A Triangular Cut-Cell Adaptive Method for High-Order Discretizations of the Compressible Navier-Stokes Equations

Krzysztof J. Fidkowski *, David L. Darmofal

Aerospace Computational Design Laboratory, Massachusetts Institute of Technology, Building 37, Room 401, USA

Abstract

This paper presents a mesh adaptation method for higher-order ($p > 1$) discontinuous Galerkin (DG) discretizations of the two-dimensional, compressible Navier-Stokes equations. A key feature of this method is a cut-cell meshing technique, in which the triangles are not required to conform to the boundary. This approach permits anisotropic adaptation without the difficulty of constructing meshes that conform to potentially complex geometries. A quadrature technique is proposed for accurately integrating on general cut cells. In addition, an output-based error estimator and adaptive method are presented, 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. Robustness of the cut-cell and adaptation technique is successfully tested for highly-anisotropic boundary-layer meshes representative of practical high Re simulations. Furthermore, 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 $p = 1$.

Key words: Triangular cut cells, Output-based error estimation, Anisotropic mesh adaptation, Discontinuous Galerkin, Compressible Navier-Stokes

1 Introduction

Computational Fluid Dynamics (CFD) has become an indispensible tool in analysis and design applications. In many cases, however, CFD is still plagued

* Corresponding author.
Email address: kfid@mit.edu (Krzysztof J. Fidkowski).

by insufficient automation and robustness in the geometry-to-solution process. Meshes are often constructed and adapted manually, or at least with significant user input; solvers do not always converge to a solution; estimates of the discretization error are rarely available, much less an indication of how the error can be decreased. In this paper, 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 [1–3] is an example of a cut-cell approach in which elements consist of squares/cubes on a regular lattice. While computationally fast and memory-lean, the Cartesian method becomes inefficient for the compressible Navier-Stokes equations, in which boundary layer and wake features demand mesh anisotropy for practical cases. A triangular/tetrahedral cut-cell approach relieves this inefficiency by allowing anisotropic adaptation in general directions. Second, output-based, anisotropic adaptation improves the automation and robustness of the solution method. Specifically, output error estimates provide the user with a measure of solution quality. These estimates are coupled with anisotropy detection that is geared for high-order discretizations, yielding an automated and efficient goal-oriented solution method. Together, triangular cut-cell mesh generation and output-based, anisotropic adaptation are the principal contributions of this work. They are tied together because cut-cell meshing becomes truly advantageous for highly-anisotropic meshes, which are most reliably generated by an automated adaptive method. The adaptive method also provides a common framework for comparing cut-cell meshes to boundary-conforming meshes.

While the combination of triangular cut cells and output-based anisotropic adaptation can be applied to any discretization, the focus of this paper 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 paper is as follows. First, for completeness, the DG discretization of the compressible Navier-Stokes equations is given in Section 2. Next, Section 3 presents the triangular cut-cell method. Sections 4 and 5 describe the output-based error estimator and the anisotropic adaptation strategy. Lastly, results from sample cases are given in Section 6, 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.

2 Compressible Navier-Stokes Discretization

The compressible Navier-Stokes system consists of K equations, where $K = 4$ for laminar flow in two dimensions. The k^{th} equation, written using index notation, reads

$$\partial_i F_{ki}(\mathbf{u}) - \partial_i F^v_{ki}(\mathbf{u}) = 0, \tag{1}$$

where i indexes the spatial dimension, and $\mathbf{u}$ is the state vector with K components. $F_{ki}(\mathbf{u})$ and $F^v_{ki}(\mathbf{u})$ are inviscid and viscous flux components, respectively. They are non-linear functions of the state vector components. It is convenient to make use of the linear dependence of F^v_{ki} on the spatial gradients $\partial_j u_l$ by writing

$$F^v_{ki} = A_{kilj}\partial_j u_l, \tag{2}$$

where A_{kilj} is a tensor that is a nonlinear function of the state vector components. Here, j indexes the spatial dimension and l indexes the state vector.

The discretization of (1) proceeds in standard finite-element fashion by triangulating the computational domain, Ω, into elements κ and searching for a solution, $\mathbf{u}_H$, in a finite-dimensional space, $\mathcal{V}_H$, for which a weak form of (1) is satisfied. In this work, $\mathcal{V}_H = [V^p_H]^K$; that is, each component of $\mathbf{u}_H$ resides in V^p_H, the space of piecewise polynomials of order p over the elements. For clarity, the weak form is presented here for one element κ with boundary $\partial\kappa$. The discrete semi-linear form, $\mathcal{R}_H(\mathbf{u}_H, \mathbf{v}_H)$, follows by summing over all elements,

$$\mathcal{R}_H\Big(\mathbf{u}_H, \mathbf{v}_H\Big) = \sum_\kappa \left(\mathbb{E}_\kappa(\mathbf{u}_H, \mathbf{v}_H) + \mathbb{V}_\kappa(\mathbf{u}_H, \mathbf{v}_H)\right) = 0,$$

where $\mathbb{E}_\kappa(\mathbf{u}_H, \mathbf{v}_H)$ is the contribution of the inviscid flux, $\mathbb{V}_\kappa(\mathbf{u}_H, \mathbf{v}_H)$ is the contribution of the viscous flux, and $\mathbf{v}_H \in \mathcal{V}_H$ denotes an arbitrary test function. In the equations that follow, v_k refers to components of $\mathbf{v}_H$, and u_k refers to components of $\mathbf{u}_H$. On $\partial\kappa$, the notation $()^+$ and $()^-$ refers to quantities taken from the interior and exterior of κ, respectively. Of particular relevance to the cut-cell algorithm is the fact that construction of the residual requires element-interior area integrals in addition to element-boundary integrals.

First, $\mathbb{E}_\kappa(\mathbf{u}_H, \mathbf{v}_H)$ can be written as

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

where n_i is the outward pointing normal, and $\widehat{F}_{ki}$ is an approximate characteristic-based flux function (Roe-averaged flux [4] in this work). Boundary conditions are imposed by setting $\widehat{F}_{ki}$ appropriately when $\partial\kappa$ is on $\partial\Omega$ [5].

