Anisotropic Mesh Adaptation for High-Order Meshes in Two Dimensions

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

We present a novel method for performing anisotropic mesh adaptation on high-order curved meshes. This process utilizes existing edge primitive operations for mesh adaptation methods and extends them to high-order meshes. Traditionally, curved meshes are made linear and then adapted and re-curved at the end of the adaptation process. Our method keeps the mesh valid and curved during the entire adaptation process leading to more robust mesh adaptation. Results for four test cases demonstrate the ability of this method to produce valid curved meshes with a variety of different aspect ratio's. The output error convergence for these cases are comparable to a global re-meshing strategy.

I. Introduction

Error estimation and mesh adaptation are important tools for having a fully automated robust and accurate computational fluid dynamics (CFD) simulation [1]. Unstructured meshes in particular are well suited for mesh adaptation as the initial mesh can be easily generated and then be adapted. For unstructured meshes to take advantage of mesh adaptation, it is important to be able to generate a mesh that conforms to a given metric field [2–4] which specifies a desired stretching and orientation of elements throughout the domain. There are many ways of generating a metric conforming mesh such as global re-meshing, local mesh operations, global node movement. Metric-conforming re-meshing has been shown to be highly effective in two dimension [5, 6] but is not always tractable or robust, especially for three-dimensional meshes with curved, anisotropic elements. Local mesh modification operators have been shown to work in two, three, and even four dimensions [7–12]. These local mesh modifications are typically done through either explicitly, known as edge primitive operations, or through general cavity operators. Edge primitive operators [7–9] explicitly handle each operation as edge splits, edge collapses, edge swaps, face swaps, etc. The general cavity operator is a general local mesh modification operation that combines traditional edge primitive operations into one general operator [10, 11]. Recently work has been done on implementing mesh optimization operators, swap and node smoothing, on high-order meshes [13, 14] and previous work has implemented curved edge split and collapse [15]. However, anisotropic mesh adaptation for curved meshes still has remained a challenge. Another option is node movement, also known as r-refinement [16–18], which moves the nodes to adapt the mesh. More recently, node movement has been used to move mesh nodes to conform to a target metric field [19]. This has shown promise as small changes that might cause element inversion with local operators can be fixed by moving the nodes on edges that are "trapped" and not metric conforming.

High-order simulations not only require an anisotropic mesh to not waste degrees of freedom they also require that faces on the geometry are curved [20]. Currently, the state of the art is to generate a linear mesh and curve it to represent the geometry. Two solutions are commonly used to curve a mesh. The first involves solving an analogous physics-based problem [21, 22] to transfer boundary displacements to the interior. The second involves solving an optimization problem [23–25] that simultaneously moves both boundary and interior vertices. However, these approaches do not always work especially for complex geometries and meshes with highly anisotropic elements.

Here we introduce a method that extends traditional edge collapse and split to adapt arbitrary order curved meshes. This adaptation approach is presented with test cases using a high-order discontinuous Galerkin finite-element discretization, which is briefly reviewed in Section II. Section III presents the output error estimation procedure, and Section IV presents the metric optimization algorithm, MOESS. Section V presents background on metric based mesh

*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.

adaptation and Section VI presents our approach for high-order mesh adaptation. Results for four test cases are shown in Section VII where we demonstrate our methods ability to generate highly anisotropic curved meshes and get comparable output error convergence to global re-meshing. Section VIII conclude with a summary of the work and future work directions.

II. Discretization

A. Governing Equations

CFD discretizing a set of partial differential equations to model the fluid. Consider a system of unsteady partial differential equations written in conservative form,

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

where $\mathbf{u} \in \mathbb{R}^s$ is the s-component state vector, $\vec{\mathbf{F}} \in \mathbb{R}^{d\times s}$ is the flux vector, d is the number of spatial dimensions, and $\mathbf{S} \in \mathbb{R}^s$ is the source term arising from the turbulence model. For the Navier-Stokes and Euler equations, we use a conservative state vector $\mathbf{u} \in \mathbb{R}^{d+2} = [\rho, \rho\vec{V}, \rho E]$, where ρ is the density, $\vec{V}$ is the velocity, and E is the total energy per unit mass. The source term is nonzero when modeling turbulence using Reynolds averaging, for which an extra equation is included [26, 27].

B. The Discontinuous Galerkin Method

The discontinuous Galerkin (DG) method [28–30] is a finite-element method in which the approximation space is discontinuous across element boundaries. The state is approximated on each element, Ω_e, with polynomials of order p on mesh of non-overlapping elements. Multiplying Equation (1) by test functions in the same space as the solution and integrating by parts to couple elements via fluxes, yields the weak form. Here we use the Roe [31] convective flux and the second form of Bassi and Rebay (BR2) [32] for viscous treatment. This yields the following semi-discrete equation,

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

where $\mathbf{U} \in \mathbb{R}^N$ 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. We neglect the time derivative term for steady simulations, which yields the system of nonlinear algebraic equations, $\mathbf{R} = \mathbf{0}$. To solve this system of non-linear equations we the Newton–Raphson method with pseudo-time continuation [33] and the generalized minimum residual (GMRES) linear solver [34], preconditioned by an element-line Jacobi smoother with coarse-level ($p = 1$) correction [35, 36].

C. The Output Adjoint

The output adjoint for a scalar output is the sensitivity of the output to residual source perturbations [1, 37]. In this work we use the steady adjoint, where we start by linearizing the residual and scalar output, $J(\mathbf{U})$, and solve the following linear system,

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

where $\mathbf{\Psi} \in \mathbb{R}^N$, is the discrete adjoint vector. This equation is solved using a transposed version of the preconditioned-GMRES used in the primal solver. The adjoint vector plays a key role in output-based error estimation and mesh adaptation because discretization errors originate as residual sources and the adjoint weights those sources to determine their effect on the output.

III. Output Error Estimation

We use the adjoint solution to estimate the numerical error in the corresponding output of interest, J, through the adjoint-weighted residual [1, 37] The adjoint-weighted residual computes the discretization error between the current coarse discretization space, H, and a finer discretization space, h, here obtained by increasing the approximation order by one, $p \rightarrow p+1$. The state vector is first prolonged from the coarse space to the fine space to get a new state $\mathbf{U}_h^H$. Next, the prolonged state is used to compute the fine-space residual, $\mathbf{R}_h(\mathbf{U}_h^H)$, and weighting this residual with the fine-space adjoint error gives an estimate for the output error between the coarse and fine spaces,

$$\delta J = J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx -\delta\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H). \tag{4}$$

The adjoint error, $\delta\boldsymbol{\Psi}_h$, is obtained by subtracting from the fine-space adjoint, solved exactly, its projection onto the coarse-space. The error estimate in Equation 4 is localized to elemental contributions, resulting in the elemental error indicators ε_e,

$$\delta J \approx \sum_{e=1}^{N_e} \delta\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \quad \Rightarrow \quad \varepsilon_e \equiv \left|\delta\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)\right|, \tag{5}$$

where N_e is the number of elements, the subscript e denotes components of the adjoint and residual vectors associated with element e.

IV. Metric Optimization

The elemental error indicator combined with additional error indicators evaluated on sub elements, via Equation 5 with projected adjoints, drive a metric optimization calculatin based on MOESS: mesh optimization through error sampling and synthesis [5, 38]. The optimized metric can be used in mesh adaptation through a variety of means such as re-meshing, local mesh operations, or node movement. Metric optimization uses a Riemannian metric field, $\mathcal{M}(\vec{x})$, to encode information about desired size and stretching of elements in a computational mesh. At each point $\vec{x}$ in physical space, a symmetric positive definite metric tensor $\mathcal{M}(\vec{x}) \in \mathbb{R}^{d\times d}$ provides a yardstick for measuring a non-dimension distance from $\vec{x}$ to a nearby, infinitesimally-close, $\vec{x} + \delta\vec{x}$,

$$\delta l = \sqrt{\delta\vec{x}^T \mathcal{M} \delta\vec{x}}. \tag{6}$$

The set of points unit measure from $\vec{x}$ is an ellipse, the principal axes and lengths of which are determined by the eigenvectors and eigenvalues respectively of $\mathcal{M}$.

Affine-invariant changes [39] to the metric field are made via a symmetric *step matrix*, $\mathcal{S} \in \mathbb{R}^{d\times d}$, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S})\, \mathcal{M}_0^{\frac{1}{2}} \tag{7}$$

Note that $\mathcal{S} = 0$ leaves the metric unchanged, while diagonal values in $\mathcal{S}$ change the metric stretching sizes.

