use Eq. (4.11) to construct the loss function.

In order to train our network, we generate trajectories from random initial conditions and divide them into training set and test set. In the training set, a part of the trajectories are randomly selected as the validation set. In the training process, by comparing the value of loss function in the training set and the validation set, we can take an early stop to prevent over fitting.

Once the Koopman invariant subspace spanned by $\{g_1,g_2,\cdots,g_q\}$ is learned, the original nonlinear problem can be lifted to a finite dimensional linear problem. Using data and DMD to obtain Koopman's eigenvalues and modes, we can predict the observed state $\boldsymbol{g}(\boldsymbol{z})$ at any time instant. By the reconstruction operator $\boldsymbol{h}$, we can estimate the state of $\boldsymbol{z}$ at any time. Fig. 4.2 shows a schematic diagram of the proposed learning. Algorithm 1 summarizes the main steps of the proposed method: dynamic mode decomposition method based on learning Koopman invariant subspace and reconstruction operator (LIR-DMD). We note that the prediction of the future state is obtained through initial values, Koopman's eigenvalues and modes, not from the iterative solution of the initial state over time. This is significantly different from the direct deep learning and can significantly improve the online computation for prediction.

5. Numerical results

In this section, we present a few numerical results for the prediction of nonlinear multiscale dynamical systems by using LIR-DMD and make some comparisons with different strategies. In Section 5.1, we apply the proposed method to a fixed point attractor problem to verify the theoretical result of DMD prediction. In Section 5.2, we compare the prediction effects using exact DMD, extended DMD and LIR-DMD for a nonlinear multi-scale parabolic problem. In Section 5.3, we show the numerical results on porous media equation using CEM-DMD (a data & model-driven method) and the fully data-driven method LIR-DMD.

Let $\Omega:=[0,1]^2$ be the spatial domain and $(0,T]$ temporal interval for the nonlinear multiscale numerical examples. We use FEM on a 100×100 fine grid to obtain a reference (truth) solution for the multiscale problems. The coarse model is

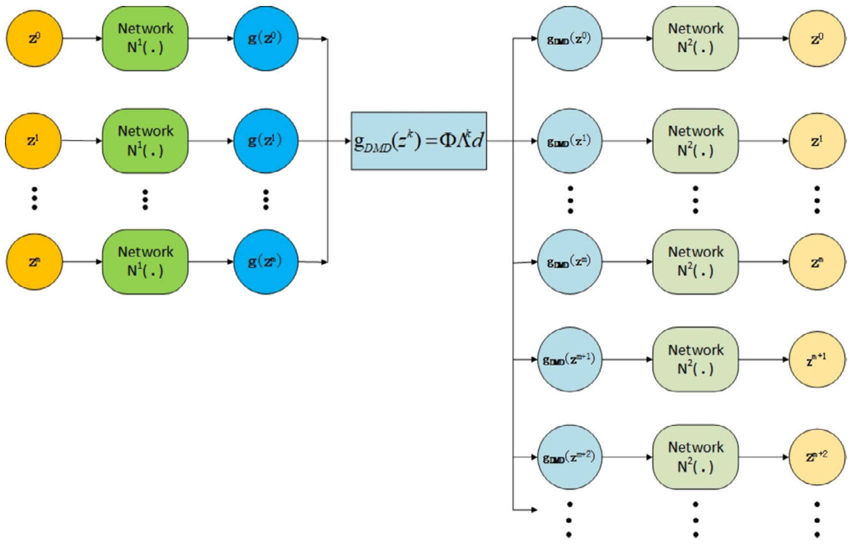

Fig. 4.2. By the first neural network, we can estimate $\boldsymbol{g}(\boldsymbol{z})$ at any time by $\boldsymbol{g}(\boldsymbol{z}^k) \approx \boldsymbol{\Phi}\boldsymbol{\Lambda}^k\boldsymbol{d}$. Then the reconstruction operator $\boldsymbol{h}$ is learned by the second network. We can estimate $\boldsymbol{z}$ at any time.

