AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-0782

Aerodynamic Shape Optimization with Curved Mesh Adaptation

Alexander W.C. Coppeans*, Krzysztof J. Fidkowski†, and Joaquim R.R.A. Martins‡
University of Michigan, Ann Arbor, MI, 48109

We present an efficient and robust strategy to control discretization error during aerodynamic shape optimization (ASO) using a high-order discretization with curved mesh adaptation. During aerodynamic shape optimization, it is important to have an accurate solution to prevent discretization error from polluting the optimum. High-order methods are promising because they offer increased accuracy for a given mesh. Mesh adaptation further improves the efficiency of high-order methods. These high-order methods require curved meshes to properly capture the simulated geometry and a mesh adaptation process that can generate curved meshes. Adapting these curved meshes needs to be robust as any failures will require human intervention inside the automated optimization loop. In this work, we introduce two key advances. The first advancement is a method for robustly splitting elements on geometric boundaries, which is necessary to properly capture the geometry. The second advancement is a concurrent mesh adaptation strategy that automatically balances accuracy and computational efficiency throughout the optimization process. Results for transonic airfoil optimization demonstrate that our method reduces the number of expensive fine-mesh evaluations by 75-90% compared to traditional approaches.

I. Introduction

Aerodynamic shape optimization (ASO) requires robust, accurate, and efficient computational fluid dynamic (CFD) [1]. The optimizer may ask the CFD solver to evaluate a design a human designer would not normally evaluate and solution failures will cause the optimization to fail. Additionally, inaccurate CFD will lead the optimizer to a spurious optimum. However, the CFD solver is the largest computational expense, so using a mesh that is too fine is computationally inefficient. A single fixed mesh that is computationally efficient and accurate for the initial design may not be accurate at the final design. This mesh may fail to capture flow features at the optimum or may be over-refined to capture flow features that move or are no longer there. Mesh adaptation allows for the optimization to start on a coarse mesh and progressively get finer. This helps prevent over-optimizing on a coarse mesh and over-refining far away from the optimum. Numerous multifidelity approaches for optimization have been previously proposed. Many approaches perform a series of optimizations at different cost or error targets [2–6]. Additionally, many approaches perform mesh adaptation on each optimization iteration to reach the prescribed error for that optimization [2, 3, 5–7]. Requiring multiple mesh adaptation iterations at each iteration significantly increases computational cost because each design requires multiple flow and adjoint solutions, and adds noise to the optimizer, reducing the convergence rate.

High-order discretization techniques paired with error estimation and mesh adaptation are promising as they can satisfy all the requirements for ASO. Error estimation and mesh adaptation are important tools for computational fluid dynamics (CFD) simulations [8]. High-order methods, which produce solutions with lower error for a given number of degrees of freedom (dof), require adaptation to optimally place and not waste their more computationally expensive degrees of freedom. Metric-based adaptation is used to generate meshes that conform to a target metric field, which encodes desired element size, stretching, and orientation throughout the domain. There have been different methods studied for performing metric-based mesh adaptation [9] such as local mesh modification [10–12] and global re-meshing [13–15].

Many of these methods successfully generate metric-conforming anisotropic linear meshes. However, high-order discretizations require curved elements on the boundaries [16]. The current state of the art is to start with a valid curved

*Ph.D Candidate, Department of Aerospace Engineering, AIAA Student Member.
†Professor, Department of Aerospace Engineering, Associate Fellow AIAA.
‡Pauline M. Sherman Collegiate Professor, Department of Aerospace Engineering, AIAA Fellow.

mesh, make it linear, perform the adaptation, and then re-curve the mesh. The re-curving process is typically done by either solving an analogous physics-based elasticity problem [17] or by solving an optimization problem to untangle the elements [18–20]. This re-curving process is computational expensive and does not always succeed for highly anisotropic meshes and for complex geometries, especially in three-dimensions. A failure of the re-curving process requires user intervention and in an optimization context will cause the optimizer to fail. However, previous work has been done to generate anisotropic metric-conforming meshes that do not require any global re-curving [21–23]. These methods keep the mesh curved during the entire adaptation procedure, leading to increased robustness compared to methods that adapt linear meshes and require re-curving. This increased robustness is required for anisotropic mesh adaptation during optimization as any failures will stop the optimization process.

In this work, we use the discontinuous-Galerkin (DG) method with high-order edge primitive (HOEP) [22] mesh adaptation. HOEP has shown to provide similar performance to existing global re-meshing with re-curving. However, more robustness is required for it to be used in an optimization framework. In this work, we present improvements to HOEP for splitting boundary faces to improve geometric resolution to prevent the optimizer from taking advantage of lack of resolution. We then present an adaptation strategy that does not require adaptation at each optimization iteration and only requires one user input for the error limit.

The rest of the paper is outlined as follows. In Section II, we describe the high-order discontinuous-Galerkin discretization and in Section III we describe the adjoint method for computing error estimates and gradients. In Section IV, we describe gradient-based aerodynamic shape optimization and the computational framework used. Section V describes metric-based mesh adaptation and metric optimization followed by Section VI, which describes high-order split operation and the robustness improvements for splits on boundary edges. Section VII describes our concurrent adaptation and optimization strategy. Results using this adaptation strategy are shown in Section VIII, and Section IX concludes the present work.

II. Discretization

In this work, we discretize a system of unsteady partial differential equations. In conservative form, the equations can be written as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the s-component state vector, $\vec{\mathbf{F}} \in \mathbb{R}^{\text{dim} \times s}$ is the flux vector, and dim is the number of spatial dimensions. For the Navier-Stokes equations, we use a conservative state vector $\mathbf{u} \in \mathbb{R}^{\text{dim}+2} = [\rho u, \rho \vec{V}, \rho E]$, where ρ is the density, $\vec{V}$ is the velocity, and E is the total internal energy per unit mass.

The CFD solver we use is xflow which discretizes the governing equation by using the discontinuous Galerkin (DG) method with the Roe [24] convective flux and the second form of Bassi and Rebay (BR2) [25] for viscous treatment. The state is approximated with polynomials of order p on an unstructured mesh of non-overlapping elements. Following a finite-element weak formulation and choice of polynomial basis functions, the semi-discrete form of the governing equations is,

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}. \tag{2}$$

Here $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, N is the total number of unknowns, $\mathbf{R}(\cdot) \in \mathbb{R}^N$ is the nonlinear spatial residual, and $\mathbf{M} \in \mathbb{R}^{N \times N}$ is the block-element sparse mass matrix. For steady simulations used in this work, the time derivative term drops out, although pseudo-time continuation [26] remains in the solver to converge the steady residual. The non-linear solver uses the Newton-Raphson method with the generalized minimum residual (GMRES) [27] linear solver, preconditioned by an element-line Jacobi smoother with coarse-level ($p = 1$) correction [28, 29].

III. The Output Adjoint

The adjoint solution for a scalar output is the sensitivity of the output to residual source perturbations [8, 30]. The steady adjoint comes from linearizing the residual vector, $\mathbf{R}(\mathbf{U})$, and the output, $J(\mathbf{U})$, with respect to the state vector, $\mathbf{U}$, and solving the linear equation,

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \mathbf{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = \mathbf{0}. \tag{3}$$

In this equation $\boldsymbol{\Psi} \in \mathbb{R}^N$ is the discrete adjoint vector, which has the same size as the residual and state vector.

A. Adjoint-Weighted Residual

The adjoint can be used for error estimation, which computes the effect of discretization error on the output,

$$\delta J = J_H(\mathbf{U}_H) - J_{\text{exact}} \approx J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h). \tag{4}$$

Here H represents a coarse-space with a state vector $\mathbf{U}_H$ and output J_H, and h represents a fine-space with a state vector $\mathbf{U}_h$ and output J_h. In this work, we use our solution with polynomial order p as our coarse-space and obtain the fine-space using polynomials of order $p+1$. The primal problem on the fine space is not solved. Instead the coarse-space solution is prolongated into the fine-space to obtain the state $\mathbf{U}_h^H$. From here the adjoint in the fine space and the residual, which will generally not be $\mathbf{0}$, are computed. The adjoint-weighted residual [8, 30] is then used to compute the output error estimate,

$$\delta J \approx J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \approx -\delta\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H). \tag{5}$$

We obtain the adjoint error, $\delta\boldsymbol{\Psi}_h^T$, by subtracting from the fine-space adjoint, solved exactly, its least-squares projection onto the coarse space. The error estimate in Equation 5 is localized to elemental contributions to the error, which can be used as an adaptive indicator,

$$\epsilon_e = |\delta\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)|. \tag{6}$$

B. Adjoint Based Gradients

The adjoint vector can also be used to efficiently compute the gradient of the outputs of interest with respect to design variables $\mathbf{x}$. In general the state and residual are functions of $\mathbf{x}$ and the PDE-constrained gradient of the output with respect to $\mathbf{x}$ is

$$\frac{dJ}{d\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{7}$$

In this work the partial derivatives in Equation (7) are computed using a forward difference which is inexpensive as it requires a single residual and function evaluation for each $\mathbf{x}$ and no additional flow or adjoint solutions.

