AIAA SciTech Forum
11–15 &amp; 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-1476

An Efficient Nonlinear Reduced-order Modeling Approach for Rapid Aerodynamic Analysis with OpenFOAM

Ping He[†][*], Rakesh Halder[†], Krzysztof J. Fidkowski[†], Kevin J. Maki[†], and Joaquim R. R. A. Martins[†]

† *University of Michigan, Ann Arbor, MI, USA 48109*
∗ *Iowa State University, Ames, IA, USA 50010*

Computational fluid dynamics (CFD) based aerodynamic design has received increasing interest because simulated high-fidelity flow fields provide useful data to guide design optimization. Aerodynamic design often requires evaluating the performance of a wide range of possible shapes. As the complexity of a design increases, the size of the CFD mesh increases drastically, and running a large number of CFD simulations can be time-consuming. To alleviate this issue, we develop an efficient projection-based reduced-order modeling (ROM) approach to speed up the aerodynamic analysis. The benefit of using a projection-based ROM approach is that we can obtain integral variables such as lift and drag, as well as three-dimensional flow fields such as pressure and velocity, compared with interpolation-based methods such as kriging or neural networks. In this paper, we elaborate on the nonlinear ROM formulation we develop and its detailed implementation in OpenFOAM; a popular open-source code for multiphysics analysis. We use the NACA0012 airfoil and the Cessna 172 wing as the benchmarks and evaluate the performance of the proposed ROM method in terms of accuracy, speed, and memory cost. We observe significant estimated speed ups from the ROM simulations, and the velocity and pressure fields simulated by the ROM approach agree well with the CFD references. The proposed ROM approach has the potential to become a useful tool for rapid aerodynamic analysis.

I. Introduction

Advances in computing power have enabled computational fluid dynamics (CFD) to become an essential tool in aerodynamic design. CFD-based aerodynamic design provides high-fidelity flow fields containing useful data to guide the design optimization process. Therefore, CFD-based aerodynamic design can significantly reduce the design period and cost, compared with traditional experiment-based design. Because of the above salient features, CFD-based design has received increasing interest in recent years.

The application of CFD-based design ranges from simple geometries such as airfoils to complex configurations like aircraft with full geometric details. As the details requiring resolution increase, the number of CFD mesh cells increases drastically and can sometimes reach over 100 million. This poses a challenge for aerodynamic design because designers often need to evaluate a wide range of designs in a short period of time. Running hundreds of CFD simulations with large meshes is computationally expensive and is the bottleneck, especially for industrial applications where large meshes are commonly needed.

To alleviate this issue, one can use the reduced-order modeling (ROM) technique to decrease the high computational cost. One commonly used approach is using an interpolation-based ROMs which builds a surrogate model (e.g., kriging, neural network) to approximate a desired response in the design space [1, 2, 3, 4, 5, 6]. An interpolation-based ROM treats the numerical model as a black box and directly maps the inputs to the outputs; therefore, it effectively reduces the computational cost when evaluating functions for a new design. However, one drawback of this approach is that only limited data, typically integral values such as lift and drag, can be evaluated, and the flow physics are completely lost in the mapping process. The interpolation-based approach is generally sufficient for design optimization, where designers mainly focus on reducing the value of an objective function. However, it is not ideal for aerodynamic analysis because designers often need three-dimensional flow fields (e.g., velocity and pressure) to interpret the behavior of a new design.

Alternatively, one can use a projection-based ROM approach [7, 8, 9, 10, 11, 12] to rapidly evaluate the aerodynamic performance. A projection-based ROM uses the proper orthogonal decomposition approach to project the full-order model into a lower-dimensional space. This allows for solving reduced equations on a much lower dimension and projecting solutions back to full-order. The main flow physics are preserved during the projection, and thus a projection-based ROM can approximate three-dimensional, full-order flow fields with a significantly reduced runtime, compared with simulating the full-order model.

The objective of this paper is to develop an efficient projection-based ROM approach to enable rapid aerodynamic analysis. The focus of this work is to evaluate the performance of the proposed approach for practical engineering design problems. To be more specific, we elaborate on our nonlinear ROM formulations and the implementation of the ROM method in a popular open-source code for multiphysics analysis: OpenFOAM [13, 14]. We use the NACA0012 airfoil and the Cessna 172 wing case as the benchmarks. To demonstrate the power of the proposed ROM method, we evaluate its performance in terms of accuracy, speed, and memory cost. We also compare the three-dimensional flow fields generated by the high-fidelity CFD and ROM.

The rest of the paper is organized as follows. In Section II, we introduce the ROM framework and detail its implementations. The CFD and ROM results are presented and discussed in Section III and we summarize our findings in Section IV.

II. Method

As mentioned earlier, the nonlinear projection-based ROM we propose is implemented in OpenFOAM. In this section, we elaborate on the detailed solutions of full-order models, followed by the implementation of our nonlinear ROM approach.

1. Full-order flow solution

We use the standard OpenFOAM solver simpleFoam to simulate steady incompressible turbulent flow by solving the Navier-Stokes equations:

$$\int_S \vec{U} \cdot \mathrm{d}\vec{S} = 0, \tag{1.1}$$

$$\int_S \vec{U}\vec{U} \cdot \mathrm{d}\vec{S} + \int_V \nabla p \,\mathrm{d}V - \int_S (\nu + \nu_t)(\nabla \vec{U} + \nabla \vec{U}^T) \cdot \mathrm{d}\vec{S} = 0, \tag{1.2}$$

where $\vec{U}$ = $[u,\ v,\ w]$ is the velocity vector; u, v, and w are the velocity components in the x, y, and z directions, respectively; $\vec{S}$ is the face-area vector; V is the volume; ν and ν_t are the molecular and turbulent eddy viscosity respectively, and p is the pressure. The finite volume method is used to discretize the continuity and momentum equations on collocated meshes. These two equations are coupled by using the semi-implicit method for pressure-linked equations (SIMPLE) algorithm [15] along with the Rhie–Chow interpolation [16].

The SIMPLE algorithm starts by discretizing the momentum equation and solving an intermediate velocity field by using the pressure field obtained from the previous iteration or an initial guess (p^0). The momentum equation is then semi-discretized as

$$a_P \vec{U}_P = -\sum_N a_N \vec{U}_N - \nabla p^0 = \vec{H}(\vec{U}) - \nabla p^0, \tag{1.3}$$

where a is the coefficient resulting from the finite-volume discretization, subscripts P and N denote the control volume cell and all of its neighboring cells, respectively, and $\vec{H}(\vec{U}) = -\sum_N a_N \vec{U}_N$ represents the influence of the velocity from all the neighboring cells. Note that to linearize the convective term, a new variable ϕ^0 (the cell-face flux) is introduced into the discretization to give

$$\int_S \vec{U}\vec{U} \cdot \mathrm{d}\vec{S} = \sum_f \vec{U}_f \vec{U}_f \cdot \vec{S}_f = \sum_f \phi^0 \vec{U}_f, \tag{1.4}$$

where the subscript f denotes the cell face. The cell-face flux ϕ^0 can be obtained from the previous iteration or from an initial guess. Solving Eq. (1.3), 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

$$\int_S \vec{U} \cdot \mathrm{d}\vec{S} = \sum_f \vec{U}_f \cdot \vec{S}_f = 0. \tag{1.5}$$

Instead of using a simple linear interpolation, $\vec{U}_f$ in this equation is computed by interpolating the cell-centered velocity $\vec{U}_P$—obtained from the discretized momentum equation (1.3)—onto the cell face as follows:

$$\vec{U}_f = \left(\frac{\vec{H}(\vec{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f. \tag{1.6}$$

This idea of momentum interpolation was initially proposed by Rhie and Chow [16] and is effective in mitigating the "checkerboard" issue resulting from the collocated mesh configuration. Substituting Eq. (1.6) into Eq. (1.5), we obtain the pressure Poisson equation:

$$\nabla \cdot \left(\frac{1}{a_P} \nabla p \right) = \nabla \cdot \left(\frac{\vec{H}(\vec{U})}{a_P} \right). \tag{1.7}$$

Solving Eq. (1.7), we obtain an updated pressure field p^1.

Finally, the new pressure field p^1 is used to update the cell-face flux by using

$$\phi^1 = \vec{U}_f \cdot \vec{S}_f = \left[\left(\frac{\vec{H}(\vec{U})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p^1)_f \right] \cdot \vec{S}_f. \tag{1.8}$$

Next, a new velocity field is computed by using Eq. (1.3) with the updated pressure and cell-face flux. The process above is repeated until the specified flow-convergence criteria are met.

The Reynolds-averaged Navier–Stokes (RANS) approach is used to model the turbulence in the flow. To connect the mean variables with the turbulence eddy viscosity ν_t, we use the Spalart–Allmaras (SA) one-equation turbulence model:

$$\int_V \nabla \cdot (\vec{U}\tilde{\nu})\,\mathrm{d}V - \frac{1}{\sigma} \int_V \nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + C_{b2}|\nabla\tilde{\nu}|^2\,\mathrm{d}V - C_{b1} \int_V \tilde{S}\tilde{\nu}\,\mathrm{d}V + C_{w1} \int_V f_w \left(\frac{\tilde{\nu}}{d} \right)^2 \mathrm{d}V = 0, \tag{1.9}$$

where $\tilde{\nu}$ is the modified viscosity, which can be related to the turbulent eddy viscosity via

$$\nu_t = \tilde{\nu} \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}. \tag{1.10}$$

The four terms in Eq. (1.9) represent the convective, diffusion, production, and near-wall destruction for the turbulent flow, respectively. The detailed definition of these terms and their parameters can be seen in Spalart and Allmaras [17]. Compared with the standard SA model [17], the f_{t2} term is ignored in the OpenFOAM SA model implementation. Moreover, a stability enhancement function f_{v3} is added to ensure a non-negative $\tilde{S}$ term.

