Journal of Computational Physics 514 (2024) 113231

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

An augmented subspace based adaptive proper orthogonal decomposition method for time dependent partial differential equations ☆

Xiaoying Dai [a,b,*], Miao Hu [a,b], Jack Xin [c], Aihui Zhou [a,b]

[a] LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
[c] Department of Mathematics, University of California at Irvine, Irvine, CA 92697, USA

ARTICLE INFO

Keywords:
Proper orthogonal decomposition
Adaptive
Augmented subspace
Galerkin projection
Error indicator

ABSTRACT

In this paper, we propose an augmented subspace based adaptive proper orthogonal decomposition (POD) method for solving the time dependent partial differential equations. We use the difference between the approximation obtained in the augmented subspace and that obtained in the original POD subspace to construct an error indicator, by which we obtain a general framework for augmented subspace based adaptive POD method. We then provide two strategies to construct the augmented subspaces, the residual type augmented subspace and the coarse-grid approximation type augmented subspace. We apply our new methods to two typical 3D advection-diffusion equations with the advection being the Kolmogorov flow and the ABC flow. Numerical results show that both the residual type augmented subspace based adaptive POD method and the coarse-grid approximation type augmented subspace based adaptive POD method are more efficient than the existing adaptive POD methods, especially for the advection dominated models.

1. Introduction

Time dependent partial differential equations arise in many important fields, e.g., the seawater intrusion [2], the heat transfer [11], semiconductor devices [41] and fluid equations [25,3,42,62,59,38]. The study about the numerical methods for the time dependent partial differential equations is an important and attractive research topic. There are some classical numerical discretization methods for the spatial discretization of the time dependent partial differential equations, e.g., the finite element method [8], the finite difference method [54] and the plane wave method [30]. Usually, the semi-discretized systems resulted from applying these classical spatial discretization methods for a time dependent partial differential equations are of huge dimensional. If we use these classical spatial discretization methods at each time interval, the computational cost will be very expensive, especially for complex systems.

Therefore, some efficient and accurate model order reduction methods have been proposed to reduce the dimension of discretized system and then the computational costs [4,49,13,40,7]. The basic idea for the model order reduction method is to project the

☆ This work was supported by the National Key R&D Program of China under grants 2019YFA0709600 and 2019YFA0709601, the National Natural Science Foundation of China under grants 92270206 and 11671389, and the NSF grants DMS-1952644, DMS-2151235 and DMS-2309520.

* Corresponding author at: LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.
E-mail addresses: daixy@lsec.cc.ac.cn (X. Dai), humiao@lsec.cc.ac.cn (M. Hu), jack.xin@uci.edu (J. Xin), azhou@lsec.cc.ac.cn (A. Zhou).

https://doi.org/10.1016/j.jcp.2024.113231
Received 16 February 2023; Received in revised form 3 June 2024; Accepted 23 June 2024

Available online 1 July 2024

continuous system onto a low-dimensional approximation subspace whose dimension is significantly less than that of the classical discretization space. Proper orthogonal decomposition (POD) is a commonly used model order reduction technique [24,33,48,39,21]. The typical steps for using a POD method to solve a time dependent partial differential equation are as follows. First, choose some classical discretization method to discretize the continuous equation in some time interval, and solve the resulted high-dimensional discretized systems to get a set of snapshots. Then, construct the POD modes by minimizing the error between these modes and the snapshots, which is equivalent to solving an n_s dimensional eigenvalue problem [53,25,51]. Here n_s is the number of snapshots. At last, project the original system onto the subspace spanned by the POD modes, which is also called POD subspace, and solve the resulted discretized systems in the following time intervals. In the actual calculation, singular value decomposition (SVD) is usually used to obtain the POD modes from the snapshots [37,48,10]. By choosing the snapshots properly, the number of the POD modes will usually be of magnitude smaller than the degrees of freedom resulted by the classical spacial discretization methods.

There are many applications of the POD method in scientific and engineering computing. For instance, in [17], a group proper orthogonal decomposition (GPOD) method was introduced to simulate the nonlinear Burgers equation. In [35], the POD method was used to solve the time-domain Maxwell's equations. In [34], a reduced-order finite element formulation based on POD method was established for the Allen-Cahn equation. Other applications include studies of turbulence [19,55], process identification [27,28] and control in chemical engineering [44,61], etc. We refer to [57,58] for more introduction to the POD method.

There are also some existing works on the error analysis for the POD method. For example, Kunisch and Volkwein estimated the error of the POD approximation for linear and nonlinear evolution equations in [31,32]. In [22], Xin et al. analyzed the convergence of the POD approximation for viscous G-equation by decomposing the data into a mean-free part and a mean part. More works about the numerical analysis for the POD method can be referred to [26,36,12,29] and references therein.

The classical POD method for time dependent partial differential equations only uses the snapshots obtained in the early time interval to construct the POD modes. Once the POD modes are obtained, they will be fixed and not updated during the time evolution any more. However, the solution of the system may change a lot over time. Therefore, if the POD modes are not updated as time evolution, the approximation error obtained by the POD method may become larger and larger.

In order to improve the accuracy of the POD method in the whole time interval, some adaptive POD methods which update the POD modes as time evolution have been introduced in recent years [14,15,50,56,45]. In [14,50,56], the authors constructed some residual type error estimators, based on which some residual based adaptive POD methods are proposed for simulation of time dependent problems. In [15], the authors proposed a two-grid based adaptive POD method. For this method, they first constructed two finite element spaces, a coarse finite element space and a fine finite element space, and then used the error obtained in the coarse finite element space to construct the error indicator, by which people then see whether it is necessary to update the POD subspace in the fine mesh or not. We refer to [15] for more details about the two-grid based adaptive POD method.

In this paper, we propose a new approach for developing some adaptive POD methods for the time dependent partial differential equations. The main idea of our approach is to use some auxiliary modes to augment the current POD subspace to build an augmented subspace. We then use the difference between the approximation obtained in the augmented subspace and that obtained in original POD subspace to develop an error indicator. Using this idea, we obtain a general framework for augmented subspace based adaptive POD method. We then provide two specific strategies to obtain the auxiliary mode, one is using the residual corresponding to the POD approximation as the auxiliary mode, the other is using a coarse-grid approximation as the auxiliary mode. Using the residual to enrich the current POD subspace is inspired by [5]. Using the coarse-grid approximation to augment the subspace is inspired by [15]. In [15], the authors used the error obtained in a coarse finite element space to act as the error indicator, while in this paper, we use the approximation in the coarse finite element space to augment the POD subspace, and then develop an error indicator. Besides, following the idea of [50], we introduce a weight to each mode when updating the POD subspace to improve the performance of the method.

The rest of this paper is organized as follows. In Section 2, we give some preliminaries, including the general framework of the adaptive POD method, the comparison of computational complexity for different POD type methods and the simple descriptions of two typical adaptive POD methods. In Section 3, we propose a general framework for the augmented subspace based adaptive POD method, and then provide two specific strategies to construct the augmented subspace, the residual type augmented subspace and the coarse-grid approximation type augmented subspace. Besides, we also introduce a weighting strategy to update the POD modes in Section 3. In Section 4, we apply the two-grid based adaptive POD method and our augmented subspace based adaptive POD methods to the simulation of some typical time dependent partial differential equations, i.e., the advection-diffusion equation with three dimensional velocity field, including both the Kolmogorov flow and the ABC flow. The numerical results show the accuracy and efficiency of our new methods. In Section 5, we give some concluding remarks. Finally, we provide some additional numerical results for the advection dominated models with different coarse meshes and different error indicator thresholds in Appendix A.

2. Preliminaries

We first recall some definition and notation. We shall use the standard notation for Sobolev spaces and their associated norms and seminorms; see, e.g., [1]. Let V be a Banach space with norm $\|*\|_V$ and $L^p(0,T;V)$ be a Banach space equipped with the norm

$$\|u\|_{L^p(0,T;V)} = \left(\int_0^T \|u(t)\|_V^p \mathrm{d}t \right)^{\frac{1}{p}}, 1 \le p < \infty.$$

We call $u \in C(0,T;V)$ if

$$\|u\|_{C(0,T;V)} = \max_{t\in[0,T]} \|u(t)\|_V < \infty.$$

In this paper, vectors and matrices will be denoted by bold letters.

We consider the following general time dependent partial differential equations:

$$\begin{cases} u_t - \epsilon\Delta u + \mathbf{B}(x,y,z,t)\cdot\nabla u + c(x,y,z,t)u = f(x,y,z,t), \quad \text{in } \Omega\times(0,T] \\ u(x,y,z,0) = h(x,y,z), \\ u(x+\kappa,y,z,t) = u(x,y+\kappa,z,t) = u(x,y,z+\kappa,t) = u(x,y,z,t), \end{cases} \tag{1}$$

where $\Omega = [0,\kappa]^3$, $f \in L^2\left(0,T;L^2(\Omega)\right)$, $c \in C\left(0,T;L^\infty(\Omega)\right)$, $\mathbf{B} \in C\left(0,T;W^{1,\infty}(\Omega)^3\right)$, $h \in L^2(\Omega)$ and ϵ is a constant. We take an inner product with v in (1) and define a bilinear form to simplify the variational form

$$a(t;u,v) = \epsilon(\nabla u,\nabla v) - \epsilon\int_{\partial\Omega}\frac{\partial u}{\partial n}v d\sigma + (\mathbf{B}\cdot\nabla u, v) + (cu,v), \forall u,v \in H^1(\Omega),$$

where $(\cdot,\cdot)$ stands for the inner product in $L^2(\Omega)$. We obtain the variational form of the Eq. (1) as follows: find $u \in L^2(0,T;V)$, $u_t \in L^2(0,T;V^*)$ such that

$$\left(u_t, v\right) + a(t;u,v) = (f,v), \forall v \in V, \tag{2}$$

where

$$V = \left\{v \in H^1(\Omega) : v|_{x=0} = v|_{x=\kappa}, v|_{y=0} = v|_{y=\kappa}, v|_{z=0} = v|_{z=\kappa}\right\},$$

and V^* is the dual space of V.

2.1. Standard discretization

We first consider the standard temporal discretization of (2). There are many existing temporal discretization methods, such as Euler method and implicit Euler method [23], which can be used to discretize (2). Here, we choose the implicit Euler method. We first divide the time interval into $N \in \mathbb{N}$ subintervals with equal length $\delta t = T/N$, and let $u^k(x,y,z)$ be the approximation of $u(x,y,z,t_k)$, $f_k(x,y,z) = f(x,y,z,t_k)$, where $t_k = k\cdot\delta t$, for $k \in \{0,1,\cdots,N\}$. Then we can get the semi-discretization scheme of (2) as follows:

$$\left(\frac{u^k(x,y,z) - u^{k-1}(x,y,z)}{\delta t}, v\right) + a\left(t_k;u^k(x,y,z),v\right) = \left(f_k(x,y,z),v\right), \quad v \in V. \tag{3}$$

We then consider the classical spatial discretization of (3). Here, we choose the finite element method to discretize (3). Let $\mathcal{T}_h$ be a shape regular family of nested conforming mesh over Ω with size h: there exists a constant γ^* such that

$$\frac{h_\tau}{\rho_\tau} \le \gamma^*, \quad \forall\, \tau \in \mathcal{T}_h, \tag{4}$$

where h_τ is the diameter of τ for each $\tau \in \mathcal{T}_h$, ρ_τ is the diameter of the biggest ball contained in τ, and $h = \max\left\{h_\tau : \tau \in \mathcal{T}_h\right\}$. Let V_h be a subspace of continuous functions on Ω such that

$$V_h = \left\{v_h : v_h|_\tau \in \mathbb{P}_\tau, \forall\, \tau \in \mathcal{T}_h \text{ and } v_h \in C^0(\overline{\Omega})\right\},$$

where $\mathbb{P}_\tau$ is a set of polynomials on element τ. Let $\left\{\phi_{h,1},\phi_{h,2},\cdots,\phi_{h,N_g}\right\}$ be a basis of V_h with N_g being the degrees of freedom. Denote

$$\mathbf{\Phi}_h := \left(\phi_{h,1},\phi_{h,2},\cdots,\phi_{h,N_g}\right).$$

The standard finite element discretization of (3) can be formulated as follows: find $u_h^k(x,y,z) \in V_h$, such that

$$\left(\frac{u_h^k(x,y,z) - u_h^{k-1}(x,y,z)}{\delta t}, v_h\right) + a\left(t_k;u_h^k(x,y,z),v_h\right) = \left(f_k(x,y,z),v_h\right), \quad v_h \in V_h. \tag{5}$$

Note that $u_h^k(x,y,z)$ can be expressed as

$$u_h^k(x,y,z) = \sum_{i=1}^{N_g} u_{h,i}^k \phi_{h,i}(x,y,z). \tag{6}$$

Inserting (6) into (5), and setting $v_h = \phi_{h,j}, j = 1,2,\cdots,N_g$, respectively, we have

Algorithm 1 `POD_Mode` $(\mathbf{U}_h, \gamma, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$.

Input: $\mathbf{U}_h, \gamma, \mathbf{\Phi}_h = \left(\phi_{h,1}, \phi_{h,2}, \cdots, \phi_{h,n}\right)$;
Output: m and POD modes $\mathbf{\Psi}_h = \left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right)$;
1: Perform SVD on $\mathbf{U}_h$ to obtain $\mathbf{U}_h = \mathbf{RSV^T}$, where $\mathbf{S} = \text{diag}\left\{\sigma_1, \sigma_2, \cdots, \sigma_r\right\}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$;
2: Set $m = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{i,i} > \gamma \cdot \text{Trace}(\mathbf{S})\right\}$;
3: $\left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right) = \mathbf{\Phi}_h \mathbf{R}[:, 1:m]$;

$$\left(\sum_{i=1}^{N_g} u_{h,i}^k \phi_{h,i} - \sum_{i=1}^{N_g} u_{h,i}^{k-1} \phi_{h,i}, \phi_{h,j}\right) + \delta t \cdot a\left(t_k; \sum_{i=1}^{N_g} u_{h,i}^k \phi_{h,i}, \phi_{h,j}\right) = \delta t \cdot \left(f_k, \phi_{h,j}\right). \tag{7}$$

Define

$$\mathbf{A}_{h,ij}^k = \left(\phi_{h,j}, \phi_{h,i}\right) + \delta t \cdot a\left(t_k; \phi_{h,j}, \phi_{h,i}\right), \quad \mathbf{u}_h^k = \left(u_{h,1}^k, u_{h,2}^k, \cdots, u_{h,N_g}^k\right)^T,$$
$$\mathbf{b}_h^k = \delta t \cdot \left(\left(f_k, \phi_{h,1}\right), \cdots, \left(f_k, \phi_{h,N_g}\right)\right)^T, \quad \mathbf{C}_{h,ij} = \left(\phi_{h,j}, \phi_{h,i}\right).$$

Then (7) can be rewritten as the following algebraic form

$$\mathbf{A}_h^k \mathbf{u}_h^k = \mathbf{b}_h^k + \mathbf{C}_h \mathbf{u}_h^{k-1}, \tag{8}$$

where $\mathbf{A}_h^k = \left(\mathbf{A}_{h,ij}^k\right)_{N_g \times N_g}$ and $\mathbf{C}_h = \left(\mathbf{C}_{h,ij}\right)_{N_g \times N_g}$.