IV. Aerodynamic Shape Optimization

A general optimization problem with equality and inequality constraints is formulated as,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{U}, \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u, \end{aligned} \tag{8}$$

where $\mathbf{x}$ is the vector of design variables, f is the objective function, $\mathbf{h}$ is the vector of equality constraints and $\mathbf{g}$ is the vector of inequality constraints. This constrained problem is solved by introducing new variables, Lagrange multipliers, and turning the problem into an unconstrained one by minimizing the Lagrangian, given as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \mathbf{s}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}) + \boldsymbol{\sigma}^T(\mathbf{g}(\mathbf{x}) + \mathbf{s} \odot \mathbf{s}), \tag{9}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ are Lagrange multipliers and $\mathbf{s}$ is a vector of slack variables used to turn inequality constraints into equality constraints and $\odot$ is the Hadamard product.

We use MACH-Aero, an open-source framework for gradient-based ASO*. The framework consists of various modules for parameterizing the geometry, mesh warping, computing CFD solutions, and performing optimization. The CFD solver we use is xflow described in Section II.

*https://mdolab-mach-aero.readthedocs-hosted.com/en/latest/

A. Geometric Parameterization

To parameterize the geometry, we use the free-form deformation (FFD) approach [31] implemented in the pyGeo package [32]. The FFD approach embeds the points on the baseline geometry into a volume with movable control points where displacing the control points results in movement of the surface nodes. This approach helps reduce the number of design variables down from the number of surface points in the CFD solver to the number of control points. Additionally, pyGeo includes modules for geometric constraints such as thickness and volume constraints.

B. Mesh Warping

After the surface nodes of the mesh are deformed using pyGeo, the rest of the volume nodes in the mesh need to be updated to remain valid. We use IDwarp [33] which uses the inverse-distance weighting method proposed by Luke et al. [34] to propagate the surface displacements to the volume. IDwarp only requires surface connectivity and treats all volume nodes as a point cloud, so this method works for both structured and unstructured meshes. IDwarp does not natively handle curved meshes so we create a surface mesh by linearly connecting high-order surface nodes. The high-order surface nodes can be chosen as all the Lagrange nodes on the surface but in this work we uniformly refine the surface representation to have a better representation of the curved elements.

C. Optimizer

For our optimizer we use SNOPT [35] which is a general-purpose optimizer designed for large optimization. SNOPT uses sequential quadratic programming (SQP) to minimize nonlinear functions subject to both linear and nonlinear constraints. SNOPT is wrapped with the python packaged pyOptSparse [36], which provides a python interface for many different optimizers.

V. Mesh Adaptation

A. High-Order Elements

To represent high-order meshes, we use Lagrange basis functions to map points on a reference triangle to physical space. The mapping performs a linear combination of Lagrange polynomials,

$$\vec{x}\left(\vec{\xi}\right) = \sum_{i}^{N_Q} \vec{x}_i \phi_i\left(\vec{\xi}\right), \tag{10}$$

where $\vec{\xi}$ is the coordinate vector in reference space, N_Q is the number of interpolating polynomials, and $\vec{x}_i$ is the coordinate vector in physical space associated with the i^{th} interpolating polynomial ϕ_i.

The mapping Jacobian, $\underline{\mathrm{J}}$, is defined as,

$$\underline{\mathrm{J}} = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \sum_{i}^{N_Q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{\xi}}\left(\vec{\xi}\right). \tag{11}$$

The determinant of the mapping Jacobian matrix is required to be positive at all points for a valid mesh.

In addition to requiring the Jacobian determinant to be positive everywhere, we also restrict the element distortion, defined as

$$\sigma\left(\vec{\xi}\right) \equiv \frac{\det \underline{\mathrm{J}}^{q=1}}{\det \underline{\mathrm{J}}\left(\vec{\xi}\right)}. \tag{12}$$

$\det \underline{\mathrm{J}}^{q=1}$ is the determinant of the constant $q = 1$ element mapping Jacobian and $\det \underline{\mathrm{J}}(\vec{\xi})$ is the determinant of the curved element mapping Jacobian, which is a polynomial of order $2(q-1)$ for triangles. We limit the distortion during the adaptation process to be in the range $0.2 \leq \sigma \leq 5$ but place no restrictions during the mesh warping stage of optimization. We found limiting the distortion in this range improves solver robustness.

To efficiently check the bounds of the distortion and the Jacobian determinant in each element, we use the method proposed by Johnen et al. [37]. A Bézier representation of the $2(q-1)$ Jacobian determinant polynomial is used to conservatively check the upper and lower bounds. If the bounds of the Bézier control points are outside the acceptable range, the domain of the polynomial is recursively subdivided to achieve a better bound until a corner control point is negative or a max number of subdivisions is reached in which case the element is deemed invalid.

B. Metric-Based Meshing

A Riemannian metric field, $\mathcal{M}(\vec{x}) \in \mathbb{R}^{\dim \times \dim}$, is a field of symmetric positive definite (SPD) tensors that are used to encode information about the desired shape and stretching of elements. For a mesh to conform to the given metric field, all edges of the mesh should be unit length in the metric space. To compute the edge length in metric space we follow the method proposed by the Unstructured Grid Adaptation Working Group (UGAWG) [9].

First, we start by computing the target mesh size at each node, where the metric is stored, in the direction of each edge attached to that node. The target mesh size, h, at a point in the direction of the unit vector $\vec{e}$ can be computed as,

$$h = \frac{1}{\sqrt{\vec{e}^T \mathcal{M} \vec{e}}} \tag{13}$$

Each edge is made up of 2 nodes a and b with mesh sizes in the direction of the edge h_a and h_b. Many ways are available to compute the edge length depending on how we assume the mesh size changes between nodes [38]. We follow the conventions of the UGAWG and assume the logarithm of h varies linearly along the edge. The edge length in metric space [9, 38], L_m, can then be computed as,

$$L_m = L \frac{\frac{1}{h_a} - \frac{1}{h_b}}{\ln(\frac{h_b}{h_a})}, \tag{14}$$

where L is the length of the edge in physical space. When $h_a = h_b$ the target mesh size at each node is the same and Equation 14 is indeterminate, so we modify the calculation to be,

$$L_m = L \frac{\frac{1}{h_a} + \frac{1}{h_b}}{2}, \text{ when } \left| \frac{1}{h_a} - \frac{1}{h_b} \right| \leq 0.001. \tag{15}$$

Additionally the metric field can be used to compute a quality metric for each element. First we compute the volume of the element under the metric field given by

$$V_{m,k} \approx \sqrt{\det \mathcal{M}_{\max}} V_k, \tag{16}$$

where $V_{m,k}$ is the metric volume of element k, V_k is the volume of element k, and $\mathcal{M}_{\max}$ is defined as

$$\mathcal{M}_{\max} = \arg \max_{\mathcal{M}_i, i \in k} \det \mathcal{M}_i. \tag{17}$$

Again, following the UGAWG, we can compute the quality of element k in two dimensions as,

$$Q_k = \frac{\left(\frac{V_{m,k}}{\hat{V}_k} \right)}{\frac{1}{3} \sum_{e \in k} \vec{v}_e^T \mathcal{M}_{\max} \vec{v}_e}, \tag{18}$$

where Q_k is the quality of element k, $\hat{V}_k$ is the area of a unit-equilateral triangle, and $\vec{v}_e$ is the vector along edge e on element k.

C. Metric Optimization

The elemental error indicator in Equation 6, together with additional error indicators evaluated on sub-elements with projected adjoints, drive a metric optimization calculation based on MOESS: mesh optimization through error sampling

and synthesis [39, 40]. The goal of the optimization problem is to determine the target metric field that minimizes error at a fixed cost. This algorithm iteratively equidistributes the marginal error to marginal cost ratio over the mesh elements. The output of MOESS is a target metric field at all the nodes in the mesh. From here, there are multiple choices to obtain a mesh that conforms to this metric field. In this work we use high-order edge primitive operations extended to curved meshes [22] described in Section V.D. In practice the mesh optimization and flow/adjoint solution are performed multiple times at a target cost, C_{target}, until the error stops changing.

D. Edge Primitive Operations

To create a mesh that conforms to the desired metric field, we use high-order edge primitive (HOEP) operations. This method uses a sequence of local edge-based operations to improve metric conformity of an input mesh. For a metric to conform to a metric field, the mesh must have all unit length edges under the metric field. However, it is not always possible to obtain exactly unit lengths everywhere, so instead we aim to achieve a quasi-unit edge mesh. Similar to what has been done previously [9–11, 41] we try to obtain a mesh with edge lengths under the metric, L_m, in the range, $\sqrt{2}/2 \leq L_m \leq \sqrt{2}$.

Previous work modified the edge operations to work for arbitrary order curved elements [22]. We compute the metric edge length on curved elements assuming the edge is linear using Equation 14. Edges that are too long, $L_m \geq \sqrt{2}$, are split while edges that are too short, $L_m \leq \sqrt{2}/2$, are collapsed. To improve metric quality we perform local edge swaps and use metric based node smoothing proposed by Caplan et al. [41] which is inspired by Bossen and Heckbert [42].