2. Reduced-order modeling

The discrete version of the partial differential equations (PDEs) in Eqs. 1.3, 1.7, 1.8, and 1.10 can be written in residual form as:

$$\vec{R}(\vec{w}, \vec{x}_v) = 0, \tag{2.1}$$

where $\vec{R}$ is the residual vector, $\vec{w}$ is the vector of state variables, i.e., velocity, pressure, face flux, and turbulence variables, and $\vec{x}_v$ is the volume mesh coordinates. $\vec{x}_v$ is typically a function of the design variable vector ($\vec{x}$), i.e., $\vec{x}_v = g(\vec{x})$. To determine the mapping function $g(\vec{x})$, we use the free-form deformation method for geometry parameterization and an inverse-distance weighted method for volume mesh deformation through the pyGeo [18] and IDWarp [19] packages. In other words, for a new set of design variables $\vec{x}$, we can compute the corresponding volume mesh coordinate vector $\vec{x}_v$. Under this treatment, the residual equation can be written as a function of $\vec{x}$ and $\vec{w}$:

$$\vec{R}(\vec{w}, g(\vec{x})) = 0. \tag{2.2}$$

Next, we elaborate on how to reduce the cost to solve the full-order residual equations. The ROM approach we propose consists of two stages: offline and online [10, 11].

In the offline stage, we first run CFD samples with a range of design variable ($\vec{x}$) values. Then, we assemble all of these CFD samples' state variables ($\vec{w}$) into a snapshot matrix $S \in \mathbb{R}^{N \times m}$, where N is the number of full-order states, and m is the number of CFD samples, and $N \gg m$.

$$S^{N \times m} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & \cdots & w_{N,m} \end{bmatrix}. \tag{2.3}$$

Based on the snapshot matrix $S \in \mathbb{R}^{N\times m}$, we compute a set of basis vectors $\Phi \in \mathbb{R}^{N\times n} = [\Phi_1, \Phi_2, \cdots, \Phi_n]$ by taking the singular value decomposition (SVD) of S, where $\Phi_i \in \mathbb{R}^{N\times 1}$ is a basis vector and n is the number of basis vectors to keep ($n \le m$). The SVD is computed using the scalable library for eigenvalue problem computations (SLEPc) library [20]. Once the SVD is complete, we save the Φ matrix to the disk in the portable, extensible toolkit for scientific computation (PETSc) [21] format so that it can be reused in the online stage later.

$$\Phi^{N\times n} = \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,n} \end{bmatrix}. \tag{2.4}$$

For small problems where the number of CFD samples is less than 10, we keep all the basis vectors by setting $n = m$. For large problems ($m > 10$) and small design variable ranges, it is typically not necessary to keep all of the basis vectors because this will significantly increase the computational cost while adding flow modes that are not significant in producing accurate solutions. Particularly, we drop basis vectors whose corresponding singular values are less than 0.1% of the largest singular values in the S matrix.

With this treatment, a full-order state-variable vector $\vec{w}$ in the design space can be approximated by a linear combination of the basis vectors:

$$\vec{w} \approx \Phi\vec{w}_r = [w_{r,1}\Phi_1, w_{r,2}\Phi_2, \cdots, w_{r,n}\Phi_n]. \tag{2.5}$$

where the reduced state variable vector $\vec{w}_r$ contains the weights.

Substituting the equation above to the full-order residual equation (2.1), we obtain:

$$\vec{R}(\vec{w}, g(\vec{x})) \approx \vec{R}(\Phi\vec{w}_r, g(\vec{x})), \tag{2.6}$$

Therefore, for a new design, we need to find the reduced-order state variable vector $\vec{w}_r$ that satisfies Eq. 2.6. Then we can use it to quickly evaluate the corresponding full-order states $\vec{w}$ using Eq. 2.5.

Although this projected system operates on a much lower dimension than the original full-order system, evaluating the full-order residual $\vec{R}$ still requires $\mathcal{O}(N)$ numerical operations and is expensive. To mitigate this high cost, we can make a further approximation to the residual vector, so that the number of required operations drops to $\mathcal{O}(k)$, where $n < k \ll N$:

$$\vec{R} \approx \Psi\vec{R}_r = [R_{r,1}\Psi_1, R_{r,2}\Psi_2, \cdots, R_{r,k}\Psi_k]. \tag{2.7}$$

This is similar to the approximation of the state in Eq. 2.5, where $\Psi \in \mathbb{R}^{N\times k}$ is a matrix containing basis vectors that approximate the full-order residual $\vec{R}$. We compute Ψ by first constructing a snapshot matrix $P \in \mathbb{R}^{N\times l}$ where $l = m \cdot q$ with q being the number of evenly spaced out residual snapshots taken per CFD sample on the way to convergence:

$$P^{N\times l} = \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,l} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ R_{N,1} & R_{N,2} & \cdots & R_{N,l} \end{bmatrix}. \tag{2.8}$$

We are interested in $k \le l$ interpolation points, and thus the partial SVD of P is computed to obtain Ψ:

$$\Psi^{N\times k} = \begin{bmatrix} \psi_{1,1} & \psi_{1,2} & \cdots & \psi_{1,k} \\ \psi_{2,1} & \psi_{2,2} & \cdots & \psi_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{N,1} & \psi_{N,2} & \cdots & \psi_{N,k} \end{bmatrix}. \tag{2.9}$$

Using the approximation of the residual in Eq. 2.7, we can use *hyper-reduction* (sparse sampling) to reduce the computational cost of evaluating $\vec{R}$ to $\mathcal{O}(k)$ numerical operations. Specifically, we define a sparse mask matrix $Z \in \mathbb{R}^{k\times N}$, in which there is a single one in each row at a chosen interpolation point. These interpolation points are found by applying the discrete empirical interpolation method (DEIM) [22], which provides an optimal set of interpolation points to our residual basis Ψ. Let $\hat{R} \in \mathbb{R}^k$ be the residual evaluated at only the k interpolation points. Using $\hat{R}$, we want to interpolate to approximate the full residual. Multiplying both sides of Eq. 2.7 by Z gives us the following:

$$Z\vec{R} = \hat{R} \approx Z\Psi\vec{R}_r. \tag{2.10}$$

Solving this equation for $\vec{R}_r$ and substituting back into Eq. 2.7, we obtain:

$$\vec{R} \approx \underbrace{\Psi(Z\Psi)^{-1}}_{C}\hat{R}. \tag{2.11}$$

In the offline stage, we can compute $C \in \mathbb{R}^{N\times k}$ and save it to the disk. In the online stage, we can first compute residuals at selected interpolation points for $\hat{R}$, and then we use Eq. 2.11 to approximate the full-order residuals. The hyper-reduction formulation is expected to significantly decrease the computational cost for the full-order residual because we need to evaluate residuals only for k interpolation points instead of at all the cells in the mesh.

Evaluating the residual at only the k interpolation points in CFD can be done by creating submeshes that contain the desired cell and its neighboring cells such that the residual can be computed correctly. However, our current implementation does not have this capability, and the full-order residual $\vec{R}$ is still calculated explicitly. $\hat{R}$ is computed by applying Z to $\vec{R}$, and thus Z is also saved to the disk during the offline stage. The full potential speed up from the hyper-reduction ROM will be achieved in our future work. In this paper, we provide estimated speed up times in the results section based on the values of N and k for the given problem. We also include results computed without using hyper-reduction (brute-force ROM). In the brute-force ROM, we approximate only the state on a lower dimension (Eq. 2.5) and not the residual.

In the online stage, we use the least-squares Petrov–Galerkin (LSPG) formulation to solve for $\vec{w}_r$. Specifically, for a new design variable vector $\vec{x}^*$, we solve for the reduced vector $\vec{w}_r$ that minimizes the L_2 norm of the approximated full-scale residuals (Eq. 2.6):

$$\text{Minimize } L_2 = ||\vec{R}(\Phi\vec{w}_r, g(\vec{x}^*))||^2, \text{ with respect to } \vec{w}_r. \tag{2.12}$$

At the minimal point, we have $\partial L_2/\partial\vec{w}_r = 0$, which gives the reduced residual equation:

$$\left[\underbrace{\frac{\partial\vec{R}}{\partial\vec{w}}}_{N\times N}\underbrace{\Phi}_{N\times n}\right]^T \underbrace{\vec{R}}_{N\times 1} = \underbrace{\Omega}_{n\times N}\underbrace{\vec{R}}_{N\times 1} = \underbrace{\vec{R}_r}_{n\times 1} = 0. \tag{2.13}$$

Algorithm 1 Newton-Krylov algorithm to solve the reduced residual equation

```
Input: $\vec{x}^*$, $\vec{w}_{\text{ref}}$                                      ▷ New design variables and initial full-order states
Output: $\vec{w}^*$                                                  ▷ New full-order state variables for the new design
1: $\vec{x}_v^* = g(\vec{x}^*)$                                      ▷ Compute volume mesh coordinates based on new design variables
2: $\vec{w}_r^0 = \Phi^T\vec{w}_{\text{ref}}$                        ▷ Compute the initial reduced state variable vector
3: for $n \in \{0, 1, \dots, n_{\max}\}$ do                          ▷ Main loop to compute $\vec{w}_r^*$
4:     $\frac{\partial\vec{R}_r}{\partial\vec{w}_r} \leftarrow \vec{w}_r^n, \vec{x}_v^*$      ▷ Compute the reduced Jacobian matrix
5:     $\vec{R}_r \leftarrow \vec{w}_r^n, \vec{x}_v^*$               ▷ Compute the reduced residual
6:     $\Delta\vec{w}_r^n \leftarrow \frac{\partial\vec{R}_r}{\partial\vec{w}_r}_{(\vec{w}_r^n, \vec{x}_v^*)}\Delta\vec{w}_r^n = -\vec{R}_{r(\vec{w}_r^n, \vec{x}_v^*)}$      ▷ Solve the linear equation to get $\Delta\vec{w}_r^n$
7:     for $m \in \{0, 1, \dots, m_{\max}\}$ do                      ▷ Backtracking line search
8:         $\alpha = 1.0$
9:         $\vec{w}_r^{n+1} = \vec{w}_r^n + \alpha\Delta\vec{w}_r^n$  ▷ Update $\vec{w}_r^n$ with step $\alpha$
10:        if $||\vec{R}_r(\vec{w}_r^{n+1})||^2 < ||\vec{R}_r(\vec{w}_r^n)||^2$ then      ▷ Reduced residual decreases, line search done
11:            break
12:        else                                                      ▷ Reduced residual not dropping, decrease $\alpha$
13:            $\alpha = 0.5\alpha$
14:    $\vec{w}_r^* = \vec{w}_r^{n+1}$                               ▷ Assign the latest solution to $\vec{w}_r^*$
15:    if $||\vec{R}_r(\vec{w}_r^{n+1})||^2/||\vec{R}_r(\vec{w}_r^0)||^2 < r_{\text{tol}}$ then      ▷ Prescribed tolerance satisfied, exit the main loop
16:        break
17: $\vec{w}^* = \Phi\vec{w}_r^*$                                    ▷ Compute the new full-order state variable vector
```

The reduced nonlinear residual vector $\vec{R}_r$ has size of $n \times 1$ and its input variables are $\vec{w}_r \in \mathbb{R}^{n\times 1}$ and $\vec{x}^* \in \mathbb{R}^{n_x \times 1}$. To solve this reduced residual equation $\vec{R}_r(\Phi\vec{w}_r, g(\vec{x}^*)) = 0$, we use the Newton-Krylov approach and its details are summarized in Algorithm 1.

