Author version

Published as:

van Schie, S. P., & Hwang, J. T. (2026). Computational framework for high-fidelity aerodynamic shape optimization using weighted POD. In AIAA AVIATION 2026 Forum (Paper No. AIAA 2026-4782). https://doi.org/10.2514/6.2026-4782

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/vanschie2026computational/

```
@inproceedings{vanschie2026computational,
  author = {Sebastiaan P. van Schie and John T. Hwang},
  title = {Computational Framework for High-Fidelity Aerodynamic Shape Optimization Using
    Weighted POD},
  booktitle = {AIAA AVIATION 2026 Forum},
  year = {2026},
  doi = {10.2514/6.2026-4782},
  url = {https://doi.org/10.2514/6.2026-4782}
}
```

Computational Framework For High-Fidelity Aerodynamic Shape Optimization Using Weighted POD

Sebastiaan P. C. van Schie*, John T. Hwang†
University of California San Diego, 9500 Gilman Dr., La Jolla, CA 92093

While high-fidelity computational models feature superior accuracy over lower-fidelity alternatives, their computational cost can be larger by several orders of magnitude. This presents a bottleneck for their use in design optimization, where many model queries are often necessary. In this work we present an initial implementation of a computational framework that is designed to solve high-fidelity aerodynamic shape optimization problems in an efficient way, by leveraging Proper Orthogonal Decomposition (POD) to minimize the number of required high-fidelity model queries. We introduce the various models that are used in this framework. This includes our Discontinuous Galerkin (DG) compressible Euler model, the Free-Form Deformation-based shape morphing approach with linear elasticity-based mesh morphing, and both a standard approach to compute POD bases, as well as a parametric approach based on the idea of snapshot weighting. We use the framework to solve a shape optimization problem of the Simple Transonic Wing benchmark. While the mesh resolution is limited, both POD approaches demonstrate median force and moment coefficient predictions within $O(0.1\%)$ of the high-dimensional compressible Euler model, while being an order of magnitude faster than the high-dimensional model. Future steps include the use of higher-order DG bases, using larger meshes and implementing hyper-reduction for further speedups.

I. Introduction

Under Moore's Law, the availability of computing power has increased by several orders of magnitude over the past two decades. This has allowed for more sophisticated, costly, higher-fidelity computational models to be leveraged for the analysis of physical systems. Design optimization algorithms query computational models multiple times in order to evaluate the performance of physical systems under design modifications. As is well known in the optimization community, the number of model evaluations that are required to solve an optimization problem depends on the number of design parameters. This number scales linearly with the number of design parameters when a gradient-based optimization method is used, and quadratically with gradient-free optimization methods [1]. Despite enjoying the benefits of multiple decades of Moore's Law, the cost of high-fidelity models is still significant, even if only a handful of evaluations are required. For a given computational budget a trade-off needs to be made between model fidelity and scope of the optimization problem. Conversely, using a high-fidelity model for a high-dimensional optimization problem comes at significant computational costs, requiring extensive use of high-performance computing hardware.

We intend to circumvent this limitation through the use of model reduction techniques in the optimization loop, in order to minimize the number of necessary high-fidelity model evaluations by instead using data-driven approximate models wherever possible. We aim to speed up the use of high-fidelity models for large-scale aerodynamic shape optimization in this way. In this work we present a significant step towards this goal. Namely, an initial implementation of a computational framework that uses the compressible Euler equations and a linear elasticity-based mesh motion model to solve aerodynamic shape optimization problems with parallel computing capabilities. Moreover, we show the benefits in simulation time that can be obtained when using Proper Orthogonal Decomposition (POD) as approximate, reduced model.

The rest of this work is structured as follows. We start in Section II by giving a brief overview of a Discontinuous Galerkin (DG)-based compressible Euler model implementation, followed in Section III by the elasticity-based mesh motion approach that is used. Next, Section IV gives a brief description of the embedding of the Euler and mesh motion models in an optimization framework. This is followed up in Section V by a description of the POD methods that have

*Ph.D. Candidate, Department of Mechanical and Aerospace Engineering, AIAA student member
†Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA senior member

been implemented; these include a standard POD approach, as well as a parametric POD approach that uses snapshot weighting to compute local reduced bases. Lastly, Section VI covers the application of our computational framework to a shape optimization problem of the Simple Transonic Wing [2], and discusses the optimization results and the efficacy of the aforementioned POD approaches.

II. High-Fidelity Aerodynamics

We use the Discontinuous Galerkin method to discretize the non-dimensionalized compressible Euler equations. Let $\Omega \subset \mathbb{R}^d$ be a spatial domain with $d \in \{2, 3\}$. We introduce the solution vector $\mathbf{u} = (\rho, \rho\mathbf{u}_v, \rho E)^\top : \Omega \to \mathbb{R}^+ \times \mathbb{R}^d \times \mathbb{R}^+$, where ρ is the fluid density, $\mathbf{u}_v$ the fluid velocity, $\rho\mathbf{u}_v$ momentum, and ρE the total energy density. The compressible Euler equations are then formulated as

$$\nabla \cdot \mathcal{F}(\mathbf{u}) = \mathbf{0} \quad \text{in } \Omega, \tag{1}$$

where the flux $\mathcal{F}$ is

$$\mathcal{F}(\mathbf{u}) = \begin{pmatrix} \rho\mathbf{u}_v \\ \rho\mathbf{u}_v \otimes \mathbf{u}_v + p\mathbf{I} \\ (\rho E + p)\mathbf{u}_v \end{pmatrix}. \tag{2}$$

Here $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity tensor. To turn (1) into a closed system of equations we supply an equation of state, which links the pressure p to the other variables through

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}||\mathbf{u}_v||^2\right), \tag{3}$$

with γ the specific heat ratio; for air $\gamma = 1.4$. When deriving the weak form corresponding to (1), DG methods apply integration by parts, resulting in

$$-\int_\Omega \mathcal{F}(\mathbf{u}) : \nabla \boldsymbol{v}\, d\Omega + \int_{\partial\Omega} (\boldsymbol{n} \cdot \mathcal{F}(\mathbf{u})) \cdot \boldsymbol{v}\, d\partial\Omega = \mathbf{0} \quad \text{in } \Omega, \tag{4}$$

with $\boldsymbol{v}$ a vector-valued test function. When discretizing (4) we cover domain Ω with a large number of cells $\{\Omega_k\}_{k=1}^{n_k}$ such that $\Omega = \bigcup_{k=1}^{n_k} \overline{\Omega}_k$ and $\Omega_i \cap \Omega_j =$ for $i \neq j$. We choose a separate p^{th}-order function basis on each cell Ω_k; this results in a multi-valued solution on internal cell boundaries. To resolve this we replace $\boldsymbol{n} \cdot \mathcal{F}(\mathbf{u})$ in (4) with the flux function $\mathbb{F}$ on cell boundaries $\partial\Omega_k$. Such flux functions depend on the local solutions on either side of the cell boundary. One attractive property of DG methods is that the flux functions can be chosen to be dissipative, thereby stabilizing the numerical model. In this work we use the Rusanov flux function. On $\partial\Omega_k$, this flux function is defined as

$$\mathbb{F}_k(\boldsymbol{u}) = \frac{1}{2}\left[\mathcal{F}(\boldsymbol{u}^-) \cdot \boldsymbol{n}_k + \mathcal{F}(\boldsymbol{u}^+) \cdot \boldsymbol{n}_k\right] + \frac{\alpha(\boldsymbol{u}^+, \boldsymbol{u}^-)}{2}(\boldsymbol{u}^+ - \boldsymbol{u}^-). \tag{5}$$

