Networks and Heterogeneous Media

NHM, 19(1): 86–105.
DOI: 10.3934/nhm.2024004
Received: 25 July 2023
Revised: 29 November 2023
Accepted: 20 December 2023
Published: 15 January 2024

http://www.aimspress.com/journal/nhm

Research article

A discontinuous Galerkin Method based on POD model reduction for Euler equation

Lan Zhu [1], Li Xu [1, 2,*], Jun-Hui Yin[1], Shu-Cheng Huang[1] and Bin Li[1]

[1] National Key Laboratory of Science and Technology on Vacuum Electronics, School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[2] Shenzhen Institute for Advanced Study, UESTC Yinxing Zhijie Phase II, Guanlan Street, Longhua District, Shenzhen, China

* **Correspondence:** Email: lixu@uestc.edu.cn; Tel: +86-028-8320-3371; Fax: +86-028-8320-3371.

Abstract: This paper considers the work of combining the proper orthogonal decomposition (POD) reduced-order method with the discontinuous Galerkin (DG) method to solve three-dimensional time-domain Euler equations. The POD-DG formulation is established by constructing the POD base vector space, based on POD technology one can apply the Galerkin projection of the DG scheme to this dimension reduction space for calculation. Its overall goal is to overcome the disadvantages of high computational cost and memory requirement in the DG algorithm, reduce the degrees of freedom (DOFs) of the calculation model, and save the calculation time while ensuring acceptable accuracy. Numerical experiments verify these advantages of the proposed POD-DG method.

Keywords: computational fluid dynamics; Euler equations; discontinuous Galerkin method; proper orthogonal decomposition; model reduction method

1. Introduction

With the emergence of high-performance computers and the development of numerical algorithms for solving physical problems accurately, computational fluid dynamics (CFD) has been developed, which is a more economical and effective method to simulate and analyze hydrodynamics problems. In the process of solving practical problems with CFD, we set up corresponding

mathematical models to describe the practical problems. The discontinuous Galerkin (DG) method, also known as the discontinuous finite element method [1–5], uses completely discontinuous piecewise polynomial space as the approximate solution. At present, this method is widely used in many fields, such as computational aerodynamics, computational acoustics, computational electromagnetics [6], and so on. DG method not only has the characteristics of local conservation and stability, but also can improve the accuracy by increasing the order of interpolation function, so it is easy to deal with complex geometry and irregular meshes with suspended nodes and has different degrees of approximation polynomials in different elements. These attributes make it better to combine with the HP adaptive method. DG method not only maintains the advantages of the finite element method [7,8], finite volume method [9,10], and finite difference method [11,12] but also overcomes their shortcomings.

However, compared with the finite volume method, the number of variables needed to be solved by the DG method in each element increases and the increase is non-linear with the improvement of accuracy. It will derive a large set of partial differential equations (PDEs) when simulating the numerical system, which leads to too many degrees of freedom in the calculation, along with a lot of calculation time and memory capacity. Besides, because of its high dimension or complexity, it is relatively difficult to simulate directly. In this context, it is particularly important to reduce the scale or order of the model effectively.

The model order reduction (MOR) method [13–17] is an effective approximation method that can greatly reduce the dimension of the model by reducing the DOFs of numerical simulation. It is a practical way for deduction of high even infinite dimensional alternative models related to Galerkin projection. Research shows that some numerical methods combined with model reduction in solving PDEs can be very efficient as they can reduce computational load and memory requirements [18]. In recent years, more and more model reduction methods have been developed rapidly, such as the Krylov subspace method based on Padé approximation, proper orthogonal decomposition method (POD), and balanced truncation method. Among these reduced-order techniques, the POD method [19–23] is the most commonly used and effective reduced-order method in simulating the physical process controlled by PDEs. This is an effective data analysis method, whose goal is to approximate the multi-dimensional physical process in a low-dimensional way and then greatly reduce the amount of needed data. The purpose is to improve computing efficiency by approaching the original model with a reduced-order model. Loeve and Karhunen first introduced the POD method in 1945 and 1946. It starts with extracting a group of instantaneous image vectors from high fidelity numerical simulation experiments and then obtains POD basis by generating the characteristic system of correlation matrix from snapshot matrix, in which snapshot matrix is listed as snapshot vector. It finds the optimal low-dimensional approximation from the given data. The POD method has been extensively and successfully employed in many fields such as optimal control, signal analysis, and so on. Recently, some POD-based reduced-order numerical methods, such as the POD Galerkin reduced-order model [24], POD finite element reduced-order model [25], POD finite volume element reduced-order model [26], POD reduced-order models based on isogeometric analysis [27], smooth B-splines [28] and NURBS basis functions [29] have been developed to calculate PDEs to reduce computational costs. The experimental data shows that the DOFs become very little with the POD method, so the calculation time and the accumulation of truncation errors can be reduced. At the same time, the theoretical and numerical errors are also very close to the original method, so it still ensures the accuracy of the calculation.

To overcome the large amount of computational cost caused by the DG method and improve computational efficiency, this paper studies the DG method based on the POD dimension reduction

model to solve three-dimensional Euler equations [30–32]. Combining with POD technology, the dimension reduction model of the original problem is reconstructed, and then we can calculate in a low dimensional space. In this paper, by constructing POD base vector space (also known as the optimal orthogonal basis functions) through transient solution with DG formula, which is a data set achieved by eigenorthogonal decomposition or singular value decomposition (SVD) [33], the model is projected to dimension reduction space for calculation. Compared with the original DG formulation, the proposed POD-DG formulation reduces the degree of freedom of the calculation model and truncation error accumulation, as well as calculation cost and calculation time while maintaining the original DG accuracy of the numerical solution, thus improving the computational efficiency.

The rest of the paper is organized as follows. In Section 2, we present the three-dimensional Euler equations and introduce the formulation of the DG method. In Section 3, we propose the POD method and deduce its implementation combined with the DG method. Section 4 shows some numerical results for the comparison between the proposed POD-DG method and the traditional DG method to verify the greater computational performance of the POD-DG method. Finally, we conclude Section 5.

2. Problem statement and discontinuous Galerkin formulation

To illustrate the proposed POD-DG formulation conveniently, we introduce the governing equations and briefly derive the traditional DG method's derivation.

2.1. Flow control equation

In Cartesian coordinates, the conservative form of three-dimensional Euler equations is as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \vec{\boldsymbol{F}}(\boldsymbol{U}) = 0\,, \tag{2.1}$$

which is defined in the computational domain Ω with domain boundary $\partial\Omega$; where $\boldsymbol{U}$ is the conservative variable, $\vec{\boldsymbol{F}}(\boldsymbol{U}) = (\boldsymbol{F}^x, \boldsymbol{F}^y, \boldsymbol{F}^z)$ is inviscid flux and its form is as follows:

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix},\ \boldsymbol{F}^x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)u \end{bmatrix},\ \boldsymbol{F}^y = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (\rho E + p)v \end{bmatrix},\ \boldsymbol{F}^z = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (\rho E + p)w \end{bmatrix}, \tag{2.2}$$

where ρ, p are density and pressure respectively; u, v, w are velocity components in Cartesian coordinates; e, E are inner energy and total energy separately;

$$E = e + \frac{u^2 + v^2 + w^2}{2}\,. \tag{2.3}$$

For complete gases, there are:

$$R = c_p - c_v,\ \gamma = \frac{c_p}{c_v},\ p = \rho RT = (\gamma - 1)(\rho E - \frac{u^2 + v^2 + w^2}{2})\,, \tag{2.4}$$

where R is gas constants, c_p, c_v, γ are specific heat at constant pressure, constant volume and specific heat ratio, respectively.

