Accelerating Unsteady Aerodynamic Simulations Using Predictive Reduced-order Modeling

Zilong Li, Ping He*

Department of Aerospace Engineering, Iowa State University, Ames, Iowa, 50011, USA

Abstract

Unsteady computational fluid dynamics (CFD) simulations are essential in aerospace engineering because they can provide high-fidelity flow fields to better understand transient physics, such as vortex shedding. However, unsteady full-order CFD simulations must repeatedly march the flow solution with a small time step and are computationally expensive. Reduced-order modeling (ROM) is a powerful approach to alleviate the above issue by decomposing the unsteady flow solutions into spatial modes and temporal coefficients, making the unsteady flow easier to simulate. Existing ROM studies mostly focused on parametric problems that use a large number of simulation samples to train an offline model (parametric ROM). Although the trained model can quickly predict any flow fields within the parameter space, the computational cost for generating the massive unsteady simulation samples is still high, especially when the number of parameters and their ranges increase. To further address the high-cost issue, we develop an efficient predictive ROM approach to accelerate individual unsteady aerodynamic simulations. We use the Galerkin projection approach to reduce the Reynolds-averaged Navier–Stokes equations, along with the discrete empirical interpolation method (DEIM) for decreasing the computation cost for nonlinear terms. In addition, we develop an efficient ROM formulation that correlates the temporal coefficients between the momentum, pressure, and turbulence equations, allowing solving fewer ROM equations at the prediction stage. The intrusive nature of the predictive ROM enables using the first portion of unsteady flow data to accelerate the rest of the simulation; no massive offline samples are needed. We use the stalled turbulent flow over the NACA0012 airfoil as the benchmark and evaluate the predictive ROM's speed and accuracy for challenging non-equilibrium scenarios caused by a large sudden change in flow conditions. The run time ratios between CFD and ROM (including the calculation of basis vectors and projection matrices) range between 13 and 45. The pressure, drag, lift, pitching moment, and flow fields simulated by the ROM approach agree reasonably well with the CFD references at various times. The proposed predictive ROM approach is, in principle, applicable to different airfoil geometries and flow conditions and can be integrated into a design optimization process for accelerating unsteady aerodynamic simulations.

Keywords: Reduced-order modeling; Galerkin projection; DEIM; unsteady aerodynamic simulations

1. Introduction

Computational fluid dynamics (CFD) of unsteady flow is essential in aerospace engineering design because it can provide high-fidelity field data to better understand transient flow physics, such as vortex shedding, flow separation, and unsteady fluid-structure interaction. The CFD-based design has been used for simple geometries such as airfoils and complex cases such as aircraft. However, unsteady flow simulations are computationally expensive because they must march the flow solution with a small step in time. There is a need to reduce the above cost and make unsteady CFD simulations more affordable.

To alleviate the above cost, one can use the reduced-order modeling (ROM) approach. ROM typically starts with computing a set of modes (basis vectors) that capture the spatial flow patterns using the proper orthogonal decomposition (POD) with the method of snapshot [1]. It can then approximate the full-order

*Corresponding author
Email address: phe@iastate.edu (Ping He)

flow solutions using a linear combination of spatial modes and temporal coefficients. Computing the temporal coefficients is generally much faster than solving the full-order unsteady CFD, making the ROM approach computationally efficient. Moreover, the main flow physics are preserved during the POD projection so that the ROM model can predict the 3D *field* variables such as velocity and pressure fields. In contrast, surrogate models such as neural networks and Kriging can typically predict *surface* variables such as drag, lift, pitching moment, and pressure distribution [2–5]. The ROM approach has been used in various aerospace applications such as aircraft, turbomachinery, combustors, and re-entry vehicles (see the review papers [6–11] for more details).

ROMs can be implemented in two ways: non-intrusive and intrusive. Non-intrusive or data-driven ROMs treat the full-order flow model as a black box. It can directly learn the relationship between the POD temporal coefficients and the flow parameters (e.g., Reynolds number, angle of attack, and airfoil shape). There are numerous non-intrusive ROM studies, and here we conduct a literature review of only a few recent works. Li et al. [12] developed a non-intrusive ROM that utilized a long short-term memory (LSTM) neural network to train the relationship. The trained ROM can then rapidly predict unsteady aerodynamic flow across different airfoil shapes. Liu et al. [13] implemented a signal interpolation approach consisting of the discrete empirical interpolation method (DEIM) and Kriging to alleviate the cost of generating large amounts of high-fidelity CFD training data. The trained ROM model exhibited good prediction accuracy across different Reynolds numbers and angles of attack. Krolick et al. [14] developed a data-driven ROM approach that preserved the state consistency to ensure physical simulation results for unsteady aero-structural coupling problems. Decker et al. [15] developed a manifold alignment-based non-intrusive multi-fidelity ROM approach to predict aerodynamic flow fields over transonic and supersonic airfoils with different shapes. Halder et al. [16] combined the deep learning convolutional autoencoders (CAE) and Gaussian process regression (GPR) to learn the temporal coefficients of a non-intrusive ROM model. They found that the CAE-GPR ROM outperformed the traditional POD-GPR method. Saltari et al. [17] used neural networks to train a non-intrusive ROM and studied the effect of wind gusts on the wing aeroelastic response. Liu et al. [18] developed a data-driven ROM framework that combined a convolutional neural network (CNN) with a convolutional LSTM to predict complex transonic flow phenomena, such as shock wave motions with buffet flow, at various angles of attack. A non-intrusive ROM that combined the operator inference method with quadratic manifolds was proposed by Geelen et al. [19]. They found the proposed ROM approach was scalable and efficient for modeling dynamic systems. Kapteyn et al. [20] incorporated data-drive ROM into a digital twin model and allowed aircraft to dynamically adjust the mission in case of structural damage. The authors validated this capability using a fixed-wing unmanned aerial vehicle prototype. In addition to aerospace engineering, data-driven ROM has been used to enable rapid aerodynamic analysis and design of automobiles [21, 22].

Intrusive ROM's implementation requires the full-order model's (FOM) governing equations. It derives the ROM equations by projecting the FOM into a reduced space, making it much faster to solve [23]. The projection-based ROM (e.g., using POD) was used to accelerate flow simulations in early studies in the 1900s [24–26]. To reduce nonlinear calculation cost in projection ROM, Chaturantabut and Sorensen [27] proposed a discrete empirical interpolation method (DEIM) that showed significant speed up with negligible errors. Carlberg et al. [28] proposed a Gauss-Newton with approximated tensors (GNAT) and gappy POD [29] method for non-linear intrusive ROM and applied it to turbulent flow over the Ahmed body. They showed that GNAT reduced the computational cost by two orders of magnitude while maintaining good accuracy. Lorenzi et al. [30] developed a POD-ROM solver for Reynolds-averaged Navier–Stoke (RANS) equations and tested it with lid-driven cavity turbulent flow. The POD-ROM solver exhibited satisfactory performance in accuracy and computational cost. Star et al. [31] later extended the POD-ROM solver for predicting turbulent heat transfer. He et al. [32] developed a least-square Petrov-Galerkin intrusive ROM solver and evaluated it using steady turbulent flow over 2D airfoils and 3D wings, and both cases achieved reasonable speedups with satisfactory accuracy. Huang et al. [33] proposed a model-form preserving least-squares with variable transformation (MP-LSVT) method and tested it using 2D and 3D reacting turbulent flow. The MP-LSVT solver exhibited better numerical stability and accuracy than standard POD ROM solvers in predicting future flow states. Garbo and Bekemeyer [34] developed an unsteady residual ROM solver and evaluated its performance using subsonic and transonic turbulent flow over 2D airfoils and 3D aircraft. They found that the proposed intrusive ROM can reduce 90% computational time compared with the full-order CFD.

Both ROM implementations have their advantages and limitations. The non-intrusive ROM does not need to access the FOM source code and is flexible to implement. However, the drawback is that they rely on a large number of simulation samples to cover the entire parameter space, so the computational cost for the offline stage can be high (e.g., hundreds of hours in [12]), especially when the number of parameters and their ranges increase. In addition, a non-intrusive model needs additional training points when dealing with new flow conditions and geometries outside of the parameter space. Non-intrusive ROMs are mostly used for parametric studies (a.k.a parametric ROMs). The intrusive ROMs incorporate the governing equations in the loop and can predict wider flow conditions and geometries. It can use the first portion of FOM simulations to train a ROM model and predict the flow for the rest of the simulation (a.k.a. predictive ROMs); no massive offline samples are needed. However, the limitation is that it requires access to the FOM source codes, especially the governing equations (e.g., the Navier–Stokes equations) and how they are discretized and solved. Therefore, it requires careful consideration of ROM robustness and efficiency, and its implementation is typically more challenging and time-consuming than non-intrusive ones. Recently, hybrid projection and data-driven ROM approaches [35–38] have been proposed to combine the advantages of intrusive and non-intrusive ROM.

The objective of this paper is to develop an efficient ROM approach to accelerate unsteady aerodynamic simulations (predictive ROM) without generating massive training samples. Therefore, we choose the intrusive ROM approach. Compared with non-intrusive ROM approaches (e.g., dynamic mode decomposition (DMD) [39, 40]), the intrusive predictive ROM has the potential of better capturing irregular, non-equilibrium flow patterns. We developed a Galerkin projection ROM solver to predict turbulent flow over cylinders and airfoils in a previous work [41]. This paper is a further step forward and improves the computational efficiency and robustness to handle more challenging predictive cases. We use the Galerkin projection approach to reduce the RANS equations. To further reduce the computation cost for nonlinear terms, we use the discrete empirical interpolation method (DEIM). To tackle the challenge of intrusive ROM implementation, we develop an efficient ROM formulation that correlates the temporal coefficients between the moment, pressure, and turbulence equations, allowing solving fewer ROM equations at the prediction stage. We also implement special treatments for improving the stability and robustness of the Galerkin DEIM-ROM solution. We use the stalled turbulent flow over the NACA0012 airfoil with a relatively high Reynolds number (10^5) as the benchmark. We first verify our ROM solver implementation using a simple periodic flow case that has reached a full equilibrium state. Then, we evaluate the ROM solver's performance in predicting more challenging non-equilibrium flow. Non-equilibrium unsteady simulations are often needed in aerodynamic design optimization. In each optimization iteration, we need to change the flow conditions or airfoil shapes using initial conditions from a previous design. The flow will undergo a transient process before reaching equilibrium for the new conditions. Having the capability to predict non-equilibrium turbulent flow can help accelerate unsteady simulations in design optimization. However, non-equilibrium flow is more challenging to predict, and to what extent intrusive ROM can predict non-equilibrium turbulent flow is not well studied. The most original contribution of this paper is developing an intrusive, predictive ROM approach that uses a small amount of training data to predict non-equilibrium flow caused by a large sudden change in flow conditions (i.e., a five-degree change in angle of attack). To our best knowledge, the capability to predict such large variations has not been demonstrated in existing ROM studies.

The rest of this paper is organized as follows. In Section II, we elaborate on the full-order model and the proposed Galerkin DEIM-ROM approach. We then discuss the benchmark results for the NACA0012 airfoil in non-equilibrium conditions in Section III. Finally, in Section IV, we summarize our conclusions, perspectives, and future improvements.

2. Method

In this section, we elaborate on the proposed predictive ROM approach. We start with the formulations for unsteady turbulent flow simulations (full-order model), followed by the reduced-order model and DEIM to accelerate the computation of nonlinear terms. We also discuss the treatments for improving the robustness of ROM and DEIM-ROM solutions.

2.1. Full-order modeling (FOM)

We use OpenFOAM's [42] built-in solver `pisoFoam` to simulate the flow. It solves three-dimensional, unsteady turbulent flow, governed by the incompressible Navier–Stokes equations:

$$\nabla \cdot \boldsymbol{U} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U} \cdot \nabla)\boldsymbol{U} + \nabla p - \nabla \cdot \nu_{\text{eff}}(\nabla \boldsymbol{U} + [\nabla \boldsymbol{U}]^T) = 0, \tag{2}$$

where t is the time, p is the kinematic pressure, $\boldsymbol{U}$ is the velocity vector $\boldsymbol{U} = [u, v, w]$, $\nu_{\text{eff}} = \nu + \nu_t$ with ν and ν_t being the kinematic and turbulent eddy viscosity, respectively.

`pisoFoam` rewrites the viscous term in the momentum equation (2) as:

$$-\nabla \cdot \nu_{\text{eff}}(\nabla \boldsymbol{U} + [\nabla \boldsymbol{U}]^T) = -\nabla \cdot (\nu_{\text{eff}} \nabla \boldsymbol{U}) - \nabla \cdot (\nu_{\text{eff}}\ \text{dev}([\nabla \boldsymbol{U}]^T)), \tag{3}$$

where $\text{dev}([\nabla \boldsymbol{U}]^T)$ denotes the deviatoric part of the $[\nabla \boldsymbol{U}]^T$ tensor. This treatment enhances the numerical stability for full-order flow simulations. We observe that maintaining this treatment for the viscous term is critical to ensure the numerical stability for the reduced-order modeling (elaborated on in the next section).

To connect the turbulent viscosity to the mean flow variables, the Spalart–Allmaras (SA) model is used:

$$\frac{\partial \tilde{\nu}}{\partial t} + \nabla \cdot (\boldsymbol{U}\tilde{\nu}) - \frac{1}{\sigma}\{\nabla \cdot [(\nu + \tilde{\nu})\nabla \tilde{\nu}] + C_{b2}|\nabla \tilde{\nu}|^2\} - C_{b1}\tilde{S}\tilde{\nu} + C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2 = 0. \tag{4}$$

The turbulent eddy viscosity ν_t is computed from $\tilde{\nu}$ via:

$$\nu_t = \tilde{\nu}\frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}. \tag{5}$$

Spalart and Allmaras [43] provide a detailed description of the terms and parameters in this model.

To solve the continuity and momentum equations (1) and (2), the pressure implicit with splitting of operators (PISO) algorithm [44] is used. First, the momentum equation is discretized, and an intermediate velocity field is solved using the pressure field obtained from the previous iteration ($p^{t-\Delta t}$). Here we use a first-order Euler implicit time scheme as an example. Extending to other time schemes is straightforward.

$$a_P \boldsymbol{U}_P^t = -\sum_N a_N \boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} - \nabla p^{t-\Delta t} = \boldsymbol{H}(\boldsymbol{U}) - \nabla p^{t-\Delta t}, \tag{6}$$

where a is the coefficient resulting from finite-volume discretization, subscripts P and N denote the control volume cell and all of its neighboring cells, respectively, and

$$\boldsymbol{H}(\boldsymbol{U}) = -\sum_N a_N \boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} \tag{7}$$

represents the influence of velocity from all the neighboring cells and the previous iteration. A new, independent variable ϕ (face flux) is introduced to linearize the convective term:

$$\int_S \boldsymbol{U}\boldsymbol{U} \cdot \mathrm{d}\boldsymbol{S} = \sum_f \boldsymbol{U}_f \boldsymbol{U}_f \cdot \boldsymbol{S}_f = \sum_f \phi \boldsymbol{U}_f, \tag{8}$$

where the subscript f denotes the cell face. ϕ can be obtained from the previous iteration or an initial guess. Solving the momentum equation (6), we obtain an intermediate velocity field that does not yet satisfy the continuity equation.

Next, the continuity equation is coupled with the momentum equation to construct a pressure Poisson equation, and a new pressure field is computed. The discretized form of the continuity equation is

$$\int_S \boldsymbol{U} \cdot \boldsymbol{dS} = \sum_f \boldsymbol{U}_f \cdot \boldsymbol{S}_f = 0. \tag{9}$$

Instead of using a simple linear interpolation, $\boldsymbol{U}_f$ in this equation is computed by interpolating the cell-centered velocity $\boldsymbol{U}_P$—obtained from the discretized momentum equation (6)—onto the cell face as follows:

$$\boldsymbol{U}_f = \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f. \tag{10}$$

This idea of momentum interpolation was initially proposed by Rhie and Chow [45] and is effective in mitigating the oscillating pressure (checkerboard) issue resulting from the collocated mesh configuration. Substituting Eq. (10) into Eq. (9), we obtain the pressure Poisson equation:

$$\nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right). \tag{11}$$

Solving Eq. (11), we obtain an updated pressure field p^t. Then, this new pressure field is used to correct the face flux

$$\phi^t = \boldsymbol{U}_f \cdot \boldsymbol{S}_f = \left[\left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p^t)_f\right] \cdot \boldsymbol{S}_f, \tag{12}$$

and velocity field

$$\boldsymbol{U}^t = \frac{1}{a_P}[\boldsymbol{H}(\boldsymbol{U}) - \nabla p^t]. \tag{13}$$

The $\boldsymbol{H}(\boldsymbol{U})$ term depends on $\boldsymbol{U}$ but has not been updated so far. To account for this, one needs to repeatedly solve the Eq. (7) and Eqs. (11) to (13) (corrector loop), such that the velocity and pressure fields fully satisfy the continuity and momentum equations at each time step. In this paper, we perform two iterations for the corrector loop.

Then, we solve the turbulence equation (4) to obtain an updated turbulence viscosity (ν_t).

The above steps will be repeated for each time step in unsteady simulations.

2.2. Galerkin projection reduced-order modeling (ROM)

The ROM approach we propose consists of the following steps: full-order flow simulation, calculation of basis vectors using the methods of snapshot [1], calculation of Galerkin projection vectors and matrices, computation of temporal coefficients by solving the ROM equation, and reconstruction of full-order flow variables for post-processing.