The flux function is integrated over the boundary of each cell Ω_k. The outward-pointing normal vector $\boldsymbol{n}_k$ is defined on this same boundary. For each part of the boundary $\partial\Omega_k$, $\boldsymbol{u}^-$ and $\boldsymbol{u}^+$ denote the solution vectors as evaluated through the function basis on cell Ω_k and the neighboring cell respectively. The parameter α calculates the maximum wave speed on either side of the boundary, such that

$$\begin{aligned} \omega(\boldsymbol{u}) &= \max\left\{|\boldsymbol{u}_v \cdot \boldsymbol{n}_k - c|, |\boldsymbol{u}_v \cdot \boldsymbol{n}_k|, |\boldsymbol{u}_v \cdot \boldsymbol{n}_k + c|\right\} \\ \alpha(\boldsymbol{u}^+, \boldsymbol{u}^-) &= \max\left\{\omega\left(\boldsymbol{u}^+\right), \omega\left(\boldsymbol{u}^-\right)\right\}, \end{aligned} \tag{6}$$

where $c = \sqrt{\frac{\gamma p}{\rho}}$ is the local speed of sound on either side of the boundary. The local maximum wave speed is then multiplied in (5) with the difference between the local solutions on either side of the cell boundary. The result acts as a local damping term. This is comparable to the Lax–Friedrichs flux function, but with locally varying viscous damping instead of a single global value for α.

Since the boundary integral in (4) is defined over internal and external cell boundaries, Dirichlet boundary conditions such as inlet, outlet and slip- & no-slip wall conditions, can be imposed as natural boundary conditions. We enforce these boundary conditions similar to the aforementioned flux function. Suppose $\boldsymbol{u}^{\text{ext}}$ is the solution on the domain boundary

that contains the boundary conditions and any unconstrained local solution components. The external boundary flux function $\mathbb{F}^{\text{ext}}$ is

$$\mathbb{F}_k^{\text{ext}} = \mathcal{F}(\boldsymbol{u}) \cdot \boldsymbol{n}_k + \alpha(\boldsymbol{u}, \boldsymbol{u}^{\text{ext}}) \left(\boldsymbol{u} - \boldsymbol{u}^{\text{ext}}\right). \tag{7}$$

This flux term is used in (4) for the parts of the boundary integral that coincide with the domain boundaries. For a more thorough and general treatment of DG methods, the interested reader is referred to [3]. The specific DG implementation used here is similar to that reported in [4].

To further stabilize the model, we implement the Symmetric Interior Penalty Galerkin (SIPG) method, similar to [4, 5]. We define a scaling parameter $C_{\text{SIPG}} \in \mathbb{R}^+$ and use it to define the SIPG viscosity

$$\mu_{\text{SIPG}} = C_{\text{SIPG}}\, \omega(\boldsymbol{u}) h, \tag{8}$$

where h is the local cell diameter and $\omega(\boldsymbol{u})$ is again the maximum wave speed on each cell. We define the SIPG stabilization as a mix of volumetric and cell boundary integrals,

$$\begin{aligned}\int_\Omega \mu_{\text{sipg}} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, d\Omega \; - \; & \int_{\partial\Omega_k} \frac{1}{2} \left(\left(\mu_{\text{sipg}} \nabla \boldsymbol{u}\right)^+ + \left(\mu_{\text{sipg}} \nabla \boldsymbol{u}\right)^- \right) \cdot \left(\boldsymbol{v}^+ - \boldsymbol{v}^-\right) \otimes \boldsymbol{n}_k \, d\partial\Omega_k \\ & - \int_{\partial\Omega_k} \frac{1}{2} \left(\left(\mu_{\text{sipg}} \nabla \boldsymbol{v}\right)^+ + \left(\mu_{\text{sipg}} \nabla \boldsymbol{v}\right)^- \right) \cdot \left(\boldsymbol{u}^+ - \boldsymbol{u}^-\right) \otimes \boldsymbol{n}_k \, d\partial\Omega_k \\ & + 10(p+1)^2 \int_{\partial\Omega_k} \frac{\mu_{\text{sipg}}^+ + \mu_{\text{sipg}}^-}{h^+ + h^-} \left(\boldsymbol{u}^+ - \boldsymbol{u}^-\right) \cdot \left(\boldsymbol{v}^+ - \boldsymbol{v}^-\right) \, d\partial\Omega_k. \end{aligned} \tag{9}$$

Here, as before, p is the degree of the DG basis. The first three terms are not active for $p = 0$, since in that case $\nabla \boldsymbol{u}$ and $\nabla \boldsymbol{v}$ are zero. The terms in (9) are added to the DG formulation shown in (4).

The result of the discretization procedure is a residual vector $\boldsymbol{R}(\boldsymbol{u}_h, \boldsymbol{\mu}) \in \mathbb{R}^n$, which depends on numerical solution $\boldsymbol{u}_h$, and an input parameter vector $\boldsymbol{\mu}$ that will be detailed in Section III. Permissible solutions to (1) correspond to $\boldsymbol{R}(\boldsymbol{u}_h, \boldsymbol{\mu}) = \boldsymbol{0}$. These solutions are computed through repeated application of the Newton-Raphson method,

$$\frac{\partial \boldsymbol{R}(\boldsymbol{u}^{(j)}, \boldsymbol{\mu})}{\partial \boldsymbol{u}^{(j)}} \Delta \boldsymbol{u}^{(j+1)} = -\boldsymbol{R}(\boldsymbol{u}^{(j)}, \boldsymbol{\mu}), \tag{10}$$

where j is the Newton-Raphson iteration index, such that

$$\boldsymbol{u}^{(j+1)} = \boldsymbol{u}^{(j)} + \theta^{(j+1)} \Delta \boldsymbol{u}^{(j+1)}. \tag{11}$$

Here $\theta^{(j+1)} \in [0, 1]$ is an under-relaxation factor. The convergence of the Newton-Raphson method can be monitored by looking at the residual norm $||\boldsymbol{R}(\boldsymbol{u}_h, \boldsymbol{\mu})||_2$. The under-relaxation factor $\theta^{(j+1)}$ in (11) is determined in two steps. We define the acceleration parameter $\tau \in \mathbb{R}^+$. Then,

$$\theta^{(j+1)} = \min\left(1, \sqrt{2\tau / ||\Delta \boldsymbol{u}^{(j+1)}||_2}\right). \tag{12}$$

The initial under-relaxation factor $\theta^{(j+1)}$ is then used in (11). If we find that any of the mass density, energy density or pressure variables are negative or zero, we halve $\theta^{(j+1)}$ and recompute (11). This check is applied iteratively until a physically admissible updated solution $\boldsymbol{u}^{(j+1)}$ is found. In practice this under-relaxation strategy can slow down convergence speed, especially during the initial Newton-Raphson iterations. To address this we use warm-starting, where the most recent converged solution is used as initial solution guess.

FEniCS(x) [6] is used to implement the DG methods outlined above. Whereas in earlier work [7] we used the `dolfin_dg` toolkit [4], we have since moved to a more lightweight, standalone implementation of the compressible Euler model discussed above. This has the added benefit of allowing us to use quadrilateral and hexahedral grids in two and three dimensions respectively, which allows for the use of structured grids. Due to the modular nature of FEniCS, the numerical flux function can be swapped out for any other flux function without any significant code changes. The code supports MPI parallelization through FEniCS's built-in PETSc support and the `petsc4py` toolbox [8], thereby

allowing for efficient simulations with high numbers of degrees of freedom.