2.2. Discretization in space via DG method

Before spatial discretization, the solution region Ω is divided into a finite number of non-overlapping unstructured elements Ω_k. For three-dimensional problems, Ω_k can be a triangular prism, tetrahedron, hexahedron and pyramid. In this paper, we dispose of a partition of $\Omega \subset \mathbb{R}^3$ into a set of tetrahedron Ω_k,

$$\Omega \approx \bigcup_{k=1}^{N_k} \Omega_k \ ,$$

where N_k is the number of tetrahedral elements, and each element has four faces $\partial\Omega_k$. We introduce virtual elements on the solution domain boundary $\partial\Omega$ to take into account the boundary conditions. Assume that there is a virtual element Ω_v adjacent to the boundary element Ω_b by the boundary face $\partial\Omega_b$ (shown in Figure 1), then the boundary $\partial\Omega$ can be defined based on boundary conditions through virtual elements.

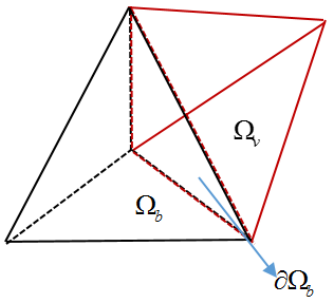


Figure 1. Boundary element and virtual element.

The purpose of DG algorithm is to look for an approximate solution $\boldsymbol{U}$ to satisfy the generalized system. We multiply Eq (2.1) by the test function $\phi_i(x,y,z)$ on the element Ω_k and take an integral:

$$\int_{\Omega_k} \frac{\partial \boldsymbol{U}}{\partial t} \phi_i d\Omega_k + \int_{\Omega_k} \nabla \cdot \vec{\boldsymbol{F}} \phi_i d\Omega_k = 0 \quad i = 1....N \ , \tag{2.5}$$

where N is the number of test function $\phi_i(x,y,z)$. Apply the Green-Gauss formula to the second terms, we have:

$$\int_{\Omega_k} \frac{\partial \boldsymbol{U}}{\partial t} \phi_i d\Omega_k + \int_{\partial\Omega_k} \vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}} \phi_i d\Gamma - \int_{\Omega_k} \vec{\boldsymbol{F}} \cdot \nabla \phi_i d\Omega_k = 0 \quad i = 1....N \ , \tag{2.6}$$

where $\vec{\boldsymbol{n}}$ is the outward unit normal vector on the surface of the tetrahedral element Ω_k, $\partial\Omega_k$ is surface of Ω_k. $\vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}}$ is the numerical flux on the interface shared by two neighboring elements Ω_{kl} and Ω_{kr}. It depends on $\vec{\boldsymbol{F}}_l$ and $\vec{\boldsymbol{F}}_r$, which are the conservative states at the left and right sides of the element interface. Because the values between two sides of the element boundary are different, they can be obtained by calculating the boundary flux in the DG method. There are some popular numerical

fluxes such as Roe flux, Lax–Friedrichs flux, and HLLC flux. In this paper, we take the HLLC flux [34–36] in our experiments.

Assuming that variables have piecewise polynomial distribution in elements:

$$U_k = \sum_{i=1}^{N_p} u_i^k(t)\phi_i^p(x,y,z), \tag{2.7}$$

where $u_i^k(t)$ (abbreviated as u_i) is the DOFs of element, $\phi_i^p(x,y,z)$ (abbreviated as ϕ_i^p) is the basis function of element, $N_p = (p+1)(p+2)(p+3)/6$ is the number of basis functions in three-dimensional case and p is the polynomial of degree. In this paper we employ the second order hierarchical scalar basis functions [37] to interpolate the field variables in the tetrahedral element to favorably carry out the higher order accuracy. Replacing the variables in Eq (2.6) with definitions from Eq (2.7) we have:

$$\frac{\partial \sum_{n=1}^{N_p} \boldsymbol{u}_n^k(t)}{\partial t} \int_{\Omega_k} \phi_n^p(x,y,z)\phi_i(x,y,z)d\Omega_k = \int_{\Omega_k} \vec{\boldsymbol{F}} \cdot \nabla \phi_i(x,y,z)d\Omega_k - \int_{\partial\Omega_k} \vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}}\phi_i(x,y,z)d\Gamma. \tag{2.8}$$

Take Gauss integral to two terms of the right side in Eq (2.8), we get volume integral and area integral respectively:

$$\begin{cases} \int_{\Omega_k} \vec{\boldsymbol{F}} \cdot \nabla \phi_i(x,y,z)d\Omega_k \approx \sum_{j=1}^{J} w_{\Omega_k}(j)\vec{\boldsymbol{F}} \cdot \nabla \phi_i(x,y,z)|\Omega_k|, \\ \int_{\partial\Omega_k} \vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}}\phi_i(x,y,z)d\Gamma \approx \sum_{l=1}^{L} w_{\partial\Omega_k}(l)\vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}}\phi_i(x,y,z)|\partial\Omega_k|, \end{cases} \tag{2.9}$$

where $w_{\Omega_k}(j)$ and $w_{\partial\Omega_k}(l)$ are Gaussian integral weights of volume and area, $|\partial\Omega_k|$ is the area of the integral surface, $|\Omega_k|$ is the volume of the element. Let

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{R}(\boldsymbol{u})_V - \boldsymbol{R}(\boldsymbol{u})_S = \int_{\Omega_k} \vec{\boldsymbol{F}} \cdot \nabla \phi_i(x,y,z)d\Omega_k - \int_{\partial\Omega_k} \vec{\boldsymbol{F}} \cdot \vec{\boldsymbol{n}}\phi_i(x,y,z)d\Gamma, \tag{2.10}$$

the generalized system is simplified as follows:

$$\boldsymbol{M} \cdot \frac{\partial \boldsymbol{u}_n(t)}{\partial t} = \boldsymbol{R}(\boldsymbol{u}) \tag{2.11}$$

where $\boldsymbol{M}_{mn} = \int_{\Omega} \phi_m(x,y,z)\phi_n(x,y,z)d\Omega$ is the mass matrix only related to element and coordinates types, $\boldsymbol{R}(\boldsymbol{u})$ is right hand side, $t \in [0, T_F]$ and $\boldsymbol{u} = (\rho, \rho u, \rho v, \rho w, \rho E)^T$ is solution vector solved by time progression.

2.3. Discretization in time via TVD-RK scheme

After space discretization, the system (2.11) becomes first order ordinary differential equations of time. These equations can be time advanced implicitly or explicitly, the steady solution can be obtained after iterative convergence. Here we propose to use the second-order explicit TVD-RK scheme [38,39] for time discretization, which has less computation and storage per time step. Divide the time span $[0, T_F]$ into N_t equal subintervals defined by

$$0 = t^0 < t^1 < \cdots < t^{N_t} = T_f,$$

where $t^n = n\Delta t\left(n \in \{0,1,\cdots,N_t\}\right)$. Δt is the time step size in reference [40].

$$\Delta t = \frac{V_k \cdot CFL}{(2p+1)\sum_e \max\left(\left|\vec{V}^l \cdot \boldsymbol{n}\right| + \boldsymbol{c}_k^l, \left|\vec{V}^r \cdot \boldsymbol{n}\right| + \boldsymbol{c}_k^r\right) \cdot S_e},$$

where V_k is the element volume, p is the order of basis function, l, r indicate the left and right sides of the element surface, c_k is the sound velocity at the center point of the element, $\vec{V}$ is the velocity at the interface of the element, S_e is the area of the four faces of the element, and $\boldsymbol{n}$ is the outer normal direction of the element boundary interface. Then the fully discrete scheme is given by

$$\begin{aligned}
\boldsymbol{u}_h^{(1)} &= \boldsymbol{u}_h^n + \Delta t \boldsymbol{M}^{-1}\boldsymbol{R}(\boldsymbol{u}_h^n) \\
\boldsymbol{u}_h^{(2)} &= \frac{3}{4}\boldsymbol{u}_h^n + \frac{1}{4}\left[\boldsymbol{u}_h^{(1)} + \Delta t \boldsymbol{M}^{-1}\boldsymbol{R}(\boldsymbol{u}_h^{(1)})\right] . \\
\boldsymbol{u}_h^{n+1} &= \frac{1}{3}\boldsymbol{u}_h^n + \frac{2}{3}\left[\boldsymbol{u}_h^{(2)} + \Delta t \boldsymbol{M}^{-1}\boldsymbol{R}(\boldsymbol{u}_h^{(2)})\right]
\end{aligned} \tag{2.12}$$

3. DG formulations based on POD

In this part, first we introduce the POD algorithm, and then combine it with the DG method. POD method uses the transient solution calculated by DG format to form data set, and then obtains the optimal orthogonal basis function by characteristic orthogonal decomposition. The DG formulation based on POD is established by constructing the POD base space.