To be more specific, we first compute the new volume mesh coordinates ($\vec{x}_v^*$) based on the new design variable vector ($\vec{x}^*$). Once $\vec{x}_v^*$ is computed, we need to update the CFD mesh metrics (e.g., cell center, surface area, and volume) based on the new volume mesh coordinates. We also need to use a reference full-order state variable vector $\vec{w}_{\text{ref}}$ (its choice will be discussed later) to compute the initial reduced state variable vector ($\vec{w}_r^0$), such that we can use it as the initial condition for the Newton-Krylov solution. In the Newton-Krylov iteration loop, we first compute the reduced residual vector $\vec{R}_r$, as well as the reduced Jacobian matrix $\partial\vec{R}_r/\partial\vec{w}_r$ using the finite-difference method. Next, we solve the linear equation to obtain $\Delta\vec{w}_r$. The linear equation is solved using the generalized minimal residual (GMRES) method implemented in the PETSc library [21]. Then, we perform a backtracking line search to compute the new reduced state variable vector $\vec{w}_r^{n+1}$. The line search requires the reduced residual vector $\vec{R}_r$ to decrease for the new $\vec{w}_r^{n+1}$ vector. If not, we reduce the step size by a factor of two. Typically, we perform maximal five line-search attempts (i.e., $m_{\max} = 5$). This process is repeated until the relative reduced-residual $\vec{R}_r$ is less than the prescribed tolerance (r_{tol}) or the Newton-Krylov reaches $n_{\max}$ iterations. Although we only drive the reduced residual to zero during the Newton iterations, the full-order residual is expected to decrease too, according to the LSPG formulation (2.13). The converged reduced state variable vector is output as $\vec{w}_r^*$. Finally, we compute the new, full-order state variable using $\vec{w}^* = \Phi\vec{w}_r^*$.

As mentioned before, for each Newton iteration, we explicitly compute and form the reduced state Jacobian matrix $\partial\vec{R}_r/\partial\vec{w}_r$ using the finite-difference method. When evaluating the reduced residual $\vec{R}_r$, we need to compute $[\partial\vec{R}/\partial\vec{w}]\Phi$. We compute this matrix-matrix product in a matrix-free manner. To this end, we first extract one basis vector $\vec{\Phi}_i$ from the Φ matrix and use the matrix-free approach to compute the matrix-vector product:

$$\frac{\partial\vec{R}}{\partial\vec{w}}\vec{\Phi}_i = \frac{\vec{R}(\vec{w}+\varepsilon\vec{\Phi}_i) - \vec{R}(\vec{w})}{\varepsilon}, \tag{2.14}$$

where $\varepsilon = 1 \times 10^{-6}$ is a small step size. In the above equation, we need to evaluate the full-order residual once for each column in the Φ matrix. The full-order residual computation is based on DAFoam [23, 24], an open-source adjoint derivative computation framework with OpenFOAM. We repeat the above step for all the basis vectors in the Φ matrix. This treatment prevents us from explicitly forming and storing the full-order state Jacobian matrix $\partial\vec{R}/\partial\vec{w}$, which saves memory and improves speed.

The Newton-Krylov method needs a preconditioner (PC) matrix $[\partial\vec{R}/\partial\vec{w}]_{PC}$ for each GMRES linear equation solution. We use the additive Schwartz method (ASM) as the global preconditioner, while for the local preconditioner, we use the incomplete lower and upper (ILU) factorization approach, as described in our previous work [25]. This PC matrix is computed in the offline stage using the state variables from a reference sample point that is closest to the baseline design. This PC matrix is used for all predictions and will not be updated in the online stage. This is reasonable because the size of reduced nonlinear equation $\vec{R}_r$ is small, and we do not observe any difficulty converging the linear equation; there is no need to update the PC matrix for every Newton iteration.

Once we obtain the converged reduced state variable vector $\vec{w}_r^*$ for a new design $\vec{x}^*$, the corresponding full-order state variable vector is computed as $\vec{w}^* = \Phi\vec{w}_r^*$.

To summarize, the nonlinear ROM consists of the following steps:

- Offline:
  - Generate CFD samples with a range of $\vec{x}$ values (different shapes)
  - Perform a SVD of the state snapshots to compute Φ and save it to the disk
  - If using hyper-reduction, perform a SVD of the residual snapshots to compute Ψ, execute the DEIM algorithm, and save C and Z to the disk
  - Compute the preconditioner matrix $[\partial\vec{R}_r/\partial\vec{w}_r]_{\text{PC}}$ and save it to the disk

- Online:
  - Calculate $\vec{R}$ in all online calculations using either hyper-reduction or brute-force approach
  - For a new design with $\Delta\vec{x}$, deform the geometry to a new design point ($\vec{x}^* = \vec{x} + \Delta\vec{x}$), compute the new volume mesh coordinates, i.e., $\vec{x}_v^* = g(\vec{x}^*)$, using the pyGeo [18] and IDWarp [19] packages.
  - Use the reference full-order state variable vector to compute the initial reduced state variable vector ($\vec{w}_r^0 = \Phi^T\vec{w}_{\text{ref}}$), and use it as the initial condition for the Newton-Krylov solution.

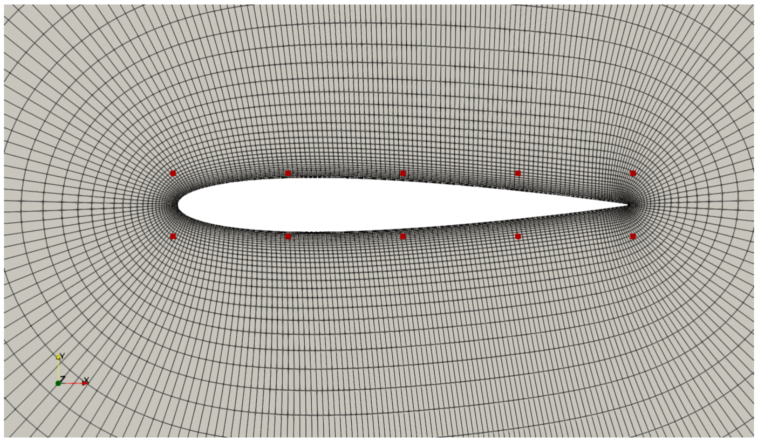

Figure 1. Mesh and FFD for the NACA0012 case.

- Use $\vec{x}_v^*$ and $\vec{w}_r^0$ as the input, solve the reduced residual equation $\vec{R}_r(\Phi\vec{w}_r, \vec{x}_v^*) = 0$ using the Newton-Krylov method, and get the converged reduced state variable vector $\vec{w}_r^*$ (Algorithm 1).
- Compute the new, full-order state variables $\vec{w}^* = \Phi\vec{w}_r^*$
- Compute the new objective functions (e.g., C_D) using the new full-order state variable $\vec{w}^*$

III. Results and Discussion

In this section, we evaluate the proposed ROM approach using two cases: the NACA0012 airfoil and the Cessna 172 wing.

1. NACA0012 airfoil

We use the NACA0012 airfoil as our first benchmark. The chord (c) is set to be 1 m, and the span is set to be 0.1 m with one cell in the spanwise (z) direction. The flow condition is incompressible with Reynolds number 10^6 and Mach number 0.03 (U_∞=10 m s^{-1} and T=273 K). We generate a coarse structured mesh with 15,232 cells using the pyHyp package [19], and the computational domain extends 30 chords from the surface, as shown in Fig. 1. The system has N = 138,034 state variables, or degrees of freedom (DOF), in total. The averaged y^+ is 32.1, so the wall function is used. The functions of interest are C_D and C_L.

In this section, we conduct two tests: The first test uses the airfoil pitch as the design variable, and the second one uses the airfoil shape as the design variable. As mentioned before, we use the FFD approach to parameterize the surface geometry. The pitch change is achieved by rotating all the FFD points, while the shape change is done by modifying the locations of individual FFD points. The FFD points are shown as red squares in Fig. 1.

