Output-based Adaptive Meshing Using Triangular Cut Cells [*]

Aerospace Computational Design Laboratory Report TR-06-2

Krzysztof J. Fidkowski and David L. Darmofal [†]

October 10, 2006

Abstract

This report presents a mesh adaptation method for higher-order ($p > 1$) discontinuous Galerkin (DG) discretizations of the two-dimensional, compressible Navier-Stokes equations. The method uses a mesh of triangular elements that are not required to conform to the boundary. This triangular, cut-cell approach permits anisotropic adaptation without the difficulty of constructing meshes that conform to potentially complex geometries. A quadrature technique is presented for accurately integrating on general cut cells. In addition, an output-based error estimator and adaptive method are presented, with emphasis on appropriately accounting for high-order solution spaces in optimizing local mesh anisotropy. Accuracy on cut-cell meshes is demonstrated by comparing solutions to those on standard boundary-conforming meshes. Adaptation results show that, for all test cases considered, $p = 2$ and $p = 3$ discretizations meet desired error tolerances using fewer degrees of freedom than $p = 1$. Furthermore, an initial-mesh dependence study demonstrates that, for sufficiently low error tolerances, the final adapted mesh is relatively insensitive to the starting mesh.

[*]An abbreviated version of this report was submitted to the Journal of Computational Physics.

[†]Department Of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 (kfid@mit.edu, darmofal@mit.edu).

Contents

1 Introduction

Computational Fluid Dynamics (CFD) has become an indispensable tool in analysis and design applications. In many applications, CFD can be used to reduce the number of expensive experiments required to achieve a reliable design. In many cases, however, CFD is still plagued by insufficient automation and robustness in the geometry-to-solution process. Mesh generation can be a bottleneck, as meshes around complex geometries are often constructed manually, or with significant user input. Solvers may not converge to a solution, or, if they do, the solution quality cannot be guaranteed. That is, rarely are estimates of the discretization error available, much less an indication of how the error can be decreased. These shortcomings in automation and robustness make CFD a risky tool in analysis, with potentially costly consequences ranging from not obtaining an answer to unknowingly obtaining an answer with large errors. In this report, two ideas are presented for improving automation and robustness in CFD: triangular, cut-cell, mesh generation and output-based, anisotropic adaptation for higher-order discretizations.

First, cut cell meshes offer a potentially more automated and robust alternative to boundary-conforming meshes for complex, curved geometries. In particular, cut cells shift the difficulty from boundary-conforming mesh generation to computational geometry. The Cartesian method [2, 9, 12] is an example of a cut-cell approach in which elements consist of squares/cubes on a regular lattice. It has been effective at automating mesh generation, even for very complex geometries: initial Cartesian meshes can be obtained in seconds without any user involvement. While computationally fast and memory-lean, the Cartesian method has the drawback that it becomes inefficient for the full Navier-Stokes equations. This is because the grid-aligned Cartesian cells are not suited for mesh anisotropy in arbitrary directions, which is required for practical resolution of boundary layer and wake features. Using triangular/tetrahedral cut-cells instead of grid-aligned Cartesian cells relieves this inefficiency by allowing anisotropic adaptation in general directions.

The second idea for improving robustness and automation is an output-based, anisotropic adaptation method. Error estimates for engineering outputs provide the user with a measure of solution quality, thereby preventing use of inaccurate solutions. In addition, the error estimates often come in the form of local error indicators, providing an indication of where mesh refinement is necessary to improve the solution. Coupled with anisotropic mesh adaptation, such error estimates can produce an efficient and automated goal-oriented solution method.

While the combination of triangular cut cells and output-based anisotropic adaptation can be applied to any discretization, the focus of this report is the discontinuous Galerkin (DG) finite element method. A particular advantage of the DG method is that the cut cell implementation does not require changes in the solution representation, which remains in the form of piecewise discontinuous polynomials, or boundary conditions, which are imposed weakly. Rather, the main requirement is the creation of integration rules for arbitrarily-cut elements. The outline for the remainder of this report is as follows. Section 2 presents the triangular cut-cell method. Sections 3 and 4 describe the output-based error estimator and the anisotropic adaptation strategy. Results from sample cases are given in Section 5, focusing on the performance of the cut-cell method in comparison to boundary-conforming meshes in an h-adaptive setting at various interpolation orders p. For completeness, the DG discretization is presented in Appendix A.

2 Cut Cells

The main feature of cut-cell meshes is that the mesh generation process does not conform to the boundary of the geometry. This concept is useful for complex geometries, where generating meshes of boundary-conforming elements is not trivial. The geometry is used to cut elements out of the non-boundary conforming mesh, resulting in irregular cut cells at the boundary. The idea of using cut cells began with the works of Purvis and Burkhalter [29], who used linear cut-cells based on uniform Cartesian meshes for finite volume solutions of the full potential equations. This work was extended to the 2D and 3D Euler equations by Clarke *et al* [11] and Gaffney *et al* [15], respectively. 3D presented a problem of heavy isotropic refinement required for geometries not aligned with the grid. In the late 1980's Boeing's TRANAIR [35] became the first industry code to employ cut cells. A finite element solver for the full potential equations, TRANAIR is still in active use at Boeing. Leveque and Berger [6] presented an adaptive finite volume Cartesian method that used a Godunov method for accounting for wave propagation through more than one cell, thereby relieving the time step restriction caused by small cut cells. Coirier and Powell [12] applied the Cartesian method "as-is" to the 2D Navier-Stokes equations, using a diamond-path reconstruction scheme for the viscous term and isotropic adaptation. They were able to obtain results in 2D but mentioned that isotropic adaptation would become prohibitive in 3D. Karman [20] considered the 3D Reynolds-averaged Navier-Stokes equations in his SPLITFLOW code, which generates a Cartesian cut-cell mesh for most of the domain, but requires a prescribed anisotropic, prismatic, boundary-layer mesh.

Aftosmis *et al* developed a 3D Cartesian solver package, Cart3d [2], which emphasizes fast and fully automated mesh generation using surface geometry triangulation intersections. Cart3d is currently in use for large scale computations, including space shuttle ascent debris simulations [23]. Ongoing work continues in computing adjoints and shape sensitivities [24] and in novel ideas for moving beyond Euler calculations [1].

This report explores the feasibility of using triangular cut-cell meshes in a discontinuous Galerkin finite element framework. For high-order approximations, the challenge of using cut cells reduces to accurate integration on cells cut by curved boundaries. The following sections describe one solution to this challenge.

2.1 Geometry Definition and Initial Mesh

For this proof-of-concept study in 2D, a geometry definition consisting of cubic-splined points is used. The data structure used allows for corners and for multiple splines. The computational domain is bounded by a set of points comprising the farfield boundary. A common farfield boundary is a square box around the embedded object(s). An initial mesh consists of a coarse uniform triangulation of the farfield-bounded domain, without regard to the embedded objects. If desired, subsequent geometry-adapted triangulations are constructed by refining elements that

- contain a spline endpoint or a spline corner,
- are intersected more than once by one or more splines,
- contain a spline segment for which the normal changes in angle by more than $\Delta\theta_{\max}$ (default is 20^o).

The details of geometry adaptation are not crucial, as only a reasonable starting mesh is sought for the solution-adaptive method.

2.2 Cutting Algorithm

Given an area-filling mesh of the computational domain, and a set of splines defining the geometry, a cutting algorithm is employed to determine which elements are cut by the splines and the precise geometry of the cuts. The cutting algorithm proceeds by solving cubic intersection problems (Appendix E) to determine intersections of spline segments with element edges. Careful attention must be given to conditioning for node or tangency intersections.

Each spline-element intersection is labeled as an "embedded face," and is identified by the two spline arc-length parameters that mark the start and end of the intersection. Elements and edges completely outside the computational domain are marked as "null." They are identified by flagging nodes as inside/outside the domain, starting at the cut elements. No flux calculation is performed on null edges.

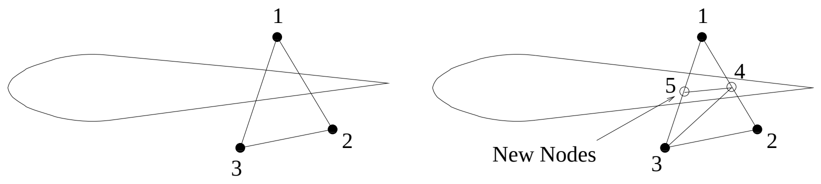


Figure 1: Mesh improvement for multiply-cut edges (edge 12 and 13) via edge splitting and element refinement.

During the development of the adaptive, triangular cut-cell algorithm, a difficulty was identified related to elements with edges cut more than once by embedded splines as shown in Figure 1. Specifically, a solution on such a mesh will often result in large errors in these cells since the cells are attempting to represent possibly very different flow fields with a single polynomial basis. The subsequent adaptation will tend to refine the mesh at multiply-cut edges. On the refined mesh, the errors often drop substantially, resulting in a request for larger elements on the next adaptation, and again producing multiply-cut cells. The result is an oscillatory behavior of the adaptive method. To eliminate this behavior, multiply-cut edges are split prior to solution, as indicated in Figure 1.

2.3 Integration

A high-order DG method requires integration over element interiors as well as boundary and interior edges. One-dimensional integration on interior cut edges and embedded faces is performed by mapping each segment to a reference interval and using numerical quadrature. A caveat for embedded faces is that each spline segment is mapped to a reference interval separately, so as to avoid integration through second-order geometric discontinuities at the spline knots.

Integration on cut cell interiors is not as straightforward, but it is still tractable. One approach, used in this work, is outlined below. The goal of this method is to produce for each cut cell a set of integration points, $\mathbf{x}_q$, and weights, w_q, to integrate arbitrary $f(\mathbf{x})$,

$$\int_\kappa f(\mathbf{x}) d\mathbf{x} \approx \sum_q w_q f(\mathbf{x}_q).$$

The key idea is to project $f(\mathbf{x})$ onto a space of high-order basis functions, $\zeta_\mathbf{i}(\mathbf{x})$. The basis functions $\zeta_\mathbf{i}(\mathbf{x})$ are chosen to allow for simple computation of the integral $\int_\kappa \zeta_\mathbf{i}(\mathbf{x}) d\mathbf{x}$. In particular, choosing $\zeta_\mathbf{i} \equiv \partial_k G_{\mathbf{i}k}$ leads, by the divergence theorem, to

$$\int_\kappa \zeta_\mathbf{i} d\mathbf{x} = \int_\kappa \partial_k G_{\mathbf{i}k} d\mathbf{x} = \int_{\partial\kappa} G_{\mathbf{i}k} n_k ds,$$

where n_k is the outward-pointing normal. The integrals over the element boundary, $\partial\kappa$, are computed using the interior edge and embedded face quadrature rules. $G_{\mathbf{i}k}$ is chosen as

$$G_{\mathbf{i}k} = x_k \Phi_\mathbf{i}(\mathbf{x}), \quad \Phi_\mathbf{i}(\mathbf{x}) = \prod_k \phi_{i_k}(x_k), \quad \mathbf{x} = [x_k], \quad \mathbf{i} = [i_k],$$

where $k \in [0, .., d-1]$ and d is the spatial dimension. The functions $\phi_i(x)$ are well-conditioned one-dimensional basis functions. In this work, Lagrange basis functions are used, with Gauss point nodes on the element bounding box intervals (except for certain ill-conditioned cases, as described at the end of this section). The order of these functions is the desired order of integration for $f(x)$. This order depends on the equation set and on the solution interpolation order, p. The same order is used for cut cells as for standard element-interior quadrature rules [13].

The factors of x_k in the definition of $G_\mathbf{i}$ ensure that the $\zeta_\mathbf{i} = \partial_k G_{\mathbf{i}k} = d\Phi_\mathbf{i}(\mathbf{x}) + x_k \partial_k \Phi_\mathbf{i}(\mathbf{x})$ span the same complete space as the tensor product functions $\Phi_\mathbf{i}$. This statement can be proved as follows. Let

$$\mathbf{x}^\mathbf{j} \equiv \prod_k x_k^{j_k}, \quad k \in [0, .., d-1],$$

be the standard monomial basis indexed by $\mathbf{j} = [j_k]$. Since the $\Phi_\mathbf{i}(\mathbf{x})$ are assumed to form a complete basis, every monomial can be written as a linear combination of the $\Phi_\mathbf{i}$,

$$\mathbf{x}^\mathbf{j} \quad = \quad a_{\mathbf{j}\mathbf{i}} \Phi_\mathbf{i}(\mathbf{x}).$$

Taking the gradient of both sides, dotting with x_k, and adding $d\,\mathbf{x}^{\mathbf{j}}$ yields

$$\left(\sum_k j_k + d\right)\mathbf{x}^{\mathbf{j}} \quad = \quad a_{\mathbf{ji}}\left(d\Phi_{\mathbf{i}}(\mathbf{x}) + x_k\partial_k\Phi_{\mathbf{i}}(\mathbf{x})\right)$$

Since $d > 0$ and the j_k are non-negative, one can divide both sides by $(\sum_k j_k + d)$ to arrive at a representation of each monomial by the $\zeta_{\mathbf{i}}$ functions. Thus, the $\zeta_{\mathbf{i}}$ span the same complete space as the $\Phi_{\mathbf{i}}$.