Algorithm 1 LIR-DMD.

```
Motivation: For a nonlinear dynamical system $z^{n+1} = f(z^n)$, we do not know the form of $f$, we can obtain a set of data through experiments. Using the data and Koopman operator, we want to predict the future state without iterations for a new trajectory.
Step 1: Learning Koopman invariant subspace and reconstruction of original state.
   Input: The measurement data $Z = \{z^{0,i}, z^{1,i}, \cdots, z^{m,i}\}_{i=1}^S$, here $S$ stands for $S$ trajectories.
   Output: Koopman invariant subspace $g(z) = \mathcal{N}^1(z;\theta^{*,1})$ and
           reconstruction operator $h = \mathcal{N}^2(y;\theta^{*,2})$
      1: These data $Z = \{z^{0,i}, z^{1,i}, \cdots, z^{m,i}\}_{i=1}^S$ are fed into the network.
      2: Find the optimal $\theta^* = (\theta^{*,1}, \theta^{*,2})$, $\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(g, h; Z)$,
         where $\mathcal{L}(g, h; Z)$ is defined by Eq. (4.10).
      3: Then $g(z) = \mathcal{N}^1(z;\theta^{*,1})$ and $h = \mathcal{N}^2(y;\theta^{*,2})$ .
Step 2: For a new trajectory, predicting the future state without iterations.
   Input: Snapshots$\{z^0, z^1, z^2, \cdots, z^m\} \not\subseteq Z$, Koopman invariant subspace
          $g(z) = \mathcal{N}^1(z;\theta^{*,1})$ and reconstruction operator $h = \mathcal{N}^2(y;\theta^{*,2})$
   Output: The future state $\{z^{m+1}, z^{m+2}, z^{m+3}, \cdots\}$
      1: The data $\{z^0, z^1, z^2, \cdots, z^m\}$ go through the first neural network
         $\mathcal{N}^1(z;\theta^{*,1})$, and we get $\{g(z^0), g(z^1), g(z^2), \cdots, g(z^m)\}$
      2: Then let $\mathbf{Y}_0 = [g(z^0) \ \cdots \ g(z^{m-1})]$, $\mathbf{Y}_1 = [g(z^1) \ \cdots \ g(z^m)]$
      3: Compute the SVD of $\mathbf{Y}_0$,$\mathbf{Y}_0 = U\Sigma V^T$
      4: Define $\widetilde{A} := U^T \mathbf{Y}_1 V \Sigma^{-1}$
      5: Compute eigenvalues and eigenvectors of $\widetilde{A}\widetilde{\Phi} = \widetilde{\Phi}\Lambda$, $\widetilde{W}^T\widetilde{A} = \Lambda\widetilde{W}^T$
      6: Set $\Phi = \mathbf{Y}_1 V \Sigma^{-1}\widetilde{\Phi}$, $W = U\widetilde{W}$
      7: Let $g_{\mathrm{DMD}}(z^k) = \Phi\Lambda^k d$, where $v(z^0) = W^T g(z^0)$,
         $k = 0, 1, 2, \cdots, m, m+1, \cdots$
      8: Go through the second neural network $\mathcal{N}^2(y;\theta^{*,2})$,
         and we get $z^k = h(g_{\mathrm{DMD}}(z^k))$, $k = 0, 1, 2, \cdots, m, m+1, \cdots$
```

defined on a 10×10 coarse grid by CEM-GMsFEM. The backward Euler method is used to define the discrete dynamical system. The training data are generated by using CEM-GMsFEM on the coarse gird.

For all numerical examples, the hyperparameters in LIR-DMD algorithm are set as follows:

$$\alpha_0 = 0.01, \quad \alpha_1 = 0.01, \quad \alpha_2 = 1.$$

5.1. *Fixed-point attractor*

In this subsection, we consider a fixed-point attractor problem to illustrate the performance of the proposed method. In particular, we want to confirm the local and global errors of the DMD prediction in Theorem 3.3. We know that it is usually difficult to find the nontrivial Koopman invariant subspaces for nonlinear problems. But the Koopman invariant subspace of the fixed point attractor can be solved directly. To this end, we consider the following nonlinear map in two variables $\boldsymbol{x}_n = [x_{1,n}, x_{2,n}]$ satisfying

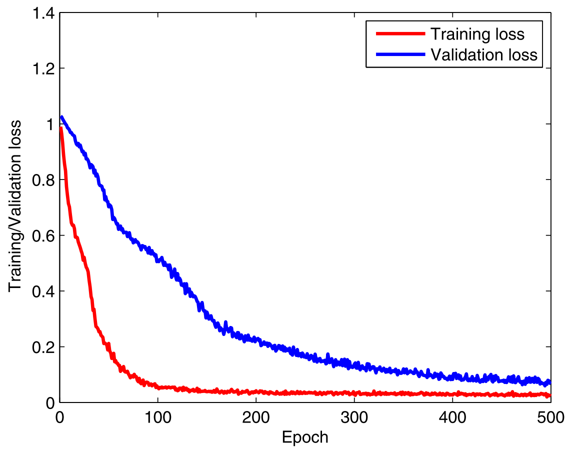

Fig. 5.3. Full connection Network training/validation losses over epochs for Fixed-point attractor. The training is performed over 500 epochs and the batch size is 15.

Table 1
Hyperparameter configuration of the network: fixed-point attractor.

	Layer structure	Activation function	Optimizer	Learning rate
Network 1	$2-10-10-3$	Relu	Adam	0.001
Network 2	$3-10-10-2$	Relu	Adam	0.001

$$\begin{cases} x_{1,n+1} = \lambda x_{1,n}, \\ x_{2,n+1} = \mu x_{2,n} + (\lambda^2 - \mu) x_{1,n}^2. \end{cases} \tag{5.12}$$

If λ, $\mu < 1$, then $\boldsymbol{x}_n$ tends to a stable equilibrium over time. Here the subscript n means the n-th moment. It is easy to know that $\{x_1, x_2, x_1^2\}$ spans a Koopman invariant subspace of system (5.12) because

$$\begin{bmatrix} x_{1,n+1} \\ x_{2,n+1} \\ x_{1,n+1}^2 \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \mu & \lambda^2 - \mu \\ 0 & 0 & \lambda^2 \end{bmatrix} \begin{bmatrix} x_{1,n} \\ x_{2,n} \\ x_{1,n}^2 \end{bmatrix}.$$