3.1. Definition of snapshot matrix

Let $\{u_i(t_n)\}_{i=0}^{N_{Dof}}$ denote the set of N_{Dof} observations (also called snapshots) of some physical process (we take transient solution of DG scheme here) taken at time t_n. Choose $l\left(l << N_t\right)$ distributed snapshot vectors from the N_t transient solution $N_t\{u_i(t_n)\}_{n=0}^{N_t}$ with DG scheme (2.11), we can establish five snapshot matrices:

$$\boldsymbol{W}_u = \begin{pmatrix} u_{i1}(t_{n_1}) & u_{i1}(t_{n_2}) & \cdots & u_{i1}(t_{n_l}) \\ u_{i2}(t_{n_1}) & u_{i2}(t_{n_2}) & \cdots & u_{i2}(t_{n_l}) \\ \vdots & \vdots & \ddots & \vdots \\ u_{iN_{dof}}(t_{n_1}) & u_{iN_{dof}}(t_{n_2}) & \cdots & u_{iN_{dof}}(t_{n_l}) \end{pmatrix} \in C^{N_{dof}\times l}, u = \rho, \rho u, \rho v, \rho w, \rho E \cdot \tag{3.1}$$

Remark 3.1. The selection of snapshot vectors is very important to the numerical simulation results. In the calculation of actual problems, one can get samples with experiments data or interpolation, which means obtain snapshot set from actual physical processes. So $\{u_i(t_n)\}_{n=0}^{N_t}$ could be the experimental or previous results.

3.2. Construction of eigenvalue problem

Let

$$\boldsymbol{U}^{T}\boldsymbol{W}_{u}\boldsymbol{V}=\begin{pmatrix}\Sigma_{r\times r} & O_{r\times(l-r)}\\ O_{(N_{dof}-r)\times r} & O_{(N_{dof}-r)\times(l-r)}\end{pmatrix}, \tag{3.2}$$

where r is the rank of $\boldsymbol{W}_u$, $\Sigma_{r\times r}=diag(\sigma_1,\sigma_2,...,\sigma_r)$ are the singular values of $\boldsymbol{W}_u$ with reduced-order $\sigma_1\geq\sigma_2\geq...\geq\sigma_r>0$, $\boldsymbol{U}=(\xi_1,\xi_2,...,\xi_{N_{dof}})$ and $\boldsymbol{V}=(\varphi_1,\varphi_2,...,\varphi_l)$ are $N_{dof}\times N_{dof}$ and $l\times l$ unitary matrices, $\xi_i(i=1,2,...,r)$ are the eigenvectors of $\boldsymbol{W}_u\boldsymbol{W}_u^{T}$ corresponding to the descending order $\sigma_i(i=1,2,...,r)$. Similarly, $\varphi_i(i=1,2,...,r)$ are the eigenvectors of $\boldsymbol{W}_u^{T}\boldsymbol{W}_u$. As we all know that

$$\begin{cases}\boldsymbol{W}_u^{T}\xi_i=\sigma_i\varphi_i,\\ \boldsymbol{W}_u\varphi_i=\sigma_i\xi_i,\end{cases} \tag{3.3}$$

where $i=1,2,...,r$, then we get:

$$\begin{cases}\boldsymbol{W}_u\boldsymbol{W}_u^{T}\xi_i=\sigma_i^2\xi_i,\\ \boldsymbol{W}_u^{T}\boldsymbol{W}_u\varphi_i=\sigma_i^2\varphi_i.\end{cases} \tag{3.4}$$

To construct POD basis vectors, we need to solve eigenvalue problem of Eq (3.4). Since the dimension N_{Dof} of $(\boldsymbol{W}_u\boldsymbol{W}_u^{T})_{N_{dof}\times N_{dof}}$ is far greater than that of l selected snapshots, it will cost a lot to solve the eigenvalues and eigenvectors of $\boldsymbol{W}_u\boldsymbol{W}_u^{T}$. Besides, it is usually infeasible to solve this eigenvalue problem directly. So we can transform it into a $l\times l$ eigenvalue problem here, in other words, we can use the space-time transformation technique to find the eigenvalues and eigenvectors of the correlation matrix $C=(\boldsymbol{W}_u^{T}\boldsymbol{W}_u)_{l\times l}$, thus obtaining the POD basis from it.

As $\boldsymbol{C}=\boldsymbol{W}_u^{T}\boldsymbol{W}_u$ is symmetric positive semidefinite, we can get a set of positive eigenvalues in descending order $\lambda_1>\lambda_2>\cdots>\lambda_r>0$ of $\boldsymbol{C}$ and $\varphi_1,\varphi_2,\ldots,\varphi_r$ is the corresponding eigenvectors. The matrix $\boldsymbol{F}=(\boldsymbol{W}V_r)_{N_{Dof}\times r}$ ($\boldsymbol{V}_r=(\varphi_1,\varphi_2,...,\varphi_r)$) is not standard orthogonal, so every column of the matrix $\boldsymbol{F}$ is divided by $\sqrt{\lambda_i},i=1,\cdots r$ to become a standard orthogonal matrix, define the POD basis as

$$\psi_i=\frac{1}{\sqrt{\lambda_i}}\boldsymbol{W}\varphi_i\ \ i=1,\cdots r. \tag{3.5}$$

In order to simplify the POD basis model, we define the relevant energy information to ignore modals corresponding to small eigenvalues, and choose a low POD basis vector space with dimension $d_u(d_u<r)$, we define the POD energy or error bound [41] as

$$\sum_{i=1}^{l}\left\|u_h(t_{ni})-\sum_{j=1}^{d}\left(u_h(t_{ni}),\psi_j\right)\psi_j\right\|^2=\sum_{j=d+1}^{r}\lambda_j\ . \tag{3.6}$$

The error bound Eq (3.6) should be less than specified tolerance ρ_t, namely $\sum_{j=d+1}^{r}\lambda_j<\rho_t$, so we define

$$I(d)=\sum_{j=1}^{d}\lambda_j\Big/\sum_{j=1}^{r}\lambda_j\ . \tag{3.7}$$

Choose d_u to make $d_u = \arg\min\{I(d): I(d) \le \sigma\}$, where $0 \le \sigma = 1 - \rho_t \le 1$ is the percentage of the total energy captured by the reduced dimension space. In order to capture most of the energy of the POD basis, σ must be selected near 1.

3.3. POD-DG formulation

We can use the above POD basis vectors to expand a reduced dimensional space $D_u = span\{\psi_1, \psi_2, \ldots, \psi_{d_u}\} (u = \rho, \rho u, \rho v, \rho w, \rho E)$ and construct a reduced dimensional DG model to solve. Using Galerkin approximation, we project the model equation into the reduced dimensional space expanded by POD basis vectors, and then obtain the solving coefficient, which is general time varying solution.

