Anisotropic Metric-Based Curved Meshing Using Prismatic Layers

Krzysztof J. Fidkowski*
University of Michigan, Ann Arbor, MI, 48188

We present a strategy for generating curved, high-order, hybrid meshes that consist of a combination of prismatic layers close to the body and unstructured elements in the rest of the domain. Such curved meshes are required for high-order discretizations, but they are difficult to generate and adapt, in particular when anisotropic elements are desired next to curved geometries. To address the problem of possible inversions in this region, the proposed strategy grows prismatic, anisotropic layers of elements close to the geometry, in a robust manner using a metric to dictate the element sizing and starting with a metric-conforming surface mesh. The curvature of the elements attenuates as the layers grow, and the prismatic regions ends once linear faces are possible. A linear unstructured mesh fills the remaining portion of the domain, and it is generated using a metric-based advancing-front algorithm. The strategy can be implemented in an adaptive solution process by using an existing mesh as scaffold for metric evaluation. Results consisting of two-dimensional prismatic layer generation and full online implementation in an adaptive setting are presented. Comparisons with global remeshing show similar performance and improved robustness due to lack of a separate mesh curving step.

I. Introduction

Current techniques used in generating and adapting meshes for high-order discretizations lack robustness when the elements become highly anisotropic, particularly in three dimensions. This is because of the propensity for generating negative mapping Jacobians upon curving elements to conform to the domain boundary, near which the majority of highly-anisotropic elements reside for industry-level problems such as solutions of the Reynolds-averaged Navier-Stokes equations (RANS). One solution is hanging-node refinement of initially structured hexahedral meshes [1, 2], and this has been the approach pursued in our work so far [3–5]. However, the generation of hexahedral meshes around complex geometries is still often user-intensive, e.g. in creating a domain blocking, and inefficient, e.g. when anisotropic elements propagate to the farfield.

At the opposite end of the spectrum is fully-unstructured mesh generation and adaptation, with robust support of anisotropic curved elements. However, given current technology for mesh curving [6–8], which has seen much progress over the last decade, this remains a challenging task in three dimensions [9]. In this work we therefore introduce an approach that bridges the gap between structured meshes and fully-unstructured mesh adaptation. This approach consists of adaptive refinement and regeneration applied to hybrid prismatic-unstructured meshes.

In this paper, Section II presents the hybrid mesh generation approach, with details on the prismatic layer growth and unstructured advancing-front methods. Section III formulates the online problem, which consists of the solution of the compressible Navier-Stokes equations and adjoint-based error estimation and metric optimization. Section IV shows results for prismatic layer mesh generation used alone and together with unstructured mesh generation, for several test cases. Section V concludes with a summary and future directions.

II. Approach

Figure 1 illustrates the mesh generation and adaptation approach. The hybrid mesh is generated in two steps. In the outer, unstructured region, triangles or tetrahedra are generated using a metric-driven mesh optimization approach, specifically mesh optimization through error sampling and synthesis (MOESS) [10, 11]. The geometry for this region is the interface between the prismatic layer and the unstructured region, and this interface is assumed sufficiently far away from the true geometry to avoid curved elements. The elements in this unstructured region could still be anisotropic, e.g. for the resolution of shocks, as linear anisotropic mesh generation is a relatively mature technology, even in three dimensions [12].

*Professor, Department of Aerospace Engineering, AIAA Associate Fellow.

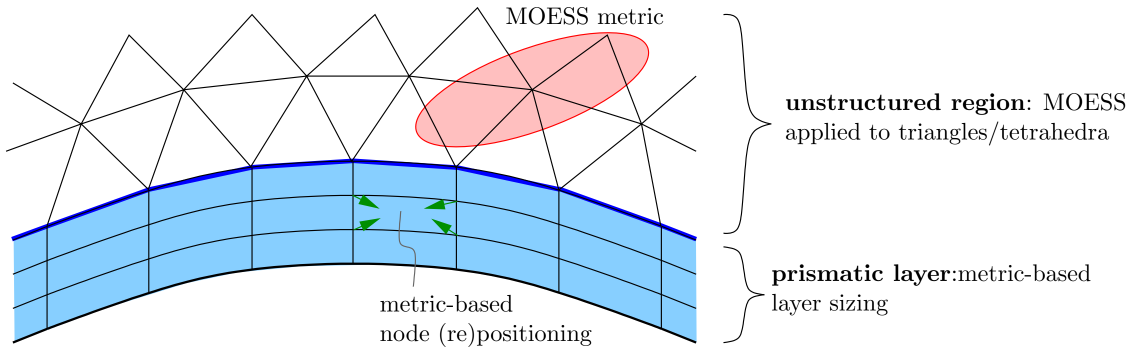

MOESS metric
unstructured region: MOESS applied to triangles/tetrahedra
prismatic layer:metric-based layer sizing
metric-based node (re)positioning

Fig. 1 Metric-driven mesh generation and adaptation for hybrid unstructured/prismatic meshes.

In the inner, prismatic region, the elements are generated by growing layers from a surface mesh. Sizing information for this growth comes from the desired metric field. The structured character of these layers prevents perfect conformity with the metric, although inefficiencies will not be excessive as large changes in the metric are not expected between the start and end of the thin prismatic region. This region ends once linear elements are possible, enabled by attenuation of the curvature through the layers.

For adaptation, meshes in both regions can be regenerated, or nodes can be moved in a metric-conforming manner [13]. The former is expected to be useful in the early stages of adaptation, when mesh changes are large, whereas the latter may be more efficient for small changes later in adaptation. In this work, we consider regeneration only, using prismatic layers and advancing fronts (PLAF).

A. Curved Elements

Curved elements are generally required for high-order discretizations around curved geometries [14]. Generating these elements is complicated by the non-local nature of the problem: curved elements can intrude on the space of adjacent elements, requiring them to be curved as well. For highly-anisotropic meshes, e.g. for high Reynolds-number flows near walls, this can lead to many layers of elements requiring curvature. The difficulty lies in keeping these elements valid, i.e. maintaining positive Jacobians, during the curving. Currently, there is no "native" curved mesh generator that remains robust in the presence of highly anisotropic elements in three dimensions. Instead, curved meshes for such problems are often generated by agglomerating structured linear meshes, at a loss of efficiency and adaptation flexibility, the latter restricted to small node movement or hanging nodes [3].

Once the domain is discretized with a mesh, the geometry of an object is represented by the shape of adjacent elements. When elements are *linear*, i.e. have straight lines for their edges, the geometry representation consists of panels. However, if the actual geometry is not piecewise linear, but instead curved, then the panel representation of the geometry may not be adequate. This occurs for high-order methods, which require more fidelity inside each element, and in such cases, the elements near the geometry need to be curved, as shown in Figure 2.

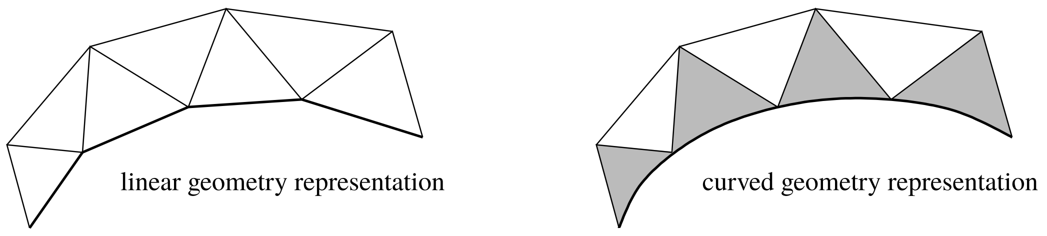
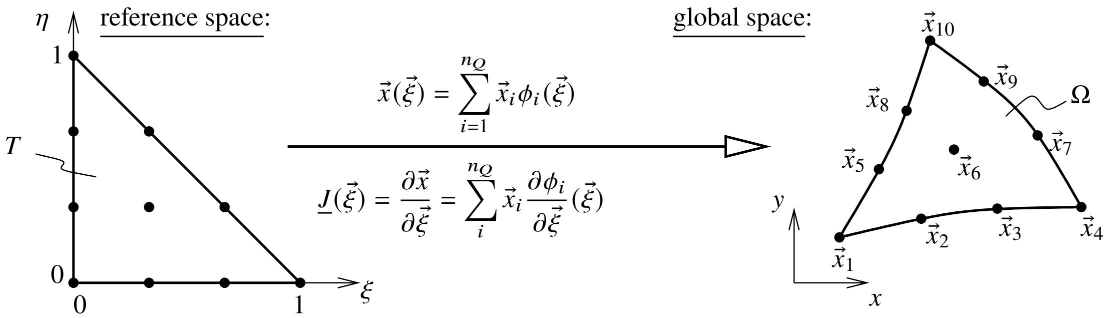

linear geometry representation
curved geometry representation

Fig. 2 Linear versus curved geometry representation.