We use the iterative Flexible Generalized Minimal Residual method (FGMRES) solution method through PETSc, with the domain decomposition-based Additive Schwarz Method (ASM) as a preconditioner. To speed up ASM we use Incomplete LU (ILU) decomposition to solve the decomposed problems. We gradually tighten the convergence tolerance of the iterative solver as the norm of the residual vector drops. This since we do not gain anything from solving the initial Newton-Raphson iterations to a tight tolerance. Their solution updates are relatively large, and thus changes that are several orders of magnitude smaller than $||\Delta \boldsymbol{u}^{(j+1)}||_2$ do not affect overall convergence much.

III. Mesh Motion

The compressible Euler model that is covered in Section II is used to solve shape optimization problems. Since our intention is to use gradient-based optimization methods, solving shape optimization problems requires the use of a mesh morphing approach. We use Free-Form Deformation (FFD) to move the boundary nodes of the body under consideration after the creation of an initial mesh. These boundary nodes are too numerous to use as shape parameters directly in an optimization loop. FFD instead immerses the body under consideration in a block, spanned by a Cartesian grid of control points. The movements of the control points are then linked to movements of the boundary nodes through a B-spline deformation map. This gives a natural way to apply shape deformations to the body, while retaining full control over the dimensionality of the FFD block through the definition of the set of control points. We note that FFD thus decouples the fidelity of the discretization of the body's boundary from the body deformations.

The movements of the body's boundary nodes need to be smoothly propagated to the interior nodes of the mesh, such that the quality of the deformed mesh does not negatively impact the robustness of the compressible Euler model. Various methods exist to achieve this, among which are inverse-distance weighting [9, 10] and radial basis function interpolation [11]. In this work we instead solve a linear elasticity problem to obtain smoothly deformed meshes. The movements of the body's boundary nodes are applied to a linear elasticity model as Dirichlet boundary conditions, implemented through FEniCS. Similarly, homogeneous Dirichlet boundary conditions are used on the other domain boundaries. The weak form of the equations of linear elasticity with vector-valued test function $\boldsymbol{v}_d$ is

$$\int_\Omega \sigma(\boldsymbol{d}) \,:\, \epsilon(\boldsymbol{v}_d)\, d\Omega = 0, \tag{13}$$

with σ the stress tensor and ϵ the symmetric strain tensor. Both depend on displacement vector $\boldsymbol{d}$, such that

$$\begin{aligned} \sigma(\boldsymbol{d}) &= \lambda\, \mathrm{tr}\,(\epsilon(\boldsymbol{d}))\, I + 2\mu\, \epsilon(\boldsymbol{d}) \\ \epsilon(\boldsymbol{d}) &= \frac{1}{2}\left(\nabla \boldsymbol{d} + (\nabla \boldsymbol{d})^T\right), \end{aligned} \tag{14}$$

where tr denotes the trace operator and I the identity tensor. Also included are the Lamé material parameters λ and μ. Equation (13) is complemented by the aforementioned Dirichlet boundary conditions, which provide a forcing term to the weak form. The resulting deformation vector is then used to move the internal mesh nodes. In this work we use a continuous linear Lagrange basis for the displacement and test function. We use the iterative Conjugate Gradient (CG) method to ensure scalability of the model, and pair it with the Geometric Agglomerated Algebraic MultiGrid (GAMG) preconditioner, both available through PETSc.

IV. Design Optimization

The general form of Partial Differential Equation (PDE)-constrained optimization problems is

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{u}, \boldsymbol{\mu}) \in \mathbb{R} \\ \text{with respect to} \quad & \boldsymbol{u}, \boldsymbol{\mu} \\ \text{subject to} \quad & \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\mu}) = \boldsymbol{0} \\ & \boldsymbol{c}(\boldsymbol{u}, \boldsymbol{\mu}) \geq \boldsymbol{0}, \end{aligned} \tag{15}$$

where f is an objective function, $\boldsymbol{R}$ is the PDE residual vector, which depends on design parameter vector $\boldsymbol{\mu}$ and PDE state $\boldsymbol{u}$, and $\boldsymbol{c}$ is a vector of (in)equality constraints which any feasible solution needs to satisfy. In this work we use a

reduced-space optimization approach, where we treat $\boldsymbol{u}$ as an implicit variable defined by $\boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\mu}) = \boldsymbol{0}$ instead of a design parameter vector. The textbook [1] gives a more elaborate overview of single- and multidisciplinary design optimization.

The compressible Euler model of Section II and the mesh motion model of Section III are both wrapped into custom operations of the Computational Systems Design Language (CSDL) [12]. This domain-specific language is designed for multidisciplinary design optimization (MDO), and constructs a computational graph to handle gradient-based optimization. Its custom operation interface handles similar to OpenMDAO, with each custom operation specifying its forward and reverse-mode derivatives. This is where using FEniCS gives some advantages, since its Unified Form Language (UFL) [13] implementation allows for automatic differentiation of any weak form. We use the modOpt library [14] to use the graph that is constructed by CSDL to carry out optimization. While in this work we focus on gradient-free optimization with the COBYLA optimizer, modOpt supports a wide variety of gradient-free and gradient-based optimization algorithms.

Figure 1 shows the current decision process during the optimization loop. After the optimizer picks a new design parameter vector $\boldsymbol{\mu}$, first the mesh motion algorithm is used to compute the displacements of the mesh nodes under the influence of $\boldsymbol{\mu}$. The resulting deformed mesh is deemed suitable for analysis if the minimal Jacobian determinant is positive. If this is the case, the compressible Euler model is solved on the deformed mesh. If not, the objective function is assigned a large value and the optimizer is made to pick a new design vector for the next iteration. Convergence of the compressible Euler simulation hinges on a maximum criterion for the norm of residual vector $||\boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\mu})||$, and on global positivity of the mass and energy density. If these conditions are not met the objective is again assigned a large value. If the Euler simulation has converged, we compute the various output quantities such as force and moment coefficients. These are then fed back to the optimizer, after which it chooses a new design parameter vector and the next iteration starts.

V. Proper Orthogonal Decomposition

As is well-documented throughout the relevant literature, the cost of solving shape optimization problems depends on various choices: While the choice of optimization problem specification, optimization approach and algorithm dictate how many model evaluations are necessary to reach a local or global optimal point, the choice for a particular physics model, or set of models, determines how costly each model evaluation will be. The fidelity of the model tends to drive its computational cost: Low-fidelity models are often quick to evaluate, but have limits to their accuracy and possible scope. On the other hand, high-fidelity models are complex enough to reasonably approximate realistic behavior, while requiring high-performance computing capabilities for anything other than a small number of model evaluations. In this work we address the cost of our compressible Euler flow model by applying a data-driven model reduction technique.

Due to its nonlinear nature, the main computational cost of the compressible Euler model is incurred through the repeated use of (10), and consists of two parts:

1) Constructing Jacobian matrix $\frac{\partial \boldsymbol{R}(\boldsymbol{u}^{(j)}, \boldsymbol{\mu})}{\partial \boldsymbol{u}^{(j)}} \in \mathbb{R}^{n \times n}$ and residual vector $\boldsymbol{R}(\boldsymbol{u}^{(j)}, \boldsymbol{\mu})$,
2) solving the resulting linearized system.