Before applying the POD method to the DG formula Euler equation, we need to solve problem (2.11) at N_t time steps and obtain the solutions $(\rho_n^l, \rho u_n^l, \rho v_n^l, \rho w_n^l, \rho E_n^l)$ at first. Then we choose l snapshots from N_t group of solutions (discrete data) and construct POD basis vectors through Section 3.1 and 3.2 such that

$$\begin{cases} D_\rho^{POD} = span\left\{\psi_1^{\rho}, \psi_2^{\rho}, \ldots, \psi_{d_\rho}^{\rho}\right\}, D_{\rho u}^{POD} = span\left\{\psi_1^{\rho u}, \psi_2^{\rho u}, \ldots, \psi_{d_{\rho u}}^{\rho u}\right\} \\ D_{\rho v}^{POD} = span\left\{\psi_1^{\rho v}, \psi_2^{\rho v}, \ldots, \psi_{d_{\rho v}}^{\rho v}\right\}, D_{\rho w}^{POD} = span\left\{\psi_1^{\rho w}, \psi_2^{\rho w}, \ldots, \psi_{d_{\rho w}}^{\rho w}\right\}, \\ D_{\rho E}^{POD} = span\left\{\psi_1^{\rho E}, \psi_2^{\rho E}, \ldots, \psi_{d_{\rho E}}^{\rho E}\right\} \end{cases} \tag{3.8}$$

the dimension reduction model of three-dimensional flow field adopts the form with Galerkin approximation as

$$\begin{cases} \rho(t) = \sum_{i=1}^{d_\rho} \alpha_i^{\rho}(t)\psi_i^{\rho}, \rho u(t) = \sum_{i=1}^{d_{\rho u}} \alpha_i^{\rho u}(t)\psi_i^{\rho u} \\ \rho v(t) = \sum_{i=1}^{d_{\rho v}} \alpha_i^{\rho v}(t)\psi_i^{\rho v}, \rho w(t) = \sum_{i=1}^{d_{\rho w}} \alpha_i^{\rho w}(t)\psi_i^{\rho w} \\ \rho E(t) = \sum_{i=1}^{d_{\rho E}} \alpha_i^{\rho E}(t)\psi_i^{\rho E} \end{cases}. \tag{3.9}$$

Unifying the reduced-basis approximation of original variables above, the approximation of the solution can be represented as

$$u^{POD} = \sum_{n=1}^{N_p} u_n{}^{POD}(t)\phi_n^p \approx \sum_{n=1}^{N_p} \tilde{u}_n(t)\phi_n^p = \sum_{n=1}^{N_p}\sum_{i=1}^{d_u} \alpha_i^u(t)\psi_i^u\phi_n^p, \tag{3.10}$$

With

$$u_n{}^{POD}(t) \approx \tilde{u}_n(t) = \sum_{i=1}^{d_u} \alpha_i^u(t)\psi_i^u, \quad u = \rho, \rho u, \rho v, \rho w, \rho E, \tag{3.11}$$

where $\tilde{u}_n(t)$ is the reduced-order solution and $\alpha_i^u(t)$ is coefficient to be found.

Take Eq (3.11) into the global system (2.11), the reduced dimension model equation can be expressed as

$$\boldsymbol{M}\cdot\frac{\partial \boldsymbol{u}^{POD}(t)}{\partial t}=\boldsymbol{R}\left(\boldsymbol{u}^{POD}\right). \tag{3.12}$$

After simplification, we get

$$\boldsymbol{\Psi}\cdot\frac{\partial \boldsymbol{\alpha}(t)}{\partial t}=\boldsymbol{M}^{-1}\cdot\boldsymbol{R}\left(\sum_{n=1}^{N_p}\sum_{i=1}^{d}\alpha_i^u(t)\psi_i^u\phi_n^p\right), \tag{3.13}$$

where $\boldsymbol{\Psi}$ is POD basis vector and $\boldsymbol{\alpha}(t)$ is coefficient vector. Applying the Galerkin projection on it, we get the POD-DG formulation

$$\boldsymbol{\Psi}^T\cdot\boldsymbol{\Psi}\cdot\frac{\partial \boldsymbol{\alpha}(t)}{\partial t}=\boldsymbol{\Psi}^T\cdot\boldsymbol{M}^{-1}\cdot\boldsymbol{R}\left(\sum_{n=1}^{N_p}\sum_{i=1}^{d}\alpha_i^u(t)\psi_i^u\phi_n^p\right). \tag{3.14}$$

Owing to the orthogonality of POD basis vector

$$\psi_i\psi_j=\begin{cases}1, i=j\\ 0, i\neq j\end{cases}, \text{ that is } \boldsymbol{\Psi}^T\cdot\boldsymbol{\Psi}=\boldsymbol{E}_{d\times d} \tag{3.15}$$

we have

$$\frac{\partial \boldsymbol{\alpha}(t)}{\partial t}=\boldsymbol{R}_n(\alpha),\ \boldsymbol{R}_n(\alpha)=\boldsymbol{\Psi}^T\cdot\boldsymbol{M}^{-1}\cdot\boldsymbol{R}\left(\sum_{n=1}^{N_p}\sum_{i=1}^{d}\alpha_i^u(t)\psi_i^u\phi_n^p\right). \tag{3.16}$$

Thus, we transform system (2.11) into reduced-order system (3.14), and solve $\boldsymbol{\alpha}(t)$ instead of $\boldsymbol{u}_n$.

Remark 3.2. During each time step, the DG formulation (2.11) contains $N_k\times N_p$ unknowns, while the POD-DG formulation (3.16) only contains $\sum_{i=\rho,u,v,w,E} d_i$ $(N_k\times N_P\, d_i\leq r\ll N_t)$ unknowns.

4. Numerical experiments

In this section, we show some numerical experimental results to evaluate DG formula based on POD dimension reduction model for solving three-dimensional Euler equations. We choose $l^2 = O(N_t)$ equidistantly distributed snapshots vectors, you also can get other choice for l from ref. [42,43]. We both test the POD-DG method and traditional DG method for comparison, in all tests the order p of the interpolation polynomials in both two methods is the same. All the simulations are implemented with C++ programming language on Microsoft Visual Studio 2010 in a Windows 10 64-bit Intel Core i5-4590 3.3-GHz and 8-GB RAM small workstation.

4.1. Isentropic vortex for inviscid flow

As the isentropic vortex test case for inviscid flow is easy to get exact value, we consider it to analyze the accuracy of the POD-DG method developed in this paper at first, it also provides a more reliable basis for the results of the following tests. The initial average flow is $(\rho,u,v,w,p)=(1,1,1,0,1)$, and the vortex is given by

$$\begin{cases} \delta u = \dfrac{\varepsilon}{2\pi} e^{\left[1-\left((x-x_0-t)^2-(y-y_0-t)^2\right)\right]/2}(5-y) \\ \delta v = \dfrac{\varepsilon}{2\pi} e^{\left[1-\left((x-x_0-t)^2-(y-y_0-t)^2\right)\right]/2}(x-5) \\ \delta w = 0.0 \\ \delta T = -\dfrac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-\left((x-x_0-t)^2-(y-y_0-t)^2\right)} \\ \delta S = 0 \end{cases},$$

and

$$p = \rho^r, T = p/\rho, S = p/\rho^r = 1,$$

where ε = 5.0, $x_0 = y_0$ = 5.0 and γ = 1.4, (x_0,y_0), is vortex center position. The whole computational domain is cube, its size is [0,10] × [0,10] × [0,10]. We employ three successively refined meshes of tetrahedral elements to study the convergence of the POD-DG method. The boundary conditions is taken as the exact solution. We take global time step as Δt = 0.0001, both test the POD-DG method and traditional DG method till time T_f = 1.0. The exact solution of temperature T of isentropic vortex at any time is as follows

$$T = 1 - \frac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-((x-x_0-t)^2-(y-y_0-t)^2)}.$$

The exact solution of density $\rho = p^{1/r}$ is used to verify the accuracy of the proposed method. We define the L^2-error in the norm as

$$\|u-u_e\|_e = \left(\sum_{e\in\Omega}\|u-u_e\|^2\right)^{1/2}.$$

Table 1. L^2-Error and convergence orders of traditional DG method and POD-DG method on three meshes: (M0) initial mesh, (M1) refined mesh, (M2) twice refined mesh.