The viscous flux term contribution is discretized using the second form of Bassi & Rebay (BR2) [6]. 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,

$$\begin{aligned} \mathbb{V}_\kappa(\mathbf{u}_H, \mathbf{v}_H) = &\int_\kappa \partial_i v_k A_{kilj}\partial_j u_l dx - \int_{\partial\kappa} \partial_i v_k^+ \Big(A_{kilj}^+ u_l^+ - \widehat{A_{kilj}u_l} \Big) n_j ds \\ &- \int_{\partial\kappa} v_k^+ \widehat{Q}_{ki} n_i ds, \end{aligned}$$

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 [7]. 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	$\{A_{kilj}\partial_j u_l\} - \eta^f\{\delta_{ki}^f\}$	$A_{kilj}^+\{u_l\}$
Boundary, Dirichlet	$A_{kilj}^+\partial_j u_l^+ - \eta^{bf}\delta_{ki}^{bf}$	$A_{kilj}^b u_l^b$
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 constant stability factors set to 3 and 3/2, respectively. $\delta_{ki}^f, \delta_{ki}^{bf} \in [V_H^p]^2$ are auxiliary variables for interior and boundary faces, respectively, that satisfy, $\forall \tau_{ki} \in [V_H^p]^2$,

$$\begin{aligned} \int_{\kappa^+} \delta_{ki}^{f+}\tau_{ki} dx + \int_{\kappa^-} \delta_{ki}^{f-}\tau_{ki} dx &= \int_{\sigma^f} \{\tau_{ki}A_{kilj}\}\Big(u_l^+ - u_l^-\Big) n_j ds, \\ \int_\kappa \delta_{ki}^{bf}\tau_{ki} dx &= \int_{\sigma^{bf}} \tau_{ki}A_{kilj}^b\Big(u_l^+ - u_l^b\Big) n_j ds, \end{aligned}$$

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.

3 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 cell shapes at the boundary. The idea of using cut cells began with the works of Purvis and Burkhalter [8], 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* [9] and Gaffney *et al* [10], 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 [11] 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 [12] 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 [1] 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 [13] 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 [3], 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 [14]. Ongoing work continues in computing adjoints and shape sensitivities [15] and in novel ideas for moving beyond Euler calculations [16].

This paper explores the feasibility of using triangular cut-cell meshes in a discontinuous Galerkin finite element framework. The motivation for this approach is to improve meshing robustness, to automate mesh generation for complex geometries, and to allow for anisotropic meshes. For DG 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.

3.1 Geometry Definition and Initial Mesh

For this proof-of-concept study in 2D, a geometry definition consisting of cubic-splined points is sufficient. Geometric corners, where the tangent vector is discontinuous, can be represented using 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 requested, subsequent geometry-adapted triangulations are constructed by refining elements that intersect the splines. The details of geometry adaptation are not crucial, as only a reasonable starting mesh is sought for the solution-adaptive method.

3.2 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 to determine intersections of spline segments with element edges and nodes. Careful attention must be given to conditioning for node and tangency intersections. The intersections are performed once and stored for the entire mesh, to prevent floating-point discrepancies from repeating calculations. 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.

The orientation of the splines is used to determine the direction of validity of each cut, where a valid direction is one that points into the computational domain. This step is also performed only once, as it requires floating-point calculations. Based on the validity directions, new cut-cell edges are constructed from intersected edges of the original mesh. Connectivity information in the form of spline-edge intersections, mesh nodes, and spline knots is used to stitch together the cut edges and embedded faces into loops that enclose disjoint cut regions. These disjoint regions represent the newly-formed cut cells. Note that cut cells may have a nearly arbitrary number of faces and neighbors, depending on the geometry of the cuts. Figure 1 shows the cut cells formed near a trailing edge of an airfoil. The triangle at the apex of the trailing edge becomes a cut cell with four neighboring cells, while the adjacent triangle straddling the airfoil is split into two cut cells.

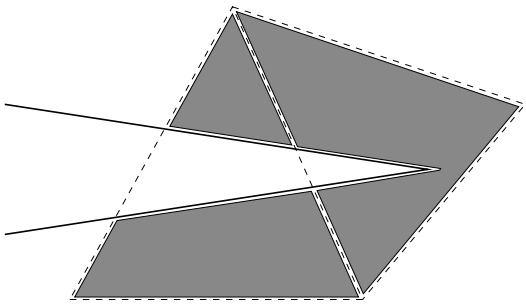


Fig. 1. Shaded areas illustrate cut cells formed at an airfoil trailing edge.

During the creation of cut cells and edges, adjacent nodes are marked as either inside or outside the computational domain. This information is propagated to all other nodes by traversing the non-cut edges and triangles. Triangles contained completely within the geometry, and hence outside the computational domain, are identified according to the status of their adjacent nodes. These triangles (if any exist) are eliminated from the computational data structure such that no finite element calculations are performed on these triangles.

3.3 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. Currently, each spline segment of an embedded face is mapped to a reference interval separately. This splitting at spline knots, where the geometry in general contains second-order discontinuities, leads to more accurate integration; specifically, exact quadrature for constant integrands. While useful in development, this splitting can be avoided in practice in order to reduce the computational costs of embedded boundary integrations.

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

$$G_{\mathbf{i}k}(\mathbf{x}) = x_k \Phi_{\mathbf{i}}(\mathbf{x}), \quad \Phi_{\mathbf{i}}(\mathbf{x}) = \prod_k \phi_{i_k}(x_k), \quad \mathbf{x} = [x_k], \quad \mathbf{i} = [i_k], \tag{3}$$

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), as shown in Figure 2. The order of these functions is the

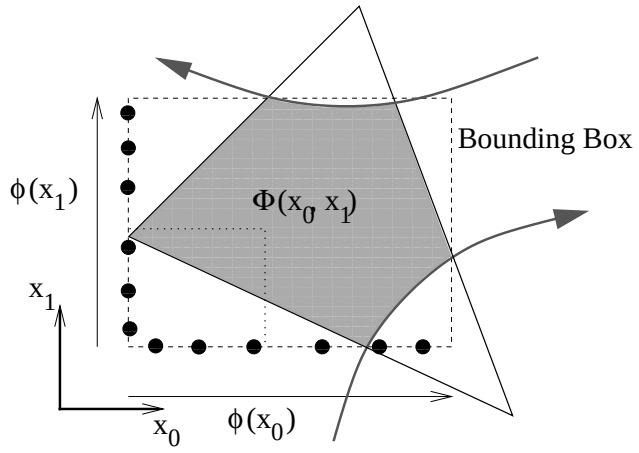


Fig. 2. The $\Phi_{\mathbf{i}}(\mathbf{x})$ are tensor products of one-dimensional Lagrange functions in each direction.

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; for compressible Navier-Stokes, it is $2p+1$. The factors of x_k in the definition of $G_{\mathbf{i}}$ ensure that $\zeta_{\mathbf{i}} = \partial_k G_{\mathbf{i}k} = d\Phi_{\mathbf{i}}(\mathbf{x}) + x_k \partial_k \Phi_{\mathbf{i}}(\mathbf{x})$ span the same complete space as the tensor product functions $\Phi_{\mathbf{i}}$ [17].