The projection of $f(\mathbf{x})$ onto $\zeta_{\mathbf{i}}(\mathbf{x})$ is performed by minimizing the least-squares error,

$$E^2 = \sum_q \left[\sum_{\mathbf{i}} F_{\mathbf{i}}\zeta_{\mathbf{i}}(\mathbf{x}_q) - f(\mathbf{x}_q)\right]^2 .$$

Specifically, the solution vector, $F_{\mathbf{i}}$, is found using QR factorization of the matrix $\zeta_{\mathbf{i}}(\mathbf{x}_q)$, leading to the following expression for the quadrature weights,

$$w_q = (R_{\mathbf{ji}})^{-1}Q_{\mathbf{j}q}\int_\kappa \zeta_{\mathbf{i}}(\mathbf{x})d\mathbf{x}, \quad \text{where} \quad \zeta_{\mathbf{i}}(\mathbf{x}_q) = Q_{q\mathbf{j}}R_{\mathbf{ji}}.$$

The choice of sampling points, $\mathbf{x}_q$, affects the conditioning of the QR factorization of $\zeta_{\mathbf{i}}(\mathbf{x}_q)$. The points should lie inside the cut cell, so that the integrand remains physical. Multiple methods exist for choosing these interior points. In this work, two methods were used. The first consists of picking points randomly in the element bounding box and checking whether or not they lie inside the element. This inside/outside check is performed by casting a segment to a point that is known to be inside or outside the element and counting the number of intersections with the 1D element boundaries. This method is sufficient for elements that are close to isotropic, and it provides a truly random point selection. However, it becomes inefficient for elements whose area is small relative to the area of the bounding box. For such elements, the random points are chosen by casting interior-bound rays from quadrature points on the 1D element boundary. These rays are directed along the normal direction with a random variation (default range is $\pm 15^{\mathrm{o}}$). The closest intersection of each ray with an element boundary marks where the ray first exits the element. A random interior point is chosen between the origin of the ray and this exit point.

Since a random set of points may possess unfavorable clusters, conditioning of the QR factorization generally improves with an increasing number of sampling points, n_q. In this work, n_q is set to four times the number of $\zeta_{\mathbf{i}}$ basis functions. In the event of a singular error in the QR factorization, another set of sampling points is chosen. In addition, using an axis-aligned element bounding box to define the $\Phi_{\mathbf{i}}(\mathbf{x})$ may pose conditioning problems for certain elements. The problematic elements include non-axis-aligned sliver elements as well as multiply-cut elements with small, disjoint regions, as shown in Figure 2. In these

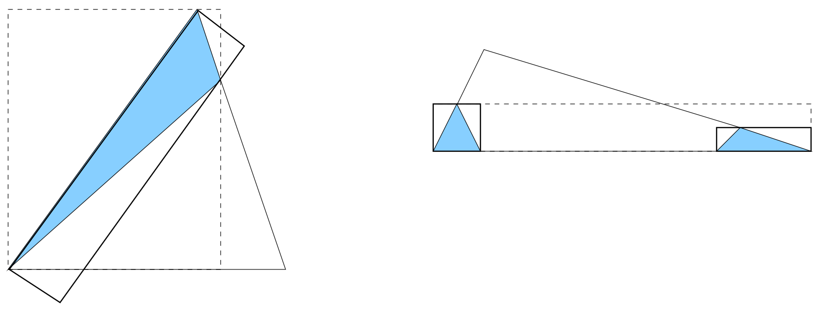


Figure 2: Cut elements requiring bounding-box adjustment. For non-axis-aligned sliver cut elements (left), the original bounding box (dashed line) is rotated to obtain a tighter fit (solid line). For an element cut into disjoint regions (right), separate bounding boxes are used for each region.

cases, conditioning is improved by adjusting the bounding box. As shown in Figure 2, this improvement can come from rotation for a tighter fit around the element, or from using a separate bounding box for each disjoint region. Finally, while better algorithms likely exist for performing the cut-cell integrations, the one presented here has been sufficient for the problems considered thus far.

2.4 Implementation

The cut cell method was implemented in an existing DG code, with minimal impact on the solver and other parts of the code. No change was made to the basic numerical integration paradigm for residual evaluation, as the cut-cell integrations take the form of quadrature sums. The cut-cell integration rules are created once in a pre-processing step, and saved, before beginning solution iteration. Interpolation functions for cut cells are defined not on the original triangles, but rather on "shadow" triangles taken to be the right triangles associated with the cut-cell bounding boxes. This choice improves conditioning of the basis for small cut cells. Even though this work deals with steady-state solution via implicit schemes, an unsteady term with local time steps is used to improve robustness in the early stages of convergence. The local time step is chosen using a global CFL number: $\Delta t_\kappa = CFL(h_\kappa / s_{\kappa,\max})$, where h_κ and $s_{\kappa,\max}$ are the element-specific size and maximum wave speed, respectively.

3 Output-Based Error Estimation

Accurate prediction of an output (e.g. drag, or lift on an airfoil) may depend on resolution of seemingly un-interesting areas. This is especially the case in hyperbolic problems, where small variations in one location can have large effects on the solution behavior downstream. Error estimators based on local criteria often fail to capture the error due to such propagation effects. Output-based error estimators address this problem by linking local residuals to outputs through the use of the adjoint solution. Moreover, adapting on several key outputs often produces a multi-purpose solution that can be used to accurately predict other outputs in post-processing.

Output-based error estimation and adaption for CFD have been studied extensively in the literature [3, 5, 16, 18, 21, 22, 28, 34]. In the following analysis, an output error estimate for a generic weighted residual statement is derived, motivated by the previously cited work. This estimate is then applied to the DG weighted residual statement.

Let $\mathbf{u} \in \mathcal{V}$ be an analytic solution to a set of nonlinear equations given by $F(\mathbf{u}) = 0$. Also, let $\mathbf{u}_H \in \mathcal{V}_H$ be the finite element solution to the corresponding weighted residual statement $\mathcal{R}_H\big(\mathbf{u}_H, \mathbf{v}_H\big) = 0$, $\forall \mathbf{v}_H \in \mathcal{V}_H$, where $\mathcal{R}_H : \mathcal{W}_H \times \mathcal{W}_H \to \mathbb{R}$ is a semi-linear form, linear in the second argument. The space $\mathcal{W}_H \equiv \mathcal{V}_H + \mathcal{V}$ is defined because $\mathcal{V}_H$ is not required to be a subspace of $\mathcal{V}$; in particular, this is the case with DG approximation.

Let $\mathcal{J}(\mathbf{u})$ be a possibly-nonlinear output of interest. The dual problem reads: find $\boldsymbol{\psi} \in \mathcal{V}$ such that,

$$\bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \boldsymbol{\psi}) = \bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}), \quad \forall \mathbf{v} \in \mathcal{V},$$

where the mean value linearizations $\bar{\mathcal{R}}_H : \mathcal{W}_H \times \mathcal{W}_H \to \mathbb{R}$ and $\bar{\mathcal{J}} : \mathcal{W}_H \to \mathbb{R}$ are given by:

$$\begin{aligned}
\bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \mathbf{w}) &= \int_0^1 \mathcal{R}'_H[\theta \mathbf{u} + (1-\theta)\mathbf{u}_H](\mathbf{v}, \mathbf{w}) d\theta, \\
\bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}) &= \int_0^1 \mathcal{J}'[\theta \mathbf{u} + (1-\theta)\mathbf{u}_H](\mathbf{v}) d\theta,
\end{aligned}$$

In the above, the primed notation denotes the Frechét derivative, with linearization performed about the state within the square brackets. Overbar notation denotes a linearization involving the arguments before the semicolon. For $\mathbf{v} = \mathbf{u} - \mathbf{u}_H$,

$$\begin{aligned}\bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \mathbf{w}) &= \mathcal{R}_H(\mathbf{u}, \mathbf{w}) - \mathcal{R}_H(\mathbf{u}_H, \mathbf{w}),\\ \bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H) &= \mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H).\end{aligned}$$

$\boldsymbol{\psi}$ is assumed to exist in the same space as $\mathbf{u}$. Assuming $\mathcal{R}_H(\mathbf{u}, \mathbf{v}) = 0$, $\forall \mathbf{v} \in \mathcal{W}_H$, the output error can be expressed as

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) = -\mathcal{R}_H(\mathbf{u}_H, \boldsymbol{\psi} - \boldsymbol{\psi}_H), \tag{1}$$

where $\boldsymbol{\psi}_H \in \mathcal{V}_H$ can be arbitrary at this point. Defining an adjoint residual,

$$\bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \mathbf{w}) \equiv \bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \mathbf{w}) - \bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}), \qquad \mathbf{v}, \mathbf{w} \in \mathcal{W}_H,$$

the output error can also be expressed as

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) = -\bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \boldsymbol{\psi}_H). \tag{2}$$

As $\mathbf{u}$ and $\boldsymbol{\psi}$ are in general not known, two approximations are employed. First, the exact mean-value linearizations are replaced by approximate linearizations about $\mathbf{u}_H$. To minimize errors in (1) and (2) due to no longer using mean-value linearizations, $\boldsymbol{\psi}_H$ is set to the finite element approximation of $\boldsymbol{\psi}$. That is, $\boldsymbol{\psi}_H$ satisfies $\bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}_H, \boldsymbol{\psi}_H) = 0$, $\forall \mathbf{v}_H \in \mathcal{V}_H$. Second, the exact solutions $\mathbf{u}$ and $\boldsymbol{\psi}$ are estimated by creating approximations to $\mathbf{u}$ and $\boldsymbol{\psi}$ on an enriched finite element space, $\mathcal{V}_h$, consisting of a uniformly refined mesh, $h = H/2$, and order $p+1$ interpolation. The approximations $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$ are created by first reconstructing $\mathbf{u}_H$ and $\boldsymbol{\psi}_H$ on $\mathcal{V}_h$. In this work, local H_1 patch reconstruction is used, in which the minimized error on each element $\kappa \in T_h$ takes the form

$$E_\kappa^2(\mathbf{v}_\kappa, \mathbf{u}_H) = \sum_{l \in \mathcal{P}_\kappa} \Big(\int_l (\mathbf{v}_\kappa - \mathbf{u}_H)^2 d\mathbf{x} + \sum_{i=0}^{d-1} c_i \int_l (\partial_i \mathbf{v}_\kappa - \partial_i \mathbf{u}_H)^2 d\mathbf{x} \Big),$$

where $\mathcal{P}_\kappa$ is the patch of neighboring elements in T_h (including κ), $\mathbf{v}_\kappa \in P^{p+1}(\mathcal{P}_\kappa)$ denotes the order $p+1$ reconstructed solution on the patch, d is the dimension, and the c_i are $O(\Delta x_i)$ scaling coefficients specific to each element, determined by the dimensions of the elemental bounding boxes. The reconstruction is performed twice so that the center subelement of each original element receives new information. Specifically, an intermediate reconstruction,

$\mathbf{u}_h^*$, is defined on each element by $\mathbf{u}_h^*|_\kappa = \mathbf{v}_\kappa^*$, where $\mathbf{v}_\kappa^*$ minimizes $E_\kappa^2(\mathbf{v}_\kappa^*, \mathbf{u}_H)$. The final reconstructed solution, $\mathbf{u}_h$ is then set according to $\mathbf{u}_h|_\kappa = \mathbf{v}_\kappa$, where, for each element, $\mathbf{v}_\kappa$ minimizes $E_\kappa^2(\mathbf{v}_\kappa, \mathbf{u}_h^*)$. $\boldsymbol{\psi}_h$ is obtained analogously. To further improve the approximation, element-Jacobi smoothing is performed on $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$.

Using $\mathbf{u_h}$ and $\boldsymbol{\psi_h}$ in place of $\mathbf{u}$ and $\boldsymbol{\psi}$ in (1) and (2) yields the following approximations to the output error (making use of $\mathcal{V}_H \subset \mathcal{V}_h$):

$$\begin{aligned}
\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) &\approx -\mathcal{R}_h(\mathbf{u}_H, \boldsymbol{\psi}_h - \boldsymbol{\psi}_H) \\
&= -\sum_{\kappa \in T_H} \sum_{l \in \kappa} \mathcal{R}_h(\mathbf{u}_H, (\boldsymbol{\psi}_h - \boldsymbol{\psi}_H)|_l), \\
\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) &\approx -\bar{\mathcal{R}}_h^\psi(\mathbf{u}_H; \mathbf{u}_h - \mathbf{u}_H, \boldsymbol{\psi}_H) \\
&= -\sum_{\kappa \in T_H} \sum_{l \in \kappa} \bar{\mathcal{R}}_h^\psi(\mathbf{u}_H; (\mathbf{u}_h - \mathbf{u}_H)|_l, \boldsymbol{\psi}_H).
\end{aligned}$$

In the above expressions, $l \in T_h$ refers to the subelements of each κ in the uniform refinement, and $|_l$ refers to restriction to element l. A local error indicator on each element is obtained by summing the sub-elemental contributions to the output error in the expressions above. Specifically, in this work, the error indicator in each element κ is taken to be

$$\epsilon_\kappa = \frac{1}{2} \sum_{l \in \kappa} \Big(\big|\mathcal{R}_h(\mathbf{u}_H, (\boldsymbol{\psi}_h - \boldsymbol{\psi}_H)|_l)\big| + \big|\bar{\mathcal{R}}_h^\psi(\mathbf{u}_H; (\mathbf{u}_h - \mathbf{u}_H)|_l, \boldsymbol{\psi}_H)\big| \Big). \tag{3}$$

For systems of equations, indicators are computed separately for each equation and summed together. The global output error estimate, $\epsilon = \sum_\kappa \epsilon_\kappa$, is not a bound on the actual error in the output, due to the approximations made in the derivation. However, the validity of the approximations is expected to increase as $\mathbf{u}_H \to \mathbf{u}$. In the literature, various other indicators are presented, using either/both the primal-based and dual-based error estimate expressions [5, 33]. Using a combination of both expressions targets errors in both the primal and the dual solutions, and has been found sufficiently effective in driving adaptation.

