Journal of Computational Physics 545 (2026) 114506

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Anisotropic mesh adaptation for high-Order meshes in two dimensions

Alexander W.C. Coppeans *, Krzysztof J. Fidkowski, Joaquim R.R.A. Martins

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109, USA

ARTICLE INFO

Keywords:
Anisotropic mesh adaptation
High-order meshes
Finite element method
Riemannian metric field
Curved meshes

ABSTRACT

Error estimation and mesh adaptation are essential for accurate and efficient computational fluid dynamics simulations, but traditional approaches for high-order curved meshes face significant robustness issues. Traditionally, curved meshes are first made linear and then adapted and re-curved at the end of the adaptation process. Making the mesh linear and re-curving the mesh risks creating negative Jacobians for anisotropic elements, which adversely affects the robustness of the mesh adaptation process. We developed a novel method for performing anisotropic mesh adaptation on high-order curved meshes. This process extends existing edge primitive operations for mesh adaptation to high-order meshes. In contrast to the traditional procedure, our method keeps the mesh valid and curved during the entire adaptation process, leading to more robust mesh adaptation. The method extends traditional linear mesh modification operators to arbitrary-order curved meshes. Results for four test cases demonstrate that this method produces valid curved meshes with various aspect ratios. The test cases show that our method recovers optimal high-order error convergence rates and performs similarly to existing global remeshing strategies. This appraoch enables reliable automated of mesh adaptation which is necessary to realize the benefits of high-order computational fluid dynamics methods.

1. Introduction

Error estimation and mesh adaptation are essential tools for fully automated, robust, and accurate computational fluid dynamics (CFD) simulations [1]. Unstructured meshes, in particular, are well-suited for mesh adaptation because the initial mesh is easily generated and adapted. For unstructured meshes to take advantage of mesh adaptation, it is crucial 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.

Many methods exist for generating a metric-conforming mesh, such as global remeshing, global node movement, and local mesh operations. Metric-conforming remeshing has been shown to be highly effective in two dimensions [5,6] but is not always tractable or robust, especially for three-dimensional meshes with curved, anisotropic elements. Node movement, also known as *r*-refinement [7–9], moves the nodes to adapt the mesh. Recently, node movement has been used to move mesh nodes to conform to a target metric field [10–13]. Local mesh modification operators have been shown to work in two [3], three [14–18], and even four dimensions [19]. These local mesh modifications are typically done either explicitly through edge primitive operations or general cavity operators. Edge primitive operators [14–16] explicitly handle each operation, such as edge splits, edge collapses, edge swaps, and face swaps.

* Corresponding author.
E-mail address: awccopp@umich.edu (A.W.C. Coppeans).

https://doi.org/10.1016/j.jcp.2025.114506
Received 7 November 2024; Received in revised form 16 October 2025; Accepted 1 November 2025
Available online 4 November 2025

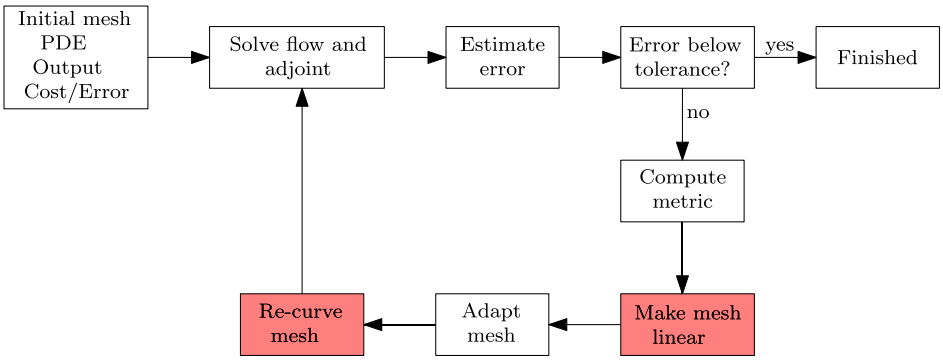
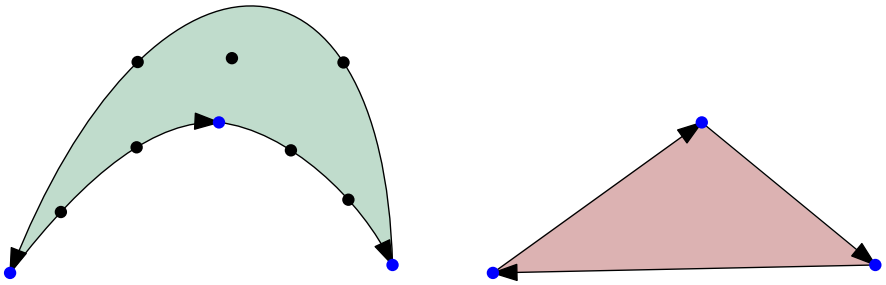


Fig. 1. Typical procedure for iterative metric-based mesh adaptation with curved meshes. Making a curved mesh linear and re-curving the mesh are potential points of failure in the automated adaptation process and are eliminated with HOEP.



(a) Valid $q = 3$ element (b) Invalid $q = 1$ element

Fig. 2. A valid $q > 1$ element with an invalid $q = 1$ representation. The ordering of $q = 1$ vertices switches from counter-clockwise to clockwise when the element is made $q = 1$.

The general cavity operator is a local mesh modification operation that combines traditional edge primitive operations into one general operator [17,19]. Mesh optimization operators such as edge swaps and node smoothing have been implemented on high-order meshes [20,21], and previous work has implemented curved edge split and collapse operators [22]. Recently, Shi and Persson [23] developed a local element patch based optimization approach used to extend the traditional linear edge operators to high-order meshes. However, anisotropic mesh adaptation for curved meshes remains a challenge.

High-order simulations require not only an anisotropic mesh to use degrees of freedom efficiently, but also curved faces on boundaries [24]. Currently, the state of the art is to generate a linear mesh and curve it to represent the geometry. Two approaches are commonly used to curve a mesh. The first involves solving an analogous physics-based problem to transfer boundary displacements to the interior [25,26]. The second involves solving an optimization problem [27–30] that simultaneously moves both boundary and interior vertices. Existing mesh curving methods face a fundamental trade-off between robustness and computational efficiency. Most approaches also ignore the target metric field when repositioning nodes after adaptation, which reduces metric conformity. Though recent work has begun incorporating metric information during curving [13,31,32] developing metric fields for curved elements remains challenging.

In an adaptive setting, the starting curved mesh is made linear, then the mesh is adapted, and finally the mesh is curved. Fig. 1 shows the stages of metric-based mesh adaptation when using curved meshes. Before the adaptation occurs, the curved mesh is made linear. While it might be expected that a valid curved mesh corresponds to a valid linear mesh, this is not always true. Fig. 2 shows an example of how reducing the order of an individual element can invert it and create an invalid linear element. If the linear mesh adaptation tool requires a valid input mesh, this order reduction would then cause failures. After the adaptation occurs, the mesh is re-curved to be used in the next simulation. This curving step, here called re-curving, is prone to failures that require human intervention in the adaptation loop. Neither of these steps that are capable of causing failures are required if the adaptation is performed on the curved mesh.

We introduce high-order edge primitive, HOEP, which is a novel approach that maintains a valid curved mesh throughout the entire adaptation process. Unlike existing methods that require linearization and re-curving steps-both potential points of failure, HOEP directly modifies the curved elements, eliminating these failure-prone transitions entirely. This improves on conventional practice and addresses a critical robustness bottleneck in high-order CFD simulations.

We demonstrate our approach through test cases solved with a high-order discontinuous Galerkin (DG) finite-element method, described in Section 2. Section 3 presents the output error estimation procedure, and Section 4 presents the metric optimization algorithm, MOESS: Mesh Optimization via Error Sampling and Synthesis. Section 5 presents background on metric-based mesh adap-

tation, and Section 6 presents our approach for high-order mesh adaptation. Results for four test cases are presented and discussed in Section 7, where we demonstrate our method's ability to generate highly anisotropic curved meshes and obtain comparable output error convergence to global remeshing. Section 8 concludes with a summary of the work and future work directions.

2. Discretization

2.1. Governing equations

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 [33,34].

2.2. The discontinuous Galerkin method

The DG method is a finite-element method in which the approximation space is discontinuous across element boundaries [35–37]. The state is approximated for each element, Ω_e, with polynomials of order p on a mesh of non-overlapping elements. Multiplying Eq. 1 by test functions in the same space as the solution and integrating by parts to couple elements via fluxes yields the weak form. We use the Roe [38] convective flux and the second form of Bassi and Rebay (BR2) [39] for viscous treatment. After choosing a basis, the weak form becomes 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 consisting of coefficients on the basis functions, 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, $\mathbf{R}(\mathbf{U}) = \mathbf{0}$. To solve this system of nonlinear equations, we use the Newton–Raphson method with pseudo-time continuation [40] and the generalized minimum residual (GMRES) linear solver [41], preconditioned by an element-line Jacobi smoother with coarse-level ($p = 1$) correction [42,43].

2.3. The output adjoint

The output adjoint for a scalar output is the sensitivity of the output to residual source perturbations [1,44]. In this work, we use the steady adjoint, where we start by linearizing the residual and scalar output, $J(\mathbf{U})$, and solve the linear system,

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

where $\mathbf{\Psi} \in \mathbb{R}^N$, is the discrete adjoint vector. This equation is solved using a transposed version of the preconditioned-GMRES method 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.

3. 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 [1,44]. 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, $p \to p+1$. The state vector is first prolongated from the coarse space to the fine space to obtain a new state $\mathbf{U}_h^H$. Next, the prolongated 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,

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

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

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

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

4. Metric optimization

The elemental error indicator combined with additional error indicators evaluated on sub-elements, via Eq. 5 with projected adjoints, drive a metric optimization calculation based on MOESS [5,45]. The optimized metric is used in mesh adaptation to generate a new metric conforming mesh. Metric optimization uses a Riemannian metric field, $\mathcal{M}(\vec{x})$, to encode information about the 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 measure of a non-dimensional distance, δl, 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 that are a unit measure from $\vec{x}$ is an ellipse whose principal axes orientations and lengths are determined by the eigenvectors and eigenvalues of $\mathcal{M}$.

Affine-invariant changes to the metric field [46] are made by a symmetric *step matrix*, $\mathcal{S} \in \mathbb{R}^{d\times d}$, defined as

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

where $\mathcal{M}_0$ is the current mesh-implied metric field. When $\mathcal{S}=0$ the metric remains the same, 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 changes. The elemental error model represents how an element's error changes if we apply a step matrix change to it. The error model we use is

$$\varepsilon_k = \varepsilon_{k0} \exp\left[\mathrm{tr}\left(\mathcal{R}_k \mathcal{S}_k\right)\right], \tag{8}$$

where ε_{k0} is the initial error on element k, and $\mathcal{R}_k$ 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, changes as we apply a step matrix to an element. The cost model is

$$C_k = C_{k0} \exp\left[\frac{1}{2}\mathrm{tr}\left(\mathcal{S}_k\right)\right], \tag{9}$$

where C_{k0} 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. Yano [45] and Fidkowski [5] provide more details on these models.

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 the marginal cost ratio over elements in the mesh. MOESS takes in a current mesh with its mesh-implied metric, $\mathcal{M}_0(\vec{x})$, elemental error indicators, ε_{e0}, and elemental rate tensor estimate, $\mathcal{R}_e$, and outputs a step matrix field $\mathcal{S}(\vec{x})$. In practice, mesh optimization and the flow and 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 the error further if it is not at an acceptable level.

5. Metric based mesh adaptation

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

Another 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 lengths are 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 according to

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

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

$$L_a = \left(\vec{v}_e^T \mathcal{M}_a \vec{v}_e\right)^{\frac{1}{2}} \quad \text{and} \quad L_b = \left(\vec{v}_e^T \mathcal{M}_b \vec{v}_e\right)^{\frac{1}{2}}, \tag{11}$$

where $\vec{v}_e$ is the vector along edge e, $\vec{v}_e = \vec{x}_b - \vec{x}_a$. The ideal length for a metric-conforming mesh is one. However, this constraint is relaxed, and we target a quasi-unit length range $L_{\text{low}} \le L_m \le L_{\text{up}}$. Typically, $L_{\text{low}} = \sqrt{2}/2$ 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 metric volume of an element k is

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

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 the vertex of 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 dimensions as

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

where, $\hat{V}_k$, is the area of a unit-equilateral triangle, $\hat{V}_k = \sqrt{3}/4$. The element quality Q_k measures how closely equilateral the triangle is in the transformed metric space. To improve element quality, we use local swap and node smoothing operators.

6. 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 performs a linear combination of Lagrange interpolating polynomials given as

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

where $\vec{\xi}$ is the reference space coordinates, N_Q is the number of interpolating polynomials, and $\vec{x}_i$ is the node coordinate in physical space associated with the i^{th} interpolating polynomial ϕ_i. The mapping Jacobian, $\underline{\mathrm{J}}$, is defined as,

$$\underline{\mathrm{J}} = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \sum_i^{N_Q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{\xi}}\left(\vec{\xi}\right). \tag{16}$$

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

While we perform 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. When we compute a metric edge length, we use Eq. 10 for the linear representation and do not consider the shape of the curved edge. Similarly, with element quality, we use Eq. 13 and only consider the three nodes that make up the linear element in two dimensions. This choice is similar to the state-of-the-art linear adaptation with re-curving because the re-curving process is used only to create a valid mesh without considering a high-order metric. Additionally, the output target metric field generated from MOESS describes only the shape of linear elements and does not provide information about how the elements should be curved. Incorporating a curved metric field has the potential to improve accuracy but generating the high-order metric field remains a challenge [32].

6.1. Validity checks

Our adaptation method is robust because it can reject any operation that would create an invalid mesh or cause the solver to fail. We use a sequence of element validity checks that increase in computational expense, resulting in only a few modifications requiring the most expensive checks. The first check is a $q = 1$ validity check, which verifies that all elements have valid linear representations, using the adaptive precision floating point library developed by Shewchuk [47]. While it is possible for a curved element to have a positive Jacobian determinant everywhere and a negative $q = 1$ Jacobian determinant, we found this constraint improves mesh quality and solver robustness. We then check the element quality using Eq. 14 and all modified edge lengths using Eq. 10 to ensure we do not reduce minimum element quality and edge length or increase the maximum edge length in the mesh. These first checks are all performed on the linear representations of the elements and are computationally inexpensive.

If all modified elements pass these linear checks, we then perform nonlinear checks to ensure that the curved elements are valid. We ensure that the determinant of the Jacobian mapping matrix is positive everywhere in each element and additionally check the maximum and minimum element distortion, defined as

$$\sigma\left(\vec{\xi}\right) \equiv \frac{\det \underline{\mathrm{J}}^{q=1}}{\det \underline{\mathrm{J}}\left(\vec{\xi}\right)}. \tag{17}$$

The determinant in the numerator corresponds to 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. In this work, the distortion is limited to $0.2 \leq \sigma \leq 5$, a range determined through numerical experiments to mitigate nonlinear primal solver convergence issues. By checking $\det \underline{\mathrm{J}}^{q=1} > 0$ in the first stage of linear element validity checks, the limits on distortion ensure $\det \underline{\mathrm{J}}(\vec{\xi}) > 0$.