The metric optimization algorithm requires models for how the elemental error indicator and cost change as the metric change. The elemental error model represents how the elemental error of an element changes if we were to apply a step matrix change to that elements. The model used here it

$$\varepsilon_e = \varepsilon_{e0} \exp\left[\mathrm{tr}(\mathcal{R}_e \mathcal{S}_e)\right], \tag{8}$$

where ε_{e0} is the initial error on element e, and $\mathcal{R}_e$ is an element-specific rate tensor determined through a sampling procedure. The cost model represents how much the cost, in this case degrees of freedom, would change as we apply a step matrix to an element. The cost model used is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right], \tag{9}$$

where C_{e0} is the initial cost, measured in degrees of freedom. This model is purely volume based: the trace of the step matrix governs changes to the element's volume, and hence the number of degrees of freedom occupying the original volume. More details of these models can be found in previous work [5].

MOESS takes in a current mesh with its mesh-implied metric, $\mathcal{M}(\vec{x})$, elemental error indicators, ε_{e0}, and elemental rate tensor estimate, $\mathcal{R}_e$, and outputs a step matrix field $\mathcal{S}(\vec{x})$ The goal of MOESS is to produce the step matrix field that minimizes the error at a given cost by iteratively equidistributing the marginal error to marginal cost ratio over elements in the mesh. In practice, mesh optimization and flow/adjoint solutions are performed several times at a given target cost, C_{target}, until the error stops changing. The target cost is then increased to reduce error further if it is not at an acceptable level.

V. Metric Based Mesh Adaptation

The output from MOESS is a step matrix field that can be used to calculate the desired metric field using Equation 7, which can then be used for mesh adaptation. One approach is to use this metric field along with the current mesh as a background mesh and globally re-mesh the domain to generate a new mesh that conforms to the target metric field. This approach has been successful in two-dimensions but has not is not robust in three-dimensions with complex geometries and highly anisotropic metric fields.

A more common approach is to perform local mesh operations on the input mesh to produce a new mesh that conforms to the desired metric. This approach works by refining regions where the edges under the desired metric field are too long and coarsening regions where the edge length is too short. With the metric field prescribed at each vertex in the input mesh, the length of each edge under the metric field, $L_{m,e}$, can be computed. We follow the conventions of the Unstructured Grid Adaptation Working Group (UGAWG) [4] and compute the length of an edge e between vertices a and b,

$$L_{m,e} = \begin{cases} \frac{L_a - L_b}{\log(L_a/L_b)} & |L_a - L_b| > 0.001 \\ \frac{L_a + L_b}{2} & \text{else} \end{cases} \tag{10}$$

Here L_a and L_b are the length of the edge under the constant metric at vertex a and b respectively given as,

$$L_a = (\vec{v}_e^T \mathcal{M}_a \vec{v}_e) \quad \text{and} \quad L_b = (\vec{v}_e^T \mathcal{M}_b \vec{v}_e), \tag{11}$$

where $\vec{v}_e$ is the vector along edge e, $\vec{v}_e = \vec{x}_b - \vec{x}_a$. The ideal length in a metric conforming mesh is one however this constraint is relaxed and we target a quasi-unit length range $L_{\text{low}} \leq L_m \leq L_{\text{up}}$. Typically $L_{\text{low}} = \frac{\sqrt{2}}{2}$ and $L_{\text{up}} = \sqrt{2}$. The adaptation procedure iteratively adapts the mesh by splitting edges that are too long and collapsing edges that are too short. In addition to computing edge lengths, element volumes can also be computed under the prescribed metric. The volume of an element k is given as,

$$V_{m,k} \approx \sqrt{\det \mathcal{M}_{\max}} V_k \tag{12}$$

where $V_{m,k}$ is the volume of element k under the metric, V_k is the Euclidean volume of the element, and $\mathcal{M}_{\max}$ is the metric of vertex on element k with the largest determinant.

$$\mathcal{M}_{\max} = \arg \max_{\mathcal{M}_i, i \in k} \det \mathcal{M}_i \tag{13}$$

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

$$Q_k = \frac{\left(\frac{V_{m,k}}{\hat{V}_k}\right)}{\frac{1}{3}\sum_{e \in k} \vec{v}_e^T \mathcal{M}_{\max} \vec{v}_e} \tag{14}$$

where, $\hat{V}k$, is the area of a unit-equilateral triangle, $\hat{V}_k = \frac{\sqrt{3}}{4}$. The element quality Q_k measures how closely to equilateral the triangle is in the transformed metric space. We use node movement to improve node quality but edge swap operators have been shown to improve mesh quality.

VI. High-Order Mesh Adaptation

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

$$\vec{x}(\xi, \eta) = \sum_i^{N_Q} \vec{x}_i \phi_i(\xi, \eta), \tag{15}$$

where (ξ, η) are the reference space coordinates, N_Q is the number of interpolating polynomials, and $\vec{x}_i$ is node the coordinate in physical space associated with the ith interpolating polynomial ϕ_i. The mapping jacobian, $\underline{\text{J}}$, is defined as,

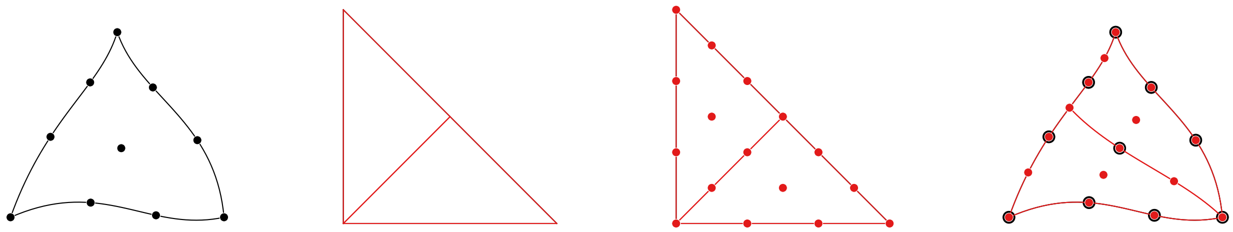
$$\underline{\text{J}} = \sum_i^{N_Q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{xi}}(\xi, \eta). \tag{16}$$

The determinant of the mapping jacobian matrix is required to be positive at all points for a valid mesh. Presently, this constraint is checked at quadrature points inside the element during the adaptation but points depend on the quadrature rule chosen. However, it has been shown it is possible to reliably detect invalid elements using Bézier functions [40]. This approach is more robust and may be required for more complicated cases than the ones presented here or in three dimensions.

While we do our mesh adaptation on curved meshes that remain curved throughout the adaptation procedure, we still use linear metric edge length calculations and element quality calculations. That is, when we compute a metric edge length we use Equation 10 and do not consider the shape of the curved edge and similarly with element quality we use Equation 13 and only consider the 3 nodes that make up the linear element.

A. Edge Split

For mesh refinement we split edges. Since we use Lagrange basis functions for our mesh subdividing elements is performed in each elements reference space. High-order node positions of child elements are first computed in the parent elements reference space and then mapped to global space using the parent element's basis functions. This process is shown in Figure 1.



(a) High-order element in global space

(b) Split reference space element.

(c) High-order node positions of child elements.

(d) Global space split elements.

Fig. 1 Edge split for high-order elements.

When splitting edges on boundaries the newly created nodes need to be projected onto the true geometry. Because we are working with an already curved mesh this process is easy as the displacement of each node on to the geometry is small. We then use linear basis functions to transfer the deformations of boundary nodes to element interior nodes. Nodes on non boundary edges are not moved. Other than limiting splits that produce elements that have negative jacobian checks, we also reject splits when the they would create edges that are shorter than the shortest edge in the current mesh. We also reject splits when they would create an element with lower quality than the lowest quality element in the current mesh.

B. Edge Collapse

The edge collapse operation is used to coarsen the mesh and targets edges shorter than $L_m < \frac{\sqrt{2}}{2}$ For edge collapses of high-order elements it is not obvious where to place high-order nodes to keep a valid mesh. Once an edge is flagged for collapsing we look at the two vertices, a and b, that make up the endpoints. We consider two possible collapses, the first collapsing vertex a into vertex b removing vertex a from the mesh, and the second option of collapsing vertex b into vertex a. We look at the quality of the linear elements that would be generated from these two possibilities and pick the one with the higher minimum quality in the affected elements. From here we will use denote node a as the node that is collapsed and removed and node b as the node that is kept in the mesh. If collapsing node a into node b fails for any reason listed below we swap a and b and try again.

