Model Reduction Using Interpolated Systems of Equations

Krzysztof J. Fidkowski*, Rakesh Halder†, and Kevin J. Maki‡
University of Michigan, Ann Arbor, MI, 48188

This paper presents a model reduction approach based on interpolation of linear systems that govern a reduced state approximation. These systems are constructed at training points, from state and residual perturbation data, in an offline stage. In the online stage, the matrix systems of equations are interpolated to a desired point in parameter space, allowing for a rapid solution of the modal state coefficients. The method is similar to traditional interpolation methods, in that quantities are interpolated in the parameter domain, but it extends the interpolated quantities beyond just outputs or state coefficients. It is also similar to projection-based methods, in that the reduced systems are obtained from a projection step using a suitably-chosen test space, but it does not require a direct link back to the full-order code for efficient online implementation. The intended application is the generation of reduced models for external aerodynamics simulations of parametrized shapes, and results are presented for prototypical two-dimensional flows with modeled turbulence. The results indicate that, when the parameter space sampling is sparse, projection-based methods offer improved solution fidelity over interpolation methods. Furthermore, the interpolated systems method exhibits results similar to projection, at a much lower implementation and run-time complexity.

I. Introduction

Numerical simulations are now ubiquitous in engineering and support support not only direct analysis, but also higher-level tasks, such as design optimization and uncertainty quantification (UQ). Sustained growth of computational power over the last several decades has enabled the application of these methods to the direct simulation of large-scale problems involving complex physics. However, challenges remain in design and UQ applications, which often involve a large number of active parameters. In these cases, the number of simulations required to adequately explore the parameter space remains large even with sophisticated dimension reduction and sparse-grid integration techniques.

When the analysis problem is expensive, such as in the case of multi-physics computational fluid dynamics, surrogate models, such as reduced-order models, are needed. Reduced models capture the most important input-output relationships from the high-fidelity equations, and they can reduce the cost of state calculations by approximating the state as lying in a relatively low-dimensional subspace of the high-fidelity state space. The simplest such model is interpolation, in which outputs and even states are interpolated from training points to other points in the parameter domain. The main advantage of interpolation is that it is non-intrusive: only standard output or state data are used, although gradient information can also be incorporated [1–3]. However, achieving high accuracy is difficult, particularly for highly-nonlinear problems, and it generally requires many training points.

An alternative to interpolation is projection-based model reduction, in which the equations are projected onto a reduced test space [4]. Projection has proven to be effective in producing efficient and accurate reduced models of high-fidelity systems in both steady and unsteady fluid dynamics simulations [5–11]. Projection methods operate in two stages: offline, involving calculations that scale with the full-order system size, and online, involving calculations that only depend on the size of the reduced system. For nonlinear problems, *hyper-reduction* methods [10–14] must be employed to avoid online solution costs that scale with the size of the high-fidelity system. These methods generally rely on limited sampling of the nonlinear residual/physics to obtain the reduced system,

Beyond projection-based methods, other works have looked at state approximation that do not remain confined to a linear manifold [15], including neural-network based state approximations [16–19]. However, such extensions are not considered in this work.

In this paper, we propose a model reduction method that blends interpolation and projection. Our goal is to obtain accuracy similar to projection, with the ease and non-intrusiveness of interpolation. The resulting method shares several characteristics with projection, including a proper orthogonal decomposition of state and residual snapshots, and the

*Professor, Department of Aerospace Engineering, AIAA Associate Fellow.
†Graduate Student, Department of Aerospace Engineering, AIAA Student Member.
‡Associate Professor, Department of Naval Architecture and Marine Engineering.

generation of projected systems of equations. However, these projected systems are constructed only at training points, and the online implementation involves simple interpolation and solution of these systems. The method yields field state data, via the POD approximation, and outputs, which can be calculated directly from the reduced coefficients via a similar procedure.

The remainder of this paper presents the blended interpolation/projection model reduction approach in the context of high-order computational fluid dynamics. The method is compared to standard interpolation and projection approaches, for several two-dimensional test cases involving parameterized shape changes.

II. Discretization

For the full-order model, we consider the steady compressible Navier-Stokes equations,

$$\nabla \cdot \vec{\mathbf{H}}(\boldsymbol{u}, \nabla \boldsymbol{u}) + \mathbf{S}(\boldsymbol{u}) \quad = \quad \mathbf{0}, \tag{1}$$

where $\boldsymbol{u}(\vec{x}, t) \in \mathbb{R}^s$ is the conservative state vector, s is the state rank, $\vec{\mathbf{H}}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \vec{\mathbf{F}}(\boldsymbol{u}) + \vec{\mathbf{G}}(\boldsymbol{u}, \nabla \boldsymbol{u})$ is the total, inviscid and viscous, flux, and $\mathbf{S}(\boldsymbol{u})$ is a source term that is active when modeling turbulence with Reynolds averaging. We use the Spalart-Allmaras (SA) model in this work [20].

We use the discontinuous Galerkin finite-element discretization [21], although the model reduction approach extends to other discretizations, such as finite-volume or finite-difference methods. The computational domain Ω is divided into N_e elements, Ω_e, in a non-overlapping tessellation T_h. Inside element Ω_e, the state is approximated by polynomials of order p, with no continuity constraints on the element boundary. Formally, we write: $\boldsymbol{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \left\{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h\right\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e.

The discontinuous Galerkin weak form of Eqn. 1 is obtained by multiplying by test functions in the same approximation space, integrating by parts, and coupling elements via single-valued fluxes that are functions of the states on the two adjacent elements:

$$-\int_{\Omega_e} \nabla \boldsymbol{w}_h^T \cdot \vec{\mathbf{H}}(\boldsymbol{u}_h, \nabla \boldsymbol{u}_h)\, d\Omega + \int_{\partial\Omega_e} \boldsymbol{w}_h^T \widehat{\mathbf{H}} \cdot \vec{n}\, ds - \int_{\partial\Omega_e} \partial_i \boldsymbol{w}_h^{+T} \mathbf{K}_{ij}^{+} \left(\boldsymbol{u}_h^{+} - \widehat{\boldsymbol{u}}_h\right) n_j\, ds \quad = \quad 0 \quad \forall \boldsymbol{w}_h \in \boldsymbol{\mathcal{V}}_h, \tag{2}$$

where $(\cdot)^T$ denotes transpose, $\widehat{\boldsymbol{u}}_h = (\boldsymbol{u}_h^{+} + \boldsymbol{u}_h^{-})/2$, and on the element boundary $\partial\Omega_e$, $(\cdot)^+, (\cdot)^-$ denote quantities taken from the element or its neighbor (or boundary condition), respectively. For the normal flux, $\widehat{\mathbf{H}} \cdot \vec{n}$, we use the Roe-approximate Riemann solver [22] and the second form of Bassi and Rebay (BR2) [23] for the viscous flux. Choosing a basis for the test and trial spaces yields a system of nonlinear equations, $\mathbf{R}(\mathbf{u}) = \mathbf{0}$, where $\mathbf{u}$ is the discrete state vector.

III. Model Reduction

The goal in model reduction is to obtain quantities of interest (outputs) and possibly states from parametrized simulations at a cost that is much lower than that of the original, full-order, solution. The full-order nonlinear system of equations is

$$\text{residual equations:} \qquad \mathbf{R}(\mathbf{u}, \boldsymbol{\mu}) = \mathbf{0}, \tag{3}$$

$$\text{outputs:} \qquad \mathbf{J}(\mathbf{u}, \boldsymbol{\mu}). \tag{4}$$

Here, $\mathbf{u} \in \mathbb{R}^N$ is the full-order state vector with $N \gg 1$ entries, $\boldsymbol{\mu} \in \mathbb{R}^p$ is the vector of (e.g. shape) parameters, $\mathbf{R} \in \mathbb{R}^N$ is the vector of residuals, and $\mathbf{J} \in \mathbb{R}^q$ is the vector of outputs.