The underlying problem with a linear geometry in a high-order solver is that the geometric fidelity is inconsistent with the fidelity of the solution approximation. Instead of solving for flow over a smooth surface, we instead solve for flow over a paneled geometry with sharp corners. This not only lowers the accuracy of the solution, negating the benefits of high order solution approximation, but can also cause solution convergence problems.

One way to generate curved elements is to use a nonlinear map from a reference element. We use a high-order polynomial to express the shape of the element in physical space. Let Q be the order of this polynomial. Then, choosing a Lagrange basis to express the map, the task reduces to specifying the global-space coordinates of the high-order geometry nodes. Figure 3 illustrates this mapping.



η
reference space:
1
T
0
0
1
ξ
$\vec{x}(\vec{\xi}) = \sum_{i=1}^{n_Q} \vec{x}_i \phi_i(\vec{\xi})$
$\underline{J}(\vec{\xi}) = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \sum_{i}^{n_Q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{\xi}}(\vec{\xi})$
global space:
$\vec{x}_{10}$
$\vec{x}_9$
Ω
$\vec{x}_8$
$\vec{x}_7$
$\vec{x}_5$
$\vec{x}_6$
y
$\vec{x}_2$
$\vec{x}_3$
$\vec{x}_4$
$\vec{x}_1$
x

Fig. 3 An order Q geometry mapping for producing a curved element in global space.

The Lagrange basis provides a convenient way to represent the geometry because the Lagrange representation interpolates the geometry between specified points. In reference space, the n_Q nodes are equally spaced. For triangles, we have $n_Q = (Q+1)(Q+2)/2$ points, while for quadrilaterals $n_Q = (Q+1)^2$ points. The Lagrange basis functions are denoted by $\phi_i(\vec{\xi})$.

Note that $\vec{\xi} = (\xi, \eta)$ is the reference space coordinate, and $\vec{x} = (x, y)$ is the global coordinate. The mapping Jacobian matrix, $\underline{J}$, is no longer constant for $Q > 1$ elements, and the determinant J can vary inside the element. Ensuring that $J > 0$ everywhere inside the element is a challenging task for general unstructured elements.

In two dimensions, the prismatic layer consists of curved quadrilaterals, in which the Lagrange basis is formed from tensor product functions. In three dimensions, the elements become triangular or rectangular prisms, depending on the surface mesh element type. The Lagrange basis for these prisms is a tensor product of the two-dimensional basis and a one-dimensional Lagrange basis along the normal direction.

B. Two Dimensional Prismatic Layer Generation

In two dimensions, the geometry is specified by clockwise loops of cubic splines, with corners allowed. The surface mesh consists of a high-order paneling of the loops. The location of linear ($Q = 1$) nodes along the spline is determined by equidistribution of metric-based length along the spline. High-order nodes are then added by uniformly subdividing the arc-length distance between adjacent linear nodes. Corner nodes are constrained to be $Q = 1$ nodes. The surface mesh defines the initial front from which the prismatic (i.e. quadrilateral) mesh is generated via an advancing-front procedure.

The first step in advancing the prismatic layers is the calculation of outward-pointing normal vectors at every point, linear and high-order, on the front. These normals can be calculated from the spline geometry or, for simplicity in subsequent layers, by averaging panel normals computed from adjacent nodes, linear and high-order. Corner nodes are identified by the change in the normal between adjacent panels. Three different corner types are possible, as illustrated in Figure 4. Right-turn corners require additional normal vectors, computed by equally dividing the corner angle. No normal vector exists at left-turn corners, at which the two panels form adjacent element sides.

New vertices are added along the normal vectors emanating from each point, linear and high-order, on the front. The layer distance is determined from the specified metric, and high-order vertices are uniformly distributed along this distance. Away from corners, the set of $(Q + 1)^2$ vertices between and including those emanating from adjacent $Q = 1$ nodes defines a high-order element. At right-turn corners, as determined by a threshold angle, additional elements are placed in the space between normals, with adjacent normals forming the two sides of a quadrilateral. The remaining two edges for these elements are obtained by intersecting perpendiculars from the normal vector endpoints, and the remaining high-order nodes are uniformly distributed inside the bilinear quadrilateral. At left-turn corners, intersections of normals define the vertices for the single high-order element.

In a metric-conforming mesh, each edge length has unit measure under the metric. The metric length of an edge e is

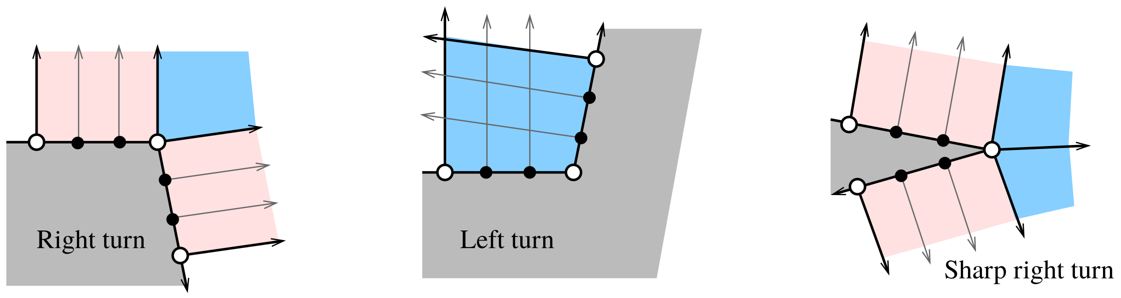


Fig. 4 Corner types in a two-dimensional advancing-front layer. Gray denotes the geometry, pink the standard prismatic elements, and blue the corner-specific elements.

given by

$$L_e^{\mathcal{M}} = \int_e \sqrt{\overrightarrow{dl}^T \mathcal{M} \overrightarrow{dl}},$$

where $\mathcal{M}$ is the metric field. This integral can be done via quadrature or in some cases analytically if the variation of the metric along the edge is known or assumed. If the metric is given on a background mesh, it is interpolated to the edge quadrature points using affine-invariant averaging [15]. In the prismatic layer, the metric is used primarily to measure the distance in the front-normal direction, i.e. along the normals. It is also used to measure the distance between points along the new front, which can be optimized through small changes in the normal direction from each point.

Quadrilateral elements in the layer are formed by grouping $(Q + 1)^2$ vertices. As the layers progress, the curvature of each new front is attenuated by adjusting the position of the high-order nodes on the new front. This is done by first computing the linear-element positions of the high-order nodes along the new front. Displacement vectors are then calculated from the normal-calculated positions to these "target" linear locations. If the magnitude of these displacements is smaller than a threshold, the displacement is applied to the curved vertices along the new front, and it is propagated to the interior nodes using linear interpolation. The threshold is a fraction, $f^{\text{atten}} = 0.2$ of the local layer height. If the displacement exceeds the threshold, it is scaled to have a magnitude equal to the threshold distance. Figure 5 illustrates this procedure.