The main benefit of this method is that if the input curved mesh is valid it outputs a curved mesh that is valid, as any operator that would create invalid elements is rejected. Typically, curved meshes are made linear and then adapted to obtain a linear the output mesh, which then needs to be re-curved. This is necessary for robust shape optimization, as for large deformations or unusual geometries, the re-curving or snapping to linear step may fail, which would break the automated optimization process.

VI. High-Order Split Improvements

We focus on improving the robustness of the edge split operation on boundaries. This is because ensuring the proper mesh resolution on boundaries is critical for the optimizer, as it will exploit regions that lack geometric and solution accuracy. In our previous work, we have determined two main reasons for edge splits to get continuously rejected, creating highly distorted elements and creating low metric quality elements. Highly-distorted elements are likely to invert and create elements with negative Jacobian determinants during the mesh warping stage of optimization. Additionally, they cause robustness issues as they can cause the primal solution to not converge. Low quality elements indicate poor metric conformity, which reduces solution accuracy. Therefore, it is critical to ensure that splits can proceed while maintaining element quality and distortion metrics.

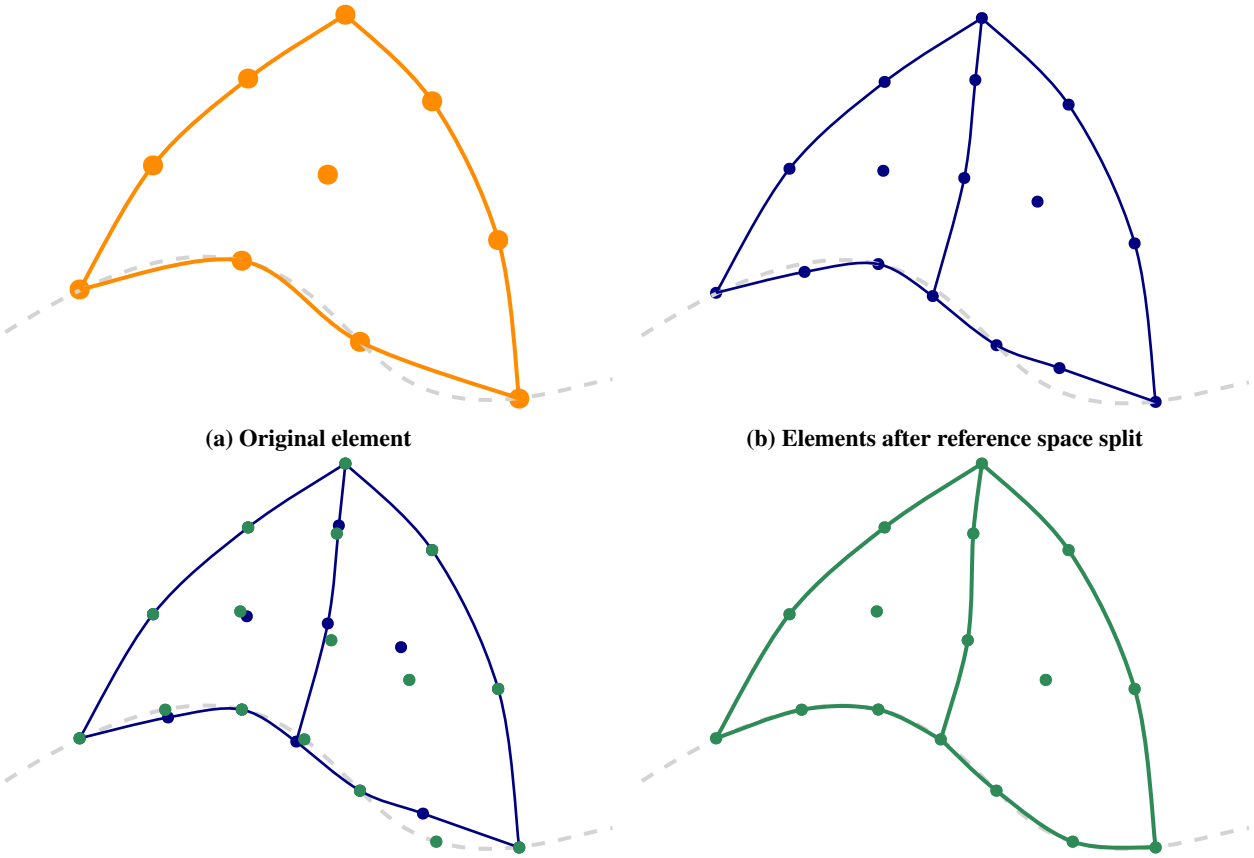
A. High-order split operation

The edge split for boundary edges is performed in three major steps shown in Figure 1. The first step splits the element in reference space and re-samples Lagrange nodes in global space to create child elements. This is the only step for interior edges. The second step for boundary edges projects the new nodes onto the geometric boundary because newly created nodes on the split edge are on the original element but are not necessarily on the geometric boundary. The final step uses an inverse-distance approach to transfer those displacements to nodes inside the elements. When performing an edge split, only the elements connected to that edge are affected, which means nodes that on the other two edges are not re-positioned.

The inverse-distance displacement transfer is computed as

$$\Delta\vec{x} = \vec{x} - \vec{x}_0 = \frac{\sum_i^{N_b} w_i\left(\vec{x}_0\right)\Delta\vec{x}_{b,i}}{\sum_i^{N_b} w_i\left(\vec{x}_0\right)}, \tag{19}$$

where $\vec{x}_0$ is the original location of the interior nodes, N_b is the number of nodes on the ball boundary, i is the index over boundary nodes, $\Delta\vec{x}_{b,i}$ is the displacement of the i^{th} boundary node, and w_i is the weight the i^{th} boundary node has on the interior node. The weights are inversely proportional to powers of the distance between the interior node and the



(a) Original element **(b) Elements after reference space split**

(c) Inverse-distance displacement transfer **(d) Final split elements**

Fig. 1 Baseline high-order edge split operation.

boundary node,

$$w_i\left(\vec{x}_0\right) = \left(\frac{1}{\left|\vec{x}_0 - \vec{x}_{b,i}\right|}\right)^{\gamma} + \left(\frac{\alpha}{\left|\vec{x}_0 - \vec{x}_{b,i}\right|}\right)^{\theta}, \tag{20}$$

where α, a, and b, are tunable parameters set to $\alpha = 0.25$, $\gamma = 3$, and $\theta = 5$ following the recommendation of Luke et al. [34].

B. Expanded warping

Previously, when performing an edge split, only the elements connected to that edge are affected. This means that nodes on other edges are not moved, so neighboring elements remain unchanged. However, we now allow for the neighbors shown in Figure 2 to be included. Elements c_1 and c_2 are child elements created from splitting the parent element, elements n_1 and n_2 are their respective neighbors, and e_1 and e_2 are the edges between the child and neighbor elements.

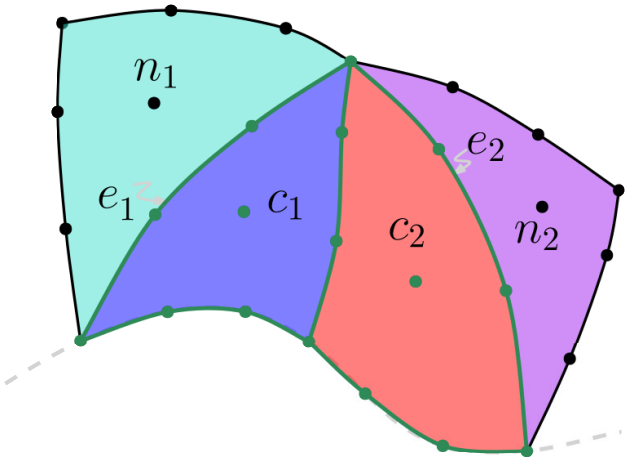


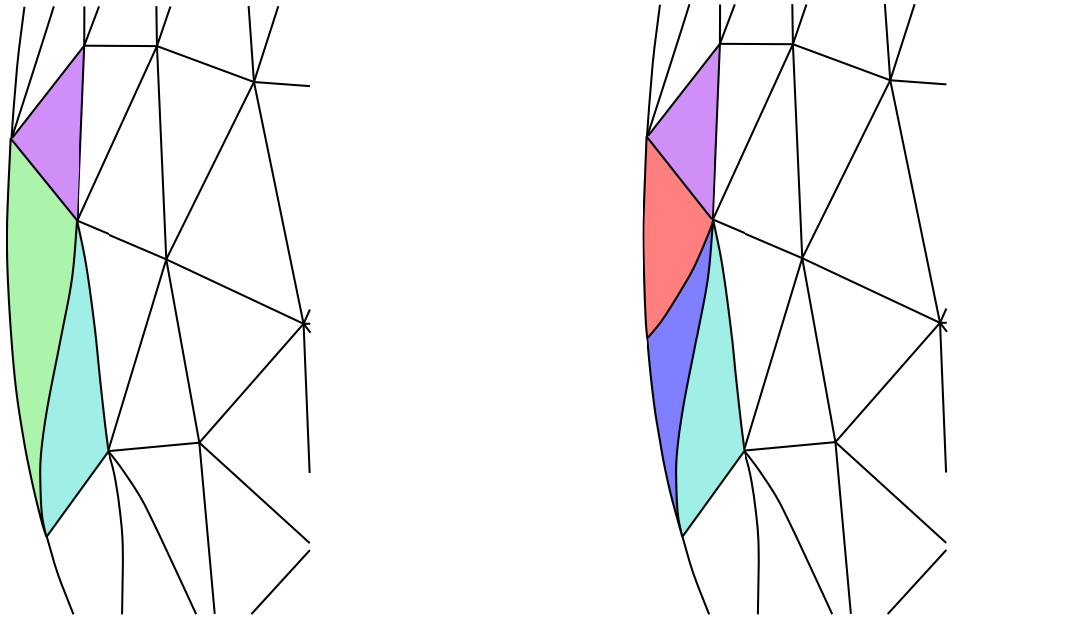
Fig. 2 Diagram labeling child elements and neighbors after edge split.