2.2. Adaptive POD method

In this subsection, we first recall the general procedure for getting a POD reduced order model of a time dependent partial differential equation. First, we choose some classical discretization method to discretize (3) in some time interval. Here we discretize (3) in the finite element space V_h for $t \in [0, T_0]$, and solve the resulted high dimensional discretized systems, and then collect the numerical solution at different times t_0, $t_{\delta M}$, $\cdots$, $t_{n_s \cdot \delta M}$ to obtain the snapshot matrix $\mathbf{U}_h$. Here, δM is an integer parameter and $n_s = \lfloor \frac{T_0}{\delta t \cdot \delta M} \rfloor$, where $\lfloor * \rfloor$ means the round down. Then, we perform SVD on $\mathbf{U}_h$, and obtain $\mathbf{U}_h = \mathbf{RSV^T}$. Note that the diagonal elements in $\mathbf{S}$ are arranged from largest to smallest, we set the number of POD modes by

$$m = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{i,i} > \gamma_1 \cdot \text{Trace}(\mathbf{S})\right\}, \tag{9}$$

where γ_1 is a given parameter, and set $\widetilde{\mathbf{R}} = \mathbf{R}[:, 1:m]$. Then the POD modes are

$$\mathbf{\Psi}_h = \left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right) := \mathbf{\Phi}_h \widetilde{\mathbf{R}}.$$

The process for constructing POD modes can be summarized as a routine `POD_Mode` $(\mathbf{U}_h, \gamma_1, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$ in Algorithm 1, see also [15]. At last, we project (3) onto the POD subspace $V_{h,\text{POD}} = \text{span}\left\{\psi_{h,1}, \cdots, \psi_{h,m}\right\}$ when $t > T_0$. The POD approximation $u_{h,\text{POD}}^k(x, y, z)$ can be expressed as

$$u_{h,\text{POD}}^k(x, y, z) = \sum_{i=1}^{m} \alpha_{h,i}^k \psi_{h,i}(x, y, z). \tag{10}$$

Inserting (10) into (3), and setting $v = \psi_{h,j}, j = 1, 2, \cdots, m$, respectively, we obtain

$$\left(\sum_{i=1}^{m} \alpha_{h,i}^k \psi_{h,i} - \sum_{i=1}^{m} \alpha_{h,i}^{k-1} \psi_{h,i}, \psi_{h,j}\right) + \delta t \cdot a\left(t_k; \sum_{i=1}^{m} \alpha_{h,i}^k \psi_{h,i}, \psi_{h,j}\right) = \delta t \cdot \left(f_k, \psi_{h,j}\right). \tag{11}$$

Define

$$\bar{\mathbf{A}}_{h,ij}^k = \left(\psi_{h,j}, \psi_{h,i}\right) + \delta t \cdot a\left(t_k; \psi_{h,j}, \psi_{h,i}\right), \quad \mathbf{u}_{h,\text{POD}}^k = \left(\alpha_{h,1}^k, \alpha_{h,2}^k, \cdots, \alpha_{h,m}^k\right)^T,$$
$$\bar{\mathbf{b}}_h^k = \delta t \cdot \left(\left(f_k, \psi_{h,1}\right), \cdots, \left(f_k, \psi_{h,m}\right)\right)^T, \quad \bar{\mathbf{C}}_{h,ij} = \left(\psi_{h,j}, \psi_{h,i}\right).$$

Then (11) can be rewritten as the following algebraic form

$$\bar{\mathbf{A}}_h^k \mathbf{u}_{h,\text{POD}}^k = \bar{\mathbf{b}}_h^k + \bar{\mathbf{C}}_h \mathbf{u}_{h,\text{POD}}^{k-1}, \tag{12}$$

where $\bar{\mathbf{A}}_h^k = \left(\bar{\mathbf{A}}_{h,ij}^k\right)_{m \times m}$ and $\bar{\mathbf{C}}_h = \left(\bar{\mathbf{C}}_{h,ij}\right)_{m \times m}$.

From the expression of the POD modes, i.e., $\mathbf{\Psi}_h = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$, the above equation can also be written as

$$\widetilde{\mathbf{R}}^T \mathbf{A}_h^k \widetilde{\mathbf{R}} \mathbf{u}_{h,\mathrm{POD}}^k = \widetilde{\mathbf{R}}^T \mathbf{b}_h^k + \widetilde{\mathbf{R}}^T \mathbf{C}_h \widetilde{\mathbf{R}} \mathbf{u}_{h,\mathrm{POD}}^{k-1},$$

where $\mathbf{A}_h^k = \left(\mathbf{A}_{h,ij}^k\right)_{N_g \times N_g}$ and $\mathbf{C}_h = \left(\mathbf{C}_{h,ij}\right)_{N_g \times N_g}$.

As we mentioned before, the classical POD method only uses the snapshots obtained in time interval $[0, T_0]$ to construct the POD modes. Once the POD modes are obtained, they will not be updated during the time evolution. However, the solution may change a lot over time. In order to improve the accuracy of the POD method in the whole time interval, more and more researchers have paid attention to the study of adaptive POD method in recent years [14,15,50,56,45]. Here, we provide a brief introduction.

Motivated by the procedure for adaptive finite element method [16], in [15], the authors summarized the procedure of adaptive POD method as a loop constructed by the following four steps:

1. **Solve**: Solve the Eq. (3) in the POD subspace span $\left\{\psi_{h,1}, \cdots, \psi_{h,m}\right\}$.
2. **Estimate**: Construct an error indicator η_k at time instant t_k to estimate the error of the approximation obtained in current POD subspace.
3. **Mark**: Mark the time instant t_k when the POD subspace is needed to be updated.
4. **Update**: Update the POD subspace at the marked time instant.

The step **Solve** is just the procedure for obtaining the approximations by the classical POD method we introduced above. The step **Estimate** is crucial for an adaptive POD method, which determines the efficiency and accuracy of the method. The step **Mark** picks out the time instant when the POD modes are needed to be updated. For the step **Update**, it is worthy of noting that, if the time instant $t = q \cdot \delta t$ is marked, we will go back to the previous time instant $t_1 = (q-1) \cdot \delta t$ to restart the collection of the approximations in the finite element space every δM time-step and get the snapshots matrix $\mathbf{W}_{h,1}$. In order to obtain the new POD modes, we perform SVD on $\mathbf{W}_{h,1}$ to obtain $\mathbf{W}_{h,1} = \mathbf{R}_1 \mathbf{S}_1 \mathbf{V}_1^T$. Then, we obtain the number of POD modes m_1 by (9) but with a different parameter γ_2. In order to keep most of the POD modes, we perform SVD on $\mathbf{W}_{h,2} = [\mathbf{R}_1[:, 1:m_1], \widetilde{\mathbf{R}}]$, and get $\mathbf{W}_{h,2} = \mathbf{R}_2 \mathbf{S}_2 \mathbf{V}_2^T$. Then, we get the number of POD modes m by (9) but with a parameter γ_3. Finally we set $\widetilde{\mathbf{R}} = \mathbf{R}_2[:, 1:m]$ and obtain the updated POD modes by

$$\mathbf{\Psi}_h = \left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right) = \mathbf{\Phi}_h \widetilde{\mathbf{R}}.$$

For the convenience of the following discussion, we summarize the process for the step **Update** as routine `Update_POD_Mode` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$ in Algorithm 2, which is first introduced in [15].

Algorithm 2 `Update_POD_Mode` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$.

Input: $\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h = (\phi_{h,1}, \phi_{h,2}, \cdots, \phi_{h,n})$, m and m old POD modes $\mathbf{\Psi}_h, \mathbf{\Psi}_h = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$.
Output: new m and new m POD modes $\left\{\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right\}$.

1: Perform SVD on $\mathbf{W}_{h,1}$ to obtain $\mathbf{W}_{h,1} = \mathbf{R}_1 \mathbf{S}_1 \mathbf{V}_1^T$, where $\mathbf{S_1} = \mathrm{diag}\left\{\sigma_{1,1}, \sigma_{1,2}, \cdots, \sigma_{1,r_1}\right\}$ with $\sigma_{1,1} \geq \sigma_{1,2} \geq \cdots \geq \sigma_{1,r_1} > 0$;
2: Set $m_1 = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{1,ii} > \gamma_2 \cdot \mathrm{Trace}(\mathbf{S}_1)\right\}$;
3: Perform SVD on $\mathbf{W}_{h,2} = [\mathbf{R}_1[:, 1:m_1], \widetilde{\mathbf{R}}]$, and obtain $\mathbf{W}_{h,2} = \mathbf{R}_2 \mathbf{S}_2 \mathbf{V}_2^T$, where $\mathbf{S_2} = \mathrm{diag}\left\{\sigma_{2,1}, \sigma_{2,2}, \cdots, \sigma_{2,r_2}\right\}$ with $\sigma_{2,1} \geq \sigma_{2,2} \geq \cdots \geq \sigma_{2,r_2} > 0$;
4: Set $m = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{2,ii} > \gamma_3 \cdot \mathrm{Trace}(\mathbf{S}_2)\right\}$, and $\widetilde{\mathbf{R}} = \mathbf{R}_2[:, 1:m]$;
5: $(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}) = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$;

We then obtain the framework of the adaptive POD method as Algorithm 3.

Algorithm 3 Framework of the adaptive POD method.

Given T_0, T, δM, δT, δt, γ_1, γ_2, γ_3, η_0 and the mesh $\mathcal{T}_h$;
Discretize (3) in the standard finite element space V_h on interval $[0, T_0]$ and obtain the snapshots matrix $\mathbf{U}_h$;
Construct POD modes $\mathbf{\Psi}_h$ by `POD_Mode` $(\mathbf{U}_h, \gamma_1, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
$t = T_0$, $k = \frac{T_0}{\delta t}$;
while $t \leq T$ **do**
    $t = t + \delta t$, k = k + 1;
    Discretize (3) in the POD subspace $V_{h,\mathrm{POD}} = \mathrm{span}\left\{\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right\}$, and obtain the POD approximations $\mathbf{u}_{h,\mathrm{POD}}^k$;
    Compute the error indicator η_k by some strategy;
    if $\eta_k > \eta_0$ **then**
        $t = t - \delta t$, k = k - 1;
        Discretize (3) in V_h on interval $[t, t + \delta T]$ and get snapshots $\mathbf{W}_{h,1}$, then update POD modes $\mathbf{\Psi}_h$ by `Update_POD_Mode` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
        $k = k + \frac{\delta T}{\delta t}$;
    end if
end while

2.3. Complexity

In this subsection, we analyze the computational complexity of the standard finite element method, the standard POD method and the adaptive POD method introduced above. By comparing the computational complexity of these methods, the advantages of the POD type methods will be shown. We use $\mathcal{O}(n)$ to represent the complexity of a function that increases linearly with respect to n.

Case I: B, c and f are separable in time and space

We first consider the case that $\mathbf{B}$, c and f are separable in time and space. In this case, $\mathbf{B}$ can be expressed as

$$\mathbf{B}(x,y,z,t) = \mathbf{B_1}(x,y,z) + \mathbf{B_2}(x,y,z)B_3(t).$$

For the standard finite element method, the main computational costs at each time instant are those for building the discretized system and solving the discretized system. We first consider the cost for building the discretized system. We note that during the building of the linear system (8), only the terms $\left(\mathbf{B}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$, $\left(c\phi_{h,j},\phi_{h,i}\right)$ and $\left(f,\phi_{h,i}\right)$ are changed as the evolution of time. Let N_g denote the degree of freedoms for the finite element discretization. We only need to compute $\left(\mathbf{B_1}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$ and $\left(\mathbf{B_2}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$ once, and then multiply $\left(\mathbf{B_2}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$ by $B_3(t)$ at each time instant to obtain $\left(\mathbf{B}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$. For a fixed i, there are only few j such that $\left(\nabla\phi_{h,j},\phi_{h,i}\right)$ is not 0. This means the computational complexity of the multiplication by $B_3(t)$ is $\mathcal{O}\left(N_g\right)$. Therefore, the computational complexity for building $\left(\mathbf{B}\cdot\nabla\phi_{h,j},\phi_{h,i}\right)$ is $\mathcal{O}\left(N_g\right)$ at each time instant. Similarly, we have that the computational complexity for $\left(c\phi_{h,j},\phi_{h,i}\right)$ and $\left(f,\phi_{h,i}\right)$ is also $\mathcal{O}\left(N_g\right)$. Therefore, the computational complexity for building the linear system (8) is $\mathcal{O}\left(N_g\right)$. We then see the cost for solving the discretized linear system. For solution of (8), since $\mathbf{A}_h^k$ is sparse, there are many solvers [52,46] which can deal with it at a cost of $\mathcal{O}\left(N_g\right)$. Hence, the computational cost at each time instant is $\mathcal{O}\left(N_g\right)$, and the total cost for the time interval $[0,T]$ is

$$\mathcal{O}\left(N_g\right)\times\frac{T}{\delta t}.$$

We then turn to see the computational complexity of the standard POD method. When $t\in[0,T_0]$, we need to discrete the system (3) in the finite element space and then solve the discretized system (8). By the analysis for the standard finite element discretization, we know that the cost is $\mathcal{O}\left(N_g\right)$ at each time instant. For the construction of the POD modes, we need to perform SVD on $\mathbf{U}_h\in\mathbb{R}^{N_g\times n_s}$, where $n_s=\lfloor\frac{T_0}{\delta t\cdot\delta M}\rfloor$ as we mentioned above. Since only the left singular vectors and the singular values need to be calculated, the computational complexity is $\mathcal{O}\left(n_s^2N_g\right)$ [20], where $n_s^2\ll N_g$. When $t>T_0$, we need to project (3) onto the POD subspace and then solve the discretized system (12). Since $\mathbf{B}$, c and f are separable in time and space, as we mentioned above, we only need to calculate $\left(\mathbf{B_1}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$ and $\left(\mathbf{B_2}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$ once, which costs $\mathcal{O}\left(N_g\right)$, and then multiply it by $B_3(t)$ at each time instant to obtain $\left(\mathbf{B}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$, where $i,j=1,2,\cdots,m$. The computational complexity for computing $\left(c\psi_{h,j},\psi_{h,i}\right)$ and $\left(f,\psi_{h,i}\right)$ is similar to that for computing $\left(\mathbf{B}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$. Hence, the complexity for building the linear system (12) is $\mathcal{O}\left(m^2\right)$ at each time instant. For solving the discretized system (12), since $\bar{\mathbf{A}}_h^k$ is a small dense matrix, we usually use some direct method to solve it, which costs $\mathcal{O}\left(m^3\right)$. Therefore the total computational cost is

$$\mathcal{O}\left(N_g\right)\times\left(\frac{T_0}{\delta t}+1\right)+\mathcal{O}\left(n_s^2N_g\right)+\mathcal{O}\left(m^3\right)\times\frac{T-T_0}{\delta t}.$$

We now look at the computational cost for the adaptive POD method. Except for the costs same as those for the standard POD method, some additional costs are needed for steps **Estimate**, **Mark** and **Update**. For the step **Estimate**, different methods have different ways to design the error indicator. We denote the computational cost for this part in each time instant as t_{est}. The cost for the step **Mark** can be neglected. The main cost for the step **Update** is that for obtaining the standard finite element approximations on interval $[t,t+\delta T]$. From the analysis for the standard finite element method, we know that the cost for this part is $\mathcal{O}\left(N_g\right)$ at each time instant. Let n_A denote the number of update for POD modes. Therefore, the total computational cost for SVD is $\mathcal{O}\left(n_s^2N_g\right)\times 2n_A$. Since the number of POD modes will increase as the updating continues, we denote m_A average number of POD modes in the adaptive POD method. By the analysis for the standard POD method, the cost for building the POD linear system is

$$\mathcal{O}\left(N_g\right)\times n_A+\mathcal{O}\left(m_A^2\right)\times\frac{T-T_0-n_A\cdot\delta T}{\delta t},$$

and the cost for solving the discretized system at each time instant is $\mathcal{O}\left(m_A^3\right)$. Therefore, the total computational cost is

$$\mathcal{O}(N_g)\times\left(\frac{T_0+n_A\cdot\delta T}{\delta t}+n_A+1\right)+\mathcal{O}\left(n_s^2N_g\right)\times\left(2n_A+1\right)+\mathcal{O}\left(m_A^3\right)\times\frac{T-T_0-n_A\cdot\delta T}{\delta t}+t_{\text{est}}\times\frac{T-T_0-n_A\cdot T}{\delta t}.$$

We summarize the total computational cost for each method in Table 1. Usually, we have $T_0\ll T$, $m^3\ll N_g$ and $m_A^3\ll N_g$. Therefore, we can see that the POD type methods usually cost less CPU time than the standard finite element method. Then we focus on the cost for the adaptive POD method. If an error indicator is cheaper, the term t_{est} will be less. If an error indicator is more sensitive, it will require fewer number of update n_A to achieve the same accuracy, then it may decrease the degree of freedoms m_A at the same time. Therefore, the construction of the error indicator plays an important role in reducing the cost for the adaptive POD method.

Table 1
Complexity of different methods for the case of $\mathbf{B}$, c and f being separable in time and space.

Method	Complexity
standard finite element	$\mathcal{O}\left(N_g\right) \times \frac{T}{\delta t}$
standard POD method	$\mathcal{O}\left(N_g\right) \times \left(\frac{T_0}{\delta t}+1\right)+\mathcal{O}\left(n_s^2 N_g\right)+\mathcal{O}\left(m^3\right) \times \frac{T-T_0}{\delta t}$
adaptive POD method	$\mathcal{O}\left(N_g\right) \times \left(\frac{T_0+n_A \cdot \delta T}{\delta t}+n_A+1\right)+\mathcal{O}\left(n_s^2 N_g\right) \times\left(2 n_A+1\right)$ $+\ \mathcal{O}\left(m_A^3\right) \times \frac{T-T_0-n_A \cdot \delta T}{\delta t}+t_{\text{est}} \times \frac{T-T_0-n_A \cdot \delta T}{\delta t}$

Table 2
Total complexity of different methods for the case of $\mathbf{B}$, c, and f being not separable in time and space.

Method	Complexity
standard finite element	$\mathcal{O}\left(N_g\right) \times \frac{T}{\delta t}$
standard POD method	$\mathcal{O}\left(N_g\right) \times \left(\frac{T}{\delta t}\right)+\mathcal{O}\left(n_s^2 N_g\right)+\mathcal{O}\left(m^3\right) \times \frac{T-T_0}{\delta t}$
adaptive POD method	$\mathcal{O}\left(N_g\right) \times \left(\frac{T}{\delta t}+n_A\right)+\mathcal{O}\left(n_s^2 N_g\right) \times\left(2 n_A+1\right)$ $+\ \mathcal{O}\left(m_A^3\right) \times \frac{T-T_0-n_A \cdot \delta T}{\delta t}+t_{\text{est}} \times \frac{T-T_0-n_A \cdot \delta T}{\delta t}$

Case II: $\mathbf{B}$, c and f are not separable in time and space

We now consider the case that $\mathbf{B}$, c and f are not separable in time and space.