The eigenvalues of Koopman operator acting on the invariant subspace are λ, μ and λ^2. We consider the state of $\boldsymbol{x}$ at different moments with $\lambda = 0.9$, $\mu = 0.5$. In this example, the methods we present are based on the fact that we do not know the specific form of the dynamic model, and want to do prediction of the future with only the measurement data $\boldsymbol{x} = [x_1, x_2]$. We use LIR-DMD to learn the Koopman invariant subspace and the reconstruction of original state. In this method, we can also get the spectrum of Koopman operator. In order to train the neural network, we generate data based on different initial values, and divide these data into training set and test set. We randomly select 20% of the training data as the validation set, and each sample is i.i.d (independently identical distribution). Fig. 5.3 shows the performance of the training over 500 epochs of training. It can be seen from the curve that the loss function tends to be flat after 100 epochs. In Fig. 5.3, the verification loss and training loss have the same downward trend, which implies that there is no overfitting during the training process. The hyperparameter configuration of the network is shown in Table 1, where the layer structure $2-10-10-3$ represents the structure of the network as input layer, hidden layer, hidden layer, and output layer, and number of neurons in each layer is 2, 10, 10, 3, respectively. The optimizer we chose is Adam, and stochastic gradient descent algorithm (SGD) can also be used.

Table 2 lists the eigenvalues of the Koopman operator. From the table, we know that the eigenvalues of Koopman operator in the invariant subspace obtained in the training process are close to the real eigenvalues. In order to make the prediction of $\boldsymbol{x}$ by the proposed method, we used exact DMD and LIR-DMD to plot the curve of x_1 and x_2 over time in Fig. 5.4. We note that $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x}$ in exact DMD. We use the data of the first 20 time steps to learn the Koopman operator for short-term prediction. From the figure, we find: (1) x_1 and x_2 are close to the equilibrium point (0,0) after the 30-th time step; (2) the prediction accuracy of LIR-DMD is better than exact DMD. The reason may be because span$\{x_1, x_2\}$ is not a Koopman invariant subspace. In order to show that the local and global errors predicted by DMD strongly depends on the number m of snapshots and the number r of truncated eigenvalues, we list the local and global errors at the 60-th time moment in Table 3, which shows that e_{local}^{60} and e_{global}^{60} substantially decrease as m and r increase.

Table 2
The eigenvalues of Koopman operator acting on the invariant subspace $\{x_1, x_2, x_1^2\}$.

	true	LIR-DMD
λ	0.9	0.9005
μ	0.5	0.5492
λ^2	0.81	0.7992

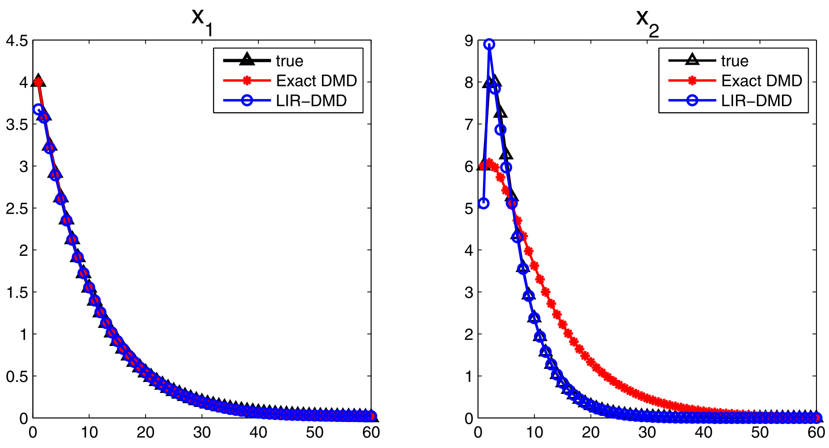

Fig. 5.4. Exact DMD and LIR-DMD simulate the trajectory of x_1 and x_2 over time.

Table 3
The local error and global error at $\boldsymbol{x}_{60}$.

m	r	e^{60}_{local}	e^{60}_{global}
20	3	0.0218	0.5301
30	3	0.0199	0.0265
40	3	0.0032	0.0148

m	r	e^{60}_{local}	e^{60}_{global}
30	1	0.1432	0.5858
30	2	0.0283	0.0749
30	3	0.0199	0.0265

5.2. Nonlinear multiscale parabolic problem

In this subsection, we present some data-driven methods to simulate the evolution of $\boldsymbol{z}$ with respect to time. Here we only have the data from the model, but do not know the equation of the model. We will use exact DMD, extended DMD and LIR-DMD to construct data-driven models and make comparison. In this example, the data is simulated from a type of Richards equation modeling unsaturated flows, where the nonlinear term in Eq. (2.3)

$$\kappa(x)b(u) = \kappa(x)\exp(u),$$

where the permeability field $\kappa(x)$ is depicted in Fig. 5.5. For the numerical simulation, we consider the homogeneous Dirichlet boundary condition, source term $h(x,t)$ and initial condition $u_0(x)$ are defined, respectively, by

$$h(x,t) = \exp(t)\sin(2\pi x_1)\sin(2\pi x_2)$$
$$u_0(x) = \sin(2\pi x_1)\sin(2\pi x_2),$$

where $x := (x_1, x_2) \in \Omega$.