In the first test, we run five simulations with a pitch angle ranging from -4 to 4 degrees and an interval of 2 degrees. The flow simulations are run for 1000 steps, for which the residuals drop eight orders of magnitudes. When using hyper-reduction, the residual is written every four steps for each sample, resulting in a total of 1250 residual snapshots. 1250 interpolation points are used, or $k = 1250$. Since there are N = 138,084 state variables in total, the number of residual evaluations required for the reduced system drops considerably. We predict C_D and C_L at -3, -1, 1, and 3 degrees using the flow field at 4 degrees as the initial condition. We require the norm of the reduced-order

Table 1. **Comparison of speed and memory cost between CFD and ROM (NACA0012 pitch). The superscript * denotes the estimated runtime.**

	Runtime (s)	Memory (GB)
CFD	148.0	0.1
ROM-BF offline	4	0.2
ROM-BF online	13.7	0.2
ROM-HR offline	–	6.0
ROM-HR online	0.478*	2.1

residual to drop two orders of magnitude ($r_{\text{tol}} = 1 \times 10^{-2}$). This is sufficient because the model-reduction errors dominate beyond $r_{\text{tol}} < 1 \times 10^{-2}$, and there is no need to converge the reduced-order residual further. The CFD and ROM simulations are conducted using the Intel Xeon E5-2640 CPU v2 (2.0 GHz) with one core (serial run).

Table 1 summarizes the speed and memory cost between the CFD and ROM computation. The offline brute-force ROM (referred to as ROM-BF) runs only once, and its runtime is negligible. When using hyper-reduction (referred to as ROM-HR), the offline costs are currently high and are not listed. The bottlenecks appear during the SVD calculation of the residual snapshot matrix, P, as well as when solving the DEIM algorithm. To overcome the high offline costs, we will improve the performance of large-scale SVD computations with more robust algorithms and direct linear solvers in our future work.

We directly compute the runtime ratio between CFD and ROM-BF. However, for ROM-HR, we only provide estimated runtime. As previously mentioned, the full residual is still being evaluted when using ROM-HR, although we are only interested in the values at the k interpolation points. When evaluating the full residual, the number of cells in which the residual is evaluated follows $\mathcal{O}(N)$. If we are only interested in the residual at the k interpolation points, the number of residual evaluations follows $\mathcal{O}(k)$. We can then approximate the estimated runtime T_{hr} as: $T_{hr} = T_{bf} \times k/N$, where T_{bf} is the runtime of ROM-BF. The averaged runtime ratio between CFD and online ROM-BF is 10.8, while the average estimated runtime ratio between CFD and online ROM-HR is 309.6, and the average speedup between ROM-HR and ROM-BF is 28.7. The memory cost of both online and offline ROMs is about 50% more than that of the CFD solution. This increases drastically when using ROM-HR. This is because the number of residual snapshots and interpolation points is much higher than the number of state snapshots, which is reflected in the size of various matrices in both the online and offline stages. In the future, we will improve the memory performance by reducing the number of required interpolation points for ROM-HR. In summary, the nonlinear brute-force ROM increases the simulation speed by a factor of ten, while requiring similar amounts of memory. The hyper-reduction ROM is expected to achieve an additional twenty times speed up compared with the brute-force ROM.

Table 2 shows the comparison of C_D and C_L computed by the CFD and both ROMs. We observe excellent agreement between CFD and ROM-BF with an averaged error of less than 1.0%. The agreement between CFD and ROM-HR is also good, with the averaged error for C_D under 3.0% and for C_L under 2.0%.

Figure 2 compares the velocity magnitude contours between CFD and ROM-BF. As mentioned before, the main advantage of using a projection-based ROM is that it can approximate full-order flow fields, compared with interpolation-based ROMs. This is clearly demonstrated here as we observe reasonably good agreement of velocity magnitude distribution between CFD and ROM-BF. Reasonably good agreement is also observed for ROM-HR (not shown). Similarly, we plot the pressure profiles from the CFD and both ROMs and observe an excellent agreement (the CFD and ROM profiles are not distinguishable visually), as shown in Fig. 3.

In the second test, we use the displacements of the 10 FFD points as the design variables to change the airfoil shape. We use the Latin hypercube sampling (LHS) approach (maximal min-distance) to generate 50 random designs for these ten design variables. The bounds of the design variables are from -0.025 to 0.025 m. Examples of CFD samples are shown in Fig. 4, along with their full-order solutions. We observe a wide range of shape variations, which provides sufficient design freedom for rapid aerodynamic analysis. The angle of attack for these cases is 4 degrees.

The flow conditions and CFD configurations are the same as the previous test. The reduced system is also similar. However, this time we use 15 state basis vectors, as using additional basis vectors does not improve the system much and increases the computational cost. The residual is written every five steps for each sample, resulting in 10,000 residual snapshots. $k = 2750$ interpolation points are used. Again, this is much lower than the number of DOF in the system. To evaluate the performance of the ROMs, we generate ten random shapes using the same LHS method. Note that the prediction points do not overlap with our CFD samples.

A factor that impacts performance greatly when solving the reduced residual equation (2.6) using the Newton-

Table 2. Comparison of simulated C_D and C_L between CFD and ROMs (NACA0012 pitch).

	CFD	ROM-BF	ROM-HR
C_D ($\theta = -3$ deg)	0.01423754	0.01412948	0.01363337
C_L ($\theta = -3$ deg)	0.10133706	0.10293115	0.10427677
C_D ($\theta = -1$ deg)	0.01520921	0.01504534	0.01533526
C_L ($\theta = -1$ deg)	0.30434491	0.30726999	0.29927512
C_D ($\theta = 1$ deg)	0.01722198	0.01710128	0.01632709
C_L ($\theta = 1$ deg)	0.50353991	0.50520139	0.51248370
C_D ($\theta = 3$ deg)	0.02040783	0.02042978	0.02066389
C_L ($\theta = 3$ deg)	0.69627769	0.69565701	0.69080539

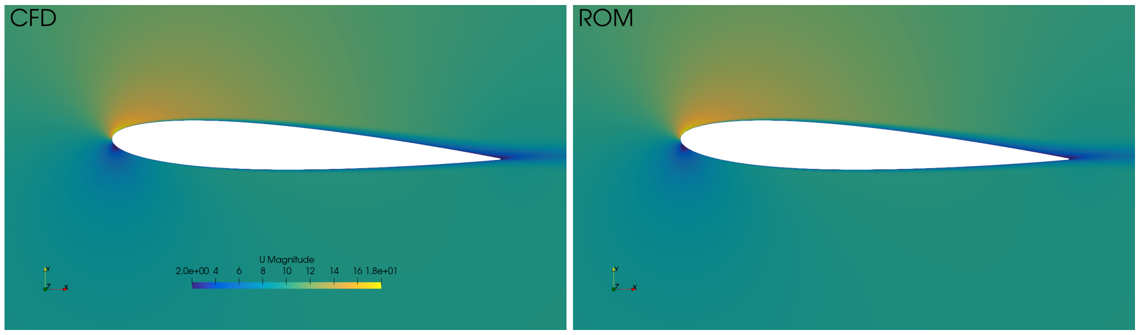

Figure 2. Comparison of velocity magnitude contours between CFD (left) and ROM (right) at $\theta = 3$ degree (NACA0012 pitch).

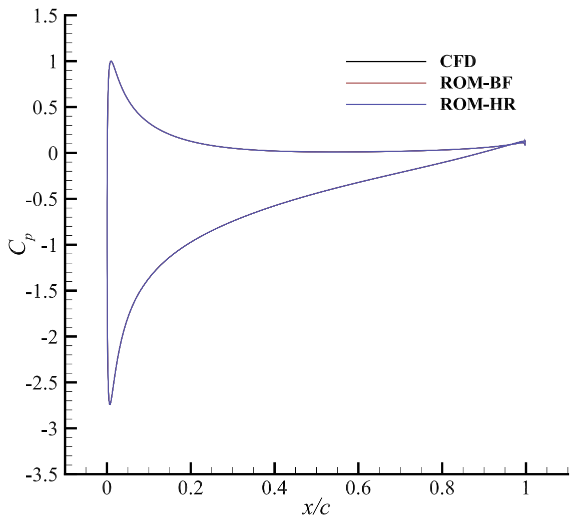

Figure 3. Comparison of pressure profile between CFD and ROM at $\theta = 3$ degree (NACA0012 pitch).

Table 3. Comparison of speed and memory cost between CFD and ROM (NACA0012 shape). The superscript $*$ denotes the estimated runtime.

	Runtime (s)	Memory (GB)
CFD	148.0	0.1
ROM-BF offline	19	0.3
ROM-BF online	35.7	0.6
ROM-HR offline	–	24.5
ROM-HR online	2.7^*	4.6

Krylov method is the initial condition. It is known that the Newton method performs reasonably well when the initial guess is relatively close to the solution, whereas a bad initial guess may cause the Newton method to diverge. Therefore, we implement a method to dynamically choose the best initial state variables from the CFD samples and use them to initialize the Newton solution. The criteria we use is based on the norm of the difference between the design variable vectors, $L = ||\vec{x}_{\text{prediction}} - \vec{x}_{\text{sample}}||$, and we will choose the CFD sample that has the minimal L.