We first run full-order flow simulations using CFD. During the simulation, we save the flow field snapshots every few iterations. Then, we assemble the flow variables such as velocity, pressure, face flux, and turbulent viscosity, into snapshot matrices, separately. Taking the velocity vector as an example (Eq. 14), its snapshot matrix is $\boldsymbol{S}_U \in \mathbb{R}^{n\times m}$, where n is the number of flow states, and m is the number of snapshots, and $n \gg m$. Note that we use all $\boldsymbol{U} = [u, v, w]$ components in the velocity snapshot matrix so $n = 3n_{\text{cell}}$ with n_{cell} being the number of mesh cells. We use the fluctuation component of state variables in the snapshot matrices, i.e., we subtract the instantaneous state variables with their mean values ($\hat{\boldsymbol{u}} = \boldsymbol{u} - \overline{\boldsymbol{u}}$). We observe that this treatment enhances the ROM's numerical stability.

$$\boldsymbol{S}_U^{n\times m} = \begin{bmatrix} \hat{\boldsymbol{u}}_{1,1} & \hat{\boldsymbol{u}}_{1,2} & \cdots & \hat{\boldsymbol{u}}_{1,m} \\ \hat{\boldsymbol{u}}_{2,1} & \hat{\boldsymbol{u}}_{2,2} & \cdots & \hat{\boldsymbol{u}}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\boldsymbol{u}}_{n,1} & \hat{\boldsymbol{u}}_{n,2} & \cdots & \hat{\boldsymbol{u}}_{n,m} \end{bmatrix}. \tag{14}$$

Once the snapshot matrices are constructed, we use them to compute the basis vectors (POD modes) for the flow variables. We use the singular value decomposition (SVD) approach to compute basis vectors. The SVD of a snapshot matrix $\boldsymbol{S} \in \mathbb{R}^{n\times m}$ can be written as:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{15}$$

where $\boldsymbol{\Sigma}$ is a $n \times m$ matrix that contains the singular values and $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ are orthogonal matrices whose columns are the left and right basis vectors, respectively. We will use the left matrix $\boldsymbol{U}$ because its columns are the function of eigenvectors for the snapshot correlation matrix $\boldsymbol{C} = \boldsymbol{S}^T\boldsymbol{S}$.

Instead of computing the basis vectors for all flow variables independently using Eq. 15, we solve the eigen-problem for the velocity correlation matrix $\boldsymbol{C}_U = \boldsymbol{S}_U^T \boldsymbol{S}_U$ and use these eigenvectors and eigenvalues to compute the basis vectors for all other flow variables, similar to [30, 46]. This treatment allows all the flow variables to share the same temporal coefficients, with the assumption that other flow variables are correlated with velocity. The benefit is that we need to solve only the reduced momentum equation and can use the temporal coefficients to reconstruct all other flow variables. This is especially useful for turbulent variables because the expansion of the eddy viscosity as the linear combination of spatial modes (basis vectors) is more flexible and less dependent on turbulent modeling; this approach applies to any RANS model. The details of the basis vector calculation are as follows.

First, we compute the correlation matrix for the velocity snapshot matrix as $\boldsymbol{C}_U = \boldsymbol{S}_U^T \boldsymbol{S}_U$. Next, we solve the eigen-problem for the $\boldsymbol{C}_U$ matrix using the scalable library for eigenvalue problem computations (SLEPc) library [47] and obtain its eigenvalues λ_i^U and eigenvectors $\boldsymbol{v}_i^U$ (the subscript denote the ith component). The ith velocity basis vector is then computed as:

$$\boldsymbol{\Phi}_i^U = \frac{1}{\sqrt{\lambda_i^U}} \boldsymbol{S}_U \boldsymbol{v}_i^U, \quad 1 \leq i \leq m. \tag{16}$$

The ith singular value is then computed as $\Sigma_i = \sqrt{\lambda_i}$.

As mentioned above, we will use the velocity eigenvalues and eigenvectors to compute the basis vectors for other flow variables. Taking the pressure basis vector as an example,

$$\boldsymbol{\Phi}_i^p = \frac{1}{\sqrt{\lambda_i^U}} \boldsymbol{S}_p \boldsymbol{v}_i^U, \quad 1 \leq i \leq m. \tag{17}$$

where $\boldsymbol{S}_p$ is the snapshot matrix for pressure. We can use a similar formulation for other variables.

We typically have a large number of snapshots (i.e., $m > 100$), so it is not necessary to keep all of the basis vectors. Because doing this will significantly increase the computational cost while adding flow modes that are not significant in producing accurate solutions. Note that the eigenvalues are stored in descending order, and the modes corresponding to the larger eigenvalues hold most of the energy present in the full-order flow. We usually drop basis vectors whose corresponding eigenvalues are less than 1% of the largest eigenvalue; however, trial-and-error is needed when dealing with a specific case (see detailed discussion in Sec. 3).

Once the basis vectors are computed, a full-order flow variable can be approximated by a linear combination of these basis vectors:

$$\boldsymbol{U} \approx \overline{\boldsymbol{U}} + \sum_{i=1}^{r} a_i(t) \boldsymbol{\Phi}_i^U \tag{18}$$

$$\boldsymbol{p} \approx \overline{\boldsymbol{p}} + \sum_{i=1}^{r} a_i(t) \boldsymbol{\Phi}_i^p \tag{19}$$

$$\boldsymbol{\phi} \approx \overline{\boldsymbol{\phi}} + \sum_{i=1}^{r} a_i(t) \boldsymbol{\Phi}_i^\phi \tag{20}$$

$$\boldsymbol{\nu}_t \approx \overline{\boldsymbol{\nu}}_t + \sum_{i=1}^{r} a_i(t) \boldsymbol{\Phi}_i^{\nu_t} \tag{21}$$

where $a_i(t)$ are the temporal coefficients for the weights, r is the number of kept modes ($r \leq m$), $\overline{\boldsymbol{U}}$, $\overline{\boldsymbol{p}}$, $\overline{\boldsymbol{\phi}}$, and $\overline{\boldsymbol{\nu}}_t$ are the mean velocity, pressure, face flux, and turbulent viscosity, respectively.

Substitute the above equations (18 to 21) into the momentum equation (2) and project it to $\boldsymbol{\Phi}_i^U$, we have the ROM equation:

$$\frac{\partial a_i}{\partial t} = C_i + \sum_{j=1}^{r} L_{ij} a_j + \sum_{j=1}^{r} \sum_{k=1}^{r} Q_{ijk} a_j a_k, \tag{22}$$

where the constant vector $\boldsymbol{C}$, linear matrix $\boldsymbol{L}$, and quadratic matrices $\boldsymbol{Q}$ are defined as:

$$C_i =< \mathbf{\Phi}_i^U, -\nabla\cdot(\overline{\boldsymbol{\phi}},\overline{\boldsymbol{U}}) - \nabla\overline{p} + \nabla\cdot\overline{\nu}_t(\nabla\overline{\boldsymbol{U}} + \mathrm{dev}[\nabla\overline{\boldsymbol{U}}]^T) + \nabla\cdot\nu(\nabla\overline{\boldsymbol{U}} + \mathrm{dev}[\nabla\overline{\boldsymbol{U}}]^T) > \tag{23}$$

$$\begin{aligned} L_{ij} =< \mathbf{\Phi}_i^U, -\nabla\cdot(\overline{\boldsymbol{\phi}},\mathbf{\Phi}_j^U) - \nabla\cdot(\mathbf{\Phi}_j^\phi,\overline{\boldsymbol{U}}) - \nabla\mathbf{\Phi}_j^p + \nabla\cdot\mathbf{\Phi}_j^{\nu_t}(\nabla\overline{\boldsymbol{U}} + \mathrm{dev}[\nabla\overline{\boldsymbol{U}}]^T) \\ +\nabla\cdot\overline{\nu}_t(\nabla\mathbf{\Phi}_j^U + \mathrm{dev}[\nabla\mathbf{\Phi}_j^U]^T) + \nabla\cdot\nu(\nabla\mathbf{\Phi}_j^U + \mathrm{dev}[\nabla\mathbf{\Phi}_j^U]^T) > \end{aligned} \tag{24}$$

$$Q_{ijk} =< \mathbf{\Phi}_i^U, -\nabla\cdot(\mathbf{\Phi}_j^\phi,\mathbf{\Phi}_k^U) + \nabla\cdot\mathbf{\Phi}_j^{\nu_t}(\nabla\mathbf{\Phi}_k^U + \mathrm{dev}[\nabla\mathbf{\Phi}_k^U]^T) > \tag{25}$$

where <> denotes the inner product.

As an example, if we substitute Eq. 19 into the pressure term (∇p) in the momentum equation and project to $\mathbf{\Phi}_i^U$:

$$< \mathbf{\Phi}_i^U, \nabla p >=< \mathbf{\Phi}_i^U, \nabla(\overline{\boldsymbol{p}} + \sum_{j=1}^{r} a_j\Phi_j^p) >=< \mathbf{\Phi}_i^U, \nabla\overline{\boldsymbol{p}} > + \sum_{j=1}^{r} a_j < \mathbf{\Phi}_i^U, \nabla\Phi_j^p > \tag{26}$$

where $< \mathbf{\Phi}_i^U, \nabla\overline{\boldsymbol{p}} >$ and $\sum_{j=1}^{r} a_j < \mathbf{\Phi}_i^U, \nabla\Phi_j^p >$ go to the constant C and linear $\sum_{j=1}^{r} L_{ij}a_j$ terms in the ROM equation, respectively. We can follow a similar approach to derive other terms in the ROM equation. We will pre-compute these vectors and matrices before solving Eq. 22.

In matrix form, Eq. 22 reads:

$$\frac{\partial \boldsymbol{a}}{\partial t} = \boldsymbol{C} + \boldsymbol{L}\boldsymbol{a} + \boldsymbol{a}^T\boldsymbol{Q}\boldsymbol{a} \tag{27}$$

Note that, during the projection procedure the continuity equation is automatically satisfied because the approximated face flux (Eq. 20) is a linear combination of the full-order face flux solutions (snapshot matrix) that already satisfy the continuity equation. So only the momentum equation is projected to obtain the ROM equations. Then, we can solve the unknown time-dependent coefficients $a_i(t)$ using the first-order explicit Euler scheme. Finally, after the temporal coefficients $a_i(t)$ are solved, we can substitute them back into Eqs. 18 to 21 to reconstruct the three-dimensional velocity and pressure fields.

The proposed ROM solution process is summarized as follows:

1. Full-order CFD results are obtained by solving Eqs. 1, 2, and 4.
2. The basis vectors are calculated using Eqs. 16 and 17.
3. The Galerkin projection vectors and matrices are computed from Eqs. 23, 24, and 25
4. The temporal coefficients $a_i(t)$ are solved based on Eq. 22.
5. The full-order flow fields are reconstructed using Eqs. 18 to 21.

2.3. Discrete empirical interpolation method (DEIM)

The above ROM formulation is computationally efficient if the number of kept modes (r) is small, e.g., $r < 20$. For complex cases (such as non-equilibrium turbulent flow), we may need more modes to fully predict the flow. In this case, the computation of the quadratic matrices becomes prohibitively slow because the cost scales with r^3. To alleviate this issue, one can use hyper-reduction methods such as DEIM [27] and gappy POD [29]. In this paper, we use DEIM to accelerate the nonlinear term computation in the momentum equation. We formulate a robust DEIM-ROM to ensure numerical stability.

First, we substitute the approximated states (Eqs. 18 to 21) to the momentum equation and project to $\mathbf{\Phi}^U$, similar to the previous subsection (Eq. 28). Here, we use a special treatment that merges the contribution of the mean state variables in the nonlinear term to the $\boldsymbol{C}$ vector and $\boldsymbol{L}$ matrix. We keep the fluctuation component of nonlinear term $\hat{\boldsymbol{N}}$ in its original inner-product format without separating the basis vectors $\mathbf{\Phi}$ and $\boldsymbol{a}$. This treatment greatly enhances the numerical stability of the DEIM-ROM simulation, as opposed to using the instantaneous nonlinear term.

$$\frac{\partial a_i}{\partial t} = C_i + \sum_{j=1}^{r} L_{ij}a_j + < \mathbf{\Phi}_i^U, \hat{\boldsymbol{N}}(\mathbf{\Phi}^U\boldsymbol{a}, \mathbf{\Phi}^\phi\boldsymbol{a}, \mathbf{\Phi}^{\nu_t}\boldsymbol{a}) >, \tag{28}$$

Algorithm 1 Greedy algorithm to compute the mask matrix $\boldsymbol{P}$ for DEIM

Input: Basis vector matrix for $\hat{\boldsymbol{N}}$: $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \cdots, \boldsymbol{\Phi}_p]$ where $\boldsymbol{\Phi} \in \mathbb{R}^{n\times p}$, $\boldsymbol{\Phi}_1 \in \mathbb{R}^{n\times 1}$, $\boldsymbol{\Phi}_{1:t} = [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \cdots, \boldsymbol{\Phi}_t]$

Output: Mask matrix: $\boldsymbol{P} = [\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_p]$, where $\boldsymbol{P} \in \mathbb{R}^{n\times p}$, $\boldsymbol{P}_1 \in \mathbb{R}^{n\times 1}$, $\boldsymbol{P}_{1:t} = [\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_t]$

1: $\boldsymbol{P} = 0$ ▷ Initialize $\boldsymbol{P}$ with zeros
2: $\gamma_1 \leftarrow \max(|\boldsymbol{\Phi}_1|)$ ▷ Find the row index of the maximal value in $\boldsymbol{\Phi}_1$
3: $\boldsymbol{P}_1[\gamma_1] = 1$ ▷ Assign 1 to the γ_1 row for $\boldsymbol{P}_1$
4: **for** $t = 1 : p-1$ **do** ▷ Loop over $p-1$
5: $\quad \boldsymbol{c}_t \leftarrow \boldsymbol{P}_{1:t}^T \boldsymbol{\Phi}_{1:t} \boldsymbol{c}_t = \boldsymbol{P}_{1:t}^T \boldsymbol{\Phi}_{t+1}$ ▷ Solve $\boldsymbol{c}_t \in \mathbb{R}^{t\times 1}$
6: $\quad \boldsymbol{R}_t = \boldsymbol{\Phi}_{t+1} - \boldsymbol{\Phi}_{1:t} \boldsymbol{c}_t$ ▷ Calculate the residual $\boldsymbol{R}_t \in \mathbb{R}^{n\times 1}$
7: $\quad \gamma_t \leftarrow \max(|\boldsymbol{R}_t|)$ ▷ Find the row index of the maximal value in $\boldsymbol{R}_t$
8: $\quad \boldsymbol{P}_{t+1}[\gamma_t] = 1$ ▷ Assign 1 to the γ_t row for $\boldsymbol{P}_{t+1}$

where the constant vector $\boldsymbol{C}$ and linear matrix $\boldsymbol{L}$ have the same formulation as in Eqs. 23 and 24, and the $\hat{\boldsymbol{N}}$ term is computed as follows:

$$\begin{aligned} \hat{\boldsymbol{N}}(\boldsymbol{\Phi}^U \boldsymbol{a}, \boldsymbol{\Phi}^\phi \boldsymbol{a}, \boldsymbol{\Phi}^{\nu_t} \boldsymbol{a}) &= -\nabla \cdot (\boldsymbol{\Phi}^\phi \boldsymbol{a}, \boldsymbol{\Phi}^U \boldsymbol{a}) + \nabla \cdot (\boldsymbol{\Phi}^{\nu_t} \boldsymbol{a})(\nabla(\boldsymbol{\Phi}^U \boldsymbol{a}) + \mathrm{dev}[\nabla(\boldsymbol{\Phi}^U \boldsymbol{a})]^T) \\ &= -\nabla \cdot (\boldsymbol{\phi} - \overline{\boldsymbol{\phi}}, \boldsymbol{U} - \overline{\boldsymbol{U}}) + \nabla \cdot (\nu_t - \overline{\nu}_t)(\nabla(\boldsymbol{U} - \overline{\boldsymbol{U}}) + \mathrm{dev}[\nabla(\boldsymbol{U} - \overline{\boldsymbol{U}})]^T), \end{aligned} \tag{29}$$

The above formulation for $\hat{\boldsymbol{N}}$ term can not be computed during the full-order flow simulations (FOM) because it depends on the time-averaged variables, e.g., $\overline{\boldsymbol{U}}$. Therefore, we compute $\hat{\boldsymbol{N}}$ using Eq. 29 after the FOM is done. We then construct the snapshot matrix using Eq. 30. Note that here the nonlinear term is already in the fluctuation form, so we do not subtract its mean field.

$$\boldsymbol{S}_N^{n\times l} = \begin{bmatrix} \hat{\boldsymbol{N}}_{1,1} & \hat{\boldsymbol{N}}_{1,2} & \cdots & \hat{\boldsymbol{N}}_{1,l} \\ \hat{\boldsymbol{N}}_{2,1} & \hat{\boldsymbol{N}}_{2,2} & \cdots & \hat{\boldsymbol{N}}_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\boldsymbol{N}}_{n,1} & \hat{\boldsymbol{N}}_{n,2} & \cdots & \hat{\boldsymbol{N}}_{n,l} \end{bmatrix}. \tag{30}$$

where l is the total number of snapshots, and we may need to save the nonlinear term more frequently than the state variables, so generally $l \geq m$. Then, we can compute the basis vector for $\hat{\boldsymbol{N}}$ using a similar approach mentioned above. Note that the nonlinear term's basis vectors also use the eigenvalues and eigenvectors of the velocity, as shown in Eqs. 16 and 17.

$$\hat{\boldsymbol{N}} \approx \boldsymbol{\Phi}^N \boldsymbol{c} \tag{31}$$

where $\boldsymbol{c} \in \mathbb{R}^{p\times 1}$ is the temporal coefficient with p being the kept modes for the nonlinear term (in general, $p \geq r$), and $\boldsymbol{\Phi}^N \in \mathbb{R}^{n\times p}$ is the basis vector matrix for $\hat{\boldsymbol{N}}$.