To accurately check the bounds of σ and ensure that the Jacobian determinant polynomial is positive everywhere within the element, we use the adaptive method of Johnen et al. [48]. This method relies on using the fact that the Jacobian determinant is a high-order polynomial. We obtain a Bézier representation of that polynomial and use the convex hull property to check the bounds using the minimum and maximum of the Bézier coefficients. If the coefficients are all positive, the element is valid. However, if any coefficients are negative and not on element corners, the element may still be valid. Therefore, the polynomial's domain is then

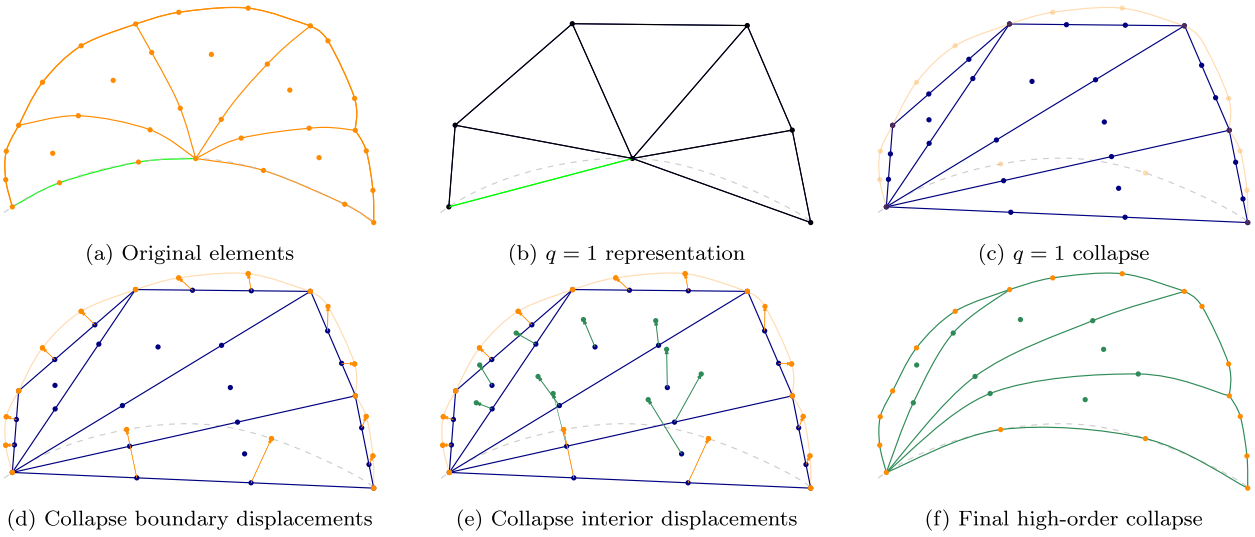


Fig. 3. Collapse operator for high-order elements. The dotted gray line is the geometry boundary. First the collapse is performed on a linear representation of the elements. Then high-order nodes on the boundary of the affected elements are moved to their original locations or projected to the geometric boundary. Finally those displacements are transferred to interior nodes through inverse-distance or RBF interpolation.

subdivided to achieve Bézier representation on sub-elements, resulting in more accurate bounds. This process is carried out recursively until all coefficients on the subdomains are positive, a corner coefficient is negative, or a maximum number of subdivisions is reached, at which point the element is assumed invalid.

6.2. Edge collapse

Mesh coarsening is performed by collapsing edges that have metric edge lengths flagged as too short. The first step of the collapse is to choose which endpoint of the targeted edge is removed, denoted as a, and which node is kept, denoted as b. If the edge contains both a boundary and an interior node as its endpoints, then only the interior node can be removed. Otherwise, to choose this node, we perform the $q = 1$ validity checks described in Section 6.1, and choose the node that leads to the highest minimum quality in the elements affected by the collapse. However, if this node fails the nonlinear Jacobian determinant checks, the other node is then tried if it passes the $q = 1$ validity checks. If both nodes fail, the mesh remains, and the collapse iteration tries a different edge.

Only the elements that contain the removed vertex are affected by the collapse; we refer to this set of elements that contain node a as the *ball* of affected elements. The elements on the collapsed edge are removed, and the other elements change their $q = 1$ connectivity to replace node a with node b. If the mesh is linear, this is all that needs to be done. However, for high-order elements, the additional Lagrange nodes need to be placed so that elements that do not contain node a are not affected, and all elements are valid. For high-order elements this process is shown in Fig. 3.

First, high-order nodes are placed on the linear representation of the elements after the $q = 1$ collapse. Then the high-order nodes on the boundary of the ball (nodes on the faces opposite to node a) are placed back to their original positions so as not to affect elements that did not contain node a, or if the face is a geometric boundary, the nodes are projected to the boundary. This is like displacing those boundary nodes from their position on the boundary linear element, $\vec{x}_b$ by some distance $\Delta\vec{x}_b$. Since only one face is curved, affected elements may remain valid with moderate curvature. We perform validity checks on these elements—if all pass, the collapse is complete. If any fail, we then reposition interior nodes within the ball. The interior nodes are repositioned through an inverse-distance based interpolation of the boundary displacements to the interior nodes, given by

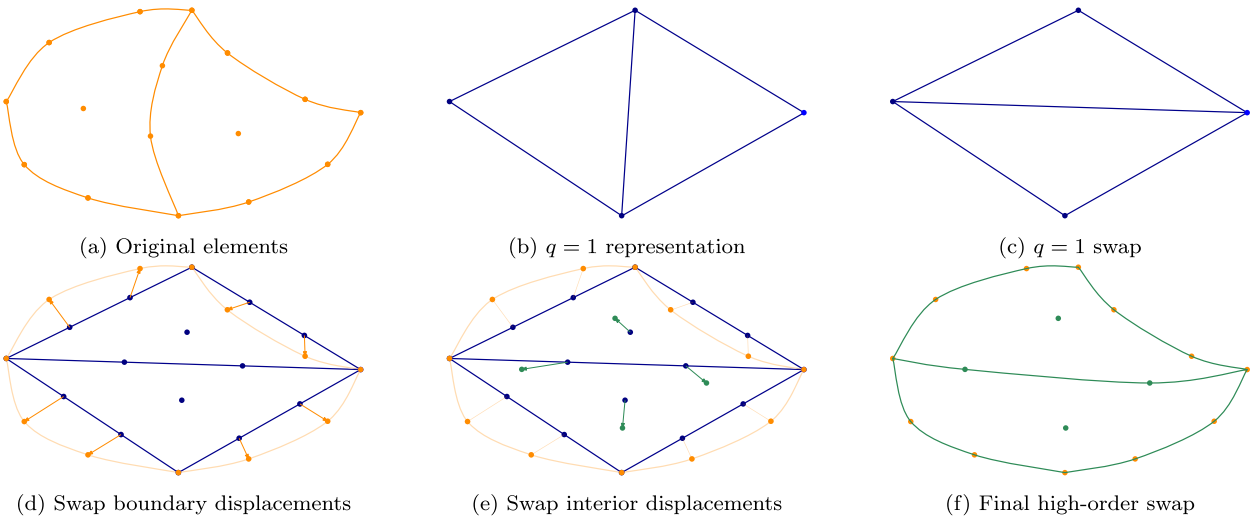
$$\Delta\vec{x} = \vec{x} - \vec{x}_0 = \frac{\sum_i^{N_b} w_i(\vec{x}_0)\Delta\vec{x}_{b,i}}{\sum_i^{N_b} w_i(\vec{x}_0)}, \tag{18}$$

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 boundary node,

$$w_i(\vec{x}_0) = \left(\frac{1}{|\vec{x}_0 - \vec{x}_{b,i}|}\right)^{\gamma} + \left(\frac{\alpha}{|\vec{x}_0 - \vec{x}_{b,i}|}\right)^{\theta}, \tag{19}$$

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. [49].

The inverse-distance method provides an explicit calculation of the interior node displacements and is computationally inexpensive. For most operations, it yields valid elements, but for operations where the curving step produces invalid elements, we move to



(a) Original elements (b) $q = 1$ representation (c) $q = 1$ swap

(d) Swap boundary displacements (e) Swap interior displacements (f) Final high-order swap

Fig. 4. Swap operator for high-order elements. First the swap is performed on the linear representation elements. Then high-order boundary nodes are displaced to their original positions. Finally the displacements are transferred to interior nodes.

another mesh curving method using radial basis functions (RBF) [50,51]. A radial basis function is a function that only depends on the distance from a specified center $\vec{c}$ such that $\phi(\vec{x}) = \phi(||\vec{x} - \vec{c}||)$. The interpolation can be written as

$$\Delta\vec{x} = \sum_{i=1}^{N_b} \vec{r}_i \phi_i(||\vec{x} - \vec{x}_{b,i}||) + \vec{p}(\vec{x}), \tag{20}$$

where $\vec{r}_i$ is the coefficient for the i^{th} RBF, and $\vec{p}(\vec{x})$ is an order p polynomial. The coefficients r_i and the polynomial $\vec{p}$ are determined by enforcing two interpolation conditions. The first enforces a specified displacement at the RBF centers,

$$\Delta\vec{x}\Big|_{\vec{x}_{b,i}} = \Delta\vec{x}_{b,i}, \tag{21}$$

and the second enforces that for any polynomial $q(\vec{x})$ with an order less than or equal to p,

$$\sum_{j=1}^{N_b} \vec{r}_i q(\vec{x}_{b,j}) = \mathbf{0}. \tag{22}$$

These conditions result in a linear system to solve for $\vec{r}_i$ and the coefficients of the polynomial $\vec{p}$. The linear system has $N_b + d + 1$ unknowns, which is small considering N_b is the number of nodes on the boundary of the elements affected by each local operation, and d is the spatial dimension.

Once we have computed the coefficients, we use Eq. 20 to compute the displacements of interior nodes. We use Wendland's C^2 function with local support [52], which is written as

$$\phi(R_i) = \begin{cases} (1 - R_i)^4(4R_i + 1) & R_i \leq 1, \\ 0, & R_i > 1. \end{cases} \tag{23}$$

where R_i is a normalized distance to the i^{th} boundary node, given as

$$R_i = \frac{||\vec{x} - \vec{x}_{b,i}||}{\rho_i}, \tag{24}$$

where ρ_i is the support radius of the i^{th} boundary node, set to be ten times the magnitude of the boundary node's displacement [53].

Once the displacements are transferred to the interior nodes, the elements are checked to ensure positive Jacobians. If all elements pass, then the mesh is updated.

6.3. Edge swap

The edge swap operator improves mesh quality. The high-order edge swap is shown in Fig. 4.

To perform the edge swap, we follow a process similar to the edge collapse. The elements surrounding the affected edge are made linear, and the edge swap is executed. Then, the $q = 1$ validity checks are performed. If the minimum quality of the elements does not increase, the swap is rejected. Otherwise, the edge swap is accepted, and the high-order nodes are placed on the linear swapped elements. Similarly to the collapse, the boundary nodes are displaced back to their original locations, and then the inverse-distance and RBF interpolation is used to transfer those displacements to interior nodes. We then verify the nonlinear Jacobian determinants of the proposed elements to confirm validity.

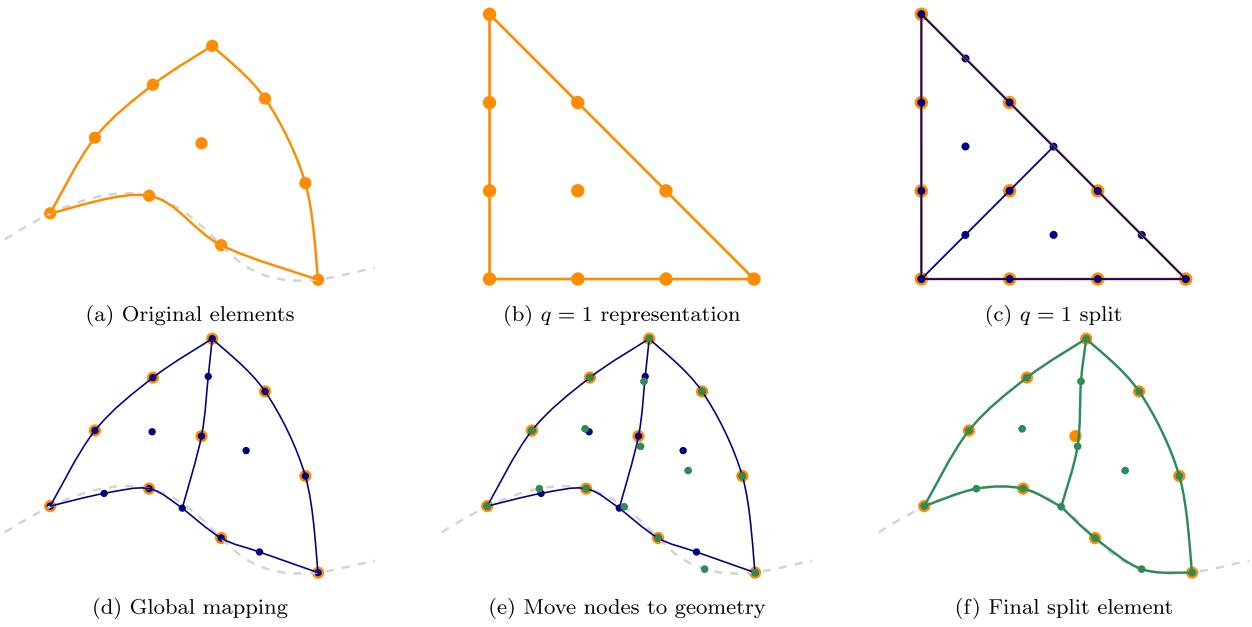


Fig. 5. Split operator for high-order elements.

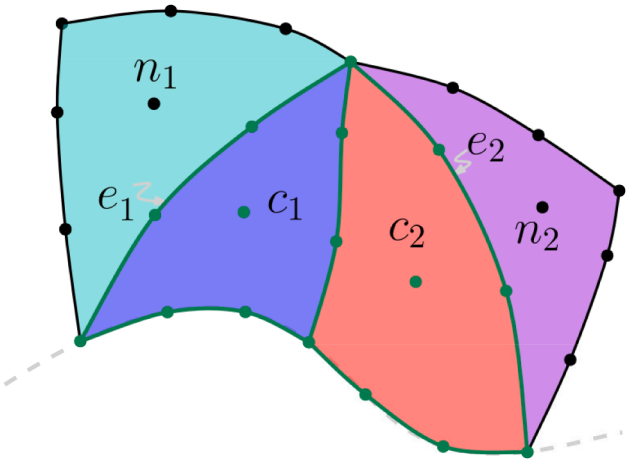


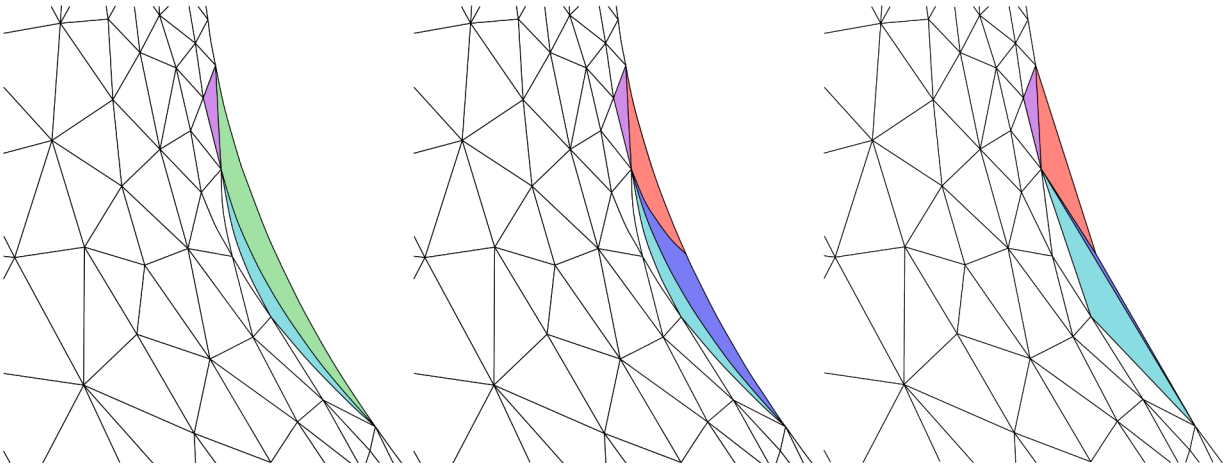
Fig. 6. Diagram labeling child elements and neighbors after an edge split.