For the standard finite element method, we have to build the linear system (8) in all time instants. Similar to the case that $\mathbf{B}$, c, and f are separable, the computational cost for building the linear system (8) in each time instant is $\mathcal{O}\left(N_g\right)$, while the cost for solving the discretized linear system is $\mathcal{O}\left(N_g\right)$. Therefore, the total time cost for all the time interval $[0,T]$ is also

$$\mathcal{O}\left(N_g\right) \times \frac{T}{\delta t}.$$

While for the standard POD method and the adaptive POD method, the only difference in computational cost between this case and the case that $\mathbf{B}$, c, and f are separable lies in building the linear system (12). For this case, we have to calculate $\left(\mathbf{B} \cdot \nabla \psi_{h,j}, \psi_{h,i}\right)$, $\left(c \psi_{h,j}, \psi_{h,i}\right)$ and $\left(f, \psi_{h,i}\right)$ in all time instants. The cost for all the other parts is the same as that for the case that $\mathbf{B}$, c, and f are separable.

Therefore, the computational cost for the standard POD method is

$$\begin{aligned}
&\mathcal{O}\left(N_g\right) \times \left(\frac{T_0}{\delta t}\right)+\mathcal{O}\left(n_s^2 N_g\right)+\left(\mathcal{O}\left(m^3\right)+\mathcal{O}\left(N_g\right)\right) \times \frac{T-T_0}{\delta t} \\
&=\mathcal{O}\left(N_g\right) \times \left(\frac{T}{\delta t}\right)+\mathcal{O}\left(n_s^2 N_g\right)+\mathcal{O}\left(m^3\right) \times \frac{T-T_0}{\delta t}
\end{aligned}$$

While the computational cost for the adaptive POD method is

$$\begin{aligned}
&\mathcal{O}\left(N_g\right) \times \left(\frac{T_0+n_A \cdot \delta T}{\delta t}+n_A\right)+\mathcal{O}\left(n_s^2 N_g\right) \times\left(2 n_A+1\right)+\left(\mathcal{O}\left(m_A^3\right)+\mathcal{O}\left(N_g\right)\right) \times \frac{T-T_0-n_A \cdot \delta T}{\delta t} \\
&+t_{\text{est}} \times \frac{T-T_0-n_A \cdot T}{\delta t} \\
&=\mathcal{O}\left(N_g\right) \times \left(\frac{T}{\delta t}+n_A\right)+\mathcal{O}\left(n_s^2 N_g\right) \times\left(2 n_A+1\right)+\mathcal{O}\left(m_A^3\right) \times \frac{T-T_0-n_A \cdot \delta T}{\delta t}+t_{\text{est}} \times \frac{T-T_0-n_A \cdot T}{\delta t}
\end{aligned}$$

Similar to the case that $\mathbf{B}$, c and f are separable in time and space, we summarize the total computational cost for each method in Table 2.

If we take a detailed look at the Table 2, we can find that the computational cost for this case is much larger than the case $\mathbf{B}$, c, and f being separable, since the computation for building the linear system (12) is $\mathcal{O}\left(N_g\right)$ at each time instant.

Here, we propose a strategy to reduce the computational cost. We take $\left(\mathbf{B} \cdot \nabla \psi_{h,j}, \psi_{h,i}\right)$ for an example to illustrate the strategy. We divide the time interval $[T_0,T]$ into $M \in \mathbb{N}$ subintervals with equal length $\Delta T=(T-T_0)/M$ and set $T_k=T_0+k \cdot \Delta T, k=1,2,\cdots,M$, $\delta t \ll \Delta T$. For each time interval $[T_k,T_{k+1}]$, let t^* be its middle point. For any $t \in [T_k,T_{k+1}]$, we expand the function $\mathbf{B}(\mathbf{x},t)$ at t^* by Taylor expansion as follows:

$$\mathbf{B}(\mathbf{x},t)=\mathbf{B}(\mathbf{x},t^*)+\left.\frac{\partial \mathbf{B}(\mathbf{x},t)}{\partial t}\right|_{t=t^*}\left(t-t^*\right)+\left.\frac{\partial^2 \mathbf{B}(\mathbf{x},t)}{\partial t^2}\right|_{t=t^*}(t-t^*)^2+o\left((t-t^*)^2\right).$$

Then we can do the similar operation as for the case $\mathbf{B}$ being separable in time and space to evaluate $\left(\mathbf{B} \cdot \nabla \psi_{h,j}, \psi_{h,i}\right)$. We only need to compute

$$\left(\mathbf{B}(\mathbf{x},t^*) \cdot \nabla \psi_{h,j}, \psi_{h,i}\right),\left(\left.\frac{\partial \mathbf{B}(\mathbf{x},t)}{\partial t}\right|_{t=t^*} \cdot \nabla \psi_{h,j}, \psi_{h,i}\right),\left(\left.\frac{\partial^2 \mathbf{B}(\mathbf{x},t)}{\partial t^2}\right|_{t=t^*} \cdot \nabla \psi_{h,j}, \psi_{h,i}\right)$$

Table 3
Total complexity of different methods with new strategy for the case of $\mathbf{B}$, c, and f being not separable in time and space.

Method	Complexity
standard POD method	$\mathcal{O}\left(N_g\right)\times\left(\frac{T}{\Delta T}+1\right)+\mathcal{O}\left(n_s^2N_g\right)+\mathcal{O}\left(m^3\right)\times\frac{T-T_0}{\delta t}$
adaptive POD method	$\mathcal{O}\left(N_g\right)\times\left(\frac{T}{\Delta T}+n_A+1\right)+\mathcal{O}\left(n_s^2N_g\right)\times\left(2n_A+1\right)$ $+\ \mathcal{O}\left(m_A^3\right)\times\frac{T-T_0-n_A\cdot\delta T}{\delta t}+t_{\text{est}}\times\frac{T-T_0-n_A\cdot\delta T}{\delta t}$

once in time interval $[T_k,T_{k+1}]$, which costs $\mathcal{O}(N_g)$. Then we multiply each element $\left(\left.\frac{\partial\mathbf{B}(\mathbf{x},t)}{\partial t}\right|_{t=t^*}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$ by $(t-t^*)$, and multiply $\left(\left.\frac{\partial^2\mathbf{B}(\mathbf{x},t)}{\partial t^2}\right|_{t=t^*}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$ by $(t-t^*)^2$ at each time instant, which costs $\mathcal{O}\left(m^2\right)$. Therefore, the cost for computing $\left(\mathbf{B}\cdot\nabla\psi_{h,j},\psi_{h,i}\right)$ on each ΔT is

$$\mathcal{O}\left(N_g\right)+\mathcal{O}\left(m^2\right)\times\frac{\Delta T}{\delta t}.$$

Similarly, the cost for computing $\left(c\psi_{h,j},\psi_{h,i}\right)$ and $\left(f,\psi_{h,i}\right)$ is also

$$\mathcal{O}\left(N_g\right)+\mathcal{O}\left(m^2\right)\times\frac{\Delta T}{\delta t}.$$

Therefore, the total cost for building the linear system (12) is

$$\mathcal{O}\left(N_g\right)\times\frac{T-T_0}{\Delta T}+\mathcal{O}\left(m^2\right)\times\frac{T-T_0}{\delta t}.$$

By using this strategy, the time cost for the standard POD method and adaptive POD method are shown in Table 3. Since $\Delta T\gg\delta t$, we see that the computational cost is largely reduced compared with Table 2.

2.4. Typical existing adaptive POD methods

There are some existing works on adaptive POD methods [14,50,15]. The main difference between different adaptive POD methods lies in the construction of the error indicator. Here, we introduce two typical methods, one is the residual based adaptive POD method [14,50], the other is the two-grid based adaptive POD method [15].

For the residual based adaptive POD method, the residual is used to construct the error indicator. In detail, the error indicator η_k at time instant $t=k\cdot\delta t$ is defined as

$$\eta_k=\frac{\|\mathbf{A}_h^k\widetilde{\mathbf{R}}\mathbf{u}_{h,\text{POD}}^k-\mathbf{b}_h^k-\mathbf{C}_h\widetilde{\mathbf{R}}\mathbf{u}_{h,\text{POD}}^{k-1}\|_2}{\|\mathbf{b}_h^k+\mathbf{C}_h\widetilde{\mathbf{R}}\mathbf{u}_{h,\text{POD}}^{k-1}\|_2}. \tag{13}$$

We see that the computational cost t_{est} for η_k is $\mathcal{O}\left(N_g\right)$, the total computational cost for all steps **Estimate** is

$$\mathcal{O}\left(N_g\right)\times\frac{T-T_0-n_A\cdot\delta T}{\delta t}.$$

For the two-grid based adaptive POD method, two finite element spaces are constructed, the coarse finite element space V_H and the fine finite element space V_h. The fine finite element space is used to construct the POD modes, while the coarse finite element space is used to design the error indicator. Let Δt be the coarse time step. The error indicator η_k at time instant $t=k\cdot\Delta t$ is constructed by the approximations in the coarse finite element space V_H, that is,

$$\eta_k=\frac{\|u_H^k-u_{H,\text{POD}}^k\|_2}{\|u_H^k\|_2}. \tag{14}$$

Here u_H^k, $u_{H,\text{POD}}^k$ are the standard finite element approximation and the adaptive POD approximation, respectively. For a given η_0, if $\eta_k>\eta_0$, the time instant $t=k\cdot\Delta t$ will be picked out.

Let N_G denote the degrees of freedom of the coarse space. According to the complexity analysis above, the cost for obtaining the finite element approximations in coarse space is $\mathcal{O}\left(N_G\right)\times\frac{T}{\Delta t}$, and the computational cost for obtaining the adaptive POD approximations in coarse space is

$$\mathcal{O}\left(N_G\right)\times\left(\frac{T_0+n_A\cdot\delta T}{\Delta t}+n_A+1\right)+\mathcal{O}\left(n_s^2N_G\right)\times\left(2n_A+1\right)+\mathcal{O}\left(m_A^3\right)\times\frac{T-T_0-n_A\cdot\delta T}{\Delta t}.$$

Therefore, the computational cost for **Estimate** is

$$\mathcal{O}\left(N_G\right)\times\left(\frac{T+T_0+n_A\cdot\delta T}{\Delta t}+n_A+1\right)+\mathcal{O}\left(n_s^2N_G\right)\times\left(2n_A+1\right)+\mathcal{O}\left(m_A^3\right)\times\frac{T-T_0-n_A\cdot\delta T}{\Delta t}.$$

Since $N_G \ll N_g$, and $\delta t \ll \Delta t$, the cost for step **Estimate** in the two-grid based adaptive POD method is usually much cheaper than that for step **Estimate** in the residual based adaptive POD method.

3. Augmented subspace based adaptive POD method

As we mentioned above, the main difference between different adaptive POD methods is the construction of the error indicator. In this section, we introduce a new approach for developing some new adaptive POD methods, based on introducing a new error indicator.

3.1. General framework of the augmented subspace based adaptive POD method

The main idea of our new approach is to use some auxiliary modes to augment the current POD subspace, and then use the gap between the approximation obtained in the augmented subspace and that obtained in the original POD subspace to develop an error indicator.

Recall that the POD subspace is denoted as $V_{h,\mathrm{POD}} = \mathrm{span}\left\{\psi_{h,1}, \cdots, \psi_{h,m}\right\}$. Let $\psi_{h,m+1}, \cdots, \psi_{h,m+r}$ be some modes which are normalized and orthogonal against each other, and orthogonal against $V_{h,\mathrm{POD}}$. Then, we augment the subspace $V_{h,\mathrm{POD}}$ by $\widetilde{V}_{h,\mathrm{POD}} = V_{h,\mathrm{POD}} \oplus \mathrm{span}\left\{\psi_{h,m+1}, \cdots, \psi_{h,m+r}\right\}$. Next we design the error indicator η_k at time instant $t = k \cdot \delta t$.

At time instant $t = (k-1) \cdot \delta t$, the POD approximation in the subspace $V_{h,\mathrm{POD}}$ can be expressed as

$$u^{k-1}_{h,\mathrm{POD}}(x,y,z) = \sum_{i=1}^{m} \alpha^{k-1}_{h,i} \psi_{h,i}(x,y,z). \tag{15}$$

The approximation in the augmented subspace $\widetilde{V}_{h,\mathrm{POD}}$ at time instant $t = k \cdot \delta t$ can be expressed as

$$\widetilde{u}^{k}_{h,\mathrm{POD}}(x,y,z) = \sum_{i=1}^{m+r} \widetilde{\alpha}^{k}_{h,i} \psi_{h,i}(x,y,z). \tag{16}$$

Inserting (15) and (16) into (3), and setting $v = \psi_{h,j}, j = 1, 2, \cdots, m+r$, respectively, we get

$$\left(\sum_{i=1}^{m+r} \widetilde{\alpha}^{k}_{h,i} \psi_{h,i} - \sum_{i=1}^{m} \alpha^{k-1}_{h,i} \psi_{h,i}, \psi_{h,j}\right) + \delta t \cdot a\left(t_k; \sum_{i=1}^{m+r} \widetilde{\alpha}^{k}_{h,i} \psi_{h,i}, \psi_{h,j}\right) = \delta t \cdot \left(f_k, \psi_{h,j}\right). \tag{17}$$

The equation (17) can be rewritten as

$$\left(\sum_{i=1}^{m+r} \widetilde{\alpha}^{k}_{h,i} \psi_{h,i}, \psi_{h,j}\right) + \delta t \cdot a\left(t_k; \sum_{i=1}^{m+r} \widetilde{\alpha}^{k}_{h,i} \psi_{h,i}, \psi_{h,j}\right) = \delta t \cdot \left(f_k, \psi_{h,j}\right) + \left(\sum_{i=1}^{m} \alpha^{k-1}_{h,i} \psi_{h,i}, \psi_{h,j}\right). \tag{18}$$

Define

$$\begin{aligned}
&\widetilde{\mathbf{A}}^{k}_{h,ij} = \left(\psi_{h,j}, \psi_{h,i}\right) + \delta t \cdot a\left(t_k; \psi_{h,j}, \psi_{h,i}\right), \\
&\widetilde{\mathbf{b}}^{k}_{h} = \delta t \cdot \left((f_k, \psi_{h,1}), \cdots, (f_k, \psi_{h,m+r})\right)^T, \widetilde{\mathbf{C}}_{h,ij} = \left(\psi_{h,j}, \psi_{h,i}\right), \\
&\mathbf{u}^{k-1}_{h,\mathrm{POD}} = \left(\alpha^{k-1}_{h,1}, \cdots, \alpha^{k-1}_{h,m}\right)^T, \widetilde{\mathbf{u}}^{k}_{h,\mathrm{POD}} = \left(\widetilde{\alpha}^{k}_{h,1}, \cdots, \widetilde{\alpha}^{k}_{h,m+r}\right)^T.
\end{aligned}$$

Then we obtain the following algebraic system from (18)

$$\widetilde{\mathbf{A}}^{k}_{h} \widetilde{\mathbf{u}}^{k}_{h,\mathrm{POD}} = \widetilde{\mathbf{b}}^{k}_{h} + \widetilde{\mathbf{C}}_{h} \mathbf{u}^{k-1}_{h,\mathrm{POD}}, \tag{19}$$

where $\widetilde{\mathbf{A}}^{k}_{h} = \left(\widetilde{\mathbf{A}}^{k}_{h,ij}\right)_{(m+r)\times(m+r)}$ and $\widetilde{\mathbf{C}}_{h} = \left(\widetilde{\mathbf{C}}_{h,ij}\right)_{(m+r)\times(m+r)}$.

We define the error indicator η_k at time instant $t = k \cdot \delta t$ as

$$\eta_k = \frac{\|\widetilde{u}^{k}_{h,\mathrm{POD}} - u^{k}_{h,\mathrm{POD}}\|_2}{\|\widetilde{u}^{k}_{h,\mathrm{POD}}\|_2}. \tag{20}$$

For the convenience of the following discussion, we summarize the process for computing the error indicator as routine `Error_Indicator` ($\mathbf{A}^{k}_{h}, \mathbf{b}^{k}_{h}, \mathbf{C}_h, \bar{\mathbf{A}}^{k}_{h}, \bar{\mathbf{b}}^{k}_{h}, \bar{\mathbf{C}}_h, \mathbf{u}^{k-1}_{h,\mathrm{POD}}, \mathbf{d}_k, \mathbf{\Psi}_h, \eta_k$) in Algorithm 4, where $\mathbf{d}_k$ denotes the r auxiliary vectors at time instant $t = k \cdot \delta t$. We now see the computational cost of the step **Estimate** at each time instant. Denote the cost for constructing the auxiliary modes as t_{aux}. The operation of orthogonalization requires dot product of two vectors $r \cdot m_A + r(r-1)/2$ times, where m_A is the number of vectors in $\widetilde{\mathbf{R}}$, and the operation of normalization requires dot product of two vectors only r times. Note that the cost for dot product of two vectors with length N_g is $\mathcal{O}\left(N_g\right)$ and $m_A, r \ll N_g$, the computational cost for all the operations of orthogonal normalization is $\mathcal{O}\left(N_g\right)$. We note that the matrix in (21) can be obtained from the previously calculated linear system (12) except for the terms which contain $\mathbf{d}_k$, we only require to compute the elements of the last r rows and the last r columns of the matrix. Thanks to the sparsity of $\mathbf{A}^{k}_{h}$ and $\mathbf{C}_h$, the cost for building the augmented linear system is $\mathcal{O}\left(N_g\right)$. Since the size of the linear system in (21)

Algorithm 4 `Error_Indicator` $(\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h, \mathbf{u}_{h,\mathrm{POD}}^{k-1}, \mathbf{u}_{h,\mathrm{POD}}^k, \mathbf{d}_k, \mathbf{\Psi}_h, \eta_k)$.