First, we get the numerical data $\{\boldsymbol{z}^0, \boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^m\}$ over the time interval $t \in [0,1]$. We use neural network to learn a Koopman invariant subspace and the reconstructor $\boldsymbol{h}$ and make the prediction of state. Because we focus on the evolution of the coarse scale state $\boldsymbol{z}$ over time, we use the loss function defined in (4.10) to train the network. The data has the form of $\{\boldsymbol{z}^{0,i}, \boldsymbol{z}^{1,i}, \boldsymbol{z}^{2,i}, \cdots, \boldsymbol{z}^{m,i}\}$, $i = 1, 2, \ldots, N_s$, where the superscript i represents the i-th sample corresponding to the initial condition u_0^i. We select 80% of the data as training samples, and the remaining 20% as test samples. Then we randomly select 20% of the training data as the validation set such that we check the learning performance on the validation set to prevent overfitting. The resultant learning curve is shown in Fig. 5.6. We observe that the validation loss is close to the training loss. This implies that the model is well trained and the overfitting issue is not observed. The structure and hyperparameter configuration of the neural network are shown in Table 4.

We use exact DMD, extended DMD and LIR-DMD to construct data-driven models and predict the state. Fig. 5.7 depicts the relative error of the prediction over the time interval [1, 3] by the three data-driven methods. This figure clearly shows

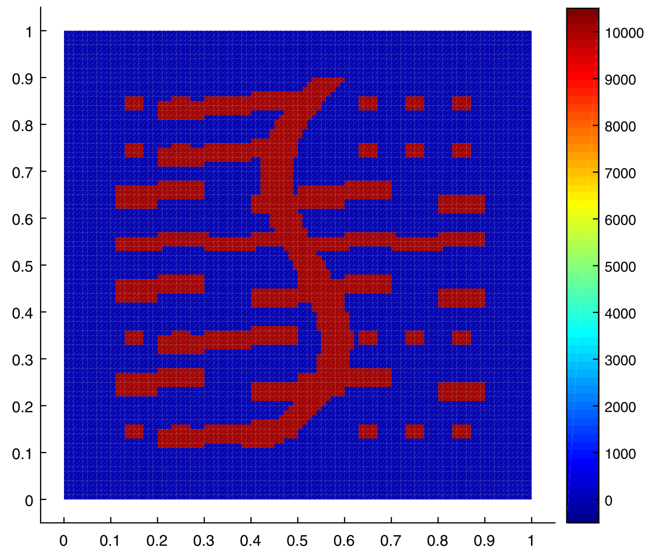

Fig. 5.5. Permeability field $\kappa(x)$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

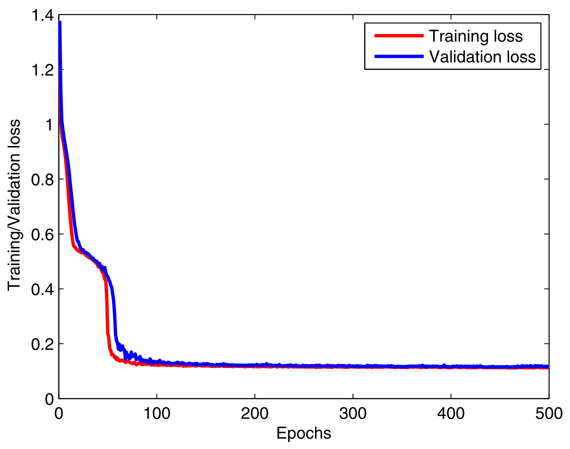

Fig. 5.6. Full connection network training/validation losses over epochs for nonlinear multiscale parabolic equation. 300 samples are used as training data. The training is performed over 500 epochs and the batch size is 15.

Table 4
Hyperparameter configuration of the Network: Nonlinear multiscale parabolic problem.

	Layer structure	Activation function	Optimizer	Learning rate
Network 1	200 − 200 − 200 − 300	Relu	Adam	0.001
Network 2	300 − 200 − 200 − 200	Relu	Adam	0.001

that the learning model by LIR-DMD gives much more accurate prediction than exact DMD and extended DMD. This figure also reveals that exact DMD using the observation function is $\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{z}$ is not able to make good prediction in long time. Extended DMD constructs a model by incorporating data and model's partial nonlinear information. For extended DMD, the nonlinear observation function is chosen to be $\boldsymbol{g}(\boldsymbol{z}) = [\boldsymbol{z}, \boldsymbol{N}(\boldsymbol{z})\boldsymbol{z}]^T$. By Fig. 5.7, we see that extended DMD is slightly better than exact DMD, but only makes a short-time prediction accurately. In order to learning the solution u on the fine grid, we use the loss function in (4.11) measured by the weighted l^2 norm to train the network. Here the data of snapshot is taken from the time interval $[0, 1]$. In Fig. 5.8, we compare the fine-scale solutions predicted by the three methods at $T = 1.2$. The figure shows that LIR-DMD predicts much more accurate solution than the other two methods. The LID-DMD solution well approximates the feature of the reference solution profile, but it is not a perfect match. This is because LIR-DMD uses the data to learn only an approximation of the Koopman invariant subspace $\boldsymbol{g}$ and of the reconstruction operator $\boldsymbol{h}$ through the neural network. It can be seen from Fig. 5.8 that in this example, exact DMD and extended DMD fail to achieve a good prediction of the future through the artificial selection of observation functions.