Then, we can construct an interpolation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times p}$ to determine the interpolation sample points. We will then use these sample points to interpolate all nonlinear terms in the flow field. $\boldsymbol{P}$ is a very sparse matrix with only one identity in each column, and it can be computed using the greedy algorithm, as shown in Algorithm 1. Note that we may have duplicated γ in the $\boldsymbol{P}$ matrix. If this happens, we run line 7 again and find the row index for the second largest value in $\boldsymbol{R}_t$ to ensure the indices in the $\boldsymbol{P}$ matrix are *unique*.

$$\boldsymbol{P}^{n\times p} = \begin{bmatrix} 0 & 0 & \cdots & 1 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{32}$$

We then multiply $\boldsymbol{P}^T$ to both sides of Eq. 31.

$$\boldsymbol{P}^T \hat{\boldsymbol{N}} \approx \boldsymbol{P}^T \boldsymbol{\Phi}^N \boldsymbol{c} \tag{33}$$

Next, we solve a dense linear equation to obtain $\boldsymbol{c}$ from the above equation. We will find the $\boldsymbol{c}$ coefficient to match the nonlinear term values at the sample points only, instead of all the mesh cells. We can then substitute the $\boldsymbol{c}$ solution back to Eq. 31.

$$\hat{\boldsymbol{N}} \approx \underbrace{\boldsymbol{\Phi}^N [\boldsymbol{P}^T \boldsymbol{\Phi}^N]^{-1}}_{\boldsymbol{A} \in \mathbb{R}^{n \times p}} \underbrace{\boldsymbol{P}^T \hat{\boldsymbol{N}}}_{\boldsymbol{b} \in \mathbb{R}^{p \times 1}} = \boldsymbol{A}\boldsymbol{b} \tag{34}$$

The mechanism of the above formulation allowing a much cheaper nonlinear term calculation is that, the $\boldsymbol{P}^T \hat{\boldsymbol{N}}$ term essentially projects the high-dimensional nonlinear vector $\hat{\boldsymbol{N}}$ to a much lower dimension p. So, we need to compute the nonlinear terms only at the p sample points where $\boldsymbol{P}$ is not zero. In other words, we do not need to compute $\hat{\boldsymbol{N}}$ for every mesh cell, we only need to compute it at p interpolation points. Then, we multiply the nonlinear term value at these p points with the $\boldsymbol{A}$ matrix to interpolate the nonlinear term values for the rest of the mesh cells. The $\boldsymbol{A}$ matrix will be pre-computed once, and the number of numerical operations is proportional to p (as opposed to r^3 for the original nonlinear term formulation in Eq. 25) and independent of the mesh size. This will greatly speed up the nonlinear term calculation.

If we subsitute Eq. 34 back to the ROM equation Eq. 28, we have the final version of the DEIM-ROM equation:

$$\underbrace{\frac{\partial \boldsymbol{a}}{\partial t}}_{r \times 1} = \underbrace{\boldsymbol{C}}_{r \times 1} + \underbrace{\boldsymbol{L}}_{r \times r} \underbrace{\boldsymbol{a}}_{r \times 1} + \underbrace{\boldsymbol{D}}_{r \times p} \underbrace{\boldsymbol{b}}_{p \times 1} \tag{35}$$

where the constant vector $\boldsymbol{C}$ and linear matrix $\boldsymbol{L}$ are the same as before and the new vectors and matrices are defined as:

$$D_{i,j} = < \boldsymbol{\Phi}_i^U, \boldsymbol{A}_j > \tag{36}$$

$$b_j = \boldsymbol{P}_j^T \hat{\boldsymbol{N}}(\boldsymbol{\Phi}^U \boldsymbol{a}, \boldsymbol{\Phi}^\phi \boldsymbol{a}, \boldsymbol{\Phi}^{\nu_t} \boldsymbol{a}) \tag{37}$$

where $\boldsymbol{A}_j$ denotes the jth column of the $\boldsymbol{A}$ matrix defined in Eq. 34, and $\boldsymbol{P}_j^T$ is the jth row of the $\boldsymbol{P}^T$ matrix.

To compute the nonlinear term at the interpolation points ($\boldsymbol{P}^T \hat{\boldsymbol{N}}$), we utilize the `fvMeshSubset` library in OpenFOAM and create a subset mesh that contains the cells for the interpolation points, as well as two levels of surrounding cells to ensure the correct stencil for the nonlinear operators. Once the subset mesh is created, we use it to compute the $\boldsymbol{P}^T \hat{\boldsymbol{N}}$ term in Eq. 37 when solving the ROM equation (35), and the original full-scale mesh is no longer used. To be more specific, for each ROM time step, we use the latest $\boldsymbol{a}$ coefficient to compute the fluctuation component of state variables, i.e., $\boldsymbol{\Phi}^U \boldsymbol{a}$, $\boldsymbol{\Phi}^\phi \boldsymbol{a}$, and $\boldsymbol{\Phi}^{\nu_t} \boldsymbol{a}$. We then assign these updated state variable fluctuations to the subset mesh variables and compute the nonlinear terms at the interpolation points using Eq. 37. For 2D problems, the total number of cells in the subset mesh is on the order of $10p$, so its computation cost is significantly lower than that for the full-scale mesh cells, which typically have $\sim$100 000 cells.

The proposed DEIM-ROM solution process is summarized as follows:

1. We first run FOM by solving Eqs. 1, 2, and 4 and save the state variable snapshots.
2. After FOM is done, we calculate the nonlinear term snapshots using Eq. 29.
3. We compute the state basis vectors for both state variables and nonlinear terms.
4. The Galerkin projection vectors and matrices are computed from Eqs. 23, 24, and 36.
5. The temporal coefficients $a_i(t)$ are solved using the first-order Euler scheme based on Eq. 35. For each time step, we compute the nonlinear term only at p interpolation points using Eq. 37.
6. The full-order flow fields are reconstructed using Eqs. 18 to 21.

2.4. Numerical stability for reduced-order modeling with DEIM

One of the major challenges in developing an intrusive unsteady ROM solver is numerical stability. The instability in the ROM simulation may result in flow divergence, gradual increase or decrease of the periodic flow magnitude, or a phase shift of the time series, especially for simulations with a long time period. See more detailed discussion in [48–54]. In this subsection, we discuss the main stability considerations in our ROM implementations.

The most important factor is the numerical consistency between the ROM and FOM formulations. We observe that using consistent numerical operators between FOM and ROM greatly stabilizes the simulation. For the viscous term in the momentum equation, if we use the original formulation for the viscous term $-\nabla \cdot \nu_{\text{eff}}(\nabla \boldsymbol{U} + [\nabla \boldsymbol{U}]^T)$, the ROM simulation will become unstable and diverge. As mentioned above, this instability will be circumvented by using the FOM-consistent viscous term: $-\nabla \cdot (\nu_{\text{eff}} \nabla \boldsymbol{U}) - \nabla \cdot (\nu_{\text{eff}} \operatorname{dev}([\nabla \boldsymbol{U}]^T))$, as shown in Eq. 3. Therefore, we used the FOM-consistent viscous operators when computing the linear and nonlinear matrices in Eqs. 24 and 25, as well as the nonlinear term in DEIM (Eq. 29). In addition to the viscous term, we need to carefully formulate the convective term. In FOM, the convective term is linearized by introducing an independent variable $\boldsymbol{\phi}$, as shown in Eq. 8. The FOM continuity equation enforces the face flux, instead of the velocity, to be divergence-free. Therefore, in the ROM formulations (Eqs. 23 to 25 and 29), we use ϕ-based formulations for the convective term in the momentum equation. For example, the $\nabla \cdot (\boldsymbol{\Phi}^{\phi}, \boldsymbol{\Phi}^{U})$ operator denotes the divergence of the $\boldsymbol{\phi}$ basis vector (defined on the face center) times the velocity basis vector that is interpolated to the cell face center. We find that this treatment ensures the conservative of the ROM formulation and enhances its numerical stability.

Another important factor is the treatment of the state variables in the ROM formulations. As mentioned before, we subtract the instantaneous state variables with their mean values when constructing the snapshot matrix, e.g., Eq. 14. There are mean state variables in the constant vector $\boldsymbol{C}$ and linear matrix $\boldsymbol{L}$ of the ROM equation (27). In the ROM equation solution process, we solve for the fluctuation component of the state variables, instead of their absolute values. We find that having the mean state variables in the $\boldsymbol{C}$ vector and $\boldsymbol{L}$ matrix enhances the ROM simulation's stability. The above treatment of mean flow variables is especially important for the DEIM's stability. As shown in Eq. 29, we use the fluctuation component of the state variables as the input to compute the fluctuation component of the nonlinear term, instead of using the instantaneous state variables to compute the instantaneous nonlinear term, in the DEIM-ROM formulation. This treatment has the advantage that the constant vector and linear matrix in the DEIM formulation are the same as the ones in the original ROM equation. Therefore, the DEIM-ROM's numerical stability is partially inherited from the original ROM formulation. In addition, during the DEIM-ROM simulation, we solve for the fluctuation component of the highly nonlinear $\hat{\boldsymbol{N}}$ term, instead of its instantaneous value. We find that this helps us circumvent the numerical instability in the DEIM-ROM simulation.

We observe satisfactory numerical stability for short-term simulations using our DEIM-ROM formulation. However, further improvement can be made by incorporating more stabilization methods to enhance long-term simulation stability. Examples are the least square Petrov–Galerkin formulation, regulation of the basis vectors, the addition of the artificial dissipation term, and the randomization of DEIM interpolation points. We will consider these treatments in future work.

3. Results and Discussion

In this section, we evaluate our ROM solver's performance using stalled turbulent flow over the NACA0012 airfoil. We first test our ROM solver using a simple periodic flow case. We run FOM until the mean state variables no longer change and use the fully evolved flow field as the initial condition. We then run FOM with the *same* flow conditions (i.e., same Reynolds number and angle of attack) and use the first portion of FOM data to train ROM. This simple setup allows us to verify our ROM implementation, and we will use it as a reference for the following cases. Next, we consider more challenging non-equilibrium cases, where we run FOM with a *different* flow condition (e.g., different angle of attack). Because of this change, the flow will undergo a transient process before reaching equilibrium for the new flow conditions. We then use the transient FOM data to train a ROM to accelerate the rest of the simulation. We consider non-equilibrium flow caused by both small perturbations and large variations. The small variation cases (e.g., the angle of attack changes less than 0.5 degrees) are mainly used in perturbation response analyses, e.g., gust wind.

Table 1: Boundary conditions for the airfoil CFD simulations (FOM). The simulation domain is a rectangular box.

	Inlet	Top	Bottom	Front	Back	Outlet
$\boldsymbol{u}$	$\boldsymbol{u} = (u_{\text{in}}, 0, 0)$	$\nabla \boldsymbol{u} \cdot \boldsymbol{n}$=0	$\nabla \boldsymbol{u} \cdot \boldsymbol{n}$=0	Symmetry	Symmetry	$\nabla \boldsymbol{u} \cdot \boldsymbol{n}$=0
p	$\nabla p \cdot \boldsymbol{n}$=0	$\nabla p \cdot \boldsymbol{n}$=0	$\nabla p \cdot \boldsymbol{n}$=0	Symmetry	Symmetry	0
$\tilde{\nu}$	$\tilde{\nu}_{\text{in}}$	$\nabla \tilde{\nu} \cdot \boldsymbol{n}$=0	$\nabla \tilde{\nu} \cdot \boldsymbol{n}$=0	Symmetry	Symmetry	$\nabla \tilde{\nu} \cdot \boldsymbol{n}$=0

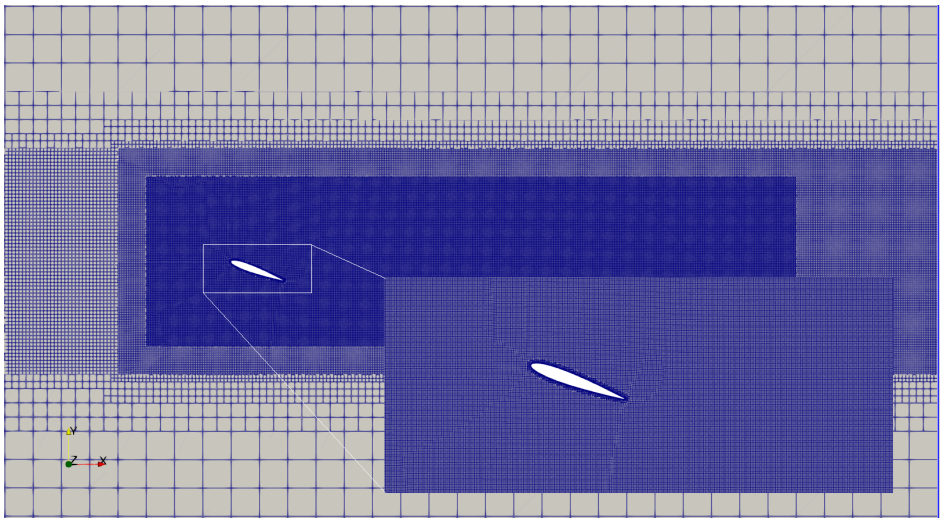

Figure 1: Unstructured mesh for the NACA0012 airfoil with 243 838 cells. We use three levels of refined mesh for the wake region. The incoming flow is in the x direction, and the airfoil have a angle of attack $\alpha = 20°$. The simulation domain is 60 times the airfoil chord length in the streamwise direction.

The large variation case (i.e., the angle of attack changes by 5 degrees) mimics a design optimization process where we simulate new designs using initial conditions from the previous design.

3.1. ROM verification using equilibrium unsteady flow

As mentioned above, we use the NACA0012 airfoil as the benchmark. The airfoil has a chord length of $c = 1.0$ m and an angle of attack (α) of 20°. The computational domain is a rectangular box with $60c$, $20c$, and $0.1c$ in the x, y, and z directions, respectively. The leading edge of the airfoil is $10c$ downstream from the inlet. This relatively large computational domain is chosen to allow the wake to dissipate freely before hitting the simulation's outer boundaries. The flow velocity at the inlet is set as $u_{\text{in}} = 1.0$ m/s, and the corresponding Reynolds number is 10^5. We use the second-order implicit Euler method for temporal discretization. We use the second-order upwind and central schemes for discretizing the convective and viscous terms, respectively. These discretization settings are typical for `pisoFoam`. The detailed FOM boundary conditions are summarized in Table 1.

We use OpenFOAM's built-in mesh generation tool `snappyHexMesh` to generate an unstructured mesh, as shown in Fig. 1. Note that we add three layers of mesh refinement to capture the details of the airfoil wake and small-scale flow modes. We find that these small-scale modes are important for representing the complete flow physics of full-order flow simulations. We evaluate the impact of mesh density and associated discretization errors on the simulation results. To this end, we generate four meshes with increasing density and compute the functions of interest (C_d, C_l, and C_m), as shown in Table 2. We find that C_d, C_l, and C_m do not change significantly when increasing the mesh density from L3 to L4. Therefore, we use the L3 mesh with 243,924 cells as the benchmark in this study.

As mentioned above, we use a fully evolved flow at $Re = 1 \times 10^5$ and $\alpha = 20°$ as the initial condition. Note that we do not include the initial field computation runtime in our FOM and ROM because it is a one-time cost. Then, we perform FOM for 10 s, approximately 5.5 periods of vortex shedding cycles. The FOM time step is 0.001 s with a Courant–Friedrichs–Lewy (CFL) number ∼0.63. The snapshots are collected in a time window of 2 s and cover approximately the first 1.2 periods of vortex shedding. We save the instantaneous

Table 2: Mesh refinement study for $Re = 10^5$ and $\alpha = 20°$. The functions of interest do not change significantly when increasing the mesh refinement from L3 to L4. Therefore, we use the L3 mesh in our study.

Level	Mesh cells	C_d	C_l	C_m
L1	63,952	0.3778	0.8983	0.1317
L2	139,428	0.3697	0.8978	0.1304
L3	243,924	0.3614	0.8519	0.1207
L4	539,982	0.3601	0.8462	0.1200

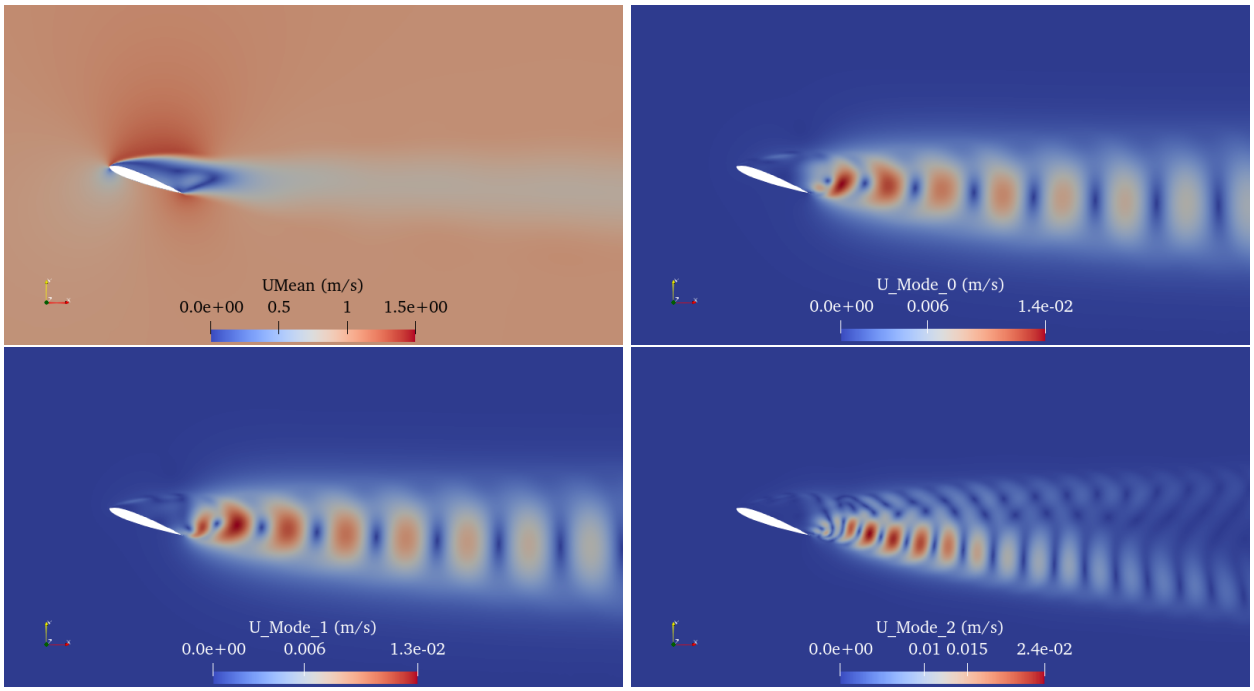

Figure 2: Mean and the first three modes of velocity for the equilibrium case. The angle of attack $\alpha = 20°$, and the Reynolds number is 10^5.

flow fields for velocity, pressure, face flux, and turbulent viscosity every 0.02 s, so we have a total number of 100 snapshots for training ROM. The number of snapshots is selected based on the balance between the accuracy and memory and computational costs. Specifically, we gradually increase the number of snapshots until the ROM accuracy no longer changes. Note that we use a similar approach to select the best number of modes and DEIM points in the following. We then perform SVD to compute the modes from these 100 snapshots; the mean and first three modes for the velocity and pressure are shown in Figs. 2 and 3. We observe finer flow structures for the modes having smaller singular values (the higher the modes, the smaller of singular values). Considering the balance between accuracy and efficiency, we keep the first 20 modes for both ROM and DEIM-ROM. The corresponding singular values for the 20^{th} mode is 0.2%. In other words, we drop the basis vectors whose corresponding singular values are less than 0.2% of the largest one, as shown in Fig. 4. We use 40 interpolation points for DEIM-ROM.