Here we focus on the second of these two parts, which we address through the use of Proper Orthogonal Decomposition. POD is a data-driven Reduced-Order Modeling (ROM) approach that identifies dominant modes in a set of given numerical solutions. These modes can then be used to project the linear system (10) to the low-dimensional subspace spanned by the dominant modes. Suppose we gather the numerical solutions $\boldsymbol{u}_h^{(j)}$ of the compressible Euler model, that correspond to design parameter vectors $\left\{\boldsymbol{\mu}^{(j)}\right\}_{j=1}^{n_s}$, in a single matrix. We refer to this matrix as the snapshot matrix,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{u}^{(1)}(\boldsymbol{\mu}^{(1)}) & \boldsymbol{u}^{(2)}(\boldsymbol{\mu}^{(2)}) & \dots & \boldsymbol{u}^{(n_s)}(\boldsymbol{\mu}^{(n_s)}) \end{bmatrix}. \tag{16}$$

We compute the Singular Value Decomposition (SVD) of $\boldsymbol{X}$,

$$\boldsymbol{X} = \boldsymbol{\Psi}\boldsymbol{\Sigma}\boldsymbol{\Xi}^T = \begin{bmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \dots \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_1^T \\ \boldsymbol{\xi}_2^T \\ \vdots \end{bmatrix}, \tag{17}$$

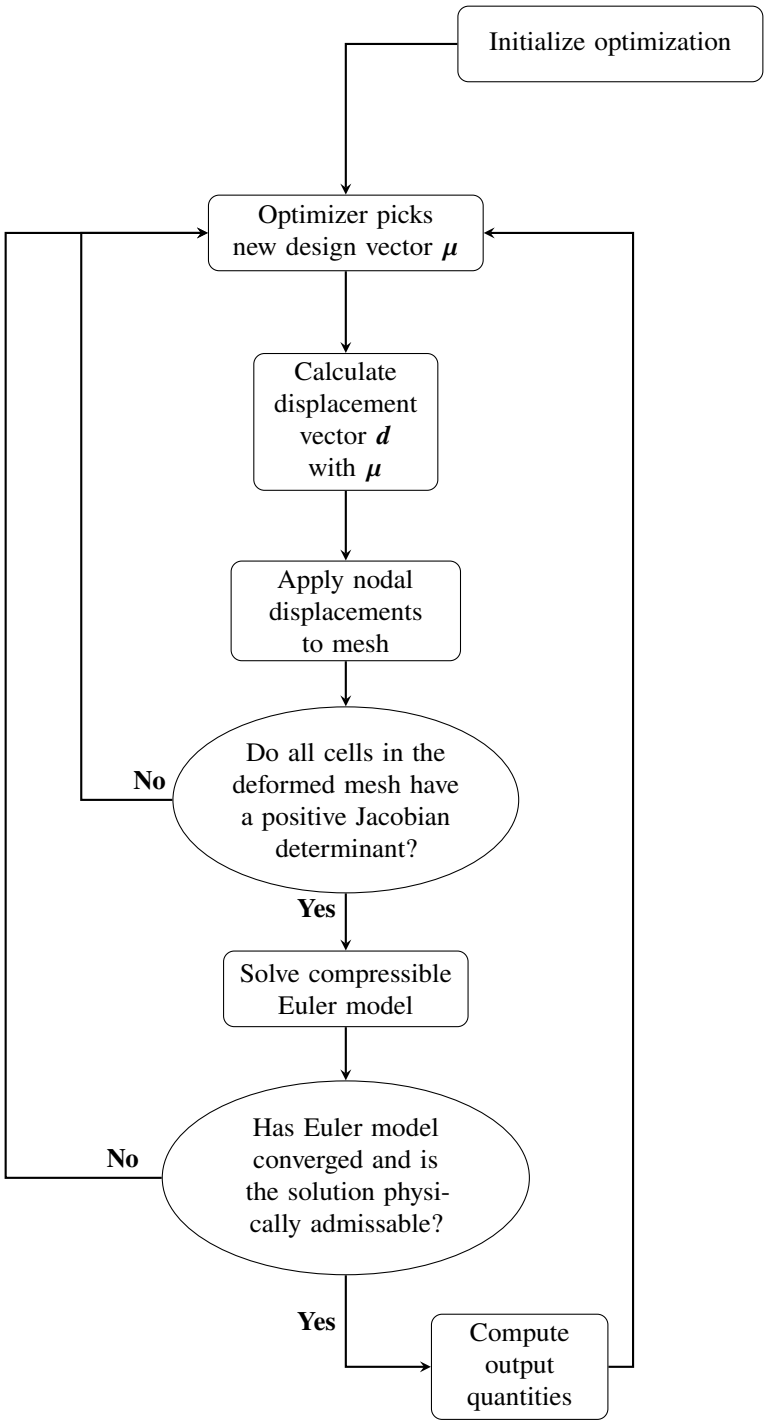


Fig. 1 Decision process and flowchart of the optimization loop

where $\boldsymbol{\Psi}$ spans the column space of $\boldsymbol{X}$. The columns of $\boldsymbol{\Psi}$ correspond to the dominant modes of the column space of $\boldsymbol{X}$ and are ordered in decreasing importance. Hence a straightforward choice for the best n_r-dimensional subspace $\boldsymbol{\Phi}$ of the column space of $\boldsymbol{X}$ is to take the first n_r columns of $\boldsymbol{\Psi}$,

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \dots & \boldsymbol{\psi}_{n_r} \end{bmatrix}. \tag{18}$$

We use $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n_r}$ to project (10) to an n_r-dimensional subspace by left-multiplying with $\boldsymbol{\Phi}^T$ and applying the projection $\boldsymbol{\Phi}\boldsymbol{u}_r^{(j+1)} = \boldsymbol{u}^{(j+1)}$,

$$\boldsymbol{\Phi}^T \frac{\partial \boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{u}_r^{(j+1)}, \boldsymbol{\mu})}{\partial \boldsymbol{u}^{(j)}} \boldsymbol{\Phi}\Delta\boldsymbol{u}_r^{(j+1)} = -\boldsymbol{\Phi}^T \boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{u}_r^{(j+1)}, \boldsymbol{\mu}), \tag{19}$$

where $\boldsymbol{\Phi}\Delta\boldsymbol{u}_r^{(j+1)} = \Delta\boldsymbol{u}^{(j+1)}$. Equation (19) consists of an n_r-dimensional linear system; for $n_r \ll n$ this means that solving (19) is considerably cheaper than solving the full-dimensional system (10).

This leaves unaddressed the question of how accurately $\boldsymbol{\Phi}\boldsymbol{u}_r^{(j+1)}$ approximates $\boldsymbol{u}^{(j+1)}$. In [15] we showed that the upper bound of this POD approximation error depends on the distance between the current parameter vector $\boldsymbol{\mu}$ and the parameter vectors $\{\boldsymbol{\mu}^{(j)}\}_{j=1}^{n_s}$ that correspond to the numerical solutions contained in $\boldsymbol{X}$. As also shown in [15], this upper bound can be reduced by applying snapshot weights w_j to the columns of $\boldsymbol{X}$, with higher weights applied to those columns for which $\left\|\boldsymbol{\mu} - \boldsymbol{\mu}^{(j)}\right\|_2$ is smallest. This results in a POD approach that is more robust and accurate for parametric models, for only a minor increase in computational cost. We populate the diagonal of a matrix with the weights w_j, remove any column for which $w_j = 0$, and define the result as weight matrix $\boldsymbol{W}$. We are interested in calculating the SVD of the weighted snapshot matrix $\boldsymbol{XW}$. This can be done efficiently by considering that

$$\boldsymbol{XW} = \boldsymbol{\Psi}\boldsymbol{\Sigma}\boldsymbol{\Xi}^T\boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{D}, \tag{20}$$

and noting that for $\boldsymbol{X} \in \mathbb{R}^{n \times n_s}$ with $n_s \ll n$, the matrix product $\boldsymbol{D}$ is at most of size $n_s \times n_s$. Its SVD is therefore cheap to compute, and is written as

$$\boldsymbol{D} = \boldsymbol{\Psi}_{\boldsymbol{D}}\boldsymbol{\Sigma}_{\boldsymbol{D}}\boldsymbol{\Xi}_{\boldsymbol{D}}^T. \tag{21}$$