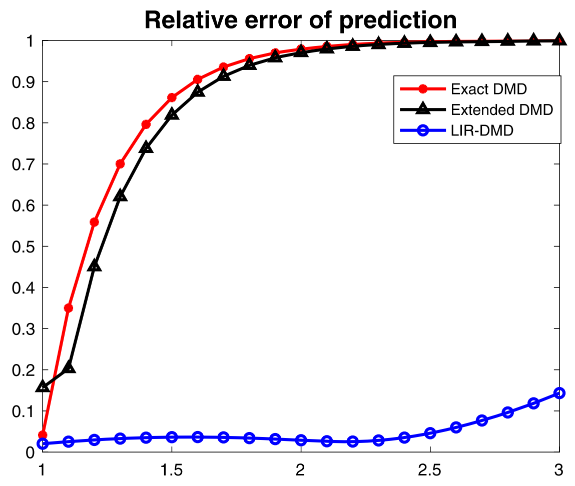

Fig. 5.7. The relative error of prediction by exact DMD, extended DMD and LIR-DMD. With the data of $t \in [0, 1]$ as snapshots, the state of $t \in [1, 3]$ is obtained by prediction.

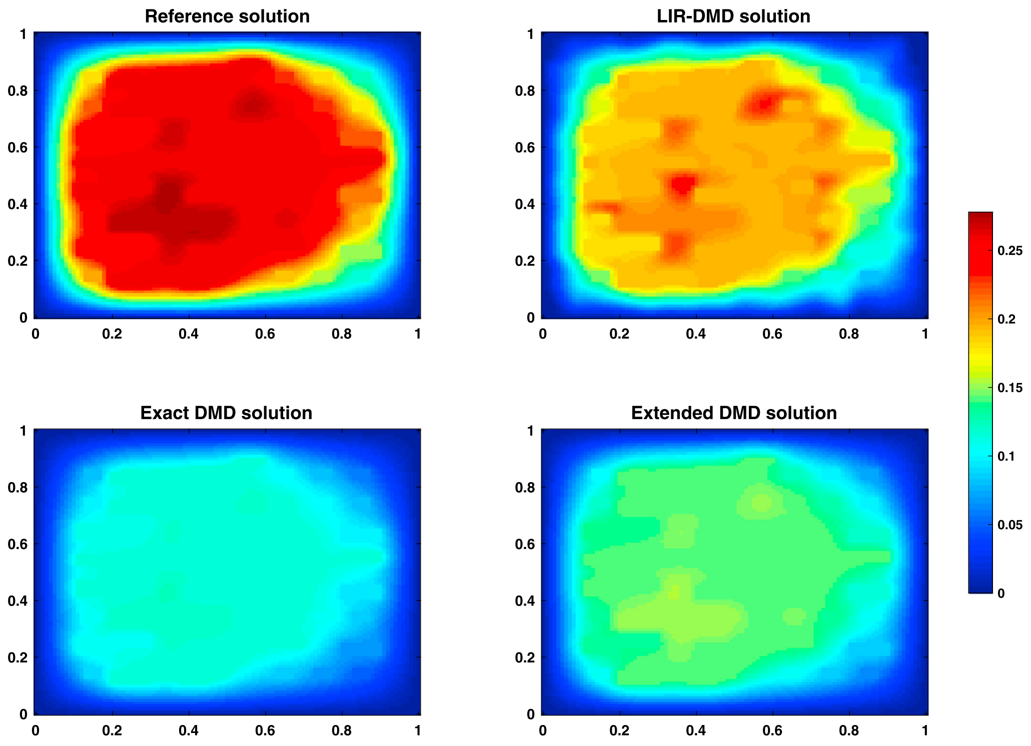

Fig. 5.8. Fine-scale solution profile $T = 1.2$. Left top: reference solution; Right top: solution predicted by LIR-DMD; Left bottom:solution predicted by exact DMD; Right bottom: solution predicted by extended DMD.

5.3. *The porous media equation*

In this subsection, we consider the porous media equation, whose nonlinearity is stronger than the model discussed in last section. We will utilize two kinds of methods: CEM-DMD and LIR-DMD and make comparison with numerical results. CEM-DMD is proposed in [16]. It uses observation data and model's nonlinear term and is a data-model driven method. But LIR-DMD only uses observation data and is a complete data-driven method. In CEM-DMD, we collect the data of the nonlinear term in the model and construct an approximation for the nonlinear term using exact DMD. Then we obtain a linear surrogate for the original nonlinear problem and use CEM-GMsFEM solve the linear surrogate to get a computational coarse model. CEM-DMD can significantly improve the computation efficiency. Although CEM-DMD incorporates nonlinear information of the model, it does not construct the Koopman invariant subspace accurately from data, and only uses the

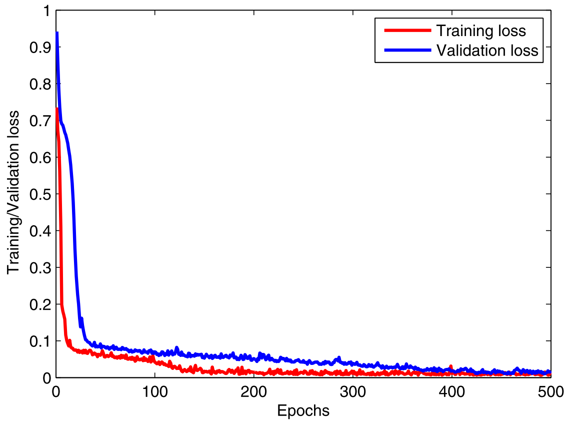

Fig. 5.9. Full connection network training/validation losses over epochs for porous medium equation. The training is performed over 500 epochs and the batch size is 20.