One example is a highly curved shared edge could cause one of the new split elements to have high distortion, as shown in Figure 3. To solve issues like this we allow for nodes on that face to be warped during the warping stage of the split. This is done by including the neighbor element and its nodes in the inverse-distance calculation. Nodes on the shared edge as well as any interior nodes in the other element are displaced. The nodes on the other two edges of the neighbor are fixed and have displacements of 0 which are included in the warping as boundary nodes.

If the split is rejected because child element c_1 has high distortion, then neighbor element n_1 is used in the inverse-distance calculation. Nodes not on edge e_1 have their displacements set to 0, and interior nodes are moved as well as nodes on edge e_1. If this case fails, then all nodes on element n_1, including nodes on e_1, are moved to their linear positions. The nodes not on e_1 have their displacements set so their positions are back to their original positions and those displacements are used in the inverse-distance calculation. The new edge split is shown in Figure 4, and does not produce any elements that have high distortion measures.

C. Split and swap combined operation

The other recurring reason a split would continuously get rejected is due to the creation of elements with low metric quality. We use the linear representation of elements for quality and edge length calculations which is a current limit as a high-order element may be completely different than its linear representation. This is shown below in Figure 5 where the split creates an element whose linear representation is a sliver with low quality and is therefore rejected. If this split were, accepted it would be possible to swap the edge of the sliver element with its neighbor to create two new elements that would pass the quality limits. This, however, is not always the case so a split cannot be accepted without knowing a swap would be possible.



(a) Element before edge splits

(b) Edge split that creates highly distorted element (blue)

Fig. 3 Edge split that is rejected due to creation of a highly distorted element.

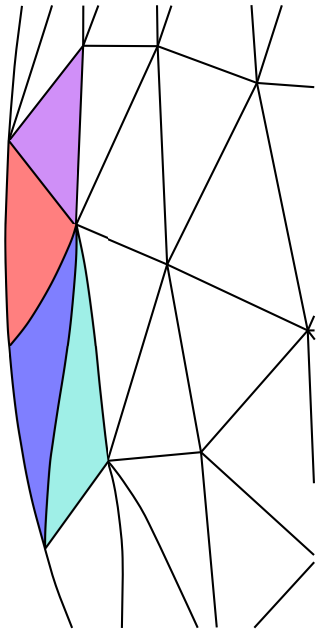
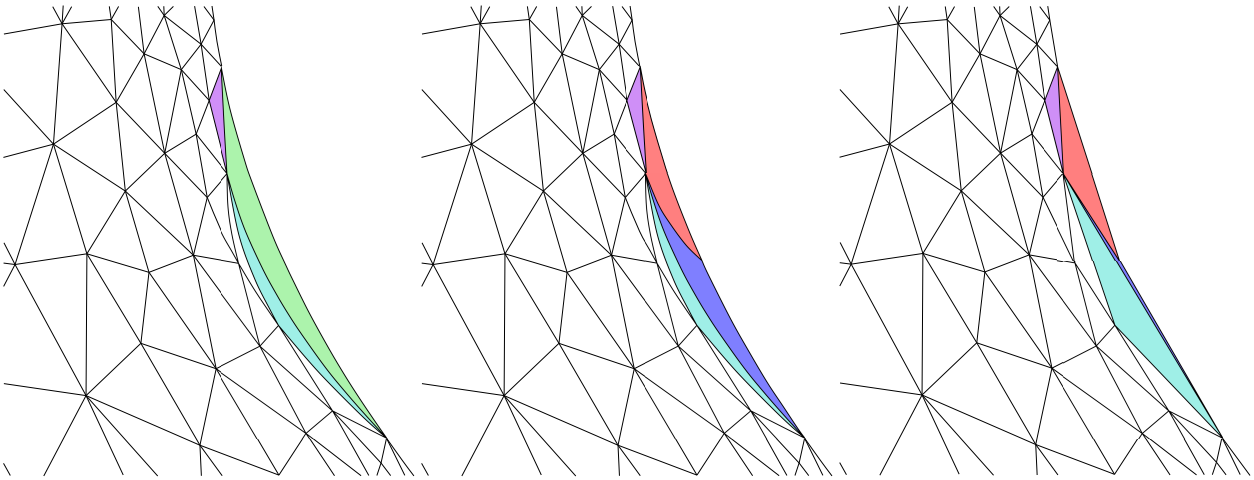


Fig. 4 Successful edge split does not create elements with high distortion.

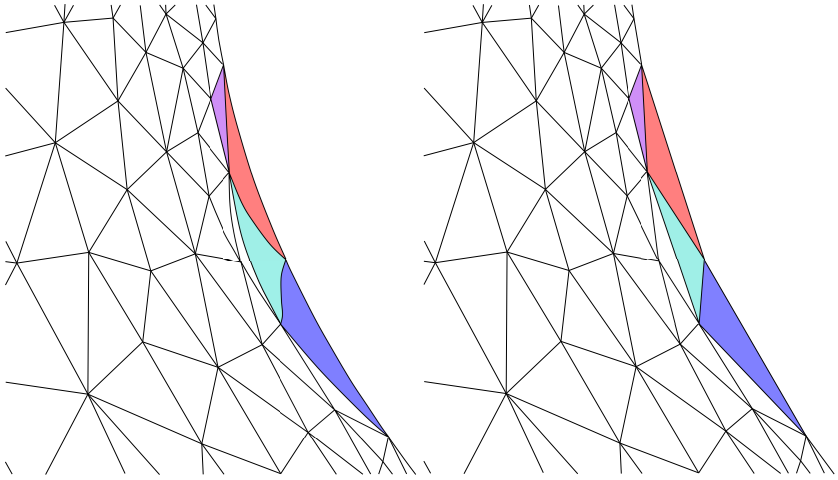


(a) Original Mesh **(b) Curved mesh after split.** **(c) Linear representation after split.**

Fig. 5 Boundary edge split rejected due to the creation of a low quality element (blue).

Here we take ideas from cavity operators, where elements are formed from the proposed split and their neighbors but not added to the mesh. Using the notation described in Figure 2, if the split is rejected due to child element c_1 failing any validity checks, we try to swap edge e_1. Two new elements are created for c_1 and n_1 and the swap is tried without updating the mesh. If the swap is successful, elements c_1 and c_2 are added to the mesh even if they produce invalid elements. Immediately after the edge split the swap on edge e_1 is performed, and the mesh is updated. This method allows for c_1 or c_2 to be invalid temporarily, as long as the swap creates valid elements, which was tested before c_1 and c_2 were added to the mesh.

The resulting mesh is shown in Figure 6.



(a) Curved mesh after split with swap **(b) Linear mesh after split with swap**

Fig. 6 Mesh after boundary forced split followed by edge swap operation.

VII. Optimization with Adaptation

During optimization, discretization errors in the objective and constraints will impact the path of the optimizer and the optimal design. For example, given a discretization with large amounts of numerical diffusion, the optimizer can find a design that uses that diffusion to smear out shocks and reduce wave drag. Under-predicting lift can cause the optimizer to choose a design that, when trimmed and analyzed on finer mesh, increases drag. Therefore, accurate CFD is required for ensuring the correct optimal design is found. To account for the effects of the constraint errors on the objective we compute the error of the Lagrangian given by

$$\delta\mathcal{L} = |\delta f| + |\boldsymbol{\lambda}^T||\delta\mathbf{h}| + |\boldsymbol{\sigma}^T||\delta\mathbf{g}|. \tag{21}$$

We use absolute values around the objective and constraint errors as well as on the Lagrange multipliers, to avoid having errors cancel out. Additionally, instead of using Equation (5) for each output error, we use the sum of element indicators from Equation (6) as this avoids cancellation of errors across elements. The sum of element indicators is a more conservative estimate, typically by an order of magnitude, as observed in numerical experimentation, so we set larger error limits.