Suppose that we have K *training* state vectors (snapshots), $\mathbf{u}_i$, $1 \le i \le K$, which are solutions of the high-fidelity problem at K different parameter values, $\boldsymbol{\mu}_i$. We assemble these vectors into a *snapshot matrix*, $\mathbf{S} \in \mathbb{R}^{N \times K}$, and compute the singular value decomposition (SVD),

$$\mathbf{S} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_K \\ | & | & & | \end{bmatrix}, \qquad \mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{5}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix of singular values and the matrices $\mathbf{U}$ and $\mathbf{V}$ contain the left and right singular vectors. We define the *state basis vectors*, $\boldsymbol{\phi}_i \in \mathbb{R}^N$, as the columns of the matrix $\mathbf{U}$, and their importance for representing the snapshots depends on the singular values, σ_i, in $\mathbf{\Sigma}$. For most practical problems, the singular values decay quickly, so that an accurate state representation can be obtained by using only a small number, $n \ll N$, of basis vectors:

$$\mathbf{u} \approx u_{r1}\boldsymbol{\phi}_1 + u_{r2}\boldsymbol{\phi}_2 + \cdots + u_{rn}\boldsymbol{\phi}_n \quad \Rightarrow \quad \mathbf{u} \approx \mathbf{\Phi}\mathbf{u}_r. \tag{6}$$

In this equation, the values u_{rj} are coefficients on the basis functions in the approximation of $\mathbf{u}$. The expression on the right is the matrix-form equivalent of this equation, in which $\mathbf{\Phi} \in \mathbb{R}^{N\times n}$ is the *basis matrix*, whose columns are the n basis vectors, $\boldsymbol{\phi}_j$. The vector $\mathbf{u}_r \in \mathbb{R}^n$ is then the *reduced state*. An important property of the SVD is that the basis vectors are orthonormal: $\boldsymbol{\phi}_i^T\boldsymbol{\phi}_j = \delta_{ij}$, which simplifies projection of a high-fidelity state to the reduced state. Multiplying the matrix form in Eqn. 6 by $\mathbf{\Phi}^T$ on both sides gives the desired reduced state, $\mathbf{\Phi}^T\mathbf{u} = \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{u}_r \;\Rightarrow\; \mathbf{u}_r = \mathbf{\Phi}^T\mathbf{u}$.

A. Interpolation

One way to obtain output and state data at a parameter value not in the training set, $\boldsymbol{\mu}$, is to interpolate them from the training points. If we are only interested in the outputs, we could interpolate $\mathbf{J}$ according to

$$\text{output iROM:} \quad \mathbf{J}(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{J}_i, \tag{7}$$

where $\mathbf{J}_i \equiv \mathbf{J}(\mathbf{u}_i, \boldsymbol{\mu}_i)$ represents the output vectors at the training points, and w_i are interpolation weights, which depend on $\boldsymbol{\mu}$. These weights could come from polynomial fitting, radial basis functions, Gaussian process regression, etc. In this work, we use Lagrange interpolation on structured one and two-dimensional lattice samples in parameter space. Note that this approach, termed *output iROM*, does not require calculation of the snapshot matrix or its SVD.

Alternatively, we can interpolate the entire state by using the reduced basis approximation. Let $\mathbf{u}_{ri} \equiv \mathbf{\Phi}^T\mathbf{u}_i$ be the reduced state representations of the snapshots. Interpolating these n numbers to the parameter point $\boldsymbol{\mu}$ gives

$$\text{state iROM:} \quad \mathbf{u}_r(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{u}_{ri}. \tag{8}$$

The entire flowfield can then be recovered from $\mathbf{u} = \mathbf{\Phi}\mathbf{u}_r$. Outputs can be computed from the recovered flowfield or from another model based directly on the reduced state. This approach, termed *state iROM*, can also be applied to distributions other than the entire flowfield, for example surface pressure/shear, or data on cut planes of the domain [24, 25]. In such cases, the snapshots are smaller as they consist of only the desired data.

B. Projection

The goal in projection-based methods is to obtain a smaller system of equations to solve for the reduced state $\mathbf{u}_r$. Using the state approximation from Eqn. 6 in the full-order equations, Eqn. 3, yields an over-determined system of N equations for n unknowns,

$$\mathbf{R}(\mathbf{\Phi}\mathbf{u}_r, \boldsymbol{\mu}) = \mathbf{0}. \tag{9}$$

We project this system using m *test vectors*, $\boldsymbol{\psi}_j$, assembled column-wise into a *test matrix*, $\mathbf{\Psi} \in \mathbb{R}^{N\times m}$, to obtain the *reduced residual vector*, $\mathbf{R}_r \in \mathbb{R}^m$,

$$\mathbf{R}_r \equiv \mathbf{\Psi}^T\mathbf{R}(\mathbf{\Phi}\mathbf{u}_r, \boldsymbol{\mu}) = \mathbf{0}. \tag{10}$$

The test vectors have a significant impact on the accuracy of the resulting reduced model. One choice is to seek test vectors that minimize the L_2 norm of the residual, and this gives rise to the least-squares Petrov-Galerkin (LSPG) method [10]. The starting point is the minimization problem:

$$\text{find } \mathbf{u}_r \text{ such that } ||\mathbf{R}||^2 = \mathbf{R}^T\mathbf{R} \text{ is minimized, where } \mathbf{R} = \mathbf{R}(\mathbf{u}_r). \tag{11}$$

We suppress the dependence of $\mathbf{R}$ on the parameter, with the understanding that a solution to this problem is sought at some arbitrary parameter vector, $\boldsymbol{\mu}$. Differentiating the squared residual norm with respect to $\mathbf{u}_r$ gives

$$\frac{\partial ||\mathbf{R}||^2}{\partial \mathbf{u}_r} = \mathbf{0} \quad \Rightarrow \quad \underbrace{\left[\frac{\partial \mathbf{R}}{\partial \mathbf{u}_r}\right]^T}_{\mathbf{\Psi}^T} \mathbf{R}(\mathbf{u}_r) = \mathbf{0}. \tag{12}$$

The test matrix for LSPG is therefore the derivative of the full-order residual with respect to the reduced state. One could compute these derivatives at the desired parameter point $\boldsymbol{\mu}$, but this would be expensive without a further, more complex, hyper-reduction approach. An alternative is to compute one test space for use with all parameter points. We do this by first calculating the n derivative vectors at each training point i, to obtain $\left.\frac{\partial \mathbf{R}}{\partial \mathbf{u}_r}\right|_i$. We next assemble these derivative vectors from all training points into a *residual snapshot* matrix, compute its SVD, and take as the test matrix $\mathbf{\Psi}$ the first m left singular vectors, where $n \leq m \leq Kn$. The purpose of the SVD is to identify a basis for the test space that best approximates the separate test spaces at the training points. We use $m = 2n$ as the number of test vectors, to yield a small, over-determined system.