4 Adaptation Strategy

Given a localized error estimate, an adaptive method modifies the computational mesh in an attempt to decrease and equidistribute the error. In high-order finite element methods, possible adaptation strategies include p, h, and hp, where p-adaptation refers to changing only the order of interpolation, h-adaptation refers to changing only the computational mesh, and hp-adaptation is a combination of both.

An advantage of p-adaptation is that the computational grid remains fixed and an exponential error convergence rate with respect to degrees of freedom (DOF) is possible for sufficiently-smooth solutions. A disadvantage, however, is difficulty in handling singularities and areas of anisotropy, and the need for a reasonable starting mesh. h-adaptation allows for the generation of anisotropic (stretched) elements, although the best attainable error convergence rate is algebraic with respect to DOF. hp-adaptation strives to combine the best of both strategies, employing p-refinement in areas where the solution is smooth, and h refinement near singularities or areas of anisotropy. Implemented properly, hp-adaptation can isolate singularities and yield exponential error convergence with respect to DOF. The difficulty of hp-adaptation methods in practice lies in making the decision between h- and p-refinement, a decision that requires either a solution regularity estimate or a heuristic algorithm. Houston and Suli [19] present a review of commonly used methods for making this decision.

The adaptation strategy chosen for this work is h-adaptation at a constant p. This strategy does not take advantage of the cost savings offered by hp-adaptation, but avoids the regularity estimation decision. The h-adaptation method consists of high-order anisotropy detection and mesh optimization.

4.1 Anisotropy in High-Order Solutions

An important ingredient in making h-adaptation efficient for aerodynamic computations is the ability to generate stretched elements in areas where the solution exhibits anisotropy. For $p = 1$, the dominant method for detecting anisotropy involves estimating the Hessian matrix of a scalar solution u [10, 17, 26],

$$H_{ij} = \frac{\partial^2 u}{\partial x_i \partial x_j}, \quad i, j \in [0, .., d-1].$$

The second derivatives can be estimated by, for example, a quadratic reconstruction of the linear solution. For the Euler or Navier-Stokes equations, the Mach number has been found to perform well as the scalar quantity, u. Of course, other quantities may also be suitable, and perhaps the most effective choice is an average or minimum of several quantities [10]. In this work, using the Mach number has produced acceptable results.

The eigenvectors of H correspond to the directions of the maximum and minimum values of the second derivative of u, while their respective eigenvalues, λ_i, are the values of the second derivatives in those directions. Since H is symmetric, the eigenvectors are orthogonal, and yield the principal stretching directions. The magnitudes of stretching in each direction are related via $h_i/h_j = (|\lambda_j|/|\lambda_i|)^{1/2}$. In "pure" Hessian-based adaptation, the absolute magnitude of stretching is controlled by an arbitrary global scaling factor [10]. Venditti and Darmofal use an output-based error indicator to determine the length magnitude, leading to a more robust adaptation process [34]. Formaggia *et al* [14] have combined Hessian-based interpolation error estimates with output-based a posteriori error analysis to arrive at output-based anisotropic error estimates.

Anisotropy detection based on the standard Hessian matrix is not suited for $p > 1$ interpolation, due to the linear interpolation assumption used in the derivation of the Hessian-matrix method. That is, the second derivatives govern, to leading order, the inability of a linear function to interpolate u. On the other hand, for general p, the $p+1$st derivatives of u govern the inability of the basis functions to interpolate the exact solution. An example demonstrating this observation is given in Appendix B. Thus, the stretching ratios, h_i/h_j, and principal directions, $\mathbf{e}_i$, should be based on estimates of the $p+1$st derivatives.

For $p = 1$, the principal directions, $\mathbf{e}_i$, are orthogonal. For $p > 1$, one method for calculating orthogonal directions proceeds as follows: let $\mathbf{e}_0$ be the direction of maximum $p+1$st derivative; let $\mathbf{e}_i$, for $i > 0$, be the direction of maximum $p+1$st derivative in the plane orthogonal to every $\mathbf{e}_j$, $j < i$. Under this definition, the final direction $\mathbf{e}_{d-1}$ is fully determined by the previous directions.

By construction, the $\mathbf{e}_i$ directions are orthogonal, and hence suitable for specifying a metric tensor of directional sizes. Equidistributing the error in each direction yields the relationships,

$$\frac{h_i}{h_j} = \left(u_{\mathbf{e}_j}^{(p+1)} / u_{\mathbf{e}_i}^{(p+1)} \right)^{1/(p+1)}, \tag{4}$$

where $u_{\mathbf{e}_i}^{(p+1)}$ is the $p+1$st derivative in the direction $\mathbf{e}_i$. (4) provides only the relative mesh sizing; the absolute values for h_i are based on the error indicator, as described in the following section.

4.2 Mesh Optimization

In h-adaptation, mesh optimization refers to deciding which elements to refine or coarsen and/or the amount of refinement or coarsening. The optimization has important implications for practical simulations: too little refinement at each adaptation iteration may result in an unnecessary number of iterations; too much refinement may ask for an expensive solve on an overly-refined mesh.

Many of the current adaptation strategies rely on some variation of the fixed fraction method [18, 19, 31], in which a prescribed fraction of elements with the highest error indicator is refined. While adequate for testing and small cases, this method poses an automation and efficiency problem for practical simulations due to the often *ad-hoc* fixed fraction parameter. More sophisticated optimization strategies attempt to meet the global tolerance while equidistributing the error among elements. Zienkiewicz and Zhu [36] define a permissible element error $e_\kappa = e_0/N$ at each adaptation iteration, where e_0 is the global tolerance, and N is the current number of elements. Coupled with an *a priori* error estimate, this "refinement prediction" method yields element sizing at each adaptation iteration. Venditti and Darmofal [33, 34], employ a similar approach and extend it to anisotropic sizing using the Hessian matrix. Compared to the fixed fraction method, refinement prediction has the advantage that it specifies the magnitude of refinement in each element.

For elliptic problems, Rannacher *et al* [5, 30], present another mesh optimization strategy in which an optimal mesh size function $h_{opt}(x)$ is constructed continuously over the entire domain. The construction is based on solving a constrained minimization problem with a Lagrangian method. Details can be found in the references, and in an earlier work by Brandt [8]. Key to this method is an assumption regarding the existence of a mesh-independent function in an expression for the global error. The authors note that this is a heuristic assumption, and that the existence of this function can be rigorously justified only under very restrictive conditions [5].

In the interest of generality, the starting point for this work is the relatively straightforward refinement prediction method of Zienkiewicz and Zhu. One drawback of this method is the fact that error equidistribution is performed over the current mesh, as opposed to some reasonable prediction of the adapted mesh. While in the asymptotic limit, the current and the predicted meshes will converge, by attempting to equidistribute the error on the predicted mesh, adaptive convergence can be accelerated. An example demonstrating this acceleration is provided in Appendix C.

Equidistributing the error on the adapted mesh involves a prediction of the number of elements, N_f, in the adapted (fine) mesh. Let n_κ be the number of fine-mesh elements contained in element κ. n_κ need not be an integer, and $n_\kappa < 1$ indicates coarsening. Denoting the current element sizes of κ by h_i^c, and the requested element sizes by h_i, n_κ can be approximated as,

$$n_\kappa = \prod_i \left(h_i^c / h_i\right). \tag{5}$$

The current sizes h_i^c are approximated by the singular values of the mapping from a unit equilateral triangle to element κ. This procedure is similar to the grid-implied metric used by Venditti [32]. The details are described in Appendix D.

To satisfy error equidistribution, each fine-mesh element is allowed an error of e_0/N_f, which means that each element κ is allowed an error of $n_\kappa e_0/N_f$. By relating changes in element size to expected changes in the local error, an expression for n_κ is obtained, from which the absolute element sizes, h_i, follow. In this work, an *a priori* estimate for the output error serves as this relation,

$$\frac{\epsilon_\kappa}{\epsilon_\kappa^c} = \left(\frac{h_0}{h_0^c}\right)^{\bar{p}_\kappa + 1}, \tag{6}$$

where ϵ_κ^c is the current error indicator, ϵ_κ is the expected error indicator, $\bar{p}_\kappa = \max(p_\kappa, \gamma_\kappa)$, and γ_κ is the lowest order of any singularity within κ [34]. This *a priori* estimate is valid for many common engineering outputs, including forces and pressure distribution norms. In this estimate, the error is assumed to scale with h_0, which corresponds to the direction of maximum $p + 1$st derivative. Equating the allowable error with the expected error from the *a priori* estimate yields

$$\underbrace{n_\kappa \frac{e_0}{N_f}}_{\text{allowable error}} = \underbrace{\epsilon_\kappa^c \left(\frac{h_0}{h_0^c}\right)^{\bar{p}_\kappa + 1}}_{\textit{a priori}\text{ estimate}}. \tag{7}$$

Expressing $\frac{h_0}{h_0^c}$ in terms of n_κ and the known relative sizes (4) yields a relation between n_κ and N_f. Substituting into $N_f = \sum_\kappa n_\kappa$ yields an equation for N_f. If all the $\bar{p}_\kappa$ are equal, this equation can be solved directly. Otherwise, it is solved iteratively. With N_f known, (7) yields n_κ, from which the h_i are calculated using (5) and (4). Adaptation iterations stop when $\epsilon \equiv \sum_\kappa \epsilon_\kappa \le e_0$, where e_0 is the requested global error tolerance.

In practice, two parameters are used to control the behavior of the optimization and adaptation algorithm: the target error fraction, $0 < \eta_t \le 1$, and the adaptation aggressiveness,

$0 \leq \eta_a < 1$. Specifically, instead of e_0 in (7), a modified requested error level, $\tilde{e}_0$ is used, where

$$\tilde{e}_0 = \max\left(\eta_a \epsilon, \eta_t e_0\right).$$

η_t prevents the adaptation convergence from stalling as the error estimate, ϵ, approaches the tolerance, e_0. The aggressiveness parameter, η_a controls how quickly the error is reduced when the error estimate is far from e_0. A value close to zero indicates aggressive adaptation, which has the danger of over-refinement, while a value close to 1 may require an excessive number of adaptation iterations to converge. Default values for these parameters that have been found to work well over a variety of cases are $\eta_t = 0.7$ and $\eta_a = 0.25$.

4.3 Implementation

An adaptive solution procedure starts by solving the primal and dual problems on an initial coarse mesh and calculating a local error indicator on each element. The local error indicators are converted to mesh size requests using the mesh optimization algorithm, and the domain is re-meshed using the new metric. The solution on the new mesh is initialized by a transfer of the solution from the old mesh, and the process repeats. The primal problem is solved using a line-preconditioned Newton GMRES method, and the dual problem is solved sequentially, costing one extra linear solve (or more for multiple outputs).

Meshing of the domain is done using the Bi-dimensional Anisotropic Mesh Generator (BAMG) [7], which takes as input a mesh with the requested metric defined at the input mesh nodes and produces a new mesh based on the requested metric. Since BAMG expects the metric prescribed at the nodes, an averaging process is required to convert the element-based metric to a node-based metric. As this averaging may smooth out small mesh size requests, the input mesh is uniformly refined twice before the call to BAMG.

For robustness, and to speed up convergence, the solution on the new mesh is initialized via a transfer of the solution from the previous mesh. The transfer is performed via an L_2 projection of the state. For solutions with large inter-element jumps (e.g. on coarse meshes), such a projection may produce a non-physical state on the new mesh. In such cases, a $p = 0$ restriction of the solution from the previous mesh is performed.

Error estimation on cut cells remains almost unchanged from that on standard cells, except that care is required to make sure the patch reconstruction does not cross null edges. Cut-cell meshes readily allow for estimates of geometric singularities that affect $\bar{p}_\kappa$ in (6), because corners can be ascribed to individual cut cells. $\bar{p}_\kappa$ on these cells can then be lowered accordingly (to 0 in practice), resulting in isolation of the singularities with fewer adaptation iterations.

Metric definition on null elements, which do not possess a solution, is performed by using the grid-implied metric of the current mesh. Due to inherent smoothing that takes place in a transfer of the metric from elements to nodes for use in the mesher, the metric on null elements is influenced by the solution on neighboring cut elements, resulting in a smooth transition of the metric from cells with a solution to null cells.

5 Results

The adaptation scheme is applied to several representative aerodynamic cases, using orders $p = 1$ to $p = 3$. Comparisons of the adapted meshes and the error convergence histories are given for both boundary-conforming and cut-cell meshes. Degrees of freedom (DOFs) are computed as the total number of unknowns, excluding the equation-specific multiplier (e.g. 4 for 2D Euler or Navier Stokes). In computing the DOF count for cut cell meshes, null cells are not included.