Table 5
Hyperparameter configuration of the network: PME.

	Layer structure	Activation function	Optimizer	Learning rate
Network 1	200 – 200 – 200 – 400	Relu	Adam	0.001
Network 2	400 – 200 – 200 – 200	Relu	Adam	0.001

identity function $\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{z}$ as the observation function. In contrast to CEM-DMD, LIR-DMD learns the Koopman invariant subspace from data and extracts essential coordinate information. For LIR-DMD, we use the loss function in (4.11) constructed by the weighted l^2 norm to learn the Koopman invariant subspace and the reconstruction operator. In order to improve the prediction accuracy of CEM-DMD, we use dynamically updated observed data. That is, when we predict the future state, we dynamically update the snapshot matrix based on the previously simulated data. We use CEM-DMD^U to denote the CEM-DMD with updating snapshots.

In this example, we consider the porous medium equation (PME), which can model different physical or engineering phenomena. For example, it can model the process involving fluid flow, heat transfer or spread of viscous fluids. One of the applications is to describe the flow of an isentropic gas through porous media [28]. In this example, we will consider the following PME:

$$\begin{cases} u_t - \operatorname{div}\big(\kappa(x)|u|^{m-1}\nabla u\big) = h(x,t) & \text{in } \Omega \times (0,T], \\ u = 0 & \text{on } \partial\Omega \times (0,T], \\ u(\cdot,0) = u_0(x) & \text{in } \Omega. \end{cases} \tag{5.13}$$

For numerical simulation, the source term $h(x,t)$ and initial condition $u_0(x)$ are set to be

$$h(x,t) = 1,$$
$$u_0(x) = \sin(2\pi x_1)\sin(2\pi x_2).$$

The training data are obtained by solving Eq. (5.13) with different initial conditions. 400 training samples are generated and the testing data contains 100 samples. We randomly choose 20% of the data from the training data as the validation set. We use minibatch training with batch size 20 and 500 epochs. The training error and validation error are shown in Fig. 5.9. From this figure, it can be seen that the network gives an accurate learning model and achieves good generalization. The hyperparameter configuration of the network is shown in Table 5.

Fig. 5.10 presents the prediction error of CEM-DMD, CEM-DMD^U and LIR-DMD. We use the snapshot of observation data from the time interval $[0, 0.4]$ to predict the state of $t \in [0.4, 1.2]$. The model established by CEM-DMD is solved iteratively to obtain the state at any time, while LIR-DMD only needs to use the observation function $\boldsymbol{g}$, the reconstructor $\boldsymbol{h}$ and Eq. (3.7) to predict the state at any time without iteration. From this figure, we find that the prediction relative error of LIR-DMD is much smaller than that of CEM-DMD. This shows that although CEM-DMD uses both data and model nonlinear information, it can not accurately capture the original nonlinear feature when the observation function is the identity function. Learning the invariant subspace through data can more accurately predict the evolution of state over time; When $t \in [0.4, 0.6]$, the prediction error of CEM-DMD^U is larger than that of LIR-DMD. This is mainly because the identity function selected by CEM-DMD^U can not constitute a Koopman invariant subspace. When $t \in [0.6, 1.2]$, the prediction error of CEM-DMD^U is smaller than that of LIR-DMD. This is because CEM-DMD^U incorporates dynamically updating data into the computational model.

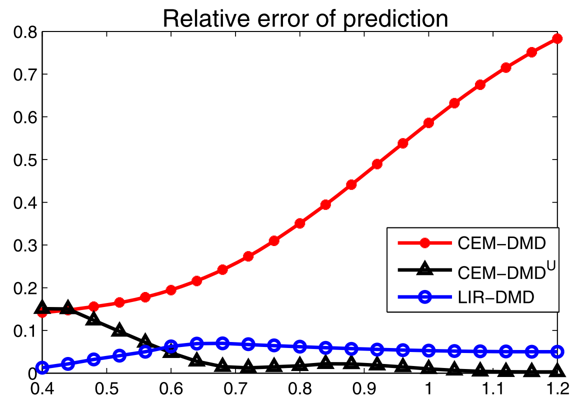

Fig. 5.10. The relative error of prediction by CEM-DMD, CEM-DMD^U and LIR-DMD. With the data of $t \in [0, 0.4]$ as snapshots, the state of $t \in [0.4, 1.2]$ is obtained by prediction.