Since the equations are nonlinear, a Newton-Raphson approach is used to obtain the reduced solution $\mathbf{u}_r$ for a given parameter vector $\boldsymbol{\mu}$. The reduced state is initialized using the state iROM approach, after which the reduced residual, $\mathbf{R}_r$, and its Jacobian, $\frac{\partial \mathbf{R}_r}{\partial \mathbf{u}_r}$, are computed. This is done through a finite-difference approach and the projection in Eqn. 10. Masked projection [26, 27] could be employed as a form of hyper-reduction in this step. The following small least-squares minimization problem is then solved,

$$\text{find } \Delta\mathbf{u}_r \text{ that minimizes } \left\| \mathbf{R}_r(\mathbf{u}_r) + \left.\frac{\partial \mathbf{R}_r}{\partial \mathbf{u}_r}\right|_{\mathbf{u}_r} \Delta\mathbf{u}_r \right\|. \tag{13}$$

This solution is obtained through a QR decomposition of $\frac{\partial \mathbf{R}_r}{\partial \mathbf{u}_r}$. The update is then applied to the reduced state and the process repeated until convergence, typically in 3-4 Newton iterations. Once the reduced state is available from this projection, or pROM, approach, the full-order state can be reconstructed and outputs can be computed using the same techniques as in the state iROM method.

C. System Interpolation

Compared to interpolation, projection-based model reduction has the advantage of using more information from the full-order system, namely the equations, in obtaining the reduced solution. This means that projection methods may require fewer training samples for a given level of accuracy. However, projection has the disadvantage of being more expensive due to the formation and solution of the projected system, and of being more intrusive into the full-order code due to the need to evaluate residuals, either full or hyper-reduced samples, in the online stage. In addition, for shape changes, the deformed geometry and mesh, or a portion of it, must be constructed for each online run to obtain the desired residuals. In this section we present a model reduction method that addresses these disadvantages while retaining some of the benefits of projection.

The method of system interpolation eliminates the intrusive aspect of projection-based methods by using only data from the training points during the online portion. As in the projection-based method, we perform an SVD of the state snapshots to obtain the state basis vectors in $\mathbf{\Phi}$, and a separate SVD of the residual derivative snapshots to obtain the test matrix, $\mathbf{\Psi}$. We then compute a reduced residual Jacobian matrix at each training point,

$$\mathbf{A}_i \equiv \left.\frac{\partial \mathbf{R}_r}{\partial \mathbf{u}_r}\right|_i \in \mathbb{R}^{m\times n}, \tag{14}$$

and define a forcing vector $\mathbf{b}_i \equiv \mathbf{A}_i \mathbf{u}_{ri} \in \mathbb{R}^m$, which is also computed at every training point. Note that the reduced state satisfies $\mathbf{A}_i \mathbf{u}_{ri} = \mathbf{b}_i$. Given a new parameter vector, $\boldsymbol{\mu}$, we interpolate the matrix and forcing vector using the iROM approach,

$$\mathbf{A}(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{A}_i, \qquad \mathbf{b}(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{b}_i, \tag{15}$$

We then solve the over-determined interpolated system, $\mathbf{A}\mathbf{u}_r = \mathbf{b}$, via a least-squares procedure, to obtain the reduced state at the desired point. Note, the matrix interpolation is presently performed entry-wise, although other interpolation techniques, coupled with matrix decomposition techniques, can also be considered.

The output calculation can also be interpolated in a similar fashion. At each training point, define the output linearization with respect to the reduced state as

$$\mathbf{C}_i \equiv \left.\frac{\partial \mathbf{J}}{\partial \mathbf{u}_r}\right|_i \in \mathbb{R}^{p\times n}. \tag{16}$$

By defining a forcing vector $\mathbf{d}_i \equiv \mathbf{J}_i - \mathbf{C}\mathbf{u}_{ri}$, also at each training point, we obtain the model

$$\mathbf{J} = \mathbf{C}\mathbf{u}_r + \mathbf{d}. \tag{17}$$

Given a new parameter vector, $\boldsymbol{\mu}$, we interpolate the output linearization and forcing vector using the iROM approach,

$$\mathbf{C}(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{C}_i, \qquad \mathbf{d}(\boldsymbol{\mu}) = \sum_{i=1}^{K} w_i \mathbf{d}_i, \tag{18}$$

From the reduced state, we then obtain the output vector directly by Eqn. 17.

Interpolation guarantees that we recover the original reduced states at the training points. The use of the reduced residual Jacobian matrix, which appears in the projection approach, brings in the physics of the problem. Note that no additional data beyond the training points are required, so that the method is not intrusive into the full-order code during the online stage. As the method shares characteristics of both projection and interpolation model reduction, we refer to it as ipROM.

IV. Results

In this section, we compare the four model reduction approaches: output iROM, state iROM, pROM, and ipROM, for fluid dynamics problems involving shape changes. The methods share the same training data, and in all cases, interpolation is performed using Lagrange polynomials. No hyper-reduction is used for the pROM approach, and derivatives with respect to the reduced state are calculated using a simple forward difference. The results are compared to exact high-fidelity CFD solutions at additional test points, i.e. parameter values not part of the training set.

A. One-Parameter Vehicle Shape

The first test case consists of a two-dimensional vehicle shape with an upper surface that has sharp corners. The single parameter in this case is the vertical position of the aft-most point on the upper surface, as illustrated in Figure 1. This parameter affects the angle of the preceding corner on the upper surface, and its variation can produce either attached or separated flow fields.

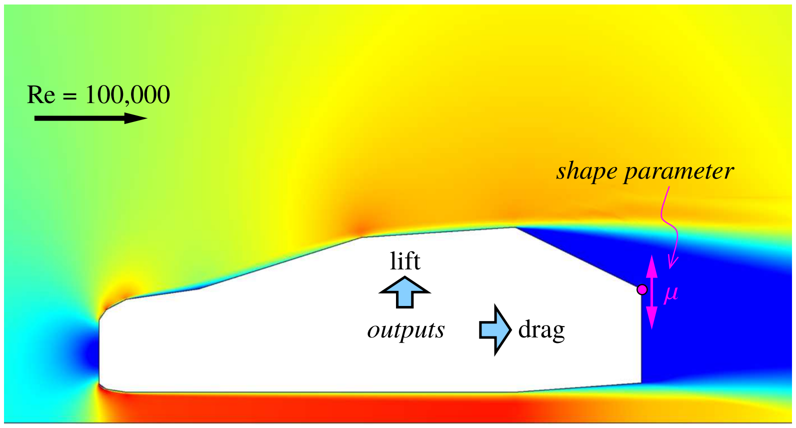
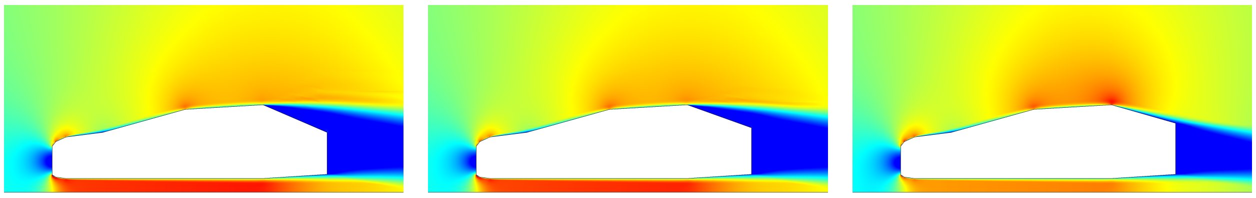


Fig. 1 Setup for the one-parameter vehicle shape case. Overlaid contours are of the velocity magnitude.

The spatial approximation order is $p = 3$, the Reynolds number based on the vehicle baseline height is 100,000, and the domain consists of a rectangular wind-tunnel with slip boundary conditions on the bottom and top surfaces. The full-state is specified at the tunnel inlet and exit, with horizontal flow at Mach number 0.1 and a ratio of turbulent to laminar free-stream viscosity of 3.