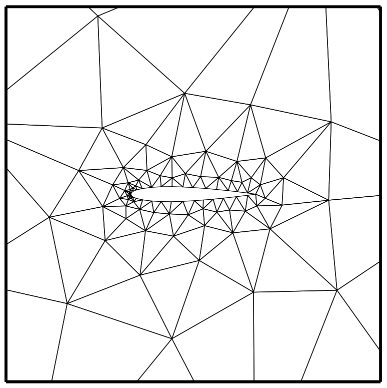
5.1 Inviscid NACA 0012, $M = 0.5$, $\alpha = 2^o$

The computational domain for this case consists of a NACA 0012 airfoil contained within a farfield box, a distance of 100 chord lengths away from the airfoil. The NACA geometry is modified to close the trailing edge gap,

$$y = \pm 0.6(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4). \tag{8}$$

The performance of the isotropic adaptation algorithm is tested using drag as the output, with a tolerance of 0.1 drag counts. Drag is calculated from the static pressure distribution on the airfoil surface with the pressure calculated using only the tangential velocity, $\mathbf{v}_t$: $p_s = (\gamma - 1)(\rho E - 0.5\rho|\mathbf{v}_t|^2)$. The "exact" output value is taken as the drag computed on a $p = 3$ run adapted to 10^{-3} drag counts, as boundary effects of the finite farfield contribute to a nonzero drag value at steady state.

Figure 3 shows the initial 235-element boundary-conforming mesh, as well as the initial 205-element cut-cell mesh. In the boundary-conforming meshes, elements adjacent to the airfoil surface are represented using cubic ($q = 3$) curved elements. These elements have to be curved


(a) Boundary-Conforming: 235 elements

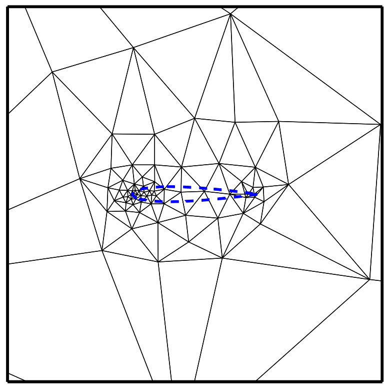

(b) Cut-Cell: 205 elements

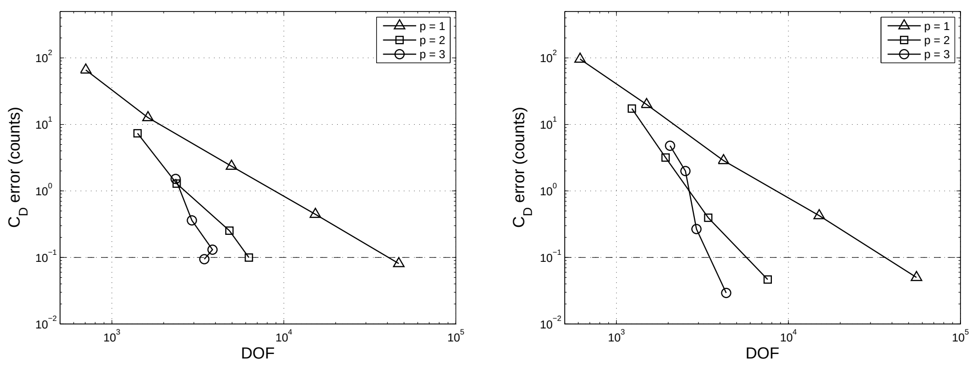
Figure 3: Initial NACA 0012 meshes.

at every adaptation iteration, since BAMG produces linear meshes. For the isotropic elements in this case, this curving does not pose a problem.

Figure 4 summarizes the results of the adaptation runs. The plots show the output error versus DOF at each adaptation iteration for every run. The horizontal dashed line marks the error tolerance of 0.1 drag counts. All runs produce final meshes with errors below this tolerance. The performance of the cut-cell meshes is quite similar to that of the boundary-conforming meshes. In both the boundary-conforming and cut-cell methods, the $p = 3$ runs achieve the desired accuracy with the fewest degrees of freedom. The advantage is roughly a factor of 2 over $p = 2$ and a factor of 10 over $p = 1$.

5.2 NACA 0012, $M = 0.5$, $Re = 5000$, $\alpha = 2^o$

In this case, the full Navier-Stokes solution is computed around a NACA 0012 at Mach number 0.5, Reynolds number 5000, and angle of attack of 2^o. The initial meshes are the same as in the inviscid case (Figure 3). Mesh optimization is performed with anisotropic elements, to efficiently resolve the boundary layer and wake. In the presence of anisotropic elements near the airfoil boundary, the boundary-curving step in post-processing the linear boundary-conforming meshes is prone to failure. That is, the curved boundary may intersect interior



(a) Boundary-Conforming (b) Cut-Cell

Figure 4: Drag error vs. degrees of freedom for the inviscid NACA 0012 runs.

edges, leading to unallowable elements. This mode of failure was observed for some of the runs. When such a failure occurred, the adaptation was re-run with slightly perturbed values for adaptation aggressiveness. No such mesh-robustness problems were encountered for the cut-cell method.

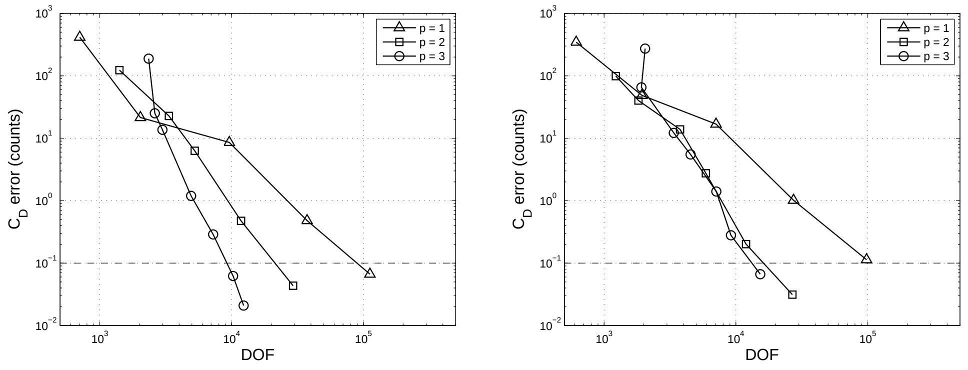
5.2.1 Drag Adaptation

The adaptation algorithm was first tested using drag as the output, with tolerance of 0.1 counts. A force output for a viscous simulation consists of two components: a pressure force and a viscous shear force, $\mathbf{f}^v$. The viscous force is obtained from the viscous flux, F^v_{ki}, with a dual-consistent correction,

$$\begin{bmatrix} \mathbf{f}^v_x \\ \mathbf{f}^v_y \end{bmatrix} = \sum_{\sigma^{bf} \in \Gamma_{\text{out}}} \int_{\sigma^{bf}} \begin{bmatrix} -F^v_{2i} n_i + \eta^{bf} \delta^{bf}_{2i} n_i \\ -F^v_{3i} n_i + \eta^{bf} \delta^{bf}_{3i} n_i \end{bmatrix} ds \tag{9}$$

where F^v_{2i} and F^v_{3i} are viscous flux component, and δ^{bf}_{ki} in the correction is the auxiliary variable presented in the discretization (Appendix A). Not including this correction leads to an adjoint solution that is not well-behaved at the boundary, Γ_{out} [21].

The "true" drag of 568.84 counts was computed on a $p = 3$ cut cell mesh, adapted to an error of 10^{-3} counts. The boundary-conforming and cut-cell runs converge to the same drag value. The corresponding drag error convergence histories are plotted in Figure 5. For the



(a) Boundary-Conforming (b) Cut-Cell

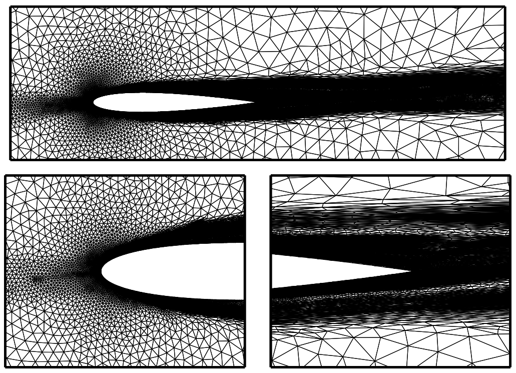
Figure 5: Drag error vs. degrees of freedom for the viscous NACA 0012 runs. Dashed line indicates prescribed tolerance of 0.1 drag counts.

boundary-conforming method, $p = 3$ requires about a factor of 10 fewer DOFs than $p = 1$, and 2.5 fewer DOFs than $p = 2$, to achieve an error of 0.1 counts. For the cut-cell meshes, the spread is smaller, with $p = 3$ performing slightly better than $p = 2$, which requires a factor of 6 fewer DOFs than $p = 1$. Overall, the cut-cell and boundary-conforming results are close.

Figures 6 and 7 show the final adapted meshes for $p = 1$ and $p = 3$. Clearly, the final drag-adapted meshes are significantly finer for $p = 1$ compared to $p = 3$. Areas of high refinement include the boundary layer, a large extent of the wake, and the flow in front of the airfoil. In the cut-cell meshes, the elements in the computational domain are very similar to those in the boundary-conforming meshes, and the metric on the null cells transitions smoothly near the airfoil boundary.

5.2.2 Lift Adaptation

A second test of the adaptation algorithm was performed using lift as the output. Each test case was adapted to 1 count of lift. The "true" lift of 369.4 counts was taken from a solution on a cut-cell mesh, adapted to 0.01 counts. The lift output error, relative to the true value, is shown for all adaptation runs in Figure 8. All runs converged to error levels below the requested tolerance. For both methods, at the requested error level, $p = 1$ adaptation is less efficient than $p = 2$ or $p = 3$ adaptation, in terms of error per DOF. The final lift-adapted


(a) Boundary-Conforming: 37344 elements

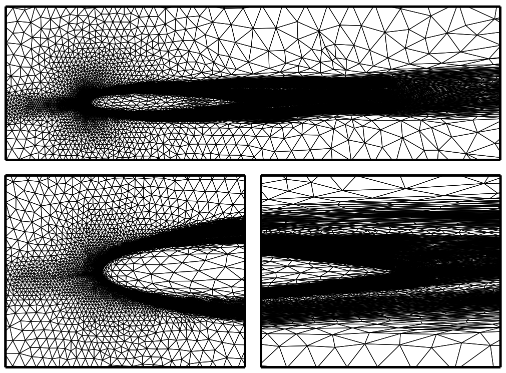
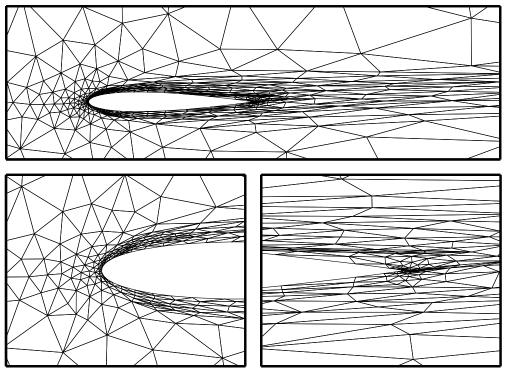

(b) Cut-Cell: 32636 elements

Figure 6: Final $p = 1$ meshes adapted on drag.


(a) Boundary-Conforming: 1233 elements

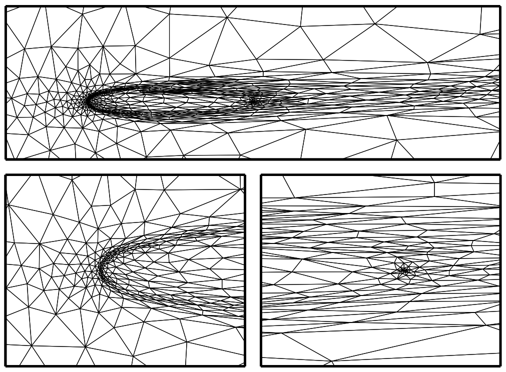
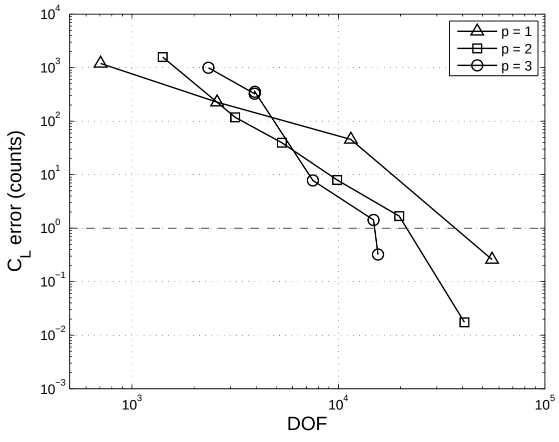
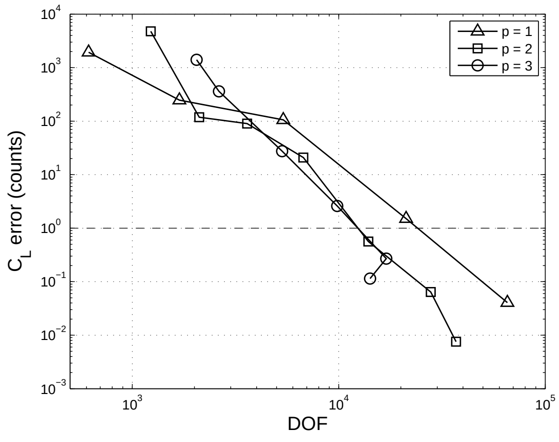

(b) Cut-Cell: 1531 elements

Figure 7: Final $p = 3$ meshes adapted on drag.



(a) Boundary-Conforming



(b) Cut-Cell

Figure 8: Degree of freedom vs. lift output error comparison for $p = 1$ to $p = 3$. Dashed line indicates prescribed tolerance of 1 lift count.