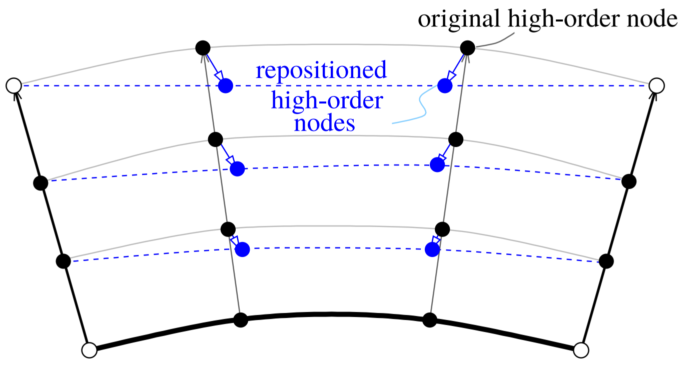


Fig. 5 Attenuation of curvature by node movement towards linear positions.

C. Metric-Based Advancing Front Mesh Generation

Outside of the prismatic region, meshing is performed using unstructured mesh generation/adaptation, driven by the metric on a background mesh. Standard linear meshing techniques can be used here because the curved elements are confined to the prismatic layer. Presently, we use a simple advancing-front strategy to grow the unstructured mesh away from the prismatic layer into the domain.

In the two-dimensional problems considered here, the fronts consist of loops of nodes. For geometrical objects surrounded by prismatic layers, the initial fronts are the outside nodes of the layers. For other boundaries, the initial fronts are formed by distributing points along splines of the boundaries, using metric-based distances.

After initialization, the fronts are grown by placing new points and making connections to form new elements, until the entire domain is filled. Many ways exist to do place points and grow fronts [16], some incorporating metric information [17], and the method chosen here is based on the following criteria:

- The resulting mesh should conform to the metric.
- Placement of new points should be robust, avoiding crossing of fronts.
- The resulting mesh should be efficient, minimizing slivers and short metric-based edge lengths.
- The process should be simple, to minimize errors, and computationally inexpensive.

Motivated by these criteria, we have developed a greedy algorithm that consists of the following steps:

Advancing front algorithm

```
1) Compute outer angles at all nodes on the fronts.
2) Choose the node with the minimum outer angle.
3) Compute directions for any new points based on the size of the outer angle.
4) Add zero or more new points using the metric to compute the distance from the original point.
5) Check for intersections with or proximity to other points; merge fronts if needed.
6) Form elements from new points or connections.
7) Return to step 1 if front nodes exist.
```

Figure 6 illustrates a step in this algorithm. Given a current state of the front, shown as a red line, a node is selected with the minimum outer angle. As shown, this is the minimum geometrical angle, which is appropriate for isotropic elements. The angle is split according to its size, in order to create n_{new} new nodes via the formula

$$n_{\text{new}} = \max\left(0, \lfloor 3\Delta\phi/\pi - 0.5 \rfloor\right),$$

where $\Delta\phi$ is the outer angle. This formula splits $\Delta\phi$ into sub-angles of approximately $\pi/3$ radians (60°). In these directions, new nodes are added at distances determined by the metric, computed at the selected node. Merging of fronts occurs if the new nodes run into an existing front. Note, using the minimum angle on the front to identify the next node promotes convexity of the front, which then reduces occurrences of crossings.

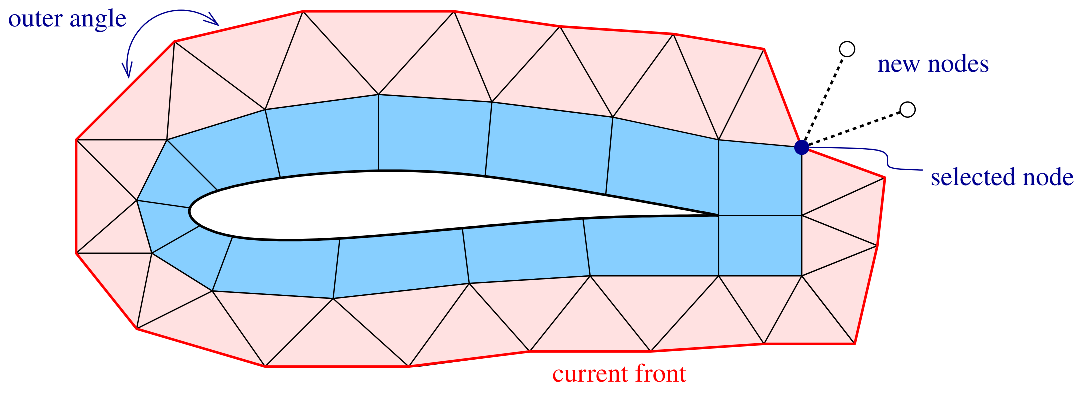


Fig. 6 Advancing-front mesh generation outside of the prismatic layer.

Using standard Euclidian measures to choose and split front outer angles is appropriate for isotropic metrics but becomes inefficient and inaccurate when anisotropy is present. This is because anisotropic elements are characterized by small/large angles, not necessarily near $\pi/3$ radians. In this work, we therefore incorporate the metric into the outer angle selection and subdivision steps. Figure 7 illustrates this process, which consists of a transformation between the

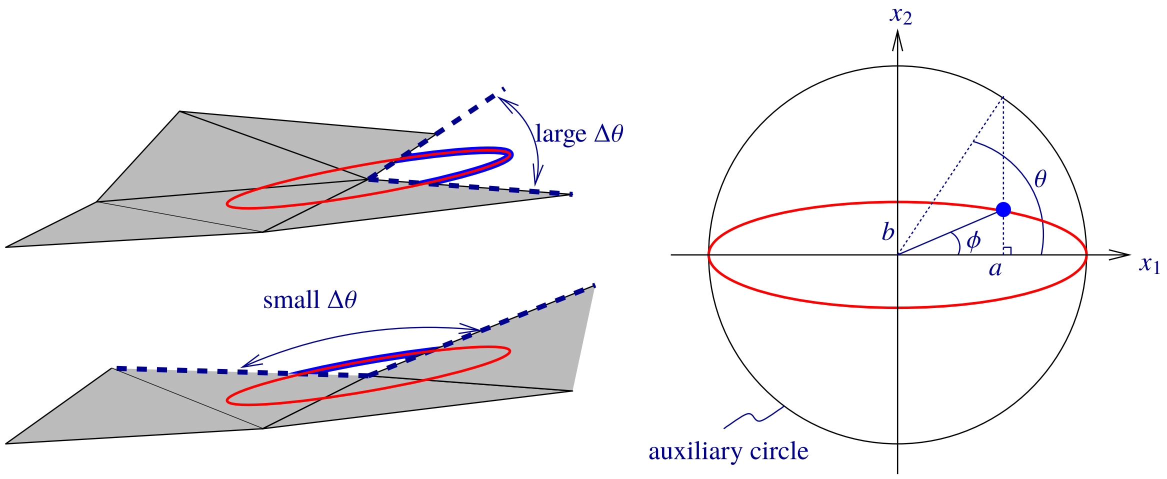


Fig. 7 Metric-based angle definition for advancing-front propagation.

geometrical outer angle, $\Delta\phi$, and an ellipse outer angle, $\Delta\theta$, which approximately represents distance along the ellipse perimeter.

The relationship between ϕ and θ is shown in Figure 7: ϕ measures the angle to a point on the ellipse, whereas θ measures the angle to a point on an auxiliary circle with the same projection onto the semi-major axis of the ellipse. The transformations between these angles are

$$\begin{aligned} x &= a\cos\theta \\ y &= b\sin\theta \end{aligned} \quad \Rightarrow \quad \phi = \tan^{-1}\left(\frac{y}{x}\right) = \tan^{-1}\left(\frac{b}{a}\tan\theta\right) \quad \Rightarrow \quad \theta = \tan^{-1}\left(\frac{a}{b}\tan\phi\right). \tag{1}$$

The eigenvectors and eigenvalues of the metric determine the directions and magnitudes of the major and minor axes. From the geometry of the front, at a particular front node, ϕ measures the angle between the major axis and the ray emanating from the node. This angle is transformed to the ellipse angle, θ, using Eqn. 1. The difference of the angles between the two rays from the front node then yields $\Delta\theta$, which is used to choose and split the minimum angle.