After the modes are computed, we compute the vectors and matrices for the ROM equation (Eq. 27). Then, we can solve the ROM equation for the temporal coefficients $a_i(t)$. Finally, we can substitute them into Eqs. 18 to 21 to reconstruct the flow fields. The simulation is conducted in serial using the Intel Xeon W-1370 CPU running at 2.9 Hz. Table 3 summarizes the performance of FOM, ROM, and DEIM-ROM. The FOM simulation takes 39 124 s (wall-clock runtime), and the training and prediction portions take 7384 s and 31 740 s, respectively. The ROM performance can be broken down into three parts. The calculation of basis vectors (BasisVec; Eqs. 16 and 17), Galerkin projection vectors and matrices (ProjMat; Eqs. 23, 24, and 25), and POD temporal coefficients $a_i(t)$ (SolveROM; Eq. 22) take 320 s, 700 s, and 25 s, respectively. The total runtime for ROM is 1045 s, and the speed-up factor between FOM (prediction) and ROM is 30.4.

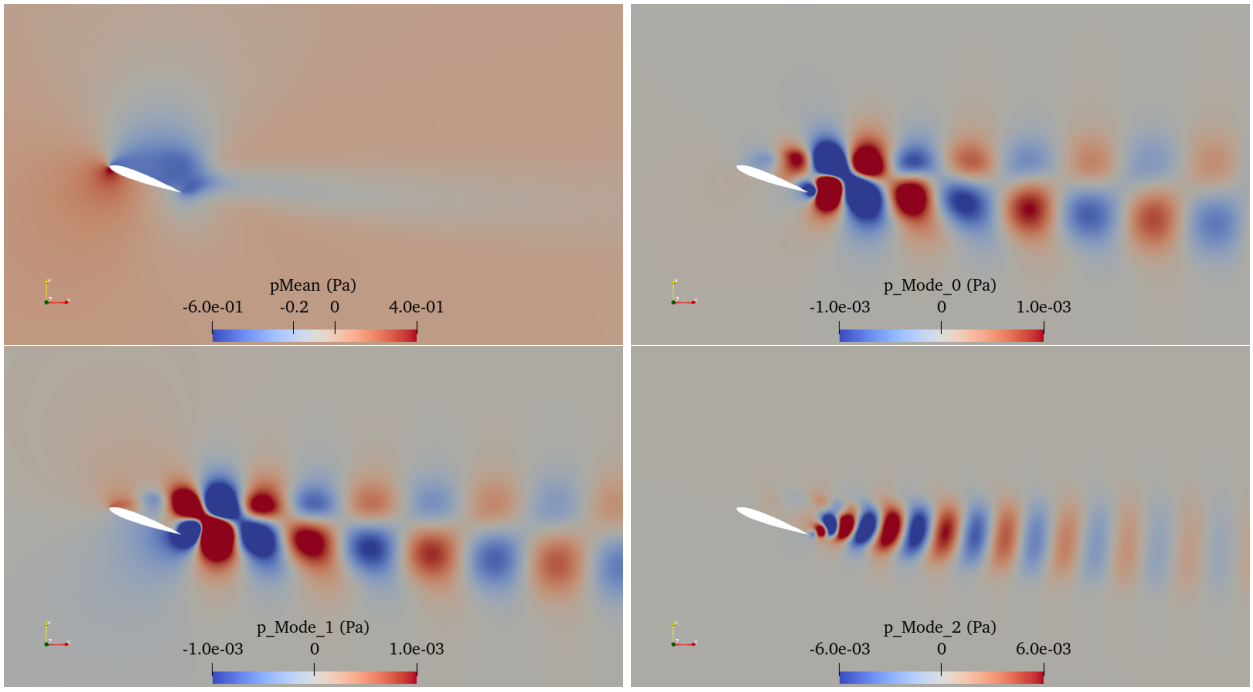

Figure 3: Same as Fig. 2 but for the pressure.

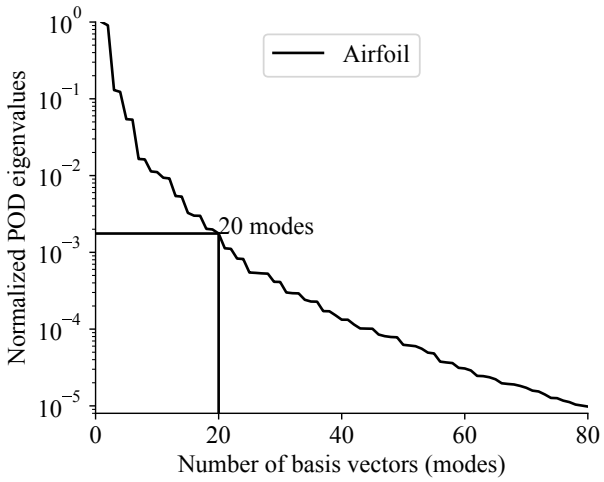

Figure 4: Normalized POD eigenvalues for the equilibrium case. We keep the first 20 modes for ROM.

The BasisVec calculation between ROM and DEIM-ROM is similar. However, the DEIM-ROM's ProjMat is much faster (352 s vs 700 s) because DEIM-ROM's computational cost no longer depends on r^3 and mesh size. Because the DEIM-ROM needs to re-compute nonlinear terms when solving $a_i(t)$, its SolveROM cost is slightly higher than ROM (40 s vs 25 s). Due to the gain in ProjMat, DEIM-ROM's speed-up factor increases to 44.5. Note that both ROMs exhibit reasonable speedup, and the extra benefit of DEIM is insignificant for the equilibrium case because we use only 20 modes. However, for the non-equilibrium cases (shown in the next subsection), the original ROM's computational cost will become prohibitive because we need many more modes, and using DEIM is necessary. Next, we will evaluate the accuracy of the proposed ROM and DEIM-ROM approaches.

The comparisons of flow fields among FOM, ROM, and DEIM-ROM are shown in Fig. 5. Overall, the flow fields agree reasonably well. For example, the separation bubble and wake structure predicted by ROM

Table 3: Breakdown of computational cost for FOM, ROM, and DEIM-ROM (equilibrium case). ROM and DEIM-ROM achieve a speedup factor of 30.4 and 44.5, respectively. The speed-up factor is calculated as the runtime ratio between the FOM (prediction) and ROM (including the computation of basis vectors, projection matrices, and POD temporal coefficients).

FOM	39 124 s
Training	7384 s
Prediction	31740 s
ROM	1045 s
BasisVec	320 s
ProjMat	700 s
SolveROM	25 s
Speed-up factor	**30.4**
DEIM-ROM	713 s
BasisVec	321 s
ProjMat	352 s
SolveROM	40 s
Speed-up factor	**44.5**

and DEIM-ROM are almost identical to that from FOM at 0.5 and 1.0 T, where T is the period of vortex shedding. Note that we observe similar accuracy between ROM and DEIM-ROM, so we show only the DEIM-ROM field prediction results in the following.

To further quantify the comparison, we plot the time series of flow variables at two probe points in Fig. 6. Here the probe points are located in the airfoil wake (coordinates: [2.5, -0.25] and [3.5, 0]) and marked as "Probe 1" and "Probe 2" in Fig. 5. The amplitude and phase of the velocity at probe points computed by DEIM-ROM (dots) agree reasonably well with FOM (lines). Though the velocity time series at probe 2 exhibit more oscillation frequencies, their agreements are as good as probe 1. This indicates that ROM is able to handle problems with wide-banded frequencies. In addition to the flow field variables, we plot the comparisons of integral variable time series, i.e., drag, lift, and pitching moment in Fig. 7 and observe reasonably good agreements between FOM and DEIM-ROM. Figure 8 shows the comparisons of pressure distribution over the airfoil between FOM and DEIM-ROM. Again, we observe reasonably good surface pressure agreements at various time instances within a period. This ensures an accurate prediction of integral variables such as drag and lift. The good agreement in the pressure prediction justifies our treatment of correlating the POD temporal coefficients between velocity, pressure, and turbulent variables, elaborated on in Sec. 2.2. This finding is slightly different from Stabile et al. [46] and Stabile and Rozza [55], where the authors found solving a reduced pressure Poisson equation was necessary for the pressure prediction accuracy. To capture a more quantitative wake structure, we plot the velocity variation in the y (vertical) direction in Fig. 9. The velocity is extracted from a slice $0.5c$ downstream from the airfoil trailing edge. We find low-speed regions (wake) oscillating in the vertical (y) direction in a time period. The above wake structures are well captured by ROM and DEIM-ROM at various time instances.

Finally, we use the normalized L^2 error to further quantify the discrepancy between FOM and ROM. For instance, the L^2 error for the velocity component u reads:

$$||\text{Error}||_{L^2} = \sqrt{\frac{< (u_{\text{FOM}} - u_{\text{ROM}}), (u_{\text{FOM}} - u_{\text{ROM}}) >_{L^2}}{< \overline{u}_{\text{FOM}}, \overline{u}_{\text{FOM}} >_{L^2}}} \tag{38}$$

where u_{FOM} is the FOM velocity, u_{ROM} is the ROM velocity, $\overline{u}_{\text{FOM}}$ is the mean (time-averaged) FOM velocity for normalization, and $<>_{L^2}$ denotes the L^2 norm (inner product). Note that we use the velocity magnitude for normalization, instead of each velocity component. For the whole flow field $<>_{L^2}$ calculation, we loop over all mesh cells. For the probe points, we loop over all time series data.

Tables 4 and 5 show the time series prediction errors. Overall, the errors are small. The L^2 error in C_d is higher than C_l and C_m, indicating that the drag force is more challenging to predict. We speculate this is because the C_d value is more sensitive to the turbulence intensity and requires an accurate surface turbulence prediction. Note that we do not directly solve a reduced turbulence model in our ROM formulation, instead, we correlate the turbulence variable's temporal coefficient with the velocity. This treatment

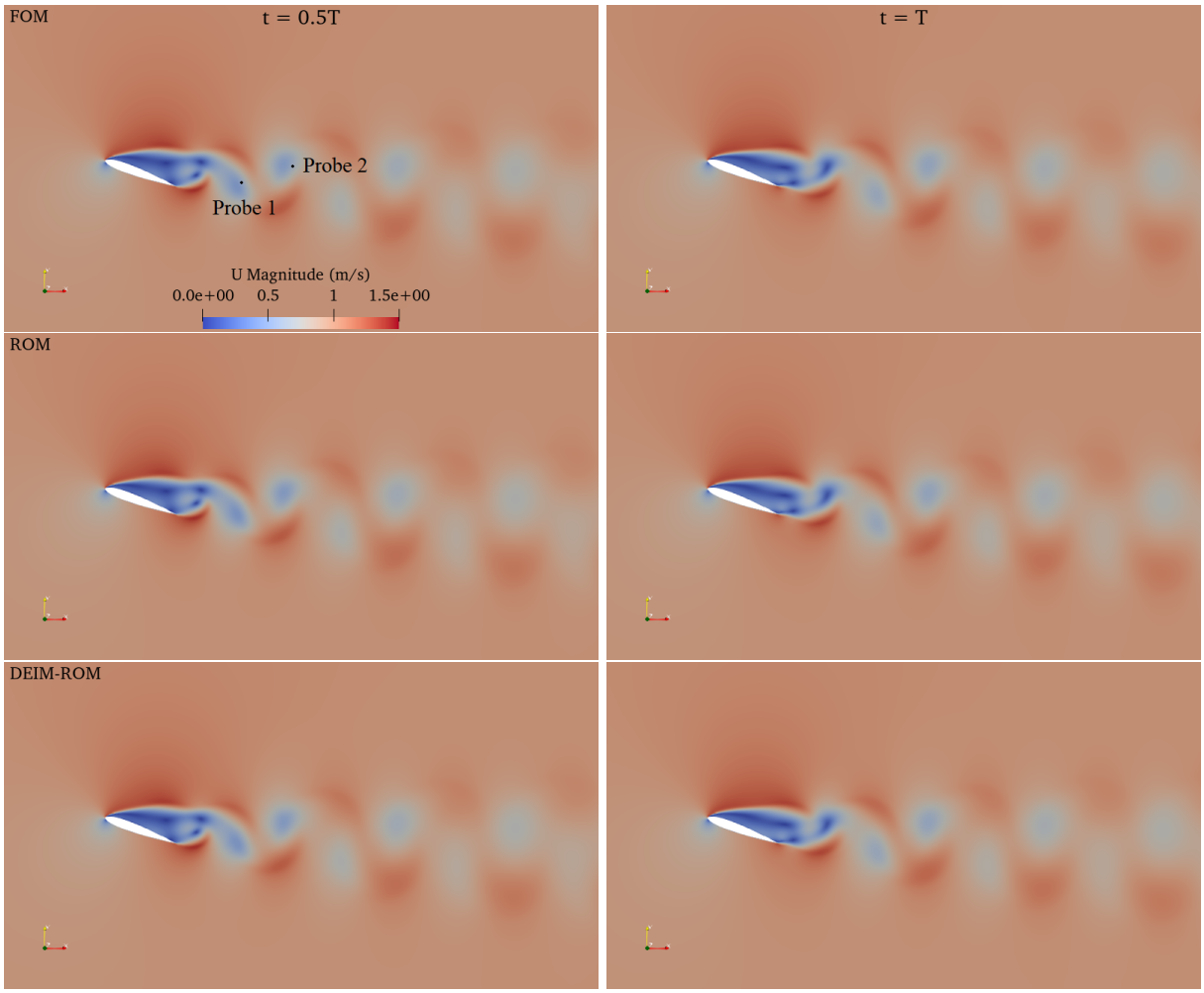

Figure 5: Comparisons of flow fields at 0.5 T (left) and 1.0 T (right) for the equilibrium case, where T is the period of vortex shedding. We achieve reasonably good agreements among FOM (top), ROM (middle), and DEIM-ROM (bottom).

provides acceptable surface turbulence prediction for the equilibrium case. For the velocity time series, we find that probe 1's errors are slightly larger than probe 2's, probably because it is closer to the airfoil. We also observe that the error in the vertical velocity (v) is larger than the one in the streamwise velocity (u). However, both errors are less than $\sim 10^{-2}$, so it is not distinguishable from Fig. 6.

Figure. 10 shows the time evolution of flow field L^2 errors ($\times 10^{-4}$). Again, the overall errors are small. The pressure and vertical velocity have relatively large errors than other variables. We find that the errors grow slowly over time, but the maximal L^2 error remains under 0.05. This trend is similar to that reported by Stabile and Rozza [55]. The authors evaluated their Galerkin ROM solver using a flow-over-cylinder case and found that the error increased with time. Table 4 shows the comparison of time-averaged drag, lift, and pitching moment between FOM and ROM. The ROM and DEIM-ROM predictions match the FOM data by four and three digits, respectively. This level of error is acceptable for unsteady aerodynamic analysis and design. Overall, our ROM implementation is accurate and can accelerate equilibrium unsteady aerodynamic simulations. As mentioned above, we find that the DEIM-ROM's error is only slightly higher than ROM's. In the next subsection, we will show only the DEIM-ROM results.

Note that the good agreement for the verification case is primarily because we use a fully evolved flow as the initial condition and run FOM and ROM with the same flow condition. Therefore, the basis vectors already cover all the possible flow structures. Other approaches, such as DMD or data-driven ROM, may have similar performance for this simple case. However, we will show in the next subsection that our intrusive ROM can also use non-equilibrium data for training and predict non-equilibrium flows.

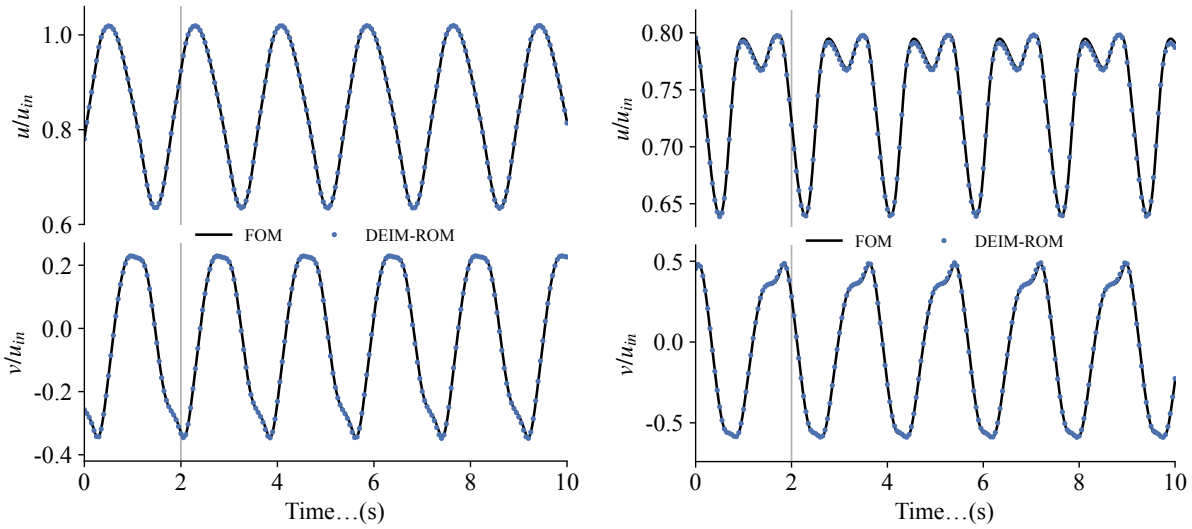

Figure 6: Comparison of velocity time-series at probe 1 (left) and probe 2 (right) for the verification case. We achieve reasonably good agreements between FOM and DEIM-ROM. We use the first 2 s data for training, as indicated by the grey line.