We follow a cavity operator like approach with the collapse. That is we look at all the elements that contain and form a cavity about node a and determine if the collapse is valid before applying changes to the mesh. We only modify mesh nodes inside the cavity while the boundary of the cavity remains fixed, except when we collapse a boundary edge and nodes need to be projected to geometric boundaries. Elements that only contain node a are modified and kept in the mesh while elements that contain both a and b are removed.

The first step is to perform the collapse on the linear representation of the elements. During this step we check that all the linear elements have positive volume. This is quick check and any if any element has negative area the collapse is immediately rejected. At this point we also do checks to make sure we do not reduce the minimum quality in the mesh or increase the maximum edge length in the mesh on the proposed linear elements. If these checks fail the collapse is rejected and we do not modify the mesh. This collapse of the linear elements in shown in Figure 2b.

We then sample the high-order node positions on these new $q = 1$ elements. Because the boundary of the cavity does not change all high-order nodes on the boundary of the cavity are displaced back to their original position. Additionally, nodes on geometric boundaries are projected to the boundary. The boundary of the cavity is now set to matches the original curved boundary of cavity and the geometry. This projection step is shown in Figure 2c. Edges inside of the cavity are still linear and elements are not necessarily valid. However, here we do a negative jacobian check at quadrature points to see if we have a valid mesh. If the mesh with just the boundary displacements is valid elements are modified and the edge is removed from the mesh.

In most cases the mesh will not be valid after moving the high-order boundary nodes. In order to solve this problem we use an explicit inverse-distance mesh deformation technique [41, 42] where we propagate the displacement of high-order nodes on the cavity boundary to the interior of the cavity. This propagation of deformations to interior nodes in the cavity is shown in Figure 2c. This idea is similar to solving analogous physics-based equations to re-curve a linear mesh, except boundary displacement are not necessarily on geometric boundaries and we only update small portion of elements in the mesh. The boundary displacements are computed as the change in node position their positions the linear element to the position on the cavity boundary or geometric boundary. The displacement of a single interior node in the cavity is given as,

$$\Delta\vec{x} = \vec{x} - \vec{x}_0 = \frac{\sum_i^{N_b} w_i(\vec{x}_0)\Delta\vec{x}_{b,i}}{\sum_i^{N_b} w_i(\vec{x}_0)}, \tag{17}$$

where $\vec{x}_0$ is the original location of the interior nodes, N_b is the number of nodes on the cavity 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 the distance between the interior node and the boundary node,

$$w_i(\vec{x}_0) = \left[\left(\frac{1}{\left|\vec{x}_0 - \vec{x}_{b,i}\right|}\right)^a \left(\frac{\alpha}{\left|\vec{x}_0 - \vec{x}_{b,i}\right|}\right)^b\right] \tag{18}$$

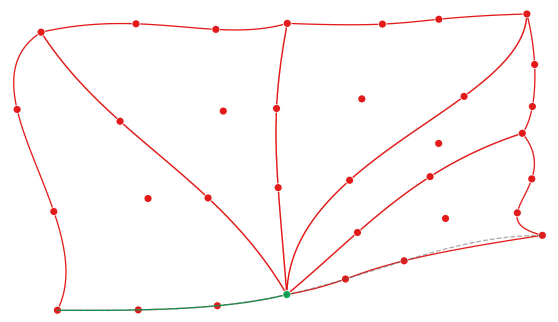
Once the displacements for all the interior nodes in the cavity are computed we check perform another check on jacobian determinants at quadrature points inside each element. If a single point has a negative jacobian the collapse is immediately rejected. This method allows us to always have a valid high-order mesh during collapses. Other mesh deformation techniques could be used to move the nodes inside the cavity but this method is explicit and cheap to compute.

Figure 2 shows this process. The initial mesh is shown in Figure 2a with the edge being collapsed and the node being removed highlighted in green. The linear collapse and re-sampling of high-order nodes is shown in Figure 2b. The displacement of nodes on the boundary of the cavity to the geometry and to the original cavity shape is shown in Figure 2c. At this step the elements are checked for negative jacobian determinants and if there are none the mesh is updated. If this step fails the displacements shown in Figure 2c are propagated to interior nodes using the inverse distance method. Those displacements are shown in Figure 2d and a final negative jacobian check is performed. If this check passes with the mesh is updated shown in Figure 2e.

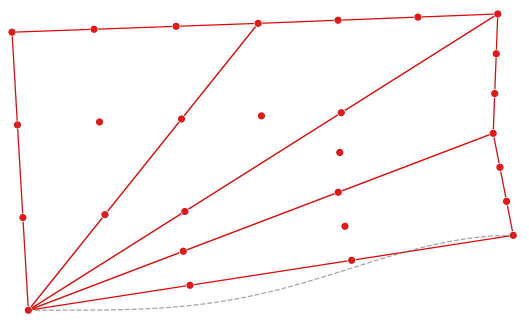
C. Metric Conforming Node Movement

As an optional smoothing step we use metric-conforming node movement detailed in [19] and summarized below. Given a target step matrix field, $\mathcal{S}^{\text{tgt}}(\vec{x})$, we try to optimize the node coordinates of the current mesh so that the step matrix between the current mesh and the new mesh $\mathcal{S}(\vec{x})$, closely matches $\mathcal{S}^{\text{tgt}}$. The connectivity remains constant during the optimization process. The objective function we try to optimize is given as,

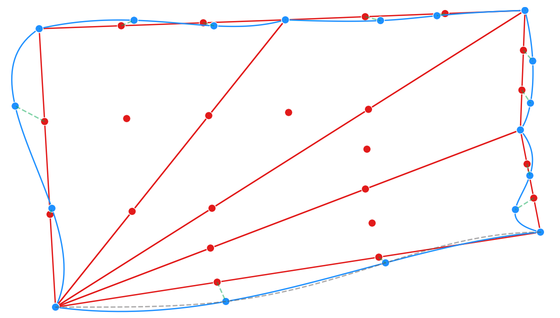
$$\Delta\mathbf{x} = \arg\min_{\Delta\mathbf{x}} J^{\text{metric}}(\Delta\text{x}) \equiv \sum_{e=1}^{N_e} \frac{1}{2}\left\|\mathcal{W}_e \circ (\mathcal{S}_e(\Delta\mathbf{x}) - \mathcal{S}_e^{\text{tgt}})\right\|_F^2 + \sum_{f=1}^{N_f} \frac{\gamma}{2}\left\|\mathcal{S}_{f^+}^{I}(\Delta\mathbf{x}) - \mathcal{S}_{f^-}^{I}(\Delta\mathbf{x})\right\|, \tag{19}$$



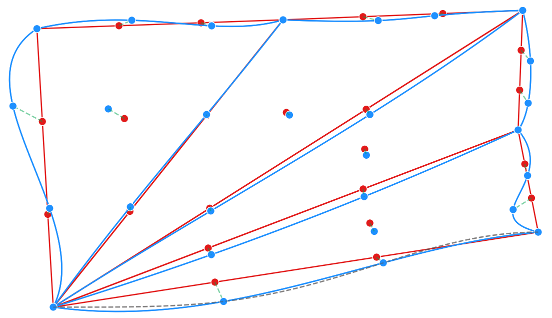
(a) Cavity of $q = 3$ elements affected by collapsing the green edge removing the green vertex.



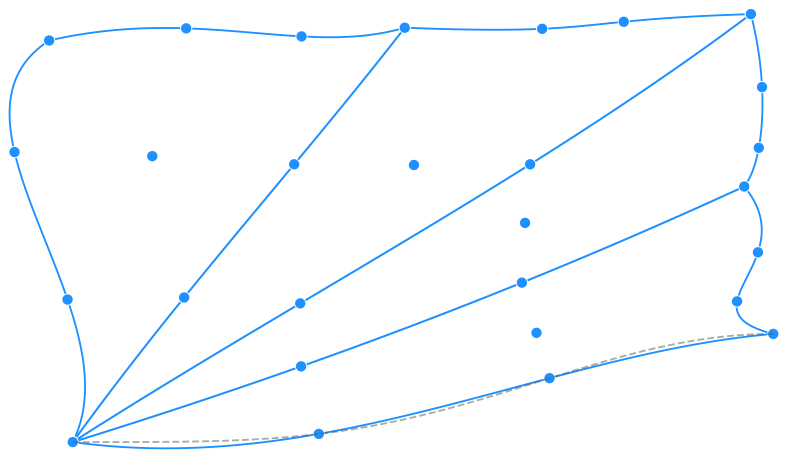
(b) Collapsing the edge on the linear elements with high-order nodes sampled on the linear element.