Node movement [13] could be applied after the advancing-front stage, but it is not used in the present work. For comparison, we also consider the relatively standard approach of regenerating the entire mesh in an unstructured linear manner, using the bi-dimensional anisotropic mesh generator (BAMG) [18], and then curving it via linear elasticity.

D. Background Metric Evaluation

In an adaptive setting, the requested metric is given at nodes or elements of the background mesh, which is the mesh from the previous iteration. During the prismatic and advancing-front mesh-generation stages, this metric needs to be evaluated at arbitrary locations in space. To do this, we first determine the element containing the desired point, and the element reference coordinates of the point. To simplify this calculation, the background mesh is snapped to linear, although keeping the mesh curved is also possible and only marginally more expensive when the number of curved elements is small. Linear basis functions are then evaluated at the reference point in the element, and the metric is interpolated from the $Q = 1$ nodes. The interpolation is done using a weighted affine averaging algorithm [15]. To accelerate the search for the containing element, a quad-tree structure is built for the elements of the background mesh. Then, given an arbitrary point, the tree is efficiently traversed starting from the root until a leaf is reached. The few elements with overlap in the leaf are tested until the containing element is found.

E. Three Dimensions

An approach for three dimensions is to first generate a metric-conforming surface mesh of high-order triangles and/or quadrilaterals. As in the two-dimensional case, normal vectors will be defined at each linear and high-order vertex on the surface, and points will be placed along these normals. Corners and edges will be specified by the geometry or obtained by measuring changes in the normal across surface elements. Elements will be added or taken away from the edge/corner cases as needed to keep a valid front. Curvature of the elements will attenuate with increasing layer until linear facets are possible on the entire front, at which point linear unstructured meshing will take over.

III. Problem Formulation

A. Equations and Discretization

We apply the proposed prismatic-layer advancing-front mesh generation technique in two dimensions to discretizations of the compressible Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{2}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the s-component state vector, consisting of the density, momentum per unit volume, and total energy per unit volume. $\vec{\mathbf{F}} \in \mathbb{R}^{\dim \times s}$ is the flux vector, dim is the spatial dimension, and $\mathbf{S}$ is a source term, which is nonzero when modeling turbulence using Reynolds averaging, for which an additional equation is included [19, 20].

To solve Eqn. 2, we use a discontinuous Galerkin (DG) finite-element method [21–23] with the Roe [24] convective flux and the second form of Bassi and Rebay (BR2) [25] for the viscous flux. After choosing order p polynomial basis functions, the steady-state equations take the form $\mathbf{R}(\mathbf{U}) = \mathbf{0}$, where $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, N is the total number of unknowns, and $\mathbf{R}$ is the residual. We solve this system using a Newton-Raphson method with pseudo-time continuation [26] and the generalized minimum residual (GMRES) [27] linear solver, preconditioned by an element-line Jacobi smoother with a coarse-level ($p = 1$) correction [23, 28].

B. Error Estimation and Adaptation

We estimate numerical errors in outputs using an adjoint-weighted residual approach [29, 30]. The adjoint solution for a scalar output is the sensitivity of the output to residual source perturbations. Linearizing the residual and scalar output, $J(\mathbf{U})$, yields a linear system for the adjoint coefficients, $\mathbf{\Psi} \in \mathbb{R}^N$,

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

This equation is solved using a transposed version of the preconditioned-GMRES method used in the Newton-Raphson primal solver. To estimate numerical errors, we denote by H a coarse/current discretization space, and by h a fine one, here obtained by increasing the approximation order, $p \rightarrow p + 1$. If $\mathbf{U}_h^H$ denotes the state prolongated from the coarse to the fine space, the output error can be estimated via

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

where $\delta \mathbf{\Psi}_h$ is the fine-space adjoint with its coarse-space projection removed. The error estimate in Eqn. 4 is localized to elements,

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

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

The elemental error indicator together with additional error indicators evaluated on sub-elements, via Eqn. 5 with projected adjoints, drives a metric optimization calculation based on MOESS: mesh optimization through error sampling and synthesis [11, 31]. This method computes the metric at linear ($Q = 1$) mesh nodes by equidistributing the marginal error-to-cost ratio over the domain, where error is measured by model whose rate coefficients are determined by sampling and least-squares projection, and where the cost model is a simple degree-of-freedom measure. The mesh optimization and flow/adjoint solution are performed several times at either a fixed target cost or with a specified cost growth rate. In this work we use the latter approach and compare the accuracy of outputs after several iterations with an increasing number of degrees of freedom

IV. Results

We have implemented the prismatic layer generation algorithm for two-dimensional geometries. This section presents first a test of the prismatic layer growth stage, followed by online tests of the combined prismatic-layer advancing-front mesh adaptation strategy.

A. Prismatic Layer Growth Demonstration

For testing the prismatic-layer approach, we consider two examples of the meshes obtained when applying this algorithm to an airfoil, the Eppler 387. In both cases, the metric used for the meshes is analytically calculated based on the geometry curvature in the direction along the surface, and a power law based on the wall distance in the front-normal direction,

$$h_\perp = h_0 g^{d/h_0}, \tag{6}$$

where h_0 is the initial layer height, g is the layer growth factor, d is the distance from the wall, and h_0 is the growth distance. For both cases, g is set to 1.2.

We first consider an initial layer height of $h_0 = 0.005c$, where c is the airfoil chord length. The prismatic-growth algorithm is run for 24 layers, resulting in an all-quadrilateral mesh. Figure 8(a) shows this mesh. The quadrilaterals have close to right angles and are of moderate anisotropy near the wall, with a maximum aspect ratio of around 10. After three layers, the elements are no longer curved due to the curvature attenuation, and no negative Jacobians are present in this mesh. Figure 8(b) demonstrates a simple flow solution obtained using a discontinuous Galerkin discretization [5] of the RANS-SA equations [32] at $Re = 60,000$, $M = 0.1$, $\alpha = 2°$. The flow exhibits the correct behavior, and the mesh is suitable for resolving the boundary layer at this relatively low Reynolds number.

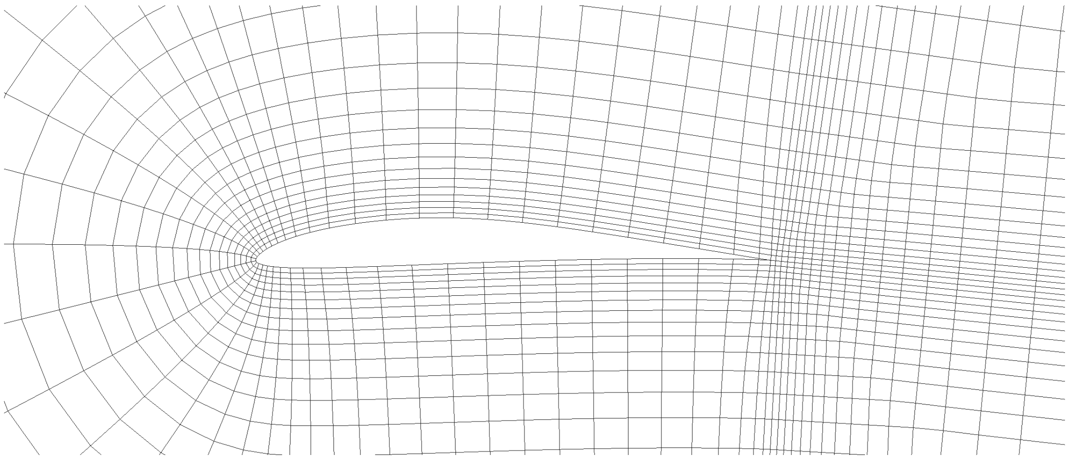
We next consider a more challenging case, in which the initial layer height is $h_0 = 5 \times 10^{-5}c$. Figure 9 shows the mesh with zoom boxes near the leading and trailing edges. The mesh contains 40 layers, and in this case, the elements are highly anisotropic near the wall: the maximum aspect ratio exceeds 1000. No negative Jacobians are present. The curvature attenuation takes longer to reach a front of all linear elements: 12 layers in this case. When coupling with an unstructured outer mesh generator, the front growth would stop at this point, to allow for more efficient conformity to the metric away from the surface.