Input: $\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h, \mathbf{u}_{h,\mathrm{POD}}^{k-1}$, $\mathbf{u}_{h,\mathrm{POD}}^k$, $\mathbf{d}_k$ and $\mathbf{\Psi}_h$, $\mathbf{\Psi}_h = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$;
Output: η_k;

1: Orthogonalize the vectors $\mathbf{d}_k$ against $\widetilde{\mathbf{R}}$ and make them normalized and orthogonal against each other. Denote the orthogonalized and normalized vectors still as $\mathbf{d}_k$;
2: Compute $\widetilde{\mathbf{u}}_{h,\mathrm{POD}}^k$ by

$$\begin{bmatrix} \bar{\mathbf{A}}_h^k & \widetilde{\mathbf{R}}^T \mathbf{A}_h^k \mathbf{d}_k \\ \mathbf{d}_k^T \mathbf{A}_h^k \widetilde{\mathbf{R}} & \mathbf{d}_k^T \mathbf{A}_h^k \mathbf{d}_k \end{bmatrix} \widetilde{\mathbf{u}}_{h,\mathrm{POD}}^k = \begin{bmatrix} \bar{\mathbf{b}}_h^k \\ \mathbf{d}_k^T \bar{\mathbf{b}}_h^k \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{C}}_h \\ \mathbf{d}_k^T \mathbf{C}_h \widetilde{\mathbf{R}} \end{bmatrix} \mathbf{u}_{h,\mathrm{POD}}^{k-1}; \tag{21}$$

3: Obtain the error indicator η_k by (20);

is only r more than that of linear system (12), similar to the analysis for the adaptive POD method, the cost for computing $\widetilde{\mathbf{u}}_{h,\mathrm{POD}}^k$ is $\mathcal{O}\left((m_A + r)^3\right)$. The computational cost for η_k in (20) is $\mathcal{O}\left(N_g\right)$. Therefore, the cost for step **Estimate** at each time instant is

$$\mathcal{O}\left(N_g\right) + \mathcal{O}\left((m_A + r)^3\right) + t_{aux}.$$

Remark 1. By some simple formal analysis, we can see that it is reasonable to use η_k defined in (20) as an error indicator. It is obvious that

$$u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k = u_{h,\mathrm{POD}}^k - u_h^k + u_h^k - \widetilde{u}_{h,\mathrm{POD}}^k. \tag{22}$$

Since $V_{h,\mathrm{POD}} \subset \widetilde{V}_{h,\mathrm{POD}} \subset V_h$, we have

$$\|u_h^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 \leq \|u_h^k - u_{h,\mathrm{POD}}^k\|_2. \tag{23}$$

From (22) and (23), we easily obtain

$$\frac{1}{2}\|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 \leq \|u_h^k - u_{h,\mathrm{POD}}^k\|_2.$$

In further, if there exists $0 < \zeta < 1$, s.t. $\|u_h^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 \leq \zeta \|u_h^k - u_{h,\mathrm{POD}}^k\|_2$, then from (22) we have

$$\|u_h^k - u_{h,\mathrm{POD}}^k\|_2 \leq \|u_h^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 + \|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 \leq \zeta \|u_h^k - u_{h,\mathrm{POD}}^k\|_2 + \|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2,$$

from which we get

$$\|u_h^k - u_{h,\mathrm{POD}}^k\|_2 \leq \frac{1}{1-\zeta}\|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2.$$

Therefore,

$$\frac{1}{2}\|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2 \leq \|u_h^k - u_{h,\mathrm{POD}}^k\|_2 \leq \frac{1}{1-\zeta}\|u_{h,\mathrm{POD}}^k - \widetilde{u}_{h,\mathrm{POD}}^k\|_2. \tag{24}$$

We want to point out that the analysis given above is some explanatory analysis. The error bound given in (24) is meaningful only when ζ is as close as possible to 0. However, it is usually very difficult to find some auxiliary modes to make the condition that ζ is as close as possible to 0 holds true.

Implementing the new error indicator defined in (20) into step 8 of Algorithm 3, we then obtain the general framework of our augmented subspace based adaptive POD method, as shown in Algorithm 5.

3.2. Specific augmented subspace based adaptive POD methods

The key for constructing the auxiliary modes includes two points: one is that the auxiliary modes can not be orthogonal to the exact solution, and it is better that the angles between these auxiliary modes and the exact solution are far away from $\pi/2$, the other is that they should be cheap to be constructed. We now provide two specific methods for obtaining the auxiliary modes. For simplicity, we only consider the case of $r = 1$.

3.2.1. Coarse-grid approximation type augmented subspace

We see from [15] that the solution obtained in the coarse finite element space is a good approximation for the solution obtained in the fine finite element space. Moreover, the computational cost for obtaining the solution approximation in the coarse finite element space is far less than that for obtaining the solution approximation in the fine finite element space. Therefore, here, we consider to use the approximated solution obtained in the coarse finite element space as the auxiliary mode to augment the current POD subspace.

We denote u_H^l the finite element approximation in coarse finite element space V_H at each time instant $t = l \cdot \Delta t$, where $l = 0, 1, \cdots, \lfloor \frac{T}{\Delta t} \rfloor$ and Δt still denotes the coarse time step, $\delta t \ll \Delta t$. We denote the interpolation of u_H^l in the fine finite element space by $u_{H,I}^l$. We set

Algorithm 5 General framework of the augmented subspace based adaptive POD method.

1: Given δt, T_0, δT, T, γ_1, γ_2, γ_3, δM, η_0 and the mesh $\mathcal{T}_h$;
2: Discretize (3) in V_h on interval $[0, T_0]$ and obtain $\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \mathbf{u}_h^k$, $\forall k \in [0, \lfloor T_0/\delta t \rfloor]$, then obtain snapshots $\mathbf{U}_h$ at different times t_0, $t_{\delta M}$, $\cdots$, $t_{n_s \cdot \delta M}$;
3: Construct POD modes $\mathbf{\Psi}_h = \{\psi_{h,1}, \cdots, \psi_{h,m}\}$ by `POD_Mode` $(\mathbf{U}_h, \gamma_1, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
4: $t = T_0$, $k = \frac{T_0}{\delta t}$;
5: **while** $t \le T$ **do**
6: $t = t + \delta t$, $k = k + 1$;
7: Discretize (3) in the POD subspace $V_{h,\mathrm{POD}} = \mathrm{span}\{\psi_{h,1}, \cdots, \psi_{h,m}\}$, then obtain $\bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h$ and $\mathbf{u}_{h,\mathrm{POD}}^k$;
8: Provide some auxiliary vectors $\mathbf{d}_k$,
9: Compute error indicator η_k by `Error_Indicator` $(\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h, \mathbf{u}_{h,\mathrm{POD}}^{k-1}, \mathbf{u}_{h,\mathrm{POD}}^k, \mathbf{d}_k, \mathbf{\Psi}_h, \eta_k)$;
10: **if** $\eta_k > \eta_0$ **then**
11: $t = t - \delta t$, $k = k - 1$;
12: Discretize (3) in V_h on interval $[t, t + \delta T]$ to get $\mathbf{u}_h^{k+i}, i = 1, \cdots, \frac{\delta T}{\delta t}$, then obtain snapshots $\mathbf{W}_{h,1}$;
13: Update POD modes $\mathbf{\Psi}_h$ by `Update_POD_Mode` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
14: $k = k + \frac{\delta T}{\delta t}$;
15: **end if**
16: **end while**

$$\mathbf{d}_k = \mathbf{u}_{H,I}^l,$$

where $k = \frac{\Delta t}{\delta t} l$.

Since we need to compute the auxiliary modes for the whole time interval in advance, the algorithm is a little different from the Algorithm 5. The whole routine of the augmented subspace based adaptive POD method with coarse-grid approximation is shown in Algorithm 6.

Algorithm 6 Augmented subspace based adaptive POD method with coarse-grid approximation.

1: Given δt, Δt, T_0, δT, T, γ_1, γ_2, γ_3, η_0, δM and the mesh $\mathcal{T}_h$, $\mathcal{T}_H$;
2: Discretize (3) in V_H on interval $[0, T]$, and obtain the approximations $\{u_H^l\}$, $\forall l \in [0, \lfloor \frac{T}{\Delta t} \rfloor]$;
3: Interpolate $\{u_H^l\}$ to the fine finite element space, then obtain the interpolations $\{u_{H,I}^l\}$;
4: Discretize (3) in V_h on interval $[0, T_0]$ and obtain $\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \mathbf{u}_h^k$, $\forall k \in [0, \lfloor T_0/\delta t \rfloor]$, then obtain snapshots $\mathbf{U}_h$ at different times t_0, $t_{\delta M}$, $\cdots$, $t_{n_s \cdot \delta M}$;
5: Construct POD modes $\mathbf{\Psi}_h = \{\psi_{h,1}, \cdots, \psi_{h,m}\}$ by `POD_Mode` $(\mathbf{U}_h, \gamma_1, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
6: $t = T_0$, $k = \frac{T_0}{\delta t}$, $w = \frac{\Delta t}{\delta t}$;
7: **while** $t \le T$ **do**
8: $t = t + \delta t$, $k = k + 1$;
9: Discretize (3) in the subspace $V_{h,\mathrm{POD}} = \mathrm{span}\{\psi_{h,1}, \cdots, \psi_{h,m}\}$, then obtain $\bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h$ and $\mathbf{u}_{h,\mathrm{POD}}^k$;
10: **if** $k \% w = 0$ **then**
11: $l = \frac{k}{w}$;
12: Compute error indicator η_k by `Error_Indicator` $(\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h, \mathbf{u}_{h,\mathrm{POD}}^{k-1}, \mathbf{u}_{h,\mathrm{POD}}^k, \mathbf{u}_{H,I}^l, \mathbf{\Psi}_h, \eta_k)$;
13: **if** $\eta_k > \eta_0$ **then**
14: $t = t - \delta t$, $k = k - 1$;
15: Discretize (3) in V_h on interval $[t, t + \delta T]$ to get $\mathbf{u}_h^{k+i}, i = 1, \cdots, \frac{\delta T}{\delta t}$, then obtain snapshots $\mathbf{W}_{h,1}$;
16: Update POD modes $\mathbf{\Psi}_h$ by `Update_POD_Mode` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, m, \mathbf{\Psi}_h)$;
17: $k = k + \frac{\delta T}{\delta t}$;
18: **end if**
19: **end if**
20: **end while**

Next we analyze the computational complexity of the step **Estimate** for this strategy. As we mentioned above, the cost for obtaining the finite element approximation at each time instant in the coarse finite element space is $\mathcal{O}(N_G)$. The cost for interpolating a function in the coarse finite element space to the fine finite element space is $\mathcal{O}(N_g)$. From the analysis in the Subsection 3.1, the cost for step **Estimate** at each time instant is

$$\mathcal{O}(N_g) + \mathcal{O}(m_A^3) + \mathcal{O}(N_G).$$

We only need to compute the error indicator at each coarse time step. Therefore, the total cost for **Estimate** is

$$\left(\mathcal{O}(N_g) + \mathcal{O}(m_A^3)\right) \times \frac{T - T_0 - n_A \cdot \delta T}{\Delta t} + \left(\mathcal{O}(N_G) + \mathcal{O}(N_g)\right) \times \frac{T}{\Delta t}.$$

Since $\delta t \ll \Delta t$ and $N_G \ll N_g$, the cost for **Estimate** is relatively cheap compared with the cost for the other parts.

3.2.2. Residual type augmented subspace

Motivated by [5], here we consider using the residual corresponding to the POD approximation at time instant $t = k \cdot \delta t$

$$\mathbf{d}_k = \mathbf{A}_h^k \widetilde{\mathbf{R}} \mathbf{u}_{h,\mathrm{POD}}^k - \mathbf{b}_h^k - \mathbf{C}_h \widetilde{\mathbf{R}} \mathbf{u}_{h,\mathrm{POD}}^{k-1}$$

as the auxiliary mode. Applying $\mathbf{d}_k$ into step 8 of Algorithm 5, we obtain the residual type augmented subspace based adaptive POD method.

Algorithm 7 `POD_Mode_Weight`$(\mathbf{U}_h, \gamma, \mathbf{\Phi}_h, \mathbf{S}, m, \mathbf{\Psi}_h)$.

Input: $\mathbf{U}_h, \gamma, \mathbf{\Phi}_h = \left(\phi_{h,1}, \phi_{h,2}, \cdots, \phi_{h,n}\right)$;
Output: m, POD modes $\mathbf{\Psi}_h = \left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right)$ and $\mathbf{S}$;
1: Perform SVD on $\mathbf{U}_h$ to obtain $\mathbf{U}_h = \mathbf{R}\mathbf{S}\mathbf{V}^{\mathbf{T}}$, where $\mathbf{S} = \text{diag}\left\{\sigma_1, \sigma_2, \cdots, \sigma_r\right\}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$;
2: Set $m = \min\left\{k | \sum_{i=1}^{k} \mathbf{S}_{i,i} > \gamma \cdot \text{Trace}(\mathbf{S})\right\}$;
3: $\left(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right) = \mathbf{\Phi}_h \mathbf{R}[:, 1:m]$;

Algorithm 8 `Update_POD_Mode_Weight` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, \boldsymbol{\alpha}, \mathbf{S}_2, m, \mathbf{\Psi}_h)$.

Input: $\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h = (\phi_{h,1}, \phi_{h,2}, \cdots, \phi_{h,n})$, $\boldsymbol{\alpha}$, $\mathbf{S}_2$, m and m old POD modes $\mathbf{\Psi}_h, \mathbf{\Psi}_h = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$.
Output: new m, new $\mathbf{S}_2$ and new m POD modes $\left\{\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}\right\}$.
1: Perform SVD on $\mathbf{W}_{h,1}$ to obtain $\mathbf{W}_{h,1} = \mathbf{R}_1 \mathbf{S}_1 \mathbf{V}_1^T$, where $\mathbf{S}_1 = \text{diag}\left\{\sigma_{1,1}, \sigma_{1,2}, \cdots, \sigma_{1,r_1}\right\}$ with $\sigma_{1,1} \geq \sigma_{1,2} \geq \cdots \geq \sigma_{1,r_1} > 0$;
2: Set $m_1 = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{1,ii} > \gamma_2 \cdot \text{Trace}(\mathbf{S}_1)\right\}$, $\hat{v}_j = \min\left\{\frac{\sigma_{2,j}}{\sqrt{\sum_{k=1}^{m}(\sigma_{2,k})^2}}, \frac{\langle|\alpha_j|\rangle}{\sqrt{\sum_{k=1}^{m}\langle|\alpha_k|\rangle^2}}\right\}, v_j = \frac{\sigma_{1,j}}{\sqrt{\sum_{k=1}^{m_1}(\sigma_{1,k})^2}}$.
3: Perform SVD on

$$\mathbf{W}_{h,2} = [\hat{v}_1 \widetilde{\mathbf{R}}[:,1], \cdots, \hat{v}_m \widetilde{\mathbf{R}}[:,m], v_1 \mathbf{R}_1[:,1], \cdots, v_{m_1} \mathbf{R}_1[:,m_1]],$$

and obtain $\mathbf{W}_{h,2} = \mathbf{R}_2 \mathbf{S}_2 \mathbf{V}_2^T$, where $\mathbf{S}_2 = \text{diag}\left\{\sigma_{2,1}, \sigma_{2,2}, \cdots, \sigma_{2,r_2}\right\}$ with $\sigma_{2,1} \geq \sigma_{2,2} \geq \cdots \geq \sigma_{2,r_2} > 0$;
4: Set $m = \min\left\{k : \sum_{i=1}^{k} \mathbf{S}_{2,ii} > \gamma_3 \cdot \text{Trace}(\mathbf{S}_2)\right\}$, and $\widetilde{\mathbf{R}} = \mathbf{R}_2[:, 1:m]$;
5: $(\psi_{h,1}, \psi_{h,2}, \cdots, \psi_{h,m}) = \mathbf{\Phi}_h \widetilde{\mathbf{R}}$;

We now see the computational cost. The cost for computing the residual is $\mathcal{O}\left(N_g\right)$. From the analysis in the Subsection 3.1, the cost for step **Estimate** at each time instant is

$$\mathcal{O}\left(N_g\right) + \mathcal{O}\left(m_A^3\right).$$

3.3. Weighting strategy

Inspired by [50], we introduce some weight to each mode when updating the POD modes. We hope this strategy can improve the efficiency of the POD modes. Recall that the POD approximation at the time instant $t = k \cdot \delta t$ can be expressed as

$$u_{h,\text{POD}}^k(x,y,z) = \sum_{i=1}^{m} \alpha_{h,i}^k \psi_{h,i}(x,y,z),$$

where m denotes the number of the POD modes. A little different from Subsection 2.2, we update the POD modes by performing SVD on

$$\mathbf{W}_{h,2} = [\hat{v}_1 \widetilde{\mathbf{R}}[:,1], \cdots, \hat{v}_m \widetilde{\mathbf{R}}[:,m], v_1 \mathbf{R}_1[:,1], \cdots, v_{m_1} \mathbf{R}_1[:,m_1]],$$

where the weights $\hat{v}_j$ and v_j are defined as

$$\hat{v}_j = \min\left\{\frac{\sigma_{2,j}}{\sqrt{\sum_{k=1}^{m}(\sigma_{2,k})^2}}, \frac{\langle|\alpha_j|\rangle}{\sqrt{\sum_{k=1}^{m}\langle|\alpha_k|\rangle^2}}\right\}, v_j = \frac{\sigma_{1,j}}{\sqrt{\sum_{k=1}^{m_1}(\sigma_{1,k})^2}}.$$

Here $\sigma_{2,j}, \sigma_{1,j}$ are the singular values corresponding to $\widetilde{\mathbf{R}}[:,j]$ and $\mathbf{R}_1[:,j]$, respectively, and $\langle|\alpha_j|\rangle$ is the temporal mean value of $|\alpha_{h,j}^k|$ provided by the POD solution over the last time interval solved by the POD method.