6.4. Edge split

The edge split is used to refine regions of the mesh where edges have metric lengths flagged as too long. For high-order elements, the split is performed in each element's reference space, shown in Fig. 5.

First, the flagged edge is split in reference space, and the high-order nodes are placed on the child elements in the parent element's reference space. The parent element's reference to global space mapping is then used in Eq. 15 to map the child element's high-order nodes into global space. Because the element is split in reference space, the resulting element has a positive Jacobian determinant everywhere. However, it is possible to create highly distorted elements after the split, so the Jacobian determinant is still checked to ensure that no such elements are created. If a boundary edge is split, the resulting nodes on the split edge are not guaranteed to be on the geometry. Therefore, we project those nodes to the geometry and use the same inverse-distance and RBF interpolation to displace interior nodes.

While the boundary node displacements are small, the splits can still be rejected. Ensuring that the geometry is well resolved is important, so if the split is rejected after the curving step, additional elements and nodes are added to the curving step. Initially, the nodes on the edges not involved in the split have their displacements set to zero, so only the child elements connected to the split edge are affected. However, we now allow for the neighbors shown in Fig. 6 to be included.



(a) Original Mesh (b) Curved mesh after split. (c) Linear mesh after split has sliver element with low quality.

Fig. 7. Boundary edge split rejected due to the creation of a low quality element (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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. If the split is rejected because child element c_i has high distortion or a negative Jacobian determinant, then the neighbor element n_i is used in the local curving. The displacements of nodes on edges other than edge e_i are set to zero, and both interior nodes and nodes on edge e_i are moved in the local curving step. The inclusion of neighboring elements in curving transfers boundary curvature information to interior mesh elements, enhancing the robustness of splitting and subsequent operations.

6.5. Combined boundary split and swap

As mentioned previously, splitting boundary edges is important to ensure geometric and solution accuracy. During numerical experimentation, many boundary edge splits were rejected due to the creation of elements with low metric quality. We use the linear representation of elements for quality and edge length calculations. This is a limitation because a high-order element may be different from its linear representation. This is shown in Fig. 7, where the split creates an element with low metric quality. This newly created element's 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.

We draw ideas from cavity operators [17], where elements are formed from the proposed split and their neighbors but are not added to the mesh. Using the notation described in Fig. 6, if the split is rejected due to child element c_i failing any validity checks, we try to swap edge e_i. Two new elements are created for c_i and n_i. The swap is then attempted without updating the mesh. If the swap is successful and creates valid elements above the minimum quality threshold, 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_i is performed, and the mesh is updated. This method allows for c_1 or c_2 to be temporarily invalid, as long as the swap creates valid elements, which is tested before updating the mesh and adding c_1 and c_2.

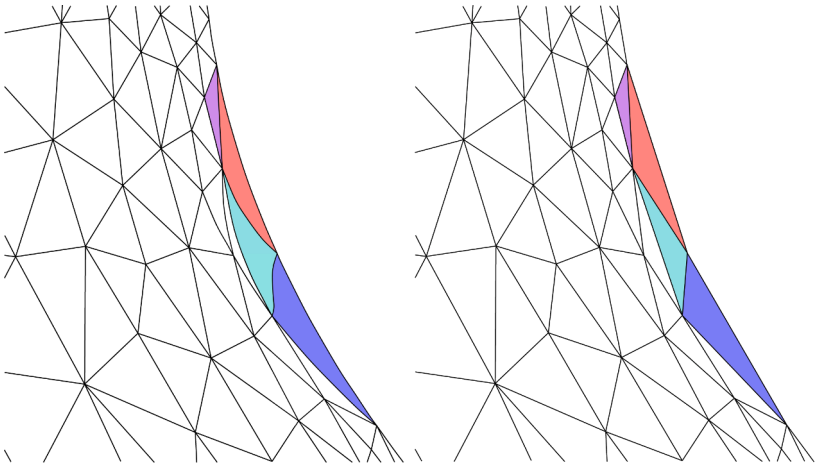
The resulting mesh is shown in Fig. 8 where the edge split rejected in Fig. 7 is performed, and an edge swap is immediately performed. The resulting linear representation of the mesh only has high quality elements with no sliver elements.

6.6. Local node smoothing

Node smoothing is performed on nodes to increase metric conformity and mesh quality. We use the method proposed by Caplan et al. [19], which is a modification of work inspired by Bossen and Heckbert [54], where the $q = 1$ vertices are moved one at a time based on the metric edge lengths of their surrounding edges. The node coordinates are updated as,

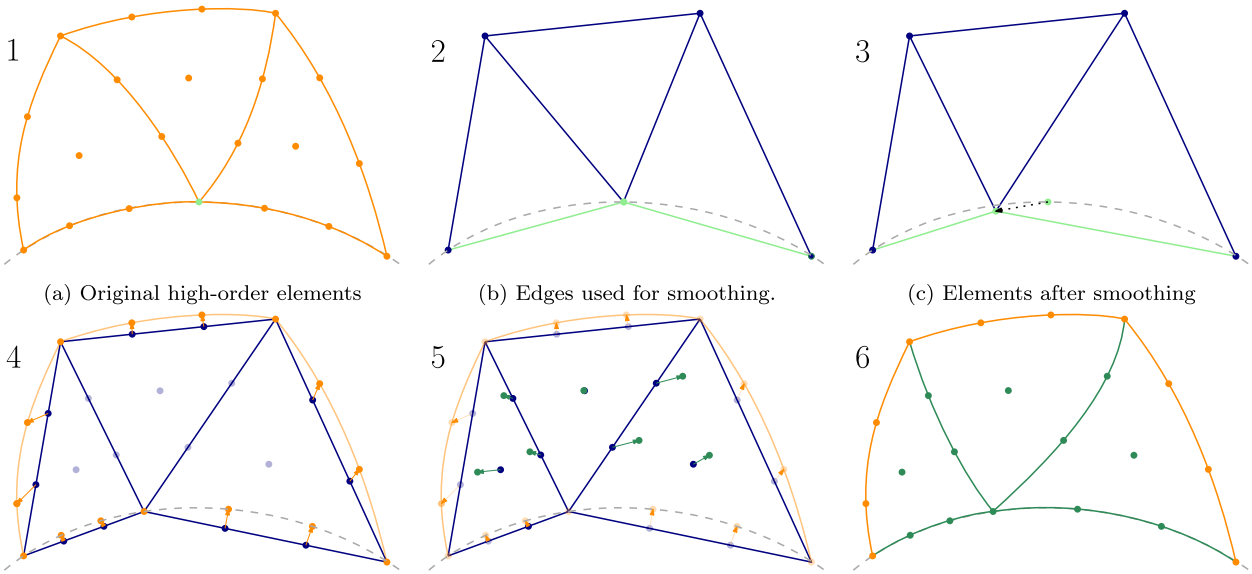
$$\vec{x}_i' = \vec{x}_i + \alpha \sum_{e \in N_i} \left(1 - L_m\right) \exp\left(-L_m\right) \hat{v}_e, \tag{25}$$

where $\vec{x}_i'$ and $\vec{x}_i$ are the updated position and original position of the i^{th} node respectively, α is a relaxation factor set to $\alpha = 0.2$, N_i is a set of edges surrounding node i, and $\hat{v}_e$ is the unit tangent for edge e around node i. For smoothing interior nodes, N_i consists of all edges around the node. However, only edges on the same boundary are used in the smoothing process for nodes on domain boundaries. Nodes that are on two different boundaries or on geometric corners are not smoothed.



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

Fig. 8. Mesh after boundary forced split followed by edge swap operation.



(a) Original high-order elements (b) Edges used for smoothing. (c) Elements after smoothing

(d) Boundary node displacements (e) Interior node displacements (f) Final high-order element

Fig. 9. Boundary node smoothing process for nodes on the geometry. The node is first smoothed using edges on the boundary, and then the node is projected to the boundary. Linear elements are created and locally curved.

Once the $q = 1$ node is moved, the high-order nodes on neighboring elements need to be updated to ensure that each element is valid. For interior nodes, this is done by using a $p = 1$ interpolation of the displacement from the $q = 1$ vertices to the interior nodes where the other $q = 1$ nodes have a displacement of $\bar{0}$. For boundary nodes, we must ensure that all high-order nodes remain on the boundary. The boundary smoothing process is illustrated in Fig. 9. The boundary node is smoothed using Eq. 25 but only considering edges that are on the same boundary as the smoothed node. This does not guarantee that the new node remains on the true geometric boundary, so the node is subsequently projected back to geometry. From there, the high-order elements are redrawn based on the linear elements. The nodes on the boundary of the ball of affected elements are then either moved back to their original positions or projected onto the geometry. Finally, these displacements are transferred to the interior nodes using either the inverse-distance method or the radial basis function method. A node smoothing operation is rejected if the minimum quality of the affected elements is reduced or if any of the validity checks outlined in Section 6.1 fail.

6.7. Adaptation algorithm

Our adaptation algorithm is called HOEP (high-order edge primitive) and is detailed in Algorithm 1. The algorithm iteratively collapses edges shorter than L_{low}, splits edges longer than L_{up} throughout the mesh, and swaps edges and smooths nodes to improve mesh quality.

Algorithm 1: HOEP metric-based mesh adaptation algorithm.

input : Initial mesh $\mathcal{T}_0$, Target metric field $\mathcal{M}(\vec{x})$
output: Adapted mesh $\mathcal{T}$
$\mathcal{T} \leftarrow \mathcal{T}_0$
Determine $q_{\max}$
Elevate elements to $q_{\max}$
$L_{\text{low}} \leftarrow \frac{\sqrt{2}}{2}$
for $i = 1$ **to** 20 **do**
  for $j = 1$ **to** 4 **do**
    if $j = 1$ ***or*** 2 **then** $L_{\text{up}} \leftarrow 2$
    else $L_{\text{up}} \leftarrow \sqrt{2}$
    $\mathcal{T}, \mathcal{M}(\vec{x}) \leftarrow \text{CollapseEdges}(\mathcal{T}, \mathcal{M}(\vec{x}), L_{\text{low}})$
    $\mathcal{T}, \mathcal{M}(\vec{x}) \leftarrow \text{SplitEdges}(\mathcal{T}, \mathcal{M}(\vec{x}), L_{\text{up}})$
    $\mathcal{T}, \mathcal{M}(\vec{x}) \leftarrow \text{SwapEdges}(\mathcal{T}, \mathcal{M}(\vec{x}), Q < 0.4)$
    $\mathcal{T}, \mathcal{M}(\vec{x}) \leftarrow \text{SwapEdges}(\mathcal{T}, \mathcal{M}(\vec{x}), Q < 0.8)$
    $\mathcal{T}, \mathcal{M}(\vec{x}) \leftarrow \text{SmoothNodes}(\mathcal{T}, \mathcal{M}(\vec{x}))$
  end
end

First, all elements in the mesh are elevated to the maximum order of elements present in the mesh. This ensures the all elements are the same order, which simplifies the implementation when operators are performed on elements of different order and allows linear elements to be made curved as needed. Although elements are all curved when performing local operations, on a group elements that were all linear to start, the boundary displacements are zero and therefore the elements stay linear but with a high-order representation. At the end of the adaptation, elements that are linear or could be made linear without inverting neighboring elements are made $q = 1$, which improves solver efficiency.

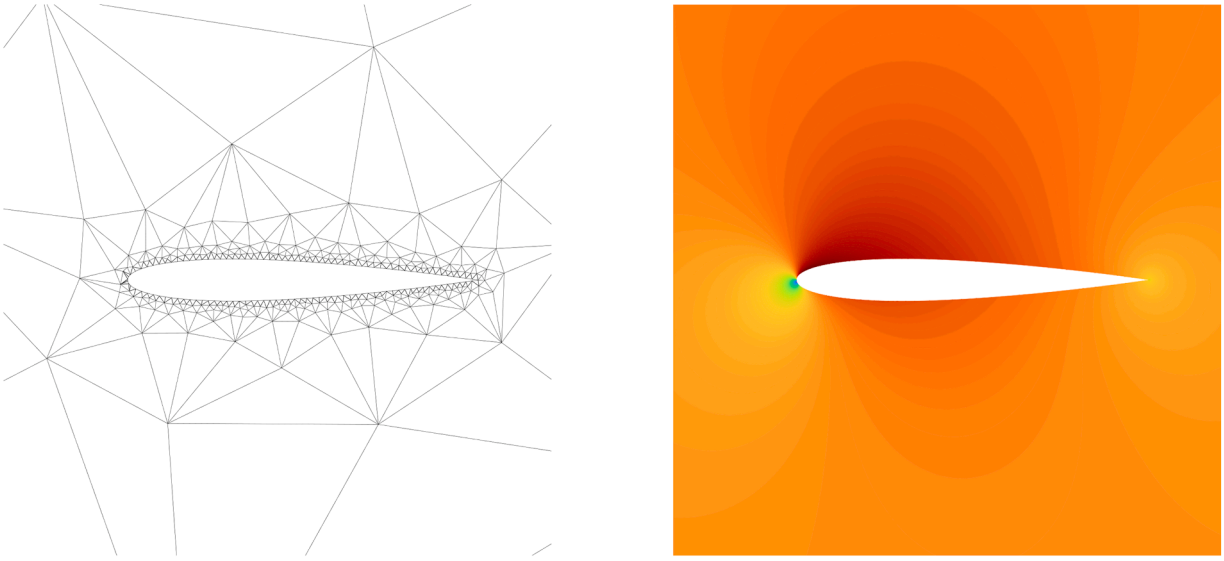
We follow an operator schedule similar to the one proposed by Caplan [55] and follow their recommendation to break each adaptation pass into two phases. The first phase sets the split target length $L_{\text{up}} = 2$, and the second phase consists of setting the split target length $L_{\text{up}} = \sqrt{2}$. Each phase sets the target length for collapses to $L_{\text{low}} = \sqrt{2}/2$, and each phase is performed twice. After each split there are two swap passes: the first targets elements with quality $Q < 0.4$, and the second targets elements with $Q < 0.8$. Following the swaps, there is a node smoothing pass to smooth all interior $q = 1$ vertices in the mesh. Therefore, each adaptation pass performs 4 collapse passes, 4 split passes, 8 swap passes, and 4 node movement passes. We perform the adaptation pass 20 times or until the number successful operations for each operator is less than 5 % of the number of edges and the number of elements changes by less than 1 %. Finally, there are limits on any new or modified edge: the metric length, L_m, must be in 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: the minimum quality of an element in the mesh can never decrease. If the restrictions are violated, the operation is rejected.

7. Results

7.1. Subsonic inviscid NACA 0012

The first test case run is inviscid flow over a NACA 0012 airfoil at a Mach number of $M_\infty = 0.2$ at an angle of attack of $\alpha = 2°$. The goal of this first test case is to show that HOEP recovers optimal convergence rates that might not be observed in more complex scenarios involving turbulence or shock capturing. We begin with an initial mesh consisting of 628 elements, which includes 124 cubic ($q = 3$) elements along the boundaries and linear ($q = 1$) elements throughout the remainder of the domain. We choose $q = 3$ for all p orders because even for $p = 1$ simulations, $q = 1$ meshes reduce the convergence rates [24,29]. For this case, we run 35 adaptation iterations at each target cost, $C^{\text{tgt}} = 2{,}000,\ 4{,}000,\ 8{,}000,\ 16{,}000,\ 32{,}000$, using $p = 1, 2, 3$ solution approximations and the average of the last five iterations as our output. The truth solution is obtained with BAMG [56] at a target cost of $C^{\text{tgt}} = 128{,}000$ using a $p = 4$ solution approximation. The initial mesh and Mach number contours from the exact solution are shown in Fig. 10.