The goal of our adaptation strategy is to balance not over refining designs far from the optimum while also not over optimizing on a coarse mesh. However, using a costly discretization far away from the optimum increases computational cost when a coarser discretization can be used to explore the design space. Taking more steps on coarser meshes allows for an SQP optimizer using a quasi-Newton approach to build a better approximate Hessian that can be used to accelerate the optimization on a finer mesh [43]. We combine the ideas of two previous approaches. The first uses the function change between successive optimization iterations [5] to determine the necessary error tolerance. The second uses relative convergence of the optimality KTT criteria to determine when to adapt [44]. The necessary discretization error for a given step is computed as

$$\mathcal{E}^{\text{tol}} = \left|f_i - f_{i-1}\right| \frac{\tau_{\text{opt},i}}{\tau_{\text{opt}}^{\text{tol}}}, \tag{22}$$

where f_i and f_{i-1} are the function values at the current and previous optimization iterations, $\tau_{\text{opt},i}$ is the optimality measure at iteration i, and $\tau_{\text{opt}}^{\text{tol}}$ is the optimality convergence tolerance. This strategy allows the optimizer to take small steps far away from the optimum on coarser meshes, as these are cheap, and helps build the approximate Hessian while ensuring small steps close to the optimum are within the error tolerance. We set a user-defined limit on the final error target, $\mathcal{E}^{\text{limit}}$, when the final optimization steps have very small function changes as the optimizer quadratically converges to the optimum. When we adapt the mesh and the error target is set to $\mathcal{E}^{\text{limit}}$, we apply a safety factor $\eta = 0.95$. This helps avoid continuously reaching the limit and oscillating around it and adapting many times close to the optimum, which impacts performance. Additionally, due to the cost required to solve the fine-space adjoint problem, discretization errors for the function and constraints are only computed every n_{adapt} optimization iterations, here set to 5.

VIII. Results

We test our adaptation strategy optimizing the RAE 2822 airfoil at a freestream Mach number of $M_\infty = .734$ and Reynolds number $Re = 6.5 \times 10^6$. We impose a lift constraint $c_\ell = 0.734$ and an area constraint $A \geq A_0$, where A_0 is the original area of the RAE 2822 airfoil. We start with a coarse mesh of 1,366 $q = 3$ elements shown in Figure 7. This mesh was generated using 10 adaptation iterations. Starting with a coarser mesh that did not properly capture the boundary layer, caused too much noise for the optimizer, and caused failures. After optimizing, designs are analyzed on a fine-grid by running 20 iteration of concurrent adaptation with re-trimming using $p = 3$ solution approximation at a target cost of 96,000 degrees of freedom.

A. Error Target

To test our optimization strategy, we at an error target of $\mathcal{E}^{\text{limit}} = 1 \times 10^{-4}$ and 5×10^{-5}, referred later as "high" and "low" error respectively, using uniform $p = 2$ and $p = 3$ solution approximations solution approximations. Table 1 summarizes the results of these four optimizations.

The results here show that our adaptation strategy is effective at significantly reducing the number of optimization iterations at the final mesh. The optimizer uses the coarser meshes to explore the design space and build up a good

Algorithm 1 Optimization with error estimation and adaptation

$\tau_{\text{opt}}^{\text{tol}}, \tau_{\text{feas}}^{\text{tol}}$ ▷ Set optimality and feasibility tolerances
$i = 0$ ▷ Major iteration counter
$x_i = x_0$ ▷ Set initial design
$\mathcal{E}^{\text{limit}}$ ▷ Set desired final error tolerance
n_{adapt} ▷ Initialize n_{adapt}
$n = 0$ ▷ Initialize number of major iterations since last adaptation
while $\tau_{\text{opt}} > \tau_{\text{opt}}^{\text{tol}}$ AND $\tau_{\text{feas}} > \tau_{\text{feas}}^{\text{tol}}$ **do** ▷ While optimality and feasibily criteria not met
  Compute $f_i, \mathbf{g}_i, \mathbf{h}_i, \nabla f_i, \nabla \mathbf{g}_i, \nabla \mathbf{h}_i$ if needed
  Compute search direction $\delta\mathbf{x}, \delta\boldsymbol{\lambda}, \delta\boldsymbol{\sigma}$
  Perform line search updating f_{i+1},$\mathbf{h}_{i+1}$,$\mathbf{g}_{i+1}$,x_{i+1}, λ_{i+1}, σ_{i+1}, τ_{opt}, and τ_{feas}
  if $i = 0$ OR $n \pmod{n^{\text{adapt}}} = 0$ **then**
    Compute δf, $\delta\mathbf{h}$, and $\delta\mathbf{g}$ ▷ Compute error estimate of objective and constraints
    $n = 0$ ▷ Reset n after computing error estimate
  end if
  $\delta\mathcal{L} = |\delta f| + |\boldsymbol{\lambda}^T||\delta\mathbf{h}| + |\boldsymbol{\sigma}^T||\delta\mathbf{g}|$ ▷ Compute Lagrangian error estimate using latest multipliers
  $\mathcal{E}^{\text{tol}} = \left|f_{i+1} - f_i\right| \frac{\tau_{\text{opt},i+1}}{\tau_{\text{opt}^{\text{tol}}}}$
  if $\delta\mathcal{L} \leq \mathcal{E}^{\text{tol}}$ AND $\delta\mathcal{L} \leq \mathcal{E}^{\text{limit}}$ **then**
    Adapt the mesh to meet $\delta\mathcal{L} \leq \max(\frac{\delta\mathcal{L}}{2}, \mathcal{E}^{\text{limit}})$
  end if
  $n = n + 1$
  $i = i + 1$
end while

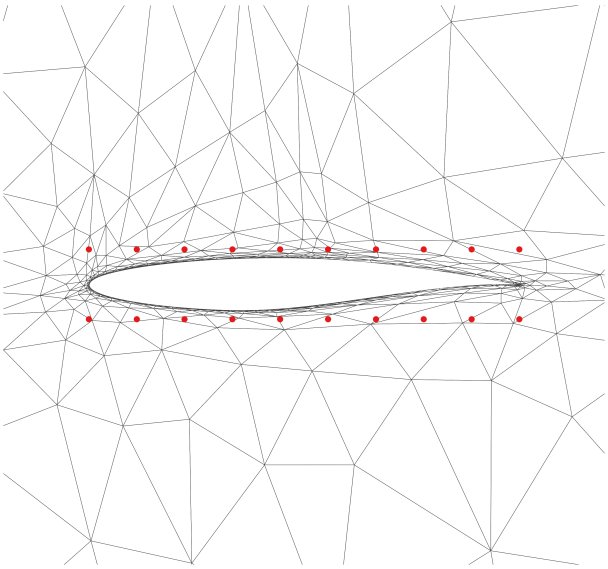


Fig. 7 Starting mesh for RAE 2822 optimizations with 20 FFD points.

Table 1 RAE 2822 optimization summary

p	$\mathcal{E}^{\text{limit}}$	Time (hr)	Final DoFs	α (deg)	c_d (counts)	Total Iterations	Final Mesh Iterations	fine-grid α (deg)	fine-grid c_d (counts)
$p = 2$	1×10^{-4}	1.58	43,734	2.44	104.81	119	13	2.41	102.38
	5×10^{-5}	3.71	69,660	2.41	103.50	135	15	2.40	102.29
$p = 3$	1×10^{-4}	2.17	32,020	2.46	106.67	110	18	2.42	102.62
	5×10^{-5}	2.48	40,530	2.44	105.64	116	13	2.41	102.47

approximate Hessian and quickly converges on the final mesh once the error limit is reached. Additionally, for the same error target, using $p = 3$ is faster and uses fewer degrees of freedom than $p = 2$ while also taking fewer optimization iterations, due to starting with a more accurate solution on the initial mesh. This is because $p = 3$ has faster error convergence, so less time is spent in the adaptation loop to reach the requested error targets. On the fine grid, all four designs perform within 0.4 drag counts. We see that reducing the error limit reduces drag when analyzing on the fine-grid which is expected as there is less numerical diffusion that can smear out shocks. Going from $p = 2$ we see the optimized designs perform worse and the original mesh shows a difference of 3–4 drag counts. This is likely due to poor error estimates with shock capturing and $p = 3$ using fewer elements for the same error, leaving too few elements to properly align with the shock. The misalignment introduces more artificial diffusion in the shock capturing.

The final geometry and meshes are shown in Figure 8. Here we see that in all cases, the adaptation targets the wave structure on the upper surface of the airfoil. Additionally, mesh resolution is added in a small separation zone downstream of the shock. The $p = 2$ mesh also adds refinement on the stagnation streamline which is due to a singularity in the adjoint solution [45]. The $p = 3$ solutions have fewer elements to use and MOESS does not target the stagnation streamline with as many anisotropic elements, instead placing them around the shocks and in the boundary layer.

The final c_p distributions are shown below in Figure 9. Here we see all of the designs as well as their c_p distributions are visually identical. The $p = 3$ designs as well as the $p = 2$ with $\mathcal{E}^{\text{limit}} = 1 \times 10^{-4}$ have larger oscillations on the upper surface due to stronger expansion and compression waves. However, the $p = 2$ design with $\mathcal{E}^{\text{limit}} = 5 \times 10^{-5}$ has a smaller amplitude in the final design, indicating the optimizer used the discretization error less to weaken the waves.