B. Laminar Flow

This and the following subsections present adaptive results for flows over the NACA 0012 and RAE 2822 airfoils. Figure 10 shows the coarse initial meshes used for these studies. The curved-element geometry order is $Q = 3$. Both meshes exhibit poor resolution in areas generally important for flow approximation, such as boundary layers and trailing-edge singularities. However, these meshes are adapted over multiple iterations to obtain high-quality meshes and accurate outputs.

As a first test case, we consider laminar flow at $Re = 5000$, $M = 0.5$, $\alpha = 2°$ over the NACA 0012 airfoil. Figure 11 shows the Mach number contours for this case. The boundary layer is not overly thin in this low Reynolds-number case, and it separates from the upper surface near the trailing edge. We are interested in the drag coefficient on the airfoil, and Figure 11 also presents the convergence of this output versus degrees of freedom, which increase as adaptation proceeds. Shown are results for two approximation orders, $p = 1$ and $p = 2$, and two meshing strategies: the present prismatic-layer advancing-front (PLAF) method, and full unstructured remeshing using the bi-dimensional anisotropic mesh generator (BAMG) and linear-elasticity mesh curving. We see that after initial erratic outputs, due to the coarseness of the starting mesh, both meshing strategies yield outputs that settle down and converge to a consistent value. For $p = 1$, both PLAF and BAMG yield nearly identical convergence curves after a few iterations. The same occurs for $p = 2$, albeit a bit later in adaptation. In general, $p = 2$ appears to converge more quickly than $p = 1$, and the performance between the mesh generators is similar.

Figure 12 shows the final adapted meshes for all four cases. Curved elements are highlighted in blue. We note that for compatibility with our present implementation of the MOESS algorithm, the prismatic layers are subdivided into triangles after mesh generation. This does not decrease the robustness of the mesh generation because the division is done in reference space of the elements. In PLAF, only one layer of prismatic elements is needed before front becomes linear. The metric-based advancing-front strategy then takes over and produces the rest of the mesh shown. We see qualitative similarity in the areas targeted for refinement, which include the trailing edge, boundary layer regions, and the leading-edge stagnation streamline. $p = 2$ meshes are coarser than $p = 1$ meshes at the same total cost, due to a


(a) Mesh

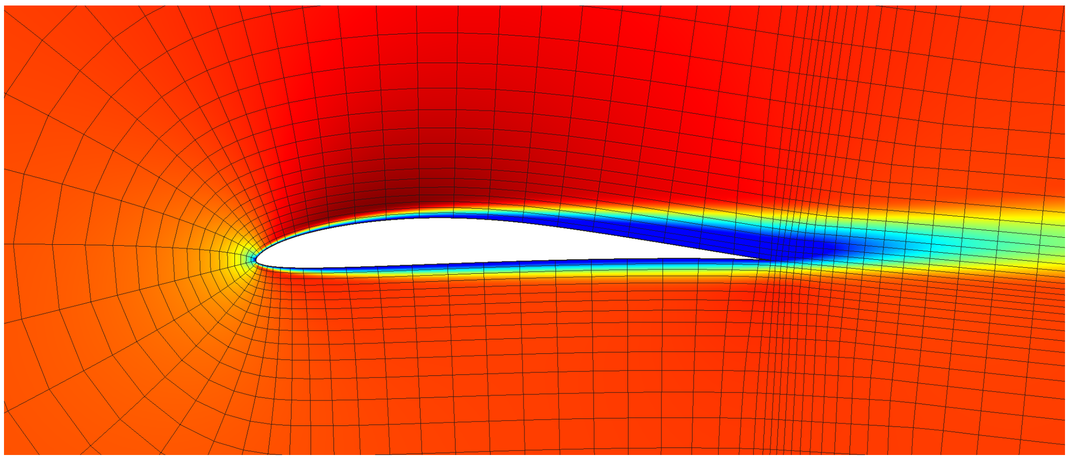
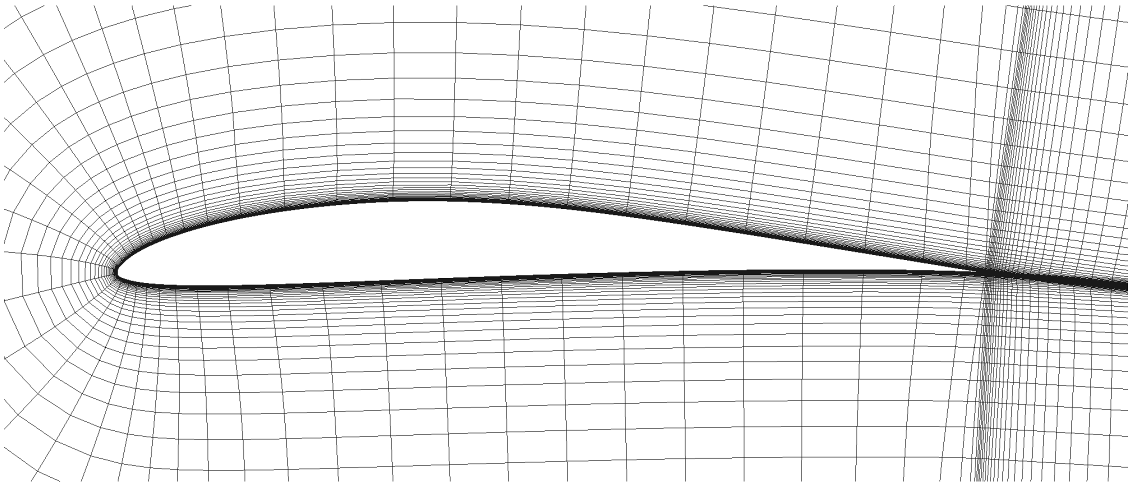
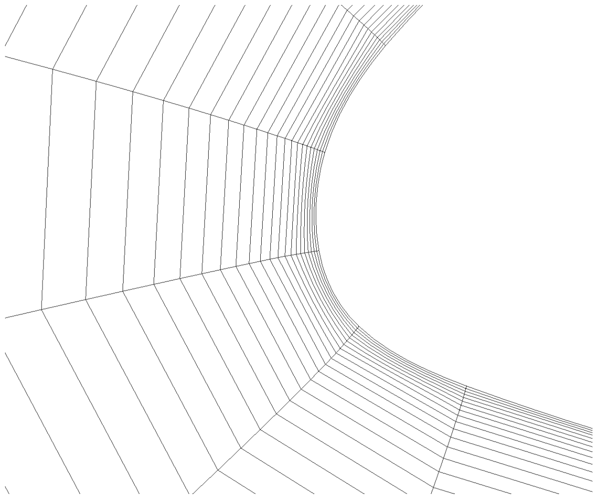

(b) Mach contours (0,0.12)

Fig. 8 Prismatic (quadrilateral) mesh and flow solution for the Eppler 387 airfoil.


(a)


(b) front zoom

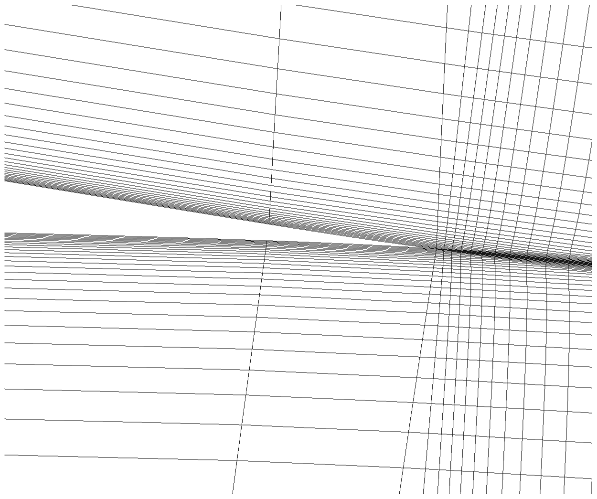
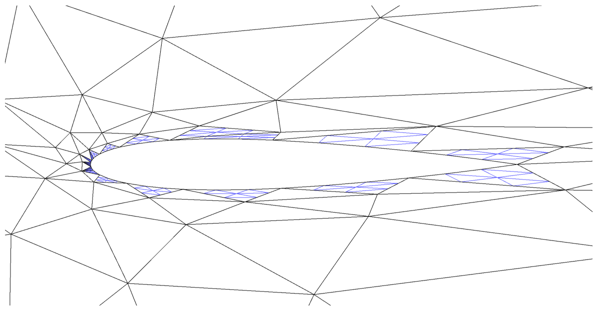

(c) rear zoom

Fig. 9 Highly anisotropic (aspect ratio over 1000) quadrilateral mesh over the Eppler 387 airfoil.