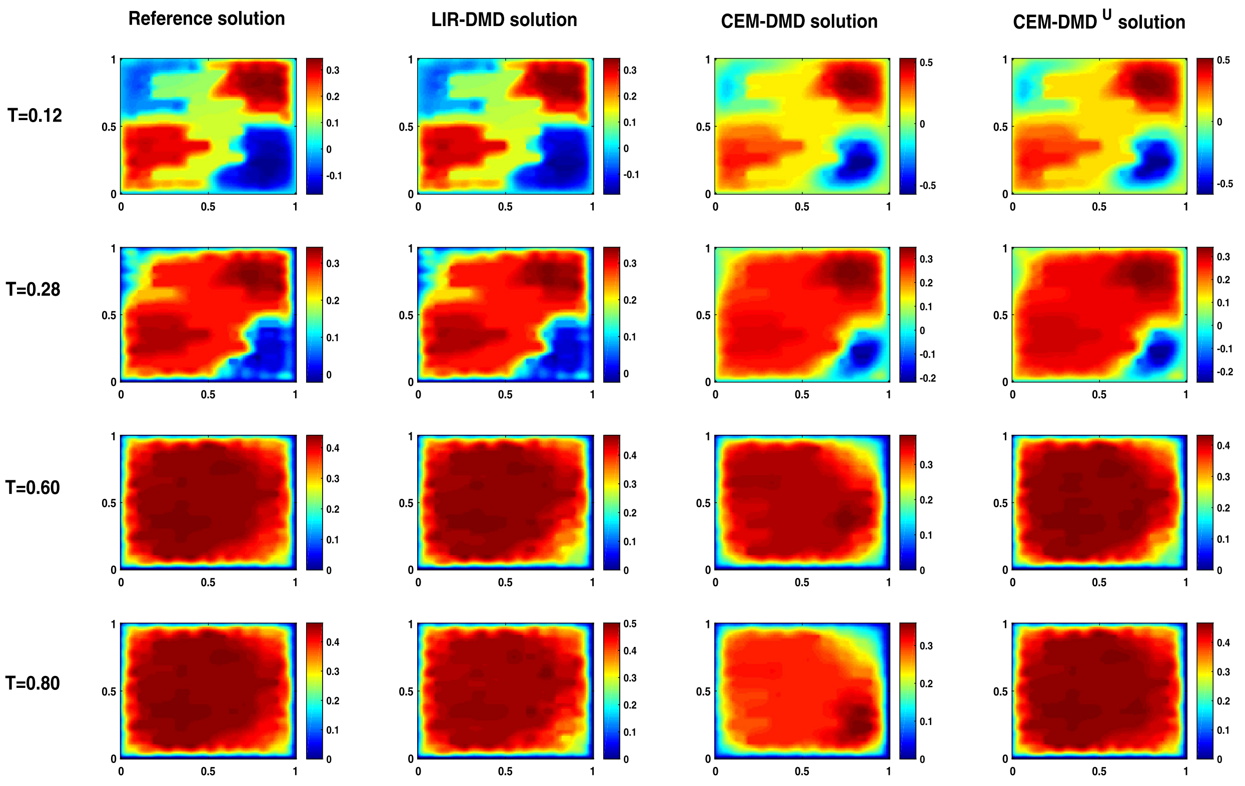

Fig. 5.11. Solution profiles for porous medium equation ($m = 3$). From the first column to the fourth column are reference solution, LIR-DMD solution, CEM-DMD solution and CEM-DMD^U solution. From the first row to the fourth row are the solutions at time $t = 0.12$, 0.28, 0.60 and 0.80, respectively.

In Fig. 5.11, we plot the reference solution, LIR-DMD solution, CEM-DMD solution and CEM-DMD^U solution at time $t = 0.12$, 0.28, 0.60, 0.80. Each row represents the solution at a fixed time by the four different methods, and each column represents the solution at the different time instants. From the figure, we can see that (1) the solution profiles of the equation change dramatically with respect to time; (2) there is no clear difference among the reference solution and LIR-DMD solution; (3) CEM-DMD solution is slightly different from the reference solution; (4) at $t = 0.12$ and $t = 0.28$, there is no obvious difference between the solutions of CEM-DMD^U and CEM-DMD, but both of them are distinct from the reference solution. This is mainly because we use the data from $t \in [0, 0.4]$ and do not update the data at these two moments; (5) at $t = 0.60$ and $t = 0.80$, the CEM-DMD^U solution is closer to the reference solution than the CEM-DMD solution, which is mainly due to using updated data for model construction.

6. Comments and conclusions

In this paper, we have presented a data-driven approach for nonlinear multiscale dynamical systems. DMD is a data-driven and equation-free method. The Koopman operator lifted nonlinear multiscale problems to finite-dimensional linear systems in invariant subspaces, and DMD can be used to predict the future of this linear system. It is very difficult to analytically find a set of observation functions that span the Koopman invariant subspace even if the nonlinear model is known. In this work, we used deep learning techniques to obtain Koopman invariant subspaces. The loss function of learning invariant subspace was established by measuring the distance between the observed data and the DMD approximation. The multiscale information was taken into account in the loss function by imposing the multiscale basis matrix. DMD can only get the future state of the observation functions. In order to map back to the original state space, we utilized the neural network to learn a reconstruction operator. This actually took restriction to the observation function in the invariant subspace. The proposed LIR-DMD can fulfill a relatively long-term prediction of the future state. The numerical results showed that the Koopman invariant subspace obtained by deep learning was effective for spatial multiscale prediction. Finally, we compared LIR-DMD with exact DMD and extended DMD by using a few numerical examples.

CRediT authorship contribution statement

The first author is Mengnan Li. She implemented the numerical examples and made substantial contribution to the draft. The corresponding author Lijian Jiang proposed the method, suggested numerical examples and made substantial contribution to the writing of the paper.

All authors of the paper have read and acknowledged the credit statement.

Declaration of competing interest

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

Acknowledgement

L. Jiang acknowledges the support of NSF of China 11871378 and support by Shanghai Science and Technology Committee 20JC1413500. We thank for the comments of referees to improve the paper.