The meshes for $C^{\text{tgt}} = 2{,}000$ are shown in Fig. 11 and the meshes for $C^{\text{tgt}} = 32{,}000$ are shown in Fig. 12. At both target costs both HOEP and BAMG add refinement is added to the leading and trailing edges of the airfoil to capture the stagnation points. The elements on the upper and lower surfaces of the airfoil are much larger as these regions only have a small impact on drag prediction predicted in MOESS. Most elements in the mesh are isotropic as there are no shocks or boundary layers to capture. Because of the large isotropic



(a) Initial mesh for the inviscid NACA 0012 case.

(b) Mach contours (0–0.27) for the inviscid NACA 0012 case.

Fig. 10. Initial mesh and exact solution Mach contours for the inviscid NACA 0012 case.

elements on the boundary of the airfoil, the meshes do not require curved interior elements. In all cases, only the elements on the wall remained $q = 3$ after adaptation. The meshes from HOEP and BAMG are visually identical because both methods have high metric conformity.

The drag coefficient error convergence is shown in Fig. 13. The figure shows that HOEP and BAMG achieve the expected super-convergence rate of $2p + 1$ for an integrated outputs such integrating forces on the airfoil to compute drag. In addition to exhibiting the same error convergence rates, the absolute error is almost identical between the two cases. This case also shows that HOEP consistently overshoots the desired number of degrees of freedom by 2.6 % while BAMG consistently undershoots by 2.4 %. Having edge primitive operations overshoot the target number of elements is consistent with previous work.

The distributions of metric edge lengths in the final adapted meshes for the $C^{\text{tgt}} = 32{,}000$ are shown in Fig. 14, with edges $L_m > 2$ clipped to 2. For BAMG we interpolate the desired metric from the background mesh to the adapted mesh and then compute edge lengths. Both HOEP and BAMG have 99 % of edges in the quasi-unit range indicating high metric conformity. The HOEP meshes tend to have more short edges while the BAMG meshes tend to have longer edges. This leads to HOEP having more degrees of freedom and BAMG having fewer degrees of freedom show in Fig. 13.

This simple test case shows that HOEP achieves both the same absolute error and error convergence rates as using BAMG with mesh re-curving.

7.2. Viscous laminar NACA 0012

The second case we tested was 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 $Re = 5{,}000$. We use the same initial starting mesh as in the previous case, shown in Fig. 10a, with a $p = 1, 2, 3$ solution approximation. Mach number contours for this case are shown in Fig. 15.

This test case was run for 30 adaptation iterations targeting drag error at each target cost, $C^{\text{tgt}} = 2{,}000$, 4,000, 8,000, 16,000, 32,000, 64,000, 128,000, using $p = 1, 2, 3$ solution approximations. The drag and number of degrees of freedom are averaged over the last five iterations and shown in Fig. 16.

This test case was run for 30 adaptation iterations targeting drag error at each target cost, $C^{\text{tgt}} = 2{,}000$, 4,000, 8,000, 16,000, 32,000, 64,000, 128,000, using $p = 1, 2, 3$ solution approximations. The drag and number of degrees of freedom are averaged over the last 5 iterations and shown in Fig. 16. The HOEP and BAMG adapted cases converge to the same drag coefficient. The difference between the drag for the $p = 3$ and $p = 2$ cases at $C^{\text{tgt}} = 128{,}000$ is 9.6×10^{-7} and 1.7×10^{-5} drag counts, respectively, and the difference for $p = 1$ is 1.6×10^{-3} drag counts.

An “exact” solution is generated by uniformly refining the final mesh twice from HOEP $p = 3$ with $C^{\text{tgt}} = 128{,}000$ and computing a $p = 4$ solution on that mesh. This “exact” solution is used to compute the drag error shown in Fig. 17.

For $p = 1$ and $p = 2$, the convergence is asymptotic, and HOEP and BAMG exhibit convergence rates between $2p$ and $2p + 1$. For $p = 3$, the convergence is much noisier because the solutions have errors lower than the $p = 1$ and $p = 2$ solutions while using fewer elements for a target cost. Using fewer elements results in the error being more sensitive to the placement of individual elements, causing noise at these low error values.

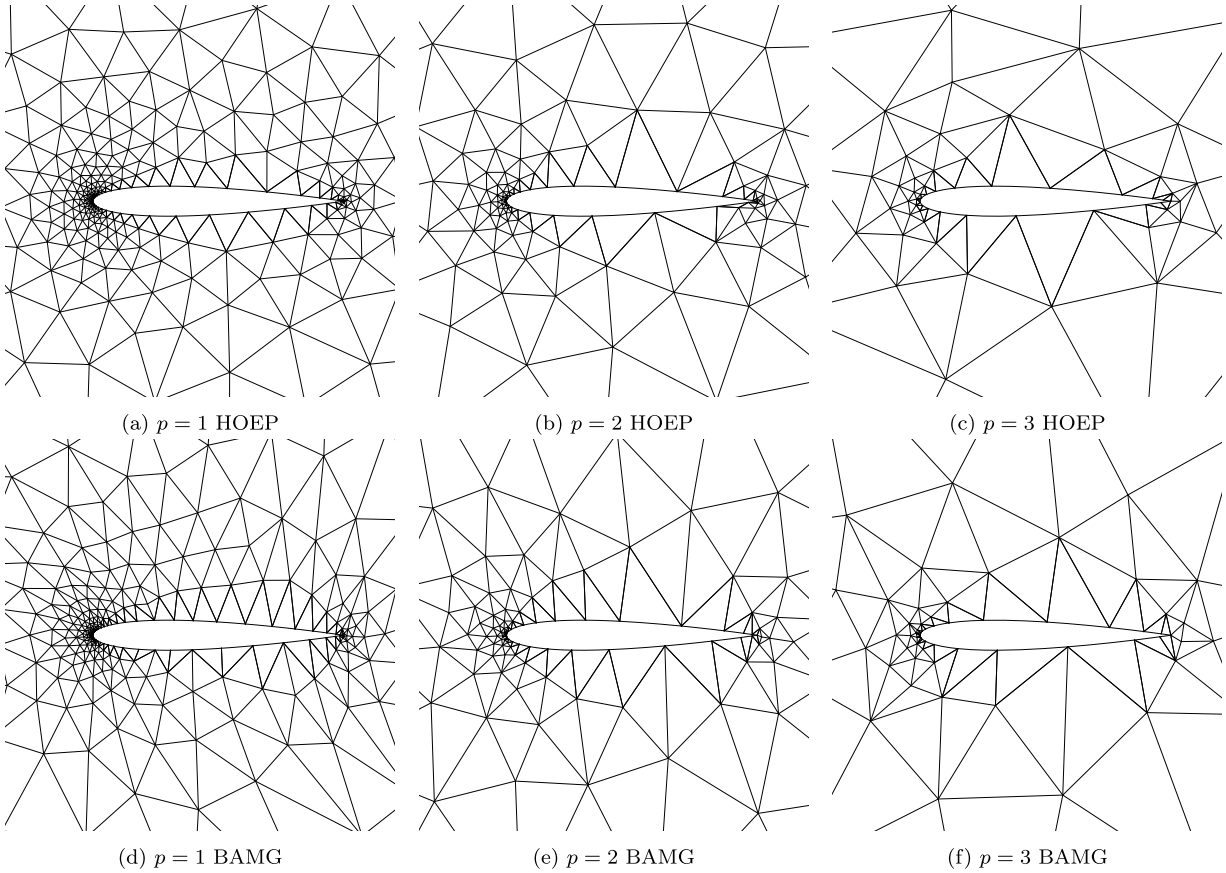


Fig. 11. Final adapted meshes for the inviscid NACA 0012 case with $C^{\text{tgt}} = 2{,}000$.

Table 1
Drag error convergence rates for the viscous NACA 0012 case.

	$p = 1$	$p = 2$	$p = 3$
HOEP	2.7	5.1	6.1
BAMG	2.6	4.4	5.8

The slopes of the error convergence between 16,000 and 32,000 are summarized in Table 1. For $p = 1$ and $p = 2$, both HOEP and BAMG exhibit rates higher than the expected optimal convergence rate for viscous problems of $2p$. This effect is due to the problem being dominated by convection: the inviscid convergence rate of $2p + 1$ is apparent until the viscous errors, initially of lower magnitude, dominate. For $p = 3$, both HOEP and BAMG exhibit the expected $2p$ convergence rate in the range of 16,000 to 32,000 degrees of freedom, but the convergence pattern is noisier than at lower polynomial orders.

The meshes for the $C^{\text{tgt}} = 32{,}000$ case are shown in Fig. 18. The adaptation for both HOEP and BAMG targets the same regions around the airfoil, such as the leading and trailing edges, as well as the boundary layer and wake.

This case requires some anisotropy near the airfoil boundary, which demands curved interior elements, though fewer than 1 % of interior elements actually require curvature. Table 2 summarizes the amount of curved elements in each of the final meshes.

The distributions of metric edge lengths in the final adapted meshes for the $C^{\text{tgt}} = 32{,}000$ are shown in Fig. 19, with edges $L_m > 2$ clipped to 2. For BAMG, we interpolate the desired from the background mesh to the adapted mesh and then compute edge lengths. Both HOEP and BAMG have 99 % of edges in the quasi-unit range, indicating high metric conformity. HOEP tends to have shorter edges than BAMG inside the quasi-unit range. The BAMG meshes have more edges that are above the range than below the range, while HOEP tends to have more short edges than long edges. This leads to HOEP having more degrees of freedom and BAMG having fewer degrees of freedom, shown in Fig. 16. HOEP shows an advantage in accuracy as the longer edges indicate regions that need refinement, so HOEP is adding elements where they need to go but is not removing elements where they are not needed. This increases computational expense but helps ensure refinement is in the most important regions. This case does not show HOEP and BAMG having similar errors, but for more complex cases, refinement is more important than coarsening for accuracy.

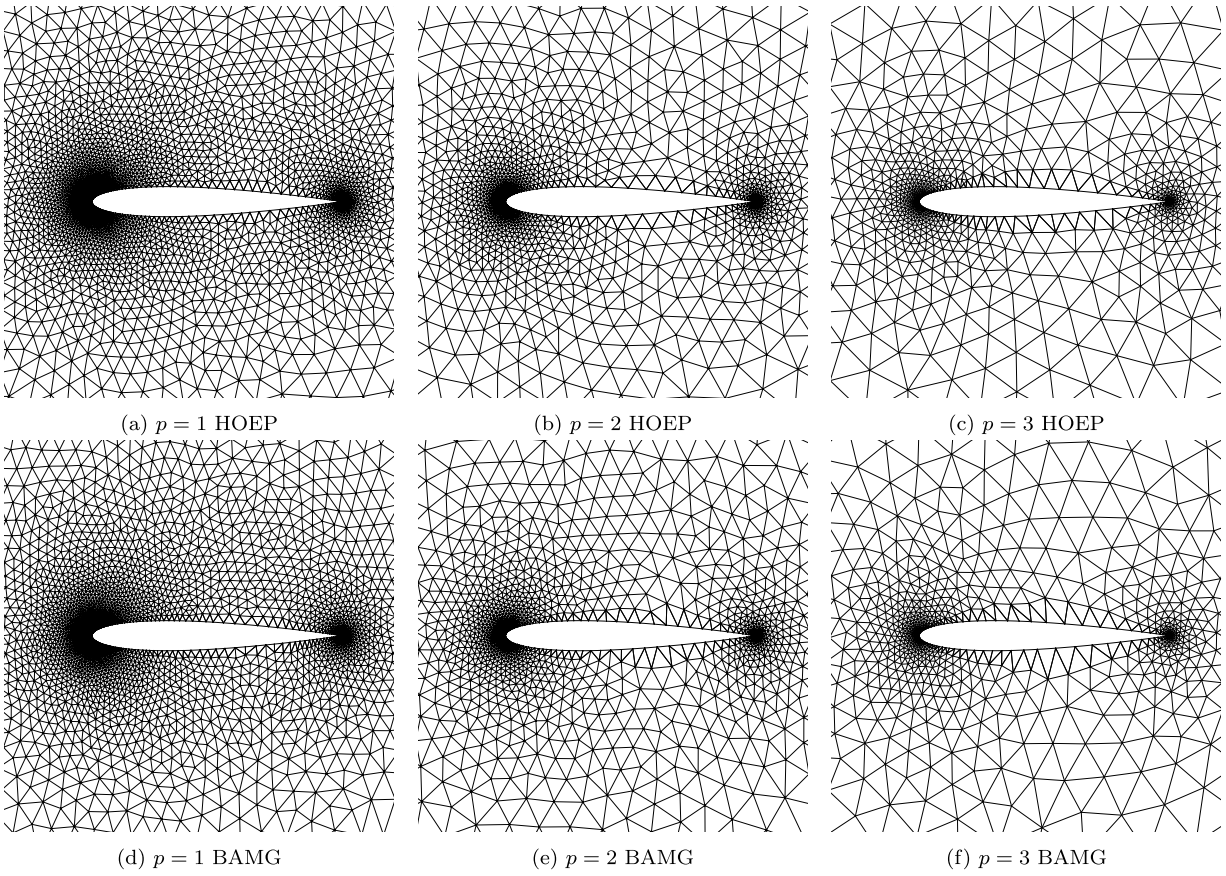


Fig. 12. Final adapted meshes for the inviscid NACA 0012 case with $C^{\text{tgt}} = 32{,}000$.

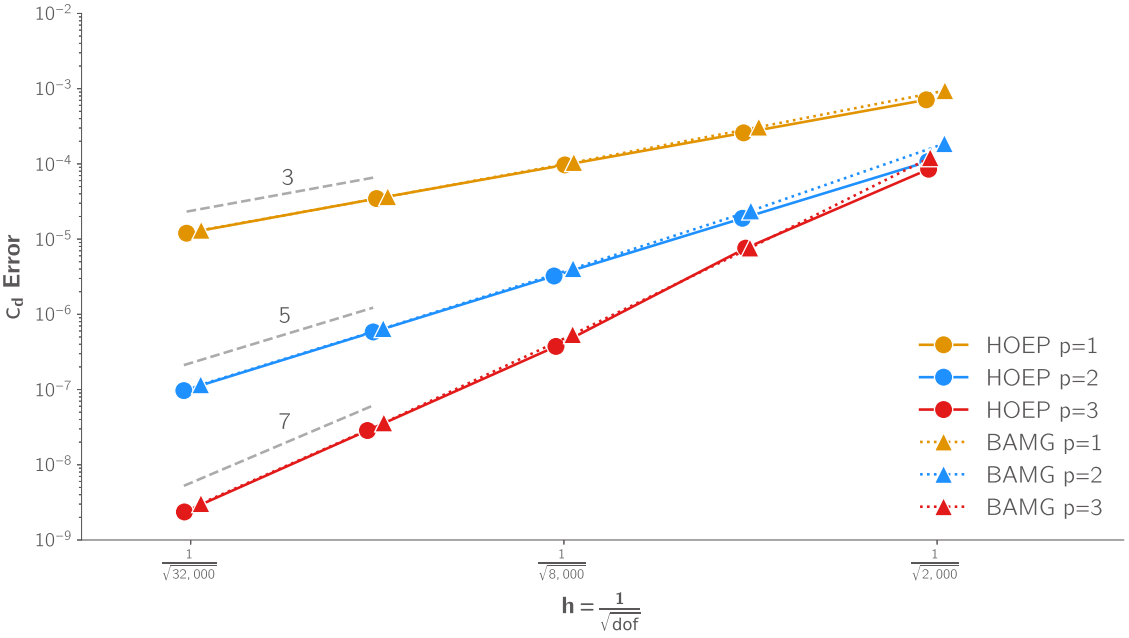
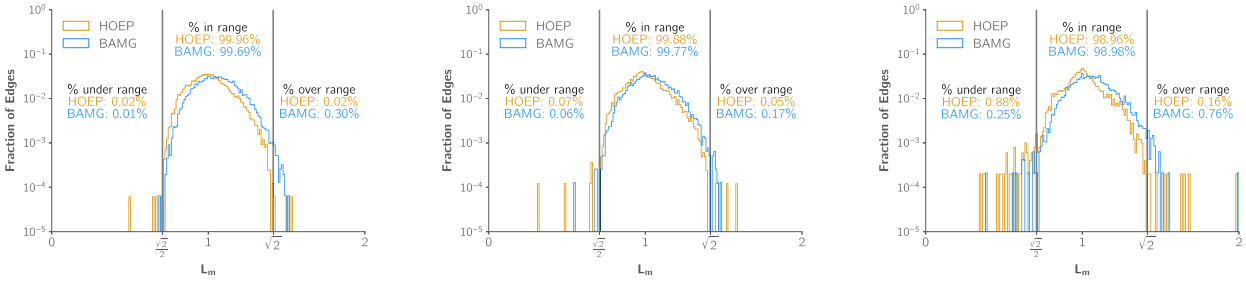


Fig. 13. Drag error convergence for the inviscid NACA 0012 case.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 14. Edge length distribution for the inviscid NACA 0012 case for $C^{\text{tgt}} = 32{,}000$ final meshes.