B. Varying the number of FFD points

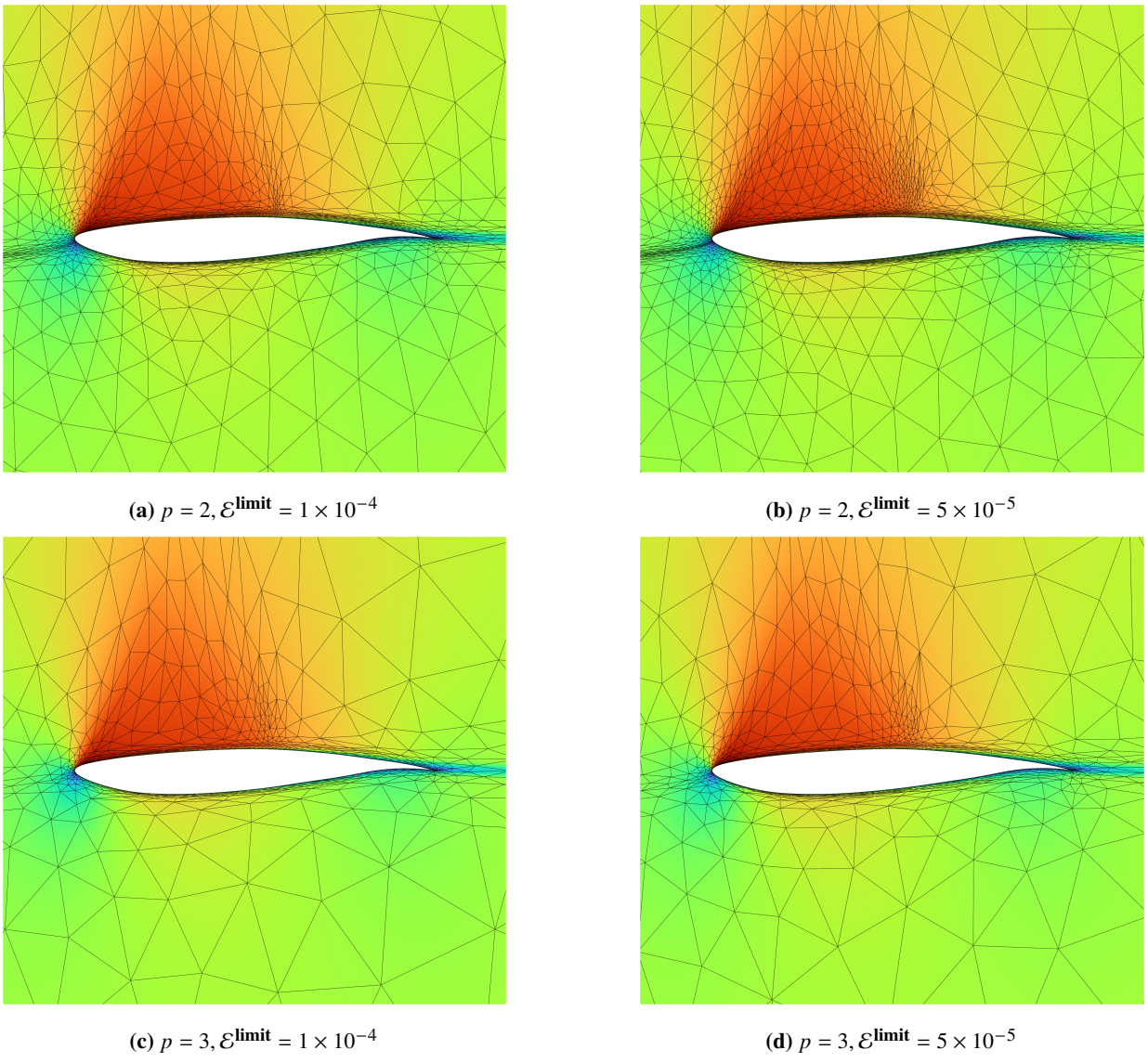
Next we study how our adaptation strategy scales with the number of design parameters. We run the same optimization problem for the RAE 2822 airfoil with $p = 2$ solution approximation and $\mathcal{E}^{\text{limit}} = 1 \times 10^{-4}$ but vary the number of FFD design variables, choosing 10, 20, and 30 FFD control points.

The results are summarized in Table 2. Here we see the number of optimization iterations increases with the number of FFD points. The actual time decreases between using 10 and 20 points due to increased time spent adapting the mesh to reach the error tolerance. Only using 10 FFD parameters is not enough to fully smooth the shock, so more elements are needed capture it as shown in Figure 11. As we increase the number of FFD points, the amount of mesh refinement around the shocks decreases due to the optimizer using the extra design freedom to reduce their strength and effect on the drag. This is why as the number of FFD points increases, the optimized drag reduces.

Table 2 RAE 2822 with varying FFD optimization summary

# FFD	Time (hr)	Final DoFs	α (deg)	c_d (counts)	Total Iterations	Final Mesh Iterations	fine-grid α (deg)	fine-grid c_d (counts)
10	2.01	49,572	2.63	105.94	81	12	2.61	103.65
20	1.58	43,734	2.44	104.81	119	13	2.41	102.38
30	6.57	45,900	2.33	104.09	397	105	2.30	101.88

Figure 12 shows the c_p distributions over the optimized airfoils. Here we see that there are oscillations on the upper surface due to a series of compression waves reflecting off the upper surface. The shock structures of these airfoils are



(a) $p = 2, \mathcal{E}^{\text{limit}} = 1 \times 10^{-4}$

(b) $p = 2, \mathcal{E}^{\text{limit}} = 5 \times 10^{-5}$

(c) $p = 3, \mathcal{E}^{\text{limit}} = 1 \times 10^{-4}$

(d) $p = 3, \mathcal{E}^{\text{limit}} = 5 \times 10^{-5}$

Fig. 8 Mach contours (0–1.4) and mesh for optimized RAE 2822 airfoils with various error limits.

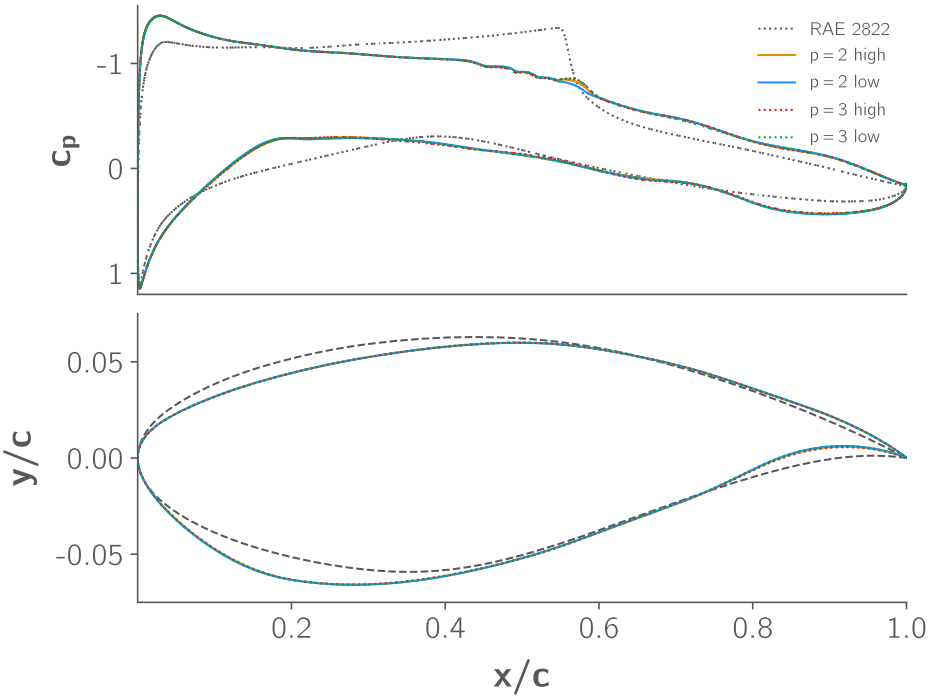
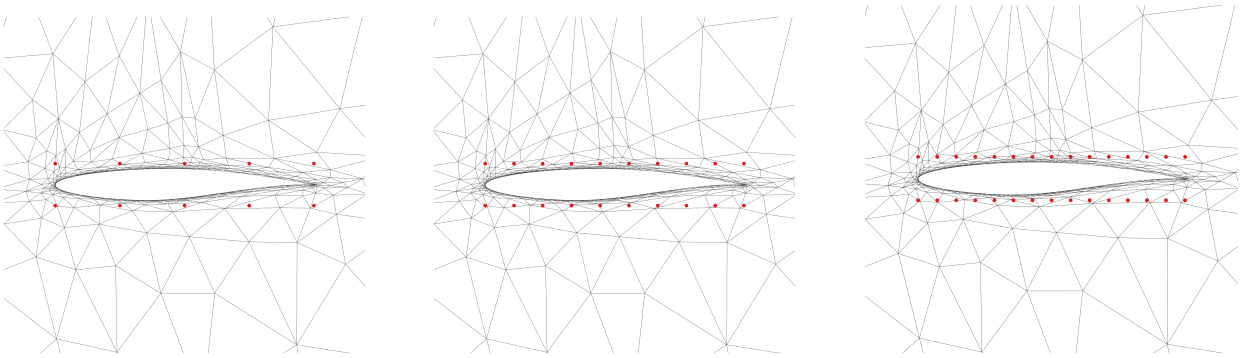
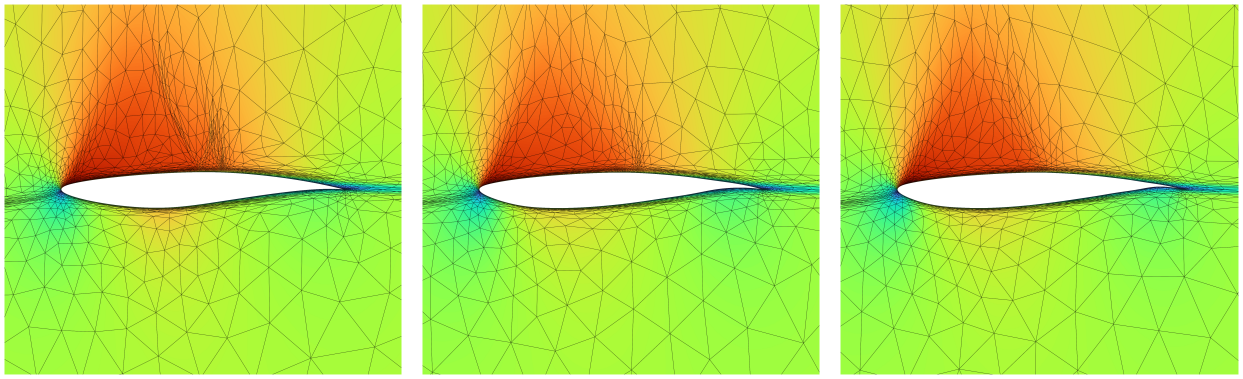