References

[1] S. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing 5 (1993) 185–196.
[2] A. Abdulle, E. Weinan, B. Engquist, E. Vanden-Eijnden, The heterogeneous multiscale method, Acta Numer. 21 (2012) 1–87.
[3] A. Bourgeat, Homogenized behavior of two-phase flows in natured reservoirs with uniform fractures distribution, Comput. Methods Appl. Mech. Eng. 47 (1984) 205–216.
[4] S. Chaturantabut, D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (2010) 2737–2764.
[5] E. Chung, Y. Efendiev, T. Hou, Adaptive multiscale model reduction with generalized multiscale finite element methods, J. Comput. Phys. 320 (2016) 69–95.
[6] E. Chung, Y. Efendiev, W. Leung, Constraint energy minimizing generalized multiscale finite element method, Comput. Methods Appl. Mech. Eng. 339 (2018) 298–319.
[7] E. Chung, Y. Efendiev, W. Leung, M. Vasilyeva, Y. Wang, Non-local multi-continua upscaling for flows in heterogeneous fractured media, J. Comput. Phys. 372 (2018) 22–34.
[8] E. Chung, Y. Efendiev, W. Leung, M. Wheeler, Nonlinear nonlocal multicontinua upscaling framework and its applications, Int. J. Multiscale Comput. Eng. 16 (2018) 487–507.
[9] M. Drohmann, B. Haasdonk, M. Ohlberger, Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation, SIAM J. Sci. Comput. 34 (2012) A937–A969.
[10] B. Dykaar, P. Kitanidis, Determination of the effective hydraulic conductivity for heterogeneous porous media using a numerical spectral approach: 1. Method, Water Resour. Res. 28 (1992) 1155–1166.
[11] W. E, B. Engquist, Z. Huang, Heterogeneous multiscale method: a general methodology for multiscale modeling, Phys. Rev. B 67 (2003) 092101.
[12] Y. Efendiev, J. Galvis, T. Hou, Generalized multiscale finite element methods (GMsFEM), J. Comput. Phys. 251 (2013) 116–135.
[13] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (1991) 251–257.
[14] T. Hou, X. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media, J. Comput. Phys. 134 (1997) 169–189.
[15] L. Jiang, Y. Efendiev, G. Victor, Multiscale methods for parabolic equations with continuum spatial scales, Discrete Contin. Dyn. Syst., Ser. B 8 (2007) 833–859.
[16] L. Jiang, M. Li, Model reduction for nonlinear multiscale parabolic problems using dynamic mode decomposition, Int. J. Numer. Methods Eng. 121 (2020) 3680–3701.
[17] P. Jenny, S. Lee, H. Tchelepi, Adaptive multiscale finite-volume method for multiphase flow and transport in porous media, Multiscale Model. Simul. 3 (2005) 50–64.
[18] B. Koopman, Hamiltonian systems and transformation in Hilbert space, Proc. Natl. Acad. Sci. USA 17 (1931) 315–318.
[19] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.
[20] S. Klus, P. Koltai, C. Schütte, On the numerical approximation of the Perron-Frobenius and Koopman operator, J. Comput. Dyn. 3 (2016) 51–79.
[21] M. Korda, I. Mezić, On convergence of extended dynamic mode decomposition to the Koopman operator, J. Nonlinear Sci. 28 (2018) 687–710.
[22] B. Koopman, J. Neumann, Dynamical systems of continuous spectra, Proc. Natl. Acad. Sci. USA 18 (1932) 255–263.

[23] J. Kutz, J. Proctor, S. Brunton, Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems, Complexity 12 (2018) 6010634.
[24] I. Kevrekidis, C. Rowley, M. Williams, A kernel-based method for data-driven Koopman spectral analysis, J. Comput. Dyn. 2 (2015) 247–265.
[25] I. Lunati, P. Jenny, Multiscale finite-volume method for compressible multiphase flow in porous media, J. Comput. Phys. 216 (2006) 616–636.
[26] B. Lusch, J. Kutz, S. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. 9 (2018) 1–10.
[27] I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dyn. 41 (2005) 309–325.
[28] M. Muskat, The Flow of Homogeneous Fluids Through Porous Media, McGraw-Hill, New York, 1937.
[29] C. Rowley, I. Mezić, S. Bagheri, P. Schlatter, D. Henningson, Spectral analysis of nonlinear flows, J. Fluid Mech. 641 (2009) 115–127.
[30] P. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28.
[31] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, Q. Appl. Math. 45 (1987) 561–571.
[32] A. Tihonov, Solution of incorrectly formulated problems and the regularization method, Sov. Math. 4 (1963) 1035–1038.
[33] N. Takeishi, Y. Kawahara, T. Yairi, Learning Koopman invariant subspaces for dynamic mode decomposition, Adv. Neural Inf. Process. Syst. (2017) 1130–1140.
[34] J. Tu, C. Rowley, D. Luchtenberg, S. Brunton, J. Kutz, On dynamic mode decomposition: theory and applications, J. Comput. Dyn. 1 (2014) 391–421.
[35] M. Williams, I. Kevrekidis, C. Rowley, A data-driven approximation of the Koopman operator: extending dynamic mode decomposition, J. Nonlinear Sci. 25 (2015) 1307–1346.
[36] E. Yeung, S. Kundu, N. Hodas, Learning deep neural network representations for Koopman operators of nonlinear dynamical systems, in: 2019 American Control Conference, IEEE, 2019, pp. 4821–4839.