The computational mesh for this study consists of structured quadrilaterals split into 1488 triangles. Separate meshes are used for each point in parameter space, with the same number of elements and constructed parametrically with respect to the geometry: the vertical positions of the nodes change smoothly with respect to the parameter.

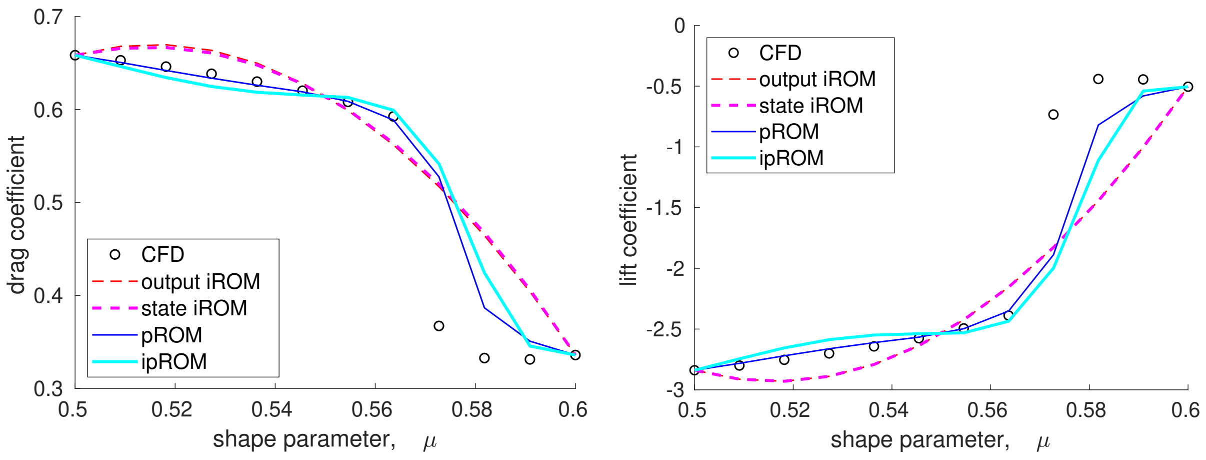


(a) $\mu = 0.5$ (b) $\mu = 0.55$ (c) $\mu = 0.6$

Fig. 2 One-parameter vehicle: state snapshots showing horizontal momentum.

Three training points are used, with a parameter range that spans separated and attached flow. Figure 2 shows the solutions at these training points. The snapshots are constructed for all state components, density, momentum, energy, taken together, and all basis vectors from the SVD are used, so that $n = K = 3$.

Starting with the training data, the four reduced models are constructed and used to predict the outputs and flow fields at intermediate parameter points. The outputs of interest are the lift and drag coefficients on the vehicle, computed by non-dimensionalizing based on the vehicle height. Figure 3 shows the results of the output comparison.



(a) Drag (b) Lift

Fig. 3 One-parameter vehicle: drag and lift coefficient prediction results.

We see that the output and state interpolated models (the two iROMs) perform similarly. That is, computing the outputs by interpolating the forces, or by interpolating the states and then computing the forces, yields nearly the same final output predictions in this case. By construction, the interpolated outputs go through the training data, but their accuracy in between the training data is not high. This is because the actual output does not vary as smoothly with respect to parameter changes: there is a sudden drop in drag and an increase in lift when the flow re-attaches around $\mu = 0.57$, and interpolation has no mechanism to pick up on this.

On the other hand, projection (pROM), does a better job at predicting the steep change associated with separation. With only three training points, the agreement is not perfect, but the overall shape and intermediate parameter values are better than with interpolation. This improved performance of pROM is due to its use of additional information, via residuals, of the full-order model.

Of particular interest here is the performance of ipROM, based on the interpolated systems and with no online link to the full-order model. We see from the data in Figure 3 that ipROM tracks pROM reasonably well, and is therefore closer to the CFD data compared with direct interpolation of the outputs or states. This improved performance can be attributed to the additional information used by ipROM: reduced residual Jacobians. However, in contrast to pROM, this information is interpolated rather than re-computed at the test points.

Figure 4 shows the flowfield reconstructions at one parameter value near the separated/attached demarcation. We see that the pROM and ipROM results closely match the CFD, whereas the state iROM predicts a "semi-separated" flow field that is an interpolant of attached and separated flows.

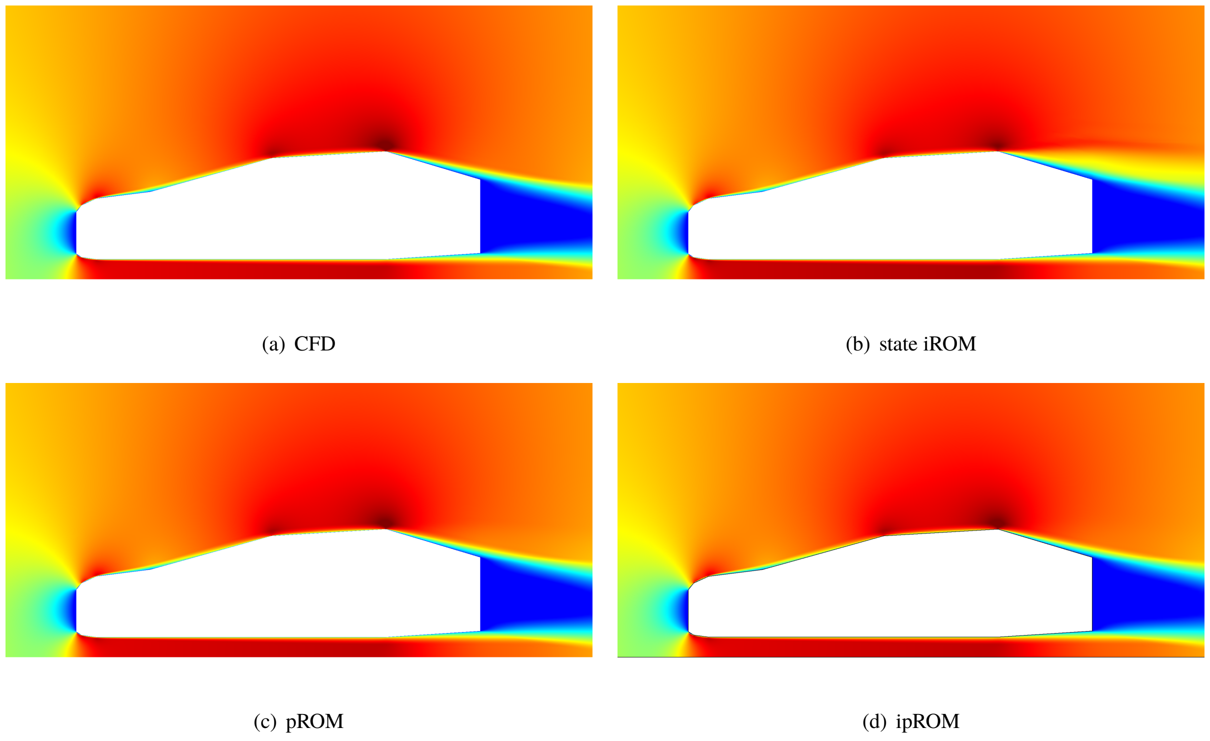


Fig. 4 One-parameter vehicle: horizontal momentum field reconstructions at $\mu = 0.591$.

B. Two-Parameter Vehicle Shape

The second test case builds on the first by considering two parameters in the vehicle study. These parameters are the vertical positions of the two corners of the cab portion of the vehicle, as illustrated in Figure 1. These parameters affect the cab corner angles and can lead to attached versus separated flow at those points.