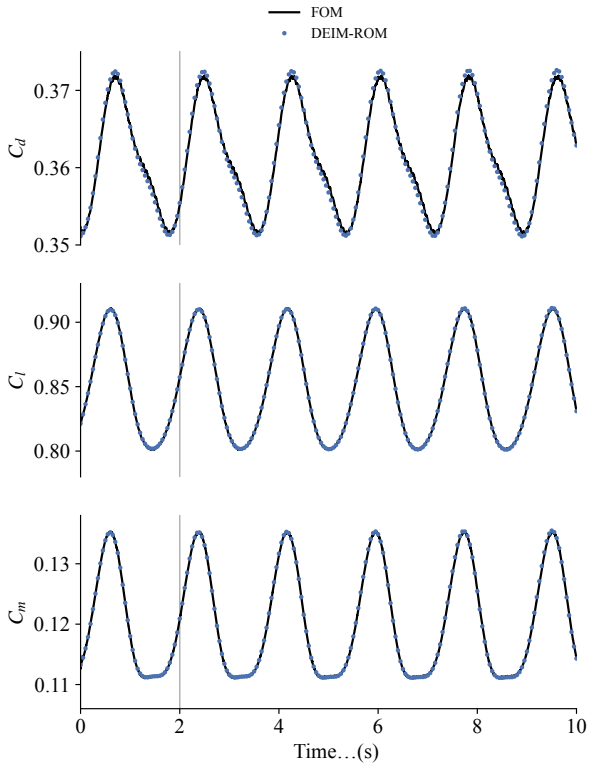

Figure 7: Comparisons of drag (top), lift (mid), and pitching moment (bottom) for the equilibrium case. We achieve reasonably good agreements between FOM and DEIM-ROM. We use the first 2 s data for training, as indicated by the grey line.

3.2. Predictive ROM for non-equilibrium unsteady flow

In this subsection, we consider a more challenging flow condition: non-equilibrium. The non-equilibrium flow settings (such as the airfoil geometry, mesh, and computational domain) are similar to the equilibrium case. We first run FOM at $Re = 10^5$ and $\alpha = 20^\circ$ until the mean flow variables no longer change. We then use the fully evolved flow as the initial condition. Next, we run FOM at a different flow condition (e.g.,

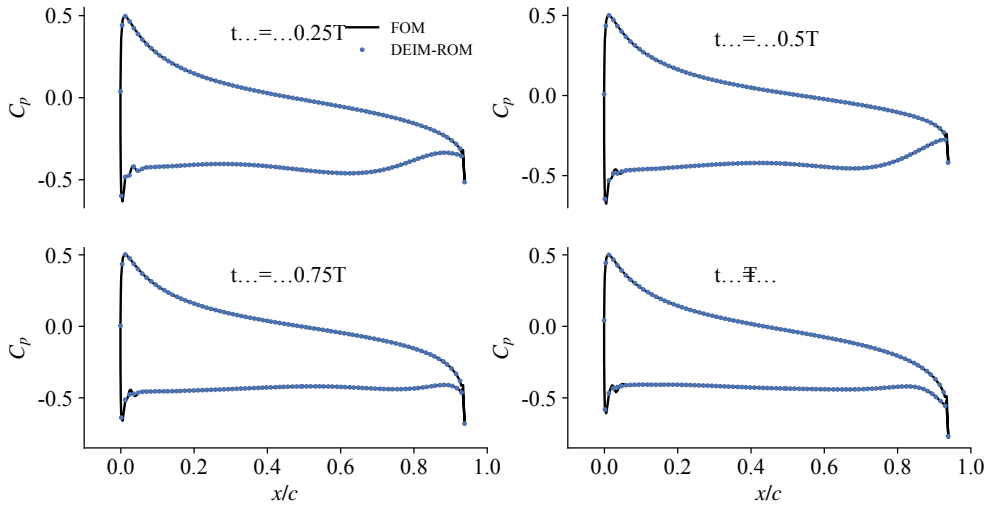

Figure 8: Comparison of pressure distribution on the airfoil surface for the verification case. We achieve reasonably good agreements between FOM and DEIM-ROM at various time instances within one flow period.

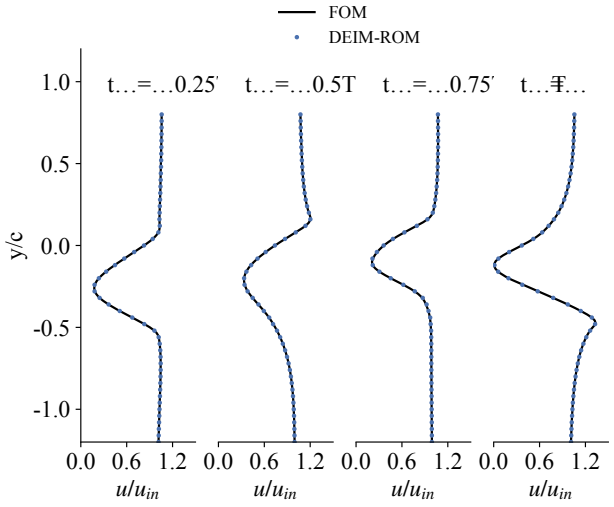

Figure 9: Comparisons of velocity profile $0.5c$ downstream of the trailing edge (verification case). We observe reasonably good agreements between FOM and DEIM-ROM at various time instances within one period.

Table 4: Normalized L^2 error ($\times 10^{-4}$) for C_d, C_l, C_m, u, and v time-series are small (verification case).

Case	C_d	C_l	C_m	Probe 1		Probe 2	
				u	v	u	v
ROM	18.07	4.469	8.498	14.07	115.1	8.897	127.5
DEIM-ROM	18.42	5.886	12.58	23.27	240.4	16.77	181.9

different Reynolds number or angle of attack). Because of this change, the flow will undergo a transient process before reaching a new equilibrium state. We use a small portion of FOM data (transient process; run at new flow conditions) to train a ROM to accelerate the rest of the flow simulation.

We consider two types of flow condition changes: Reynolds number and angle of attack, and call them Reynolds number and angle of attack predictive cases, respectively. The non-equilibrium predictive ROM can be used to accelerate unsteady flow simulations for two aerospace applications. The first application is the analysis of perturbation response. For example, aircraft may be subject to a sudden change in the flow

Table 5: Comparisons of time-averaged drag, lift, and pitching moment for the verification case. ROM and DEIM-ROM match with FOM for almost all four digits.

	FOM	ROM	DEIM-ROM
C_d	0.3614	0.3614	0.3614
C_l	0.8519	0.8519	0.8520
C_m	0.1207	0.1207	0.1207

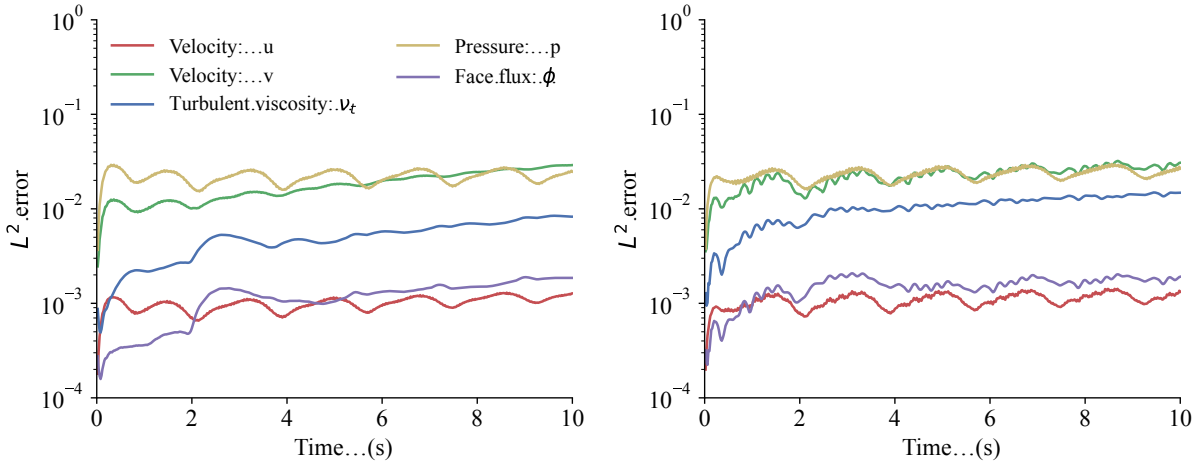

Figure 10: Normalized L^2 error of the whole flow field for the verification case: ROM (left) and DEIM-ROM (right). L^2 errors of state variables remain at a low level and grow slowly.

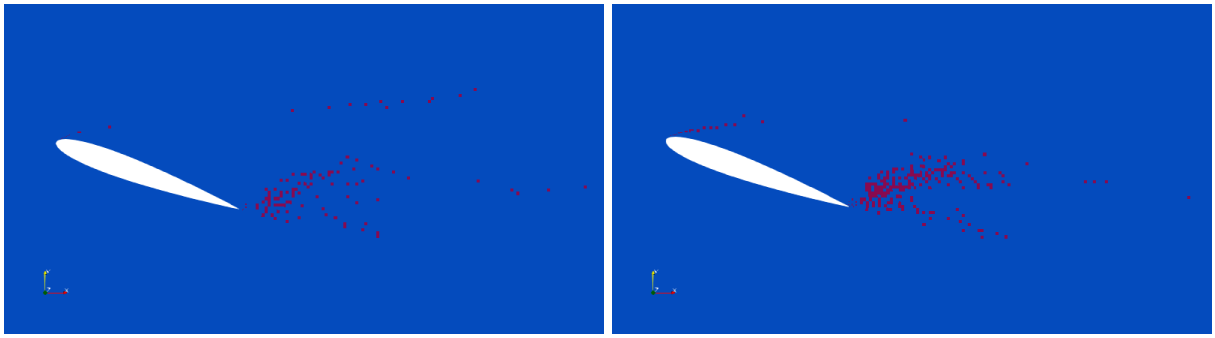

Figure 11: DEIM points distribution in the $x-y$ plane for the non-equilibrium case. We use 120 and 230 DEIM points for $Re = 2 \times 10^5$ (left) and for $Re = 3 \times 10^5$ (right) respectively.

condition (e.g., gust wind). One needs to simulate and analyze the non-equilibrium unsteady flow response before it reaches a new steady-state flight. The second application is aerodynamic shape optimization. In each optimization iteration, we need to simulate unsteady flow with a new design (e.g., new airfoil shape, Reynolds number, or angle of attack) using the initial conditions from the previous design. Therefore, each optimization iteration must simulate unsteady flows undergoing a transient process. As mentioned above, we consider non-equilibrium flow caused by both small perturbations and large variations. For perturbation response analysis, the flow condition change is typically small. So we can use the first few seconds of data for training (see Secs. 3.2.1 and 3.2.2). For design optimization, a more drastic flow change can happen. Therefore, we will use the data from a pre-equilibrium state for training (see Sec. 3.2.3).

3.2.1. Reynolds number predictive cases (small perturbation)

We use the fully evolved flow field at $Re = 10^5$ as the initial condition and predict the flow at five different Reynolds numbers, i.e., $Re = [1.5,\ 2,\ 2.5,\ 3,\ 3.5] \times 10^5$. The angle of attack is kept at $\alpha = 20°$. Without the loss of generality, we show the results for $Re = 2 \times 10^5$ and 3×10^5 only. Note that we change the Reynolds

Table 6: Breakdown of computational cost for FOM and DEIM-ROM (non-equilibrium, Reynolds number predictive cases). DEIM-ROM achieves a speedup factor of 13.6. The speed-up factor is calculated as the runtime ratio between the FOM (prediction) and ROM (including the computation of basis vectors, projection matrices, and POD temporal coefficients).

FOM	61 593 s
Training	23149 s
Prediction	38444 s
DEIM-ROM	2838 s
BasisVec	1123 s
ProjMat	1000 s
SolveROM	715 s
Speed-up factor	**13.6**

number by changing the fluid kinematic viscosity ν, instead of the incoming flow velocity. Therefore, the whole flow field is immediately impacted by the flow condition change.

Because the flow evolution is more complex, we need to use more modes to better represent the flow. To balance the accuracy and computational efficiency, we use 100 modes for the non-equilibrium case. Using 100 modes is made possible by using the DEIM-ROM formulation. We used 120 and 230 DEIM-ROM interpolation points for the predictive cases $Re = 2 \times 10^5$ and 3×10^5, respectively. The optimal number of DEIM interpolation points is selected based on trials and errors, as mentioned above. Figure 11 shows the distribution of the DEIM points. We find that most of the DEIM points are clustered in the wake region, where the vortex energy is strongest (see Fig. 2). Only a small portion of them are located near the leading edge. We observe a similar DEIM point distribution pattern for all simulations.

We run FOM for a total of 16 s. Then, we save the instantaneous flow fields (snapshots) every 0.02 s, use the first 6 s of the FOM data (approximately three vortex shedding periods) for ROM training, and predict the flow for the rest of 10 s. Note that we exclude the first snapshot at 0.02 s to eliminate the impact of flow spin-up. In other words, we use a total number of 299 snapshots for training. We find that this treatment improves the accuracy of ROM prediction for non-equilibrium cases. Compared with the above verification case, we need more data for training. This is because the flow is no longer in equilibrium, so using one flow period's data is insufficient to capture the trend in the transient process. In this study, we run the simulations for a relatively short time (16 s) to demonstrate the predictive ROM's capability. To evaluate the ROM's capability for long-term prediction, e.g., $\mathcal{O}(100)$ s, we will further improve the robustness of our ROM implementation in future work (e.g., using the Petrov–Galerkin formulation).

The breakdown of computational cost for DEIM-ROM is summarized in Table 6. The time cost is the average cost among the five cases. We achieve a speedup factor of 13.6. As expected, the speedup factor is less than the ROM verification case. This is because we use more modes and snapshots and the calculation of basis vector and projection matrices are much more expensive. Moreover, we need more DEIM interpolation points, so the solveROM step is more costly (note that we used only 40 interpolation points in the verification case).

Figure 12 shows the comparison of drag, lift, and pitching moment time series between FOM and DEIM-ROM. Overall, the drag, lift, and pitching moment increase with increasing the Reynolds number. If the Reynolds number increases by a factor of two ($Re = 2 \times 10^5$), the time series undergo a relatively short transient increase and eventually reach a new equilibrium state within 16 s. However, with a slightly larger Reynolds number change ($Re = 3 \times 10^5$), the transient process takes much longer, and the flow does not reach a full equilibrium state within 16 s. As expected, the non-equilibrium flow is more challenging to predict, and the agreement between FOM and ROM is not as good as the verification case. The overall trend is well captured, although the phase and the amplitude of the time series predicted by ROM are slightly off. This level of time-series prediction error is similar to that reported in [33, 54, 56].

The comparison of velocity time series at probe point 2 is shown in Fig. 13. The change in the Reynolds number has a larger impact on the streamwise velocity u (top figures) than the vertical velocity v (bottom figures). Also, a larger Reynolds number change results in a larger variation in the transient process (when comparing the left and right figures), especially for u. The ROM time series agree better in v than u. Again, the error is larger than the ROM verification case.

Figure 14 depicts the surface pressure distribution within 8.5 to 10 s (approximately one flow period

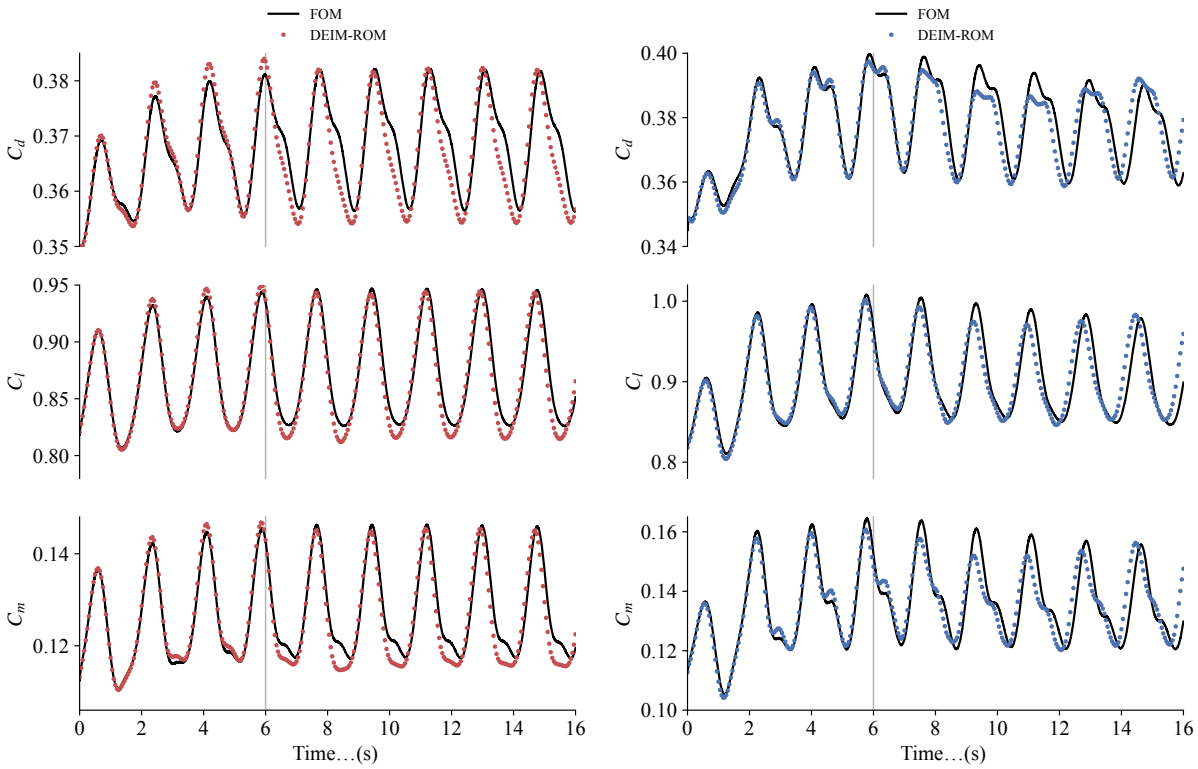

Figure 12: Comparisons of drag (top), lift (mid), and pitching moment (bottom) for the non-equilibrium case. We use $Re = 10^5$ as the initial condition to predict $Re = 2 \times 10^5$ (left) and 3×10^5 (right). We use the first 6 s data for training, as indicated by the grey line. We achieve reasonable agreements between FOM and DEIM-ROM.