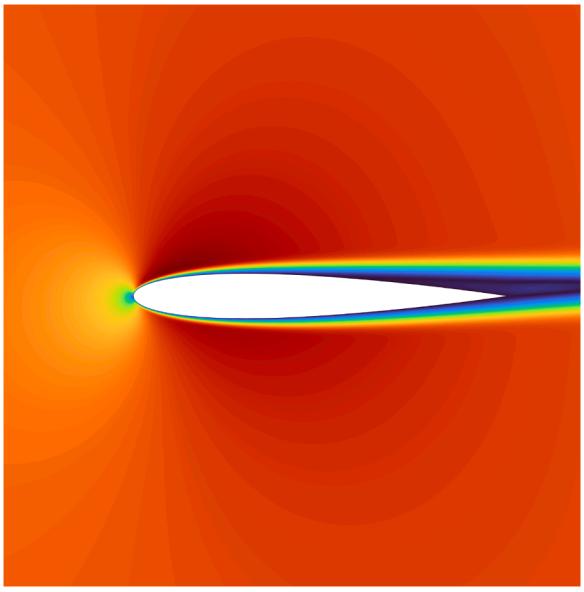


Fig. 15. Mach contours (0–0.62) for the viscous NACA 0012 case.

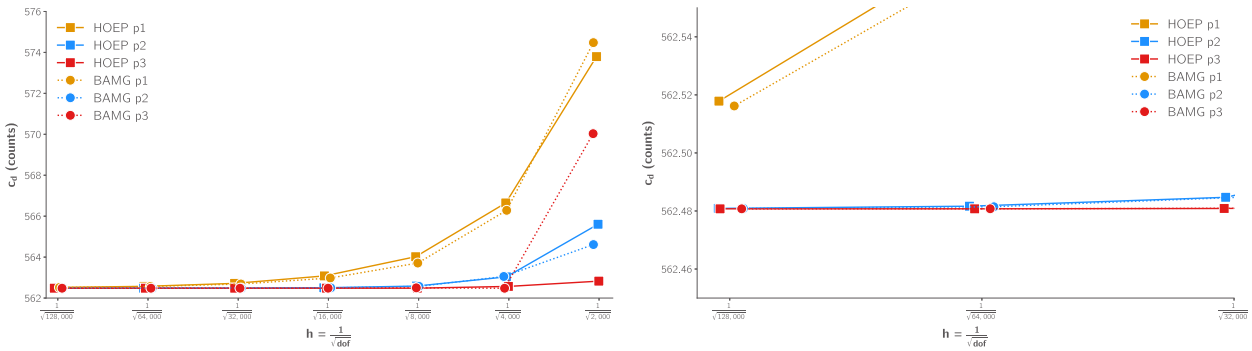
Table 2
Curved element mesh summary for the viscous NACA 0012 case.

Method	p	# Elements	# Curved Elements	# Curved Boundaries	# Interior Curved
HOEP	$p = 1$	11,101	142	123	19
	$p = 2$	5435	177	173	4
	$p = 3$	3281	111	103	8
BAMG	$p = 1$	10,454	170	118	52
	$p = 2$	5222	144	142	2
	$p = 3$	3158	102	92	10

This case shows that for a case that is more complicated than an inviscid flow, we achieve the same near-optimal convergence rates as using BAMG. Additionally, it shows that using a large number of degrees of freedom, HOEP and BAMG converge to the same value for the output on which we are adapting.

7.3. Turbulent transonic RAE 2822

The third 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 [57] and element-wise constant artificial viscosity for shock capturing. The high Reynolds number for this case requires cells with a high aspect ratio to adequately capture the boundary layer without wasting degrees of freedom. We run 30 adaptive iterations at target costs of $C^{\text{tgt}} = 2{,}000$, 4,000, 8,000, 16,000, 24,000, 32,000, 48,000, 64,000, 96,000, 128,000 using $p = 1, 2, 3$ solution approximations. We use HOEP for $q = 3$ and



(a) All $\mathcal{C}^{\text{tgt}}$. (b) Zoom around highest $\mathcal{C}^{\text{tgt}}$.

Fig. 16. Drag coefficient convergence for the viscous NACA 0012 case.

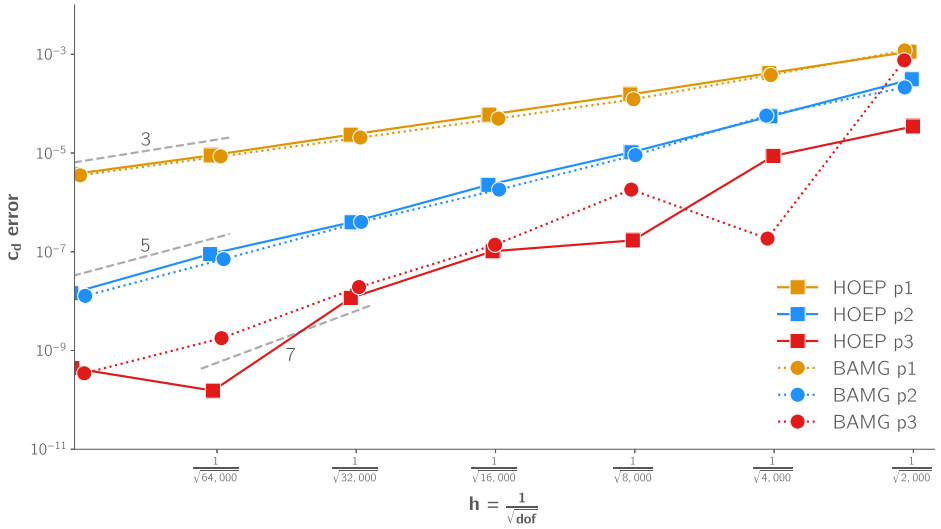


Fig. 17. Drag error convergence for the viscous NACA 0012 case.

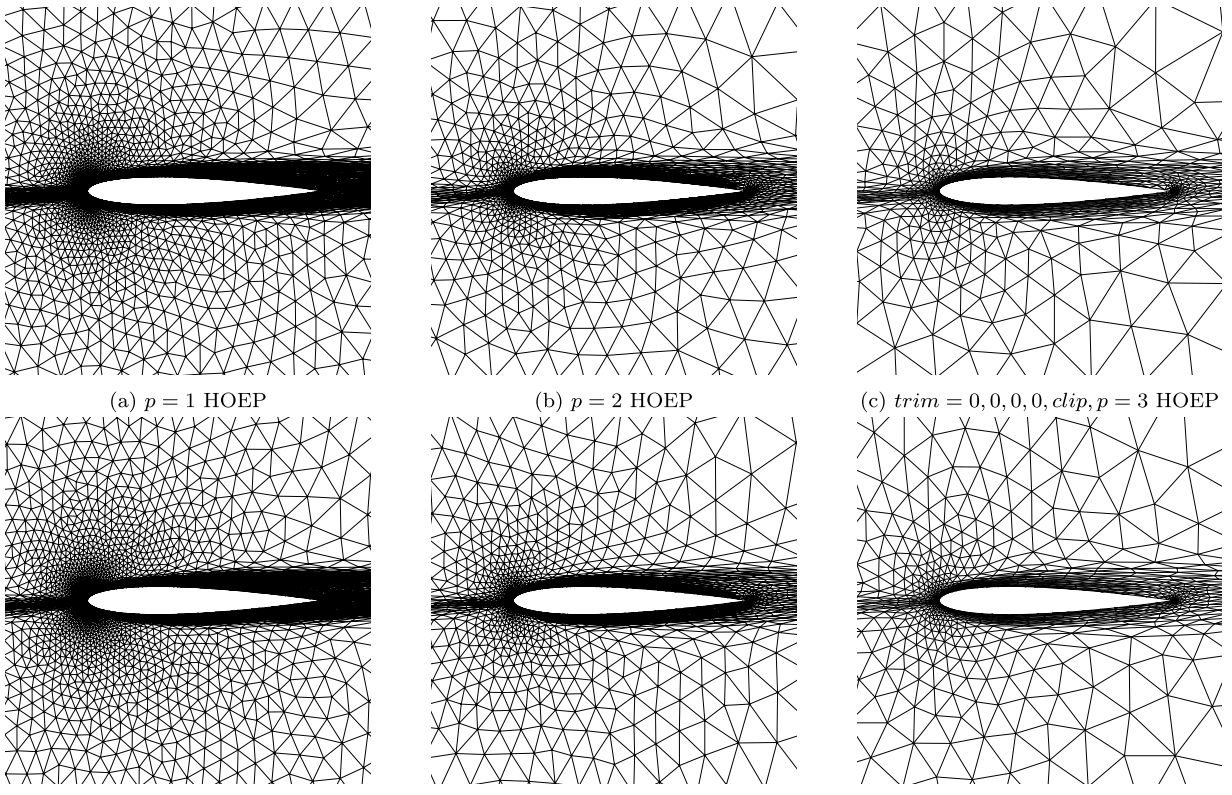
compare it with BAMG with linear elasticity re-curving. We attempted to try $q = 1$ HOEP with re-curving as an additional comparison. However, there were consistent failures during the re-curving and making the mesh linear stages due to high anisotropy. The initial mesh as well as Mach number contours for the $\mathcal{C}^{\text{tgt}} = 128{,}000$ with $p = 3$ using HOEP are shown in Fig. 20.

The convergence for the drag coefficient is plotted in Fig. 21. The final meshes for the $\mathcal{C}^{\text{tgt}} = 128{,}000$ are shown in Fig. 22.

Fig. 21 shows that HOEP consistently overshoots the target number of degrees of freedom while BAMG consistently undershoots it. At costs lower than 32,000, there is a significant difference between HOEP and BAMG. This is because there are not enough degrees of freedom to capture all flow features that impact drag prediction. However, after 32,000 degrees of freedom, the variation between HOEP and BAMG for a given approximation order is within 0.2 drag counts. At the highest target cost, $\mathcal{C}^{\text{tgt}} = 128{,}000$, the drag predictions for the $p = 3$ and $p = 2$ cases are within 0.04 drag counts. Additionally, BAMG consistently produces meshes that undershoot the target number of degrees of freedom, while HOEP consistently overshoots the target. This does not significantly impact outputs as the degrees of freedom are usually within 3 %, and both converge to the same drag value.

The final meshes at $\mathcal{C}^{\text{tgt}} = 128{,}000$ are shown in Fig. 22. Both HOEP and BAMG add anisotropy to the boundary layer, stagnation streamline, shock, wake, and λ feature in the adjoint field. The meshes for the $p = 1$ and $p = 2$ cases are nearly identical. HOEP targets the stagnation streamline slightly more than BAMG. The $p = 3$ meshes both have anisotropic adaptation behind the shock. This is likely due to using piecewise constant shock capturing and having elements that are poorly aligned with the shock. These effects cause entropy to be generated, which is then advected downstream and later targeted for anisotropic refinement due to differences in entropy produced.

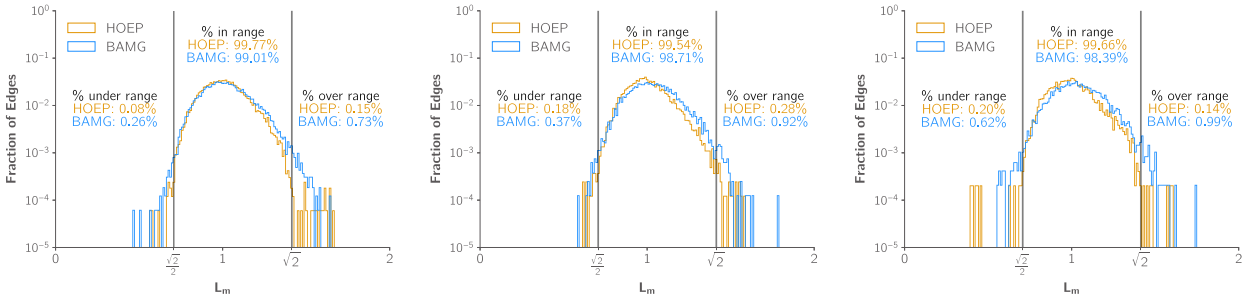
Due to the high anisotropy required around the boundary of the airfoil, more interior elements need to be curved than in the previous cases. Table 3 summarizes the number of curved elements in each of the final meshes. Both BAMG and HOEP have a similar number of elements on the wall for a given p, but BAMG consistently has a much higher number of curved interior elements. This is because HOEP only makes elements that need to be curved when doing local operations near the boundary, while BAMG is paired



(a) $p = 1$ HOEP (b) $p = 2$ HOEP (c) $trim = 0, 0, 0, 0, clip, p = 3$ HOEP

(d) $p = 1$ BAMG (e) $p = 2$ BAMG (f) $p = 3$ BAMG

Fig. 18. Final adapted meshes for the viscous NACA 0012 case with $C^{\text{tgt}} = 32{,}000$.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

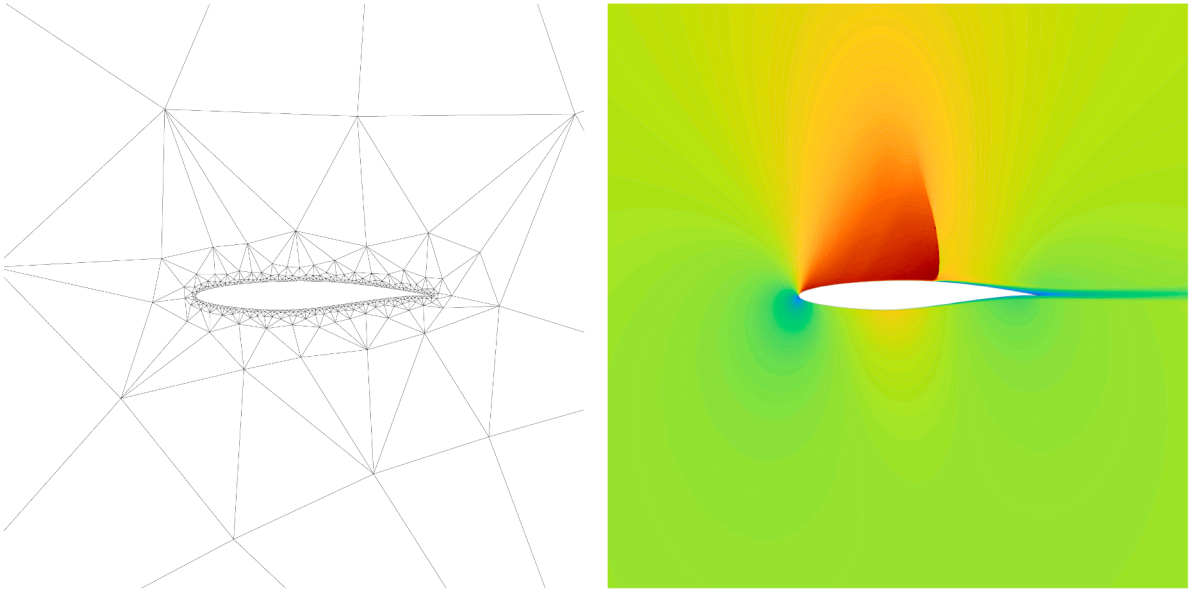
Fig. 19. Edge length distribution for the viscous NACA 0012 case for $C^{\text{tgt}} = 32{,}000$ final meshes.

with linear elasticity to propagate curvature further from the boundary. Having more curved elements in the domain reduces solver efficiency by requiring a higher quadrature order for element and face integrations in the residual evaluation.