(c) Projecting the boundary nodes back to their original positions and to the geometry (grey).



(d) Displacements of boundary nodes are propagated to interior nodes through inverse distance method.



(e) Final valid curved mesh after edge collapse.

Fig. 2 Edge collapse process for high-order elements.

where N_e is the number of elements, $||\cdot||_F$ is the Frobenius norm, N_f is the number of faces, and f^+/f^- denote the two adjacent elements to face f. For an element e, $\mathcal{S}_e^I$ is the step matrix between the identity matrix, i.e. reference element, and the new metric, which reduces to the logarithm of the new metric, $\mathcal{S}_e^I = \mathcal{S}(\mathcal{I}, \mathcal{M}_e(\Delta\mathbf{x})) = \log(\mathcal{M}_e(\Delta\mathbf{x}))$. The first term in Equation 19 is a measure of the error between the current step matrix and the target step matrix on each element. The second term in Equation 19 consists of a sum over faces of the step matrix difference between adjacent elements. This term helps prevent large changes in element size and shape and is controlled by the non-dimensional parameter γ which is set to $\gamma = 0.03$, determined through numerical experimentation. The tensor $\mathcal{W}_e$ in Equation 19 is an optional weight on the step matrix errors. The Hadamard product, $\circ$, applies the weight component-wise. In this work, when using MOESS, we use the normalized linearization of the localized output error with respect to the step matrix, $\mathcal{W}_e = \frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$, which is available from MOESS. The optimization is solved using the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [43].

Because we rely on edge splitting and collapsing to do the majority of the adaptation the node movement optimization is ran once at the end of the adaptation process to improve mesh quality.

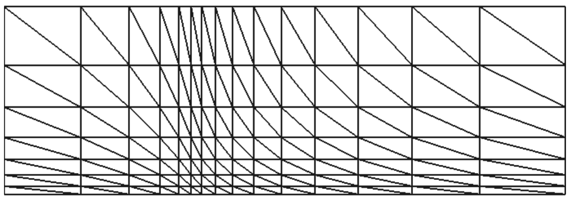
D. Adaptation Algorithm

Our adaptation algorithm consists of iteratively collapsing edges shorter than L_{low} and splitting edges longer than L_{up} throughout the mesh. We follow an operator schedule similar to the one proposed by Caplan [44]. We break each adaptation pass into two phases, the first phase sets the split target length $L_{\text{up}} = 2$ and the second phase consists of setting the split target length $L_{\text{up}} = \sqrt{2}$. Each phase sets the target length for collapses to $L_{\text{low}} = \sqrt{2}/2$ and each phase is performed twice. Therefore, each adaptation pass performs 4 split passes and 4 collapse passes. We perform the adaptation pass 5 times or until a pass does not perform any splits or collapses. Finally there are limits the length of edges created to always be within the current the range $L_{\min} \leq L_m \leq L_{\max}$, where $L_{\min}$ and $L_{\max}$ are the minimum and maximum edge lengths in the mesh respectively. We also impose the same restriction on quality, that is the minimum quality of an element in the mesh can never decrease. After all the edge split and collapse passes are done we optionally run the node optimization algorithm which our results show acts as a smoothing step to further improve metric conformity and increase element quality.

VII. Results

A. Laminar Flat Plate

The first test case we ran was the laminar flow over a flat plate. The flat plate is unit length and starts one third of the way down stream from the inflow. The boundary conditions are a non-slip adiabatic wall on the plate, total temperature and pressure at the inflow, static-pressure at the outflow and top boundary, and symmetry on the bottom boundary ahead of the plate. The Mach number is set to $M_\infty = 0.5$, and the Reynolds number based on the unit distance along the flat plate is $Re = 10^4$. We started with a mesh of 210 linear triangles, shown in Figure 3a, and used a solution order of $p = 3$ yielding 2,100 degrees of freedom.


(a) Initial mesh (210 elements)

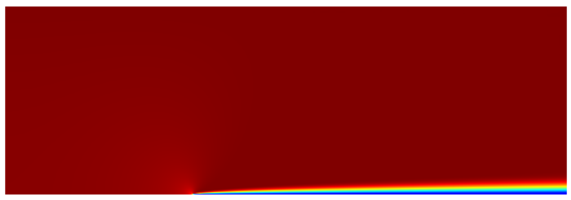
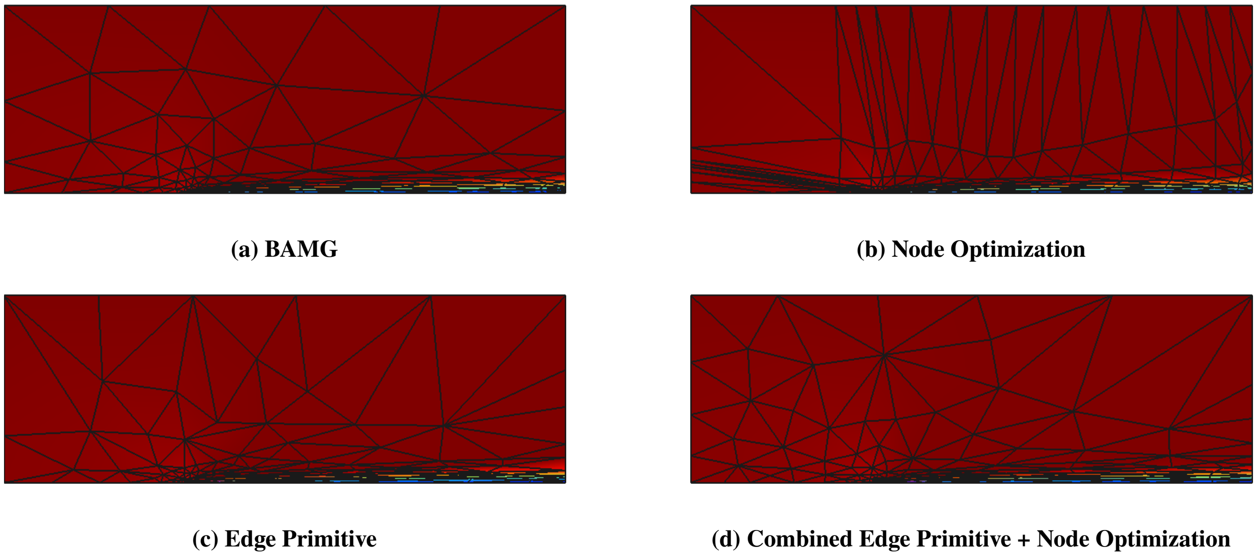

(b) Mach contours from "exact" solution (0-0.5)

Fig. 3 Initial mesh and Mach contours for laminar flat plate case.

For our adaptive cases we used the drag on the plate as our output and set the target cost in MOESS to 2, 100 degrees of freedom. We ran 20 adaptation iterations using global re-meshing using BAMG [45], node-movement, edge-primitive operations, and a combined approach using both edge-primitive and node movement for adaptation. The combined approach ran node movement after the edge-primitive adaptation was complete in each iteration. To compute error after each adaptation iteration we compared the drag to a truth solution obtained by using 20 iterations of BAMG at 10, 000 degrees of freedom, contours are shown in Figure 3b. The meshes at the end of each adaptation iteration are shown below in Figure 4. Here we see all adaptation mechanism move degrees of freedom into the boundary layer region on


(a) BAMG

(b) Node Optimization

(c) Edge Primitive

(d) Combined Edge Primitive + Node Optimization

Fig. 4 Adapted meshes for laminar flat plate case with Mach contours (0-0.5).

the plate. Node optimization is limited due to the fixed topology during the adaptation process. This leads to wasted degrees of freedom in regions away from the plate, where the output is less affected, and not enough degrees of freedom in the region directly next to the plate. When comparing our edge primitive adaptation with BAMG we can see the mesh from BAMG is noticeably smoother and our mesh we have some anisotropic elements, which have lower element quality, away from the plates. When combining in node optimization with edge primitive adaptation we see that there are still large elements far from the plate, but now there are fewer anisotropic elements. This shows that the node optimization step acted as a smoothing step improving element quality in the domain. When looking at the minimum and average quality in the edge primitive and combined cases, we see the minimum quality improves from 0.30 to 0.49 and the average quality in the mesh increases from 0.77 to 0.85. We also see that the number of edges outside the quasi-unit range decreases from 43 to just 13. This shows that we can use edge primitive operations for large mesh topology changes and use node optimization as a smoothing step to increase mesh quality and further improve metric conformity.