Table 3 shows the runtime and memory cost for the NACA0012 shape case. Again, the offline costs when using hyper-reduction are very large and not listed. A larger amount of memory is required for this case in both the online and offline stages, as the number of residual snapshots and interpolation points is considerably higher. The offline costs without hyper-reduction are again low, as well as the memory costs in both the online and offline stages, although they are slightly higher than the previous case due to a higher number of state snapshots and basis vectors. The runtime for the online stage increases and the average runtime ratio between CFD and ROM-BF is 4.1. The average estimated runtime ratio between CFD and ROM-HR is 54.8, and the average speedup between ROM-HR and ROM-BF is 13.4. This is lower than the estimated average speedup between the ROMs when the airfoil's pitch was the only design

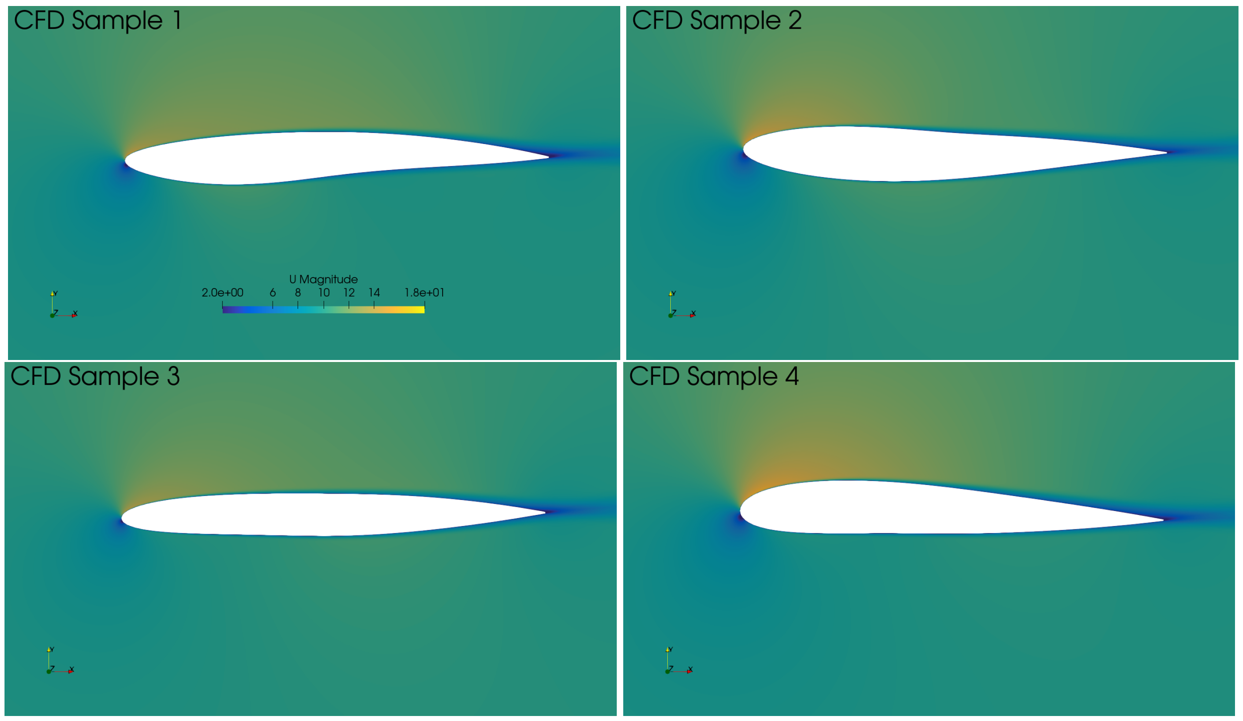

Figure 4. Examples of random airfoil shapes for the CFD samples and their velocity magnitude distributions (NACA0012 shape).

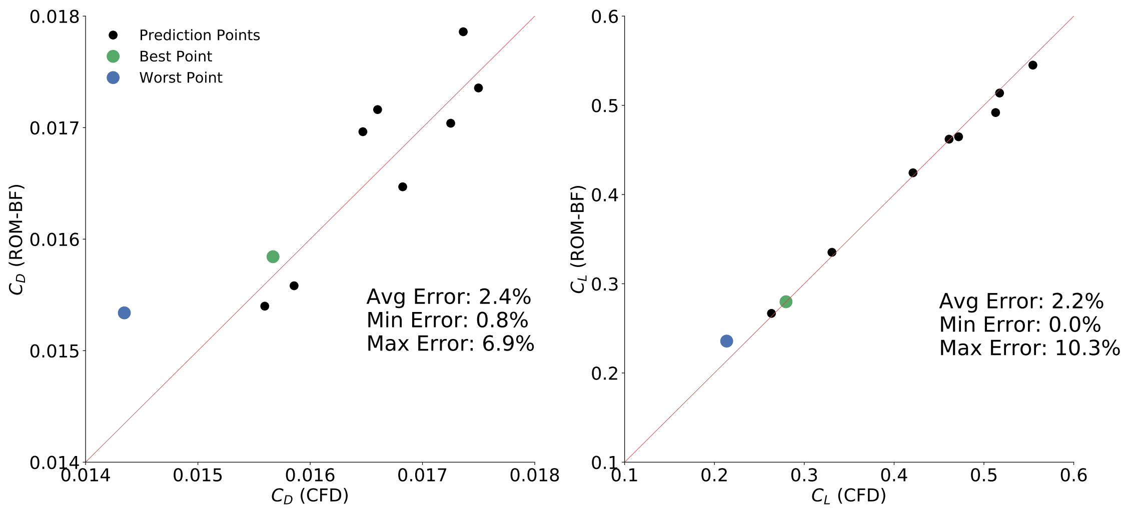

Figure 5. Comparison of C_D and C_L predicted by CFD and ROM-BF (NACA0012 shape).

variable because there are more interpolation points. The overall increase in runtime is primarily due to the increase in basis vectors. As shown in Eqs. 2.13 and 2.14, we need to compute the matrix-vector product for each basis vector in the Φ matrix. For each matrix-vector product computation, we need to evaluate the residual once. This has become the bottleneck of brute-force ROM because evaluating the full-order residual is expensive. This limitation will be mitigated by using the hyper-reduction ROM, as mentioned above.

Figure 5 shows the comparison of C_D and C_L computed through CFD and predicted by ROM-BF. The agreement in C_L and C_D are similar, with the averaged errors both under 2.5%. The maximum error in C_L is much higher than in C_D, at over 10%. Figure 6 shows the comparison of C_D and C_L, this time comparing ROM-HR against CFD. The agreement in C_L is better than C_D this time, although both are below 2.5%. ROM-HR performs similarly to ROM-BF in predicting C_D and does better in predicting C_L. Next, we extract the best prediction point and the worst prediction

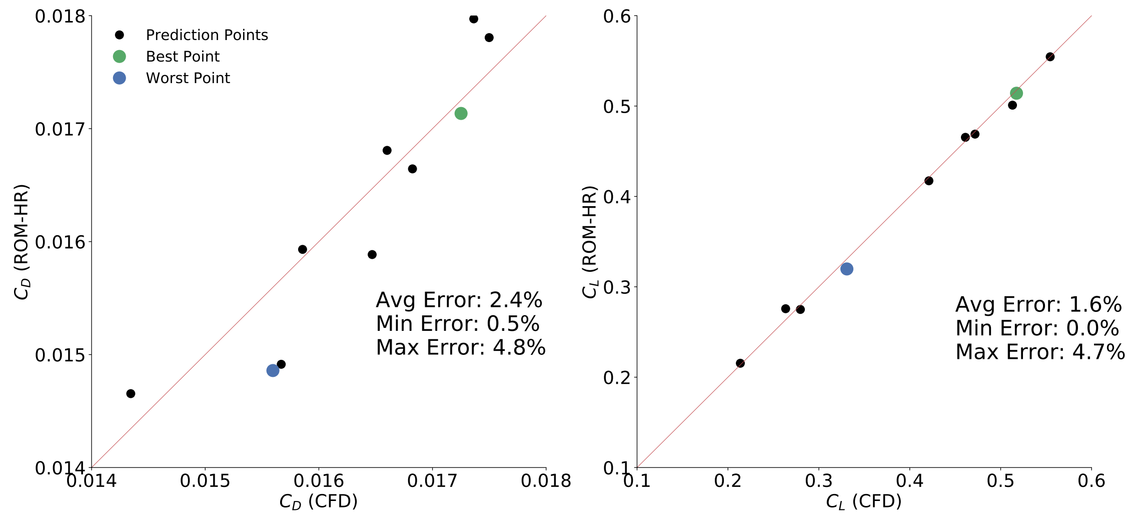

Figure 6. Comparison of C_D and C_L predicted by CFD and ROM-HR (NACA0012 shape).

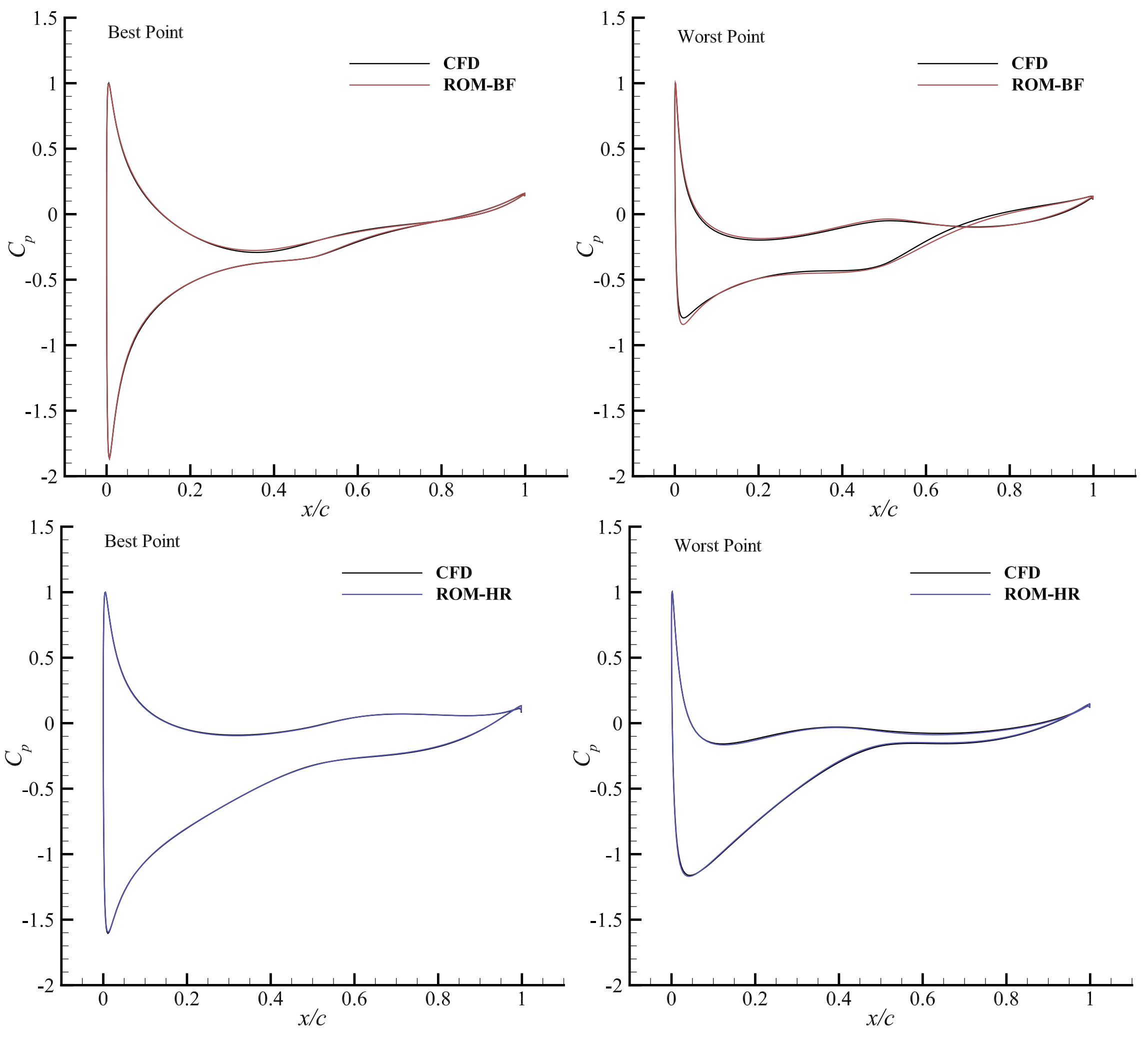

Figure 7. Comparison of pressure profile between CFD and ROM (NACA0012 shape).