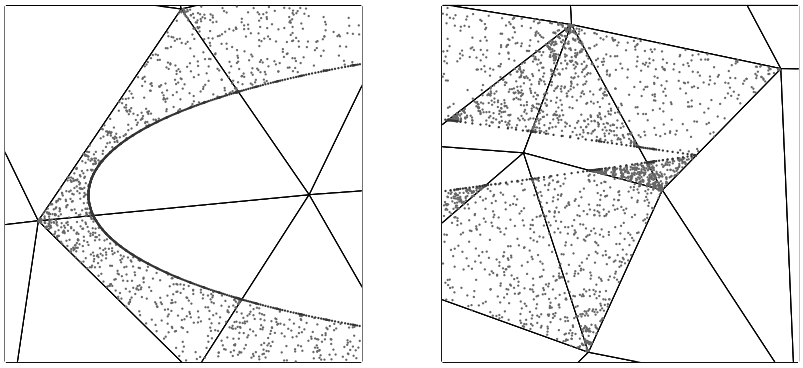
The projection of $f(\mathbf{x})$ onto $\zeta_{\mathbf{i}}(\mathbf{x})$ is performed by minimizing the least-squares error,

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

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,

$$w_q = Q_{q\mathbf{j}}(R^{-T})_{\mathbf{ji}} \int_\kappa \zeta_{\mathbf{i}}(\mathbf{x}) d\mathbf{x}, \quad \text{where} \quad \zeta_{\mathbf{i}}(\mathbf{x}_q) = Q_{q\mathbf{j}} R_{\mathbf{ji}}. \tag{4}$$

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, the points are chosen randomly, 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 $\pm 15^o$). 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. Example sampling points for a cut airfoil are shown in Figure 3.



(a) Leading Edge (b) Trailing Edge

Fig. 3. Example of sampling points on a cut NACA 0012 mesh.

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 non-axis-aligned sliver elements, as shown in Figure 4. In this case, conditioning is improved by rotating the bounding box for a tighter fit around the element. Specifically, for each element, two new bounding boxes are constructed, oriented along the diagonals of the original axis-aligned bounding box. The tightest-fitting bounding box, that is, the one with the smallest area, is used.

Better algorithms likely exist for performing the cut-cell integrations, or for improving the proposed method. For example, the sampling point selection process can be made more sophisticated. Random selection was used here for its simplicity, justified by the fact that unfavorable clusters are detected by

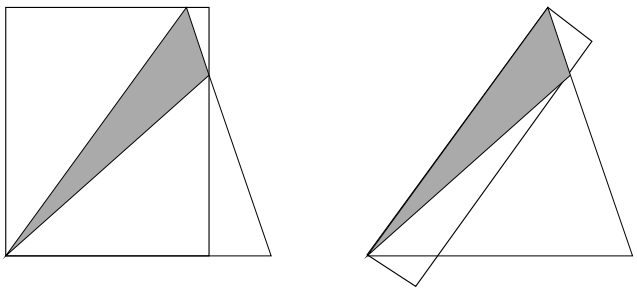


Fig. 4. For non-axis-aligned sliver cut elements (shaded area), the original bounding box (left) is rotated to obtain a tighter fit (right). Also shown is the original triangle from which the sliver element was cut.

degeneracy in the weight calculation, in which case the selection process is repeated. An improved selection process may be more expensive, but it may allow for fewer sampling points. This is an area of possible future research.

3.4 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: $\Delta t_\kappa = \mathrm{CFL}(h_\kappa / s_{\kappa,\max})$, where h_κ and $s_{\kappa,\max}$ are the element-specific size and maximum wave speed, respectively.

4 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.

Output-based error estimation and adaptation for CFD have been studied extensively in the literature [7, 18–24]. 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. $\boldsymbol{\psi}$ is assumed to exist in the same space as $\mathbf{u}$. Assuming $\mathcal{R}_H(\mathbf{u}, \mathbf{w}) = 0$, $\forall \mathbf{w} \in \mathcal{W}_H$, the output error can be expressed as

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) = -\mathcal{R}_H(\mathbf{u}_H, \boldsymbol{\psi} - \boldsymbol{\psi}_H), \tag{5}$$

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

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

the output error can also be expressed as

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) = -\bar{\mathcal{R}}^{\psi}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \boldsymbol{\psi}_H). \tag{6}$$

As $\mathbf{u}$ and $\boldsymbol{\psi}$ are in general not known, two approximations are employed to make the above output error estimates practical. First, the exact mean-value linearizations are replaced by approximate linearizations about $\mathbf{u}_H$. To minimize errors in (5) and (6) 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 $\mathcal{R}_H^{\psi}(\mathbf{u}_H; \mathbf{v}_H, \boldsymbol{\psi}_H) = 0,\ \forall \mathbf{v}_H \in \mathcal{V}_H$, where $\mathcal{R}_H^{\psi}$ is the adjoint residual computed with linearization only about $\mathbf{u}_H$. Second, the exact solution errors $\mathbf{u} - \mathbf{u}_H$ and $\boldsymbol{\psi} - \boldsymbol{\psi}_H$ are replaced by $\mathbf{u}_h - \mathbf{u}_H$ and $\boldsymbol{\psi}_h - \boldsymbol{\psi}_H$, respectively, where $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$ are approximations to $\mathbf{u}$ and $\boldsymbol{\psi}$ on an enriched finite element space, $\mathcal{V}_h$. Following the work of Lu [7], $\mathcal{V}_h$ is constructed from $\mathcal{V}_H$ by increasing the interpolation order to $p+1$.

The approximations $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$ are created by a reconstruction process on $\mathcal{V}_h$. In this work, local H_1 patch reconstruction is used, in which the minimized error for 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 reconstructed solution, $\mathbf{u}_h$, is 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, one element-Jacobi smoothing iteration 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 (5) and (6) yields the following approximations to the output error (making use of $\mathcal{V}_H \subset \mathcal{V}_h$ and $T_h = T_H$):

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

In the above expressions, $|_\kappa$ refers to restriction to element κ. A local error indicator on each element is obtained by averaging primal-residual and adjoint-residual contributions to the output error in the above expressions. Specifically, in this work, the error indicator in each element κ is taken to be

$$\epsilon_\kappa = \frac{1}{2}\Big(\big| \mathcal{R}_h(\mathbf{u}_H, (\boldsymbol{\psi}_h - \boldsymbol{\psi}_H)|_\kappa) \big| + \big| \mathcal{R}_h^{\psi}(\mathbf{u}_H; (\mathbf{u}_h - \mathbf{u}_H)|_\kappa, \boldsymbol{\psi}_H) \big| \Big). \tag{7}$$

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 [19, 25]. 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.