Since the product of two unitary matrices is also unitary, the SVD of the weighted snapshot matrix $\boldsymbol{XW}$ follows from the SVDs of $\boldsymbol{X}$ and $\boldsymbol{D}$:

$$\begin{aligned} \boldsymbol{\Psi}_{\boldsymbol{XW}} &= \boldsymbol{\Psi}\boldsymbol{\Psi}_{\boldsymbol{D}} \\ \boldsymbol{\Sigma}_{\boldsymbol{XW}} &= \boldsymbol{\Sigma}_{\boldsymbol{D}} \\ \boldsymbol{\Xi}_{\boldsymbol{XW}} &= \boldsymbol{\Xi}_{\boldsymbol{D}}. \end{aligned} \tag{22}$$

As in (18), the POD basis corresponding to the weighted snapshot matrix consists of the first n_r columns of $\boldsymbol{\Psi}_{\boldsymbol{XW}}$. Let $\hat{\boldsymbol{\mu}}$ be the parameter vector for which we are determining the weighted POD basis. Similar to [7, 15], the weights w_j are defined as a weighted combination of the minimum and maximum snapshot distance. We let

$$\begin{aligned} \delta_{\min} &= \min\{\left\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{(j)}\right\|_2\}_{j=1}^{n_s} \\ \delta_{\max} &= \max\{\left\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{(j)}\right\|_2\}_{j=1}^{n_s} \\ \hat{\delta} &= c\delta_{\min} + (1-c)\delta_{\max}, \end{aligned} \tag{23}$$

where $\hat{\delta}$ is the maximum distance at which snapshot weights are nonzero. This distance depends on the hyperparameter $c \in (0, 1]$. In this work we use $c = 0.6$. Any monotonically non-increasing function can then be used to assign nonzero weights to the snapshots. Following [15], we use a cubic polynomial. Other approaches, such as inverse-distance weighting or radial basis functions, can be used as well. The final step is to normalize the snapshot weights such that they form a partition of unity, such that

$$\sum_{j=1}^{n_s} w_j = 1. \tag{24}$$

In this work we consider two POD approaches. The first is outlined above, with different weights assigned to each snapshot, depending on the current parameter vector of interest, $\hat{\boldsymbol{\mu}}$. We refer to this approach as the weighted POD

approach. The second approach is referred to as the global POD approach, which uses the SVD of the snapshot matrix $\boldsymbol{X}$ directly to determine a non-weighted POD basis, through (18). This basis is independent of $\hat{\boldsymbol{\mu}}$ and can be considered as a special case of the weighted POD approach, wherein each snapshot is assigned a weight of $1/n_s$. After computing an n_r-dimensional POD solution, we project it back to the full-dimensional solution space with $\boldsymbol{\Phi}\boldsymbol{u}_r^{(j+1)} = \tilde{\boldsymbol{u}}^{(j+1)}$. In this work we accept the POD solution as being accurate enough when $||\boldsymbol{R}(\tilde{\boldsymbol{u}}^{(j+1)}, \boldsymbol{\mu}||_2 < 2.5 \cdot 10^{-3}$. If this criterion is not met, we instead run the original full-dimensional compressible Euler model, and append the obtained solution to the snapshot matrix $\boldsymbol{X}$.

VI. Results

We use the methods and models that are described in previous sections to solve a simple three-dimensional wing shape optimization. We use the Simple Transonic Wing, a wing geometry benchmark that was originally proposed by Gray and Martins in [2] for use in aero-elastic optimization codes. Since then various studies have presented aero-elastic optimization results using this geometry [16–19], including works that make direct comparisons between optimization codes from multiple institutions [20]. Along its entire span the wing cross-section consists of an RAE2822 airfoil. Figure 2 shows the Simple Transonic Wing with the surface mesh visible. This is also the mesh that was used for analysis and optimization. It is the coarsest of the publicly-available structured meshes that are available online* for this benchmark, consisting of $1.8 \cdot 10^5$ hexahedral cells arranged in a semi-sphere around the wing geometry. In all of the analyses shown in this work we use 0^{th}-order basis functions, leading to approximately $9 \cdot 10^5$ degrees of freedom in the three-dimensional compressible Euler model, with an SIPG parameter of $C_{\text{SIPG}} = 10^{-3}$. We note that the meshes for the Simple Transonic Wing benchmark were created with Navier-Stokes fluid models in mind. As such they contain layers of small cells near the wing surface. While these are not necessary for Euler models, we decided to stick with this family of meshes due to them being an established benchmark.

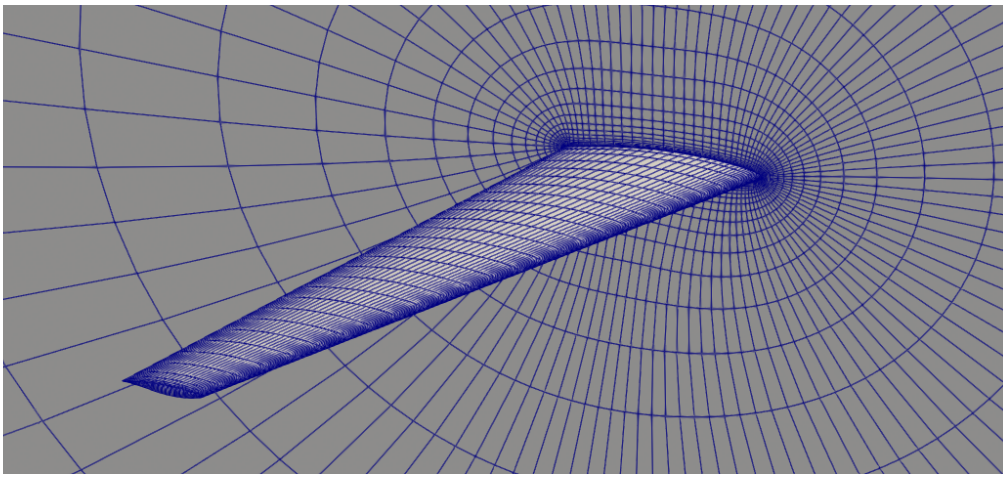


Fig. 2 Simple Transonic Wing geometry with superimposed surface mesh

Three approaches are used to solve the same optimization problem:

1) The compressible Euler model of Section II, referred to as the Full-Order Model (FOM).
2) The non-weighted POD approach of Section V applied to the compressible Euler model, referred to as the global POD approach.
3) The weighted POD approach of Section V applied to the compressible Euler model, referred to as the weighted POD approach.

To define the optimization problem we impose a Mach number of 0.8 and an angle of attack of 2 degrees. We focus on lift-constrained drag minimization: C_D is the objective function, while C_L has to be equal to or greater than the lift coefficient $C_L \approx 0.285$ of the undeformed wing. The drag coefficient of the undeformed wing design is $C_D = 0.0743$. The design parameters are the vertical (z) displacements of the FFD control points; these are allowed to move in a range of $[-0.1, 0.1]$ meters. We define five spanwise (y) locations, including wing root and tip, at which a Cartesian grid of

*https://mdobenchmarks.github.io/MDOAeroelasticBenchmark/

Table 1 Optimal designs obtained for each method considered, with $n_r = 10$ for the POD methods; shown are the force and moment coefficients of the optimized design as predicted by each POD approach and those predicted by the FOM for the same design, and the number of model calls, split into POD and FOM solves

Model	POD			FOM			Model calls	POD	FOM
	C_D	C_L	C_M	C_D	C_L	C_M			
FOM	—	—	—	0.0550	0.2858	0.4007	226	—	226
Global POD	0.0603	0.2875	0.4452	0.0603	0.2867	0.4454	344	222	122
Weighted POD	0.0649	0.2819	0.4825	0.0649	0.2826	0.4825	265	194	71