(a) NACA 0012

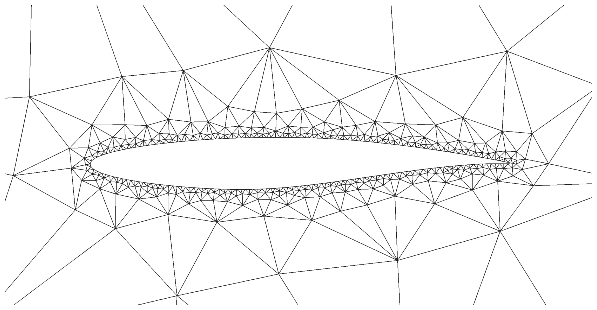
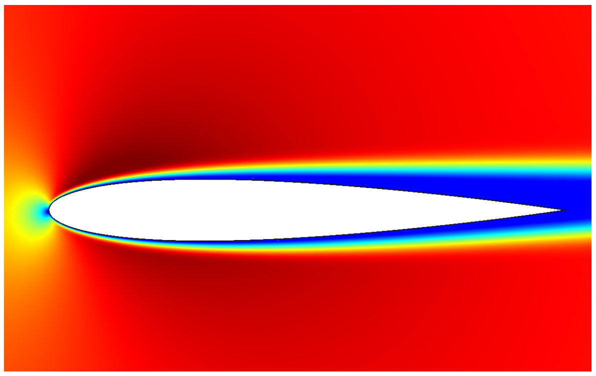
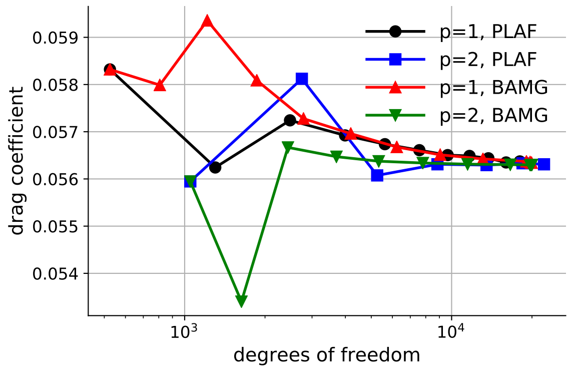

(b) RAE 2822

Fig. 10 Initial meshes for adaptive studies.

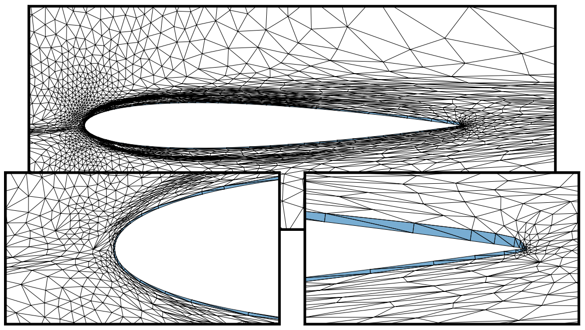

(a) Mach contours (0-0.6)

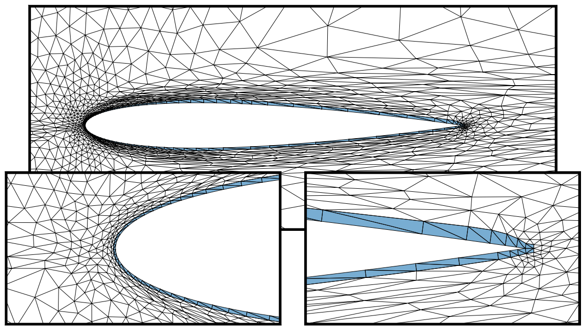


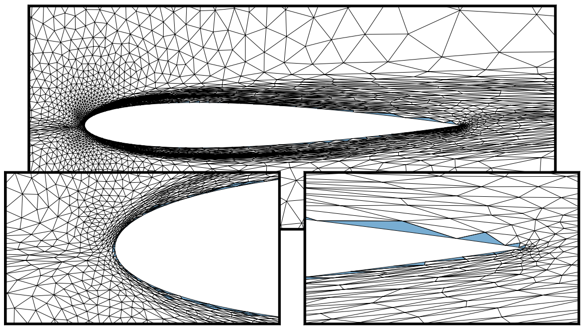
(b) Output convergence

Fig. 11 Flow solution and output convergence for the laminar flow case.

higher degree of freedom count per element for $p = 2$. PLAF has the advantage of not needing a separate mesh curving step, due to its use of the prismatic layers with curvature attenuation.


(a) PLAF, $p = 1$


(b) PLAF, $p = 2$


(c) BAMG, $p = 1$

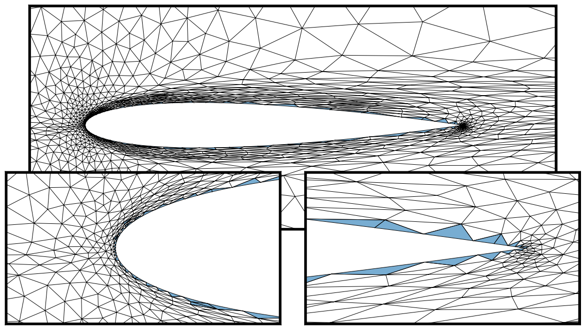
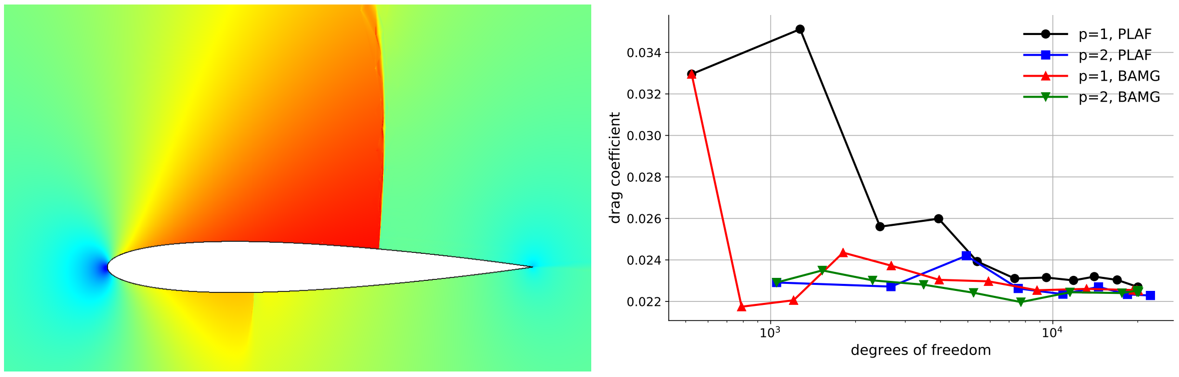

(d) BAMG, $p = 2$

Fig. 12 Adapted meshes for the laminar flow test case.

C. Inviscid Transonic Flow

For the next case, we simulate transonic inviscid flow at $M = 0.8$, $\alpha = 1.25°$ over the NACA 0012 airfoil. Figure 13(a) shows the Mach number contours for this flow, which exhibits a strong normal shock on the upper surface, and a weak one on the lower suface. Shock capturing is performed using element-constant artificial viscosity. No boundary layers are present in this invisicd case, and the important areas consist of the leading/trailing edges and the shocks.