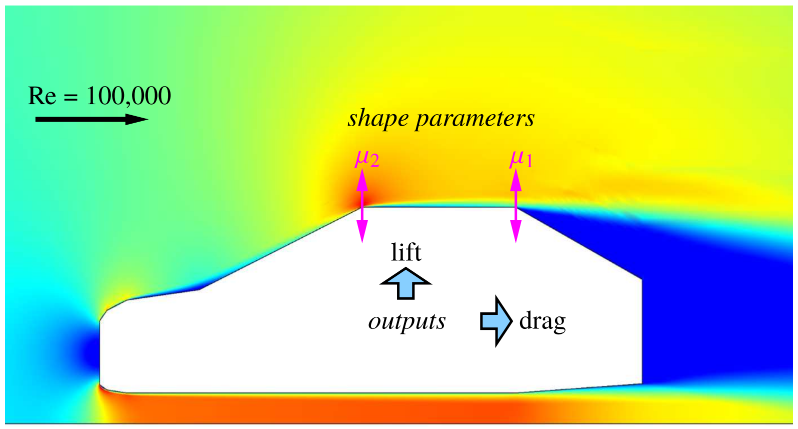
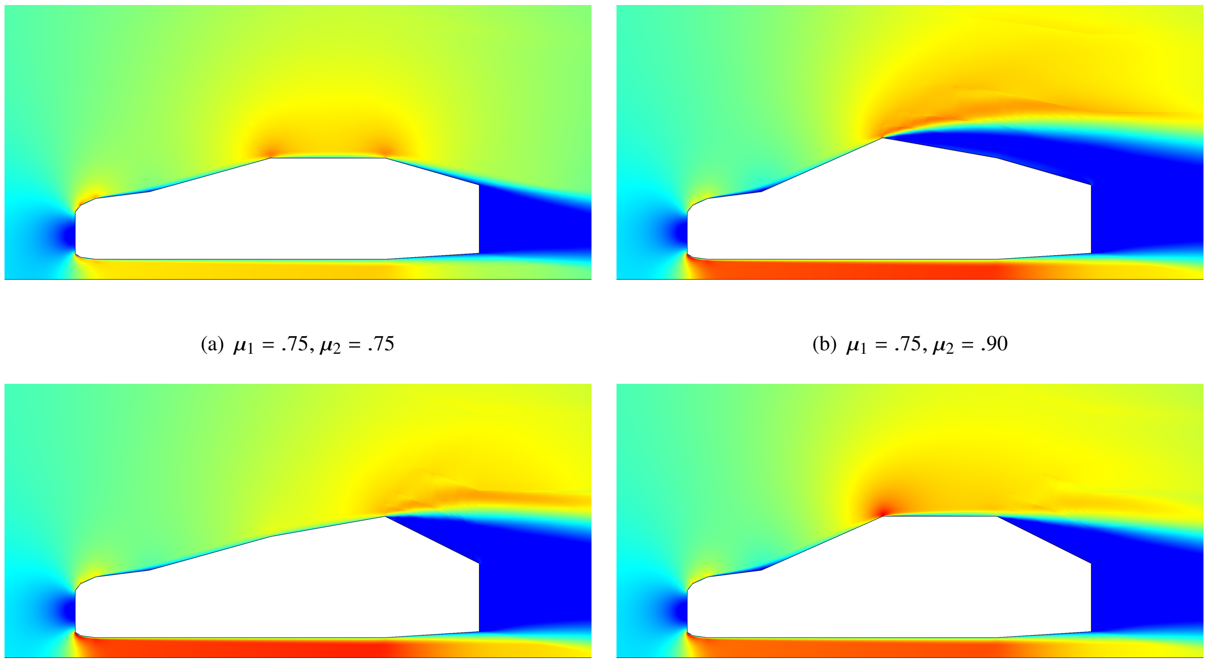


Fig. 5 Setup for the two-parameter vehicle shape case.

The flow parameters and mesh construction technique are the same as in the one-parameter vehicle case. Four training points are used: the corners of the rectangular parameter space. Figure 6 shows the states at these points, which includes flowfield on the upper surface that are (1) fully-attached, (2) separated at the first corner, and (3,4) separated at the second corner. The basis set is of the same dimension as the snapshots: $n = K = 4$.



(a) μ_1 = .75, μ_2 = .75

(b) μ_1 = .75, μ_2 = .90

(c) μ_1 = .90, μ_2 = .75

(d) μ_1 = .90, μ_2 = .90

Fig. 6 Two-parameter vehicle: state snapshots showing horizontal momentum.

Starting with the training data, the four reduced models are constructed and used to predict the lift and drag coefficient outputs and flowfields at intermediate parameter points. Figure 7 shows the results of the drag coefficient comparison. Direct state and output interpolation yield nearly the same results, and hence only the state iROM results are shown. As shown the iROM results are not very good, in that they completely miss the drag valley caused by attached flowfields that appear at intermediate parameter values. As expected, interpolating only one attached corner case with three separated ones will always show some separation.

The projection approach shows results that are closer to the CFD compared to direct interpolation. pROM identifies the area of reduced drag at the low parameter combinations, and the rapid increase in drag associated with large μ_1 but small μ_2. The agreement with CFD is not excellent, but it is based on only four training points in a rich parameter space.

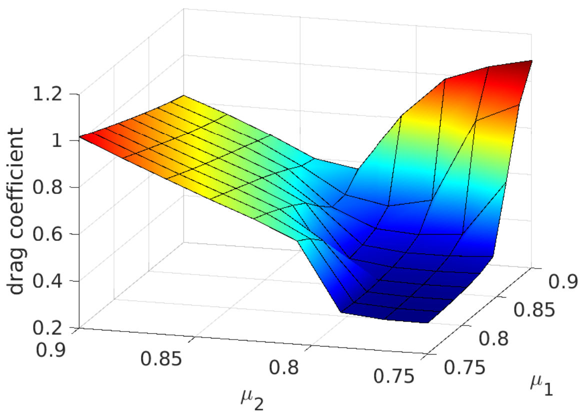
Finally, as in the one-parameter case, the ipROM approach introduced in this work yields results that align closely with the pROM. We therefore see that two-dimensional interpolation of the system matrices yields results that are similar to those obtained with re-evaluating the matrices at the test points.

Figure 8 shows the lift coefficient results for all of the methods. The observations here are similar as those for the drag: direct state/output interpolation misses the high lift plateau associated with attached flow, whereas pROM and ipROM pick up on it. The response surfaces of both pROM and ipROM are not in perfect agreement with the CFD, but they do show the right trends and are much richer than direct interpolation.

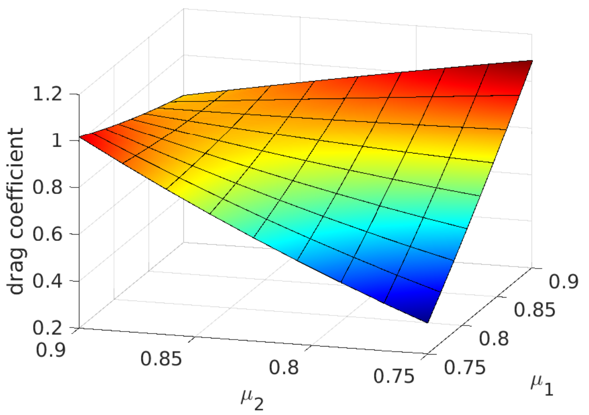
Figure 9 shows the flowfield reconstructions at one parameter point in which the CFD solution is attached. Representing this attached flow is difficult in this case, as the basis arises from snapshots in which three of the four solutions contain separated flow at one of the corners. However, the emphasis on the attached solution is evident in the pROM and ipROM cases, which are also similar to each other. The iROM field on the other hand is much further from physical, as it exhibits separation at both corners.