Table 2 Optimal designs obtained for each method considered, with $n_r = 20$ for the POD methods; shown are the force and moment coefficients of the optimized design as predicted by each POD approach and those predicted by the FOM for the same design, and the number of model calls, split into POD and FOM solves

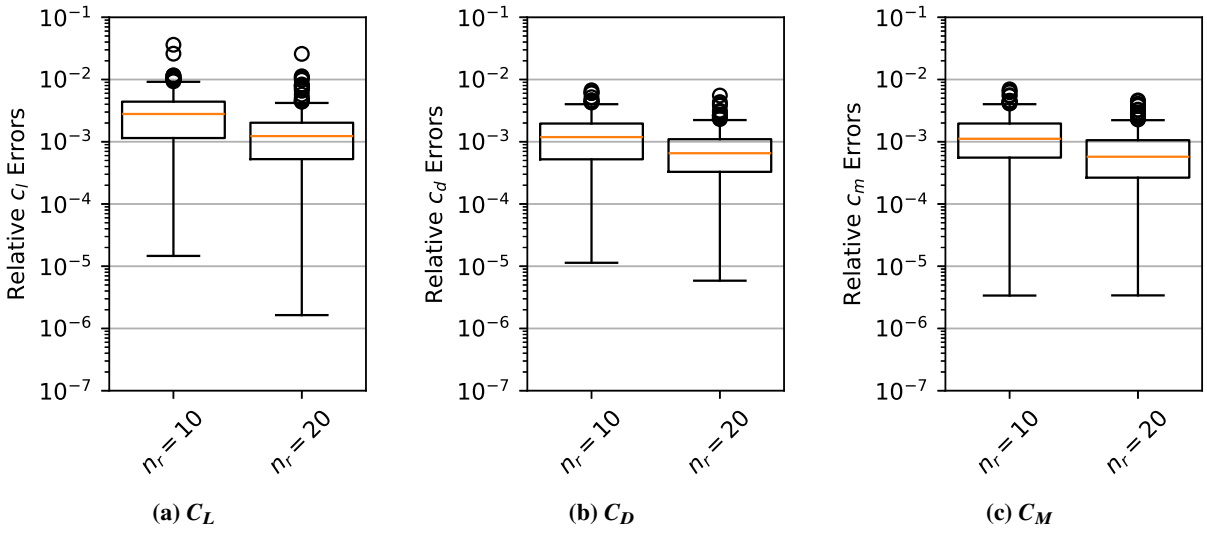
Model	POD			FOM			Model calls	POD	FOM
	C_D	C_L	C_M	C_D	C_L	C_M			
FOM	—	—	—	0.0550	0.2858	0.4007	226	—	226
Global POD	0.0602	0.2858	0.4496	0.0602	0.2866	0.4497	335	289	46
Weighted POD	0.0673	0.2838	0.4985	0.0673	0.2839	0.4985	222	170	52

four control points in chordwise (x) direction and two control points in the vertical (z) direction are defined. This FFD block of $4\times5\times2$ control points envelopes the wing, making for a total of 40 design parameters. For the FFD deformation map we use quadratic B-splines in x and y, and linear B-splines in z to calculate the movements of wing boundary nodes under the effect of control point displacements. We note that this shape optimization problem thus does not affect the wing planform parameters, and is more akin to simultaneous airfoil optimization at multiple spanwise stations. Both of the aforementioned POD approaches are constructed with reduced basis sizes up to $n_r = 10$ or $n_r = 20$. As mentioned before, POD solutions are accepted when $||\boldsymbol{R}(\tilde{\boldsymbol{u}}^{(j+1)}, \boldsymbol{\mu})||_2 < 2.5 \cdot 10^{-3}$, and rejected otherwise. All simulations were carried out on the Triton Shared Computing Cluster at the San Diego Supercomputer Center [21] with four MPI processes.

Table 1 shows the force and moment coefficients corresponding to the optimal designs for all three considered models, and how many model evaluations were necessary to reach these optimal designs. For the POD models the corresponding coefficient values as computed by the FOM are shown as well, along with the number of required POD and FOM evaluations. Moreover, a maximum reduced basis size of $n_r = 10$ is used for the POD models. One can see that the FOM finds the best-performing optimal design, followed by the global POD approach, and then the weighted POD approach. The POD and FOM coefficient predictions of the optimal designs are closely matched. This indicates that the COBYLA optimizer is likely to blame for the weighted POD method converging to a design that does not satisfy $C_L > 0.285$. Lastly, we see that the FOM converges with the fewest model evaluations, followed by the weighted and global POD approaches, in that order. Furthermore, the weighted POD approach requires significantly fewer model evaluations (71) than the global POD approach (122). Table 2 shows the same optimization outputs, now for $n_r = 20$. We see that the force and moment coefficient results of the optimized designs are similar to those shown in Table 1. With $n_r = 20$ both POD approaches require significantly less data, as they are able to use more POD modes in each simulation. The global POD approach now uses slightly less data than the weighted POD approach, but does require significantly more POD evaluations to reach its optimal design. We note that in all optimization runs shown here, the optimizer reduces the drag coefficient by reducing the wing thickness.

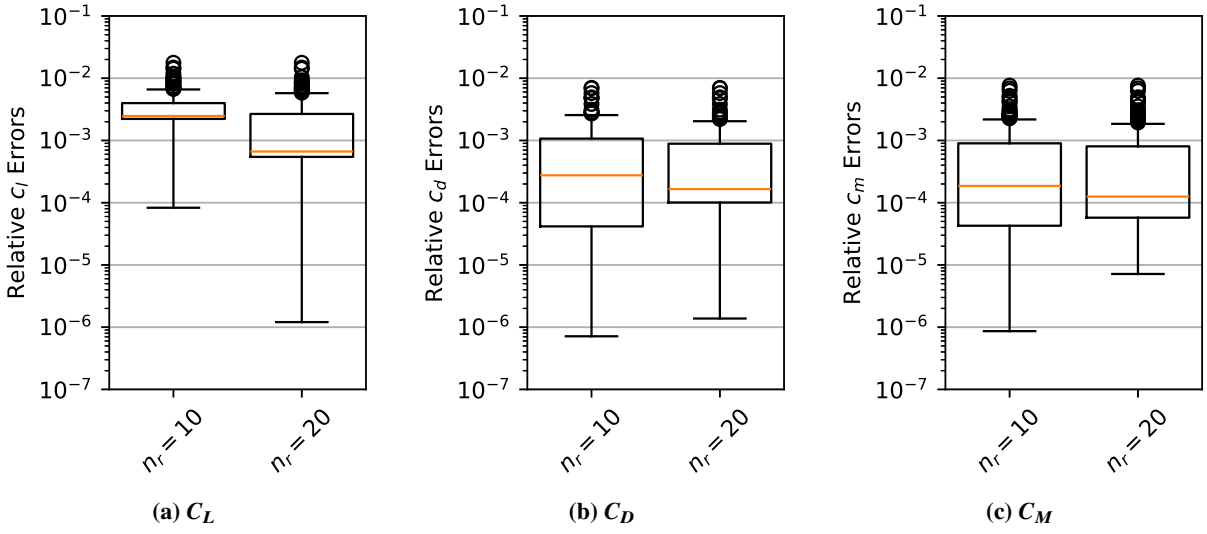
Next we look at the relative force and moment coefficient prediction errors. These are shown in Figure 3 for the global POD method and Figure 4 for the weighted POD approach. For each coefficient shown the weighted POD approach is more accurate than the global approach, in places up to an order of magnitude. However, even the global POD approach is quite accurate, with median force and moment coefficient predictions off by at most 0.1 to 0.2% of the FOM value. This high level of overall accuracy is at least in part likely due to the limited spatial resolution of the mesh that was used in these optimizations, coupled with the 0^{th}-order DG basis.