We take the drag coefficient as the output of interest in this flow. Figure 13 shows convergence of this output for $p = 1$ and $p = 2$ approximation orders, for both PLaF and BAMG. We see large variations in the output early in adaptation as the poor-quality initial mesh is refined and flow features are resolved. After several iterations, the outputs appear to converge. $p = 1$ PLAF takes the longest, whereas the other runs are similar in convergence. The presence of the shock in this case, and the required shock-capturing scheme, make for a more challenging test case for high-order, and $p = 2$ does not show a clear improvement over $p = 1$.

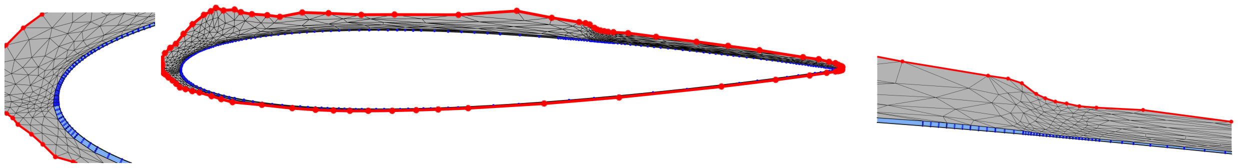


(a) Mach contours, (0-1.6)

(b) Output convergence

Fig. 13 Flow solution and output convergence for the inviscid transonic flow case.

Figure 14 shows an intermediate stage of mesh generation using PLAF, in the advancing-front stage. The active front is highlighted in red, and it shows increased resolution in the important areas: particularly in the shock. The front does not remain strictly convex, due to differences in metric distances when placing new points.



(a) Leading edge

(b) Airfoil

(c) Shock region

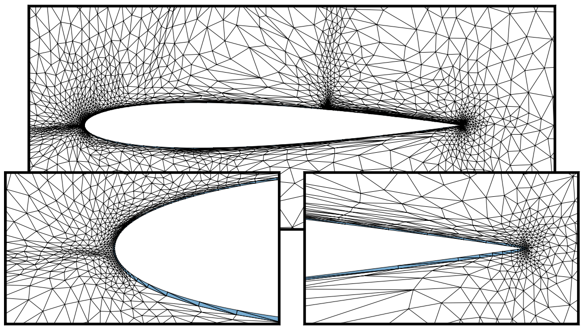
Fig. 14 Advancing-front stage of mesh generation for the inviscid transonic case.

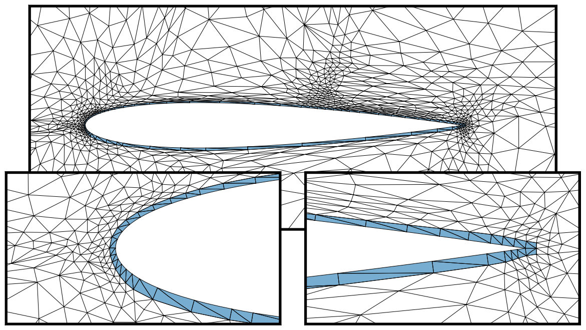
The adapted meshes for this case are shown in Figure 15. Refinement is generally consistent with the flow solution: shocks and leading/trailing edges receive more elements. The λ-shaped adjoint feature in the supersonic region is also targeted. $p = 2$ yields coarser meshes, particularly near the airfoil surface. Differences between the PLAF and BAMG meshes are apparent, e.g. in the resolution of the lower shock ($p = 2$ BAMG targets it, $p = 2$ PLAF does not) and trailing edge, but the resulting meshes are qualitatively similar.

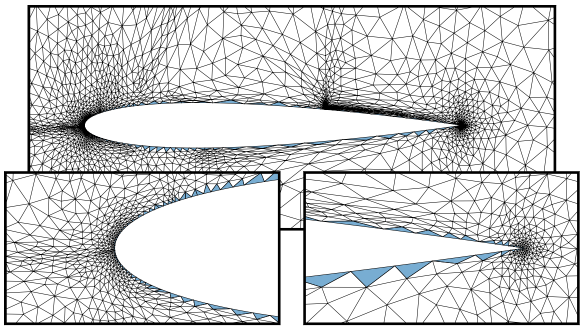
D. Reynolds-averaged Turbulent Transonic Flow

As a final test case, we consider turbulent transonic flow over the RAE 2822 airfoil. The conditions are $M = 0.734$, $Re = 6.5 \times 10^6$, and $\alpha = 2.59°$. Figure 16(a) shows the flow solution, which exhibits a shock on the upper surface and a thin boundary layer that thickens after passing through the shock. This test case requires much finer meshes due to the combination of thin boundary layer and shock. The case is also more challenging for curved mesh generation because of the presence of highly-anisotropic elements near the curved boundary.

We consider the lift coefficient as the output of interest in this flow. Figure 16(b) shows the output convergence, which takes longer in adaptive iterations due to the very coarse initial mesh used for the level of resolution required for accurate output prediction in the flow. All four combinations of $p = 1/p = 2$ and PLAF/BAMG exhibit similar trends for much of the convergence history. Towards the finer meshes, $p = 2$ appears to settle to a final value, at least for PLAF.


(a) PLAF, $p = 1$


(b) PLAF, $p = 2$


(c) BAMG, $p = 1$

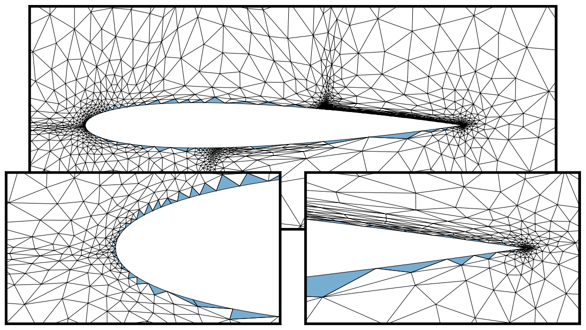
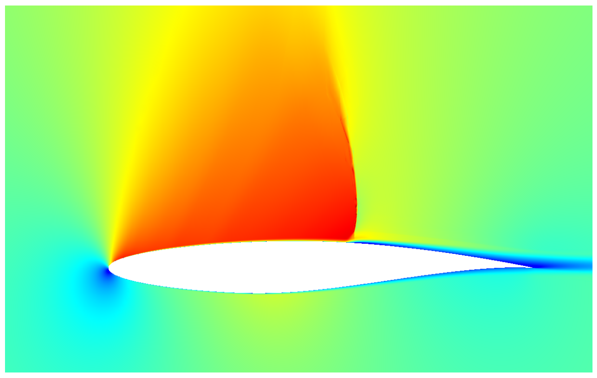
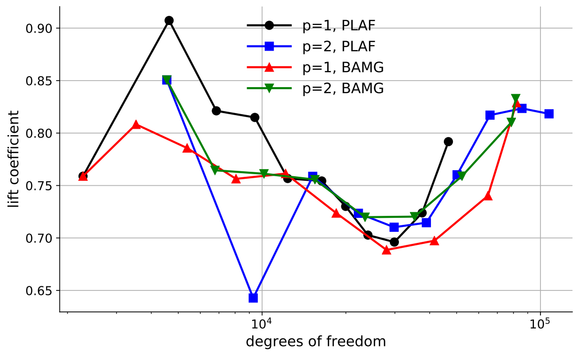

(d) BAMG, $p = 2$

Fig. 15 Adapted meshes for the inviscid transonic flow test case.