We set $\boldsymbol{\alpha} := [\langle|\alpha_1|\rangle, \langle|\alpha_2|\rangle, \cdots, \langle|\alpha_m|\rangle]$. For the convenience of the following discussion, we summarize this process for the POD modes as routine `POD_Mode_Weight` $(\mathbf{U}_h, \gamma, \mathbf{\Phi}_h, \mathbf{S}, m, \mathbf{\Psi}_h)$ in Algorithm 7 and the step **Update** as routine `Update_POD_Mode_Weight` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, \boldsymbol{\alpha}, \mathbf{S}_2, m, \mathbf{\Psi}_h)$ in Algorithm 8.

Replace Algorithm 1 and Algorithm 2 in Algorithm 5 by Algorithm 7 and Algorithm 8, respectively, and compute $\boldsymbol{\alpha}$ before updating the POD modes, then we obtain the general framework of the weighting augmented subspace based adaptive POD method, see Algorithm 9 for the details.

Algorithm 9 General framework of the weighting augmented subspace based adaptive POD method.

Given δt, T_0, δT, T, γ_1, γ_2, γ_3, η_0, δM and the mesh $\mathcal{T}_h$;
Discretize (3) in V_h on interval $[0, T_0]$ and obtain $\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \mathbf{u}_h^k$, $\forall k \in [0, \lfloor T_0/\delta t \rfloor]$, then obtain snapshots $\mathbf{U}_h$ at different times t_0, $t_{\delta M}$, $\cdots$, $t_{n_s \cdot \delta M}$;
Construct POD modes $\mathbf{\Psi}_h = \{\psi_{h,1}, \cdots, \psi_{h,m}\}$ by `POD_Mode_Weight` $(\mathbf{U}_h, \gamma_1, \mathbf{\Phi}_h, \mathbf{S}, m, \mathbf{\Psi}_h)$;
$t = T_0$, $k = \frac{T_0}{\delta t}$;
while $t \le T$ **do**
  $t = t + \delta t$, $k = k + 1$;
  Discretize (3) in the POD subspace $V_{h,\mathrm{POD}} = \mathrm{span}\{\psi_{h,1}, \cdots, \psi_{h,m}\}$, then obtain $\bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h$ and $\mathbf{u}_{h,\mathrm{POD}}^k$;
  Provide some auxiliary vectors $\mathbf{d_k}$,
  Compute error indicator η_k by `Error_Indicator` $(\mathbf{A}_h^k, \mathbf{b}_h^k, \mathbf{C}_h, \bar{\mathbf{A}}_h^k, \bar{\mathbf{b}}_h^k, \bar{\mathbf{C}}_h, \mathbf{u}_{h,\mathrm{POD}}^{k-1}, \mathbf{u}_{h,\mathrm{POD}}^k, \mathbf{d}_k, \mathbf{\Psi}_h, \eta_k)$;
  if $\eta_k > \eta_0$ **then**
    $t = t - \delta t$, $k = k - 1$;
    Compute $\boldsymbol{\alpha} = [\langle|\alpha_1|\rangle, \langle|\alpha_2|\rangle, \cdots, \langle|\alpha_m|\rangle]$;
    Discretize (3) in V_h on interval $[t, t + \delta T]$ to get $\mathbf{u}_h^{k+i}, i = 1, \cdots, \frac{\delta T}{\delta t}$, then obtain snapshots $\mathbf{W}_{h,1}$;
    Update POD modes $\mathbf{\Psi}_h$ by `Update_POD_Mode_Weight` $(\mathbf{W}_{h,1}, \gamma_2, \gamma_3, \mathbf{\Phi}_h, \boldsymbol{\alpha}, \mathbf{S}, m, \mathbf{\Psi}_h)$;
    $k = k + \frac{\delta T}{\delta t}$;
  end if
end while

4. Numerical examples

In this section, we will use two typical fluid advection fields with chaotic streamlines, the Kolmogorov flow and the ABC flow, to show the accuracy and efficiency of our augmented subspace based adaptive POD method.

We use the standard finite element method approximation as the reference solution, and compare our new methods with the standard POD method and the two-grid based adaptive POD method, respectively. Since the two-grid based adaptive POD method has been compared with some other existing adaptive POD methods and has shown to be more efficient than the other adaptive POD methods in [15], we do not compare our new methods with the other adaptive POD methods here.

The relative error of the approximation obtained at each time instant is calculated by

$$\mathrm{Error} = \frac{\|u_h^k - u_{h,*}^k\|_2}{\|u_h^k\|_2}, \tag{25}$$

where u_h^k and $u_{h,*}^k$ represent the finite element approximations and different types of the POD approximations at different times $t = t_k$, respectively. The numerical experiments are carried out on the high performance computers LSSC-IV of the State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences, and our code is based on the toolbox PHG [47].

In the following discussions, we denote the standard finite element method, the standard POD method and the two-grid based adaptive POD method as "FEM", "POD" and "TG-APOD", respectively. For the augmented subspace based adaptive POD methods, we denote the methods based on residual and coarse-grid approximation as "Res-Aug-APOD" and "Coarse-Aug-APOD", respectively. We add "-W" to each method to denote the method in Algorithm 9 where some weights are introduced to each mode when updating the POD modes. To compare more clearly, we will bold the best results among those obtained by different methods for each case of ϵ.

4.1. Kolmogorov flow

We consider the following advection-diffusion equation with the advection being the Kolmogorov flow [43,6],

$$\begin{cases} u_t - \epsilon \Delta u + \mathbf{B}(x, y, z, t) \cdot \nabla u = f(x, y, z, t), \quad (x, y, z) \in \Omega, t \in [0, T], \\ u(x, y, z, 0) = 0, \\ u(x + 2\pi, y, z, t) = u(x, y + 2\pi, z, t) = u(x, y, z + 2\pi, t) = u(x, y, z, t), \end{cases} \tag{26}$$

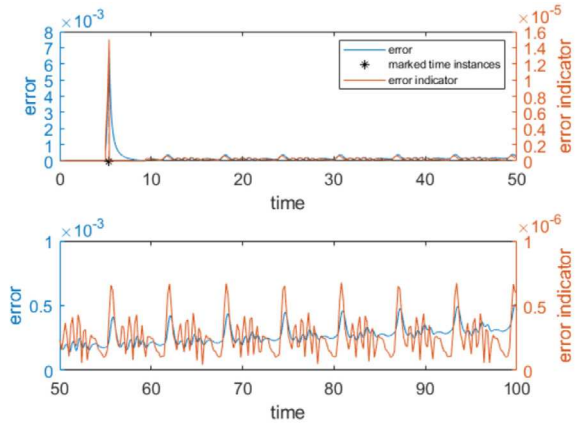
where

$$\mathbf{B}(x, y, z, t) = (\cos(y), \cos(z), \cos(x)) + (\sin(z), \sin(x), \sin(y))\cos(t),$$

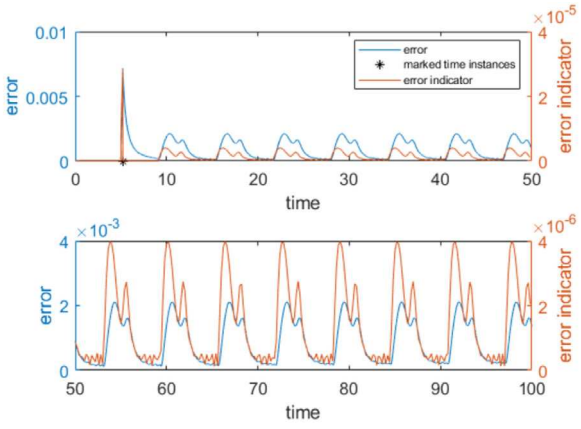
$$f(x, y, z, t) = -\cos(y) - \sin(z) \cdot \cos(t),$$

$$\Omega = [0, 2\pi]^3, T = 100.$$

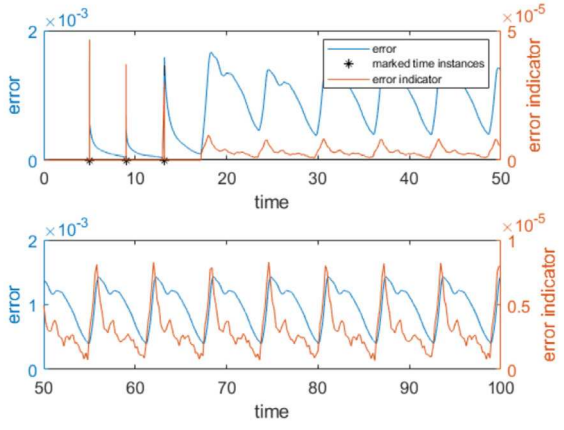
We will test 6 different cases with $\epsilon = 1, 0.5, 0.1, 0.05, 0.01, 0.005$, respectively. We first divide Ω into 6 tetrahedrons to act as the initial mesh. Then we refine the initial mesh 23 times uniformly using bisection to obtain the fine mesh for cases of $\epsilon = 1, 0.5, 0.1, 0.05$, and refine the initial mesh 24 times to obtain the fine mesh for cases of $\epsilon = 0.01, 0.005$. We use the piecewise linear function as the finite element basis and set $\delta t = 5 \times 10^{-3}$. For all the POD type methods, we set $T_0 = 5$, $\delta M = 20$. For the adaptive POD methods, we set $\delta T = 4$. In all the numerical experiments, we choose the parameters γ_i $(i = 1, 2, 3)$ as $\gamma_1 = \gamma_2 = 0.999$, $\gamma_3 = 1.0 - 1.0 \times 10^{-8}$. For the methods TG-APOD and Coarse-Aug-APOD, we refine the initial mesh 14 times to obtain the coarse mesh for cases of $\epsilon = 1, 0.5, 0.1$, and refine the initial mesh 15 times to obtain the coarse mesh for cases of $\epsilon = 0.05, 0.01$ and refine the initial mesh 16 times to obtain the coarse mesh for case of $\epsilon = 0.005$. The time steps corresponding to the coarse finite element spaces are 0.2, 0.125 and 0.1,



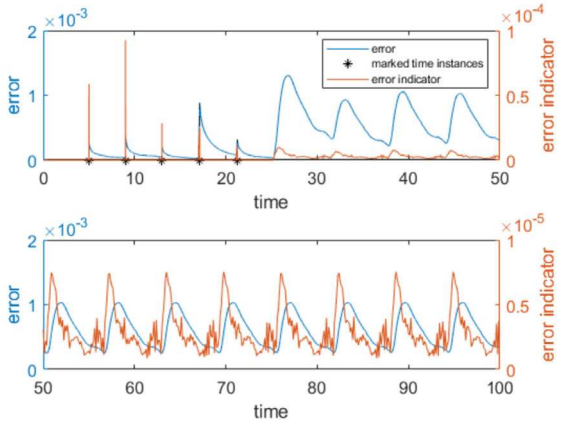
(a) $\epsilon = 1$.



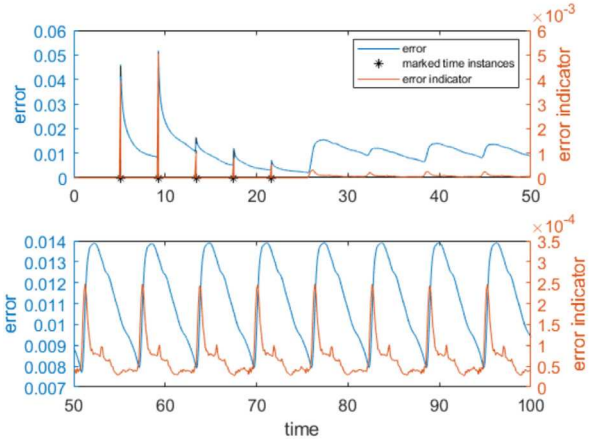
(b) $\epsilon = 0.5$.



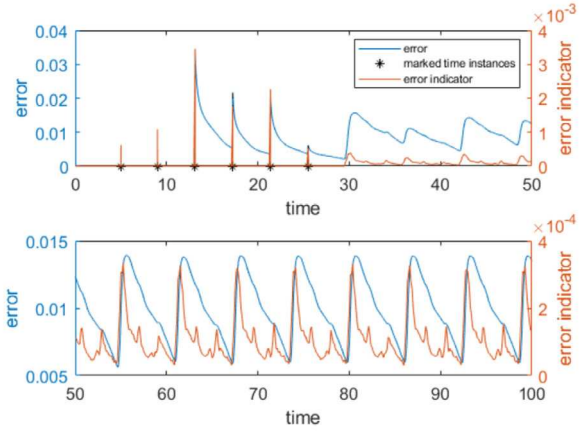
(c) $\epsilon = 0.1$.



(d) $\epsilon = 0.05$.



(e) $\epsilon = 0.01$.

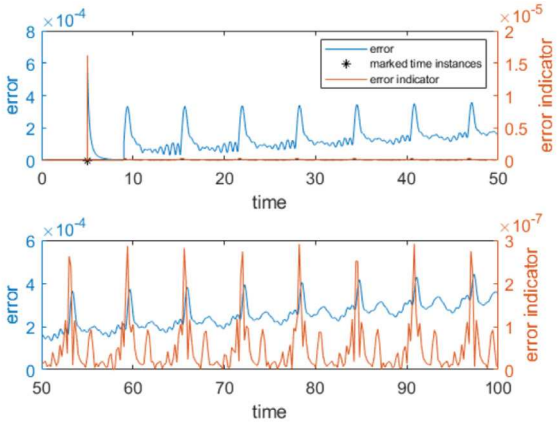


(f) $\epsilon = 0.005$.

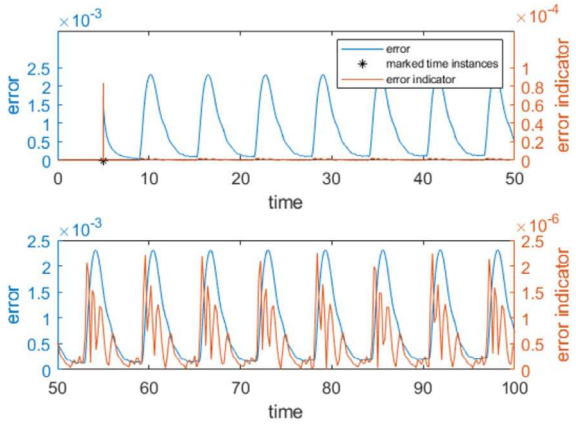
Fig. 1. The evolution curves of the error indicator and the error obtained by the method Residual-Aug-APOD for solution of (26) with $\epsilon = 1, 0.5, 0.1, 0.05, 0.01, 0.005$, respectively. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

respectively. We use 72 processors to do the simulation for cases of the fine mesh being obtained by refining the initial mesh 23 times, and 180 processors for cases of the fine mesh being obtained by refining the initial mesh 24 times.