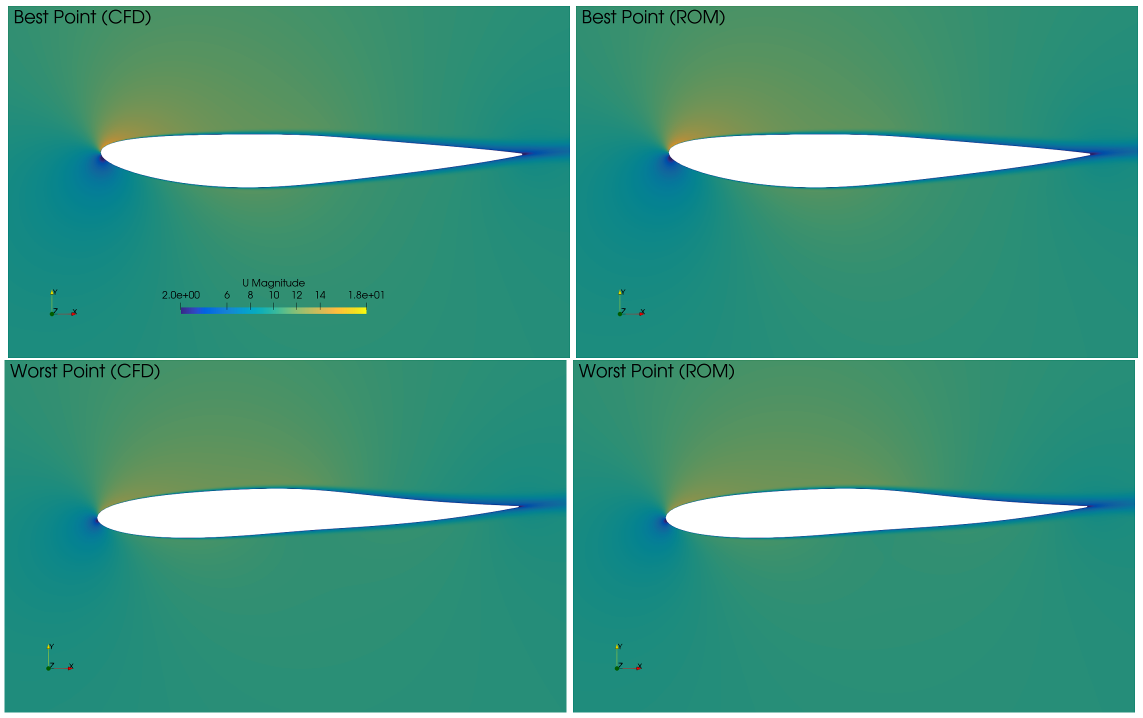

Figure 8. Comparison of velocity magnitude contours predicted by CFD and ROM (NACA0012 shape; brute-force).

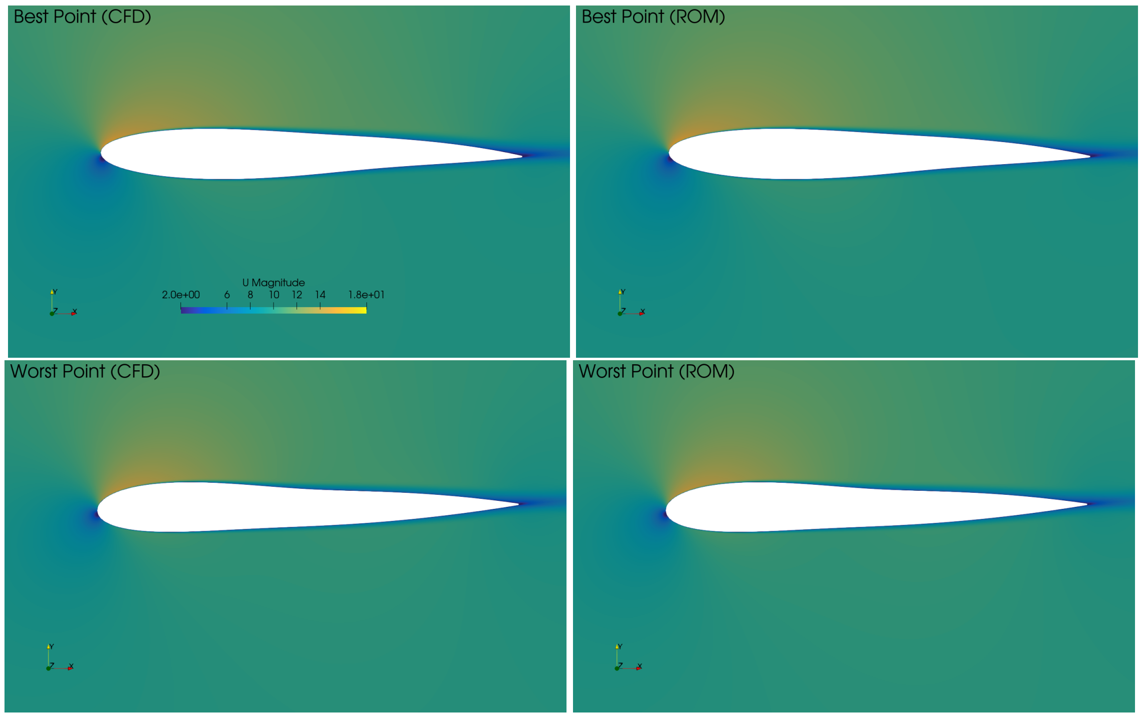

Figure 9. Comparison of velocity magnitude contours predicted by CFD and ROM (NACA0012 shape; hyper-reduction).

point and analyze their velocity and pressure distributions from CFD and ROM simulations, as shown in Figs. 7, 8, and 9. As expected, the velocity and pressure agree well for the best prediction point for both ROMs and at the worst prediction point for ROM-HR. At the worst prediction point for ROM-BR, we observe a noticeable difference in the pressure profile compared to CFD; however, this difference is still acceptable for practical aerodynamic analysis.

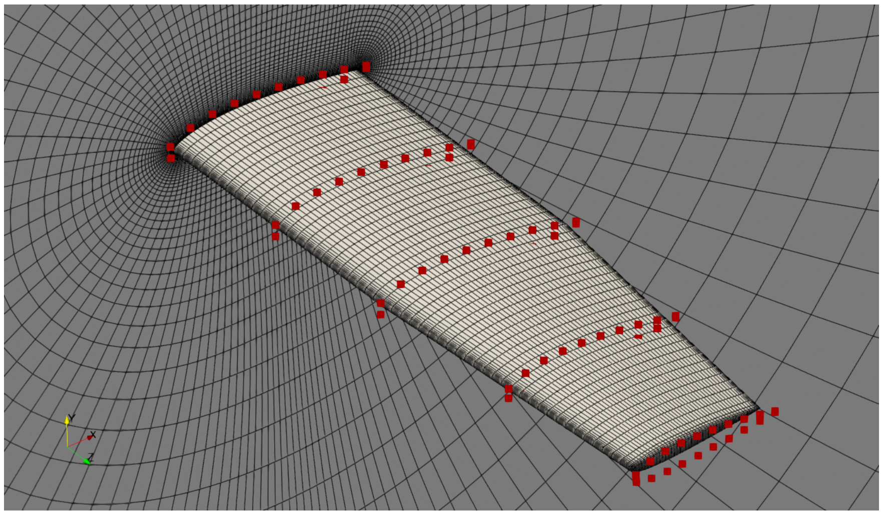

Figure 10. Mesh and FFD for the Cessna 172 case.

2. Cessna 172 wing

Next, we use the Cessna 172 wing as the test case. The root chord (c) is 1.67 m, and the semi-span aspect ratio is 3.2. The flow condition is incompressible with Reynolds number 7×10^6 and Mach number 0.187 (U_∞=63.8 m s^{-1}). We generate a structured mesh with 609,280 cells using the pyHyp package [19], and the computational domain extends 30 chords from the surface, as shown in Fig. 10. The averaged y^+ is 55.1, so the wall function is used. The functions of interest are C_D and C_L. Since this case is computationally much larger, we encountered issues implementing ROM-HR, primarily due to disk space and memory. Writing the residual to disk every few steps was not an issue in the NACA0012 cases but required a large amount of storage for this case. Additionally, a larger than the available amount of memory is required when using ROM-HR, and thus results are only presented for ROM-BF. In the future, we will improve upon our method so significantly fewer interpolation points and residual snapshots are required for accurate predictions.

The jobs are run on the Skylake nodes of the Stampede 2 supercomputer [a]. The Skylake nodes are equipped with Intel Xeon Platinum 8160 CPUs running at 2.1 GHz, where each node has 48 CPU cores and 196 GB of memory. We run the Cessna 172 case in parallel using 48 CPU cores and 2 Skylake nodes.