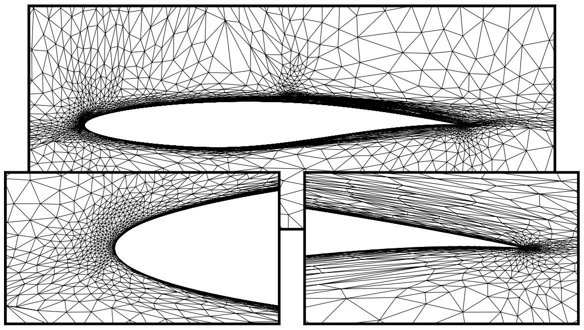

(a) Mach contours (0-1.5)

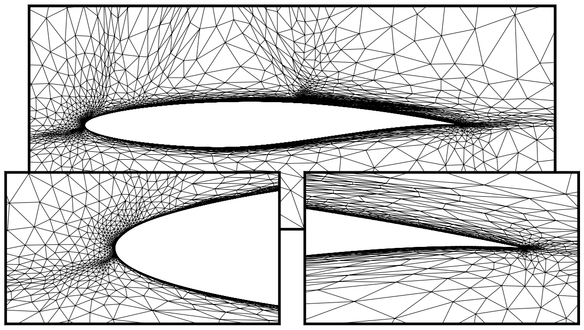


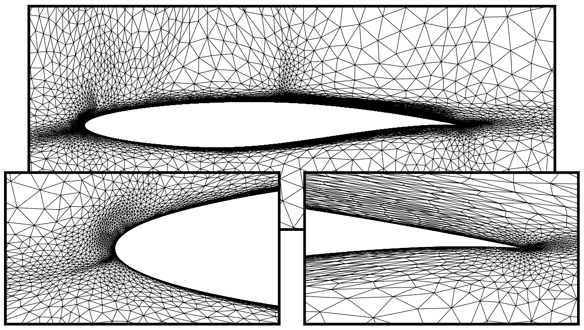
(b) Output convergence

Fig. 16 Flow solution and output convergence for the RANS flow test case.

Figure 17 shows the adapted meshes, which are quite similar for PLAF and BAMG. The boundary layer is quite thin and requires heavy resolution near the surface. This resolution extends further away from the airfoil after passing through the shock, which is not overly resolved, particularly at $p = 2$. The λ adjoint feature is again apparent in all meshes. In this case, elements of high anisotropy are present, reaching over 500 in the boundary layer. Mesh generation using BAMG requires linear elasticity for curving the linear mesh, as a first-layer-only curving approach lacks robustness due to negative Jacobians for highly anisotropic triangles near the wall. Hence, PLAF is less expensive in this case, and in fact, since it uses prismatic layers with a high degree of orthogonality at the surface, only 2-3 layers are required before the front becomes linear.


(a) PLAF, $p = 1$


(b) PLAF, $p = 2$


(c) BAMG, $p = 1$

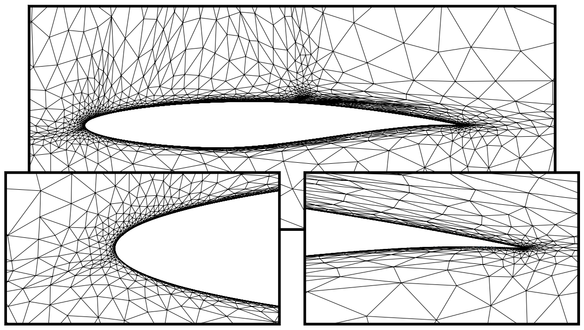

(d) BAMG, $p = 2$

Fig. 17 Adapted meshes for the RANS flow test case.

V. Conclusions

This paper presents the formulation and results of a new method for generating meshes with curved anisotropic elements. Such meshes are required for high-order discretizations of many aerodynamics problems, particularly those at high Reynolds numbers, but they are difficult to generate due to the propensity for element inversions near curved boundaries when the anisotropy is high. The method introduced generates a mesh in two stages: the first consists of a layering of prismatic elements from the curved surface to a distance away from the boundary that enables the use of linear faces; the second is an unstructured meshing of the remainder of the domain using only linear elements. The first stage is robust against inversions due to the orthogonality of the prismatic layers, which grow normally outward from the geometry. The second stage is robust due to the use of linear elements. Both stages use metric information to size the elements, so that they are suitable for mesh adaptation using error estimation and metric optimization. The unstructured stage consists of an advancing-front strategy that uses the metric for angle and distance measurements. Results are shown first for the prismatic growth stage, which attenuates the curved surface information to linear in a small number of prismatic layers, depending on the anisotropy. Next, results for several flow cases, including laminar, transonics, and turbulent, show the meshes and outputs obtained by using the proposed mesh generation method online, in an output-based adaptive setting with metric optimization. Comparisons are made with an existing two-step approach of linear mesh regeneration and curving. These observed convergence histories are similar between the two methods, at least for the orders and cases tested. An advantage of PLAF is that it does not need a separate curving step. In addition, the outer advancing-front mesher is simple and surprisingly robust. Directions of future research include testing/handling multiple objects in close proximity and extension to three dimensions.

References

[1] Leicht, T., and Hartmann, R., "Multitarget Error Estimation and Adaptivity in Aerodynamic Flow Simulations," *International Journal for Numerical Methods in Fluids*, Vol. 56, 2008, pp. 2111–2138.

[2] Leicht, T., and Hartmann, R., "Error estimation and anisotropic mesh refinement for 3D laminar aerodynamic flow simulations," *Journal of Computational Physics*, Vol. 229, 2010, pp. 7344–7360.

[3] Ceze, M. A., and Fidkowski, K. J., "An anisotropic hp-adaptation framework for functional prediction," *AIAA Journal*, Vol. 51, 2013, pp. 492–509. https://doi.org/10.2514/1.J051845.

[4] Ceze, M. A., and Fidkowski, K. J., "Drag Prediction Using Adaptive Discontinuous Finite Elements," AIAA Paper 2013-0051, 2013. https://doi.org/10.2514/6.2013-51.

[5] Fidkowski, K. J., "Three-Dimensional Benchmark RANS Computations Using Discontinuous Finite Elements on Solution-Adapted Meshes," AIAA Paper 2018–1104, 2018. https://doi.org/10.2514/6.2018-1104.

[6] Li, X., Shephard, M. S., and Beall, M. W., "Accounting for curved domains in mesh adaptation," *International Journal for Numerical Methods in Engineering*, Vol. 58, 2003, pp. 247–276.

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

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

[9] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA Contractor Report 218178, 2014.

[10] Yano, M., and Darmofal, D., "An optimization-based framework for anisotropic simplex mesh adaptation," *Journal of Computational Physics*, Vol. 231, No. 22, 2012, p. 7626–7649. https://doi.org/10.1016/j.jcp.2012.06.040.

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

[12] Galbraith, M. C., Caplan, P. C., Carson, H. A., Park, M. A., Balan, A., Anderson, W. K., Michal, T., Krakos, J. A., Kamenetskiy, D. S., Loseille, A., Alauzet, F., Frazza, L., and Barral, N., "Verification of Unstructured Grid Adaptation Components," *AIAA Journal*, Vol. 58, No. 9, 2020, pp. 3947–3962. https://doi.org/10.2514/1.J058783.

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

[14] 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.

[15] 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.

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

[17] Marcum, D., and Alauzet, F., "Aligned Metric-Based Anisotropic Solution Adaptive Mesh Generation," *Procedia Engineering*, Vol. 82, 2014, pp. 428–444.

[18] Borouchaki, H., George, P., Hecht, F., Laug, P., and Saltel, E., "Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes," INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.

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

[20] Ceze, M. A., and Fidkowski, K. J., "High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions," AIAA Paper 2015–1532, 2015. https://doi.org/10.2514/6.2015-1532.

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

[22] 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. https://doi.org/https://doi.org/10.1023/A:1012873910884.

[23] 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. https://doi.org/10.1016/j.jcp.2005.01.005.

[24] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. https://doi.org/https://doi.org/10.1016/0021-9991(81)90128-5.

[25] Bassi, F., and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 197–207. https://doi.org/https://doi.org/10.1002/fld.338.

[26] Ceze, M. A., and Fidkowski, K. J., "Constrained pseudo-transient continuation," *International Journal for Numerical Methods in Engineering*, Vol. 102, 2015, pp. 1683–1703. https://doi.org/10.1002/nme.4858.

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

[28] Persson, P.-O., 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. https://doi.org/https://doi.org/10.1137/070692108.

[29] 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.

[30] 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. https://doi.org/10.2514/1.J050073.

[31] 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.

[32] Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aérospatiale*, , No. 1, 1994, pp. 5–21.