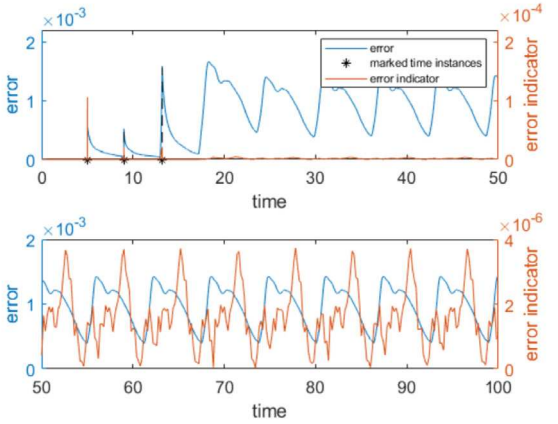
We first show the variation of the error and the error indicator obtained by our methods Res-Aug-APOD and Coarse-Aug-APOD in Fig. 1 and Fig. 2, respectively. The sub-figures at the top describe the variation curves of error and error indicator on time interval



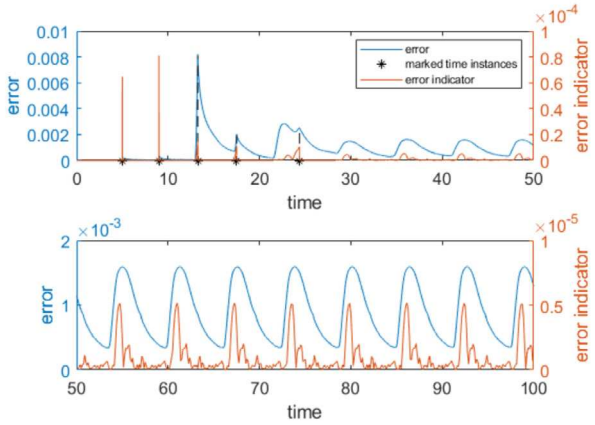
(a) $\epsilon = 1$.



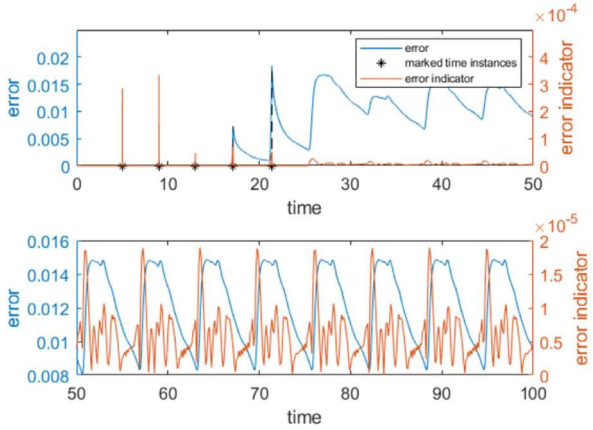
(b) $\epsilon = 0.5$.



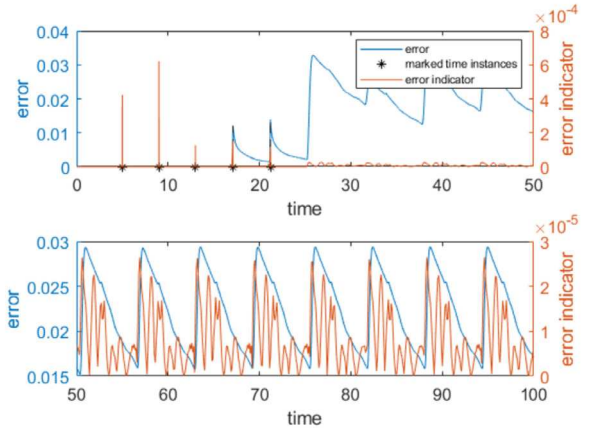
(c) $\epsilon = 0.1$.



(d) $\epsilon = 0.05$.



(e) $\epsilon = 0.01$.



(f) $\epsilon = 0.005$.

Fig. 2. The evolution curves of the error indicator and the error obtained by the method Coarse-Aug-APOD for solution of (26) with $\epsilon = 1, 0.5, 0.1, 0.05, 0.01, 0.005$, respectively.

Table 4
The results of (26) with different ϵ obtained by methods FEM, POD, TG-APOD, Res-Aug-APOD, Coarse-Aug-APOD, Res-Aug-APOD-W and Coarse-Aug-APOD-W, respectively.

ϵ	Method	η_0	Update Times	DOFs	Error	Average Error	Time(s)
1	FEM	-	-	10485760	-	-	15041.77
	POD	-	-	19	0.003877	0.001094	1859.07
	TG-APOD	1×10^{-3}	1	13	0.000350	0.000207	1829.59
	Res-Aug-APOD	1×10^{-5}	1	13	0.000465	0.000240	2582.48
	Coarse-Aug-APOD	1×10^{-5}	1	14	0.000347	0.000184	1900.86
	Res-Aug-APOD-W	1×10^{-5}	1	13	0.000465	0.000240	2590.02
	Coarse-Aug-APOD-W	1×10^{-5}	1	13	0.000347	0.000184	1885.10
0.5	FEM	-	-	10485760	-	-	14772.71
	POD	-	-	22	0.002337	0.009928	1805.14
	TG-APOD	5×10^{-3}	1	15	0.001546	0.001568	1685.59
	Res-Aug-APOD	1×10^{-5}	1	16	0.001349	0.000872	2448.96
	Coarse-Aug-APOD	1×10^{-5}	1	16	0.000706	0.000808	1726.65
	Res-Aug-APOD-W	1×10^{-5}	1	16	0.001349	0.000872	2429.59
	Coarse-Aug-APOD-W	1×10^{-5}	1	16	0.000706	0.000808	1752.03
0.1	FEM	-	-	10485760	-	-	14998.48
	POD	-	-	29	0.140401	0.223960	1943.66
	TG-APOD	1×10^{-3}	3	47	0.001719	0.001238	3154.18
	Res-Aug-APOD	1×10^{-5}	3	48	0.001428	0.000839	3874.90
	Coarse-Aug-APOD	1×10^{-5}	3	48	0.001428	0.000839	3182.90
	Res-Aug-APOD-W	1×10^{-5}	3	48	0.001428	0.000839	3845.91
	Coarse-Aug-APOD-W	1×10^{-5}	3	48	0.001428	0.000839	3141.46
0.05	FEM	-	-	10485760	-	-	13188.19
	POD	-	-	34	0.325160	0.415044	1448.42
	TG-APOD	1×10^{-3}	5	85	0.000620	0.000958	4754.66
	Res-Aug-APOD	1×10^{-5}	5	88	0.000334	0.000479	5654.76
	Coarse-Aug-APOD	1×10^{-5}	5	89	0.001293	0.000890	4858.62
	Res-Aug-APOD-W	1×10^{-5}	5	88	0.000334	0.000479	5663.85
	Coarse-Aug-APOD-W	1×10^{-5}	5	89	0.001293	0.000890	4822.26
0.01	FEM	-	-	16777216	-	-	11784.99
	POD	-	-	43	0.919008	0.777626	1267.46
	TG-APOD	1×10^{-3}	6	166	0.007042	0.004837	5072.23
	Res-Aug-APOD	2×10^{-4}	6	172	0.006008	0.003650	6338.19
	Coarse-Aug-APOD	1×10^{-5}	6	172	0.006208	0.003351	5280.25
	Res-Aug-APOD-W	2×10^{-4}	6	170	0.006011	0.003644	6196.64
	Coarse-Aug-APOD-W	1×10^{-5}	6	170	0.006169	0.003339	5249.77
0.005	FEM	-	-	16777216	-	-	12526.50
	POD	-	-	45	1.010868	0.871304	1306.83
	TG-APOD	1×10^{-3}	7	235	0.008957	0.007559	6661.74
	Res-Aug-APOD	3×10^{-4}	8	270	0.002095	0.002329	8605.53
	Coarse-Aug-APOD	1×10^{-5}	7	239	0.007957	0.004724	6778.84
	Res-Aug-APOD-W	3×10^{-4}	8	267	0.002094	0.002346	8655.01
	Coarse-Aug-APOD-W	1×10^{-5}	7	237	0.007974	0.004732	6679.24

$[0, 50]$, and the sub-figures at the bottom describe those on time interval $[50, 100]$. The x-axis of each sub-figure is the time, the y-axis in the left of each sub-figure is the error which is defined by (25), while the y-axis in the right of each sub-figure is the error indicator defined by (20). In each sub-figure, the blue curve describes the variation of the error, the orange curve describes the variation of the error indicator, and the black star denotes the marked time instant when the POD modes are to be updated. It should be pointed out that the error at the marked time instant is that obtained before the update of the POD modes.

From each sub-figure at the bottom of Fig. 1 and Fig. 2, the time instants when the error indicator achieves its local maximizer coincide very well with those when the error achieves its local maximizer. This indicates that our error indicator is very effective in picking out the time instant when the POD subspace is needed to be updated.

However, we also observe that the error indicator is about one to two order of magnitude smaller than the exact error. In our opinion, the error indicator we used in this paper is not an error estimator, it is just an error detector. For an error estimator, we require that it can bound the error from below and above. An error detector, which may not be able to bound the error from below and above, but can just tell us that if the error is observable or not. In an adaptive POD method, we only need an indicator to tell us if we need update the POD subspace or not, whether it can bound the exact error is not important.

Some numerical results obtained by different methods are shown in Table 4. We set different thresholds η_0 for different adaptive POD methods, since the meanings of the two error indicators are different. The error indicator for the method TG-APOD can be viewed as an error estimator, which can bound the exact error when the approximation obtained on the coarse mesh is a good approximation to that obtained on the fine mesh. In fact, we can observe from the numerical tests in [15] that the variation of the error indicator

Table 5
The results of (26) with $\epsilon = 0.01, 0.005$ for some cases of the weighting strategy performing well.

ϵ	Method	γ_3	Update Times	DOFs	Error	Average Error	Time (s)
0.01	Res-Aug-APOD	$1 - 1 \times 10^{-4}$	6	164	0.008184	0.005161	6191.93
	Res-Aug-APOD-W	$1 - 1 \times 10^{-4}$	6	143	0.006527	0.003916	5852.54
	Coarse-Aug-APOD	$1 - 1 \times 10^{-4}$	7	180	0.008273	0.004145	5741.76
	Coarse-Aug-APOD-W	$1 - 1 \times 10^{-4}$	6	127	0.006546	0.003728	4833.13
0.005	Coarse-Aug-APOD	$1 - 1 \times 10^{-5}$	7	232	0.008209	0.004821	6726.24
	Coarse-Aug-APOD-W	$1 - 1 \times 10^{-5}$	8	228	0.003266	0.003669	7111.19

coincide the exact error very well. However, as stated above, the error indicator for the augmented subspace based adaptive POD method constructed in this paper is just an error detector. Although it is good enough to indicate if the current POD space need to be updated or not, the error indicator is about one to two order of magnitude smaller than the exact error. That's why that the η_0 used in our augmented subspace based APOD method is much smaller than the exact error, while the η_0 used in the two-grid based APOD method is in the same magnitude as the exact error. Anyway, both the error estimator and error detector are OK to act as an indicator to tell us if we need update the POD subspace or not.

To judge the performance of different methods, we choose the best result among those obtained by each method with different choice of η_0 to be compared by a trade-off of the computational error and cost. We think this comparison is fair and meaningful.

In Table 4, "Update Times" means the number of update for the POD modes in the adaptive POD methods, "DOFs" means the degrees of freedom, "Error" denotes the relative error for numerical solution at t = T, "Average Error" denotes the average of relative error for numerical approximation at each time instant on time interval $[0, T]$ and "Time" means the wall time for the simulation.

From Table 4, we can first see that although the number of update and the degrees of freedom for the POD type methods increase as the decrease of ϵ, the degrees of freedom for the POD type methods for all cases of different ϵ are much smaller than those for the standard finite element method. We also see that as the decrease of ϵ, the error obtained by the standard POD method increases dramatically, which makes the results unreliable when ϵ is close to 0. This shows that the smaller the ϵ, the more difficult the model to be simulated. Fortunately, results obtained by the three adaptive POD methods are still of high accuracy even for case of $\epsilon = 0.005$, and the CPU time cost by the adaptive POD methods is almost less than one-half of that used by the standard finite element method. This shows that the adaptive POD methods behave much better than the standard POD method.

We now compare the three adaptive POD methods, TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD. We see from Table 4 that the both two methods Coarse-Aug-APOD and Res-Aug-APOD can achieve higher accuracy than the method TG-APOD with similar degrees of freedom for cases of ϵ being very close to 0. Moreover, for all cases of ϵ, the method Coarse-Aug-APOD can achieve higher accuracy than the method TG-APOD while costing similar CPU time. For the comparison between our two augmented subspace based adaptive POD methods, we can see that the method Res-Aug-APOD can obtain results with comparable or even higher accuracy than the method Coarse-Aug-APOD, but takes more CPU time. Therefore, taking into account both the accuracy and the cost, the method Coarse-Aug-APOD is the most recommended one.

To see more clearly, we show the variation curves of the error obtained by the methods TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD in Fig. 3.

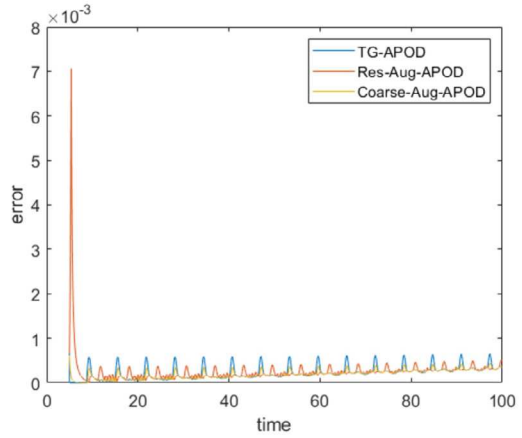
In Fig. 3, the x-axis is the time, the y-axis is the relative error of numerical solution obtained by adaptive POD methods. For each case of ϵ, the blue curve, orange curve and yellow curve denote the error obtained by the method TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD, respectively.

From Fig. 3, we can easily see that the error over the entire time interval obtained by our methods Res-Aug-APOD and Coarse-Aug-APOD is almost always smaller than that obtained by the method TG-APOD. Taking into account the wall time reported in Table 4, we can see that our new method Coarse-Aug-APOD is the most efficient one.

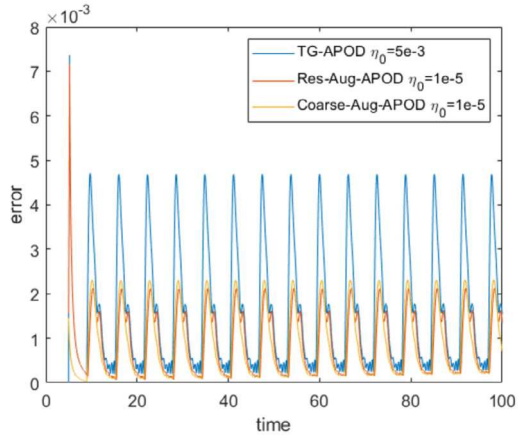
We then see the performance of the two methods Res-Aug-APOD-W and Coarse-Aug-APOD-W in Table 4. By comparing the results obtained by these two methods with those obtained by methods Res-Aug-APOD and Coarse-Aug-APOD respectively, we can see that the method Res-Aug-APOD-W has almost similar behavior with Res-Aug-APOD, while Coarse-Aug-APOD-W has almost similar behavior with Coarse-Aug-APOD. Especially, for case of $\epsilon = 0.01$, the method Coarse-Aug-APOD-W can obtain results with a little higher accuracy than the method Coarse-Aug-APOD.

Besides, we have done some more tests for the Res-Aug-APOD-W and Coarse-Aug-APOD-W by using different parameter γ_3. We use the cases of $\epsilon = 0.01, 0.005$ as examples to do the tests. By comparing the results obtained by the methods Coarse-Aug-APOD and Coarse-Aug-APOD-W with the same parameters, we can see that for most of γ_3, the two methods behave almost the same, however, there are still some cases where Coarse-Aug-APOD-W behaves better than the method Coarse-Aug-APOD, especially for cases of smaller γ_3. By comparing the results obtained by methods Res-Aug-APOD and Res-Aug-APOD-W with the same parameters, we can obtain the similar conclusion. We pick out the cases where the methods with weighting strategy perform better than those without the weighting strategy, and show those results in Table 5. Anyway, from our tests we can see that the weighting strategy can improve the effect of the POD modes sometimes, especially for smaller γ_3.

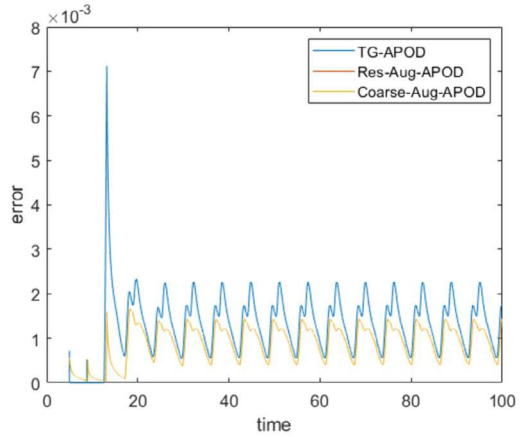
For the methods TG-APOD and Coarse-Aug-APOD, we have done more tests with some other η_0 and different coarse meshes. Some results for cases of $\epsilon = 0.01$ and $\epsilon = 0.005$ are provided in Appendix A.



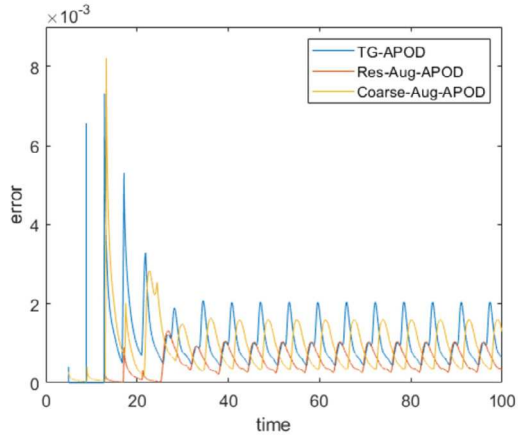
(a) $\epsilon = 1$.