P-order	Mesh	traditional DG		POD-DG	
		L2 Error	convergence order	L2 Error	convergence order
1	(M0)	7.42e−2	-	7.76e−2	-
	(M1)	5.29e−2	2.110	5.58e−2	2.049
	(M2)	1.93e−2	2.355	2.37e−2	2.003
	(M3)	1.04e−2	2.362	1.39e−2	2.0533
2	(M0)	2.42e−2	-	2.56e−2	-
	(M1)	1.54e−2	2.829	1.66e−2	2.709
	(M2)	4.06e−3	3.119	4.57e−3	3.017
	(M3)	1.78e−3	3.124	9.01 e−3	3.055

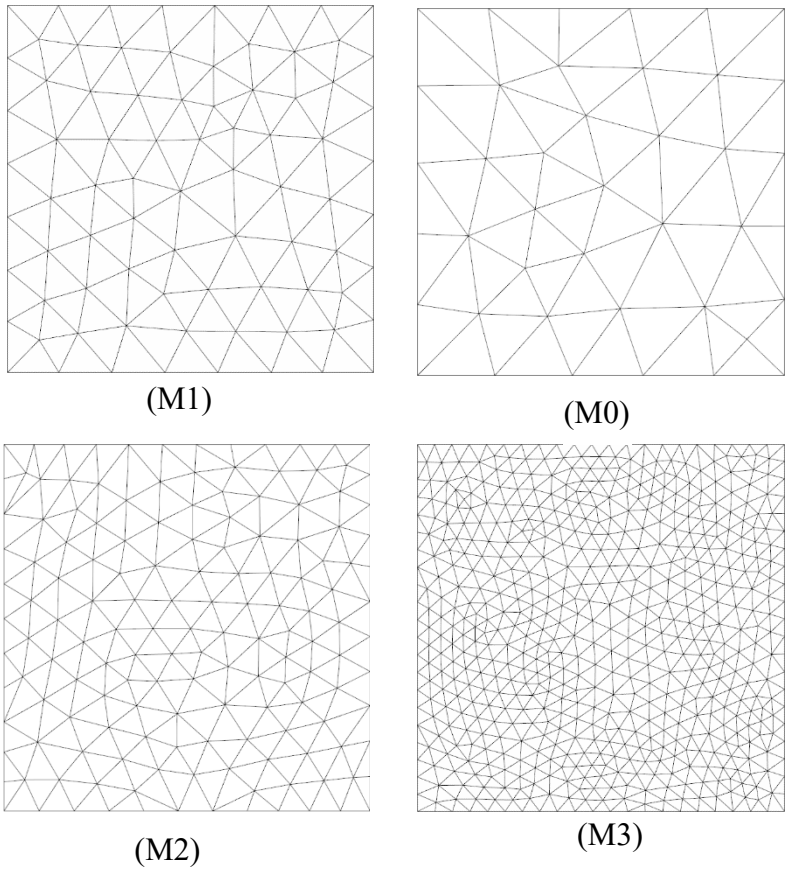
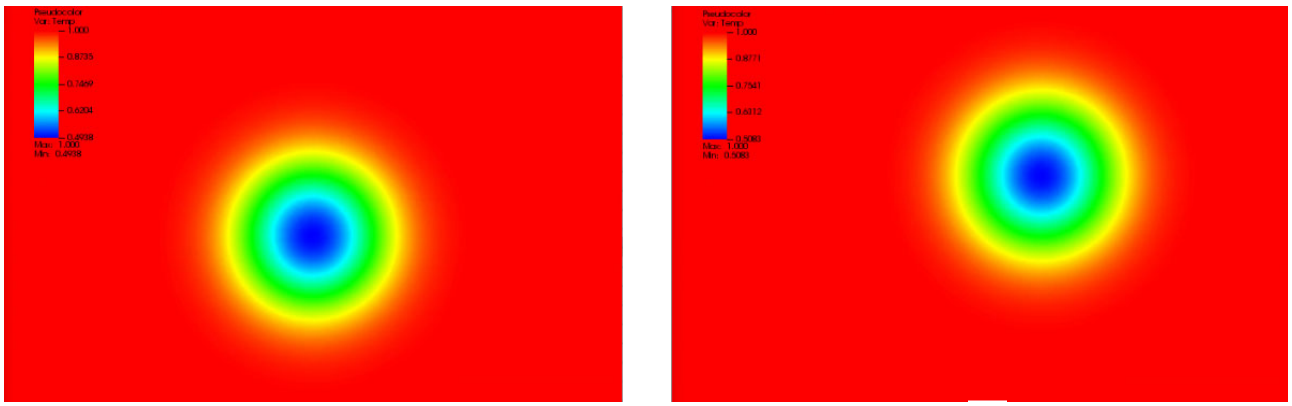


Figure 2. Sequences of the four successively globally refined tetrahedral meshes.

We give four successively refined meshes of tetrahedral elements as shown in Figure 2 to test the POD-DG method and traditional DG method respectively to compare, the numbers of elements, points, and faces for the coarse grids (M0), medium grids (M1), fine grids (M2) and finest grids (M3) are 929, 4311, 10802 and 21417 respectively. Under POD-DG method, the POD modes used in the meshes M0, M1 and M2 are 12, 12 and 18 with order p = 1, 17, 17 and 18 with order p = 2 respectively. Convergence results and the errors of the density for both two methods are presented in Table 1. We observe that under four successively refined meshes of tetrahedral elements, although POD-DG method lost a little bit of precision compared with traditional DG method, it's still within the acceptable range, the calculation results are not affected. Besides, both two methods converge at about 2 in order p = 1 and 3 in order p = 2. Figure 3 shows the density distribution of POD-DG method, Figure 3a is the initial density distribution at starting time $T_0 = 0$ and Figure 3b is the final density distribution at final time $T_f = 1.0$. It can be seen from Figure 2 that the vortex moves from (5,5,0) at $T_0 = 0$ to (6,6,0) at $T_f = 1.0$. Note that this example only does the accuracy verification of the POD-DG algorithm because the computing time is stationary from $T_0 = 0$ to final time $T_f = 1.0$.



(a) (b)

Figure 3. Density distribution of POD-DG method.

4.2. Subsonic flow

Then we give the numerical results of an experiment about a subsonic flow at March number $M_\infty = 0.2$ and adopt solid wall boundary condition and far field boundary condition. We also give three successively refined meshes of tetrahedral elements as shown in Figure 4 to test the POD-DG method and traditional DG method respectively for comparison, the numbers of elements for the coarse grids (M0), medium grids (M1), fine grids (M2) and finest grids (M3) are 7893, 37551, 51640 and 100483 respectively, and the global time steps take as $\Delta t = 10^{-5}, 10^{-6}, 10^{-6}, 10^{-7}$ respectively.
In order to prove that POD-DG method doesn't influence the accuracy of the original DG method, we define the L^2-errors in entropy ε_{ent} as:

$$\varepsilon_{ent} = \frac{P}{P_\infty} \Big/ \left(\frac{\rho}{\rho_\infty} \right)^\gamma - 1,$$

where P_∞ and ρ_∞ are the pressure and density of the free stream respectively. And the convergence rate r is obtained as follows

$$r = \frac{\log\left((\varepsilon_{ent})_\alpha / (\varepsilon_{ent})_\beta \right)}{\log\left(h_\alpha / h_\beta \right)},$$

where h is the grid size of α and β.