5 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 mesh 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 Süli [26] 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.

5.1 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 [27–29],

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

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 [29]. 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 $h_i/h_j = (|\lambda_j|/|\lambda_i|)^{1/2}$.

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. 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, and $\mathbf{e}_1$ the direction of maximum $p+1$st derivative in the plane orthogonal to $\mathbf{e}_0$. 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,

$$\frac{h_i}{h_j} = \left(u_{\mathbf{e}_j}^{(p+1)} / u_{\mathbf{e}_i}^{(p+1)} \right)^{1/(p+1)}, \tag{8}$$

where $u_{\mathbf{e}_i}^{(p+1)}$ is the $p+1$st derivative in the direction $\mathbf{e}_i$. (8) provides only the relative mesh sizing; the absolute values for h_i are based on the error indicator, as described in the following section.

5.2 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 [22, 26, 30], 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 [31] 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 [24, 25], 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* [19, 32], 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 [33]. 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 [19].

Anisotropy detection introduces another variable into the mesh optimization process; namely the stretching of the elements. In "pure" Hessian-based adaptation, the absolute magnitude of stretching is controlled by an arbitrary global scaling factor [29]. Venditti and Darmofal use an output-based error indicator

to determine the length magnitude, leading to a more robust adaptation process [24]. Formaggia *et al* [34] have combined Hessian-based interpolation error estimates with output-based *a posteriori* error analysis to arrive at output-based anisotropic error estimates.

In the interest of generality, this work adopts a variation of the refinement prediction method of Zienkiewicz and Zhu, modified to allow for mesh anisotropy. One drawback of straightforward refinement prediction 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 mesh will converge, by attempting to equidistribute the error on the predicted mesh, adaptive convergence can be accelerated [17].

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, where again i indexes the spatial dimensions, n_κ can be approximated as,

$$n_\kappa = \prod_i \left(h_i^c / h_i\right). \tag{9}$$

The current sizes h_i^c are calculated as the singular values of the mapping from a unit equilateral triangle to element κ. The resulting grid-implied metric is similar to that used by Venditti [35]. Such a calculation ensures that an isotropic metric is retained for a mesh of equilateral triangles. (9) is based on an approximate volume comparison between the current and refined elements and, therefore, does not depend on the principal directions associated with h_i^c and h_i.

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,

$$\frac{\epsilon_\kappa}{\epsilon_\kappa^c} = \left(\frac{h_0}{h_0^c}\right)^{\bar{p}_\kappa + 1}, \tag{10}$$

where ϵ_κ^c is the current error indicator, ϵ_κ is the expected error indicator, $\bar{p}_\kappa = \min(p_\kappa, \gamma_\kappa)$, and γ_κ is the lowest order of any singularity within κ [31]. γ_κ is generally set to p_κ, except on cut cells that contain geometric singularities, such as corners or trailing edges. On these cells, γ_κ is lowered (to 0 in

practice), resulting in isolation of geometric singularities with fewer adaptation iterations. The *a priori* estimate is valid for many common engineering outputs, including forces and pressure distribution norms. In the estimate, the error is assumed to scale with h_0, which corresponds to the direction of maximum $p+1$st derivative. Implicit in the estimate is that the principal directions corresponding to the requested size, h_0, and the current size, h_0^c, align. One option for accounting for a difference in principal directions is to replace h_0^c in (10) with $h^c(\mathbf{e}_0)$, the current, grid-implied size in the principal direction $\mathbf{e}_0$. However, as $h_0^c \leq h^c(\mathbf{e})$ for any direction, $\mathbf{e}$, using h_0^c in (10) leads to a more conservative estimate for h_0 in the early stages of adaptation. Furthermore, the assumption that h_0 and h_0^c align becomes more valid as the adaptation progresses. Equating the allowable error with the expected error from the *a priori* estimate yields

$$\underbrace{n_\kappa \frac{e_0}{N_f}}_{\text{allowable error}} = \underbrace{\epsilon_\kappa^c \left(\frac{h_0}{h_0^c}\right)^{\bar{p}_\kappa+1}}_{\textit{a priori}\text{ estimate}} . \tag{11}$$

Expressing $\frac{h_0}{h_0^c}$ in terms of n_κ and the known relative sizes (8) yields a relation between n_κ and N_f. For example, in two dimensions,

$$n_\kappa \frac{e_0}{N_f} = \epsilon_\kappa^c \left[\frac{1}{n_\kappa}\frac{h_0}{h_1}\frac{h_1^c}{h_0^c}\right]^{(\bar{p}_\kappa+1)/2} \quad \Rightarrow \quad n_\kappa^{(\bar{p}_\kappa+3)/2} = \frac{\epsilon_\kappa^c}{e_0/N_f}\left[\frac{h_0}{h_1}\frac{h_1^c}{h_0^c}\right]^{(\bar{p}_\kappa+1)/2} .$$

Substituting for n_κ 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, (11) yields n_κ, from which the h_i are calculated using (9) and (8). Adaptation iterations stop when $\epsilon \equiv \sum_\kappa \epsilon_\kappa \leq e_0$, 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, $0 < \eta_t \leq 1$, and the adaptation aggressiveness, $0 \leq \eta_a < 1$. Specifically, instead of e_0 in (11), 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 $\eta_a = 0.25$.

5.3 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) [36], 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. Metric definition on triangles completely contained within the geometry, which do not possess a solution, is performed by using the grid-implied metric on the background mesh. 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 by 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, detected by testing for non-physical states at quadrature points, a $p = 0$ restriction of the solution from the previous mesh is performed.

6 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 in terms of degrees of freedom (DOF). The number of degrees of freedom in a solution is computed as the total number of unknowns, excluding the equation-specific multiplier (e.g. 4 for 2D Euler or Navier Stokes).

For comparing different interpolation orders, p, a better metric than degrees of freedom is computational time. However, relative run times depend heavily on the implementation and the hardware. Nevertheless, a more accurate estimate of the computational work is possible by assuming a work expression of the form, $W \sim N_e(n(p))^a$, where N_e is the number of elements, $n(p) = (p+1)(p+2)/2$ is the degrees of freedom per element, and a is a measure of the computational complexity per element. Using $DOF = N_e n(p)$, the work estimate may be written as $W \sim DOF(n(p))^{a-1}$. From experience, at least for orders up to $p = 3$, the work is dominated by the matrix-vector products during the GMRES linear solve; hence, $a \sim 2$. As $n(1) = 3$, $n(2) = 6$, and $n(3) = 10$, for the same DOF, $p = 2$ is expected to be twice as expensive as $p = 1$, while $p = 3$ is expected to be over three times more expensive than $p = 1$.