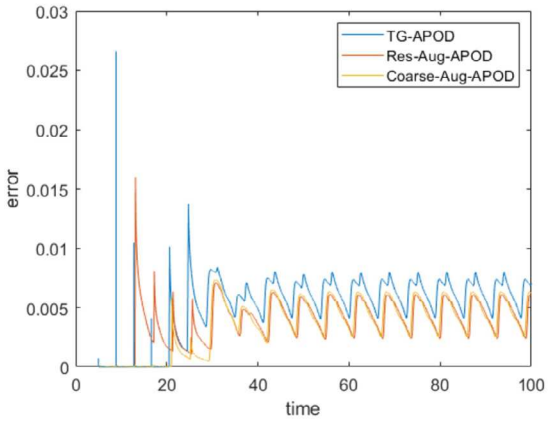
(b) $\epsilon = 0.5$.



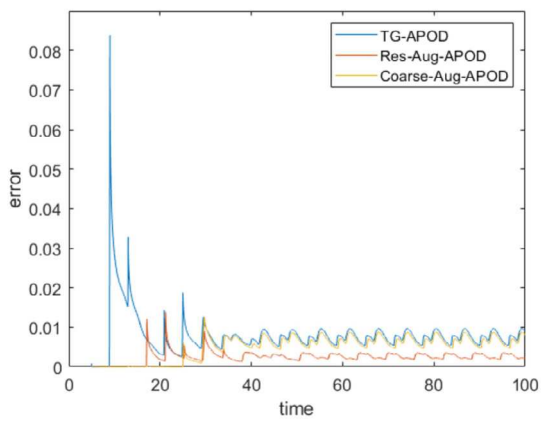
(c) $\epsilon = 0.1$.



(d) $\epsilon = 0.05$.



(e) $\epsilon = 0.01$.



(f) $\epsilon = 0.005$.

Fig. 3. The evolution curves of the error for solution of (26) with different ϵ by methods TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD, respectively.

4.2. Arnold-Beltrami-Childress (ABC) flow

We then consider another advection-diffusion equation with the advection being the ABC flow, which plays an important role in fluid dynamics [18,60,9].

$$\begin{cases} u_t - \epsilon\Delta u + \mathbf{B}(x,y,z,t)\cdot\nabla u = f(x,y,z,t), \quad (x,y,z)\in\Omega, t\in[0,T], \\ u(x,y,z,0) = 0, \\ u(x+2\pi,y,z,t) = u(x,y+2\pi,z,t) = u(x,y,z+2\pi,t) = u(x,y,z,t), \end{cases} \tag{27}$$

where

$$\begin{aligned} \mathbf{B}(x,y,z,t) =& (\sin(z+\sin\omega t)+\cos(y+\sin\omega t), \sin(x+\sin\omega t) \\ &+\cos(z+\sin\omega t), \sin(y+\sin\omega t)+\cos(x+\sin\omega t)), \\ f(x,y,z,t) =& -\sin(z+\sin\omega t)-\cos(y+\sin\omega t), \\ \Omega =& [0,2\pi]^3, T = 100. \end{aligned}$$

For this example, we also test 6 different cases with $\epsilon = 1, 0.5, 0.1, 0.05, 0.01, 0.005$, respectively.

Similar to the example of Kolmogorov flow, we first divide Ω into 6 tetrahedrons to act as the initial mesh. Then we refine the initial mesh 23 times uniformly using bisection to obtain the fine mesh for cases of $\epsilon = 1, 0.5, 0.1, 0.05$, and refine the initial mesh 24 times to obtain the fine mesh for cases of $\epsilon = 0.01, 0.005$. We set $\omega = 1.0$, $\delta t = 5\times 10^{-3}$ and choose the piecewise linear function as the finite element basis. For the POD type methods, we choose the same parameters as those for solving (26) expect the following settings. For the methods TG-APOD and Coarse-Aug-APOD, we refine the initial mesh 14 times to obtain the coarse mesh for cases of $\epsilon = 1, 0.5, 0.1$, and refine the initial mesh 15 times to obtain the coarse mesh for cases of $\epsilon = 0.05, 0.01$ and refine the initial mesh 17 times to obtain the coarse mesh for case of $\epsilon = 0.005$. The time steps corresponding to the coarse finite element spaces are 0.2, 0.125 and 0.05 respectively. We use the similar number of processors as those for the Kolmogorov flow to do the simulation, that is, we use 72 processors and 180 processors for cases of the fine mesh being obtained by refining the initial mesh 23 times and 24 times, respectively.

Some numerical results obtained by different methods are shown in Table 6. The notations in Table 6 have the same meanings as those in Table 4.

Similar to the first example, from Table 6, we see that the POD type methods can save much CPU time compared with the finite element method. We can also see that the results obtained by the standard POD method are unreliable as the decrease of ϵ. Besides, the three adaptive POD methods can obtain results with high accuracy even for the case of $\epsilon = 0.005$ with approximately a half of the CPU time compared with the standard finite element method. This shows that adaptive POD methods behave much better than the standard POD method.

We now compare the three adaptive POD methods, TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD. We see from Table 6 that the both method Coarse-Aug-APOD and method Res-Aug-APOD can achieve higher accuracy than the method TG-APOD with similar degrees of freedoms, especially for cases of ϵ being very close to 0. The method Coarse-Aug-APOD can achieve higher accuracy than the other two methods with similar or even less CPU time cost for cases of $\epsilon = 0.05, 0.01, 0.005$. Therefore, the method Coarse-Aug-APOD is the most accurate and efficient one.

Then we see the behavior of the weighting strategy. By comparing the results obtained by methods Res-Aug-APOD-W and Coarse-Aug-APOD-W with those obtained by methods Res-Aug-APOD and Coarse-Aug-APOD respectively, we see that the weighting strategy can improve the effect of the POD modes sometimes. Specifically, for cases of $\epsilon = 0.01, 0.005$, the method Res-Aug-APOD-W can obtain results with higher accuracy but less CPU time cost than the method Res-Aug-APOD.

Similar to the first example, we show the error obtained by the methods TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD in Fig. 4. The x-axis of each figure is the time, the y-axis is the relative error of numerical solution obtained by the adaptive POD methods.

From Fig. 4, we see clearly that the methods Res-Aug-APOD and Coarse-Aug-APOD can obtain results with higher accuracy than those obtained by the method TG-APOD over the entire time interval for cases of $\epsilon = 0.05, 0.01, 0.005$. From these comparisons, we can also see that our new methods Res-Aug-APOD and Coarse-Aug-APOD behave much better than the method TG-APOD.

Similarly, we have tested using some other η_0 and different coarse meshes to do the simulation, and some more numerical results are provided in Appendix A.

5. Concluding remarks

In this paper, we have proposed an augmented subspace based strategy for developing an error indicator for the POD approximation of some time dependent partial differential equations. Based on this strategy, we obtain a general framework for the augmented subspace based adaptive POD method. Besides, we have provided two strategies for augmenting the POD subspace, one is using the residual corresponding to the current POD approximation to augment the subspace, the other is using the approximation obtained on a coarse grid to augment the subspace.

We have used some numerical experiments for two typical 3D advection-diffusion equations, one with the advection being the Kolmogorov flow and the other one with the advection being the ABC flow, to show the efficiency of our new approach. Numerical

Table 6
The results of (27) with different ϵ obtained by methods FEM, POD, TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD, Res-Aug-APOD-W and Coarse-Aug-APOD-W, respectively.

ϵ	Method	η_0	Update Times	DOFs	Error	Average Error	Time (s)
1	FEM	-	-	10485760	-	-	20427.20
	POD	-	-	24	0.000974	0.013559	1681.67
	TG-APOD	3×10^{-3}	1	17	0.000335	0.000517	2304.13
	Res-Aug-APOD	1×10^{-5}	1	17	0.000755	0.000585	3149.87
	Coarse-Aug-APOD	5×10^{-6}	1	17	0.000341	0.000522	2340.55
	Res-Aug-APOD-W	1×10^{-5}	1	17	0.000755	0.000585	3133.89
	Coarse-Aug-APOD-W	5×10^{-6}	1	17	0.000341	0.000522	2344.09
0.5	FEM	-	-	10485760	-	-	18432.76
	POD	-	-	26	0.007569	0.069137	1625.80
	TG-APOD	1×10^{-3}	2	28	0.000145	0.000234	3044.43
	Res-Aug-APOD	1×10^{-5}	2	28	0.000279	0.000445	3835.46
	Coarse-Aug-APOD	5×10^{-6}	2	28	0.000154	0.000244	3064.83
	Res-Aug-APOD-W	1×10^{-5}	2	28	0.000279	0.000445	3865.66
	Coarse-Aug-APOD-W	5×10^{-6}	2	28	0.000154	0.000244	3018.50
0.1	FEM	-	-	10485760	-	-	17362.39
	POD	-	-	35	0.136214	0.373726	1727.87
	TG-APOD	1×10^{-3}	4	68	0.001716	0.001258	4940.78
	Res-Aug-APOD	1×10^{-5}	4	67	0.001167	0.000842	5630.27
	Coarse-Aug-APOD	5×10^{-6}	4	67	0.001767	0.000998	4889.35
	Res-Aug-APOD-W	1×10^{-5}	4	67	0.001167	0.000842	5630.14
	Coarse-Aug-APOD-W	5×10^{-6}	4	67	0.001767	0.000998	4805.39
0.05	FEM	-	-	10485760	-	-	17391.44
	POD	-	-	40	0.257052	0.514289	1814.08
	TG-APOD	3×10^{-3}	5	100	0.003220	0.002760	6209.27
	Res-Aug-APOD	5×10^{-5}	5	99	0.001698	0.001797	7080.34
	Coarse-Aug-APOD	5×10^{-6}	5	99	0.001352	0.001507	6222.50
	Res-Aug-APOD-W	5×10^{-5}	5	98	0.001711	0.001805	7031.27
	Coarse-Aug-APOD-W	5×10^{-6}	5	99	0.001352	0.001507	6093.91
0.01	FEM	-	-	16777216	-	-	16065.81
	POD	-	-	44	0.577333	0.697981	1515.37
	TG-APOD	3×10^{-3}	6	192	0.015234	0.012518	6640.86
	Res-Aug-APOD	8×10^{-4}	6	197	0.016227	0.012242	8117.78
	Coarse-Aug-APOD	8×10^{-6}	6	193	0.014533	0.008428	6808.99
	Res-Aug-APOD-W	8×10^{-4}	6	193	0.014695	0.010914	7953.35
	Coarse-Aug-APOD-W	8×10^{-6}	6	190	0.014493	0.008460	6923.94
0.005	FEM	-	-	16777216	-	-	16836.62
	POD	-	-	44	0.812194	0.836567	1554.84
	TG-APOD	5×10^{-3}	9	323	0.017107	0.010361	10735.34
	Res-Aug-APOD	1×10^{-3}	7	262	0.014846	0.009793	11758.40
	Coarse-Aug-APOD	1×10^{-5}	7	258	0.010908	0.007093	9406.65
	Res-Aug-APOD-W	1×10^{-3}	7	258	0.013419	0.009468	11285.76
	Coarse-Aug-APOD-W	1×10^{-5}	7	256	0.010918	0.007077	9395.15

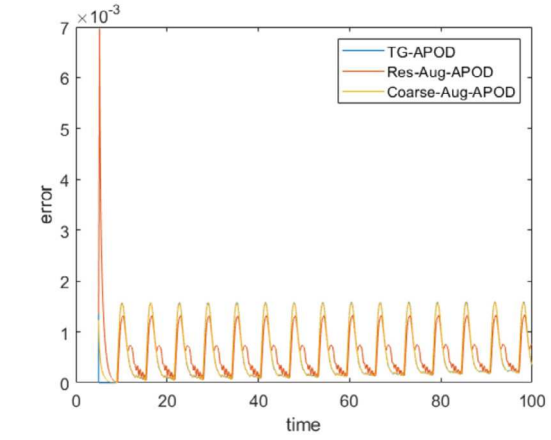
results show that both the residual type adaptive POD method and the coarse grid approximation type adaptive POD method are more efficient than the existing adaptive methods, especially for cases with small ϵ. For the two methods we proposed, the coarse-grid approximation type adaptive POD method has been shown to be more efficient. Besides, we have also introduced the weighting strategy to the update of POD modes. In future, we will pay more effort on the strict numerical analysis and the application to other types of time dependent partial differential equations as well as the construction of some other effective error indicators.

CRediT authorship contribution statement

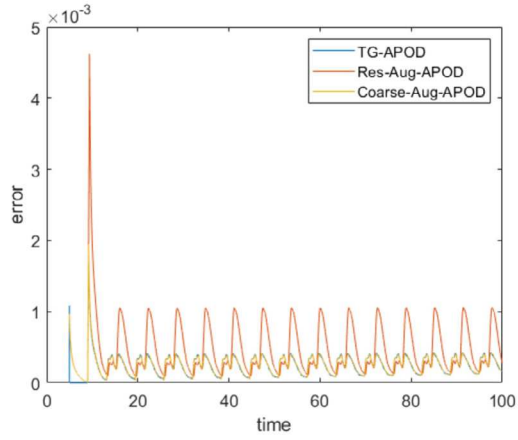
Xiaoying Dai: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Writing – original draft. **Miao Hu:** Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft. **Jack Xin:** Conceptualization, Funding acquisition, Methodology, Supervision, Validation, Writing – review & editing. **Aihui Zhou:** Conceptualization, Funding acquisition, Methodology, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

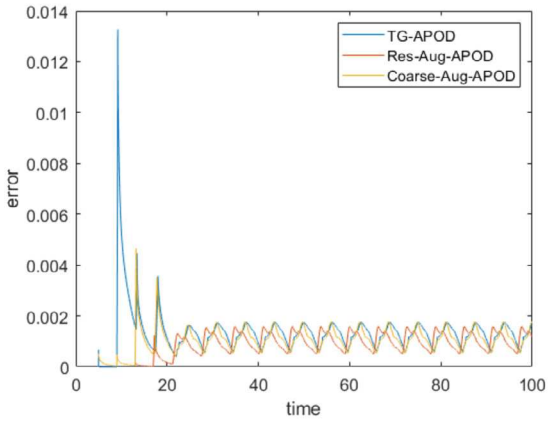
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.



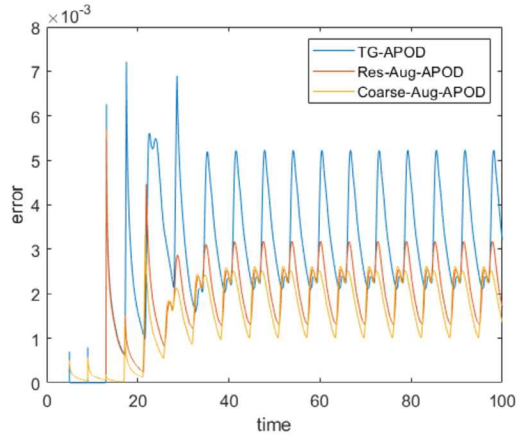
(a) $\epsilon = 1$.



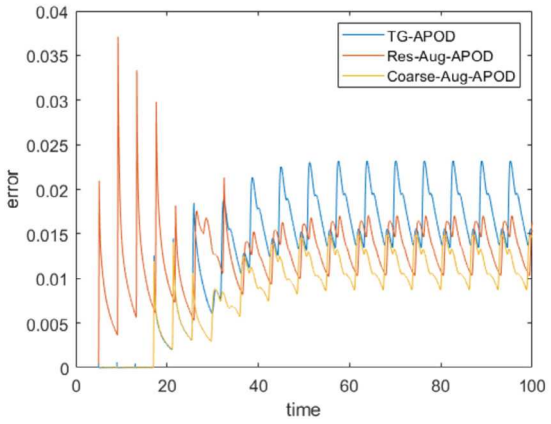
(b) $\epsilon = 0.5$.



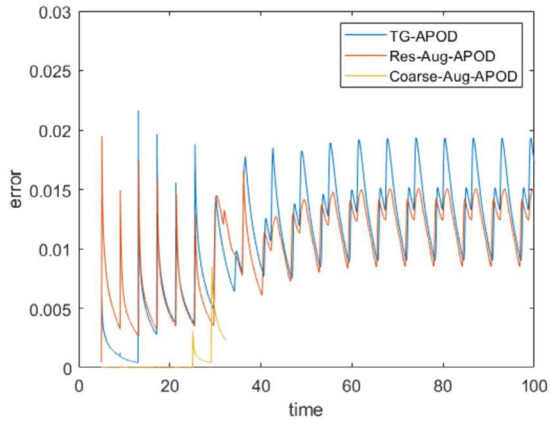
(c) $\epsilon = 0.1$.



(d) $\epsilon = 0.05$.



(e) $\epsilon = 0.01$.



(f) $\epsilon = 0.005$.