Fig. 9 c_p **distribution for optimized RAE 2822 airfoils with various error limits.**



(a) 10 FFD points **(b) 20 FFD points** **(c) 30 FFD points**

Fig. 10 Starting mesh and different FFD parameterizations for the RAE 2822 airfoil.



(a) 10 FFD points **(b) 20 FFD points** **(c) 30 FFD points**

Fig. 11 Mach contours (0–1.4) and mesh for optimized RAE 2822 airfoils with varying number of FFD points.

shown in Figure 13, which shows Mach number contours obtained after re-analyzing these designs on finer adapted meshes at the same $c_\ell = 0.824$. As we increase the number of design variables, the oscillations reduce in magnitude as the optimizer reduces the strength of those waves. Additionally, the 10 FFD parameterized airfoil has a noticeable shape difference on the upper and lower surfaces from the 20 and 30 FFD optimized airfoils. Between the 20 and 30 FFD optimized airfoils, the upper surface is nearly identical but on the lower surface, especially at the trailing edge, the shapes have large differences.

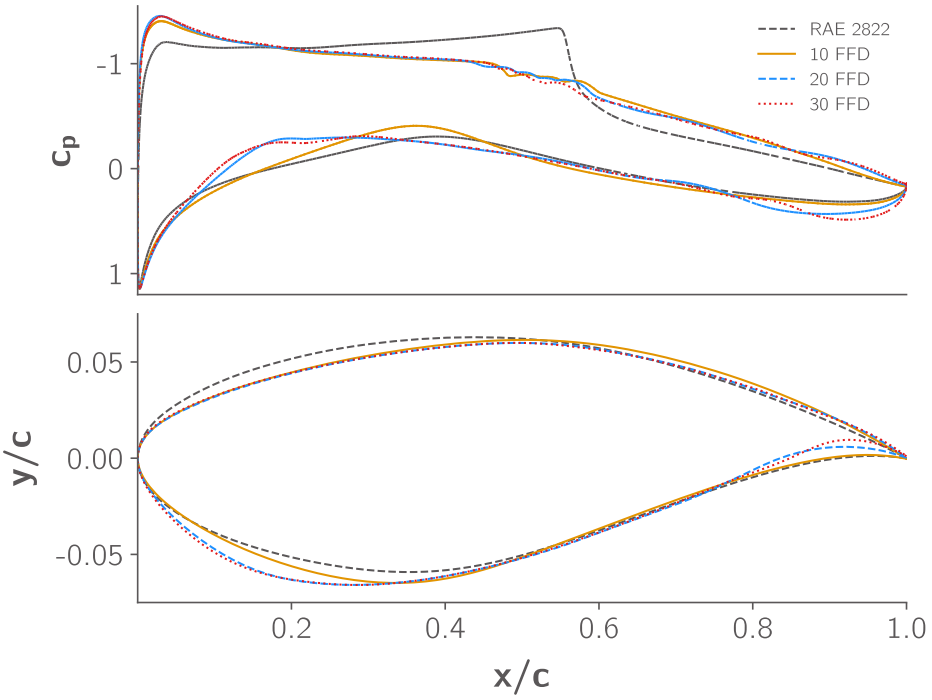
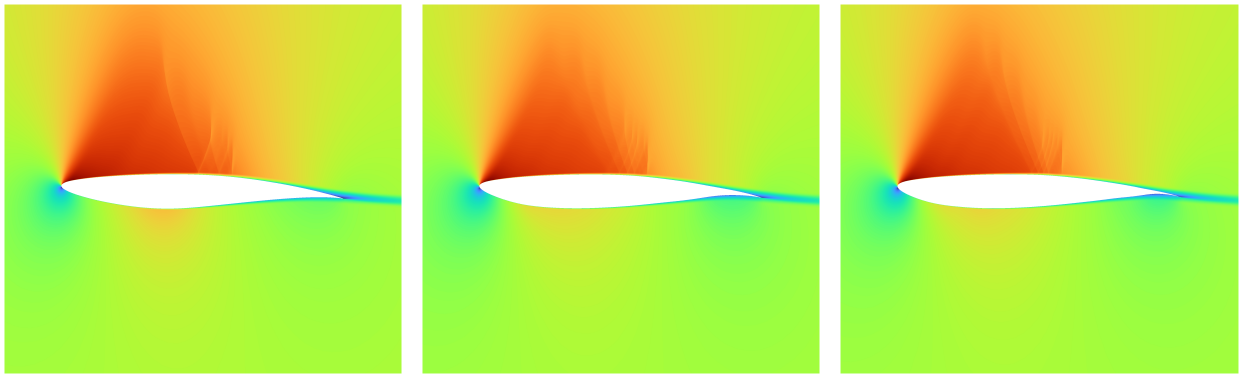


Fig. 12 c_p distribution for optimized RAE 2822 airfoils with various error limits.

The flow field shown in Figure 13 shows that using only 10 FFD points, there one wave that is very strong that reflects multiple times off the upper surface and the sonic line. However, the 20 and 30 FFD optimized airfoils have a series of three compression waves that reflect off the airfoil before the final terminal normal shock. This shows that the

optimizer adds a series of weaker shocks to slow the flow down over the top of the airfoil to reduce wave drag and separation.



(a) 10 FFD points **(b) 20 FFD points** **(c) 30 FFD points**

Fig. 13 Mach contours (0–1.4) from fine-grid for optimized RAE 2822 airfoils with varying FFD points.

IX. Conclusion

In this paper, we present a method for improving robustness of curved element adaptation and a strategy to use that curved adaptation to control discretization error during aerodynamic shape optimization. Aerodynamic shape optimization and high-order CFD methods both benefit from mesh adaptation. During optimization, the design changes and regions of interest move in the domain and important flow features that are present at the initial design may not be present on the optimized design. High-order methods benefit by optimally placing degrees of freedom in the domain to not waste degrees of freedom in regions of that are not important for output prediction. However, high-order methods require curved meshes which are more challenging to generate and adapt. Optimization adds an increased robustness requirement to avoid failures or user intervention.

In this work we solve these problems with two new advancements. The first improves edge splitting on boundary edges to ensure proper geometric resolution. This prevents the optimizer from exploiting a poor geometry representation and improves robustness of the mesh warping as poor quality elements on the surface are likely to invert. The second advancement is our concurrent adaptation and optimization strategy that prevents over-optimizing on a coarse mesh and over-refining a design far from the optimum. Combining these two advancements, we have a robust method that balances computational accuracy and cost throughout the optimization process. HOEP ensures no user intervention is required during the mesh adaptation step and our concurrent adaptation and optimization strategy ensures no user intervention to decide mesh resolution.

Our results show HOEP is robust enough for automated high Reynolds number transonic shape optimization. The present adaptation strategy significantly reduces computational cost by only using 10 to 25% of optimization iterations on the finest mesh. Additionally, using $p = 3$ further improves time by taking advantage of its higher error convergence rates to use 25 to 40% fewer degrees of freedom and spend less time adapting the mesh to reach a target error. Studying the number of design variables showed a significant increase in cost as the FFD parameterization is poorly conditioned. However, by adding more design variables the optimizer used that freedom to increase the number of compression waves, while decreasing their strength, on the upper surface to reduce wave drag. Further increasing the number of design variables from 20 to 30, the shape has significant changes but only decreases the drag by 0.5 counts. We recommend using as many design variables as can be computationally afforded to ensure the optimizer has enough control over the geometry to find the true optimum. In future work, we plan to extend HOEP to three dimensions and combine h adaptation with p adaptation to more efficiently adapt and test our strategy on more complicated cases such as multi-point or three-dimensional cases. Additionally, we would like to perform a more detailed study exploring different geometric parameterizations and number of design variables.