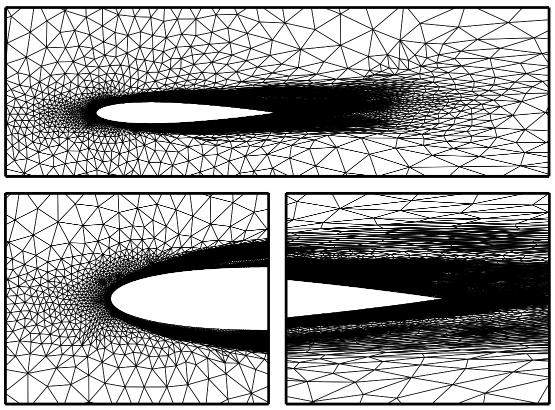
meshes are shown for in Figures 9 and 10 for $p = 1$ and $p = 3$, respectively. Areas targeted for refinement (boundary layer and portions of the wake) are similar to the drag-adaptation case. Anisotropy is very much present in both $p = 1$ and $p = 3$ meshes.

A question of engineering interest is how, for example, lift behaves on drag-adapted meshes, or vice-versa. Figure 11 shows the lift error for solutions computed on the drag-adapted meshes. For both $p = 2$ and $p = 3$, the lift error drops below 1 count on the final meshes. This result suggests that meshes adapted to one output may serve well for the prediction of other (similar) outputs.

5.2.3 Sensitivity to Initial Mesh

For the adaptation method to be practical, the final adapted meshes should not be highly sensitive to the starting meshes. The sensitivity is tested for a drag-adapted viscous NACA 0012, with an error tolerance of 1 drag count. Runs are performed with several different cut-cell starting meshes, including a set of uniform triangulations of the entire domain, as well as two meshes adapted on geometry to different levels of fineness (see Figure 12).

Figure 13 shows the adaptation histories of the runs for $p = 1, 2, 3$. For the finer, uniform starting meshes, the DOFs decrease rapidly in the first adaptation iteration, due to coarsening


(a) Boundary-Conforming: 18522 elements

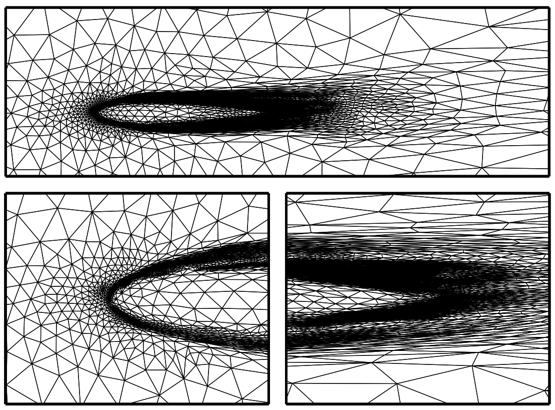
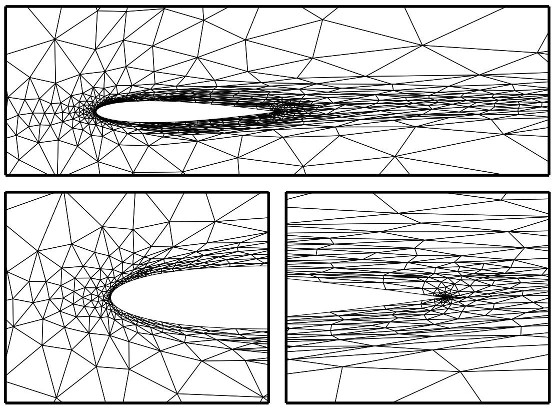

(b) Cut-Cell: 21880 elements

Figure 9: Final $p = 1$ meshes adapted on lift.


(a) Boundary-Conforming: 1557 elements

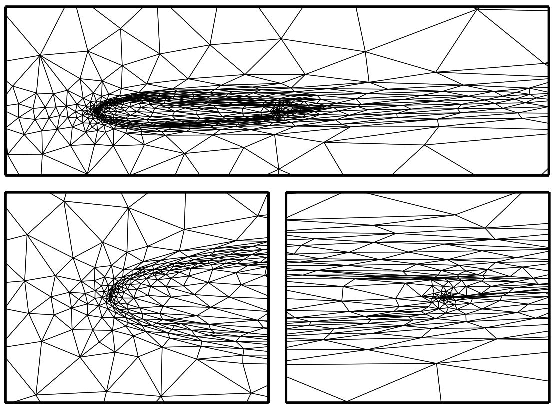
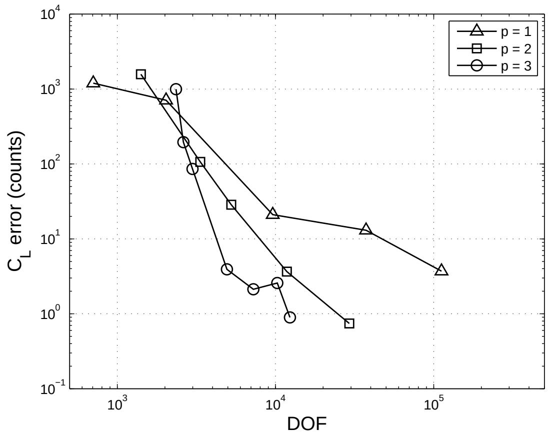
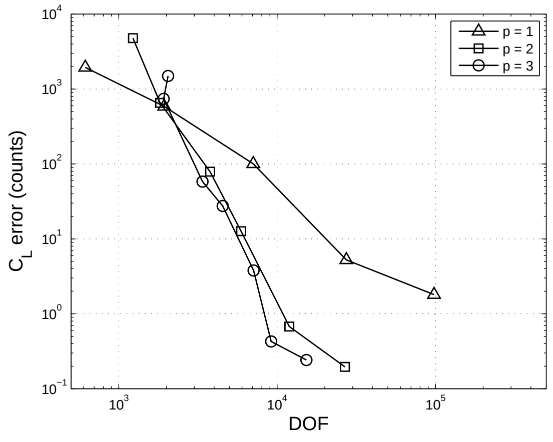

(b) Cut-Cell: 1419 elements

Figure 10: Final $p = 3$ meshes adapted on lift.



(a) Boundary-Conforming



(b) Cut-Cell

Figure 11: Lift output error for runs adapted on drag.

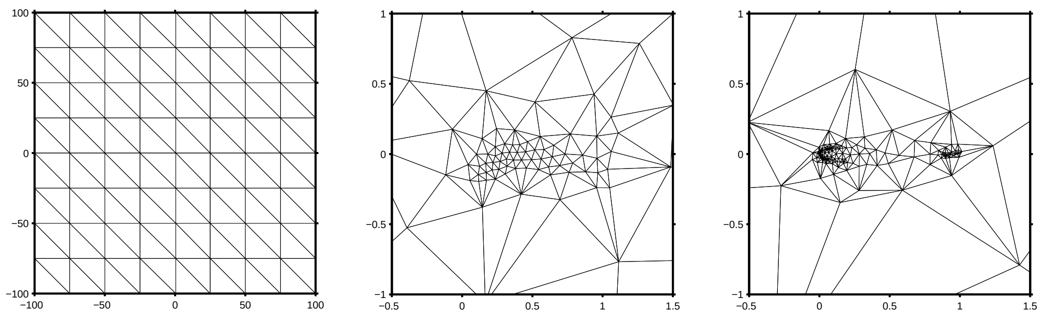
of the mesh away from the airfoil, where the mesh is initially relatively too fine.

The adaptation histories appear somewhat scattered for the first several iterations, but then converge as the error decreases. For a given p, the final adapted meshes are close not only in DOF count, but also in DOF spatial distribution. This observation is made by qualitatively comparing locations of refinement and element aspect ratio. In summary, the histories show that for a low-enough error tolerance, the final meshes generated by the adaptation algorithm are relatively insensitive to the initial mesh.

5.3 NACA 0005, $M = 0.4$, $Re = 50000$, $\alpha = 0^o$

This case considers a thin airfoil in a subsonic, high Re flow. The NACA 0005 geometry is given by (8) with 0.25 for the leading coefficient instead of 0.6. Only cut-cell meshes are presented for this case, as boundary conforming meshes readily yielded invalid elements upon curving the highly-stretched boundary elements. The initial mesh for this case, Figure 14, was created by starting from a $p = 2$, $Re = 5000$ solution mesh and adapting several times to a coarse drag error requirement. The resulting 946 element mesh is relatively coarse, but contains adequate resolution to allow for initial $p = 2$ and $p = 3$ solves at the high Re.

The adaptation algorithm was tested using drag as the output, with a strict tolerance of



(a) Uniform 8x8

(b) Adapted on geom.: 245 elements

(c) Adapted on geom.: 478 elements

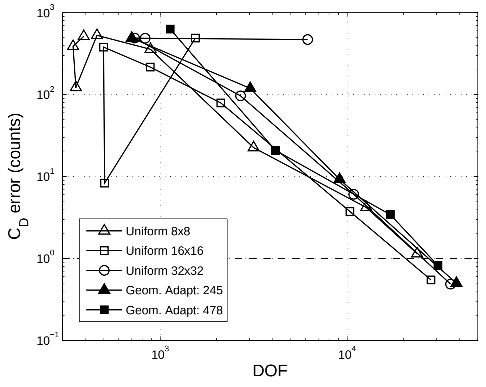
Figure 12: Initial meshes for sensitivity study.

0.04 counts. The "true" drag was computed on a $p = 3$ solution on a mesh obtained by uniformly refining the final $p = 2$ mesh. The drag error histories for $p = 1, 2, 3$ are shown in Figure 14. $p = 3$ converges rapidly, and attains a low drag error under .002 counts with about 15,000 DOF. $p = 2$ exhibits slightly poorer performance, but is still greatly superior to $p = 1$, which requires over $100,000$ DOF to drop below the tolerance.

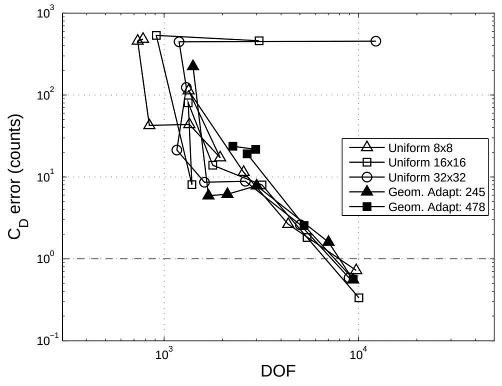
The final adapted meshes are shown in Figure 15 for $p = 1$ and $p = 3$. Clearly, $p = 3$ yields a much coarser mesh. Highly-anisotropic elements are present in the boundary layer and the wake. This high anisotropy makes the boundary-conforming meshes prone to failure, as curving such thin boundary elements inevitably produces invalid elements.

5.4 Diamond airfoil, $M = 1.5$, $\alpha = 0^o$

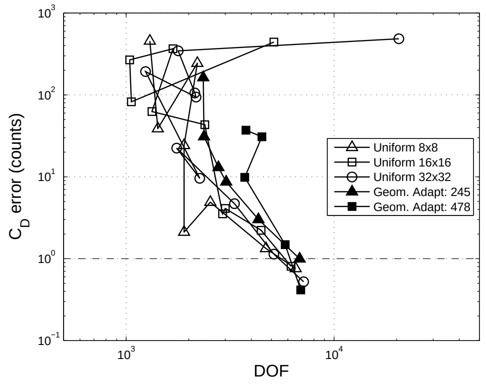
This case considers a diamond airfoil in an inviscid, supersonic flow. The wedge half-angle of the airfoil is 4.995^o, and the farfield is 100 chord lengths away from the airfoil. High-order shock capturing was performed using artificial viscosity and an entropy-based indicator, in a fashion similar to the work of Persson and Peraire [27]. The output of interest in this case is the pressure distribution caused by the sonic boom far away from the airfoil (Figure 17). Specifically, the output chosen is the integral of the pressure along the line segment of interest



(a) $p = 1$



(b) $p = 2$



(c) $p = 3$

Figure 13: Adaptation histories for $p = 2$ and $p = 3$, starting from various initial meshes shown in Figure 12.