C. One-Parameter Airfoil Camber Change

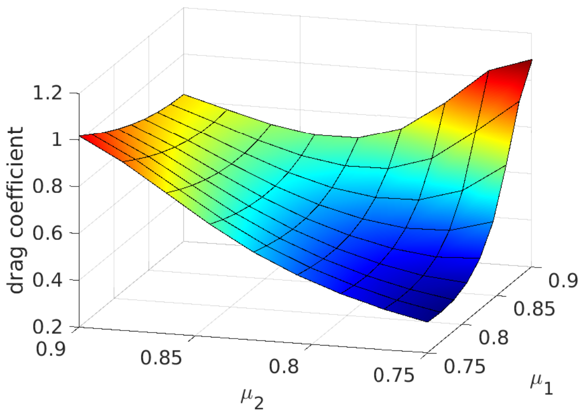
The final test case consists of a two-dimensional NACA X412 airfoil, where the maximum camber is the single parameter, μ, in the range -0.1 to 0.1. The spatial approximation order is $p = 3$, the Reynolds number based on the airfoil chord is 100,000, the angle of attack is $\alpha = 5^\circ$, and the free-stream Mach number is $M = 0.25$. Full-state boundary



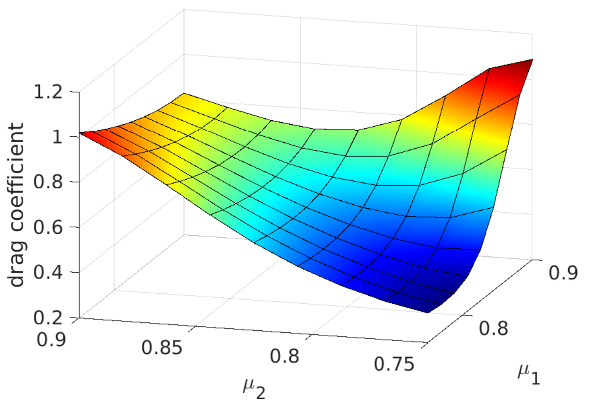
(a) CFD



(b) state iROM



(c) pROM



(d) ipROM

Fig. 7 Two-parameter vehicle: drag coefficient prediction results.

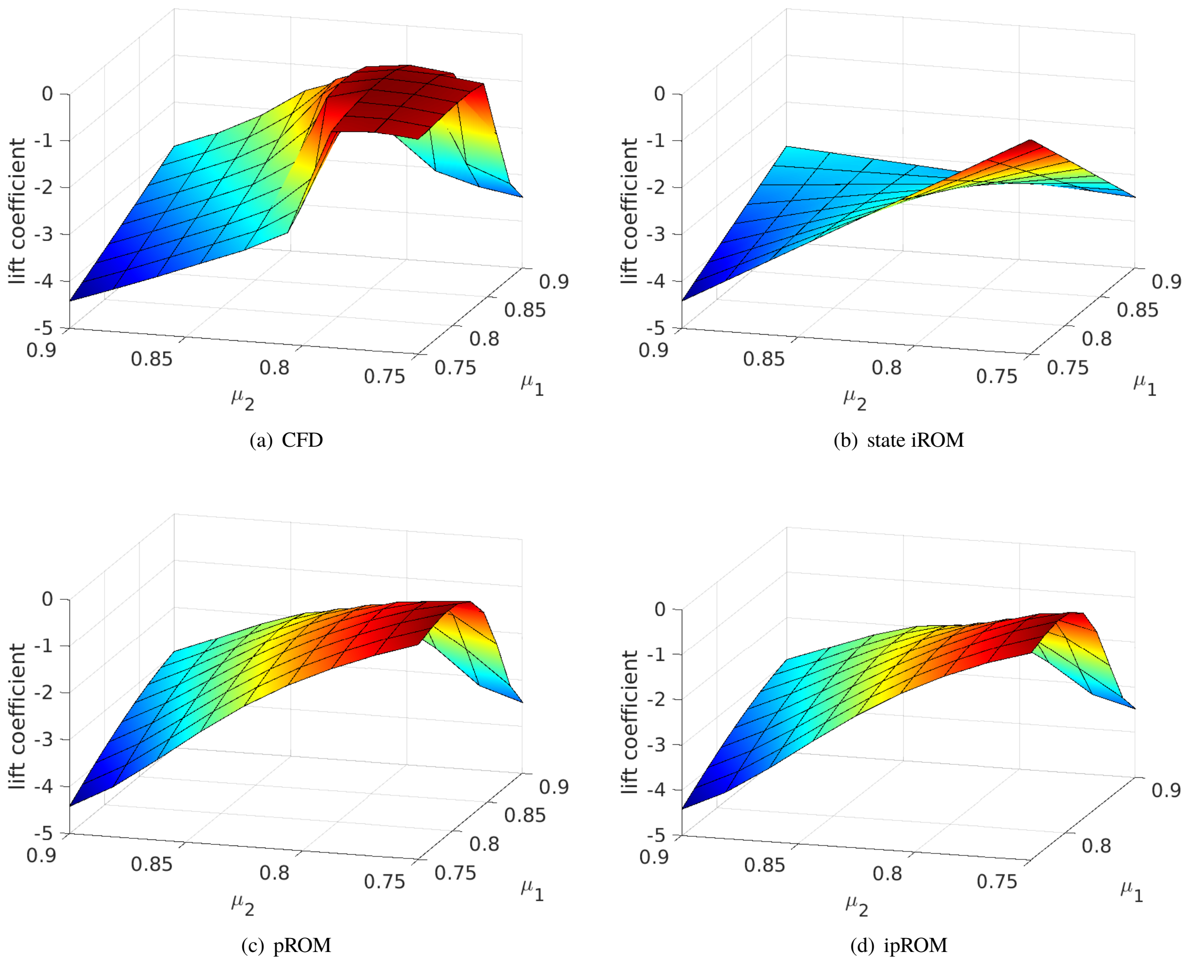
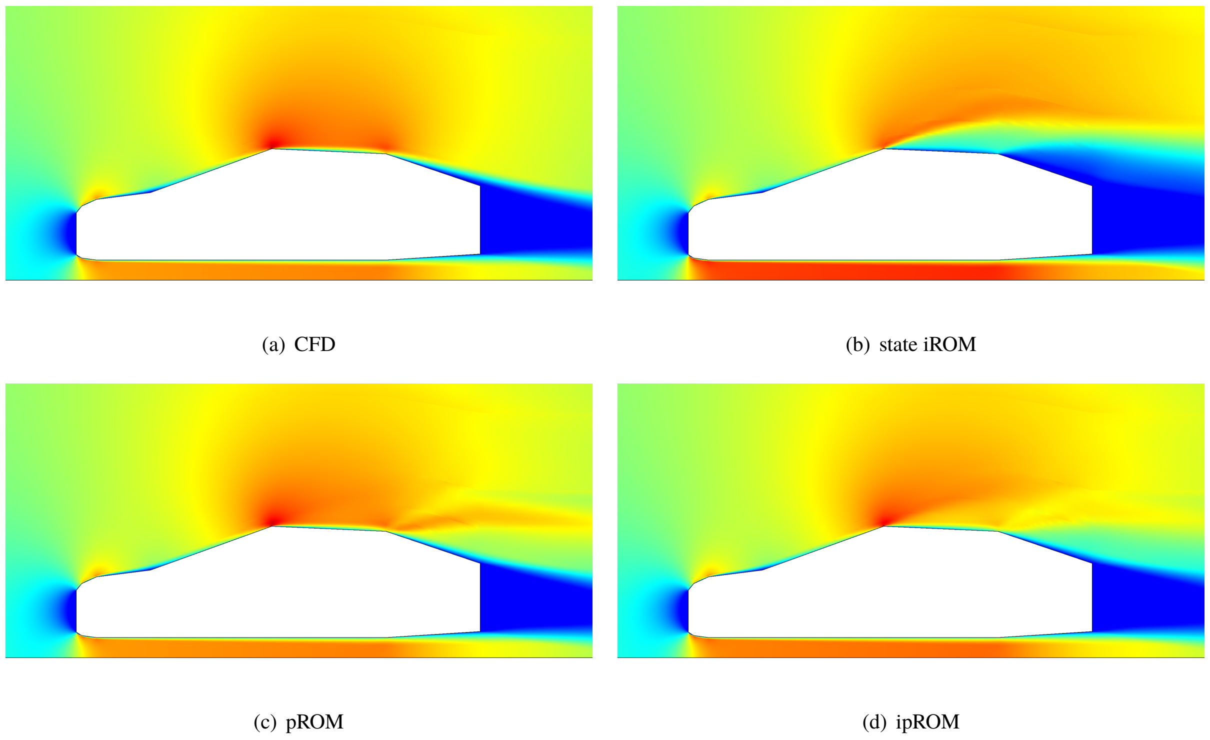