We use 100 FFD points to parameterize the wing surface at five spanwise locations. The design variables are the five twists at these spanwise locations, as shown in Fig. 10. Twisting is achieved by rotating a set of FFD points at a given spanwise section.

Similar to the previous case, we use the LHS approach (maximal min-distance) to generate 25 random designs for these five twist design variables. The bounds of the twist variables are from -1 to 1 degree. The angle of attack is kept at 2.5 degrees. We run the CFD simulations for 2000 steps, and the flow residuals drop eight order of magnitudes. The reduced system's configuration is similar to the previous case, except that we keep only ten basis vectors.

Table 4 shows the speed and memory cost for the offline and online ROM simulations. The memory cost and offline ROM speed are similar to what we observed for the airfoil case. However, the online ROM speed further decreases, and the averaged runtime ratio between the CFD and ROM is only 1.9. Again, the reduced speed is partially due to the extension from two-dimensional to three-dimensional cases where the cost of evaluating the full-order residual increases drastically. In addition, we observe an even more challenging convergence for the reduced Newton iterations. More specifically, the line search requires more attempts, and the Newton method generally requires more iterations than the two-dimensional airfoil case. We will improve the speed of the online ROM using hyper-reduction.

Figure 11 shows the comparison of C_D and C_L predicted by CFD and ROM. Here we observe excellent agreement between the CFD and ROM solutions with an averaged error less than 1% and a maximal error less than 2.0%. To further demonstrate this excellent agreement, we use the worst prediction point for detailed analysis. We plot the pressure contours at the upper and lower surfaces of the wing and compare the results between CFD and ROM, as shown in Fig. 12. As expected, the difference between the CFD and ROM simulations are not visually distinguishable. We then plot the pressure profiles at the root and tip sections of the wing, shown in Fig. 13. We confirm the excellent agreement of the pressure profiles at the root section. In the tip section, the difference between the CFD and ROM is

[a]https://www.tacc.utexas.edu/systems/stampede2

Table 4. Comparison of speed and memory cost between CFD and ROM (Cessna 172 twist).

	Runtime (s)	Memory (GB)
CFD	108	5.9
ROM-offline	9	4.8
ROM-online	57	10.3

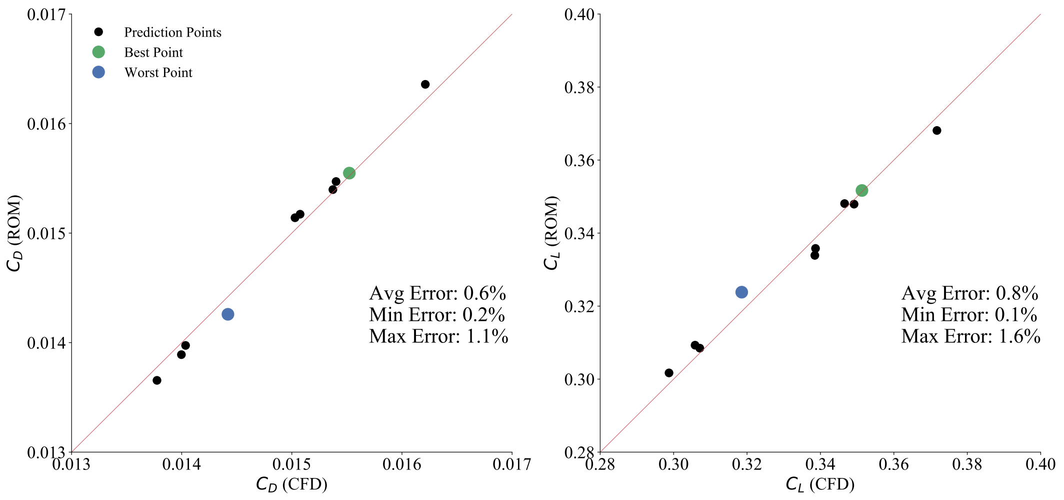

Figure 11. Comparison of C_D and C_L predicted by CFD and ROM.

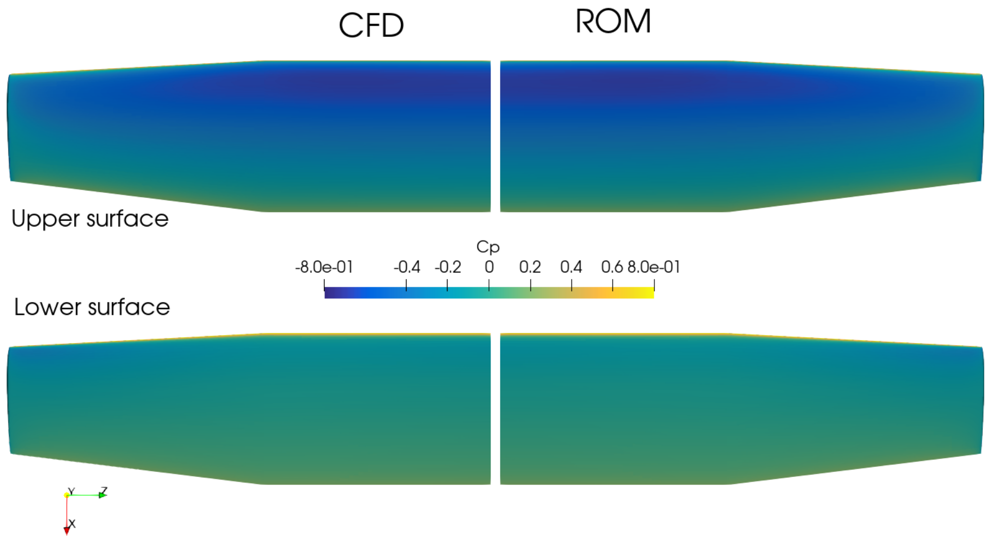

Figure 12. Comparison of pressure contours between the CFD and ROM simulations.

noticeable, but it is still well acceptable in the practical aerodynamic analysis. Finally, we plot the velocity magnitude contour at the root and tip sections in Fig. 14. As expected, the ROM simulation provides approximated three-dimensional flow fields, and the agreement between the CFD and ROM results is excellent.

IV. Conclusion

In this paper, we elaborate on the nonlinear reduced-order modeling (ROM) formulations we recently developed. To be more specific, we first use the state variables from a CFD dataset with various designs to generate snapshot

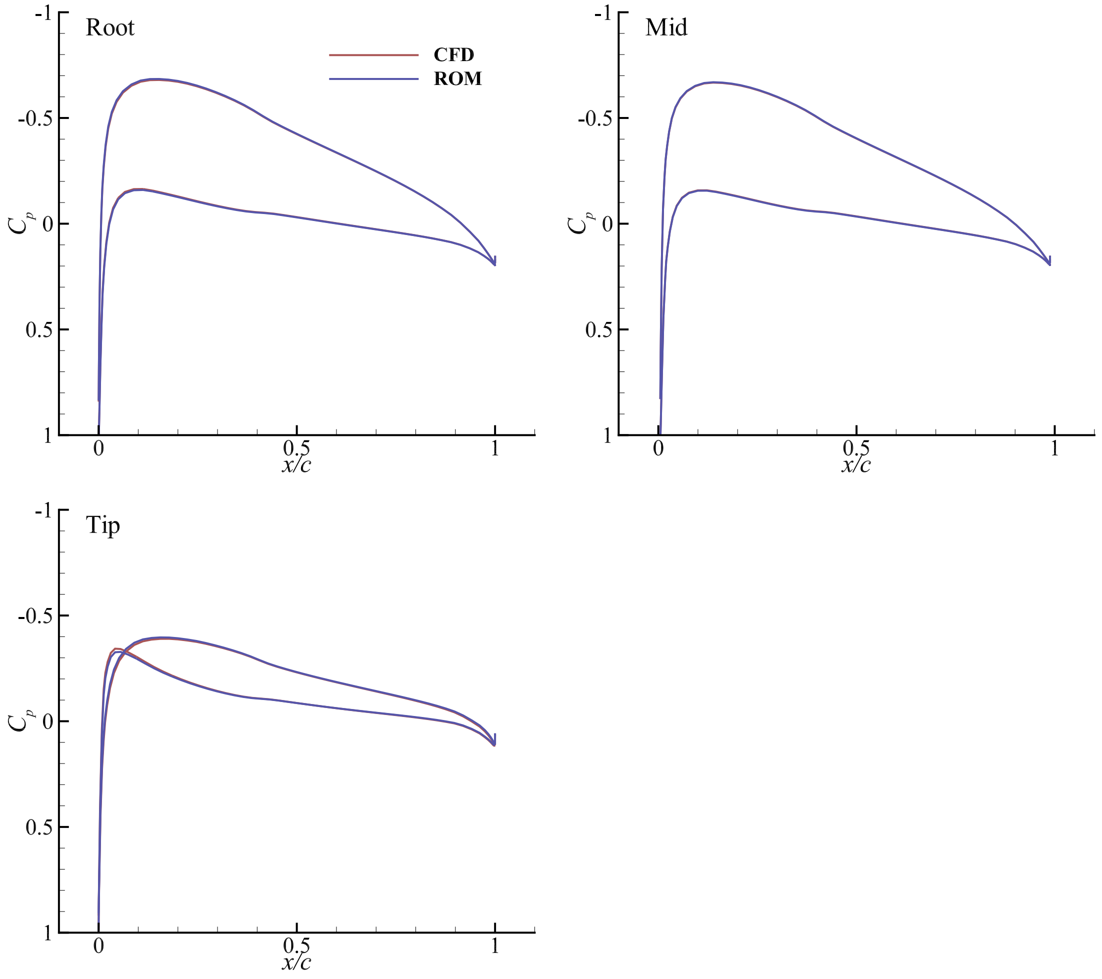

Figure 13. Comparison of pressure profiles between the CFD and ROM simulations at the root, mid, and tip sections.

matrices of the state as well as the residual values if using hyper-reduction. Then, we compute the singular value decomposition (SVD) to find the basis vectors from the snapshot matrices and project the full-order state variables to a lower-dimensional space. Hyper-reduction, although not fully implemented, eliminates the need to evaluate the full-order residual and rather evaluates it at a set of interpolation points that are used to approximate the full order-residual. Next, we use the least-square Petrov-Galerkin method to formulate the reduced-order residual equation and use the Newton-Krylov method to solve it.