The error convergence for this case is shown in Figure 5. The convergence for both th edge primitive and the combined approach cases are very noisy this is due to large discrete changes in mesh topology between adaptive iteration. Here we see BAMG converges quite quickly and is not as noisy as the other cases. Additionally, we are seeing that the combined approach is able to reach similar drag error as BAMG, while edge primitive and node optimization alone do not. For this case edge primitive adaptation has similar final error as node movement, which is surprising due to the fact that edge primitive operations are able to re-distribute degrees of freedom in the domain. However, because we are using high order polynomials, $p = 3$, the initial topology was not as much of a limit for node optimization because the initial mesh had enough elements on the plate.

B. Inviscid Transonic NACA 0012

The second test case consists of the NACA 0012 airfoil in inviscid flow at a free-stream Mach number of $M_\infty = 0.8$ and an angle of attack of $\alpha = 1.25°$. The initial mesh is shown in Figure 6 and consists of 533 cubic, $q = 3$, elements

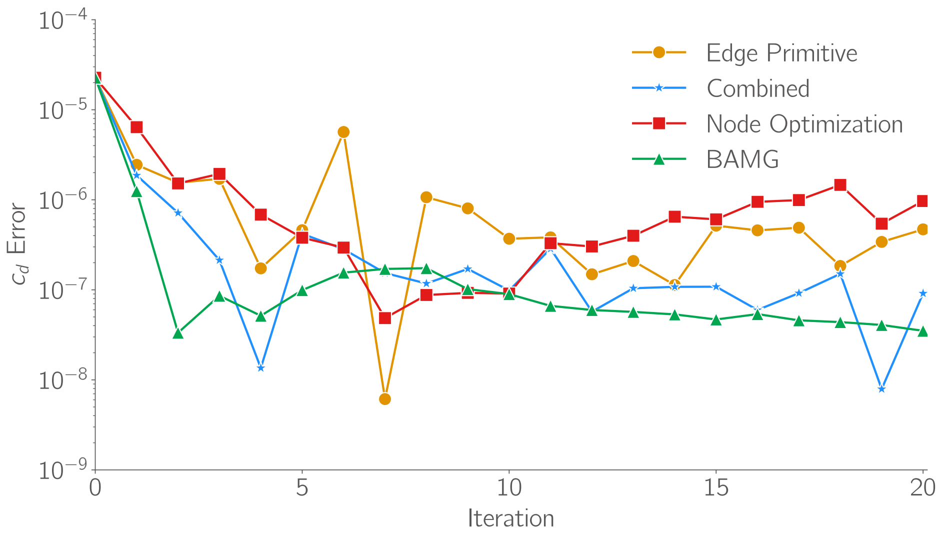
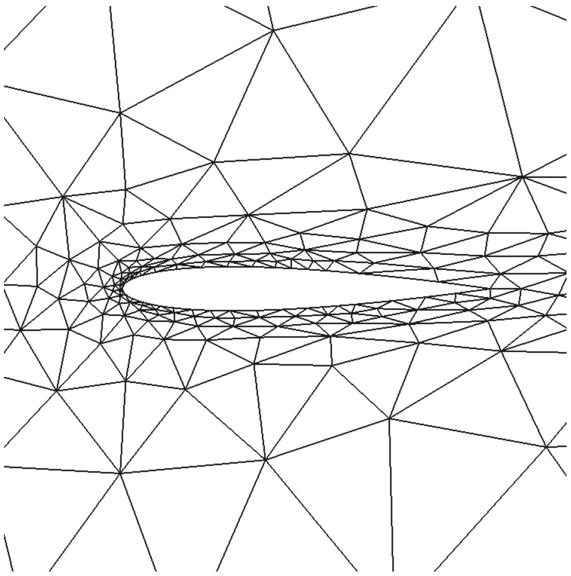
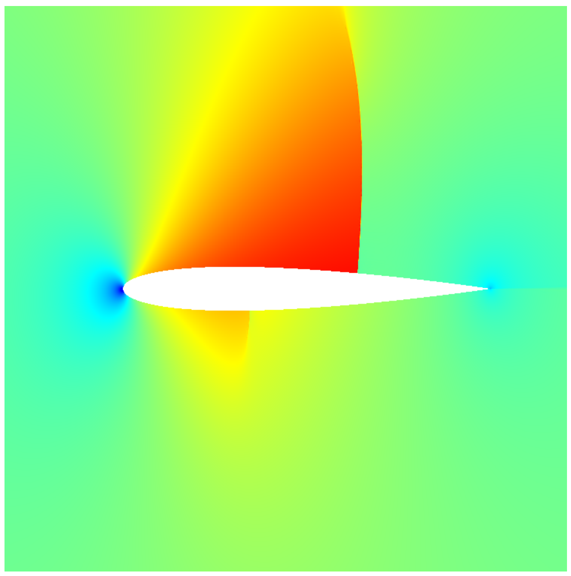


Fig. 5 Output error convergence for flat plate case

curved to the geometry. The solution approximation order is set to $p = 2$, and element-wise constant artificial-viscosity is used to capture the shocks.



(a) Initial mesh $q = 3$ (533 elements)



(b) Mach contours (0-0.5)

Fig. 6 Initial mesh and Mach contours for inviscid transonic NACA 0012 case.

We perform 20 adaptation iterations with a set target cost of 6,000 degrees of freedom. This requires our adaptation process to nearly double the number of elements in the mesh, for this reason we don't run node optimization by itself. We compare re-meshing with BAMG, our high-order edge primitive operation adaptation, and a combined approach with edge primitive adaptation with node optimization at the end of each adaptation iteration. This case demonstrates our ability to perform edge splits and collapses on curved elements while maintaining a valid curved mesh during the entire process. With BAMG we re-curve the mesh using linear elasticity. We compare the solutions at each adaptation iteration with a truth solution, contours shown in Figure 6b obtained from using BAMG with 50,000 degrees of freedom.

The final meshes after the 20 adaptations iterations finished are shown in Figure 7. Here we see the all adaptation

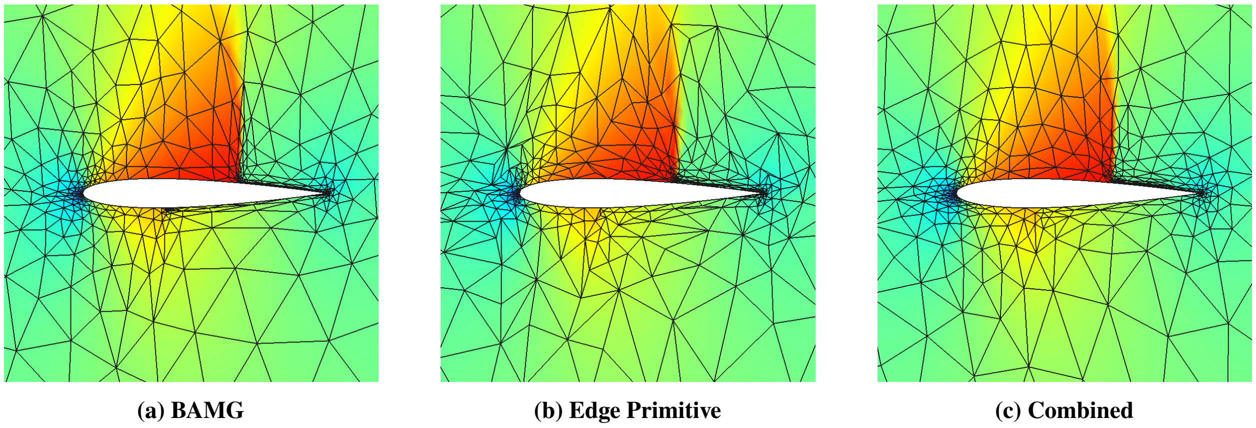

(a) BAMG (b) Edge Primitive (c) Combined

Fig. 7 Adapted meshes for transonic NACA 0012 case with Mach contours (0-1.6).

mechanisms adapt the leading edge, trailing edge and shock on the upper surface. BAMG does a better job capturing the weak shock on the lower surface however, this does not significantly improve the output error. We see that edge primitive operations alone leave a mesh with many sliver elements in regions where isotropic elements would be okay leading to a lower quality mesh. Additionally, because there sliver elements are not in regions that are not as impactful on the output degrees of freedom are being wasted. We see that when combining edge primitive operations with the node movement the mesh looks much smoother and there are fewer sliver elements off of the airfoil. Again comparing the quality at the last iteration between the edge primitive only and the combined approach we see that the minimum quality improves from 0.09 to 0.25. We also see that the average quality improves from 0.77 to 0.87 and the number of edges outside the quasi-unit range decreases from 161 to 76.