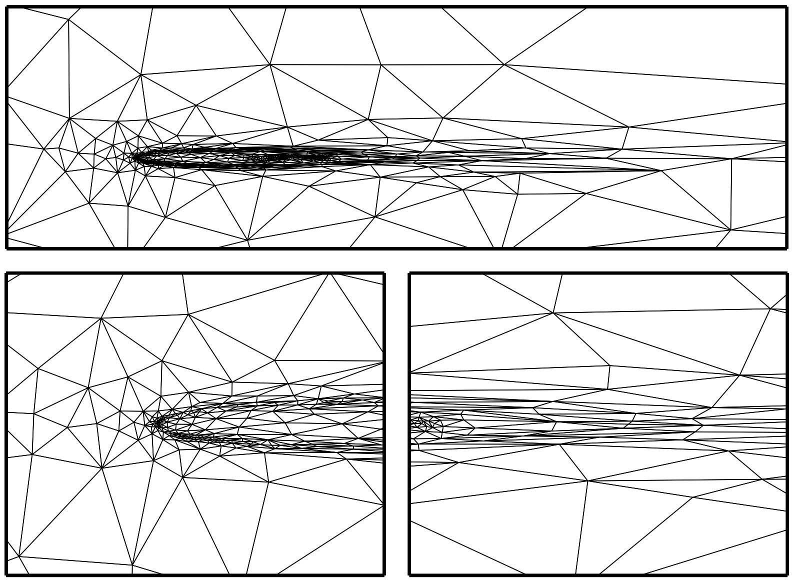


Figure 14: Initial Mesh for NACA 0005 case: 946 elements

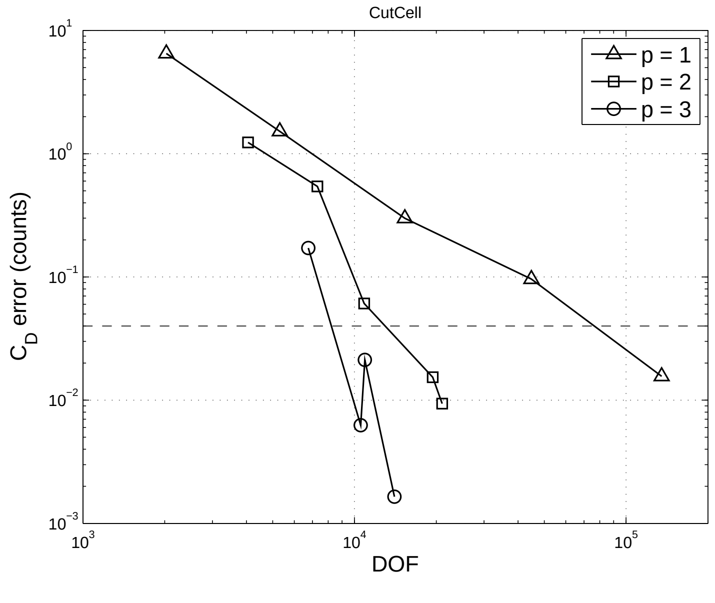
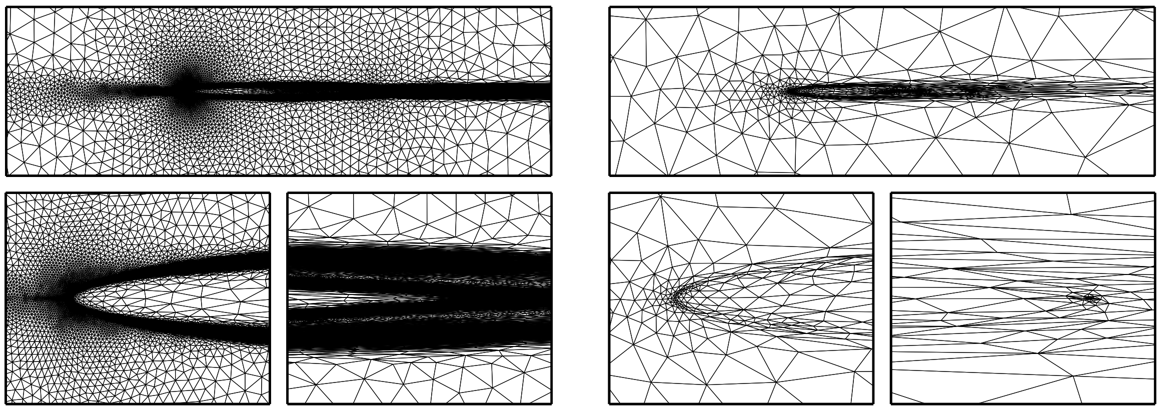


Figure 15: Degree of freedom vs. drag output error comparison for $p = 1$ to $p = 3$. Dashed line indicates prescribed tolerance of .04 drag counts



(a) $p = 1$: 45037 elements (b) $p = 3$: 1403 elements

Figure 16: Final cut-cell meshes adapted on drag.

$(\mathbf{x}_0 - \mathbf{x}_1)$,

$$J = \int_{\mathbf{x}_0}^{\mathbf{x}_1} p(\mathbf{x}) d\mathbf{x}. \tag{10}$$

The initial diamond airfoil meshes are shown in Figure 18. The boundary-conforming mesh is coarse, but the cut-cell mesh is even coarser, with only 74 elements.

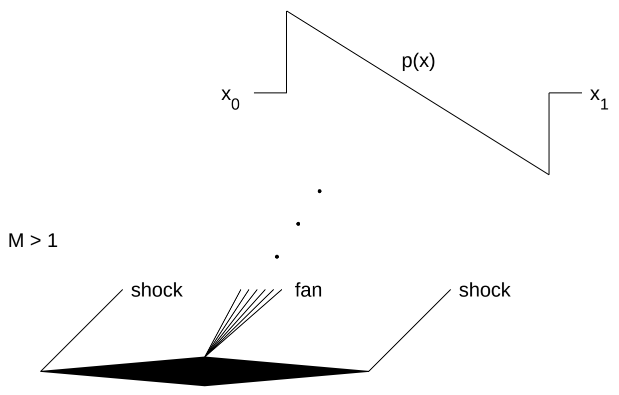
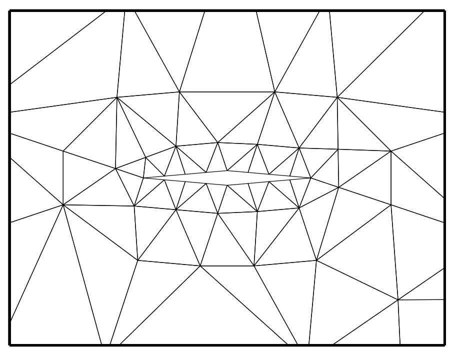


Figure 17: Setup for diamond airfoil case.

For the adaptation runs, the pressure was measured along a line with $x_0 = (17.5, 20)$.


(a) Boundary-Conforming: 206 elements

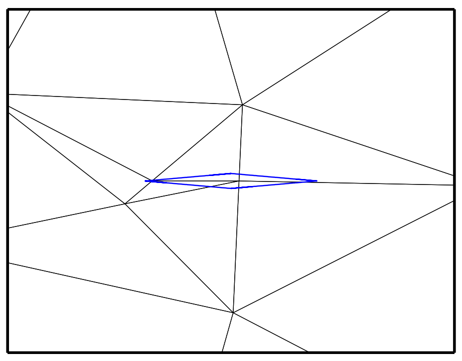
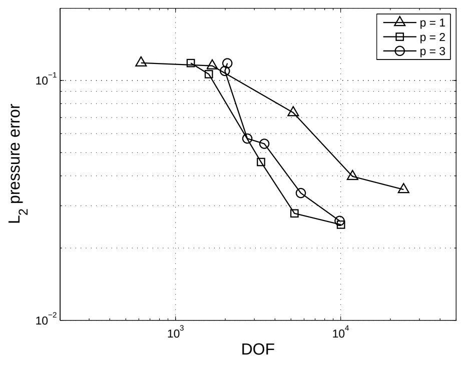

(b) Cut-Cell: 74 elements

Figure 18: Initial diamond airfoil meshes.

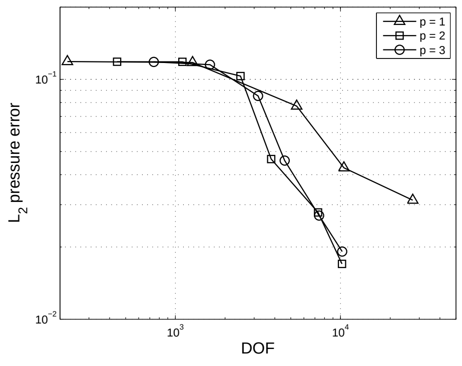
$x_1 = (27.5, 20)$, measured in chord lengths. The requested output error for the pressure integral was set to .0075, normalized by the freestream static pressure and the chord length. The "true" pressure distribution was taken to be the one from a $p = 2$ solution on a $30,000$-element mesh, which was obtained by uniformly refining an adapted mesh. The L_2 pressure error, normalized by the freestream static pressure and the chord, is plotted in Figure 19. For all runs, the L_2 error decreases with adaptation iteration, which suggests that the pressure integral output leads to pressure matching in distribution. The performance of $p = 3$ is comparable to $p = 2$ in both methods, while $p = 1$ is less efficient.

The pressure distributions on the intermediate and final meshes at $p = 1, 2, 3$ are shown in Figure 20 for the cut-cell meshes. The distributions for the boundary-conforming meshes are similar. The pressure in these plots has been normalized by the static freestream pressure. The exact distribution is shown as a dashed line. Oscillations near the pressure discontinuities appear on some of the $p = 2$ and $p = 3$ solutions, but they are not significant in the final adapted solutions.

Figures 21 and 22, show the final adapted meshes for $p = 1$ and $p = 3$. Each figure shows a distant view of the airfoil and adapted shocks, as well as two closeups: one near the pressure integral line, and one near the airfoil. The heavy refinement of the shocks for the $p = 1$ mesh becomes lighter for $p = 3$. Near the airfoil, the grid also becomes coarser from

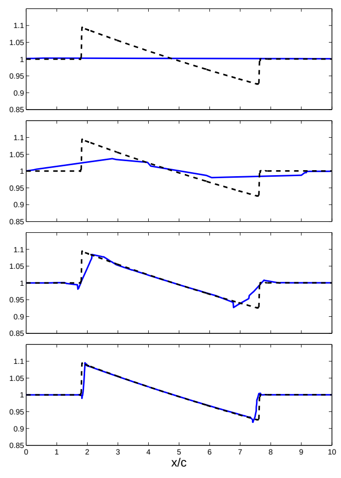


(a) Boundary-Conforming

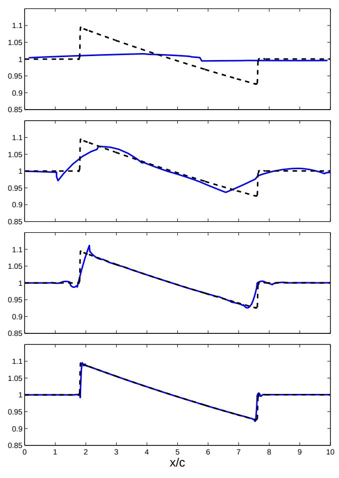


(b) Cut-Cell

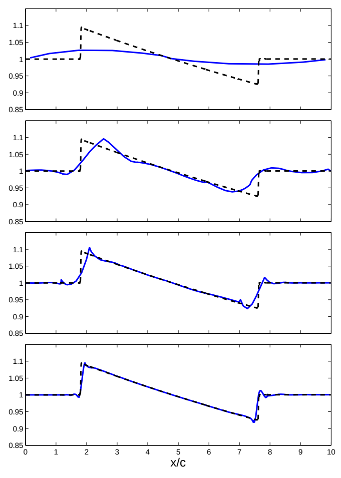
Figure 19: Normalized L_2 pressure error vs. degrees of freedom for the supersonic diamond airfoil case.



(a) p = 1

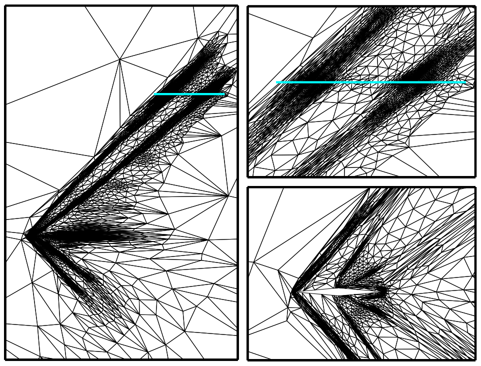


(b) p = 2



(c) p = 3

Figure 20: Evolution of pressure distributions on the test line, for the cut-cell runs. Plots show last four adapted grids for each order.


(a) Boundary-Conforming: 8011 elements

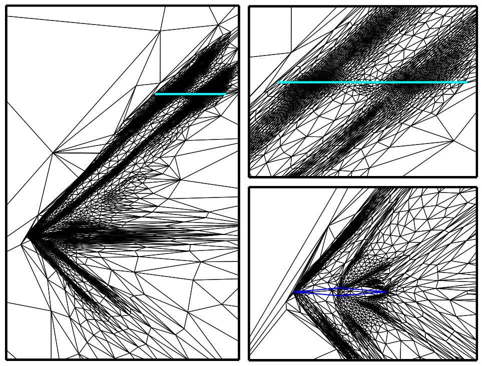
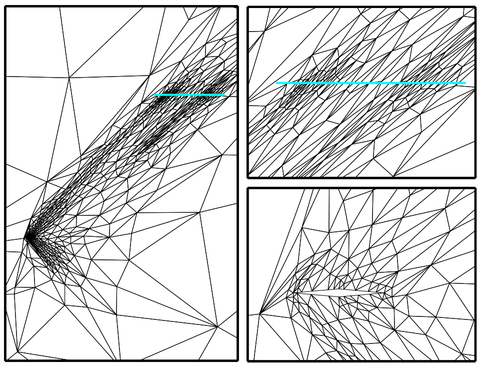

(b) Cut-Cell: 9131 elements

Figure 21: Final $p = 1$ meshes.


(a) Boundary-Conforming: 985 elements

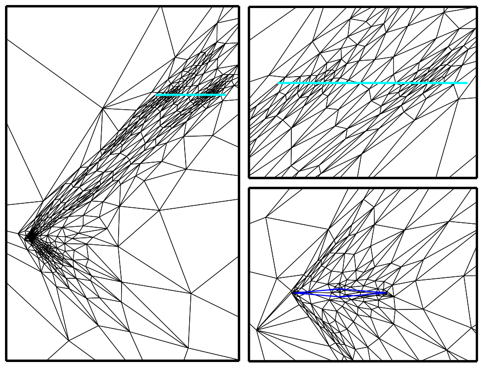

(b) Cut-Cell: 1024 elements

Figure 22: Final $p = 3$ meshes.

$p = 1$ to $p = 3$. The $p = 1$ solutions contain significant anisotropic refinement behind the diamond airfoil, which is not present for the $p = 2$ or $p = 3$ runs. This effect could be due to a wake caused by the increased dissipation of the $p = 1$ discretization.