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

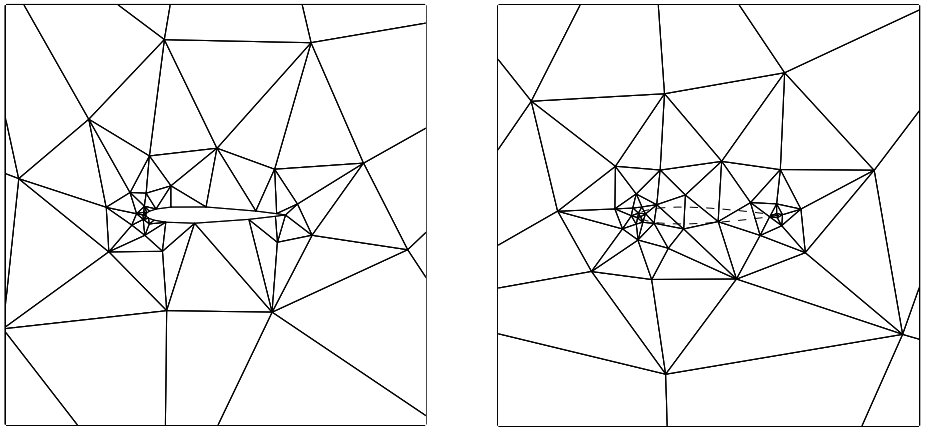
The computational domain for this case consists of a NACA 0012 airfoil contained within a farfield box, a distance of 100 chord lengths away from the airfoil. The NACA geometry is modified to close the trailing edge gap,

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

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 5 shows the initial 123-element boundary-conforming mesh, as well as the initial 124-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 at every adaptation iteration, since BAMG produces linear meshes. For the isotropic elements in this case, this curving does not pose a problem.

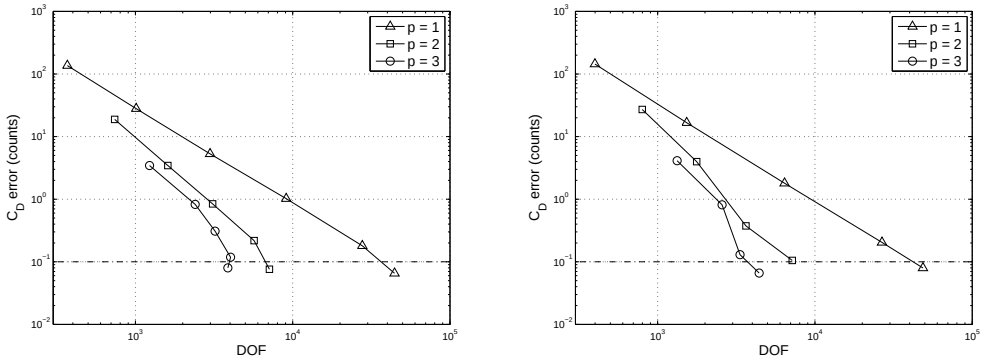
Figure 6 summarizes the results of the adaptation runs. The plots show the output error versus DOF at each adaptation iteration for every run. The horizontal dashed line marks the error tolerance of 0.1 drag counts. In both the boundary-conforming and cut-cell methods, the $p = 3$ runs achieve the desired accuracy with the fewest degrees of freedom. The advantage is roughly a factor of 2 over $p = 2$ and a factor of 10 over $p = 1$. In terms of the computa-



(a) Boundary-conforming: 123 elements

(b) Cut-cell: 124 elements

Fig. 5. Initial NACA 00012 meshes.



(a) Boundary-conforming

(b) Cut-cell

Fig. 6. Drag error vs. degrees of freedom for the inviscid NACA 0012 runs.

tional work estimate discussed at the beginning of this section, the differences are diminished, but $p = 3$ is still slightly less expensive than $p = 2$, which is almost three times less expensive than $p = 1$. The convergence of the cut-cell runs is comparable to that of the boundary-conforming runs.

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

In this case, a 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 isotropic, geometry-adapted, with roughly 250 elements.

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 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.

6.2.1 Drag Adaptation

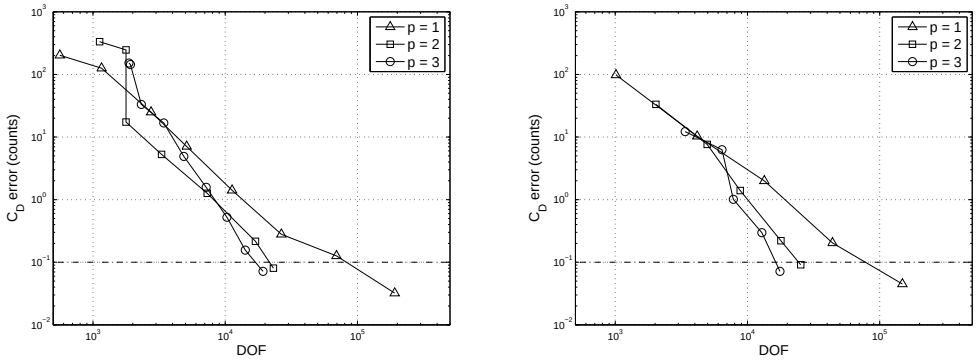
The adaptation algorithm was tested using drag as the output, with tolerance of 0.1 counts. A force output for a viscous simulation consists of two components: a pressure force and a viscous force, $\mathbf{f}^v$. The viscous force is obtained from the viscous flux, F^v_{ki}, with a dual-consistent correction,

$$\begin{bmatrix} \mathbf{f}^v_x \\ \mathbf{f}^v_y \end{bmatrix} = \sum_{\sigma^{bf} \in \Gamma_{\text{out}}} \int_{\sigma^{bf}} \begin{bmatrix} -F^v_{2i} n_i + \eta^{bf} \delta^{bf}_{2i} n_i \\ -F^v_{3i} n_i + \eta^{bf} \delta^{bf}_{3i} n_i \end{bmatrix} ds \tag{12}$$

where F^v_{2i} and F^v_{3i} are viscous flux component, and δ^{bf}_{ki} is the auxiliary variable presented in Section 2. Not including this correction leads to an adjoint solution that is not well-behaved at the airfoil boundary, Γ_{out} [7].