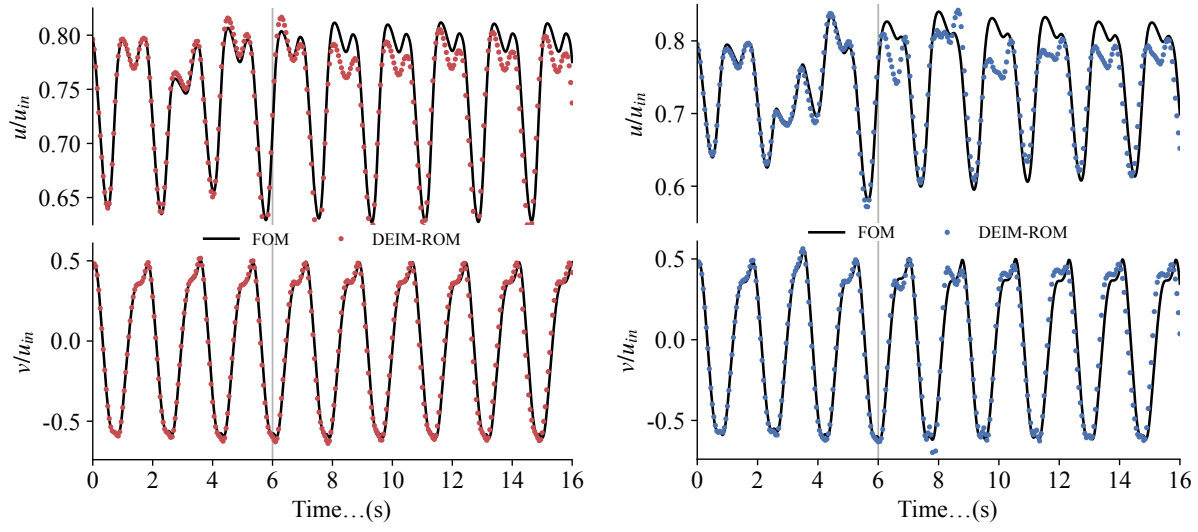

Figure 13: Same as Fig. 12 but for the velocity time-series at probe point 2.

in the prediction stage). Overall, the pressure agreements between FOM and ROM are better than the time-series results shown in Figs. 12 and 13. As expected, the ROM pressure prediction has larger errors for a larger Reynolds number change, especially near the trailing edge. We also evaluate the variations of streamwise velocity in the y direction in Fig. 15. Overall, ROM captures the wake structure well. We find that ROM performs better in predicting spatial velocity distribution (Fig. 15) than the temporal variation

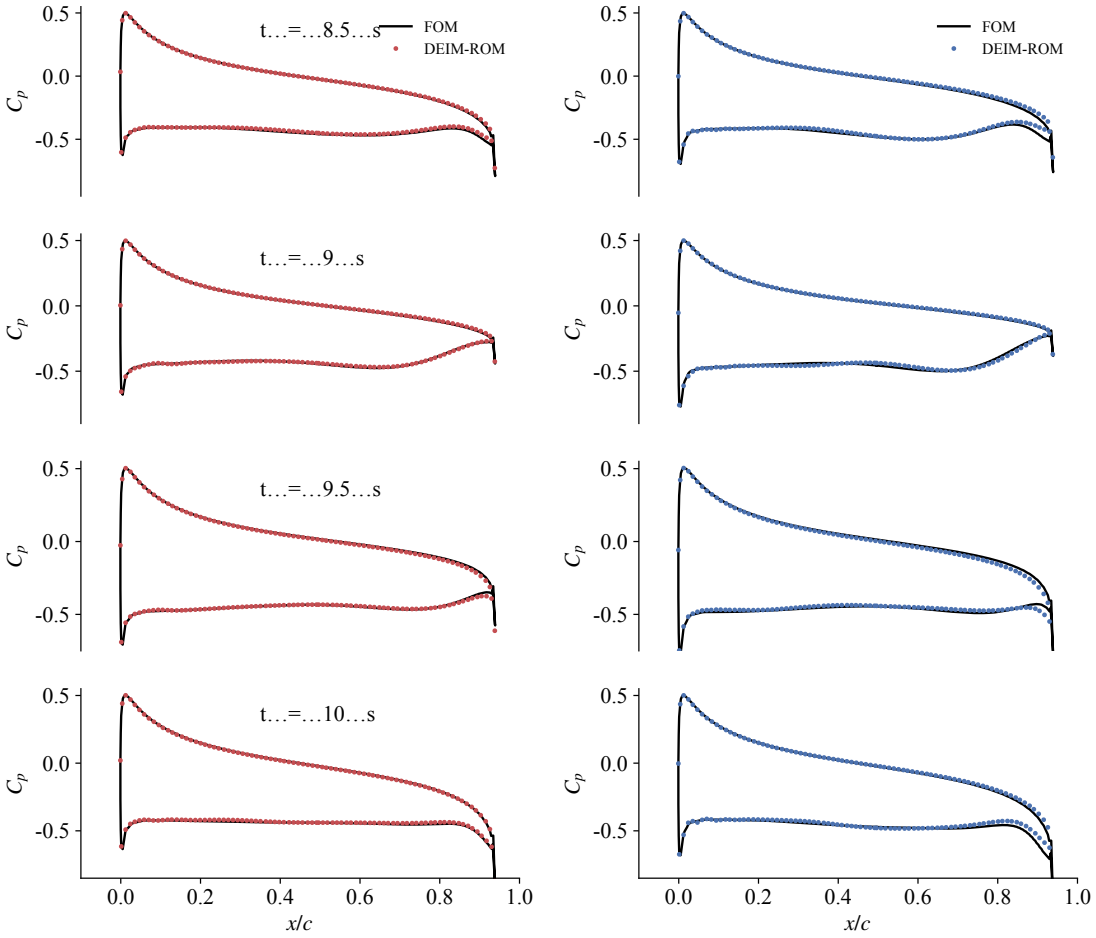

Figure 14: Comparison of pressure distribution on the airfoil surface (non-equilibrium). We use $Re = 10^5$ as the initial condition to predict $Re = 2 \times 10^5$ (left) and 3×10^5 (right). We achieve good agreements between FOM and DEIM-ROM.

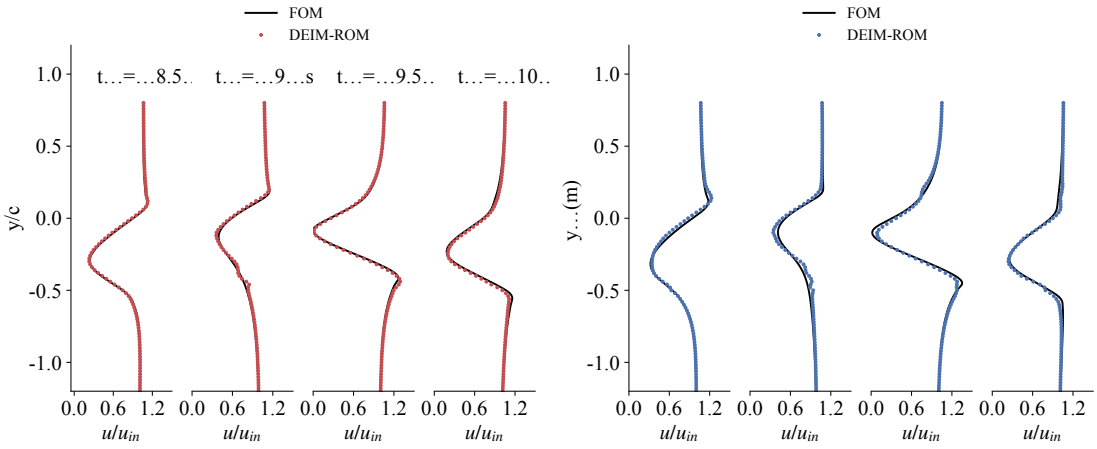

Figure 15: Comparisons of velocity profile $0.5c$ downstream of the trailing edge (non-equilibrium): We use $Re = 10^5$ as the initial condition to predict $Re = 2 \times 10^5$ (left) and 3×10^5 (right). We observe good agreements between FOM and DEIM-ROM.

(Fig. 13).

The normalized L^2 errors of time series and flow fields with various predictive Reynolds numbers are

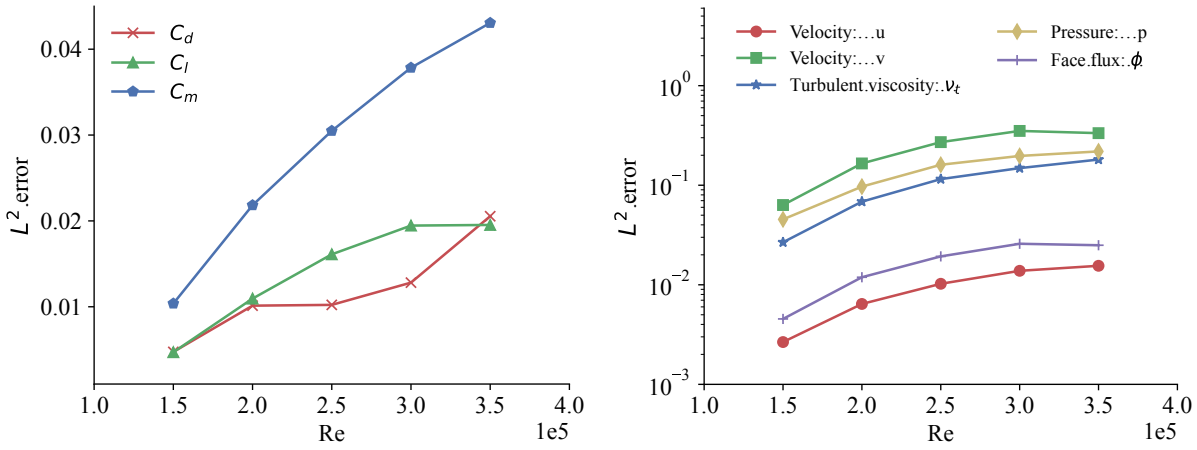

Figure 16: Normalized L^2 errors of C_d, C_l, and C_m time series (left) and whole flow field (right) for the Reynolds number predictive cases (non-equilibrium).

given in Fig. 16. Overall, the L^2 errors grow as the Reynolds number change increases. This is expected because a larger Reynolds number change results in a longer transient process (see Fig. 12 right), making the flow fields more complex and challenging to predict. For the time series, the error in C_m is the largest and grows most rapidly when the predictive Reynolds number increases, compared with C_d and C_l. This trend differs from the verification case. As shown in Table. 4, the error for C_d time series was the largest. For the field variables, we find the trend is similar to the verification case: the v and p errors are the largest among other flow variables. However, the field error growth rate is less than the time series one. This indicates that the surface variables are more challenging to predict than the field variables for the non-equilibrium case.

3.2.2. Angle of attack predictive cases (small perturbation)

In this subsection, we consider non-equilibrium flow caused by a small change in the angle of attack. Instead of changing the far field velocity, we rotate the airfoil to change the angle of attack. Therefore, the flow around the airfoil is immediately impacted by the angle of attack change. This predictive setup is different from the Reynolds number cases shown in the previous subsection because it needs to change the CFD mesh. To ensure the new mesh has the same mesh cells and topology, we use pyGeo to rotate the airfoil surface mesh. pyGeo is an open-source tool that uses the free-form deformation (FFD) approach to parameterize the geometry [57]. Then, we use IDWarp to propagate the surface mesh deformation to the volume mesh. IDWarp uses an inverse-distance weighting approach to smoothly deform the volume mesh and can preserve the near-wall mesh orthogonality [58].

We use the fully evolved flow field with $\alpha = 20°$ as the initial condition and predict five different angles of attack, i.e., $\alpha = [20.1°, 20.2°, 20.3°, 20.4°, 20.5°]$. The Reynolds number is kept at $Re = 10^5$. Again, we mainly analyze the predictive results for $\alpha = 20.3°$ and $20.5°$. Similar to the Reynolds number predictive cases, we use 100 modes and 300 snapshots, which cover 6 s of FOM data. We use 160 and 238 DEIM interpolation points for the $\alpha = 20.3°$ and $20.5°$ cases, respectively. We then use the trained ROM to predict the flow for the rest of the 10 s.

The breakdown of computational cost for DEIM-ROM is summarized in Table 7. Again, the time cost is the average cost among the five cases. We achieve a speedup factor of 13.4. This number is much lower than the ROM verification case, but it is slightly lower than the speed-up factor from the Reynolds number predictive cases. This is because more DEIM interpolation points are used in the angle of attack predictive cases.

Fig. 17 shows the comparisons of drag, lift, and pitching moment time series between FOM and DEIM-ROM. All surface variables decrease with increasing the angle of attack, confirming that the airfoil is in a stall condition. Changing the angle of attack has a larger impact on the surface variables than changing the Reynolds number (i.e., comparing Figs. 12 and 17). The flow undergoes a longer transient period and does not reach a full equilibrium state within 16 s. This is probably because we change the airfoil pitch

Table 7: Breakdown of computational cost for FOM and DEIM-ROM (non-equilibrium, angle of attack predictive cases). DEIM-ROM achieves a speedup factor of 13.4. The speed-up factor is calculated as the runtime ratio between the FOM (prediction) and ROM (including the computation of basis vectors, projection matrices, and POD temporal coefficients).

FOM	60 763 s
Training	23524 s
Prediction	37239 s
DEIM-ROM	2772 s
BasisVec	1032 s
ProjMat	1010 s
SolveROM	730 s
Speed-up factor	**13.4**

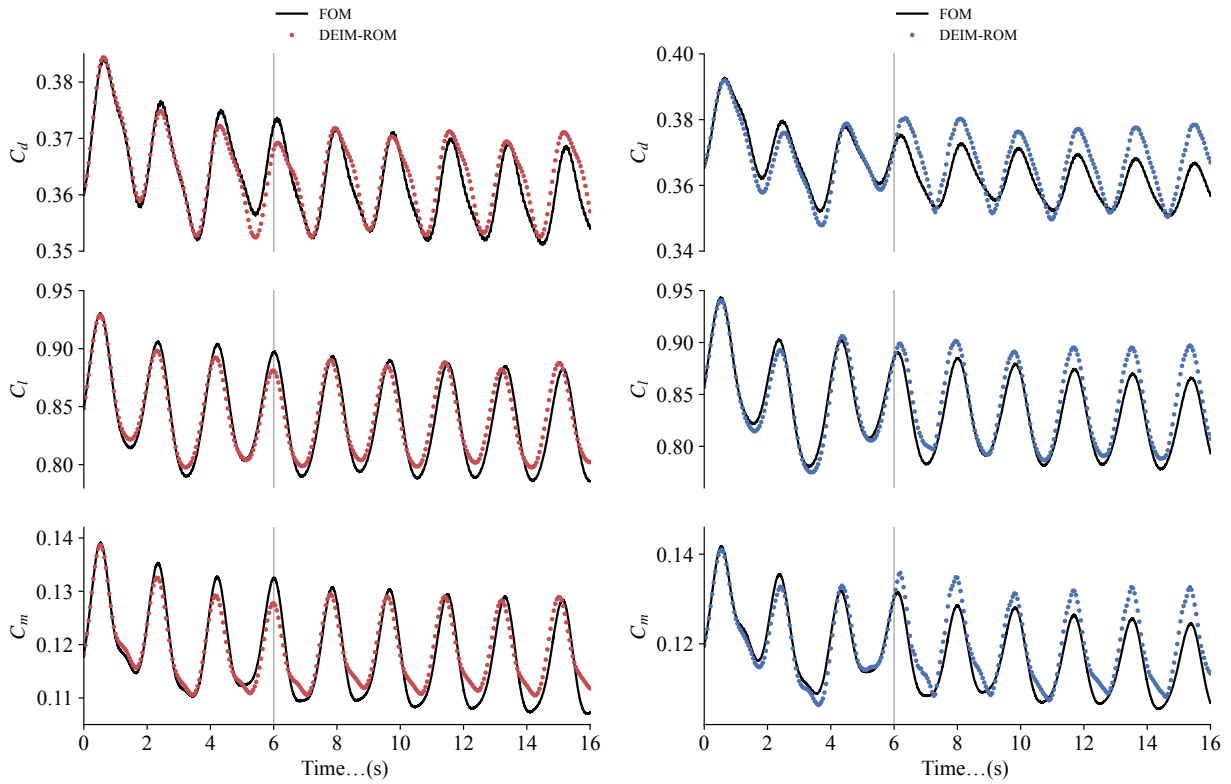

Figure 17: Comparisons of drag (top), lift (mid), and pitching moment (bottom) for the non-equilibrium cases. We use $\alpha = 20°$ as the initial condition to predict $\alpha = 20.3°$ (left) and $20.5°$ (right). We use the first 6 s data for training, as indicated by the grey line. We achieve good agreements between FOM and DEIM-ROM.

angle and the mesh, both directly impact the surface variables. Because the flow becomes more complicated, the agreement between FOM and ROM is less satisfactory than the Reynolds number predictive cases. For example, the oscillation magnitudes of C_d, C_l, and C_m are over-predicted for the $\alpha = 20.5°$ case (Fig. 17 right). However, ROM successfully captures the overall decreasing trend of the time series. Again, the time-series prediction error is similar to the one reported in recent intrusive ROM studies [33, 54, 56].

Fig. 18 plots the velocity time series at probe point 2 for $\alpha = 20.3°$ and $20.5°$. In contrast to the surface variables, the variations of velocity time series are less sensitive. This indicates that changing the angle of attack has a larger impact on the surface variables than the field variables. Again, we find that the agreement of v is better than that of u, and ROM captures the velocity variation reasonably well.