These mesh quality improvements lead to slightly better output convergence but due to the presence of shocks, the coarse mesh, and the artificial-viscosity shock capturing the adaptations are noisy and are not monotonic. The output convergence is shown in Figure 8. BAMG has the least noisy adaptation while the cases that use edge primitive operations to adapt are much more noisy due to more discrete changes that lead to the shock jumping. However, here we do see that we are able to get similar final output error as BAMG without having to re-curve the mesh each adaptation iteration.

C. Viscous Laminar NACA 0012

The third case consists of viscous flow over a NACA 0012 airfoil at a Mach number of $M_\infty = 0.5$, an angle of attack of $\alpha = 2°$, and a Reynolds number of $Re = 5000$. We use the same initial starting mesh as the previous case, shown in Figure 6, with a $p = 2$ solution approximation. Mach number contours for this case are shown in Figure 9.

We perform 20 adaptation iterations at a target cost of 15,000 with drag on the airfoil as the target output. The adaptation is again performed using global re-meshing with BAMG, our high-order edge primitive adaptation, and our combined approach with edge primitive operations and node optimization. We look at the error in drag during each adaptive iteration and compare it to a truth solution obtained by a solution with 50,000 degrees of freedom. The final meshes of the adaptation are shown in Figure 10

This case has some anisotropic elements with elements in each mesh with a maximum aspect ratio $\mathcal{AR} \sim O(100)$. Looking at the meshes we see the similar trends as previous cases. The BAMG mesh targets the boundary layer and the region outside the boundary layer around the airfoil consists of mostly isotropic elements with smooth size transitions between them. The edge primitive only adaptation on the other hand still targets smaller and more anisotropic elements in the boundary layer but has many sliver elements where BAMG has isotropic elements. The combined approach achieves a mesh that looks very similar to the mesh produced by BAMG with mostly isotropic elements outside the boundary layer. Looking at quality measures for the edge primitive only and the combined approach show that minimum quality improves from 0.19 to 0.27 and the average quality improves from 0.82 to 0.86. The node optimization step also

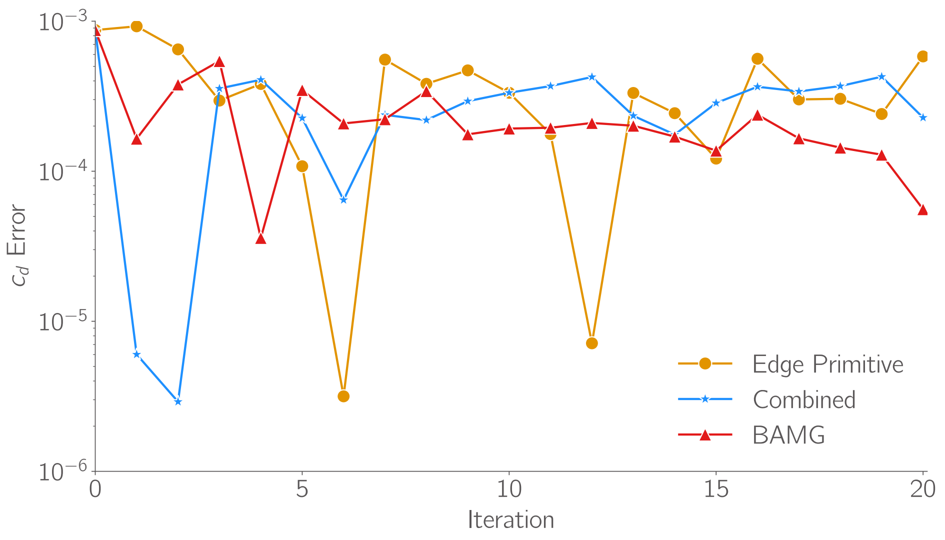


Fig. 8 Output error convergence for the inviscid NACA 0012 case.

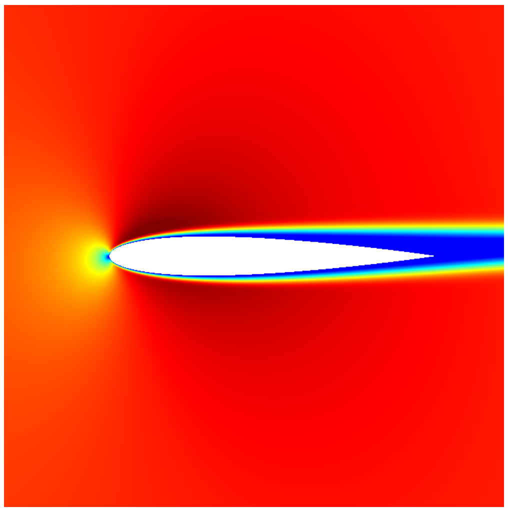


Fig. 9 Mach contours for viscous NACA 0012 case (0-0.6).

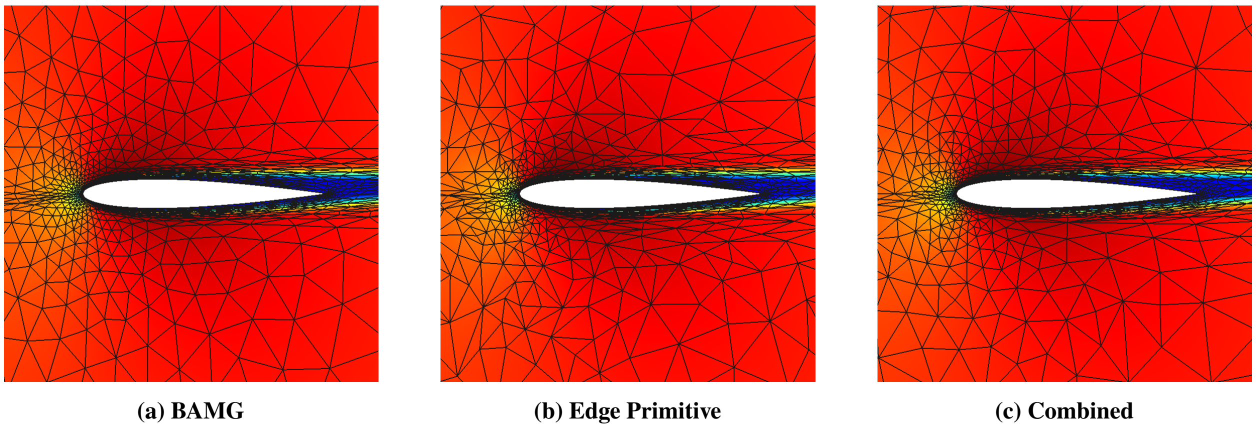

(a) BAMG (b) Edge Primitive (c) Combined

Fig. 10 Adapted meshes for laminar NACA 0012 case with Mach contours (0-0.6).

reduced the number of edges outside the quasi-unit range from 185 to 131. This case again shows that adding node optimization provides a smoothing step that helps metric conformity and mesh quality.

The drag error convergence for the adaptation iterations is shown in Figure 11. Here we see a that BAMG provides the best error convergence but our edge primitive adaptation is not far behind. Adding in node optimization at the end of each adaptation iteration further reduces the drag and the error between the combined approach and BAMG is 0.04 drag counts. This shows promise for extending this method to three dimensions where mesh curving and metric based re-meshing for complex geometries is a challenge.