Fig. 8 Two-parameter vehicle: lift coefficient prediction results.


(a) CFD

(b) state iROM

(c) pROM

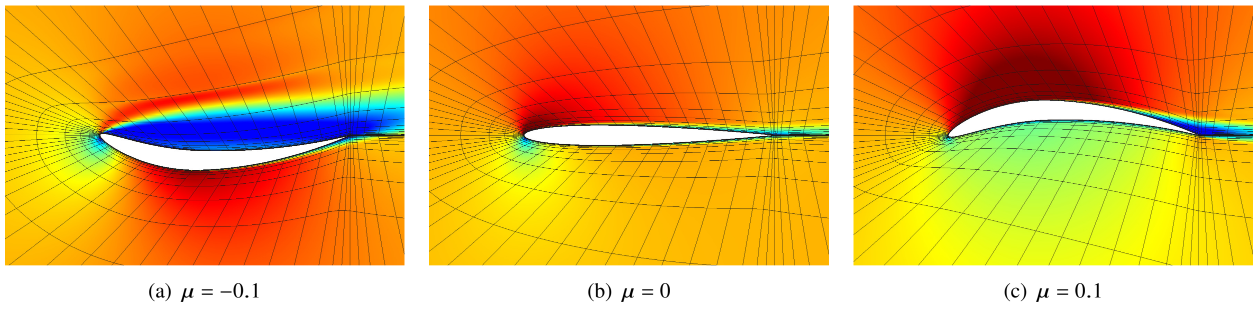
(d) ipROM

Fig. 9 Two-parameter vehicle: horizontal momentum field reconstructions at $\mu_1 = 0.79, \mu_2 = 0.81$.

conditions are specified on the domain boundary, approximately 100 chords away from the airfoil, and the ratio of turbulent to laminar free-stream viscosity is 3.

The computational mesh for this study consists of 1794 quadrilaterals. Separate meshes are used for each point in parameter space, with the same number of elements and constructed parametrically with respect to the geometry: the mesh node coordinates change smoothly with respect to the parameter.

Three training points are used, with a parameter range that spans separated and attached flow. Figure 10 shows the solutions at these training points. As in the previous examples, the snapshots are constructed for all state components, density, momentum, energy, taken together, and all basis vectors from the SVD are used, so that $n = K = 3$.


(a) $\mu = -0.1$

(b) $\mu = 0$

(c) $\mu = 0.1$

Fig. 10 One-parameter airfoil: state snapshots showing the Mach number.

Starting with the training data, the four reduced models are constructed and used to predict the drag coefficient and flow fields at intermediate parameter points. Figure 11 shows the results of the output comparison. The high-fidelity (CFD) results exhibit a drop in the drag coefficient when the maximum camber exceeds approximately -.03 chords. This is due to the flow becoming attached for this positive angle of attack case, when the camber is not overly negative. The state and output interpolation models do not accurately capture this drop, together with the slowly increasing drag

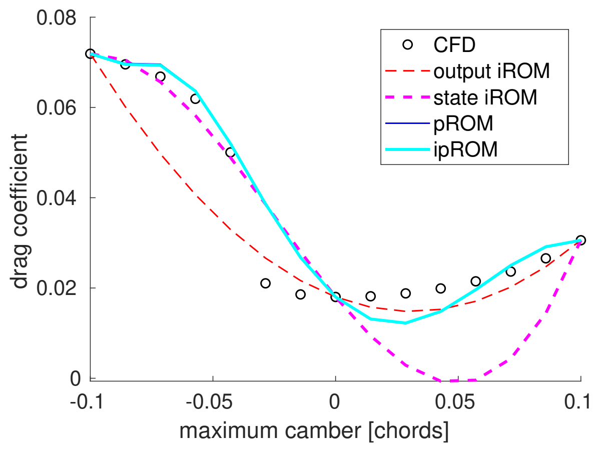


Fig. 11 One-parameter airfoil: drag coefficient prediction results.

coefficient afterwards, using only the three training points. The projection and ipROM models, which exhibit nearly identical behavior, better match the CFD results. As noted previously, ipROM has the advantage of not requiring any additional residual or Jacobian evaluations after training, as the reduced matrices are interpolated.

Figure 12 shows the flowfield reconstructions at one parameter value in the attached regime. We see that the pROM and ipROM results closely match the CFD, whereas the state iROM predicts suction-side velocities that are too large and a less-prominent wake. Both of these lead to an under-prediction of the drag.