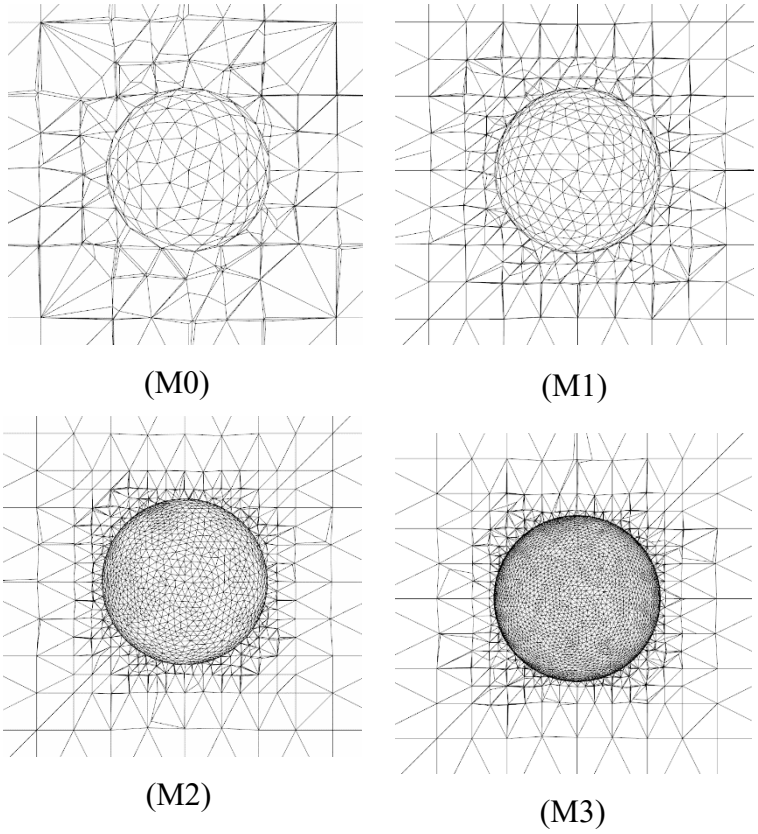
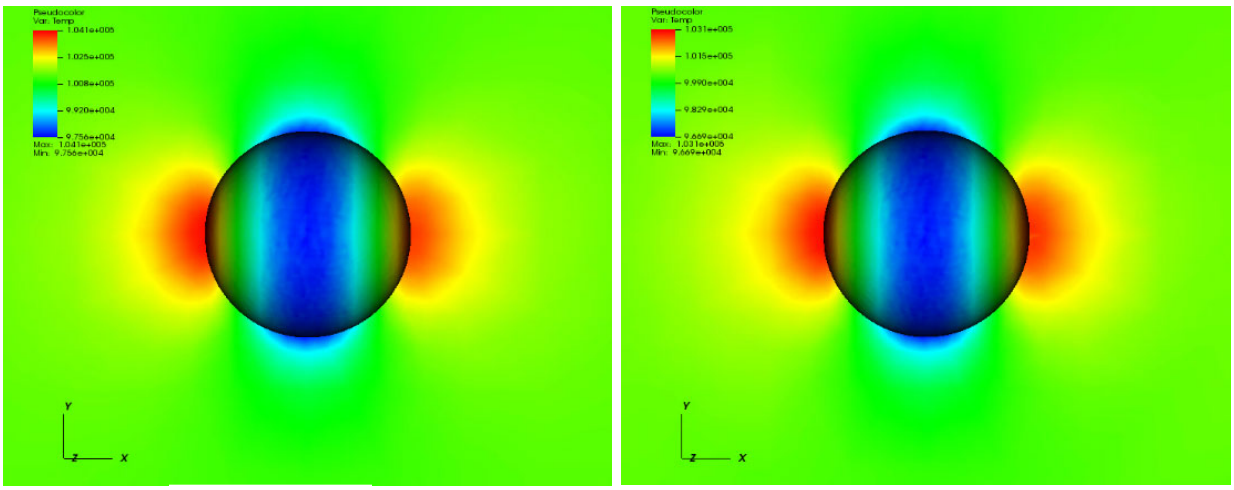


Figure 4. Sequences of the four successively globally refined tetrahedral meshes.



(a) DG method (b) POD-DG method

Figure 5. Pressure distribution of DG method and POD-DG method.

Convergence results of both POD-DG and traditional DG methods and the L^2-errors in entropy are presented in Table 2. We observe that both POD-DG and traditional DG methods converge at the optimal order, which indicates POD-DG method maintains accuracy of original DG method. Besides, the comparison in terms of CPU time and the number of DOFs of two types of methods using different refined meshes are reported in Table 3, from it we can see that with the coarse meshes in (M0), the

proposed POD-DG method can speed 1.9 times compared with the traditional DG scheme for whole simulation time due to smaller number of DOFs, which means it accelerates the convergence rate. And with meshes refined the POD-DG method speed faster, in (M1), the speed-up ratio reaches 2.5, in (M2) and (M3), the speed-up ratio can reach 2.9. This presents that with the refinement of the mesh, the acceleration effect of proposed method is more obvious. Figure 5 shows the pressure distribution maps of stable flow under POD-DG and traditional DG methods, respectively. It can be seen from the figure that the result of POD-DG method is consistent with DG method, it conforms to the distribution of inviscid flow.

Table 2. L^2-error and convergence results between POD-DG method and traditional DG method.

Mesh	traditional DG		POD-DG	
	ε_{ent}	convergence rate	ε_{ent}	convergence rate
(M0)	1.11e−2	-	1.22e−2	-
(M1)	5.12e−3	1.12515	5.26e−3	1.22189
(M2)	2.09e−3	1.29081	1.91e−3	1.45718
(M3)	7.91e-4	1.4018	6.87e-4	1.4752

Table 3. Calculation results comparison between POD-DG method and traditional DG method.

Mesh	traditional DG		POD-DG		Speed up
	DOFs	Computing Time (s)	DOFs	Computing Time (s)	
(M0)	31572	7822	38	4114	1.901
(M1)	150204	17073	35	6789	2.515
(M2)	206560	251076	68	84854	2.959
(M3)	401932	364640	68	123023	2.964

4.3. ONERA-M6

Finally, we consider a numerical simulation of inviscid subsonic flow around ONERA-M6 wing with March number $M_\infty = 0.4$ and angle of attack $\alpha = 0^0$ to test the efficiency of the developed POD-DG method. The ONERA-M6 wing [44] is a swept back wing with a root chord of about 0.8 m and a half span of about 1.2 m, and its profile shape is NACA0012. The geometric layout is shown in Figure 6 and its tetrahedral meshes is shown in Figure 7.

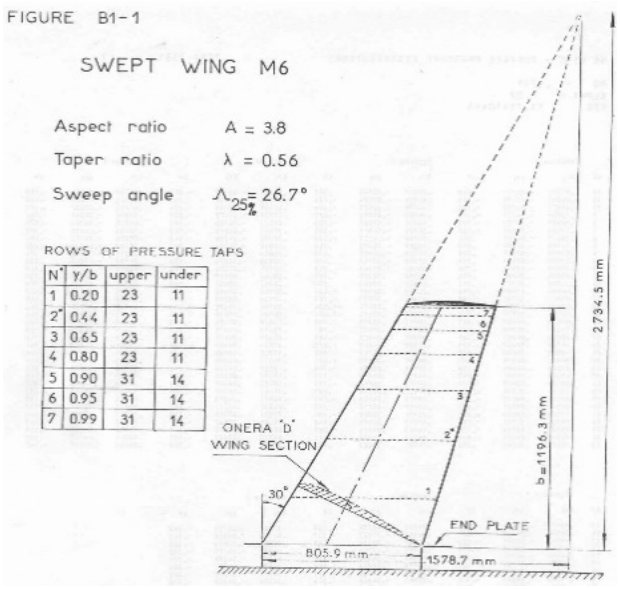


Figure 6. Geometric layout of ONERA-M6 wing.