6 Conclusions

This report presents a complete output-based mesh adaptation procedure for higher-order discontinuous Galerkin discretizations in two dimensions. While the target application is aerodynamics, the adaptation method is readily extendable to different equation sets. Similarly, while this work considered only two dimensions, most of the ideas are extendable to three dimensions. Specifically, the output-based error estimation and mesh optimization strategy can be readily extended from two to three dimensions. Regarding cut cells, the required three-dimensional intersection problems certainly become more difficult. However, the idea of tetrahedral cut cells offers an alternative to the difficult problem of generating boundary-conforming meshes around intricate, curved three-dimensional geometries.

From the two-dimensional results given in this work, several conclusions can be drawn about the performance of the adaptation algorithm. First, the output-based error estimate, while not a bound on the error, successfully drives the adaptation on the cases tested to produce solutions that meet the prescribed error tolerance on the output. Second, adaptation on $p = 2$ and $p = 3$ is observed to produce final meshes that more efficiently use degrees of freedom compared to $p = 1$ meshes. The difference in degrees of freedom is largest for the smooth, inviscid case, and still remains significant for the viscous and supersonic cases. Third, by running the adaptation algorithm using a variety of initial meshes, the conclusion can be made that the final meshes are relatively insensitive to the starting meshes, given a low enough error tolerance.

Finally, the cut cell method is shown to produce the same results on qualitatively similar grids as compared to the boundary-conforming method. Moreover, increased robustness of the cut-cell method is observed for anisotropic adaptation, in which the boundary-conforming meshes are prone to failure in post-processing of the curved boundaries. These results support the concept of using triangular cut cells for automated, robust, and efficient meshing.

A Compressible Navier-Stokes Discretization

While the error estimation, adaptation, and cut-cell methods presented are valid for general equations, the target application is the compressible Navier-Stokes equations. For completeness, this appendix presents the discretization used. The compressible Navier-Stokes equations written using index notation read

$$\partial_i F_{ki} - \partial_i F^v_{ki} = 0, \tag{11}$$

where i and j index the spatial dimension, while k and l index the physical state vector (e.g. size 4 for Navier-Stokes in two dimensions). F_{ki} and F^v_{ki} are inviscid and viscous flux components, respectively. In two dimensions [25],

$$F_{ki} = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho v u \\ \rho u v & \rho v^2 + p \\ \rho u H & \rho v H \end{bmatrix}$$

$$F^v_{ki} = \begin{bmatrix} 0 & 0 \\ \frac{2}{3}\mu(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}) & \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \frac{2}{3}\mu(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}) \\ \frac{2}{3}\mu(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y})u + \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})v + \kappa^T \frac{\partial T}{\partial x} & \frac{2}{3}\mu(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x})v + \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})u + \kappa^T \frac{\partial T}{\partial y} \end{bmatrix}.$$

Relevant physical quantities are density (ρ), velocity (u, v), total energy (E), total enthalpy (H), pressure (p), temperature (T), gas constant (R), specific heat ratio (γ), dynamic viscosity (μ), and thermal conductivity (κ^T). These quantities are related via

$$\begin{aligned} H &= E + p/\rho, \\ p &= (\gamma - 1)\left(\rho E - \frac{1}{2}\rho(u^2 + v^2)\right). \end{aligned}$$

Sutherland's law is used for μ. The fluxes are non-linear functions of the state variables, u_k. It is convenient to make use of the linear dependence of F^v_{ki} on the spatial gradients $\partial_j u_l$ by writing

$$F^v_{ki} = A_{kilj}\partial_j u_l, \tag{12}$$

where A_{kilj} is a tensor that is a nonlinear function of u_k.

The discretization of (11) proceeds in standard finite-element fashion by triangulating the computational domain, Ω, into elements κ and searching for a solution in a finite-dimensional space, $\mathcal{V}^p_h$, for which the weak form is satisfied. In this work, the space of piecewise polynomials of order p over the elements is used. For clarity, the weak form is presented here for one element κ with boundary $\partial\kappa$. The discrete semi-linear form $\mathcal{R}_h$ follows by summing over all elements. In the equations that follow, $v_k \in \mathcal{V}^p_h$ denotes an arbitrary test function. To lighten notation, u_k henceforth refers to the approximate solution in $\mathcal{V}^p_h$. On $\partial\kappa$, the notation $()^+, ()^-$ refers to quantities taken from the interior, exterior of κ, respectively.

The contribution of the inviscid flux term to the weak form is

$$E_\kappa = -\int_\kappa \partial_i v_k F_{ki} d\mathbf{x} + \int_{\partial\kappa} v_k \widehat{F}_{ki}(u^+, u^-) n_i ds,$$

where n_i is the outward pointing normal, and $\widehat{F}_{ki}$ is an approximate characteristic-based flux function (Roe-averaged flux in this work). Boundary conditions are imposed by setting $\widehat{F}_{ki}$ appropriately when $\partial\kappa$ is on $\partial\Omega$. Details are given in [25].

The viscous flux term contribution is discretized using the second form of Bassi & Rebay (BR2) [4]. In this form, the Navier-Stokes equations are re-written as a system of first order equations by introducing Q_{ki},

$$\begin{aligned} \partial_i F_{ki} - \partial_i Q_{ki} &= 0, \\ Q_{ki} - A_{kilj}\partial_j u_l &= 0. \end{aligned}$$

Discretizing both equations and using integration by parts yields the viscous contribution to the weak form,

$$V_\kappa = \int_\kappa \partial_i v_k A_{kilj}\partial_j u_l dx - \int_{\partial\kappa} \partial_i v^+_k \left(A^+_{kilj} u^+_l - \widehat{A_{kilj} u_l}\right) n_j ds - \int_{\partial\kappa} v^+_k \widehat{Q}_{ki} n_i ds,$$

where $\widehat{\cdot}$ denotes flux averaging for discontinuous quantities. The choice of averaging is not unique, but only certain choices produce discretizations that are both consistent, dual-consistent, and compact [21]. The set of fluxes used in this work is shown in Table 1.

Table 1: Viscous fluxes

	$\widehat{Q}_{ki}$	$\widehat{A_{kilj} u_l}$
Interior	$\{A_{kilj}\partial_j u_l\} - \eta^f \{\delta^f_{ki}\}$	$A^+_{kilj}\{u_l\}$
Boundary, Dirichlet	$A^+_{kilj}\partial_j u^+_l - \eta^{bf}\delta^{bf}_{ki}$	$A^b_{kilj} u^b_l$
Boundary, Neumann	$\Big(A_{kilj}\partial_j u_l\Big)^b$	$A^+_{kilj} u^+_l$

The operator $\{\cdot\}$ denotes the mean, $\{\cdot\} = \frac{1}{2}\big((\cdot)^+ + (\cdot)^-\big)$, the superscript b indicates values taken from states appropriately constructed using boundary conditions, and η^f and η^{bf} are stability coefficients that could depend on the triangulation [25]. $\delta^f_{ki}, \delta^{bf}_{ki} \in [\mathcal{V}^p_h]^2$ are auxiliary variables for interior and boundary faces, respectively, that satisfy, $\forall \tau_{ki} \in [\mathcal{V}^p_h]^2$,

$$\begin{aligned} \int_{\kappa^+} \delta^{f+}_{ki}\tau_{ki} dx + \int_{\kappa^-} \delta^{f-}_{ki}\tau_{ki} dx &= \int_{\sigma^f} \{\tau_{ki} A_{kilj}\}\Big(u^+_l - u^-_l\Big) n_j ds, \\ \int_{\kappa} \delta^{bf}_{ki}\tau_{ki} dx &= \int_{\sigma^{bf}} \tau_{ki} A^b_{kilj}\Big(u^+_l - u^b_l\Big) n_j ds, \end{aligned}$$

where σ^f and σ^{bf} denote interior and boundary faces, respectively, and κ^+, κ^- are elements on either side of σ^f. This viscous discretization yields a compact stencil in that the element-to-element influence is only nearest neighbor. A weighted residual statement over the entire domain is obtained by summing the E_κ and V_κ contributions from each element, and setting that sum to zero,

$$\mathcal{R}_h\big(\mathbf{u}_h, \mathbf{v}_h\big) = \sum_{\kappa} (E_\kappa + V_\kappa) = 0,$$

where $\mathbf{u}_h$ and $\mathbf{v}_h$ denote the collection of u_k and v_k over all elements κ.

B Adapting for $p > 1$ Interpolation

The following example shows the problem of using the Hessian matrix for measuring high-order anisotropy.

Consider the following function u on $[0,1] \times [0,1]$:

$$u = 1.0 + (x^2 + 16y^2) + \epsilon(64x^3 + y^3), \tag{13}$$

where $\epsilon << 1$. The Hessian matrix is

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 32 \end{bmatrix} + O(\epsilon). \tag{14}$$

Ignoring $O(\epsilon)$ terms, the required element aspect ratio (AR) is calculated to be

$$AR \equiv \frac{\Delta x}{\Delta y} = \sqrt{\frac{32}{2}} = 4.0.$$

To test how interpolation accuracy varies with grid AR, a sequence of three grids, $AR = 0.25$, $AR = 1.0$, and $AR = 4.0$ was constructed, each with 32 elements (Figure 23).

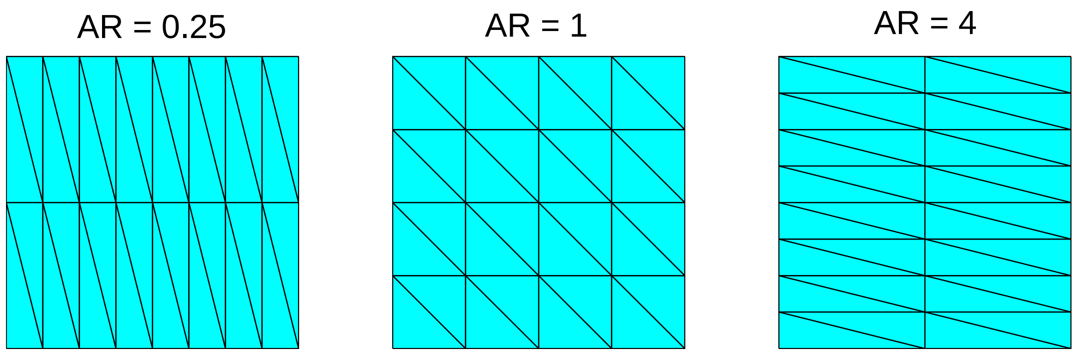


Figure 23: Three grids used for the interpolation of the function u in (13). Hessian-based analysis predicts $AR = 4.0$ as the ideal for interpolation.

The L_2 norm of the interpolation error on each grid was calculated for $p = 1$ and $p = 2$ interpolation. $\epsilon = 10^{-4}$ was used for the calculations. Table 2 shows the results. As expected

Table 2: L_2 norm of interpolation error on each grid for $p = 1$ and $p = 2$. $\epsilon = 10^{-4}$.

	$p = 1$	$p = 2$
$AR = 0.25$	2.31E-1	2.19E-7
$AR = 1.0$	5.91E-2	1.42E-6
$AR = 4.0$	2.37E-2	1.14E-5

from the Hessian matrix analysis, for $p = 1$, the grid with $AR = 4.0$ shows the smallest interpolation error. For $p = 2$, however, $AR = 0.25$ produces the smallest interpolation error per degree of freedom, out of the three cases tested. $AR = 4.0$ produces an error 50 times larger. Thus, using the results from the Hessian matrix for high order interpolation can produce a clearly sub-optimal grid.

Considering the third order derivatives in (13) gives insight into the AR expected for $p = 2$ interpolation. Specifically, $u_{xxx} = 384\epsilon$ and $u_{yyy} = 6\epsilon$. For this problem, these values happen to be the maximum and minimum third-order derivatives, and they also happen to occur in orthogonal directions (x and y). Thus, elements that equidistribute the interpolation error in these directions should have:

$$u_{xxx}(\Delta x)^3 = u_{yyy}(\Delta y)^3 \quad \Rightarrow \quad AR = \frac{\Delta x}{\Delta y} = \left(\frac{u_{yyy}}{u_{xxx}}\right)^{1/3} = \left(\frac{6\epsilon}{384\epsilon}\right)^{1/3} = .25$$

This is the AR that produced the smallest $p = 2$ interpolation error out of the three cases tested (c.f. Table 2).

C Refinement Prediction Example

One drawback of Zienkiewich and Zhu's standard refinement prediction method [36] lies in its error equidistribution capability. In particular, the method may lead to over-refinement of elements with large error indicators, and under-refinement of elements with small error indicators. This problem occurs due to the use of error equidistribution on the current mesh rather than on some reasonable prediction of the final mesh. To illustrate the significance of this problem, consider a two-element one-dimensional mesh, with error indicators $\epsilon_0 = 1$ and $\epsilon_1 = 16$, and a global error target of $e_0 = 1.0$. Assume that the error indicators are additive (e.g. L_1 norm of error), and that the *a priori* error estimate is $\epsilon_k \sim h_k^1$.

Using the refinement prediction method with $N = 2$, the permissible error on each element is $\bar{e}_0 = e_0/N = 1.0/2 = 0.5$. Calculation of the new h_k via the *a priori* estimate leads to element 0 being split into 2 children and element 1 being split into 32 children, for a total of 34 elements in the new grid (lower left in Figure 24). Assuming the *a priori* estimate is exact, the global error target is met, since each of the areas covered by the original two elements contributes 0.5 to the global error. However, the error is not equidistributed on the new grid. Each child of element 0 has an error of $0.5/2 = 0.25$, while each child of element 1 has an error of $0.5/32 = 0.015625$. That is, element 1 has been over-refined while element 0 has not been refined enough to satisfy error equidistribution on the resulting mesh.