Acknowledgments

The first author is supported by the Department of Defense through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program and by the University of Michigan Rackham Merit Fellowship. He is also supported in part by the Michigan Institute for Computational Discovery and Engineering (MICDE) Graduate Fellowship. Additionally we would like to thank Eytan Adler and Andrew Lamkin for their insightful discussion on geometry and post-processing.

References

[1] Martins, J. R. R. A., "Perspectives on Aerodynamic Design Optimization," *AIAA SciTech Forum*, AIAA, Orlando, FL, 2020. doi:10.2514/6.2020-0043.

[2] Nemec, M., and Aftosmis, M., "Output error estimates and mesh refinement in aerodynamic shape optimization," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013, p. 865.

[3] Anderson, G. R., Nemec, M., and Aftosmis, M. J., "Aerodynamic shape optimization benchmarks with error control and automatic parameterization," *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1719.

[4] Hicken, J. E., and Alonso, J. J., "PDE-constrained optimization with error estimation and control," *Journal of Computational Physics*, Vol. 263, 2014, pp. 136–150. doi:10.1016/j.jcp.2013.12.050.

[5] Chen, G., and Fidkowski, K. J., "Discretization Error Control for Constrained Aerodynamic Shape Optimization," *Journal of Computational Physics*, Vol. 387, 2019, pp. 163–185. doi:10.1016/j.jcp.2019.02.038.

[6] Chen, G., and Fidkowski, K. J., "Variable-fidelity multipoint aerodynamic shape optimization with output-based adapted meshes," *Aerospace Science and Technology*, Vol. 105, 2020. doi:10.1016/j.ast.2020.106004.

[7] Li, D., and Hartmann, R., "Adjoint-based airfoil optimization with discretization error control," *International Journal for Numerical Methods in Fluids*, Vol. 77, No. 1, 2015, pp. 1–17.

[8] Fidkowski, K. J., and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694. doi:10.2514/1.J050073.

[9] Ibanez, D., Barral, N., Krakos, J., Loseille, A., Michal, T., and Park, M., "First benchmark of the Unstructured Grid Adaptation Working Group," *Procedia Engineering*, Vol. 203, 2017, pp. 154–166. doi:https://doi.org/10.1016/j.proeng.2017.09.800, 26th International Meshing Roundtable.

[10] Park, M. A., and Darmofal, D. L., "Parallel Anisotropic Tetrahedral Adaptation," AIAA Paper 2008-917, 2008.

[11] Michal, T., and Krakos, J., "Anisotropic Mesh Adaptation Through Edge Primitive Operations," AIAA Paper 2012–0159, 2012. doi:10.2514/6.2012-159.

[12] Caplan, P., Haimes, R., Darmofal, D., and Galbraith, M., "Extension of local cavity operators to 3d+t space-time mesh adaptation," Aiaa paper, 2019. doi:10.2514/6.2019-1992.

[13] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C., "Adaptive remeshing for compressible flow computations," *Journal of Computational Physics*, Vol. 72, 1987, pp. 449–466.

[14] Peraire, J., Peiró, J., and Morgan, K., "Adaptive Remeshing for Three-Dimensional Compressible Flow Computations," *Journal of Computational Physics*, Vol. 103, 1992, pp. 269–285.

[15] Borouchaki, H., George, P., Hecht, F., Laug, P., Mohammadi, B., and Saltel, E., "Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie II: Applications," INRIA-Rocquencourt, France. Tech Report No. 2760, 1995.

[16] Bassi, F., and Rebay, S., "High–order accurate discontinuous finite element solution of the 2-D Euler equations," *Journal of Computational Physics*, Vol. 138, 1997, pp. 251–285.

[17] Persson, P.-O., and Peraire, J., "Curved mesh generation and mesh refinement using Lagrangian solid mechanics," AIAA Paper 2009-0949, 2009.

[18] Toulorge, T., Geuzaine, C., Remacle, J.-F., and Lambrechts, J., "Robust untangling of curvilinear meshes," *Journal of Computational Physics*, Vol. 254, 2013, pp. 8–26.

[19] Karman, S. L., Erwin, J. T., Glasby, R. S., and Stefanski, D., "High-order mesh curving using WCN mesh optimization," *46th AIAA Fluid Dynamics Conference*, 2016, p. 3178.

[20] Ruiz-Gironés, E., Sarrate, J., and Roca, X., "Generation of curved high-order meshes with optimal quality and geometric accuracy," *Procedia engineering*, Vol. 163, 2016, pp. 315–327.

[21] Fidkowski, K. J., "Output-Based Mesh Optimization Using Metric-Conforming Node Movement," AIAA Paper 2023–2369, 2023. doi:10.2514/6.2023-2369.

[22] Coppeans, A. W., Fidkowski, K. J., and Martins, J. R. R. A., "Anisotropic Mesh Adaptation for High-Order Meshes in Two Dimensions," *AIAA SciTech Forum*, Orlando, FL, 2024. doi:10.2514/6.2024-1020.

[23] Shi, A., and Persson, P.-O., "Local element operations for curved simplex meshes," *International Journal for Numerical Methods in Engineering*, Vol. 125, No. 2, 2024, p. e7379.

[24] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. doi:10.1016/0021-9991(81)90128-5.

[25] Bassi, F., and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes, equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 197–207.

[26] Ceze, M., and Fidkowski, K., "Pseudo-transient Continuation, Solution Update Methods, and CFL Strategies for DG Discretizations of the RANS-SA Equations," *21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, 2013. doi:10.2514/6.2013-2686.

[27] Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869. doi:10.1137/0907058.

[28] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113. doi:10.1016/j.jcp.2005.01.005, URL http://www.sciencedirect.com/science/article/pii/S0021999105000185.

[29] Persson, P., and Peraire, J., "Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 2709–2733. doi:10.1137/070692108, URL https://epubs.siam.org/doi/abs/10.1137/070692108.

[30] Becker, R., and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[31] Sederberg, T. W., and Parry, S. R., "Free-form Deformation of Solid Geometric Models," *SIGGRAPH Comput. Graph.*, Vol. 20, No. 4, 1986, pp. 151–160. doi:10.1145/15886.15903.

[32] Hajdik, H. M., Yildirim, A., Wu, N., Brelje, B. J., Seraj, S., Mangano, M., Anibal, J. L., Jonsson, E., Adler, E. J., Mader, C. A., Kenway, G. K. W., and Martins, J. R. R. A., "pyGeo: A geometry package for multidisciplinary design optimization," *Journal of Open Source Software*, Vol. 8, No. 87, 2023, p. 5319. doi:10.21105/joss.05319.

[33] 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*, Vol. 59, No. 4, 2021, pp. 1151–1168. doi:10.2514/1.J059491.

[34] Luke, E., Collins, E., and Blades, E., "A Fast Mesh Deformation Method Using Explicit Interpolation," *Journal of Computational Physics*, Vol. 231, No. 2, 2012, pp. 586–601. doi:10.1016/j.jcp.2011.09.021.

[35] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131. doi:10.1137/S0036144504446096.

[36] Wu, N., Kenway, G., Mader, C. A., Jasa, J., and Martins, J. R. R. A., "pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, Vol. 5, No. 54, 2020, p. 2564. doi:10.21105/joss.02564.

[37] Johnen, A., Remacle, J.-F., and Geuzaine, C., "Geometrical validity of curvilinear finite elements," *Journal of Computational Physics*, Vol. 233, 2013, pp. 359–372.

[38] "Size gradation control of anisotropic meshes," *Finite Elements in Analysis and Design*, Vol. 46, No. 1, 2010, pp. 181–202. doi:https://doi.org/10.1016/j.finel.2009.06.028, mesh Generation - Applications and Adaptation.

[39] Fidkowski, K. J., "A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization," AIAA Paper 2016–0835, 2016. doi:10.2514/6.2016-0835.

[40] Yano, M., "An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[41] Caplan, P. C., Haimes, R., Darmofal, D. L., and Galbraith, M. C., "Four-Dimensional Anisotropic Mesh Adaptation," *Computer-Aided Design*, Vol. 129, 2020, p. 102915. doi:https://doi.org/10.1016/j.cad.2020.102915.

[42] Bossen, F. J., and Heckbert, P. S., "A pliant method for anisotropic mesh generation," *5th Intl. Meshing Roundtable*, Vol. 63, Citeseer, 1996, p. 76.

[43] Wu, N., Mader, C. A., and Martins, J. R. R. A., "A Gradient-based Sequential Multifidelity Approach to Multidisciplinary Design Optimization," *Structural and Multidisciplinary Optimization*, Vol. 65, 2022, pp. 131–151. doi:10.1007/s00158-022-03204-1.

[44] Coppeans, A. W. C., Fidkowski, K. J., and Martins, J. R. R. A., "Comparison of Finite Volume and High-Order Discontinuous Galerkin Based Aerodynamic Shape Optimization," *AIAA SciTech Forum*, National Harbor, MD, 2023. doi:10.2514/6.2023-1845.

[45] Giles, M. B., and Pierce, N. A., "Adjoint equations in CFD: duality, boundary conditions and solution behavior," AIAA Paper 97-1850, 1997.