The comparisons of surface pressure distribution for $\alpha = 20.3°$ and $20.5°$ are shown in Fig. 19. The surface pressure agreement between FOM and ROM is slightly worse than the Reynolds number predictive cases (comparing Figs. 14 and 19). Moreover, changing the angle of attack introduces relatively large errors

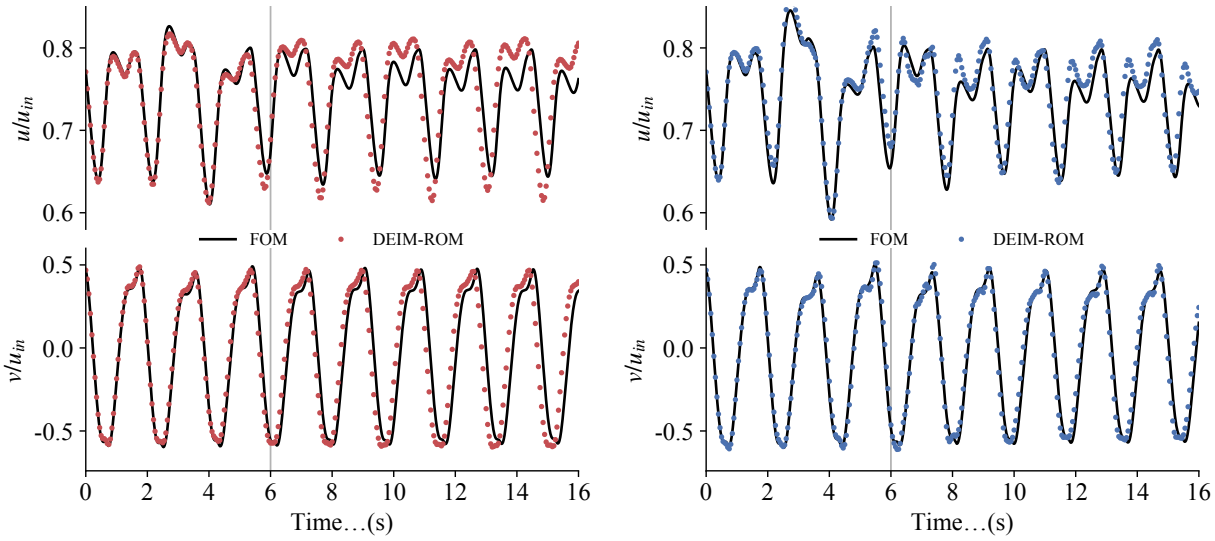

Figure 18: Same as Fig. 17 but for the velocity time-series at probe point 2.

near the leading and trailing edges. However, changing the Reynolds number mostly impacts the ROM prediction accuracy near the trailing edge. Fig. 20 plots the comparisons of streamwise velocity variations in the y direction. ROM captures the overall wake structure, although the error is generally larger than the Reynolds number predictive cases. Fig. 21 shows the L^2 errors for C_d, C_l, and C_m, and field variables. Overall, we observe the same trend as the Reynolds number predictive cases, i.e., the error grows as the change of angle of attack increases with C_m, p, and v having relatively large errors.

3.2.3. Angle of attack predictive cases (large perturbation)

In this subsection, we consider non-equilibrium flow caused by a large variation in the angle of attack. As mentioned above, design optimization typically uses initial conditions from the previous design to simulate the flow for a new design; these two designs can have significantly different flow conditions. To mimic such a scenario, we use the fully evolved flow field at the angle of attack $\alpha = 20°$ as the initial condition and predict the flow for $\alpha = 25°$. The Reynolds number is kept at $Re = 10^5$.

Due to the large variation in the angle of attack, the time series of the surface and field variables (see Fig. 22 left and Fig. 23 left) have much larger changes at the beginning than the small perturbation cases (see Figs. 17 and 18). These longer and irregular transient phrases pose greater challenges for the predictive ROM. There are two main reasons. First, the mean flow changes drastically during the initial transient phase (e.g., 0 to 10 s in Fig. 22 left). If we still follow a similar setting as the small perturbation case and use the first 6 s of ROM data for training, the mean flow variables ($\overline{U}$, $\overline{p}$, $\overline{\phi}$, and $\overline{\nu_t}$) between the training and prediction will be significantly different. This will break our current ROM formulation because the constant vector $\boldsymbol{C}$ and linear matrix $\boldsymbol{L}$ (Eq. 22) assume the mean flow variables are time-invariant or change only slightly. Note that this is the case for the small perturbation cases (e.g., Figs. 12 and 17). The second reason is that the flow pattern changes rapidly during the initial transient phase, so the trained modes will not be able to represent the quasi-equilibrium flow later.

Given the above two reasons, we use the FOM data from 10 s to 16 s to train the ROM instead of the first 6 s. Similar to the small angle of attack perturbation predictive cases, we use 100 modes, 300 snapshots, and 240 DEIM interpolation points. We then predict the flow for the following 18 s (i.e., from 16 s to 34 s). The ROM speed-up factor is 24.3. We get a higher speed-up factor than the small perturbation cases because we have a longer prediction time here.

Figure 22 right shows the zoom-in time series of drag, lift, and moment. Our ROM prediction agrees with the FOM data reasonably well. Note that a similar capability was demonstrated using non-intrusive DMD approaches [39, 40]. The authors used pre-equilibrium flow for training and predicted *full equilibrium* flow. Our study is a further step forward because both training and prediction flow fields are not in fully equilibrium

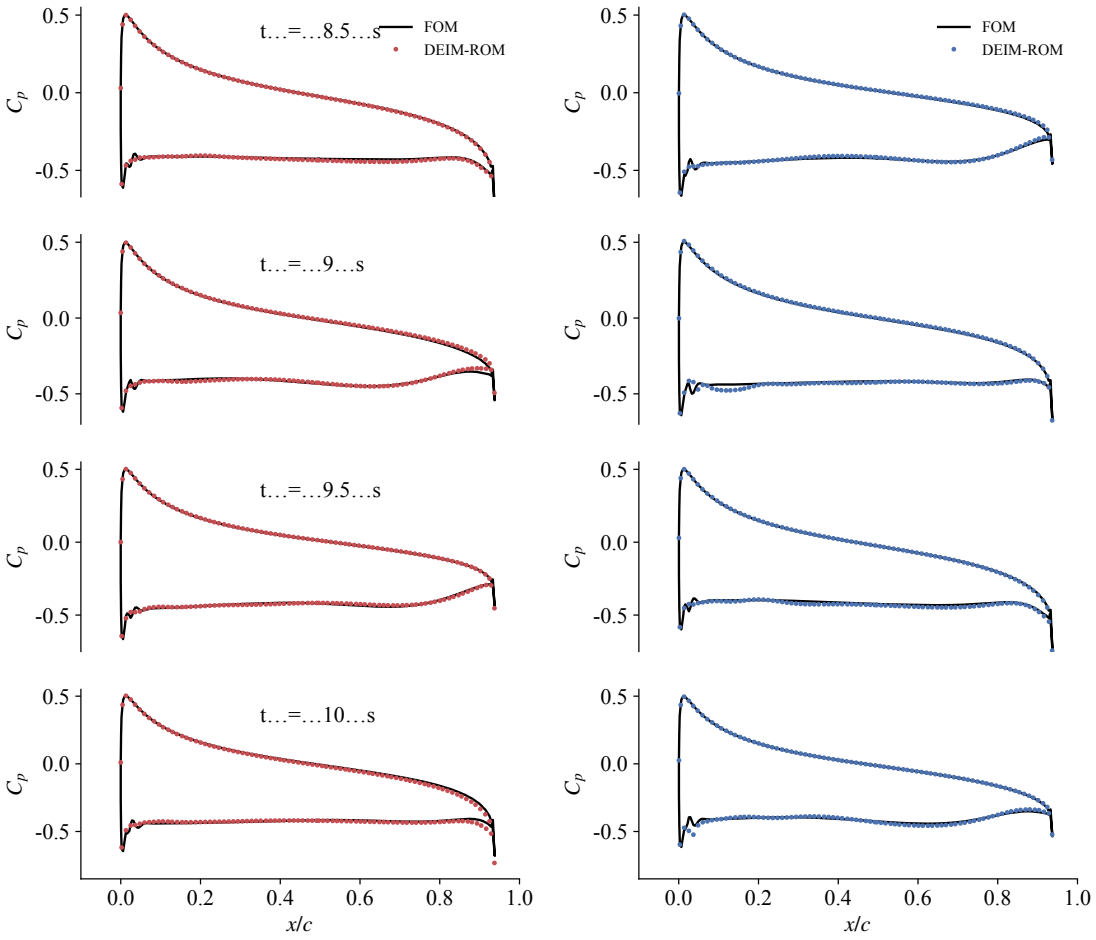

Figure 19: Comparison of pressure distribution on the airfoil surface (non-equilibrium). We use $\alpha = 20°$ as the initial condition to predict $\alpha = 20.3°$ (left) and $20.5°$ (right). We achieve good agreements between FOM and DEIM-ROM.

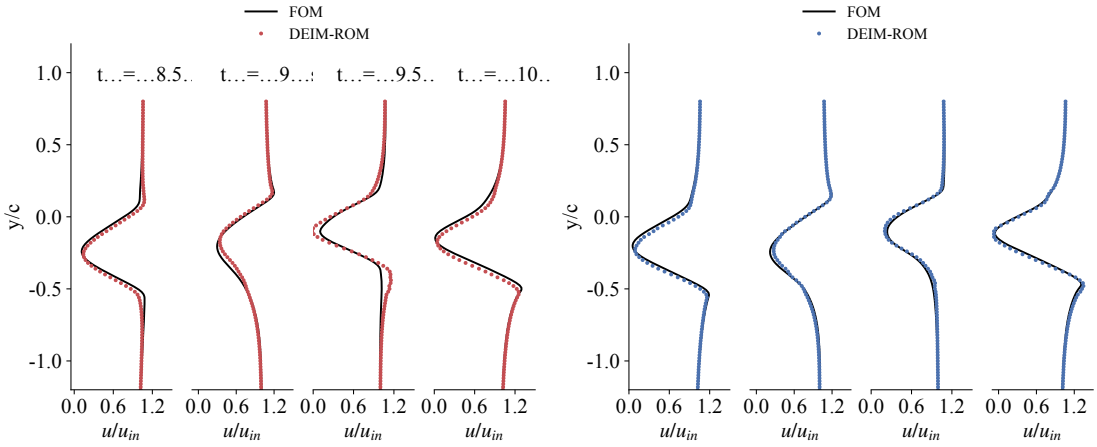

Figure 20: Comparison of velocity profile $0.5c$ downstream of the trailing edge (non-equilibrium). We use $\alpha = 20°$ as the initial condition to predict $\alpha = 20.3°$ (left) and $20.5°$ (right). We achieve good agreements between FOM and DEIM-ROM.

states. This new capability was primarily attributed to the intrusive nature of our ROM approach, which has the flexibility of predicting more irregular flow features.

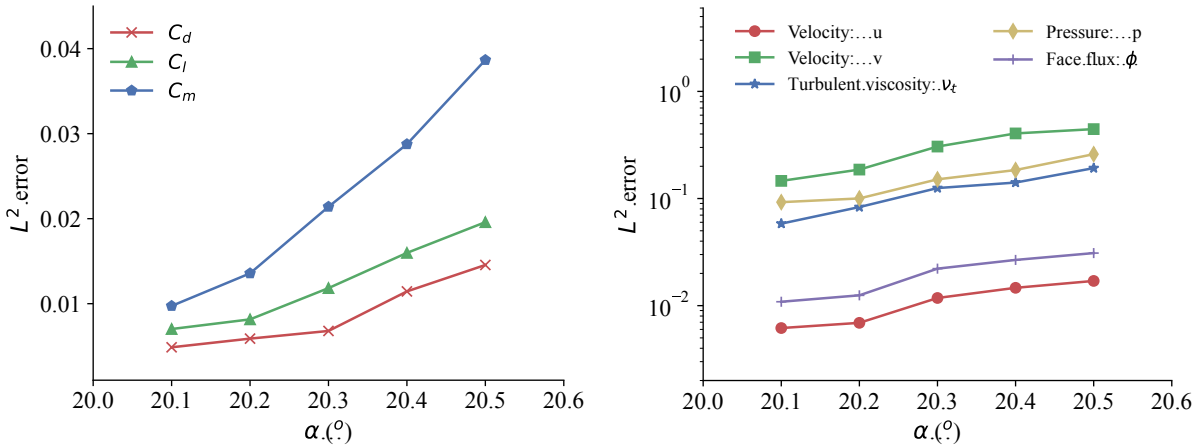

Figure 21: Normalized L^2 errors of C_d, C_l, and C_m time series (left) and whole flow field (right) for the angle of attack α predictive cases (non-equilibrium).

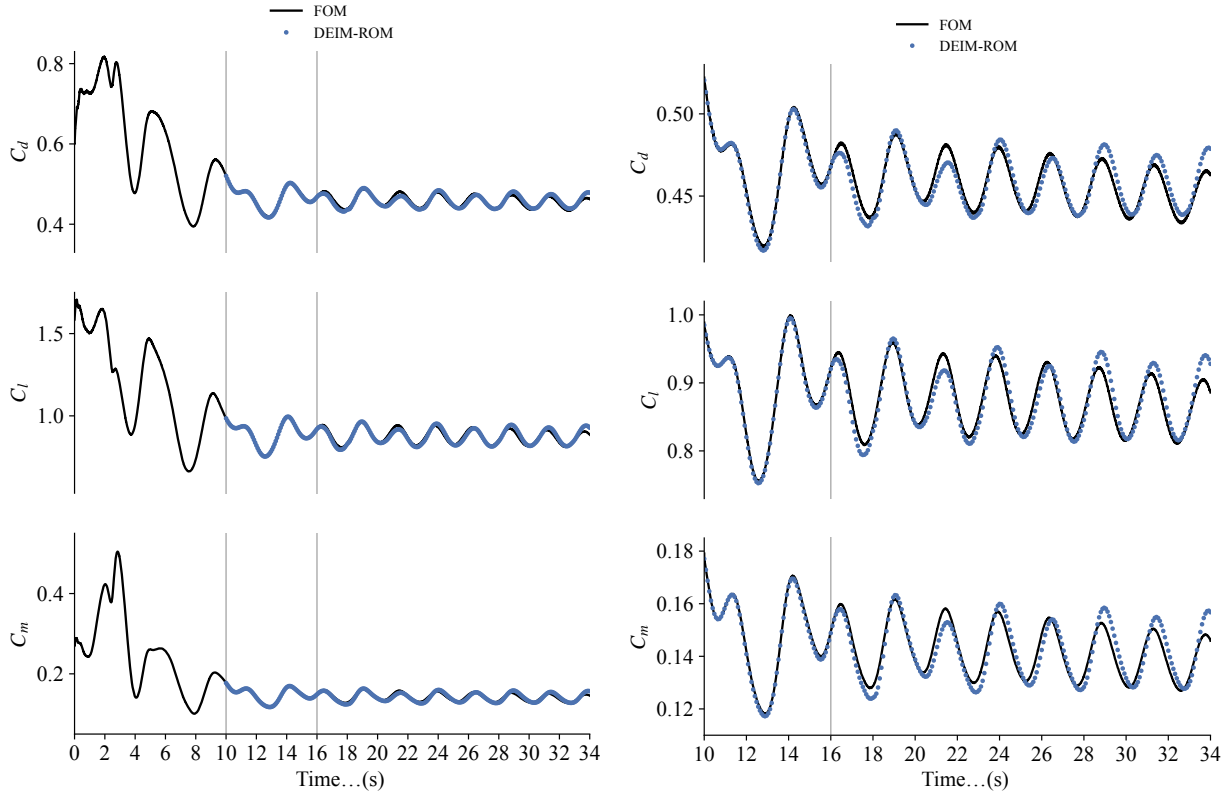

Figure 22: Comparisons of drag (top), lift (mid), and pitching moment (bottom) for the large angle of attack case: zoomed out (left) and zoomed in (right). We use $\alpha = 20°$ as the initial condition to predict $\alpha = 25°$. We use the FOM data from 10 s to 16 s for training, as indicated by the grey line. We achieve good agreements between FOM and DEIM-ROM.

Figure 23 shows the time series of velocity at probe point 2. Compared with the small perturbation cases in the previous subsection, we observe larger errors. This is expected because the velocity fluctuation is much more complicated, and the velocity in the prediction time window (16 s to 34 s) has not reached a full equilibrium state, especially the u velocity component.

Compared with temporal evolution, the agreement for spatial distribution between FOM and ROM is much better. As shown in the comparison of pressure distribution (Fig. 24) and wake structure (Fig. 25)

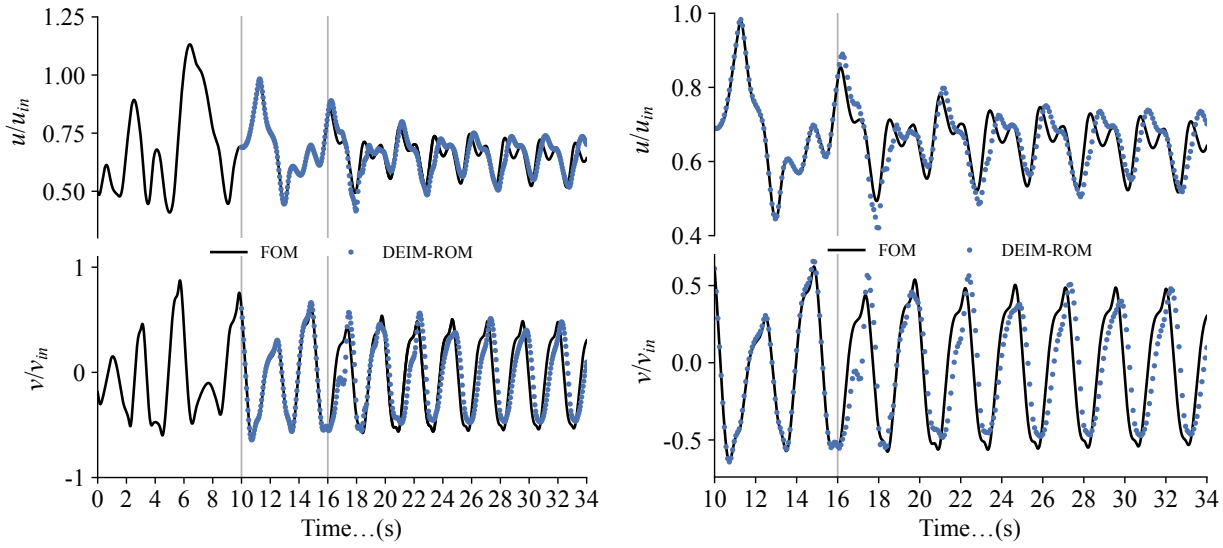

Figure 23: Same as Fig. 22 but for the velocity time-series at probe point 2.