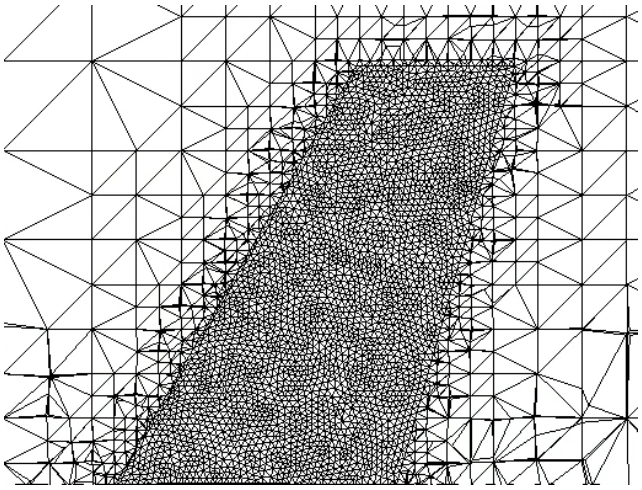
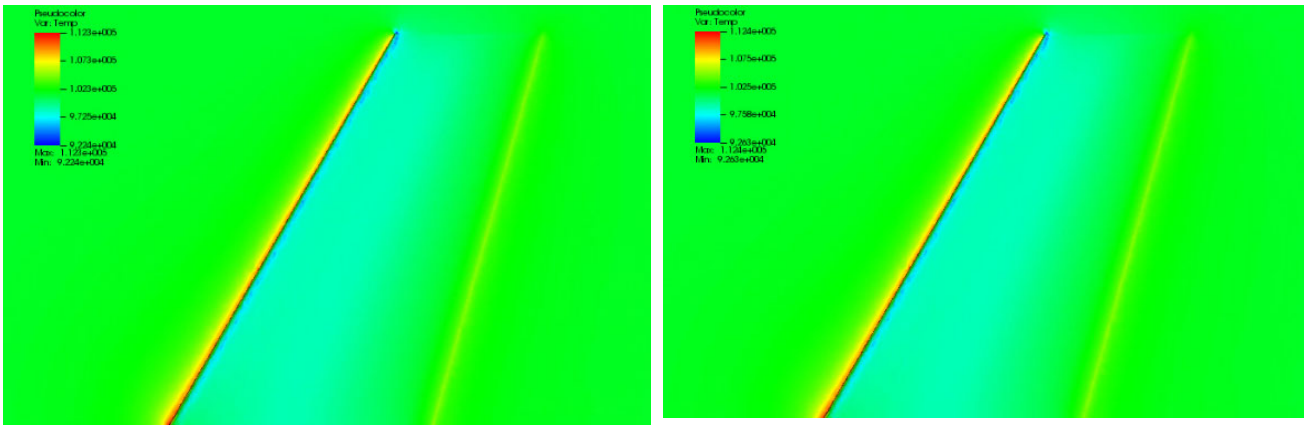


Figure 7. Meshes of ONERA-M6 wing.

As early as 1972, the National Space Research Institute of France had completed the wind tunnel of ONERA-M6 on the ONERA S2MA wind tunnel, and obtained quite abundant experimental data. At the same time, the experiment calculation results are numerous, so the data are accurate and reliable. We here give two successively refined meshes of tetrahedral elements to test the POD-DG method and traditional DG method respectively for comparison, the numbers of elements for the coarse grids (M0) and fine grids (M1) are 29992 and 95270 respectively, and the global time steps both take as $\Delta t = 10^{-6}$.



(a) DG method (b) POD-DG method

Figure 8. Pressure distribution of DG method and POD-DG method.

Table 4. Calculation results comparison between POD-DG method and traditional DG method.

traditional DG		POD-DG		Speed up
DOFs	Computing Time (s)	DOFs	Computing Time (s)	
119968	15664	19	8855	1.769
381080	118881	21	48418	2.455

Figure 8 shows the pressure distribution of both POD-DG and traditional DG methods. It can be seen from the figure that the result of POD-DG method are consistent with DG method, it conforms to the distribution of inviscid flow. The comparison in terms of CPU time and the number of DOFs of two types of methods are presented in Table 4, from which we can see that compared with the traditional DG scheme, the proposed POD-DG method can speed up 1.769 times for whole simulation time under coarse grids and speed up 2.455 times under fine grids, again we find that the proposed POD-DG method speeds up the convergence rate and can save CPU time due to smaller number of DOFs.

5. Conclusion

In order to overcome the disadvantages of too many DOFs in DG algorithm, which leads to high computational cost and memory requirement, in this paper, we employ a proposed DG method based on POD model reduction (the POD-DG method) to improve the computational efficiency of DG algorithm when solving three-dimensional Euler equations. Combining with POD technology, one can construct snapshot matrixes which consist of transient solution with DG formula, build the POD base vector space and project the model to dimension reduction space via eigenorthogonal decomposition or SVD for calculation, thus DG algorithm can reduce the DOFs of calculation model, speed up convergence and save calculation time while maintaining acceptable accuracy. Some numerical tests are presented to validate its computational efficiency, by using POD-DG method, one can speed up the convergence rate and save CPU time because of the reduction in global dimension. It turns out that this proposed method is almost twice as fast as the original DG method, and with the refinement of the mesh, the acceleration effect is more obvious. Furthermore, the theoretical and numerical errors are

still within acceptable range with POD-DG method, it converges at almost same rate as traditional DG method, which indicates the proposed method ensures the accuracy of original method.

In the near future, we will consider the POD-DG method on CFD problems with unsteady flow and combine nonlinear hyperreduction techniques to further reduce the global calculation time.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments (All sources of funding of the study must be disclosed)

This work was supported by National Natural Science Foundation of China (No. 62071102, 12102087, 61771105 and 61921002) and Fundamental Research Funds for Central Universities (No. 2672018ZYGX2018J037).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. C. R. Nastase, D. J. Mavriplis, A parallel hp-multigrid solver for three-dimensional discontinuous Galerkin discretizations of the Euler equations, *45th AIAA Aerospace Sciences Meeting and Exhibit*, (2007), 512. https://doi.org/10.2514/6.2007-512
2. H. Luo, J. D. Baum, R. Löhner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, *J. Comput. Phys.*, **211** (2006), 767–783. https://doi.org/10.1016/j.jcp.2005.06.019
3. R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, *J. Comput. Phys.*, **183** (2002), 508–532. https://doi.org/10.1063/1.5033621
4. F. Bassi, S. Rebay., High-order accurate discontinuous finite element solution of the 2D Euler equations, *J. Comput. Phys.*, **138** (1997), 251–285. https://doi.org/10.1006/jcph.1997.5454
5. F. Bassi, S. Rebay., A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations, *J. Comput. Phys.*, **131** (1997), 267–279. https://doi.org/10.1006/jcph.1996.5572
6. L. Chen, M. B. Ozakin, R. Zhao, H. Bagci, A locally-implicit discontinuous Galerkin time-domain method to simulate metasurfaces using generalized sheet transition conditions, *IEEE Trans. Antennas Propag.*, **71** (2023), 869–881. https://doi.org/10.1109/AP-S/USNC-URSI47032.2022.9887215
7. G. S. Baruzzi, Wagdi Habashi, A second order finite element method for the solution of the transonic Euler and Navier-Stokes equations, *Int. J. Numer. Methods Fluids.*, **20** (1995), 671–693. https://doi.org/10.1002/fld.1650200802
8. M. Gurris, D. Kuzmin, S. Turek., Implicit finite element schemes for the stationary compressible Euler equations, *Int. J. Numer. Methods Fluids.*, **69** (2012), 1–28. https://doi.org/10.1002/fld.2532