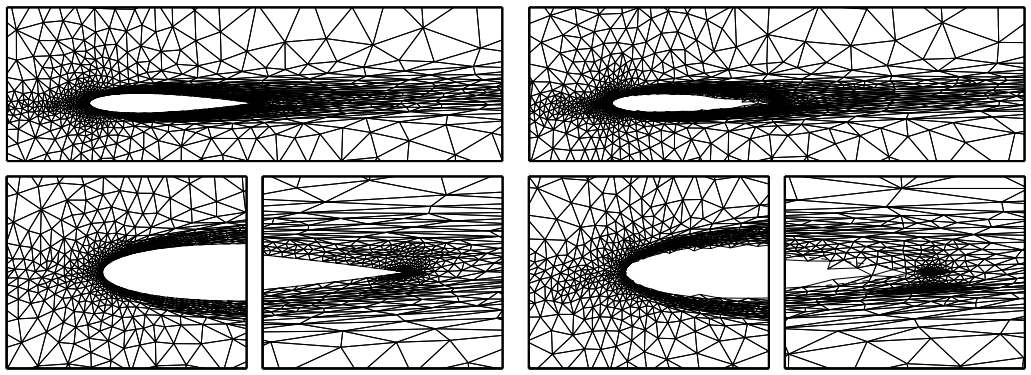
The "true" drag of 568.84 counts was computed on a $p = 3$ cut-cell mesh, adapted to an error of 10^{-3} counts. The boundary-conforming and cut-cell runs converge to the same drag value. The corresponding drag error convergence histories are plotted in Figure 7. Overall, the cut-cell and boundary-conforming results are similar. For both the boundary-conforming and the cut-cell cases, $p = 3$ requires only slightly fewer degrees of freedom than $p = 2$ at the error tolerance. Thus, in terms of estimated work, $p = 2$ becomes slightly advantageous to $p = 3$ in this case. $p = 1$, however, remains the most expensive, requiring a factor of 4 more degrees of freedom than $p = 2$, which translates to an estimated work increase of about a factor of 2.

Figures 8 and 9 show the final adapted meshes for $p = 2$ and $p = 3$. The final adapted meshes for $p = 1$ are much finer: 63751 elements for the boundary-conforming case, and 50515 for the cut-cell case. They are not shown here because the elements are practically indiscernible on the scale used. In all meshes, areas of high refinement include the boundary layer, a large extent of the wake, and, to a lesser extent, the flow in front of the airfoil. In the cut-cell meshes, the airfoil boundary location is marked by a dashed line. Triangles completely contained within the airfoil are not shown. The similarity



(a) Boundary-conforming (b) Cut-cell

Fig. 7. Drag error vs. degrees of freedom for the viscous NACA 0012 runs. Dashed line indicates prescribed tolerance of 0.1 drag counts.



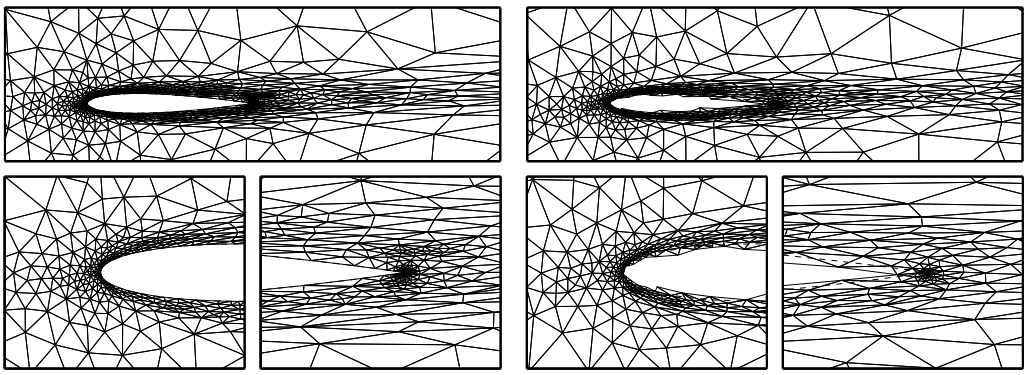
(a) Boundary-conforming: 3852 elements (b) Cut-cell: 4399 elements

Fig. 8. Final $p = 2$ meshes adapted on drag.

in element sizes between the cut-cell meshes and their boundary-conforming counterparts is evident.

6.2.2 Sensitivity to Initial Mesh

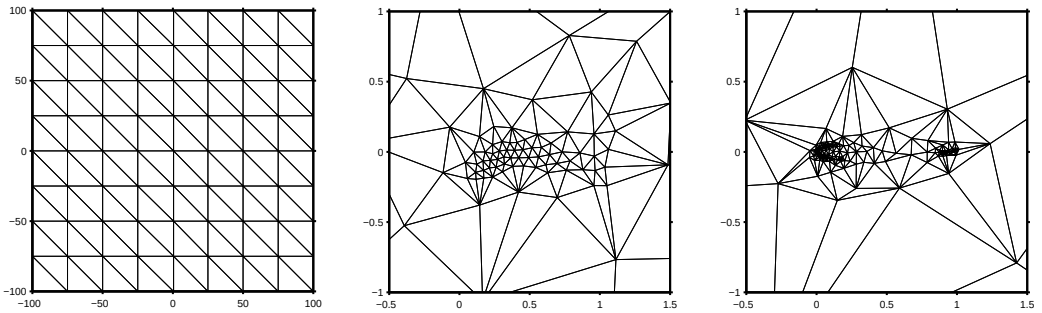
For the adaptation method to be practical, the final adapted meshes should not be highly sensitive to the starting meshes. The sensitivity is tested for a drag-adapted viscous NACA 0012, with an error tolerance of 1 drag count. Runs are performed with several different cut-cell starting meshes, including a set of uniform triangulations of the entire domain, as well as two meshes adapted on geometry to different levels of fineness (see Figure 10).



(a) Boundary-conforming: 1929 elements

(b) Cut-cell: 1840 elements

Fig. 9. Final $p = 3$ meshes adapted on drag.



(a) Uniform 8x8

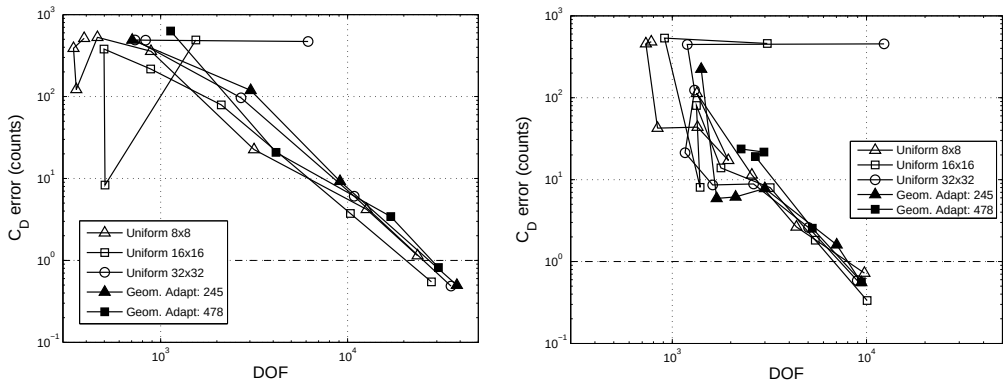
(b) Adapted on geometry: 245 elements

(c) Adapted on geometry: 478 elements

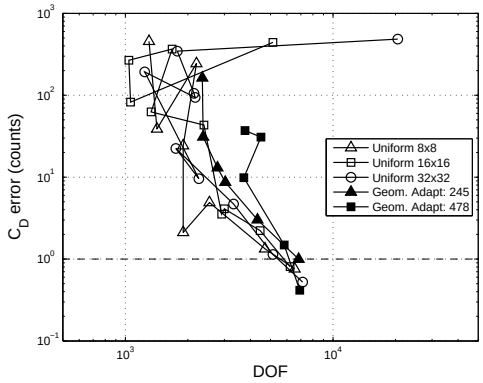
Fig. 10. Initial meshes for sensitivity study.