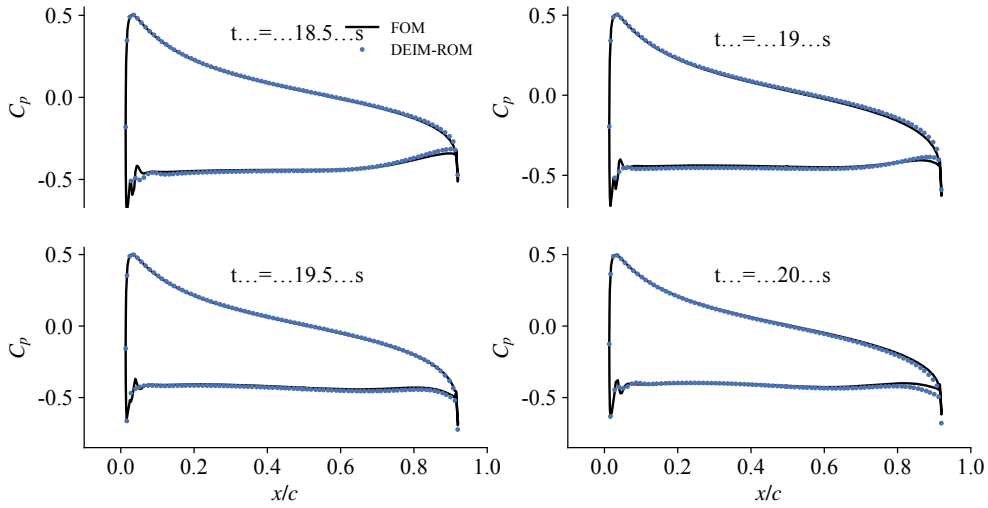

Figure 24: Comparison of pressure distribution on the airfoil surface (large angle of attack). We use $\alpha = 20°$ as the initial condition to predict $\alpha = 25°$. We achieve good agreements between FOM and DEIM-ROM.

plots. The agreement for the large variation case is similar to the small perturbation cases. This implies that the main challenge for large variation cases is capturing the temporal evolution.

4. Conclusion

In this paper, we develop a predictive reduced-order modeling (ROM) approach to accelerate unsteady aerodynamic simulations. We use the Galerkin projection approach to reduce the Reynolds-averaged Navier–Stokes (RANS) equations into a quadratic ordinary differential equation. We then incorporate the discrete empirical interpolation method (DEIM) into the above ROM equation to avoid computing the quadratic matrices (result from the RANS nonlinear terms) and reduce the computational cost. In addition, we develop an efficient ROM formulation that correlates the temporal coefficients between the momentum, pressure, and turbulence equations. This treatment allows us to solve fewer ROM equations at the prediction stage. Moreover, the ROM formulation can be applied to any turbulence model because we don't need to solve a

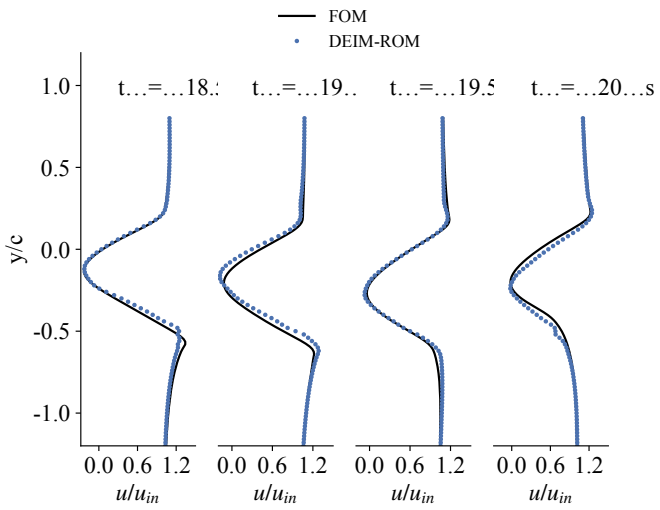

Figure 25: Comparison of velocity profile $0.5c$ downstream of the trailing edge (large angle of attack). We use $\alpha = 20°$ as the initial condition to predict $\alpha = 25°$. We achieve good agreements between FOM and DEIM-ROM.

reduced turbulence equation explicitly. Owing to the intrusive nature of the proposed ROM approach, we can use the first portion of unsteady flow data to accelerate the rest of the simulation.

The speed and accuracy of the proposed Galerkin ROM and DEIM-ROM have been tested using stall turbulent flow over the NACA0012 airfoil. We consider two unsteady flow scenarios: equilibrium and non-equilibrium. For the equilibrium cases, we achieve speed-up factors of 30.4 (ROM) and 44.5 (DEIM-ROM). Regarding accuracy, the velocity contours, wake structures, surface pressure distribution, and time series of drag, lift, pitching moment, and velocities computed by ROM agree reasonably well with the FOM references. The overall error grows slowly with time and remains at a relatively low level. The good agreement is mostly attributed to the training data being in an equilibrium state and containing all the possible spatial modes. For the non-equilibrium cases, the flow undergoes a transient process before reaching equilibrium for new flow conditions and is more challenging to predict. We need to use more data for training and more modes to better represent the flow; therefore, the speed-up factor reduces to about 13. ROM predicts the overall trend of the non-equilibrium turbulent flow reasonably well, although the agreement is not as well as in the equilibrium case. In addition, we find that ROM performs better in predicting spatial variations (e.g., pressure distribution and wake structure) than temporal variations (e.g., time-series data).

The proposed predictive ROM approach is, in principle, applicable to different airfoil geometries and flow conditions and can be integrated into a design optimization process for accelerating unsteady aerodynamic simulations. In the future, we will further improve the robustness and accuracy of our ROM formulation for long-term non-equilibrium flow simulations where the mean flow changes significantly. In addition, we will enable the ROM capability for predicting flow with different boundary conditions (e.g., far-field velocity magnitude and direction). With the above improvements implemented, we will parallelize our ROM solver and extend its capability for predicting three-dimensional flow problems. Moreover, we will extend our ROM implementation to more disciplines, e.g., aerostructural coupling.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number 2223676.

[1] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, Quarterly of applied mathematics 45 (1987) 561–571.

[2] T. D. Robinson, M. S. Eldred, K. E. Willcox, R. Haimes, Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping, AIAA Journal 46 (2008) 2814–2822. doi:10.2514/1.36043.

[3] L. Leifsson, S. Koziel, Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction, Journal of Computational Science 1 (2010) 98–106.

[4] J. Li, M. A. Bouhlel, J. R. Martins, Data-based approach for fast airfoil analysis and optimization, AIAA Journal 57 (2019) 581–596. doi:10.2514/1.J057129.

[5] X. Du, P. He, J. R. Martins, Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling, Aerospace Science and Technology 113 (2021). doi:10.1016/j.ast.2021.106701.

[6] D. J. Lucia, P. S. Beran, W. A. Silva, Reduced-order modeling: new approaches for computational physics, Progress in aerospace sciences 40 (2004) 51–117.

[7] W. H. a. Schilders, H. a. V. D. Vorst, J. Rommes, Model Order Reduction: Theory, Research Aspects and Applications, volume 13, Springer, 2008.

[8] T. Lassila, A. Manzoni, A. Quarteroni, G. Rozza, Model Order Reduction in Fluid Dynamics: Challenges and Perspectives, Reduced Order Methods for Modeling and Computational Reduction (2014) 235–273. doi:10.1007/978-3-319-02090-7_9.

[9] F. Chinesta, A. Huerta, G. Rozza, K. Willcox, Model order reduction, Encyclopedia of computational mechanics (2016).

[10] J. Yu, C. Yan, M. Guo, Non-intrusive reduced-order modeling for fluid problems: A brief review, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 233 (2019) 5896–5912.

[11] J. Kou, W. Zhang, Data-driven modeling for unsteady aerodynamics and aeroelasticity, Progress in Aerospace Sciences 125 (2021) 100725.

[12] K. Li, J. Kou, W. Zhang, Unsteady aerodynamic reduced-order modeling based on machine learning across multiple airfoils, Aerospace Science and Technology 119 (2021). doi:10.1016/j.ast.2021.107173.

[13] H. Liu, X. Gao, Z. Chen, F. Yang, Efficient reduced-order aerodynamic modeling in low-Reynolds-number incompressible flows, Aerospace Science and Technology 119 (2021) 107199. doi:10.1016/j.ast.2021.107199.

[14] W. C. Krolick, J. I. Shu, Y. Wang, K. Pant, State consistence of data-driven reduced order models for parametric aeroelastic analysis, SN Applied Sciences 3 (2021) 1–24. doi:10.1007/s42452-021-04252-w.

[15] K. Decker, N. Iyengar, D. Rajaram, C. Perron, D. Mavris, Manifold Alignment-Based Nonintrusive and Nonlinear Multifidelity Reduced-Order Modeling, AIAA Journal 61 (2022) 1–21. doi:10.2514/1.j061720.

[16] R. Halder, K. J. Fidkowski, K. J. Maki, Non-intrusive reduced-order modeling using convolutional autoencoders, International Journal for Numerical Methods in Engineering 123 (2022) 5369–5390. doi:10.1002/nme.7072.

[17] F. Saltari, M. Pizzoli, F. Gambioli, C. Jetzschmann, F. Mastroddi, Sloshing reduced-order model based on neural networks for aeroelastic analyses, Aerospace Science and Technology 127 (2022) 107708. doi:10.1016/j.ast.2022.107708.

[18] Z. Liu, R. Han, M. Zhang, Y. Zhang, H. Zhou, G. Wang, G. Chen, An enhanced hybrid deep neural network reduced-order model for transonic buffet flow prediction, Aerospace Science and Technology 126 (2022) 107636. doi:10.1016/j.ast.2022.107636.

[19] R. Geelen, S. Wright, K. Willcox, Operator inference for non-intrusive model reduction with nonlinear manifolds, Computer Methods in Applied Mechanics and Engineering 403 (2023) 115717. doi:10.1016/j.cma.2022.115717.

[20] M. G. Kapteyn, D. J. Knezevic, D. Huynh, M. Tran, K. E. Willcox, Data-driven physics-based digital twins via a library of component-based reduced-order models, International Journal for Numerical Methods in Engineering 123 (2022) 2986–3003.

[21] A. Bertram, C. Othmer, R. Zimmermann, Towards real-time vehicle aerodynamic design via multi-fidelity data-driven reduced order modeling, in: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2018, p. 0916.

[22] M. Mrosek, C. Othmer, R. Radespiel, Reduced-order modeling of vehicle aerodynamics via proper orthogonal decomposition, SAE International Journal of Passenger Cars-Mechanical Systems 12 (2019) 225–237.

[23] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Review 57 (2015) 483–531. doi:10.1137/130932715.

[24] A. E. Deane, I. G. Kevrekidis, G. E. Karniadakis, S. A. Orszag, Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders, Physics of Fluids A 3 (1991) 2337–2354. doi:10.1063/1.857881.

[25] H. M. Park, M. W. Lee, An efficient method of solving the Navier-Stokes equations for flow control, International Journal for Numerical Methods in Engineering 41 (1998) 1133–1151. doi:10.1002/(SICI)1097-0207(19980330)41:6<1133::AID-NME329>3.0.CO;2-Y.

[26] K. Y. Tang, W. R. Graham, J. Peraire, Active flow control using a reduced order model and optimum control, in: 1996 Fluid Dynamics Conference, 1996, pp. 1–12. doi:10.2514/6.1996-1946.

[27] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (2010) 2737–2764. doi:10.1137/090766498.

[28] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows, Journal of Computational Physics 242 (2013) 623–647. doi:10.1016/j.jcp.2013.02.028.

[29] K. Willcox, Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition, Computers and Fluids 35 (2006) 208–226. doi:10.1016/j.compfluid.2004.11.006.

[30] S. Lorenzi, A. Cammi, L. Luzzi, G. Rozza, POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations, Computer Methods in Applied Mechanics and Engineering 311 (2016) 151–179. doi:10.1016/j.cma.2016.08.006.

[31] S. Star, G. Stabile, G. Rozza, J. Degroote, A POD-Galerkin reduced order model of a turbulent convective buoyant flow of sodium over a backward-facing step, Applied Mathematical Modelling 89 (2021) 486–503. doi:10.1016/j.apm.2020.07.029.

[32] P. He, R. Halder, K. J. Fidkowski, K. J. Maki, J. R. Martins, An efficient nonlinear reduced-order modeling approach for rapid aerodynamic analysis with openfoam, in: AIAA Scitech 2021 Forum, 2021, pp. 1–17. doi:10.2514/6.2021-1476.

[33] C. Huang, C. R. Wentland, K. Duraisamy, C. Merkle, Model reduction for multi-scale transport problems using model-form preserving least-squares projections with variable transformation, Journal of Computational Physics 448 (2022). doi:10.1016/j.jcp.2021.110742.

[34] A. Garbo, P. Bekemeyer, Unsteady physics-based reduced order modeling for large-scale compressible aerodynamic applications, Computers and Fluids 239 (2022). doi:10.1016/j.compfluid.2022.105385.

[35] S. Hijazi, G. Stabile, A. Mola, G. Rozza, Data-driven POD-Galerkin reduced order model for turbulent flows, Journal of Computational Physics 416 (2020) 109513. doi:10.1016/j.jcp.2020.109513.

[36] S. A. McQuarrie, C. Huang, K. E. Willcox, Data-driven reduced-order models via regularised operator inference for a single-injector combustion process, Journal of the Royal Society of New Zealand 51 (2021) 194–211.

[37] M. Zancanaro, M. Mrosek, G. Stabile, C. Othmer, G. Rozza, Hybrid neural network reduced order modelling for turbulent flows with geometric parameters, Fluids 6 (2021) 296. doi:10.3390/fluids6080296.

[38] K. J. Fidkowski, R. Halder, K. J. Maki, Model Reduction Using Interpolated Systems of Equations, in: AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022, 2022, p. 2323. doi:10.2514/6.2022-2323.

[39] J. Kou, S. Le Clainche, W. Zhang, A reduced-order model for compressible flows with buffeting condition using higher order dynamic mode decomposition with a mode selection criterion, Physics of Fluids 30 (2018) 016103.

[40] J. Kou, W. Zhang, Dynamic mode decomposition with exogenous input for data-driven modeling of unsteady flows, Physics of fluids 31 (2019) 057106.

[41] Z. Li, P. He, Airfoil Unsteady Aerodynamic Analysis Using a Galerkin Reduced-order Modeling Approach, in: AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022, 2022. doi:10.2514/6.2022-0080.

[42] H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Computers in Physics 12 (1998) 620. doi:10.1063/1.168744.

[43] P. Spalart, S. Allmaras, A One-Equation Turbulence Model for Aerodynamic Flows, in: 30th Aerospace Sciences Meeting and Exhibit, 1992. doi:10.2514/6.1992-439.

[44] R. I. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, Journal of Computational Physics 62 (1985) 40–65. doi:10.1016/0021-9991(86)90099-9.

[45] C. M. Rhie, W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA Journal 21 (1983) 1525–1532. doi:10.2514/3.8284.

[46] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, G. Rozza, POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: Vortex shedding around a circular cylinder, Communications in Applied and Industrial Mathematics 8 (2017) 210–236. doi:10.1515/caim-2017-0011.

[47] V. Hernandez, J. E. Roman, V. Vidal, SLEPC: A scalable and flexible toolkit for the solution of eigenvalue problems, ACM Transactions on Mathematical Software 31 (2005) 351–362. doi:10.1145/1089014.1089019.

[48] A. Iollo, S. Lanteri, J. A. Désidéri, Stability properties of POD-Galerkin approximations for the compressible Navier-Stokes equations, Theoretical and Computational Fluid Dynamics 13 (2000) 377–396. doi:10.1007/s001620050119.

[49] S. Sirisup, G. E. Karniadakis, Stability and accuracy of periodic flow solutions obtained by a POD-penalty method, Physica D: Nonlinear Phenomena 202 (2005) 218–237. doi:10.1016/j.physd.2005.02.006.

[50] I. Akhtar, A. H. Nayfeh, C. J. Ribbens, On the stability and extension of reduced-order Galerkin models in incompressible flows, Theoretical and Computational Fluid Dynamics 23 (2009) 213–237. doi:10.1007/s00162-009-0112-y.

[51] M. Bergmann, C. H. Bruneau, A. Iollo, Enablers for robust POD models, Journal of Computational Physics 228 (2009) 516–538. doi:10.1016/j.jcp.2008.09.024.

[52] F. Ballarin, A. Manzoni, A. Quarteroni, G. Rozza, Supremizer stabilization of POD-Galerkin approximation of parametrized steady incompressible Navier-Stokes equations, International Journal for Numerical Methods in Engineering 102 (2015) 1136–1161. doi:10.1002/nme.4772.

[53] K. Carlberg, M. Barone, H. Antil, Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction, Journal of Computational Physics 330 (2017) 693–734. doi:10.1016/j.jcp.2016.10.033.

[54] C. Huang, K. Duraisamy, C. L. Merkle, Investigations and improvement of robustness of reduced-order models of reacting flow, AIAA Journal 57 (2019) 5377–5389. doi:10.2514/1.J058392.

[55] G. Stabile, G. Rozza, Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations, Computers and Fluids 173 (2018) 273–284. doi:10.1016/j.compfluid.2018.01.035.

[56] D. Xiao, F. Fang, A. G. Buchan, C. C. Pain, I. M. Navon, J. Du, G. Hu, Non-linear model reduction for the Navier-Stokes equations using residual DEIM method, Journal of Computational Physics 263 (2014). doi:10.1016/j.jcp.2014.01.011.

[57] G. K. Kenway, G. J. Kennedy, J. R. Martins, A CAD-free approach to high-fidelity aerostructural optimization, in: 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2010, Fort Worth,∼TX, 2010. doi:10.2514/6.2010-9231.

[58] N. R. Secco, G. K. Kenway, P. He, C. Mader, J. R. Martins, Efficient mesh generation and deformation for aerodynamic shape optimization, AIAA Journal 59 (2021) 1151–1168. doi:10.2514/1.J059491.