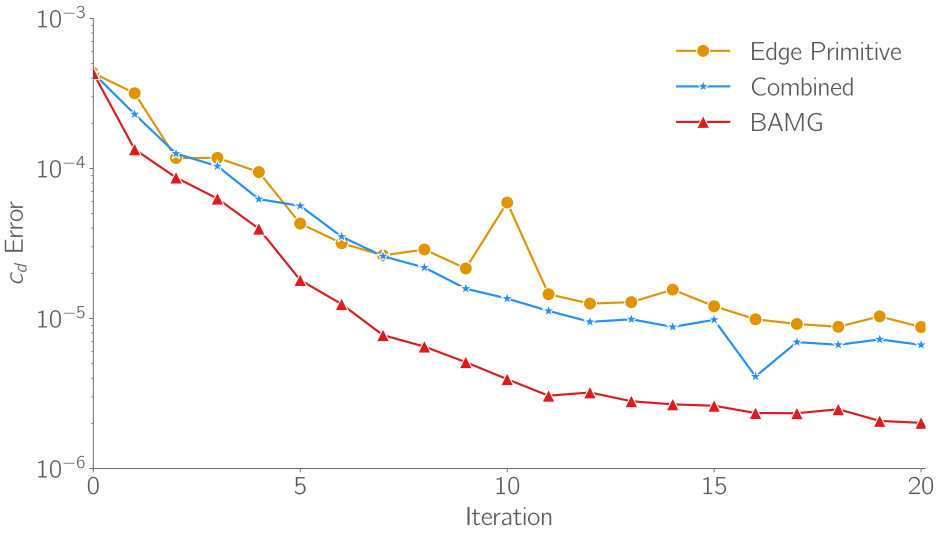
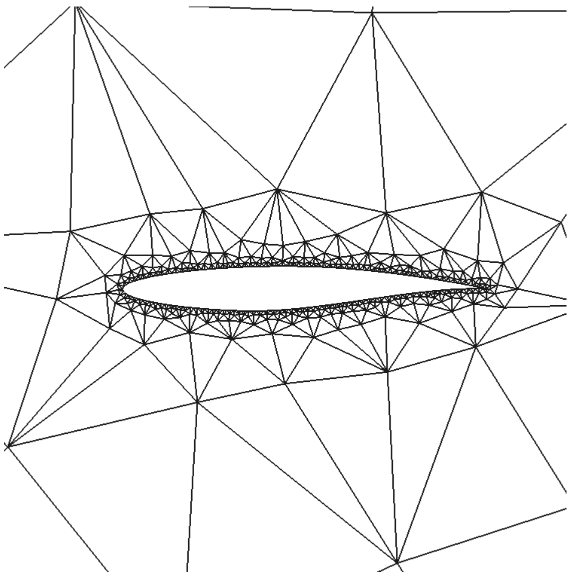


Fig. 11 Output error convergence for the laminar viscous NACA 0012 case.

D. Turbulent Transonic RAE 2822

The final test case is the RAE 2822 airfoil in transonic turbulent flow with a Mach number of $M_\infty = 0.734$, angle of attack of $\alpha = 2.79°$, and a Reynolds number of $Re = 6.5 \times 10^6$. We use the Spalart-Allmaras turbulence model [46] and element-wise constant artificial-viscosity for shock capturing. The thin boundary layer in this test case requires elements with an aspect ratio $\text{Æ} \sim O(10^4)$. We start with a cubic mesh, $q = 3$, with 758 elements, shown in Figure 12, and we use a solution order of $p = 2$.


(a) Initial mesh $q = 3$ (758 elements)

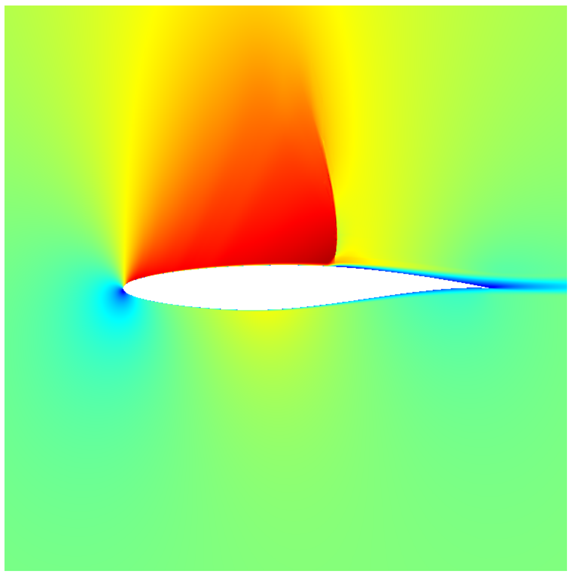

(b) Mach contours (0-1.4)

Fig. 12 Initial mesh and Mach contours for RAE 2822 case.

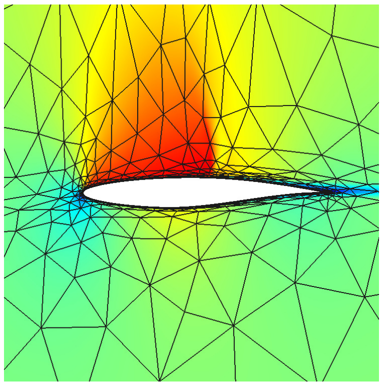
We target the drag on the airfoil as our output that drives the adaptation and perform 15 adaptation iterations at $8,000$, $16,000$, $24,000$, $32,000$, $48,000$, and $64,000$ degrees of freedom. In this case we do not use any node optimization as the high levels of anisotropy proved too difficult for the global optimization problem. We therefore, only compare global re-meshing with our high-order edge primitive adaptive strategy. The final adapted meshes at $8,000$, $32,000$, and $64,000$ degrees of freedom are shown in Figure 13.

Here we see that at $8,000$ degrees of freedom both cases are not able to capture the shock well due to most of the degrees of freedom being put into the boundary layer. Additionally, the edge primitive mesh has many sliver elements away from the airfoil which means more degrees of freedom are being wasted in this case. When increasing to $32,000$ degrees of freedom we see that the meshes begin to capture target and align with the shock. The BAMG mesh does a much better job of this because again the edge primitive mesh has sliver elements that wastes degrees of freedom and cannot put them along the shock. Nevertheless, this case is able to successful generate a valid curved mesh with highly anisotropic elements in the boundary layer. At $64,000$ degrees of freedom we see both cases have highly anisotropic elements in the boundary layer as well as aligned with the shock, and around the adjoint features around the leading edge. We see both meshes have mostly isotropic elements away from the airfoil with the edge primitive mesh having very few sliver elements. This is due to having more vertices in the mesh to specify the metric and therefore changes between the metric across the edges are not as large.

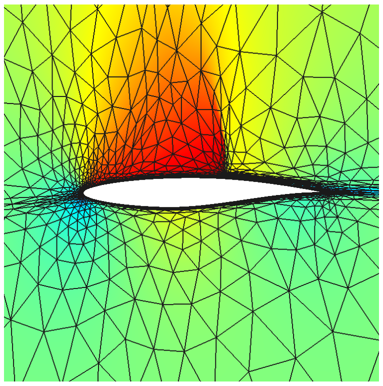
The drag coefficient convergence for these cases are shown in Figure 14. The edge primitive and BAMG mesh at $64,000$ degrees of freedom only differ by 0.3 drag counts. This test case shows that our method is able to generate highly anisotropic curved meshes with no re-curving. While the mesh does not align with all the flow features that the BAMG meshes does drag prediction is quite close and more robust as we always have a valid curved mesh during the adaptation process.

VIII. Conclusion

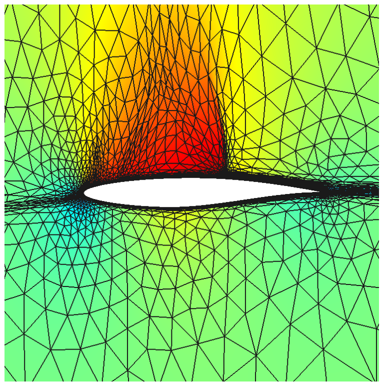
This paper presents a method for adapting high-order unstructured curved meshes using traditional edge splits and collapses. Many previous works have focused on mesh adaptation for linear meshes and for high-order applications re-curved the mesh by solving either solving an analogous physics based problem or an optimization problem. We present a novel method for performing traditional edge primitive operations, in particular the edge collapse, for high-order meshes. This process is able to reject operators that would create an invalid mesh and therefore the mesh stays curved