Fig. 23 shows the final mesh around the leading edge of the $p = 3$ $C^{\text{tgt}} = 128{,}000$ case for both HOEP and BAMG. Curved elements are highlighted blue, and we see that BAMG has more curved elements in this region than HOEP. In this region, the HOEP mesh exhibits curved elements mostly on the wall, with some first-level neighbors and only one second-level neighbor. The BAMG mesh, on the other hand, has all wall elements and their first-level neighbors curved, and most second-level neighbors curved.

The average CPU times to generate the meshes the last five meshes for the $C^{\text{tgt}} = 128{,}000$ case with HOEP and BAMG are shown in Table 4. The time for BAMG includes the call to BAMG as well as linear elasticity mesh curving. These times show that HOEP has a computational cost that is lower than BAMG with re-curving. The additional cost of re-curving the mesh is traded off with the more expensive nonlinear Jacobian determinant checks for each local operator. However, HOEP is more robust as the mesh can never become invalid.

This case requires cells with high aspect ratios in the boundary layer. Having demonstrated HOEP's capability to match BAMG in drag predictions, we now examine how the resulting mesh conforms to the desired metric. We analyze the metric edge lengths in



(a) Initial mesh for the RAE 2822 case. (b) Mach contours for the RAE 2822 case (0–1.4).

Fig. 20. Initial mesh and exact solution Mach contours for the RAE 2822 case.

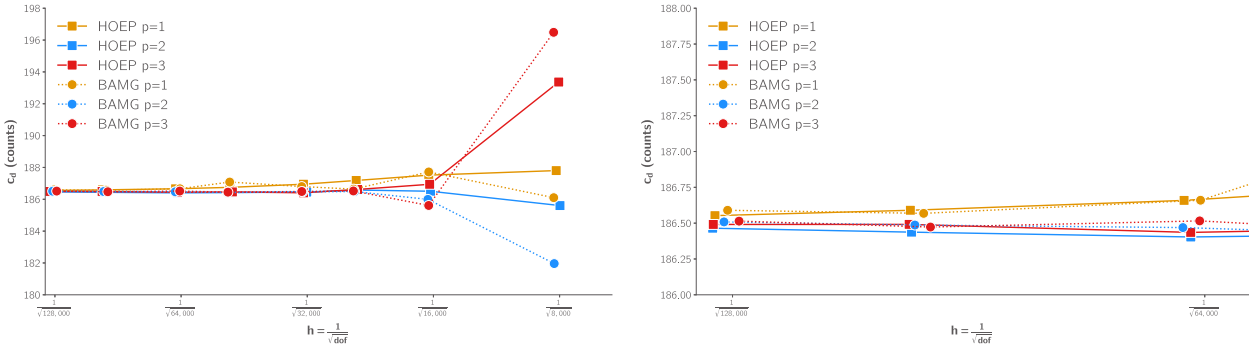


Fig. 21. Drag coefficient convergence for the RAE 2822 case.

Table 3
Curved element mesh summary for the RAE 2822 case.

Method	p	# Elements	# Curved Elements	# Curved Boundaries	# Interior Curved
HOEP	$p = 1$	43,539	876	637	239
	$p = 2$	21,925	1048	827	221
	$p = 3$	13,070	520	442	78
BAMG	$p = 1$	43,040	1572	756	816
	$p = 2$	21,219	989	765	224
	$p = 3$	12,652	1224	434	790

Table 4
Mesh adaptation CPU times (in seconds) for $C^{tgt} = 128{,}000$.

	$p = 1$	$p = 2$	$p = 3$
HOEP	148.5	60.3	27.9
BAMG	196.0	77.2	32.9

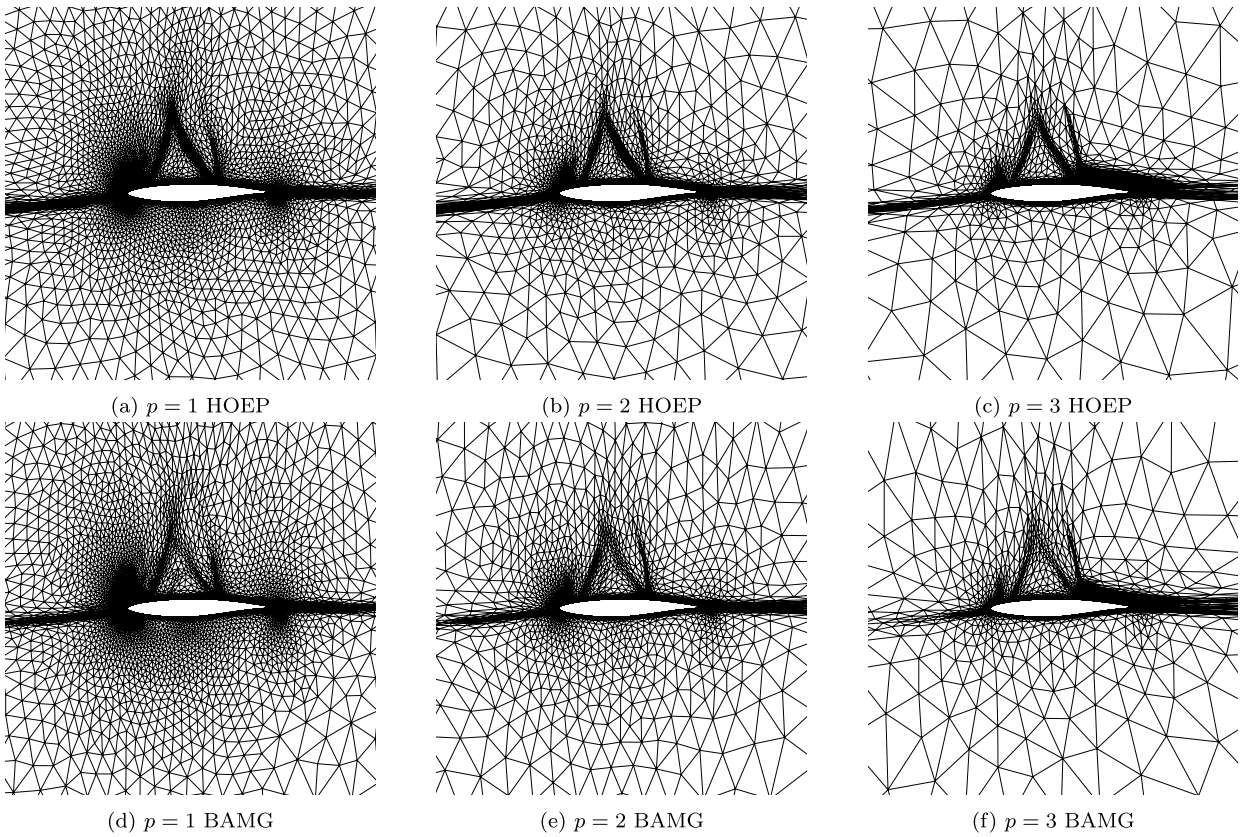


Fig. 22. Final adapted meshes for the RAE 2822 case with $C^{\text{tgt}} = 128{,}000$.

the final mesh generated for $C^{\text{tgt}} = 128{,}000$. For BAMG, we interpolate the desired metric from the background mesh to the adapted mesh and then compute edge lengths. Fig. 24 shows the distribution of edges in both meshes, with any edge length $L_m > 2$ clipped to length 2.

HOEP and BAMG both achieve 97 % or more edges in the quasi-unit range, indicating excellent metric conformity. However, HOEP consistently has more edges below the $\sqrt{2}/2$ limit, while BAMG consistently has more edges greater than the $\sqrt{2}$ limit. This results in HOEP meshes having more elements because having shorter edges means that edges need to be collapsed and elements need to be removed. On the other hand, BAMG undershoots the target cost because the longer edges indicate that more refinement is needed, as shown in Fig. 21.

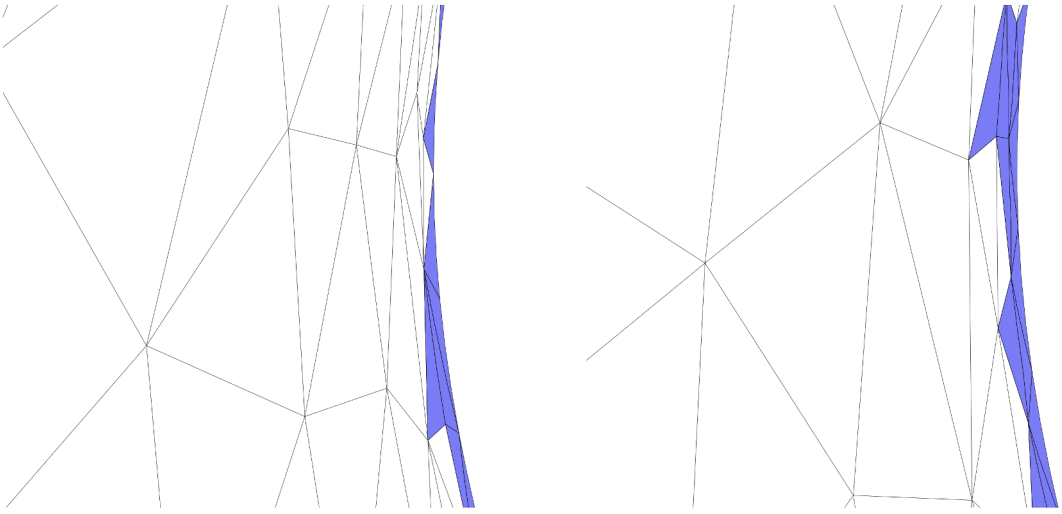
Examining at element quality for the final meshes at $C^{\text{tgt}} = 128{,}000$ in Fig. 25 shows that HOEP produces meshes with higher metric quality. We see that HOEP has more elements around $Q_m = 1$ than the BAMG meshes. The quality measure only takes into account the $q = 1$ area and edge lengths of the elements, but we see that HOEP has a larger minimum metric, and most elements are above the 0.4 and 0.8 thresholds used for swapping. The BAMG meshes for $p = 1$ and $p = 2$ have many elements with $Q_m < 0.4$ which is likely due to having small edges with $L_m < 0.2$ or long edges $L_m > 2$ as shown in Fig. 24. At $p = 3$, the BAMG mesh improves the minimum quality and is more similar to HOEP. The large differences are likely because HOEP uses this quality metric to optimize the mesh, whereas BAMG uses different measures for metric conformity.

Next, we examine the element aspect ratios. Fig. 26 shows a histogram of the distribution of desired element aspect ratios given to HOEP and BAMG, as well as the actual distribution of element aspect ratios for the final iteration at $C^{\text{tgt}} = 128{,}000$ for $p = 1, 2, 3$.

This case highlights HOEP's ability to generate high-aspect ratio curved meshes efficiently and match the performance of global remeshing with re-curving. HOEP predicted drag for this case within 0.04 drag counts of BAMG combined with linear elasticity re-curving and has a slightly better computational time without any additional mesh re-curving.

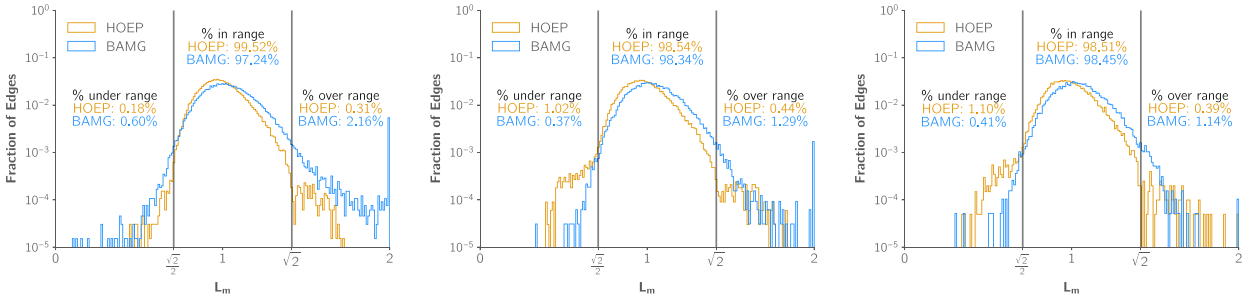
7.4. Subsonic turbulent MDA 30P-30N multi-element airfoil

The final test case is the McDonnel Douglas Aerospace (MDA) three-element airfoil (30P-30N). This airfoil is tested at a free stream Mach number $M_\infty = 0.2$ and a Reynolds number, based on the chord of the baseline undeflected airfoil, of $Re = 558{,}800$ at an angle of attack of $\alpha = 5°$. We run 30 adaptive iterations targeting drag error at target costs of $C^{\text{tgt}} = 8{,}000, 16{,}000, 24{,}000, 32{,}000, 48{,}000, 64{,}000, 96{,}000, 128{,}000$ using solution approximations $p = 1, 2, 3$. We use HOEP for $q = 3$ and compare it with BAMG with linear elasticity re-curving. We attempted to try $q = 1$ HOEP with re-curving as an additional comparison, but



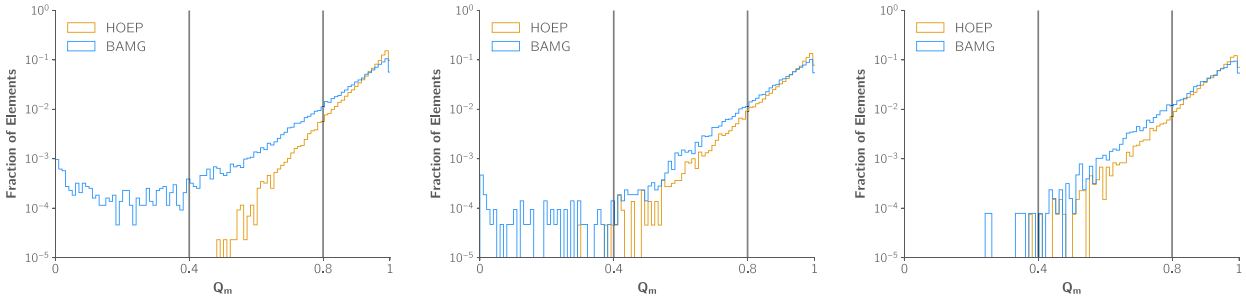
(a) HOEP (b) BAMG

Fig. 23. Final adapted meshes for the RAE 2822 case with $p = 3$ and $C^{\text{tgt}} = 128{,}000$ zoomed around the leading edge.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 24. Edge length distribution for the RAE 2822 case for $C^{\text{tgt}} = 128{,}000$ final meshes.

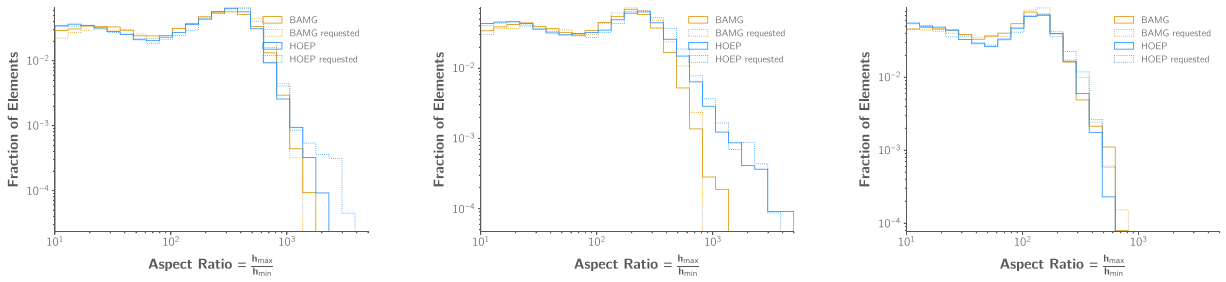


(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 25. Element quality distribution for the RAE 2822 case for $C^{\text{tgt}} = 128{,}000$ final meshes.

due to high anisotropy, we observed consistent failure during re-curving and making the mesh linear. Even though the final meshes were valid linear meshes, the curving step via linear elasticity repositioned $q = 1$ nodes such that the $q = 1$ elements were invalid. The initial mesh has $2,775$ elements with 552 $q = 3$ elements on the boundaries, and the rest of the elements are $q = 1$ and are shown in Fig. 27.