Figure 11 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 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.



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



(c) $p = 3$

Fig. 11. Adaptation histories for $p = 1, 2, 3$, starting from the various initial meshes shown in Figure 10.

6.3 High Peclet Number Flow Over a Joukowski Airfoil

One of the proposed advantages of the cut-cell method over the boundary-conforming method lies in the robustness of cut-cells in dealing with anisotropic meshes near curved boundaries. For very highly anisotropic meshes, however, the cut-cell method also faces a robustness challenge stemming primarily from cutting a mesh whose edges in the boundary layer are nearly parallel to the geometry. The results in this section address this issue by applying cut cells to boundary-layer cases representative of practical high Reynolds number simulations.

An ideal test of the cut-cell adaptive method would be a Reynolds-averaged Navier-Stokes (RANS) simulation with a derivative quantity, such as heat transfer, as the output of interest. However, as RANS discretization and so-

lution is currently under development, a simpler equation set, convection-diffusion, was chosen to assess the robustness of the cut-cell adaptive method. The convection-diffusion equation is given by,

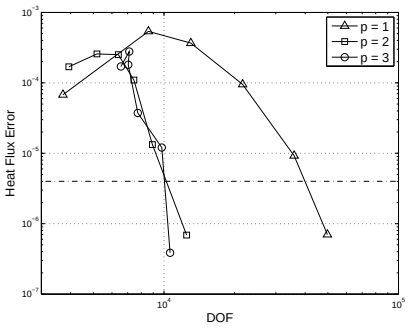
$$\nabla \cdot (\mathbf{V}u) - \nabla \cdot (\nu \nabla u) = 0, \qquad Pe = \frac{V_\infty L}{\nu}, \tag{13}$$

where u is the unknown scalar concentration of interest, $\mathbf{V}$ is a prescribed velocity field, ν is the diffusion coefficient, V_∞ is a constant farfield velocity, L is a reference length scale, and Pe is the Peclet number, measuring the strength of convection relative to diffusion.

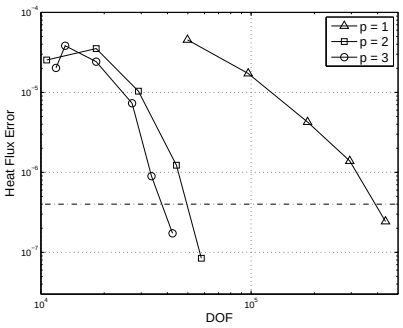
The discontinuous Galerkin discretization of (13) proceeds similarly to that of the Navier-Stokes equations, using full up-winding for the convection term and BR2 for the diffusion term. Boundary-layer behavior can be observed at high Pe when a boundary condition specifies a concentration of u different from that in the bulk flow. To be concrete, in the following examples u will represent a temperature field, and the case of interest will be a heated airfoil in a high-speed flow. Standard potential flow around a Joukowski airfoil is used for $\mathbf{V}$. The output of interest for adaptation is the total heat flux out of the airfoil, with an error tolerance of approximately 1% of the true heat flux.

Adaptive runs were performed for $p = 1, 2, 3$ at three Peclet numbers: $Pe = 4 \times 10^6$, $Pe = 4 \times 10^8$, and $Pe = 4 \times 10^{10}$. The largest value, $Pe = 4 \times 10^{10}$, approximately simulates the thickness of the viscous sublayer ($y^+ = 10$) in a turbulent Navier-Stokes computation at $Re = 10^7$. A geometry-adapted mesh of roughly 900 elements served as the initial mesh for the $Pe = 4 \times 10^6$ runs. Thereafter, the final adapted meshes at each Pe served as initial meshes for the next Pe. Such staggering ensured reasonable accuracy of the error estimate on the initial meshes. The "true" values for the heat flux in each case were computed using a $p = 3$ solution on a uniformly refined version of the final adapted $p = 2$ mesh.

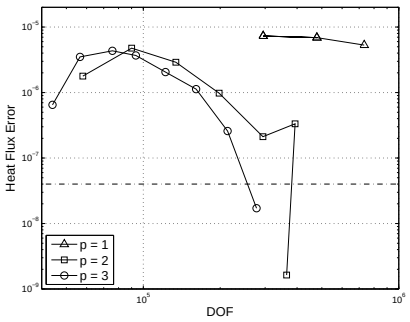
Figure 12 shows the adaptation histories for all three Peclet numbers. For $Pe = 4 \times 10^6$, at the error tolerance, $p = 2$ and $p = 3$ require about the same number of DOF, while $p = 1$ requires four times more. In terms of approximate work, $p = 1$ is still the most expensive, while $p = 2$ is the least expensive by a factor of two. For $Pe = 4 \times 10^8$, the difference in DOF grows, with $p = 1$ now requiring almost an order of magnitude more degrees of freedom. Finally, for $Pe = 4 \times 10^{10}$, only three adaptation iterations are shown for $p = 1$ because the mesher, BAMG, failed to return a valid mesh after the third adaptation iteration. Specifically, the returned mesh contained triangles with areas that were negative. This failure is likely due to areas of very high anisotropy requested in the $p = 1$ adaptive run. On the other hand, both $p = 2$ and $p = 3$ converged successfully to satisfy the error tolerance.



(a) $Pe = 4 \times 10^6$



(b) $Pe = 4 \times 10^8$



(c) $Pe = 4 \times 10^{10}$

Fig. 12. Output error vs. DOF adaptation history.

The $p = 2$ convergence history exhibits a slight spike, which is likely caused by a meshing irregularity or the resolution of a previously under-resolved area. Finally, while the error convergence slopes are steep for many of the runs, this steepness can be attributed to a large number of extra elements that are required to initially "uncover" a thin boundary layer; the large error drops in the latter adaptation iterations reflect the placing of the final elements within the boundary layer.