We then implement the formulation into an incompressible flow solver in OpenFOAM, a popular open-source code for multiphysics analysis. We use two cases to demonstrate the performance of our ROM implementation: the NACA0012 airfoil and the Cessna 172 wing. We observe that the ROM simulations increase the speed of the flow simulations with runtime ratios for ROM-BF ranging from 1.9 to 10.8. When using ROM-HR, the estimated runtime ratios for the NACA0012 cases range from 55 to 310. In addition, the memory ratio between ROM-BF and CFD simulations is no more than two. In terms of accuracy, the averaged error in C_D and C_L ranges from 1.6% to 2.4%. The ROMs provide approximated three-dimensional flow fields (pressure and velocity), and the agreements between the CFD and both ROMs are very reasonable for aerodynamic design optimization. Future work will include the implementation of submeshes to lower the number of residual evaluations when using hyper-reduction to fully realize the gain in speed. Overall, the proposed ROM approach has the potential to become a useful tool for rapid aerodynamic analysis.

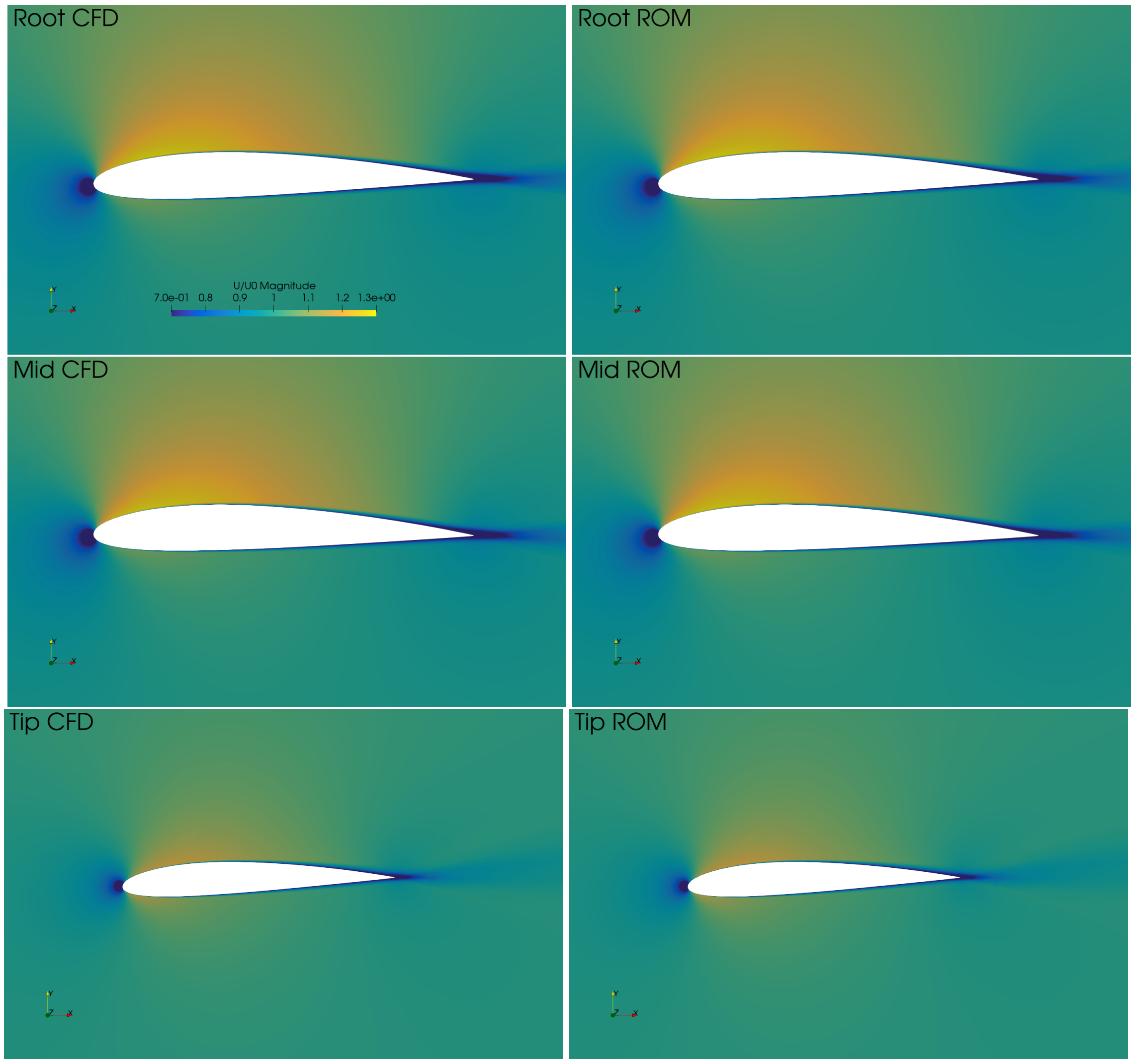

Figure 14. Comparison of velocity magnitude contours predicted by CFD and ROM at the root, mid, and tip sections.

V. Acknowledgments

The computations were done in the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant number ACI-1548562.

References

[1] Venter, G., Haftka, R. T., and Starnes Jr, J. H., "Construction of response surface approximations for design optimization," *AIAA journal*, Vol. 36, No. 12, 1998, pp. 2242–2249.

[2] Robinson, T., Eldred, M. S., Willcox, K. E., and Haimes, R., "Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping," *AIAA journal*, Vol. 46, No. 11, 2008, pp. 2814–2822.

[3] Koziel, S. and Leifsson, L., "Surrogate-based aerodynamic shape optimization by variable-resolution models," *AIAA journal*, Vol. 51, No. 1, 2013, pp. 94–106.

[4] Li, J., Bouhlel, M. A., and Martins, J. R. R. A., "Data-based Approach for Fast Airfoil Analysis and Optimization," *AIAA Journal*, Vol. 57, No. 2, February 2019, pp. 581–596. doi:10.2514/1.J057129.

[5] Bouhlel, M. A., He, S., and Martins, J. R. R. A., "Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes," *Structural and Multidisciplinary Optimization*, Vol. 61, March 2020, pp. 1363–1376. doi:10.1007/s00158-020-02488-5.

[6] Du, X., He, P., and Martins, J. R. R. A., "A B-Spline-based Generative Adversarial Network Model for Fast Interactive Airfoil Aerodynamic Optimization," *AIAA SciTech Forum*, AIAA, Orlando, FL, Jan. 2020. doi:10.2514/6.2020-2128.

[7] LeGresley, P. and Alonso, J., "Airfoil design optimization using reduced order models based on proper orthogonal decomposition," *Fluids 2000 conference and exhibit*, 2000, p. 2545.

[8] Willcox, K. and Peraire, J., "Balanced model reduction via the proper orthogonal decomposition," *AIAA journal*, Vol. 40, No. 11, 2002, pp. 2323–2330.

[9] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for numerical methods in engineering*, Vol. 86, No. 2, 2011, pp. 155–181.

[10] Collins, G., Fidkowski, K., and Cesnik, C. E., "Output Error Estimation for Projection-Based Reduced Models," *AIAA Aviation 2019 Forum*, 2019, p. 3528.

[11] Collins, G., Fidkowski, K., and Cesnik, C. E., "Petrov-Galerkin projection-based model reduction with an optimized test space," *AIAA Scitech 2020 Forum*, 2020, p. 1562.

[12] Tsiolakis, V., Giacomini, M., Sevilla, R., Othmer, C., and Huerta, A., "Nonintrusive proper generalised decomposition for parametrised incompressible flow problems in OpenFOAM," *Computer physics communications*, Vol. 249, 2020, pp. 107013.

[13] Jasak, H., Jemcov, A., and Tuković, Z., "OpenFOAM: A C++ Library for Complex Physics Simulations," *International Workshop on Coupled Methods in Numerical Dynamics, IUC*, Citeseer, 2007.

[14] Weller, H. G., Tabor, G., Jasak, H., and Fureby, C., "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computers in Physics*, Vol. 12, No. 6, 1998, pp. 620–631. doi:10.1063/1.168744.

[15] Patankar, S. V. and Spalding, D. B., "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *International Journal of Heat and Mass Transfer*, Vol. 15, No. 10, 1972, pp. 1787–1806. doi:10.1016/0017-9310(72)90054-3.

[16] Rhie, C. and Chow, W. L., "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1525–1532. doi:10.2514/3.8284.

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

[18] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, No. AIAA 2010-9231, Fort Worth, TX, Sept. 2010. doi:10.2514/6.2010-9231.

[19] Secco, N., Kenway, G., He, P., CA, M., and JRRA, M., "Efficient mesh generation and maninpulation for aerodynamic shape optimization," *AIAA Journal*, 2020, (In press).

[20] Hernandez, V., Roman, J. E., and Vidal, V., "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems," *ACM Trans. Math. Software*, Vol. 31, No. 3, 2005, pp. 351–362.

[21] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018.

[22] Chaturantabut, S. and Sorensen, D., "Discrete Empirical Interpolation for Nonlinear Model Reduction," Vol. 32, 01 2010, pp. 4316 – 4321. doi:10.1109/CDC.2009.5400045.

[23] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "An Aerodynamic Design Optimization Framework Using a Discrete Adjoint Approach with OpenFOAM," *Computers & Fluids*, Vol. 168, May 2018, pp. 285–303. doi:10.1016/j.compfluid.2018.04.012.

[24] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "DAFoam: An Open-Source Adjoint Framework for Multidisciplinary Design Optimization with OpenFOAM," *AIAA Journal*, 2020. doi:10.2514/1.J058853.

[25] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, 2019.