The drag convergence is shown in Fig. 28. HOEP and BAMG have similar drag predictions and are within 0.1 drag counts. The final adapted meshes are shown in Fig. 29. Both methods move anisotropy into the boundary layer, and stagnation streamlines each element of the airfoil. However, HOEP achieves more anisotropy and better alignment with the stagnation streamlines.

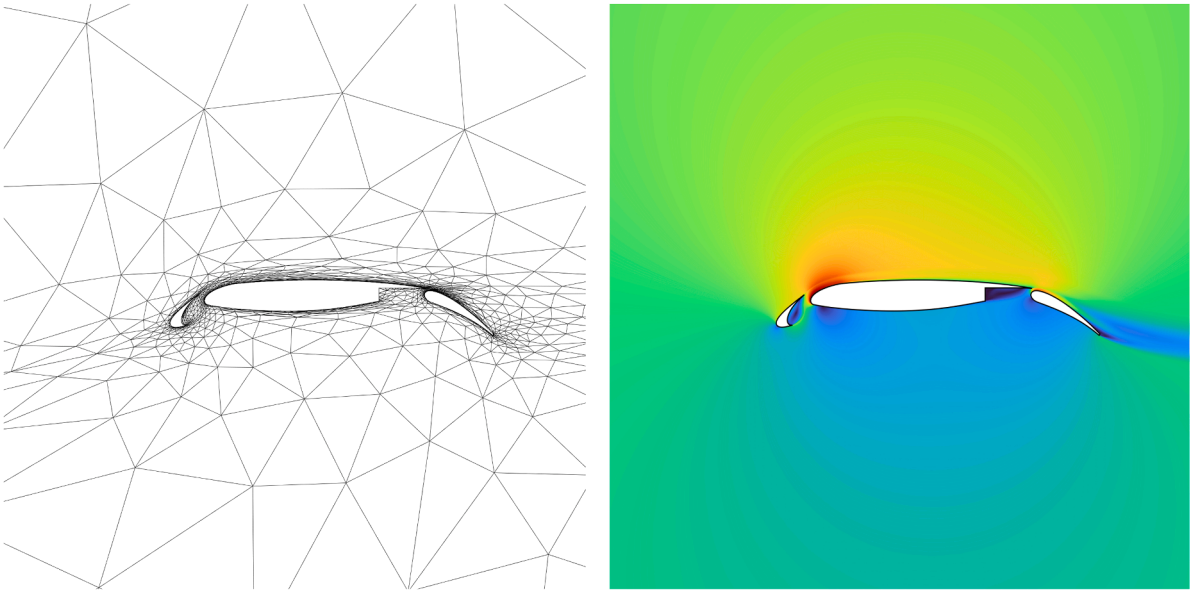


(a) Requested and resulting aspect ratios for $p = 1$

(b) Requested and resulting aspect ratios for $p = 2$

(c) Requested and resulting aspect ratios for $p = 3$

Fig. 26. Requested metric field element aspect ratio compared to mesh implied metric field for the final $C^{\text{tgt}} = 128{,}000$ RAE 2822 meshes.



(a) Initial mesh for the MDA 30P-30N case.

(b) Mach contours for the MDA 30P-30N case (0–0.5).

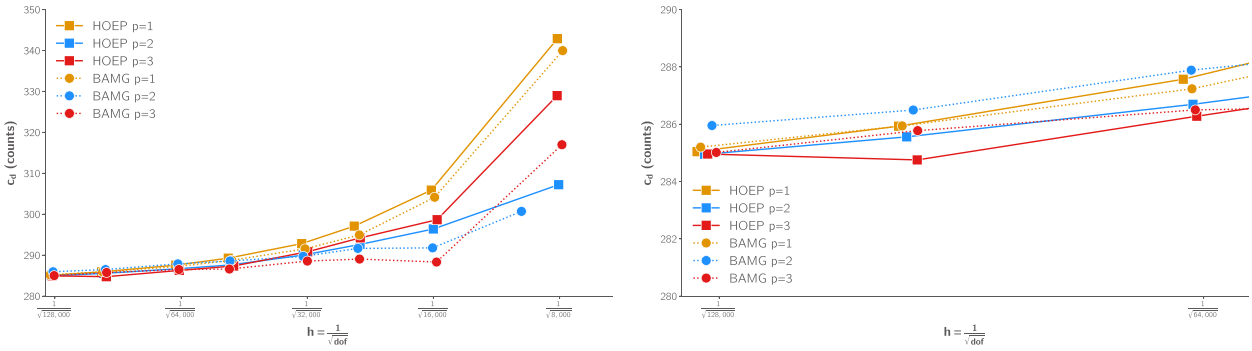
Fig. 27. Initial mesh and fine solution Mach contours for the MDA 30P-30N case.

Fig. 30 shows the $p = 3$ mesh zoomed in around the flap leading edge. The HOEP mesh achieves more anisotropy than the BAMG mesh. For the BAMG mesh, we also plot the requested metric field on the boundary of the mesh. The BAMG mesh does not precisely conform to the requested metric field and has some elements that are not as anisotropic along the boundary. This could be due to the boundary curvature in this region. Because the values in the step matrix are limited, it takes many adaptive iterations to attain high levels of anisotropy starting from an isotropic mesh. BAMG does not quite achieve the requested anisotropy, so it never drifts to the highly anisotropic leading edge that HOEP achieves.

Table 5 summarizes the numbers of curved elements in each of the final meshes at $C^{\text{tgt}} = 128{,}000$. BAMG has more than double the number of elements on the wall than HOEP does, but neither of them has a large number of elements curved on the interior of the mesh. BAMG having double the number of elements is likely due to not being able to generate as anisotropic meshes as HOEP, which results in needing more wall elements to capture the boundary layer.

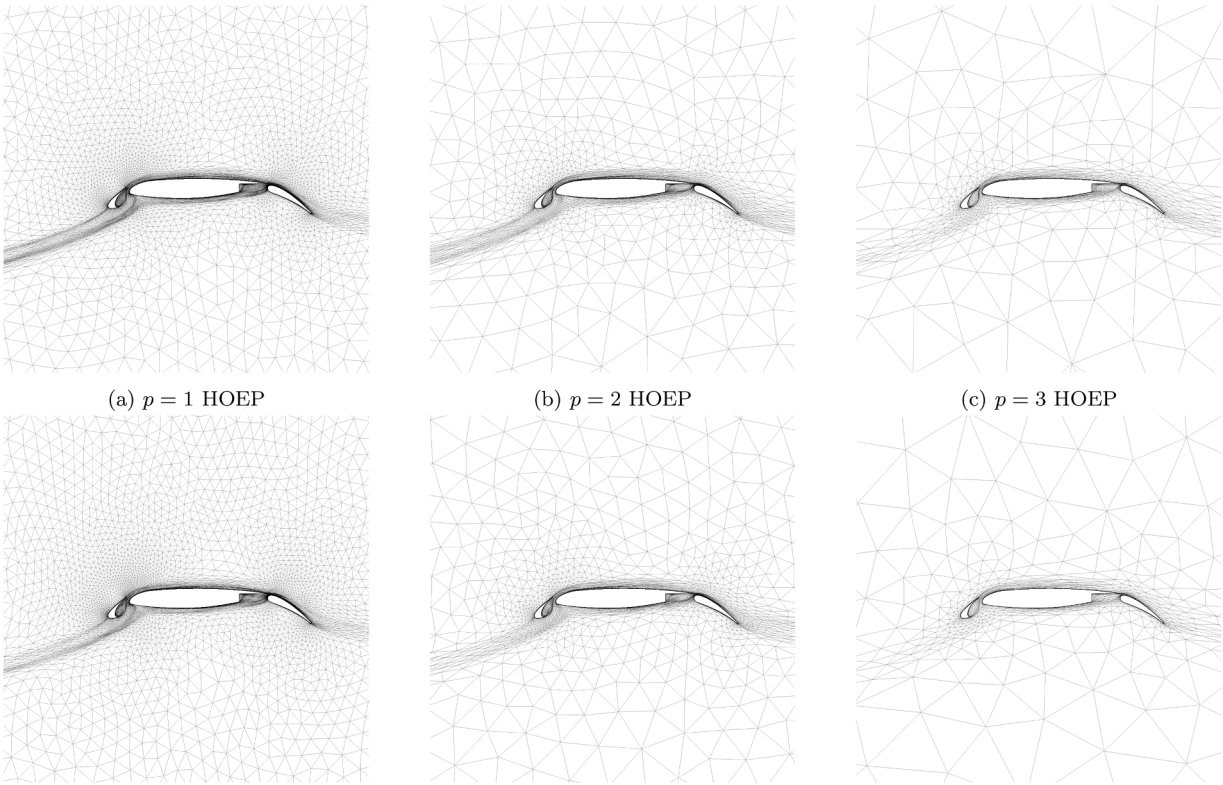
Fig. 31 shows the mesh around the upper surface of the main airfoil. For both the HOEP and BAMG meshes, only the elements on the boundary are curved. Additionally, we see that for the same view window, the HOEP mesh has one element on the boundary in this region, while the BAMG mesh has three elements on the boundary. This difference persists on other portions of the airfoil, which leads to the BAMG meshes having more elements on the curved boundaries.

Fig. 32 shows the metric edge lengths in the final adapted meshes for the $C^{\text{tgt}} = 128{,}000$. In this case, we see lower metric conformity than in previous cases. As the solution order p increases, both HOEP and BAMG see a decrease in the fraction of edges inside the target range. HOEP and BAMG both have a large increase in the fraction of edges that are too short while only a moderate increase in edges that are too long. Additionally, HOEP has no edges that are greater than $L_m > 2$, indicating that refinement is added to the correct regions of the domain. The $p = 2$ meshes have the largest disparity between metric conformity, with HOEP achieving 96 % of



(a) All $\mathcal{C}^{\text{tgt}}$. (b) Zoom around highest $\mathcal{C}^{\text{tgt}}$.

Fig. 28. Drag coefficient convergence for the MDA 30P-30N case.



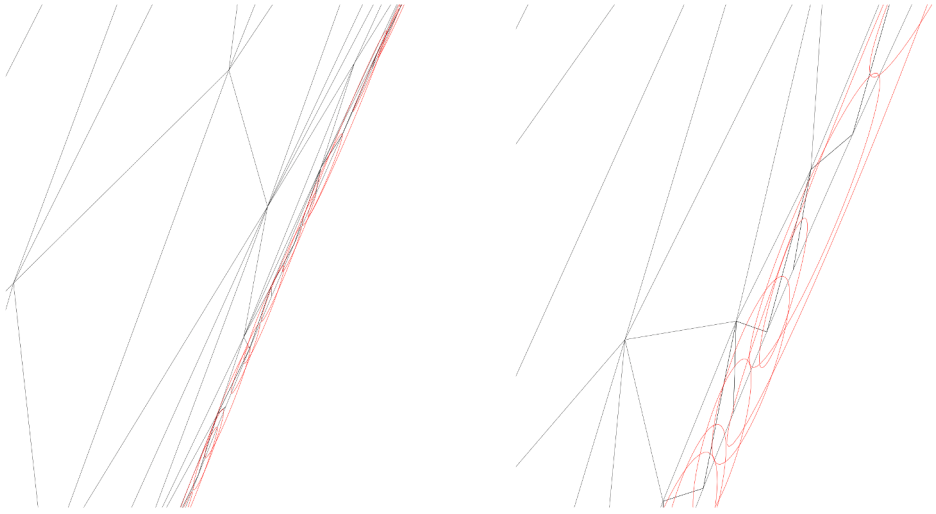
(a) $p = 1$ HOEP (b) $p = 2$ HOEP (c) $p = 3$ HOEP

(d) $p = 1$ BAMG (e) $p = 2$ BAMG (f) $p = 3$ BAMG

Fig. 29. Final adapted meshes for the MDA 30P-30N case with $\mathcal{C}^{\text{tgt}} = 128{,}000$.

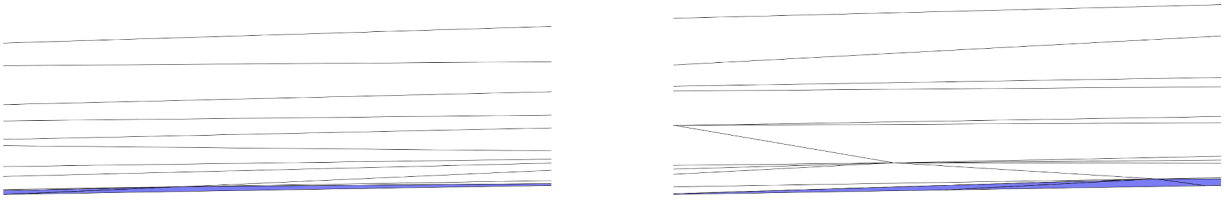
Table 5
Curved element mesh summary for the MDA 30P-30N case.

Method	p	# Elements	# Curved Elements	# Curved Boundaries	# Interior Curved
HOEP	$p = 1$	44,003	2263	2205	58
	$p = 2$	21,939	1977	1835	142
	$p = 3$	12,375	1992	1911	81
BAMG	$p = 1$	44,106	4648	4598	50
	$p = 2$	21,980	4353	4328	25
	$p = 3$	13,110	4168	4158	10



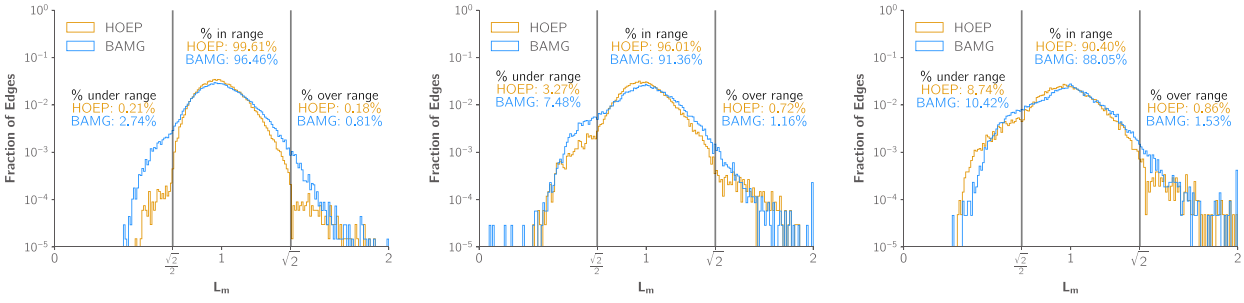
(a) HOEP (b) BAMG

Fig. 30. Final adapted meshes for the MDA 30P-30N case with $\mathcal{C}^{\text{tgt}} = 128{,}000$ zoomed in around the leading edge of the flap.



(a) HOEP (b) BAMG

Fig. 31. Final adapted meshes for the MDA 30P-30N case with $\mathcal{C}^{\text{tgt}} = 128{,}000$ zoomed around the upper surface of the main airfoil.