9. A. Jameson, D. Mavriplis, Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh, *AIAA J*, **24** (1986), 611–618. https://doi.org/10.2514/3.9315
10. W. K. Anderson, Comparison of finite volume flux vector splittings for the Euler equations, *AIAA J.*, **24** (2015), 1453–1460. https://doi.org/10.2514/3.9465
11. L. Acedo, S. B. Yuste, An explicit finite difference method and a new von Neumann-type stability analysis for fractional diffusion equations, *SIAM J. Numer. Anal.*, **42** (2005), 1862–1874. https://doi.org/10.1137/030602666
12. R. F. Warming, B. J. Hyett, The modified equation approach to the stability and accuracy of finite difference method, *J. Comput. Phys.*, **14** (1974), 159–179. https://doi.org/10.1016/0021-9991(74)90011-4
13. U. Baur, P. Benner, L. Feng, Model order reduction for linear and nonlinear systems: A system-theoretic perspective, *Arch. Comput. Methods Eng*., **21** (2014), 331–358. https://doi.org/10.1007/s11831-014-9111-2
14. Z. Bai, Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, *Appl. Numer. Math.*, **43** (2002), 9–44. https://doi.org/10.1016/S0168-9274(02)00116-2
15. S. Renee, M. Laura, P. Benjamin, W. Karen, Projection-based model reduction: Formulations for physics-based machine learning, *Comput Fluids.*, **179** (2019), 704–717. https://doi.org/10.1016/j.compfluid.2018.07.021
16. J. Jiang, Y. Chen, N. Akil, Offline-enhanced reduced basis method through adaptive construction of the surrogate training set, *J Sci Comput.*, **73** (2017), 853–875. https://doi.org/10.1007/s10915-017-0551-3
17. Z. Peng, Y. Chen, Y. Cheng, F. Li, A reduced basis method for radiative transfer equation, *J Sci Comput.*, **91** (2022), 5. https://doi.org/10.1007/s10915-022-01782-2
18. D. Binion., X. Chen, A Krylov enhanced proper orthogonal decomposition method for efficient nonlinear model reduction, *Finite Elem. Anal. Des.*, 47 (2011), 728–738. https://doi.org/10.1016/j.finel.2011.02.004
19. L. Sirovich., Turbulence and the dynamics of coherent structures part I: Coherent structures, *Appl. Math*., **45** (1986), 561–571. https://doi.org/10.1090/qam/910464
20. C. L. Pettit, P. S. Beran., Application of proper orthogonal decomposition to the discrete Euler equations, *Int. J. Numer. Methods Eng.*, **55** (2002), 479–497. https://doi.org/10.1002/nme.510
21. J. Goss, K. Subbarao., Inlet shape optimization based on POD model reduction of the Euler equations, *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (2008), 5809. https://doi.org/10.2514/6.2008-5809
22. G. Kerschen, J. C. Golinval, A. F. Vakakis, L. A. Bergman, The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview, *Nonlinear Dyn.*, **41** (2005), 147–169. https://doi.org/10.1007/s11071-005-2803-2
23. Q. Wang, N. Ripamonti, J. S. Hesthaven, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, *J Comput Phys.*, **410** (2020), 109402. https://doi.org/10.1016/j.jcp.2020.109402
24. T. Akman, Error estimates for space–time discontinuous Galerkin formulation based on proper orthogonal decomposition, *Appl. Anal.*, **96** (2017), 461–482. https://doi.org/10.1080/00036811.2016.1143930
25. C. Gräßle, M. Hinze, POD reduced order modeling for evolution equations utilizing arbitrary finite element discretizations, *Adv. Comput. Math.*, **44** (2018), 1941–1978. https://doi.org/10.1007/s10444-018-9620-x

26. Z. D. Luo, F. Teng, J. Chen, A POD-based reduced-order Crank-Nicolson finite volume element extrapolating algorithm for 2D Sobolev equations, *Math. Comput. Simul.*, 146 (2018) 118–133. https://doi.org/10.1016/j.matcom.2017.11.002
27. S. F Zhu, L. Dedè, A. Quarteroni, Isogeometric analysis and proper orthogonal decomposition for parabolic problems, *Numer Math*., **135** (2017), 333–370. https://doi.org/10.1007/s00211-016-0802-5
28. R. C. Li, Q. B. Wu, S. F. Zhu, Proper orthogonal decomposition with SUPG-stabilized isogeometric analysis for reduced order modelling of unsteady convection-dominated convection-diffusion-reaction problems, *J Comput Phys.*, **387** (2019), 280–302. https://doi.org/10.1016/j.jcp.2019.02.051
29. S. F Zhu, L. Dedè, A. Quarteroni, Isogeometric analysis and proper orthogonal decomposition for the acoustic wave equation, *ESAIM: M2AN*, **51** (2017), 1197–1221. https://doi.org/10.1051/m2an/2016056
30. S. Jun, K. H Park, H. M Kang, D. H Lee, M. Cho, Reduced order model of three-dimensional Euler equations using proper orthogonal decomposition basis, *J Mech Sci Technol.*, **24** (2010), 601–608. https://doi.org/10.1007/s12206-010-0106-0
31. R. Hartmann, Error estimation and adjoint based refinement for an adjoint consistent DG discretisation of the compressible Euler equations, *Int J Comput Sci Mat*, **1** (2007), 207–220. https://doi.org/10.1504/IJCSM.2007.016532
32. X. Zhang, C. Shu, Positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations with source terms, *J Comput Phys.*, **230** (2011), 1238–1248. https://doi.org/10.1016/j.jcp.2010.10.036
33. M Boizard, R Boyer, G Favier, P. Larzabal, Fast multilinear Singular Values Decomposition for higher-order Hankel tensors, *2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, (2014), 437–440. https://doi.org/10.1109/SAM.2014.6882436
34. P. Batten, N. Clarke, C. Lambert, D. M. Causon, On the choice of wavespeeds for the HLLC Riemann solver, *SIAM J Sci Comput*, **18** (1997), 1553–1570. https://doi.org/10.1137/S1064827593260140
35. S. Simon, J. C. Mandal, A cure for numerical shock instability in HLLC Riemann solver using antidiffusion control, *Comput Fluids.*, **174** (2018), 144–166. https://doi.org/10.1016/j.compfluid.2018.07.001
36. S. Simon, J. C. Mandal, A simple cure for numerical shock instability in the HLLC Riemann solver, *J Comput Phys.*, **378** (2019), 477–496. https://doi.org/10.1016/j.jcp.2018.11.022
37. Y. Zhu., A. C. Cangellaris, *Multigrid Finite Element Methods for Electromagnetic Field Modeling*, New York: John Wiley & Sons, 2006.
38. B. Cockburn, S. Hou, C. W Shu, TVD Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, *Math Comp*, **55** (1990), 545–581. https://doi.org/10.1090/S0025-5718-1990-1010597-0
39. H. Zhang, J. Y Yan, X. Qian, X. M Gu, S. H Song, On the preserving of the maximum principle and energy stability of high-order implicit-explicit Runge-Kutta schemes for the space-fractional Allen-Cahn equation, *Numer. Algorithms*, **88** (2021), 1309–1336. https://doi.org/10.1007/s11075-021-01077-x
40. H. Luo, J. Baum, R. Löhner, A fast p-Multigrid Discontinuous Galerkin Method for Compressible Flows at All Speeds, *AIAA J*, **46** (2008), 635–652. https://doi.org/10.2514/1.28314

41. 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. https://doi.org/10.1016/j.cma.2006.04.004
42. Z. Luo, J. Gao, A POD reduced-order finite difference time-domain extrapolating scheme for the 2D Maxwell equations in a lossy medium, *J. Math. Anal. Appl.*, **444** (2016), 433–451. https://doi.org/10.1016/j.jmaa.2016.06.036
43. K. Kunisch, S. Volkwein, Galerkin proper orthogonal decomposition methods for parabolic problems, *Numer. Math.*, **90** (2001), 117–148. https://doi.org/10.1007/s002110100282
44. V. Schmitt, Pressure distributions on the ONERA M6-wing at transonic Mach numbers, experimental data base for computer program assessment, *AGARD AR138*, (1979).

AIMS Press