Fig. 4. The evolution curves of the error for solution of (27) with different ϵ by methods TG-APOD, Res-Aug-APOD and Coarse-Aug-APOD, respectively.

Table A.7
Time steps corresponding to different coarse meshes.

N-Refine	14	15	16	17
Time step	0.2	0.125	0.1	0.05

Table A.8
The results of (26) with $\epsilon = 0.01$ obtained by different POD type methods with different coarse meshes and η_0, respectively.

ϵ	Method	η_0	Update Times	DOFs	Error	Average Error	Time (s)
0.01	FEM	-	-	16777216	-	-	11784.99
	POD	-	-	43	0.919008	0.777626	1267.46
	TG-APOD-14	1×10^{-3}	6	167	0.009079	0.011065	5180.53
	TG-APOD-15	1×10^{-3}	6	166	0.007042	0.004837	5072.23
	TG-APOD-16	1×10^{-3}	7	196	0.005060	0.003834	5789.31
	Coarse-Aug-APOD-14	5×10^{-5}	5	149	0.022220	0.015238	4453.28
	Coarse-Aug-APOD-14	1×10^{-5}	6	173	0.008892	0.004957	5302.05
	Coarse-Aug-APOD-15	5×10^{-5}	5	147	0.009849	0.009620	4560.84
	Coarse-Aug-APOD-15	1×10^{-5}	6	172	0.006208	0.003351	5280.25
	Coarse-Aug-APOD-16	5×10^{-5}	5	147	0.008976	0.009054	4557.52
	Coarse-Aug-APOD-16	1×10^{-5}	6	172	0.006253	0.003377	5335.85

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors would like to thank the anonymous referees for their constructive suggestions which enrich the contents, including the contents about the residual type augmented subspace adaptive POD method and the weighting strategy for updating the POD modes.

Appendix A. Numerical experiments for different coarse meshes and different η_0

In this section, we use more numerical experiments to illustrate that our new method performs better than the two-grid based adaptive POD method, especially for cases of $\epsilon = 0.01, 0.005$. We denote the methods TG-APOD and Coarse-Aug-APOD with the initial mesh being refined N times as TG-APOD-N and Coarse-Aug-APOD-N, respectively. The time steps corresponding to different coarse finite spaces are listed in Table A.7. With a fixed fine grid, we have tested (26) with $\epsilon = 0.01, 0.005$ and (27) with $\epsilon = 0.005$ by different coarse meshes and different threshold η_0, respectively.

A.1. Kolmogorov flow with $\epsilon = 0.01$

Some numerical results of (26) with $\epsilon = 0.01$ obtained by different methods and different parameters are shown in Table A.8. From Table A.8, we can first see that all adaptive POD methods can obtain results with much higher accuracy than the standard POD method, for all different choices for the coarse mesh and the tolerance η_0. For the method TG-APOD, the finer the coarse mesh, the better the approximation, and of course, the higher the computational cost. While for the method Coarse-Aug-APOD, we see that if we reduce the size of the coarse mesh properly, the accuracy of the approximation will be improved. However, a further reduction of the size of coarse mesh may not improve the accuracy of the approximation obviously. Besides, we can see from these results that, different from the method TG-APOD, the computational cost may not change a lot as the increase of the size for the coarse mesh. We can also see that using the same coarse mesh, our method Coarse-Aug-APOD can obtain results with a little higher accuracy than those obtained by the method TG-APOD, while costing similar CPU time.

A.2. Kolmogorov flow with $\epsilon = 0.005$

Some numerical results of (26) with $\epsilon = 0.005$ obtained by different methods and different parameters are shown in Table A.9. For this case of ϵ, we can also get the similar conclusion as that from Table A.8, that is, the method Coarse-Aug-APOD can obtain results with a little higher accuracy than those obtained by the method TG-APOD.

Table A.9
The results of (26) with $\epsilon = 0.005$ obtained by different POD type methods with different coarse meshes and η_0, respectively.

ϵ	Method	η_0	Update Times	DOFs	Error	Average Error	Time (s)
0.005	FEM	-	-	16777216	-	-	12526.50
	POD	-	-	45	1.010868	0.871304	1306.83
	TG-APOD-14	1×10^{-3}	6	200	0.022221	0.024539	5727.87
	TG-APOD-15	1×10^{-3}	7	227	0.012810	0.009396	6811.02
	TG-APOD-16	1×10^{-3}	7	235	0.008957	0.007559	6661.74
	Coarse-Aug-APOD-14	5×10^{-5}	5	176	0.018007	0.018023	4893.47
	Coarse-Aug-APOD-14	1×10^{-5}	6	207	0.014098	0.007749	5696.05
	Coarse-Aug-APOD-15	5×10^{-5}	5	176	0.017100	0.017024	5084.80
	Coarse-Aug-APOD-15	1×10^{-5}	7	239	0.008122	0.004791	6891.73
	Coarse-Aug-APOD-16	5×10^{-5}	5	176	0.017162	0.016912	5295.97
	Coarse-Aug-APOD-16	1×10^{-5}	7	239	0.007957	0.004724	6778.84

Table A.10
The results of (27) with $\epsilon = 0.005$ obtained by different POD type methods with different coarse meshes and η_0, respectively.

ϵ	Method	η_0	Update Times	DOFs	Error	Average Error	Time (s)
0.005	FEM	-	-	16777216	-	-	16836.62
	POD	-	-	44	0.812194	0.836567	1554.84
	TG-APOD-15	5×10^{-3}	6	226	0.027286	0.023730	7147.38
	TG-APOD-16	5×10^{-3}	7	259	0.025176	0.020051	8409.76
	TG-APOD-17	5×10^{-3}	9	323	0.017107	0.010361	10735.34
	Coarse-Aug-APOD-15	5×10^{-5}	4	159	0.065712	0.051971	5581.35
	Coarse-Aug-APOD-15	1×10^{-5}	5	192	0.026378	0.027074	6517.28
	Coarse-Aug-APOD-15	5×10^{-6}	11	390	0.001740	0.003459	12508.26
	Coarse-Aug-APOD-16	5×10^{-5}	5	192	0.032155	0.031447	6468.33
	Coarse-Aug-APOD-16	1×10^{-5}	9	323	0.011047	0.014037	10392.84
	Coarse-Aug-APOD-16	5×10^{-6}	12	421	0.001581	0.003247	14281.87
	Coarse-Aug-APOD-17	5×10^{-5}	5	192	0.025166	0.026153	7374.91
	Coarse-Aug-APOD-17	1×10^{-5}	7	258	0.010908	0.007093	9406.65
	Coarse-Aug-APOD-17	5×10^{-6}	11	389	0.001429	0.001716	12466.38

A.3. ABC flow with $\epsilon = 0.005$

In Table A.10, we show some results of (27) with $\epsilon = 0.005$ obtained by different methods and different parameters. Similar to the example of the Kolmogorov flow, we see from these results that the method Coarse-Aug-APOD can obtain results with a little higher accuracy than those obtained by the method TG-APOD.

References

[1] R.A. Adams, Sobolev Spaces, Academic Press, New York, 1975.
[2] M. Bakker, Simple groundwater flow models for seawater intrusion, in: Proceedings of SWIM16, WolinIsland, Poland, 2000, pp. 180–182.
[3] G.K. Batchelor, An Introduction to Fluid Dynamics, second edition, Cambridge University Press, Cambridge, 1999.
[4] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (2015) 483–531.
[5] P. Benner, P. Kürschner, J. Saak, Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations, Electron. Trans. Numer. Anal. 43 (2014) 142–162.
[6] V. Borue, S.A. Orszag, Numerical study of three-dimensional Kolmogorov flow at high Reynolds numbers, J. Fluid Mech. 306 (1996) 293–323.
[7] S. Boyaval, C. Le Bris, T. Lelievre, Y. Maday, N.C. Nguyen, A.T. Patera, Reduced basis techniques for stochastic problems, Arch. Comput. Methods Eng. 17 (2010) 435–454.
[8] S. Brenner, L. Scott, The Mathematical Theory of Finite Element Methods, third edition, Springer, New York, 2008.
[9] N.H. Brummell, F. Cattaneo, S.M. Tobias, Linear and nonlinear dynamo properties of time-dependent ABC flows, Fluid Dyn. Res. 28 (2001) 237–265.
[10] J. Burkardt, M. Gunzburger, H.C. Lee, POD and CVT-based reduced-order modeling of Navier-Stokes flows, Comput. Methods Appl. Mech. Eng. 196 (2006) 337–355.
[11] J.R. Cannon, The One-Dimensional Heat Equation, Addison-Wesley, Mento Park, 1984.
[12] S. Chellappa, L. Feng, P. Benner, Adaptive basis construction and improved error estimation for parametric nonlinear dynamical systems, Int. J. Numer. Methods Eng. 121 (2020) 5320–5349.
[13] F. Chinesta, A. Huerta, G. Rozza, K. Willcox, Model Reduction Methods, second edition, Encyclopedia of Computational Mechanics, 2017, pp. 1–36.
[14] X. Dai, X. Kuang, Z. Liu, J. Xin, A. Zhou, An adaptive proper orthogonal decomposition Galerkin method for time dependent problems, preprint 2017.
[15] X. Dai, X. Kuang, J. Xin, A. Zhou, Two-grid based adaptive proper orthogonal decomposition method for time dependent partial differential equations, J. Sci. Comput. 84 (2020) 1–27.
[16] X. Dai, J. Xu, A. Zhou, Convergence and optimal complexity of adaptive finite element eigenvalue computations, Numer. Math. 110 (2008) 313–355.
[17] B.T. Dickinson, J.R. Singler, Nonlinear model reduction using group proper orthogonal decomposition, Int. J. Numer. Anal. Model. 7 (2010) 356–372.
[18] T. Dombre, U. Frisch, J.M. Greene, M. Hénon, A. Mehr, A.M. Soward, Chaotic streamlines in the ABC flows, J. Fluid Mech. 167 (1986) 353–391.
[19] P. Druault, J. Delville, J.P. Bonnet, Proper orthogonal decomposition of the mixing layer flow into coherent structures and turbulent Gaussian fluctuations, C. R., Méc. 333 (2005) 824–829.

[20] G.H. Golub, C.F. van Loan, Matrix Computations, fourth edition, JHU Press, 2013.
[21] C. Gräßle, M. Hinze, POD reduced-order modeling for evolution equations utilizing arbitrary finite element discretizations, Adv. Comput. Math. 44 (2018) 1941–1978.
[22] A. Gu, J. Xin, Z. Zhang, Error estimates for a POD method for solving viscous G-equation in incompressible cellular flows, SIAM J. Sci. Comput. 43 (2021) A636–A662.
[23] Y. He, The Euler implicit/explicit scheme for the 2D time-dependent Navier-Stokes equations with smooth or non-smooth initial data, Math. Comput. 77 (2008) 2097–2124.
[24] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, vol. DXC, Springer, 2016.
[25] P. Holmes, J.L. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, Cambridge, 1996.
[26] C. Homescu, L.R. Petzold, R. Serban, Error estimation for reduced-order models of dynamical systems, SIAM J. Numer. Anal. 43 (2005) 1693–1714.
[27] T. Katayama, H. Kawauchi, G. Picci, Subspace identification of closed loop systems by the orthogonal decomposition method, Atomatica 41 (2005) 863–872.
[28] M. Khalil, S. Adhikari, A. Sarkar, Linear system identification using proper orthogonal decomposition, Mech. Syst. Signal Process. 21 (2007) 3123–3145.
[29] B. Koc, S. Rubino, M. Schneier, J. Singler, T. Iliescu, On optimal pointwise in time error bounds and difference quotients for the proper orthogonal decomposition, SIAM J. Numer. Anal. 59 (2021) 2163–2196.
[30] D. Kosloff, R. Kosloff, A Fourier method solution for the time dependent Schrödinger equation as a tool in molecular dynamics, J. Comput. Phys. 52 (1983) 35–53.
[31] K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for parabolic problems, Numer. Math. 90 (2001) 117–148.
[32] K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics, SIAM J. Numer. Anal. 40 (2002) 492–515.
[33] K. Kunisch, S. Volkwein, Optimal snapshot location for computing POD basis functions, ESAIM: Math. Model. Numer. Anal. 44 (2010) 509–529.
[34] H. Li, Z. Song, A reduced-order finite element method based on proper orthogonal decomposition for the Allen-Cahn model, J. Math. Anal. Appl. 500 (2021) 125103.
[35] K. Li, T. Huang, L. Li, S. Lanteri, POD-based model order reduction with an adaptive snapshot selection for a discontinuous Galerkin approximation of the time-domain Maxwell's equations, J. Comput. Phys. 396 (2019) 106–128.
[36] S. Locke, J. Singler, New proper orthogonal decomposition approximation theory for PDE solution data, SIAM J. Numer. Anal. 58 (2020) 3251–3285.
[37] H.V. Ly, H.T. Tran, Modeling and control of physical processes using proper orthogonal decomposition, Math. Comput. Model. 33 (2001) 223–236.
[38] J. Lyu, Z. Wang, J. Xin, Z. Zhang, A convergent interaction particle method and computation of KPP front speeds in chaotic flows, SIAM J. Numer. Anal. 60 (2022) 1136–1167.
[39] J. Lyu, J. Xin, Y. Yu, Computing residual diffusivity by adaptive basis learning via spectral method, Numer. Math. Theory Methods Appl. 10 (2017) 351–372.
[40] Y. Maday, Reduced basis method for the rapid and reliable solution of partial differential equations, in: Proceedings of International Conference of Mathematicians, European Mathematical Society, vol. III, 2006, pp. 1255–1270.
[41] P.A. Markowych, C.A. Ringhofer, C. Schmeiser, Semiconductor Equations, Springer, Vienna, 1990.
[42] J. Nolen, M. Rudd, J. Xin, Existence of KPP fronts in spatially-temporally periodic advection and variational principle for propagation speeds, Dyn. Partial Differ. Equ. 2 (2005) 1–24.
[43] A.M. Obukhov, Kolmogorov flow and its laboratory simulation, Rus. Uspekhi Mat. Nauk 38 (1983) 101–111.
[44] R. Padhi, S.N. Balakrishnan, Proper orthogonal decomposition based optimal neurocontrol synthesis of a chemical reactor process using approximate dynamic programming, Neural Netw. 16 (2003) 719–728.
[45] B. Peherstorfer, K. Willcox, Dynamic data-driven reduced-order models, Comput. Methods Appl. Mech. Eng. 291 (2015) 21–41.
[46] PETSc, http://www.mcs.anl.gov/petsc/.
[47] PHG, http://lsec.cc.ac.cn/phg/.
[48] R. Pinnau, Model reduction via proper orthogonal decomposition, in: Model Order Reduction: Theory, Research Aspects and Applications, Springer, Berlin, Heidelberg, 2008.
[49] A. Quarteroni, G. Rozza, Reduced Order Methods for Modeling and Computational Reduction, vol. IX, Springer, Berlin, 2014.
[50] M.L. Rapún, F. Terragni, J.M. Vega, Adaptive POD-based low-dimensional modeling supported by residual estimates, Int. J. Numer. Methods Eng. 104 (2015) 844–868.
[51] C. Rowley, T. Colonius, R. Murray, Model reduction for compressible flows using POD and Galerkin projection, Physica D 189 (2004) 115–129.
[52] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[53] L. Sirovich, Turbulence and the dynamics of coherent structures, part I: coherent structures, Q. Appl. Math. 45 (1987) 561–571.
[54] J. Strikwerda, Finite Difference Schemes and Partial Differential Equations, SIAM, Philadelphia, 2004.
[55] M.V. Tabib, J.B. Joshi, Analysis of dominant flow structures and their flow dynamics in chemical process equipment using snapshot proper orthogonal decomposition technique, Chem. Eng. Sci. 63 (2008) 3695–3715.
[56] F. Terragni, J.M. Vega, Simulation of complex dynamics using POD 'on the fly' and residual estimates, in: Dynamical Systems, Differential Equations and Applications AIMS Proceedings, 2015, pp. 1060–1069.
[57] S. Volkwein, Model Reduction Using Proper Orthogonal Decomposition, Lecture Notes, Institute of Mathematics and Scientific Computing, vol. MXXV, University of Graz, Graz, 2011.
[58] S. Volkwein, Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling, Lecture Notes, Department of Mathematics and Statistics, University of Konstanz, 2013.
[59] Z. Wang, J. Xin, Z. Zhang, Sharp uniform in time error estimate on a stochastic structure-preserving Lagrangian method and computation of effective diffusivity in 3D chaotic flows, SIAM Multiscale Model. Simul. 19 (2021) 1167–1189.
[60] J. Xin, Y. Yu, A. Zlatos, Periodic orbits of the ABC flow with A = B = C = 1, SIAM J. Math. Anal. 48 (2016) 4087–4093.
[61] C. Xu, Y. Ou, E. Schuster, Sequential linear quadratic control of bilinear parabolic PDEs based on POD model reduction, Automatica 47 (2011) 418–426.
[62] P. Zu, L. Chen, J. Xin, A computational study of residual KPP front speeds in time-periodic cellular flows in the small diffusion limit, Physica D 311 (2015) 37–44.