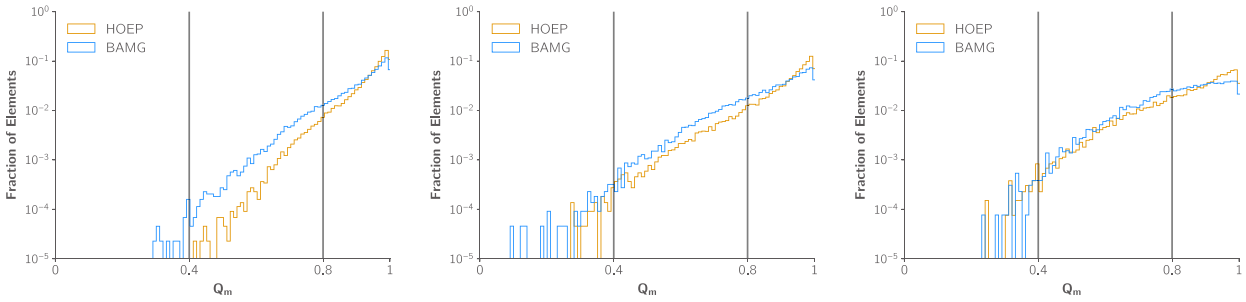


(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 32. Edge length distribution for the MDA 30P-30N case for $\mathcal{C}^{\text{tgt}} = 128{,}000$ final meshes.

edges inside the target range, compared to 91 % for BAMG. This difference in metric conformity likely leads to the relatively large 1 drag count difference between the two meshes.

Examining the element quality for the final meshes at $\mathcal{C}^{\text{tgt}} = 128{,}000$ in Fig. 33 shows that HOEP produces meshes with higher metric quality. HOEP has more elements around $Q_m = 1$ than the BAMG meshes. The BAMG meshes have lower minimum quality and more elements below the 0.4 threshold. This is likely due to differences in the quality metric used in BAMG and HOEP. One major difference is that HOEP optimizes the mesh with this particular quality metric in Eq. 14.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 33. Element quality distribution for the MDA 30P-30N case for $C^{tgt} = 128{,}000$ final meshes.

8. Conclusion

This paper presents a robust and computationally efficient method for mesh adaptation on high-order curved meshes by extending traditional edge split, edge collapse, edge swap, and node movement operations, and compares it to traditional re-meshing and re-curving based adaptation. Mesh adaptation for linear meshes has become quite mature, and there are many methods for producing metric-conforming meshes. However, curved meshes are required for performing simulations with high-order methods to represent the geometry accurately. Currently, the state of the art is to perform mesh adaptation on the linear representation of the curved mesh and then to re-curve the mesh by either solving an optimization problem or an analogous physics-based elasticity problem. Both of these methods are computationally expensive and are not robust as they require valid linear input meshes, which are not always available from high-order meshes, and do not always guarantee a valid curved mesh. Other non-traditional and more robust methods exist but are typically more computationally expensive than the traditional elasticity methods. We present a method to solve this problem by extending existing mesh modification operators used in mesh adaptation to work on curved meshes. By not requiring any additional re-curving step, the mesh adaptation process becomes more robust for a similar computational cost. This method extends the use of existing linear mesh modification operations and combines them with ideas from mesh warping to move high-order nodes efficiently. This process rejects any operation that would create invalid curved elements and keeps a valid mesh that is representative of the actual geometry. While the additional restrictions from remaining valid and curved reduce the number of operators that can proceed, the mesh can always be used to find a primal and adjoint solution if more adaptation is needed.

We present four test cases that show this method's ability to adapt at a variety of Reynolds numbers, which directly impacts the required anisotropy of the mesh and the target number of elements. All of the test cases showed that HOEP has a very high-level of metric conformity similar to that of BAMG. The first test case showed that our method maintains the DG optimal error convergence rates and has error values similar to those of existing metric-based remeshing with re-curving. The following test cases showed that our method can match remeshing with re-curving for drag prediction in more complex cases. Using high-order edge operations showed the ability to generate highly anisotropic meshes with elements with aspect ratio $\mathcal{O}(10^3)$. Additionally, we showed the benefits of using local operations instead of global remeshing to capture shocks without losing accuracy in drag prediction. Finally, the multi-element airfoil case showed the benefit of using our method to generate anisotropic meshes around highly curved boundaries. The improved robustness did not come at any additional computational cost or loss of solution accuracy.

CRediT authorship contribution statement

Alexander W.C. Coppeans : Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization; **Krzysztof J. Fidkowski:** Writing – review & editing, Supervision, Methodology; **Joaquim R.R.A. Martins:** Writing – review & editing, Supervision, Methodology.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The first author acknowledges the support received from the Department of Defense (DoD) through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program and the Rackham Graduate School at the University of Michigan through

the Rackham Merit Fellowship. The first author received partial support from the University of Michigan Institute for Computational Discovery and Engineering Fellowship.

References

[1] K.J. Fidkowski, D.L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA J. 49 (4) (2011) 673–694. https://doi.org/10.2514/1.J050073

[2] M. Yano, J.M. Modisette, D.L. Darmofal, The importance of mesh adaptation for higher-order discretizations of aerodynamics flows, AIAA Paper 2011–3852, 2011. https://doi.org/10.2514/6.2011-3852

[3] M.A. Park, A. Loseille, J.A. Krakos, T.R. Michal, Comparing anisotropic output-based grid adaptation methods by decomposition, AIAA Paper 2015–2292, 2015. https://doi.org/10.2514/6.2015-2292

[4] D. Ibanez, N. Barral, J. Krakos, A. Loseille, T. Michal, M. Park, First benchmark of the unstructured Grid adaptation working group, Procedia Eng. 203 (2017) 154–166. 26th International Meshing Roundtable. https://doi.org/10.1016/j.proeng.2017.09.800

[5] K.J. Fidkowski, A local sampling approach to anisotropic metric-Based mesh optimization, in: AIAA SciTech Forum, 2016–0835, 2016. https://doi.org/10.2514/6.2016-0835

[6] M. Yano, D.L. Darmofal, An optimization-based framework for anisotropic simplex mesh adaptation, J. Comput. Phys. 231 (22) (2012) 7626–7649. https://doi.org/10.1016/j.jcp.2012.06.040

[7] P.J. Capon, P.K. Jimack, On the adaptive finite element solution of partial differential equations using h-r refinement, Technical Report 96.03, University of Leeds, School of Computing, 1996.

[8] D.S. McRae, R-Refinement grid adaptation algorithms and issues, Comput. Methods Appl. Mech. Eng. 2000 (4) (189) 1161–1182. https://doi.org/10.1016/S0045-7825(99)00372-2

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

[10] K.J. Fidkowski, Output-based mesh optimization using metric-conforming node movement, AIAA Paper 2023–2369, 2023. https://doi.org/10.2514/6.2023-2369

[11] V. Dobrev, P. Knupp, T. Kolev, K. Mittal, V. Tomov, Hr-Adaptivity for nonconforming high-order meshes with the target matrix optimization paradigm, Eng. Comput. 38 (4) (2022) 3721–3737. https://doi.org/10.1007/s00366-021-01407-6

[12] L. Rochery, A. Loseille, Fast high-Order mesh correction for metric-Based cavity remeshing and a posteriori curving of P2 tetrahedral meshes, Comput.-Aided Des. 163 (2023) 103575. https://doi.org/10.1016/j.cad.2023.103575

[13] G. Aparicio-Estrems, A. Gargallo-Peiró, X. Roca, Combining high-order metric interpolation and geometry implicitization for curved r-adaption, Comput.-Aided Des. 157 (2023) 103478. https://doi.org/10.1016/j.cad.2023.103478

[14] T. Michal, J. Krakos, Anisotropic Mesh adaptation through edge primitive operations, AIAA Paper 2012–0159, 2012. https://doi.org/10.2514/6.2012-159

[15] M.A. Park, D.L. Darmofal, Parallel Anisotropic Tetrahedral Adaptation, AIAA Paper 2008-917, 2008. https://doi.org/10.2514/6.2008-917

[16] X. Li, M.S. Shephard, M.W. Beall, 3D Anisotropic mesh adaptation by mesh modification, Comput. Methods Appl. Mech. Eng. 194 (48–49) (2005) 4915–4950. https://doi.org/10.1016/j.cma.2004.11.019

[17] A. Loseille, F. Alauzet, V. Menier, Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes, Comput.-Aided Des. 85 (2017) 53–67. https://doi.org/10.1016/j.cad.2016.09.008

[18] T. Coupez, Génération de maillage et adaptation de maillage par optimisation locale, Revue européenne des éléments finis 9 (4) (2000) 403–423. https://doi.org/10.1080/12506559.2000.10511454

[19] P.C. Caplan, R. Haimes, D.L. Darmofal, M.C. Galbraith, Four-Dimensional anisotropic mesh adaptation, Comput.-Aided Des. 129 (2020) 102915. https://doi.org/10.1016/j.cad.2020.102915

[20] R. Feuillet, A. Loseille, D. Marcum, F. Alauzet, Connectivity-change moving mesh methods for high-order meshes: toward closed advancing-layer high-order boundary layer mesh generation, in: 2018 Fluid Dynamics Conference, 2018, p. 4167. https://doi.org/10.2514/6.2018-4167

[21] R. Feuillet, A. Loseille, F. Alauzet, Optimization of P2 meshes and applications, Comput.-Aided Des. 124 (2020) 102846. https://doi.org/10.1016/j.cad.2020.102846

[22] X.-J. Luo, M.S. Shephard, R.M. O'bara, R. Nastasia, M.W. Beall, Automatic p-version mesh generation for curved domains, Eng. Comput. 20 (2004) 273–285. https://doi.org/10.1007/s00366-004-0295-1

[23] A. Shi, P.-O. Persson, Local element operations for curved simplex meshes, Int. J. Numer. Methods Eng. 125 (2) (2024) e7379. https://doi.org/10.1002/nme.7379

[24] F. Bassi, S. Rebay, High–order accurate discontinuous finite element solution of the 2-D euler equations, J. Comput. Phys. 138 (1997) 251–285. https://doi.org/10.1006/jcph.1997.5454

[25] P.-O. Persson, J. Peraire, Curved mesh generation and mesh refinement using lagrangian solid mechanics, in: 47Th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2009-0949, 2009. https://doi.org/10.2514/6.2009-949

[26] R. Abgrall, C. Dobrzynski, A. Froehly, A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems, Int. J. Numer. Methods Fluids 76 (4) (2014) 246–266. https://doi.org/10.1002/fld.3932

[27] T. Toulorge, C. Geuzaine, J.-F. Remacle, J. Lambrechts, Robust untangling of curvilinear meshes, J. Comput. Phys. 254 (2013) 8–26. https://doi.org/10.1016/j.jcp.2013.07.022

[28] S.L. Karman, J.T. Erwin, R.S. Glasby, D. Stefanski, High-order mesh curving using WCN mesh optimization, in: 46Th AIAA Fluid Dynamics Conference, 2016, p. 3178. https://doi.org/10.2514/6.2016-3178

[29] E. Ruiz-Gironés, J. Sarrate, X. Roca, Generation of curved high-order meshes with optimal quality and geometric accuracy, Procedia Eng. 163 (2016) 315–327. https://doi.org/10.1016/j.proeng.2016.11.108

[30] E. Ruiz-Gironés, X. Roca, Automatic penalty and degree continuation for parallel pre-conditioned mesh curving on virtual geometry, Comput.-Aided Des. 146 (2022) 103208. https://doi.org/10.1016/j.cad.2022.103208

[31] L. Rochery, M.C. Galbraith, D.L. Darmofal, S.R. Allmaras, A generalized continuous mesh framework for explicit mesh curving, in: AIAA SCITECH 2024 Forum, 2024-0787, 2024. https://doi.org/10.2514/6.2024-0787

[32] D.P. Sanjaya, A. Rangarajan, C.F. Ollivier Gooch, Error sampling and synthesis for high-Order node movement, in: AIAA SCITECH 2025 Forum, 2025-0780, 2025. https://doi.org/10.2514/6.2025-0780

[33] S.R. Allmaras, F.T. Johnson, P.R. Spalart, Modifications and clarifications for the implementation of the spalart-allmaras turbulence model, big island, hawaii, 2012. http://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf.

[34] M.A. Ceze, K.J. Fidkowski, High-order output-based adaptive simulations of turbulent flow in two dimensions, AIAA J. 54 (9) (2016). https://doi.org/10.2514/1.J054517

[35] W. Reed, T. Hill, Triangular Mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[36] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (3) (2001) 173–261. https://doi.org/10.1023/A:1012873910884

[37] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, *P*-multigrid solution of high–order discontinuous Galerkin discretizations of the compressible navier-Stokes equations, J. Comput. Phys. 207 (2005) 92–113. https://doi.org/10.1016/j.jcp.2005.01.005

[38] P.L. Roe, Approximate riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372. https://doi.org/10.1016/0021-9991(81)90128-5

[39] F. Bassi, S. Rebay, Numerical evaluation of two discontinuous Galerkin methods for the compressible navier-Stokes, equations, Int. J. Numer. Methods Fluids 40 (2002) 197–207. https://doi.org/10.1002/fld.338

[40] M. Ceze, K. Fidkowski, Pseudo-transient continuation, solution update methods, and CFL strategies for DG discretizations of the RANS-SA equations, in: 21St AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2013. https://doi.org/10.2514/6.2013-2686
[41] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869. https://doi.org/10.1137/0907058
[42] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, P-multigrid solution of high-order discontinuous galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 207 (1) (2005) 92–113. https://doi.org/10.1016/j.jcp.2005.01.005
[43] P. Persson, J. Peraire, Newton-GMRES preconditioning for discontinuous galerkin discretizations of the Navier–Stokes equations, SIAM J. Sci. Comput. 30 (6) (2008) 2709–2733. https://doi.org/10.1137/070692108
[44] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, in: A. Iserles (Ed.), Acta Numerica, Cambridge University Press, 2001, pp. 1–102. https://doi.org/10.1017/S0962492901000010
[45] M. Yano, 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.
[46] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, Int. J. Comput. Vis. 66 (1) (2006) 41–66. https://doi.org/10.1007/s11263-005-3222-z
[47] J.R. Shewchuk, Adaptive precision floating-point arithmetic and fast robust geometric predicates, Discrete Comput. Geometry 18 (1997) 305–363. https://doi.org/10.1007/PL00009321
[48] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, J. Comput. Phys. 233 (2013) 359–372. https://doi.org/10.1016/j.jcp.2012.08.051
[49] E. Luke, E. Collins, E. Blades, A fast mesh deformation method using explicit interpolation, J. Comput. Phys. 231 (2) (2012) 586–601. https://doi.org/10.1016/j.jcp.2011.09.021
[50] S. Jakobsson, O. Amoignon, Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization, Comput. Fluids 36 (6) (2007) 1119–1136. https://doi.org/10.1016/j.compfluid.2006.11.002
[51] A. De Boer, M.S. Van der Schoot, H. Bijl, Mesh deformation based on radial basis function interpolation, Comput. Struct. 85 (11–14) (2007) 784–795. https://doi.org/10.1016/j.compstruc.2007.01.013
[52] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 (1) (1995) 389–396. https://doi.org/10.1007/BF02123482
[53] B.E. Frigoletto, Output-Based Error Estimation and Mesh Adaptation for High-Fidelity Fluid-Structure Interaction, Ph.D. thesis, University of Michigan, Ann Arbor, Michigan, 2024.
[54] F.J. Bossen, P.S. Heckbert, A pliant method for anisotropic mesh generation, in: 5Th Intl. Meshing Roundtable, 63, Citeseer, 1996, p. 76.
[55] P.C. Caplan, Four-Dimensional Anisotropic Mesh Adaptation for Spacetime Numerical Simulations, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2019.
[56] H. Borouchaki, P. George, F. Hecht, P. Laug, E. Saltel, Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes, INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.
[57] P. Spalart, S. Allmaras, A One-Equation Turbulence Model for Aerodynamic Flows, in: 30Th Aerospace Sciences Meeting and Exhibit, 1992. https://doi.org/10.2514/6.1992-439