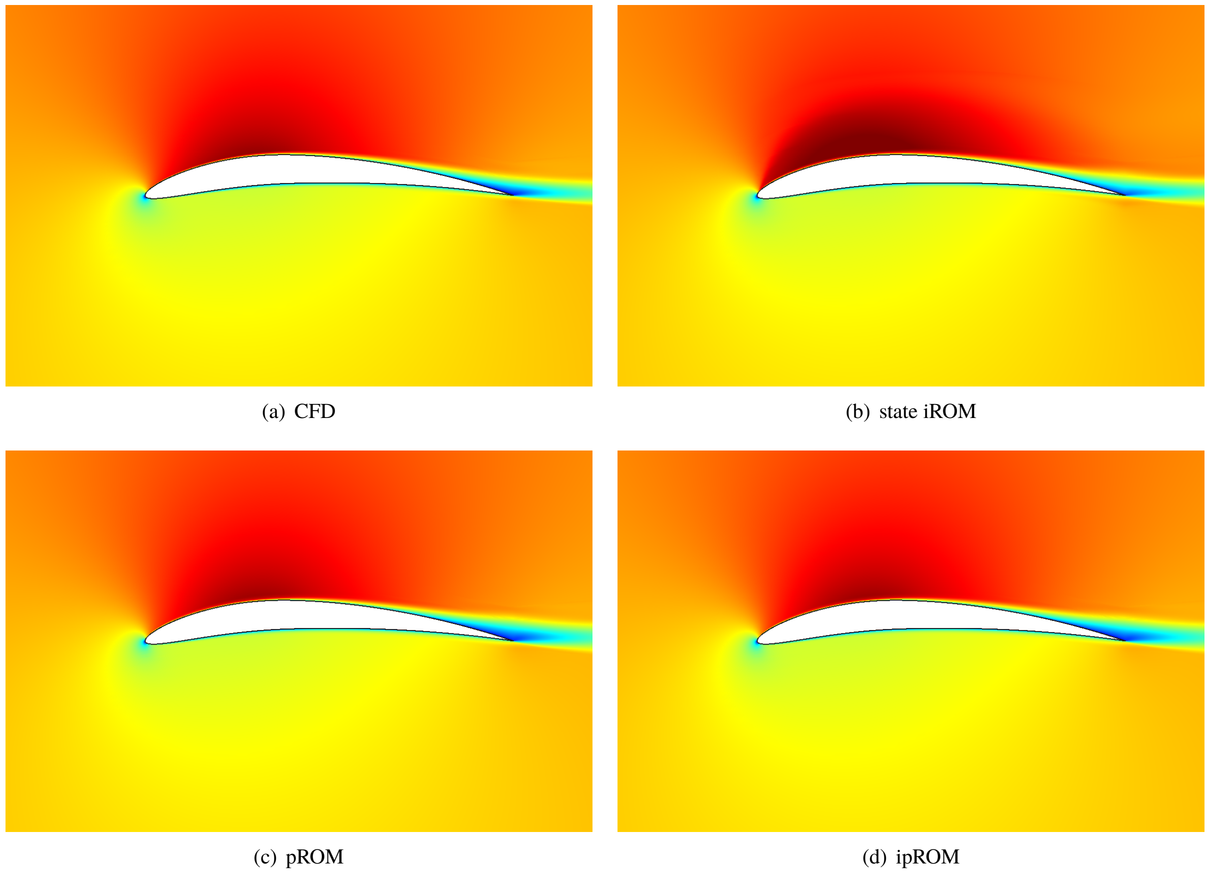


Fig. 12 One-parameter airfoil: x-momentum field reconstructions at $\mu = 0.071$.

V. Conclusions

This paper presents a new technique for efficiently constructing reduced models through interpolation of system matrices in parameter space. The method is more accurate than direct interpolation of states or quantities of interest, as it retains the “physics” of the problem through the use of residual Jacobian matrices. As such, it is closer to projection-based approaches, which also use reduced systems of equations constructed at the desired parameter points. However, whereas standard projection methods rely on a link to the full-order model to obtain this information in the online stage, the proposed ipROM approach interpolates it from the training data. This removes some of the implementation barriers of projection, particularly for large, complex cases. In addition, ipROM does not require construction of the geometries or meshes associated with the parameters of interest, which would be required to obtain residual information in projection. The results for the cases studied demonstrate that such system interpolation yields results that are much closer to projection than to direct state or output interpolation. Future studies will address more shape parameters, unstructured training points,more complex, three-dimensional flowfields, and unsteady simulations.

References

[1] Chung, H.-S., and Alonso, J. J., “Using Gradients to Construct Response Surface Models for High-Dimensional Design Optimization Problems,” AIAA Paper 2001-0922, 2001.

[2] Chung, H.-S., and Alonso, J. J., “Using Gradients to Construct Cokriging Approximation Models for High-Dimensional Design Optimization Problems,” AIAA Paper 2002-0317, 2002.

[3] Laurenceau, J., and Sagaut, P., “Building Efficient Response Surfaces of Aerodynamic Functions with Kriging and Cokriging,” *AIAA Journal*, Vol. 46, No. 2, 2008, pp. 498–507.

[4] Benner, P., Gugercin, S., and Willcox, K., “A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems,” *SIAM Review*, Vol. 57, No. 4, 2015, pp. 483–531.

[5] LeGresley, P. A., and Alonso, J. J., “Airfoil design optimization using reduced order models based on proper orthogonal decomposition,” AIAA Paper 2000–2545, 2000.

[6] Cohen, K., Siegel, S., and McLaughlin, T., “Sensor placement based on proper orthogonal decomposition modeling of a cylinder wake,” *33rd AIAA Fluid Dynamics Conference, Orlando, AIAA*, Vol. 4259, 2003, p. 2003.

[7] Hall, K., Thomas, J. P., and Dowell, E. H., “Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows,” *AIAA journal*, Vol. 38, No. 10, 2000, pp. 1853–1862.

[8] Kunisch, K., and Volkwein, S., “Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition,” *Journal of optimization theory and applications*, Vol. 102, No. 2, 1999, pp. 345–371.

[9] Dowell, E., Hall, K., Thomas, J., Florea, R., Epureanu, B., and Heeg, J., “Reduced order models in unsteady aerodynamics,” 1999.

[10] 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. https://doi.org/10.1002/nme.3050, URL http://dx.doi.org/10.1002/nme.3050.

[11] Amsallem, D., Zahr, M. J., and Farhat, C., “Nonlinear Model Order Reduction Based on Local Reduced-order bases,” *International Journal for Numerical Methods in Engineering*, Vol. 92, No. 10, 2012, pp. 891–916. https://doi.org/10.1002/nme.4371, URL http://dx.doi.org/10.1002/nme.4371.

[12] Chaturantabut, S., and Sorensen, D. C., “Nonlinear Model Reduction via Discrete Empirical Interpolation,” *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[13] Galbally, D., Fidkowski, K., Willcox, K., and Ghattas, O., “Nonlinear Model Reduction for Uncertainty Quantification in Large-Scale Inverse Problems,” *International Journal for Numerical Methods in Engineering*, Vol. 81, 2009, pp. 1581–1608. https://doi.org/10.1002/nme.2746.

[14] Nguyen, N. C., Patera, A. T., and Peraire, J., “A ’Best Points’ Interpolation Method for Efficient Approximation of Parmetrized Functions,” *International Journal for Numerical Methods in Engineering*, Vol. DOI: 10.1002/nme.2086, 2007.

[15] Lee, K., and Carlberg, K. T., “Model Reduction of Dynamical Systems on Nonlinear Manifolds Using Deep Convolutional Autoencoders,” *arXiv preprint arXiv:1812.08373v1*, 2018.

[16] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," , 2017.

[17] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations," *arXiv preprint arXiv:1711.10566*, 2017.

[18] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K., "Projection-based model reduction: Formulations for physics-based machine learning," *Computers and Fluids*, Vol. 179, 2019, pp. 704–717. https://doi.org/10.1016/j.compfluid.2018.07.021.

[19] Wang, Q., Hesthaven, J. S., and Ray, D., "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem," *Journal of Computational Physics*, Vol. 384, 2019, pp. 289 – 307. https://doi.org/https://doi.org/10.1016/j.jcp.2019.01.031.

[20] Allmaras, S., Johnson, F., and Spalart, P., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.

[21] Reed, W., and Hill, T., "Triangular Mesh Methods for the Neutron Transport Equation," Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[22] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[23] Bassi, F., and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by B. Cockburn, G. Karniadakis, and C.-W. Shu, Springer, Berlin, 2000, pp. 197–208.

[24] Bertram, A., Othmer, C., and Zimmerman, R., "Towards Real-time Vehicle Aerodynamic Design via Multi-fidelity Data-driven Reduced Order Modeling," AIAA Paper 2018-0916, 2018.

[25] Wang, Q., Medeiros, R. R., Cesnik, C. E., Fidkowski, K. J., Brezillon, J., and Bleecke, H. M., "Techniques for Improving Neural Network-based Aerodynamics Reduced-order Models," AIAA Paper 2019–1849, 2019. https://doi.org/10.2514/6.2019-1849.

[26] Willcox, K., "Unsteady Flow Sensing and Estimation Using the Gappy Proper Orthogonal Decomposition," *Computers and Fluids*, Vol. 35, 2006, pp. 208–226.

[27] Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing Point Estimation in Models Described by Proper Orthogonal Decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.