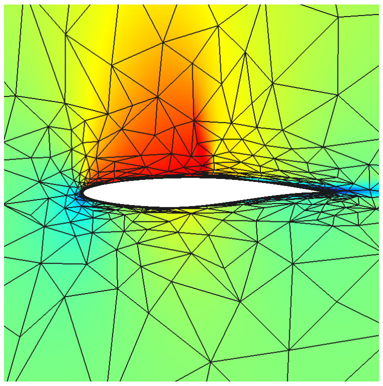
(a) BAMG $C^{\mathrm{tgt}} = 8,000$



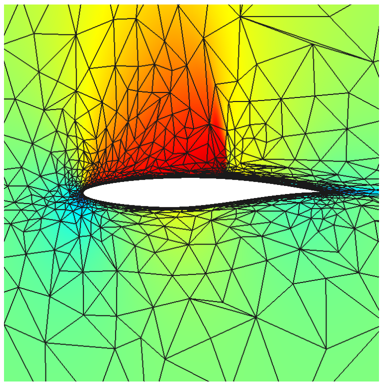
(b) BAMG $C^{\mathrm{tgt}} = 32,000$



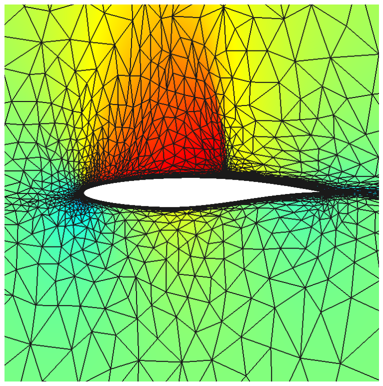
(c) BAMG $C^{\mathrm{tgt}} = 64,000$



(d) Edge Primitive $C^{\mathrm{tgt}} = 8,000$



(e) Edge Primitive $C^{\mathrm{tgt}} = 32,000$



(f) Edge Primitive $C^{\mathrm{tgt}} = 64,000$

Fig. 13 Adapted meshes for laminar RAE 2822 case with Mach contours (0-1.4).

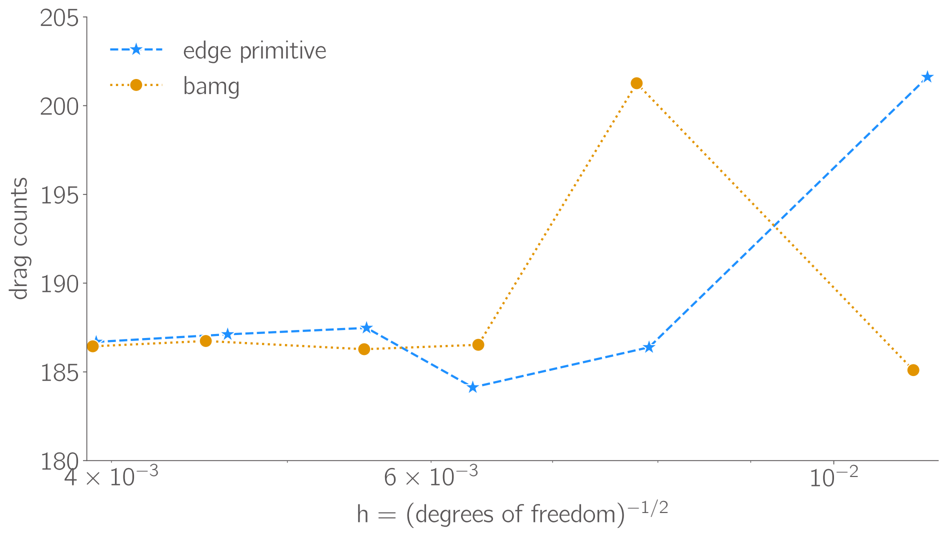


Fig. 14 Initial mesh and Mach contours for RAE 2822 case.

and representative of the true geometry during the entire adaptation process. By not having to re-curve the mesh after each adaptation iteration, the adaptation process becomes more robust because there is always a valid mesh to solve the primal and adjoint on and try the adaptation process again. We presented a variety of test cases and showed that this adaptation process is able to generate valid curved meshes that are coarse or fine. Additionally, we demonstrated the ability to generate highly anisotropic curved meshes, with aspect ratios $\text{Æ} \sim O(10^3)$, which are necessary for simulations at high Reynolds number. These meshes performed very similarly to meshes generated from an existing metric based re-meshing software. We also showed for cases where lower anisotropy is required coupling with node optimization significantly improved mesh quality and metric conformity. Future work will be to extend this method to three dimensions and add swap operators to improve mesh quality. Additionally, this work only focused on linear edge length, in the future we plan to combine this method with adaptation that moves the high-order nodes to control discretization error.

References

[1] 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.

[2] Yano, M., Modisette, J., and Darmofal, D., "The importance of mesh adaptation for higher-order discretizations of aerodynamics flows," AIAA Paper 2011-3852, 2011.

[3] Park, M. A., Loseille, A., Krakos, J. A., and Michal, T. R., "Comparing Anisotropic Output-Based Grid Adaptation Methods by Decomposition," AIAA Paper 2015-2292, 2015. doi:10.2514/6.2015-2292.

[4] 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.

[5] Fidkowski, K. J., "A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization," *AIAA SciTech Forum*, 2016. doi:10.2514/6.2016-0835.

[6] Yano, M., and Darmofal, D., "An Optimization Framework for Anisotropic Simplex Mesh Adaptation: Application to Aerodynamic Flows," AIAA Paper 2012-0079, 2012.

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

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

[9] Li, X., Shephard, M. S., and Beall, M. W., "3D anisotropic mesh adaptation by mesh modification," *Computer methods in applied mechanics and engineering*, Vol. 194, No. 48-49, 2005, pp. 4915–4950.

[10] 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.

[11] Loseille, A., Alauzet, F., and Menier, V., "Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes," *Computer-Aided Design*, Vol. 85, 2017, pp. 53–67.

[12] Coupez, T., "Génération de maillage et adaptation de maillage par optimisation locale," *Revue européenne des éléments finis*, Vol. 9, No. 4, 2000, pp. 403–423.

[13] Feuillet, R., Loseille, A., Marcum, D., and Alauzet, F., "Connectivity-change moving mesh methods for high-order meshes: Toward closed advancing-layer high-order boundary layer mesh generation," *2018 Fluid Dynamics Conference*, 2018, p. 4167.

[14] Feuillet, R., Loseille, A., and Alauzet, F., "Optimization of P2 meshes and applications," *Computer-Aided Design*, Vol. 124, 2020, p. 102846.

[15] Luo, X.-J., Shephard, M. S., O'bara, R. M., Nastasia, R., and Beall, M. W., "Automatic p-version mesh generation for curved domains," *Engineering with Computers*, Vol. 20, 2004, pp. 273–285.

[16] Capon, P. J., and Jimack, P. K., “On the Adaptive Finite Element Solution of Partial Differential Equations Using h-r Refinement,” Tech. Rep. 96.03, University of Leeds, School of Computing, 1996.

[17] McRae, D. S., “r-Refinement Grid Adaptation Algorithms and Issues,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 2000, No. 4, 189, pp. 1161–1182.

[18] Ding, K., Fidkowski, K. J., and Roe, P. L., “Continuous adjoint based error estimation and r-refinement for the active-flux method,” AIAA Paper 2016–0832, 2016. doi:10.2514/6.2016-0832.

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

[20] 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.

[21] Persson, P.-O., and Peraire, J., “Curved mesh generation and mesh refinement using Lagrangian solid mechanics,” *47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, 2009. doi:10.2514/6.2009-949.

[22] Abgrall, R., Dobrzynski, C., and Froehly, A., “A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems,” *International Journal for Numerical Methods in Fluids*, Vol. 76, No. 4, 2014, pp. 246–266.

[23] 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.

[24] 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.

[25] 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.

[26] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., “Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model,” Big Island, Hawaii, 2012. URL `http://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf`.

[27] Ceze, M. A., and Fidkowski, K. J., “High-order output-based adaptive simulations of turbulent flow in two dimensions,” *AIAA Journal*, Vol. 54, No. 9, 2016. doi:10.2514/1.J054517.

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

[29] Cockburn, B., and Shu, C.-W., “Runge-Kutta discontinuous Galerkin methods for convection-dominated problems,” *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261.

[30] 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, 2005, pp. 92–113. doi:10.1016/j.jcp.2005.01.005.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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`.

[36] 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`.

[37] 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.

[38] 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.

[39] Pennec, X., Fillard, P., and Ayache, N., "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, Vol. 66, No. 1, 2006, pp. 41–66.

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

[41] 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.

[42] 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.

[43] Liu, D. C., and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, Vol. 45, No. 1–3, 1989, pp. 503–528. doi:10.1007/bf01589116.

[44] Caplan, P. C., "Four-Dimensional Anisotropic Mesh Adaptation for Spacetime Numerical Simulations," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2019.

[45] Hecht, F., "BAMG: Bidimensional Anisotropic Mesh Generator," INRIA–Rocquencourt, France, 1998. `www.freefem.org`.

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