Weighted POD can also be seen to outperform global POD in Figure 5 and Figure 6, where the L_2 errors integrated over the entire domain and over only the wing surface are shown respectively. These L_2 errors are normalized with the corresponding L_2 norm of the FOM solution. Since all of the significant solution variation happens on the wing surface and directly around the wing, it makes sense that the domain-integrated relative L_2 errors are several orders of magnitude smaller than the corresponding L_2 errors when only integrating over the wing surface.



(a) C_L **(b) C_D** **(c) C_M**

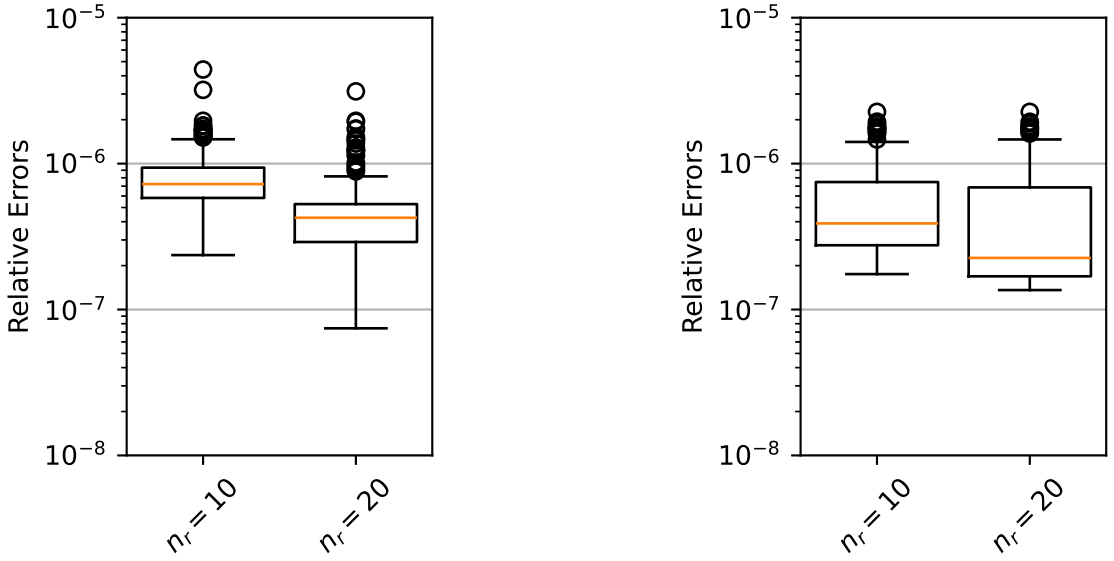
Fig. 3 Relative force and moment coefficient errors obtained with the global POD approach



(a) C_L **(b) C_D** **(c) C_M**

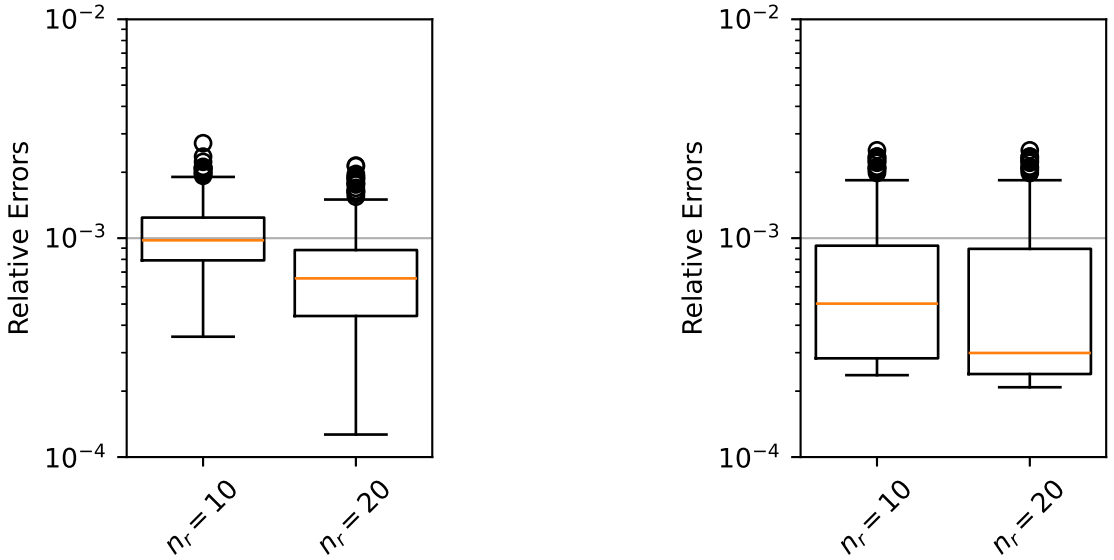
Fig. 4 Relative force and moment coefficient errors obtained with the weighted POD approach

Lastly, we look at the POD and FOM wall times evaluated at the same parameter vectors. These are shown in Figure 7. Right away we see that both POD methods are at least an order of magnitude faster than the corresponding FOM evaluations. This is much larger than we encountered in previous studies [7, 15] and is due to the use of 0^{th}-order DG basis functions in this work. This makes the assembly cost of the Jacobian matrix a much smaller part of the total cost of each Newton-Raphson iteration. We expect the speedup factor when using POD to decrease as the order of the DG basis is increased. From comparing the wall times of both POD approaches, we see that they are comparable,



(a) Global POD (b) Weighted POD

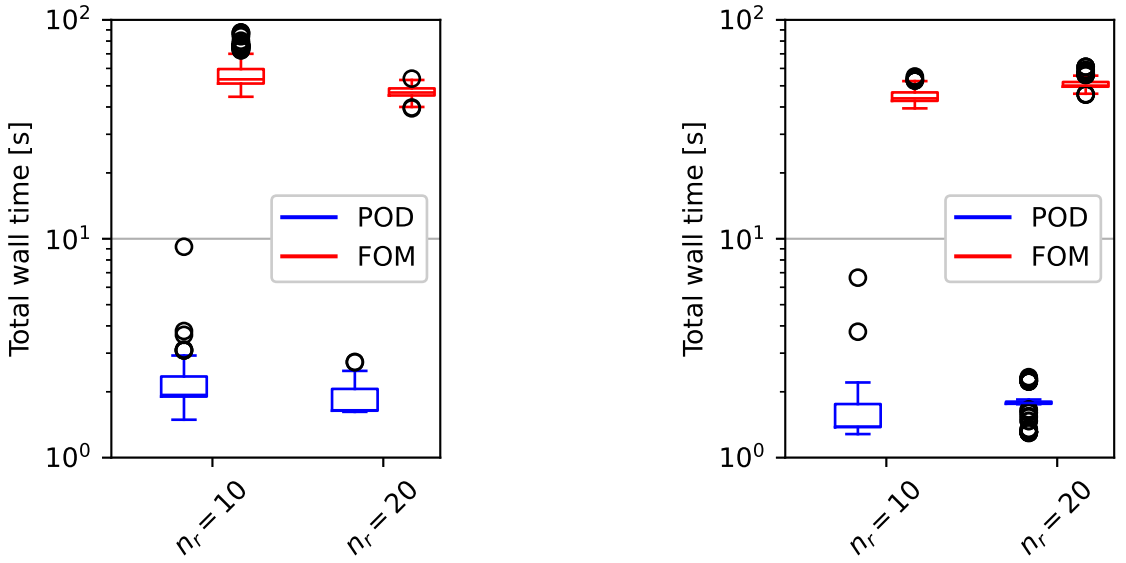
Fig. 5 Relative L_2 errors of total state vectors u of global and weighted POD approaches



(a) Global POD (b) Weighted POD

Fig. 6 Relative wing surface L_2 errors of total state vectors u of global and weighted POD approaches

especially relative to their corresponding FOM times, despite the extra basis calculation step in the weighted POD approach.