Now consider a refinement obtained using the technique described in Section 4.2, which calls for element 0 being split into 5 elements and element 1 being split into 20 elements. The

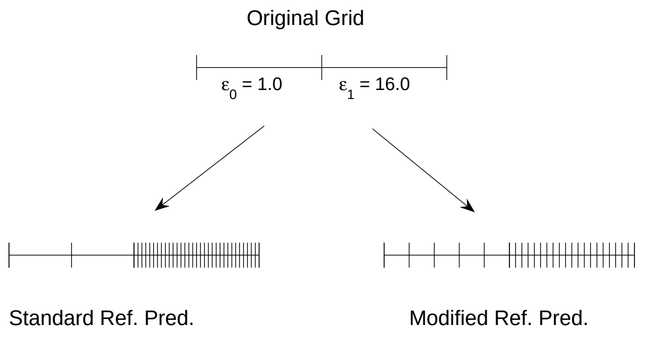


Figure 24: Example of global mesh modification on a 1D mesh using standard refinement prediction (lower left) and using the technique described in Section 4.2 (lower right).

total is 25 elements in the new mesh (lower right in Figure 24). Altogether, the children of element 0 now contribute $1/5$ to the global error, so each contributes $1/5^2 = 1/25$. Also, the children of element 1 contribute a cumulative $16/20 = 4/5$ to the global error, so each contributes $(4/5)/20 = 1/25$. The global error target is met ($1/5 + 4/5 = 1.0$), and the error is equidistributed on the final mesh. Finally, the number of elements in this mesh is fewer than the 34 obtained from refinement prediction.

D Grid-Implied Metric Calculation

In calculating anisotropic mesh sizing, the existing (grid-implied) metric is required for each element. This metric is calculated via a mapping, $\mathbf{M}$, from a unit equilateral triangle to the element in question. The motivation for using this mapping is that it describes stretching and rotation associated with the current element.

One way to calculate $\mathbf{M}$ that makes use of existing functions in the code is to write

$$\mathbf{M} = \mathbf{J}_\kappa \mathbf{J}_E^{-1},$$

where $\mathbf{J}_\kappa$ is the Jacobian matrix for triangle κ and $\mathbf{J}_E$ is the Jacobian matrix for the unit equilateral triangle. Geometrically, the Jacobian $\mathbf{J}_\kappa$ maps the reference unit right triangle to κ. Thus, $\mathbf{J}_\kappa \mathbf{J}_E^{-1}$ maps the unit equilateral triangle to κ, as shown in Figure 25.

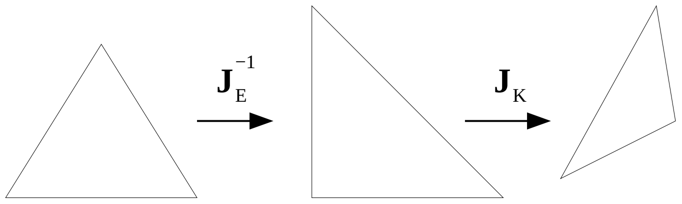


Figure 25: Mapping from unit equilateral triangle to triangle κ.

Writing the singular value decomposition of $\mathbf{M}$,

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

gives insight into the rotation and stretching for triangle κ. Specifically, the two singular values in $\mathbf{\Sigma}$ dictate the amount of stretching in each of the principal directions (h_1 and h_2), while the rotation $\mathbf{U}$ dictates the orientation of these principal directions (θ_1 and θ_2).

E Cubic Spline Intersection

The intersections between a cubic spline segment and an ordinary line can be found analytically. Consider a spline segment between two knots (1 and 2), as shown in Figure 26.

The spline information provides, at each knot, the knot coordinates as well as the derivatives of each coordinate with respect to the arc length parameter, s. Specifically, at knot 1, we have $S_1, X_1, Y_1, XS_1, YS_1$, where $XS \equiv dx/ds, YS \equiv dy/ds$, and similarly for knot 2. The (x, y) coordinates along the curved spline segment can be written as

$$\begin{aligned} x(t) &= tX_2 + (1-t)X_1 + (t-t^2)\big[(1-t)CX_1 - tCX_2\big], \\ y(t) &= tY_2 + (1-t)Y_1 + (t-t^2)\big[(1-t)CY_1 - tCY_2\big], \end{aligned}$$

where $\Delta s = S_2 - S_1$, $t = (s - S_1)/\Delta s$, and

$$\begin{aligned} CX_1 &= XS_1\Delta s - (X_2 - X_1), \\ CX_2 &= XS_2\Delta s - (X_2 - X_1). \end{aligned}$$

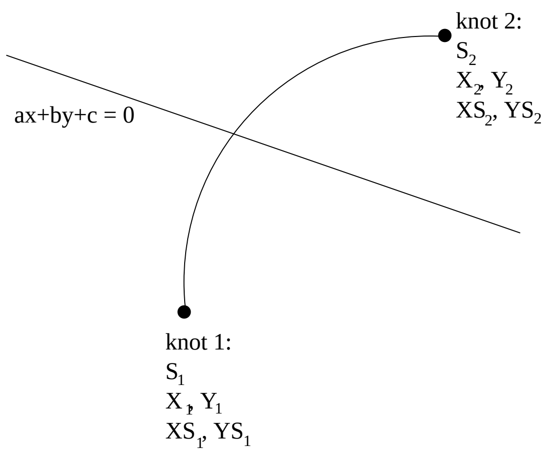


Figure 26: Intersection between a spline segment and a line. The root-finding formula is used to determine where a spline segment cuts an interior edge.

CY_1 and CY_2 are defined similarly. A line segment is defined as points x, y that satisfy $ax + by + c = 0$. Thus, a spline/line intersection must satisfy

$$\begin{aligned} 0 &= ax(t) + by(t) + c \\ &= \Big[a(CX_1 + CX_2) + b(CY_1 + CY_2)\Big]t^3 + \Big[a(-2CX_1 - CX_2) + b(-2CY_1 - CY_2)\Big]t^2 \\ &\quad + \Big[a(X_2 - X_1 + CX_1) + b(Y2 - Y_1 + CY_1)\Big]t + \Big[aX_1 + bY_1 + c\Big] \\ &\equiv a_3t^3 + a_2t^2 + a_1t + a_0. \end{aligned} \tag{15}$$

Thus, an intersections is a solution to a cubic polynomial. Analytical formulas for cubic solutions exist. However, one must be careful of special cases and numerical conditioning. An example of a cubic-root formula is as follows: given a cubic equation in t,

$$t^3 + b_2t^2 + b_1t + b_0 = 0, \tag{16}$$

define

$$Q \equiv \frac{3b_1 - b_2^2}{9}, \qquad R \equiv \frac{9b_2b_1 - 27b_0 - 2b_2^3}{54}, \qquad D \equiv Q^3 + R^2.$$

- if $D = 0$: All roots are real, and at least two are the same:

$$\begin{aligned} t_0 &= -b_2/3 + 2R^{1/3}, \\ t_1 = t_2 &= -b_2/3 - R^{1/3}. \end{aligned}$$

- if $D > 0$: One real root:

$$t_0 = -b_2/3 + \left(R + \sqrt{D}\right)^{1/3},$$

- if $D < 0$: Note this implies $Q < 0$. Let $\theta \equiv \cos^{-1}\left[R/(-Q)^{3/2}\right]$. Three distinct roots:

$$\begin{aligned} t_0 &= 2\sqrt{-Q}\cos(\theta/3) - b_2/3 \\ t_1 &= 2\sqrt{-Q}\cos((\theta + 2\pi)/3) - b_2/3 \\ t_3 &= 2\sqrt{-Q}\cos((\theta + 4\pi)/3) - b_2/3 \end{aligned}$$

The above cubic formula assumes that the leading coefficient (t^3) is 1. However, a straightforward division by a_3 in (15) may not be well-conditioned if a_3 is small, which is often the case when a spline segment is close to linear. Ideally, we would like the b_i coefficients to be neither much smaller nor much larger than unity. A transformation that has worked in practice is to solve for $r = a_3^{1/3}t$. Substituting for t in (15) yields:

$$r^3 + a_2 a_3^{-2/3} r^2 + a_1 a_3^{-1/3} r + a_0 = 0.$$

t is then obtained as $t = a_3^{-1/3} r$. The special case of $|a_3|$ close to machine precision reduces the equation to a quadratic (at most), which is solved via a similar conditioning scaling.

Valid solutions consist of those t that are between 0 and 1 and for which the associated (x, y) lie within the line segment bounding box (the desired intersections are generally for finite line segments not infinite lines). Double roots, or tangency intersections are not counted as intersections, while triple roots are. Node intersections near spline-segment endpoints have to be handled carefully, especially for near-tangency intersections. Specifically, if the cubic solution yields t close to 0 or 1, $t = 0$ or $t = 1$ are tried to see if they yield a better answer (in terms of equation residual).

References

[1] Michael J. Aftosmis, Marsha J. Berger, and Juan J. Alonso. Applications of a Cartesian mesh boundary-layer approach for complex configurations. Paper 2006-0652, AIAA, 2006.

[2] M.J. Aftosmis, M.J. Berger, and J.M. Melton. Adaptive Cartesian mesh generation. In J.F Thompson, B.K. Soni, and N.P. Weatherill, editors, *Handbook of Grid Generation*. CRC Press, 1998.

[3] Timothy Barth and Mats Larson. A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes. In R. Herban and D. Kröner, editors, *Finite Volumes for Complex Applications III*, London, 2002. Hermes Penton.

[4] F. Bassi and S. Rebay. GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations. In Karniadakis Cockburn and Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, pages 197–208. Springer, Berlin, 2000.

[5] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. In A. Iserles, editor, *Acta Numerica*. Cambridge University Press, 2001.

[6] Marsha J. Berger and Randall J. Leveque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. Paper 1989-1930, AIAA, 1989.

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

[8] Achi Brandt. *Guide to Multigrid Development*. Springer-Verlag, 1982.

[9] Donna Calhoun and Randall J. LeVeque. A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries. *Journal of Computational Physics*, 157:143–180, 2000.

[10] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulations. *International Journal for Numerical Methods in Fluids*, 25:475–491, 1997.

[11] D. Keith Clarke, M. D. Salas, and H. A. Hassan. Euler calculations for multielement airfoils using Cartesian grids. *AIAA Journal*, 24(3):353, 1986.

[12] William J. Coirier and Kenneth G. Powell. Solution-adaptive cut-cell approach for viscous and inviscid flows. *AIAA Journal*, 34(5):938–945, 1996.

[13] Krzysztof J. Fidkowski. A high-order discontinuous Galerkin multigrid solver for aerodynamic applications. MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.

[14] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation with applications to CFD problems. In H. A. Mang, F. G. Rammerstorfer, and J. Eberhardsteiner, editors, *Fifth World Congress on Computational Mechanics*, Vienna, Austria, July 7-12 2002.

[15] R. L. Gaffney, M. D. Salas, and H. A. Hassan. Euler calculations for wings using Cartesian grids. Paper 1987-0356, AIAA, 1987.

[16] M. B. Giles and E. Süli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. In *Acta Numerica*, volume 11, pages 145–236, 2002.

[17] Wagdi G. Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. part i: general principles. *Int. J. Numer. Meth. Fluids*, 32:725–744, 2000.

[18] Ralf Hartmann and Paul Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *Journal of Computational Physics*, 183(2):508–532, 2002.

[19] Paul Houston and Endre Suli. A note on the design of hp-adaptive finite element methods for elliptic partial differential equations. *Comput. Methods Appl. Mech. Engrg*, 194:229–243, 2005.

[20] Steve L. Karman. Splitflow: A 3d unstructured Cartesian/prismatic grid CFD code for complex geometries. Paper 1995-0343, AIAA, 1995.

[21] James Lu. *An a Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[22] J. D. Müller and M. B. Giles. Solution adaptive mesh refinement using adjoint error analysis. Paper 2001-2550, AIAA, 2001.

[23] Scott M. Murman, Michael J. Aftosmis, and Stuart E. Rogers. Characterization of space shuttle ascent debris aerodynamics using CFD methods. Paper 2005-1223, AIAA, 2005.

[24] Marian Nemec, Michael Aftosmis, Scott Murman, and Thomas Pulliam. Adjoint formulation for an embedded-boundary Cartesian method. Paper 2005-0877, AIAA, 2005.

[25] Todd A. Oliver. Multigrid solution for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. MS thesis, M.I.T., Department of Aeronautics and Astronautics, August 2004.

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

[27] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. AIAA Paper 2006-112, 2006.

[28] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, 2000.

[29] James W. Purvis and John E. Burkhalter. Prediction of critical Mach number for store configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[30] R. Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128:205–233, 2001.

[31] P. Solin and L. Demkowicz. Goal-oriented hp-adaptivity for elliptic problems. *Comput. Methods Appl. Mech. Engrg.*, 193:449–468, 2004.

[32] D. A. Venditti. *Grid Adaptation for Functional Outputs of Compressible Flow Simulations.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[33] D. A. Venditti and D. L. Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–39, 2002.

[34] D. A. Venditti and D. L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, 2003.

[35] David P. Young, Robin G. Melvin, Michael B. Bieterman, Forrester T. Johnson, Satish S. Samant, and John E. Bussoletti. A higher-order boundary treatment for Cartesian-grid methods. *Journal of Computational Physics*, 92:1–66, 1991.

[36] O.C. Zienkiewicz and J. Z. Zhu. Adaptivity and mesh generation. *International Journal for Numerical Methods in Engineering*, 32:783–810, 1991.