(a) Global POD **(b) Weighted POD**

Fig. 7 Wall times of global and weighted POD approaches, paired with FOM wall time distributions

VII. Conclusions

This work presents an early-stage implementation of a computational framework for high-fidelity aerodynamic shape optimization with built-in Reduced-Order Modeling (ROM) capabilities. An MPI-enabled FEniCS-based Discontinuous Galerkin (DG) approach is used to discretize the compressible Euler equations, and is paired with a linear elasticity model to convert shape deformations into mesh motions. B-spline-based Free-Form Deformation allows for control over object shapes independent from the meshes that are used for analysis. Two Proper Orthogonal Decomposition (POD) ROM approaches are implemented: One is the standard POD approach that generates global bases for the entire design space, whereas the other uses snapshot weighting to compute local reduced bases that vary throughout the design space. These models all interact with one another through the Computational Systems Design Language, which in turn allows us to use modOpt as the optimization module, thereby giving us access to a large range of optimization algorithms.

The framework is applied to the lift-constrained drag minimization of the Simple Transonic Wing benchmark model at a Mach number of 0.8, with a total of 40 design parameters. Both the global and the weighted POD basis approach managed to produce median force and moment coefficients within $O(0.1\%)$ of the high-dimensional Euler model, with the weighted POD basis getting within $O(0.01\%)$ in most cases with reduced basis sizes of 10 and 20 vectors. In part this reflects the fact that this optimization problem was carried out on a relatively coarse grid, with somewhat predictable solution variations throughout the parameter space. Lastly, both POD methods attained wall time speedups about 25 times over the high-dimensional model. This is to a large part due to the use of 0^{th}-order DG bases; we expect this speedup factor to decrease as linear or higher-order DG bases are used.

Following [7], this framework presents the next step in our research on data-driven model reduction for large-scale aerodynamic shape optimization. We aim to make the code implementation more robust, and have started looking into higher-order DG simulations and using finer meshes. Since with higher DG basis orders the assembly time of the Jacobian matrices is expected to rise, we intend to add hyper-reduction to the current set of ROM capabilities. Moreover, as we look at larger numbers of design parameters, a natural extension is to implement the derivatives that are necessary for gradient-based optimization.

Acknowledgments

The material presented in this paper is supported by NASA under award No. 80NSSC21M0070.

References

[1] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, Cambridge University Press, Cambridge, UK, 2022. https://doi.org/10.1017/9781108980647.

[2] Gray, A. C., and Martins, J. R., "A Proposed Benchmark Model for Practical Aeroelastic Optimization of Aircraft Wings," *AIAA SCITECH 2024 Forum*, 2024. https://doi.org/10.2514/6.2024-2775.

[3] Hesthaven, J. S., and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis and Applications*, No. 54 in Texts in Applied Mathematics, Springer, 2008.

[4] Houston, P., and Sime, N., "Automatic Symbolic Computation for Discontinuous Galerkin Finite Element Methods," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. C327–C357. https://doi.org/10.1137/17M1129751.

[5] Hartmann, R., and Houston, P., "An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 227, No. 22, 2008, pp. 9670–9685. https://doi.org/10.1016/j.jcp.2008.07.015.

[6] Baratta, I. A., Dean, J. P., Dokken, J. S., Habera, M., Hale, J. S., Richardson, C. N., Rognes, M. E., Scroggs, M. W., Sime, N., and Wells, G. N., "DOLFINx: the next generation FEniCS problem solving environment," preprint, 2023. https://doi.org/10.5281/zenodo.10447666.

[7] van Schie, S. P., and Hwang, J. T., "Towards Hyper-Reduced Weighted POD for Large-Scale Aerodynamic Design Optimization," *AIAA SCITECH 2026 Forum*, 2026. https://doi.org/10.2514/6.2026-1212.

[8] Dalcin, L. D., Paz, R. R., Kler, P. A., and Cosimo, A., "Parallel distributed computing using Python," *Advances in Water Resources*, Vol. 34, No. 9, 2011, pp. 1124–1139. https://doi.org/10.1016/j.advwatres.2011.04.013, new Computational Methods and Software Tools.

[9] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, 2021. https://doi.org/10.2514/1.J059491.

[10] Witteveen, J., and Bijl, H., "Explicit Mesh Deformation Using Inverse Distance Weighting Interpolation," *19th AIAA Computational Fluid Dynamics*, 2009. https://doi.org/10.2514/6.2009-3996.

[11] de Boer, A., van der Schoot, M., and Bijl, H., "Mesh deformation based on radial basis function interpolation," *Computers & Structures*, Vol. 85, No. 11, 2007, pp. 784–795. https://doi.org/10.1016/j.compstruc.2007.01.013, fourth MIT Conference on Computational Fluid and Solid Mechanics.

[12] Gandarillas, V., Joshy, A. J., Sperry, M. Z., Ivanov, A. K., and Hwang, J. T., "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis," *Structural and Multidisciplinary Optimization*, Vol. 67, No. 76, 2014. https://doi.org/10.1007/s00158-024-03792-0.

[13] Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N., "Unified form language: A domain-specific language for weak formulations of partial differential equations," *ACM Trans. Math. Softw.*, Vol. 40, No. 2, 2014. https://doi.org/10.1145/2566630.

[14] Joshy, A. J., and Hwang, J. T., "modOpt: A modular development environment and library for optimization algorithms," *Advances in Engineering Software*, Vol. 213, 2026, p. 104084. https://doi.org/10.1016/j.advengsoft.2025.104084.

[15] van Schie, S. P. C., Kramer, B., and Hwang, J. T., "Weighted Proper Orthogonal Decomposition for High-Dimensional Optimization," , 2025. URL https://arxiv.org/abs/2508.09084, under review.

[16] Gray, A. C., and Martins, J. R., "Aerostructural Optimization of the Simple Transonic Wing Using MPhys, ADflow, and TACS," *AIAA SCITECH 2025 Forum*, 2025. https://doi.org/10.2514/6.2025-2813.

[17] Gray, A. C., and Martins, J. R., "Aerostructural Optimization of the Simple Transonic Wing Considering Buffet and Takeoff Performance," *AIAA SCITECH 2026 Forum*, 2026. https://doi.org/10.2514/6.2026-1010.

[18] Mitrotta, F. M. A., Pirrera, A., and Cooper, J. E., "Preliminary Aeroelastic Optimization of the Simple Transonic Wing With Nonlinear Structural Stability Constraints," *AIAA SCITECH 2025 Forum*, 2025. https://doi.org/10.2514/6.2025-2811.

[19] Jacobson, K., Stanford, B., and Thelen, A. S., "Investigations of an Aeroelastic Optimization Benchmark Problem With MPhys and FUN3D," *AIAA SCITECH 2025 Forum*, 2025. https://doi.org/10.2514/6.2025-2812.

[20] Gray, A. C., Martins, J., Volle, F., Burke, B. J., Engelstad, S., and Kennedy, G., "Comparisons of Results for a High-fidelity Aeroelastic Optimization Benchmark Problem," *AIAA SCITECH 2026 Forum*, 2026. https://doi.org/10.2514/6.2026-0762.

[21] San Diego Supercomputer Center, Triton Shared Computing Cluster, University of California, San Diego, 2022. https://doi.org/10.57873/T34W2R.