Close-ups of the final adapted meshes for each of the runs are shown in Figure 13. For all Pe, the coarseness in the boundary layer mesh allowed by $p > 1$ is clearly evident. A rough count of the average number of cells within the boundary layer in a direction normal to the airfoil boundary yields about 25-40 for the final adapted $p = 1$ meshes, 5-6 for the final adapted $p = 2$ meshes, and 2-3 for the final adapted $p = 3$ meshes. The fact that these meshes, with edges nearly parallel to the geometry, were successfully cut demonstrates the robustness of the cut-cell method for modeling the very thin boundary layers expected in practical simulations. Furthermore, attempting these cases with the boundary-conforming method produces invalid meshes in the early stages

of adaptation.

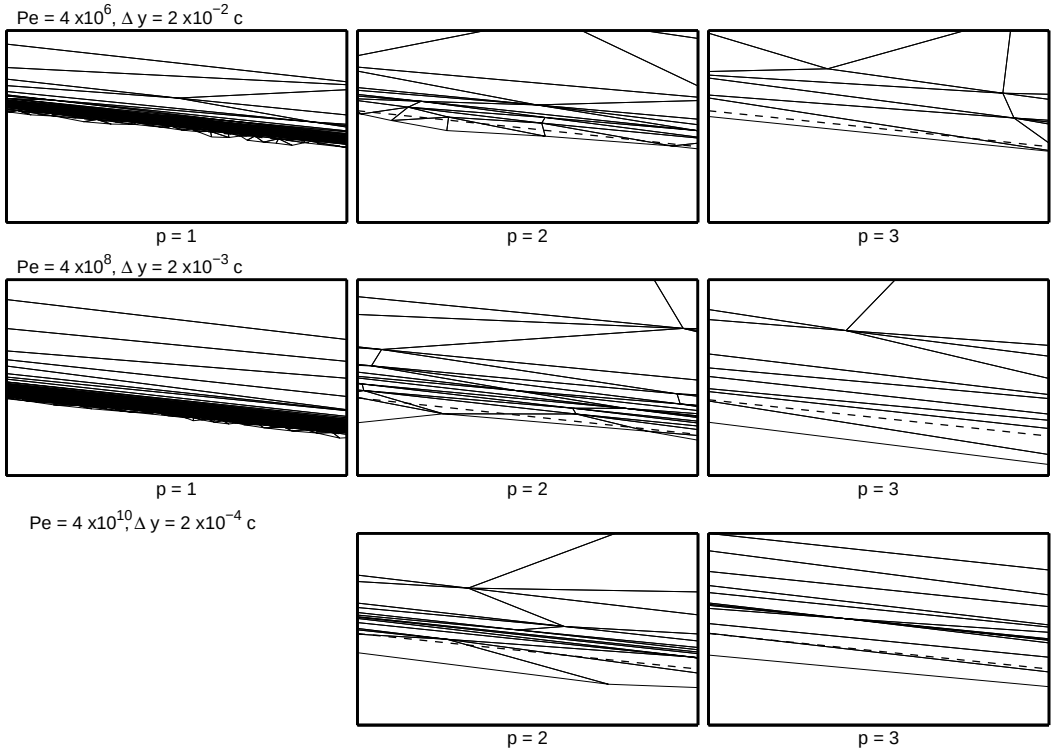


Fig. 13. Close-ups of the boundary-layer meshes for each of the runs. Dashed line indicates the airfoil boundary. Δy refers to the y-axis range in each of the plots, in terms of the airfoil chord length, c.

7 Conclusions

This paper presents a complete output-based mesh adaptation procedure for higher-order discontinuous Galerkin discretizations in two dimensions. 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 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.

In addition to the adaptation procedure, a triangular cut-cell meshing technique is introduced as an alternative to boundary-conforming meshing. The

cut-cell method is shown to produce adaptive results similar to those obtained with boundary conforming meshes. 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. Cut-cell meshes, on the other hand, remain robust even for practical boundary-layer simulations. These results support the concept of using triangular cut cells for automated, robust, and efficient meshing.

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, as presented in this paper, are also valid for three dimensions.

Regarding cut cells, the required three-dimensional intersection problem certainly becomes more difficult. One possible extension, currently under development, makes use of quadratic patches to represent the geometry. These patches can be created from standard triangular surface tesselations by inserting additional nodes at edge midpoints. The intersection problem between the patches and tetrahedra becomes one of intersecting conic sections in patch reference space. The two-dimensional integration method described in this work is used to calculate sampling points and weights for boundary and cut-face integrations. A straightforward extension of the method to three dimensions then yields volume sampling points and weights. While the cut-cell method becomes relatively more expensive in three dimensions, the idea of tetrahedral cut cells offers an alternative to the difficult problem of generating boundary-conforming meshes around intricate, curved, three-dimensional geometries. This extension to three dimensions is the subject of ongoing work.

8 Acknowledgements

The authors thank the Project X development team for the many contributions during the course of this work. K. Fidkowski's work was supported by the Department of Energy Computational Science Graduate Fellowship, under grant number DE-FG02-97ER25308.

References

[1] W. J. Coirier, K. G. Powell, Solution-adaptive cut-cell approach for viscous and inviscid flows, AIAA Journal 34 (5) (1996) 938–945.

[2] D. Calhoun, R. J. LeVeque, A Cartesian grid finite-volume method for

the advection-diffusion equation in irregular geometries, Journal of Computational Physics 157 (2000) 143–180.

[3] M. Aftosmis, M. Berger, J. Melton, Adaptive Cartesian mesh generation, in: J. Thompson, B. Soni, N. Weatherill (Eds.), Handbook of Grid Generation, CRC Press, 1998.

[4] P. L. Roe, Approximate Riemann solvers, parametric vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372.

[5] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, Journal of Computational Physics 207 (2005) 92–113.

[6] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, in: K. Cockburn, Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000, pp. 197–208.

[7] J. Lu, An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2005).

[8] J. W. Purvis, J. E. Burkhalter, Prediction of critical Mach number for store configurations, AIAA Journal 17 (11) (1979) 1170–1177.

[9] D. K. Clarke, M. D. Salas, H. A. Hassan, Euler calculations for multielement airfoils using Cartesian grids, AIAA Journal 24 (3) (1986) 353.

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

[11] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, J. E. Bussoletti, A higher-order boundary treatment for Cartesian-grid methods, Journal of Computational Physics 92 (1991) 1–66.

[12] M. J. Berger, R. J. Leveque, An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries, Paper 1989-1930, AIAA (1989).

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

[14] S. M. Murman, M. J. Aftosmis, S. E. Rogers, Characterization of space shuttle ascent debris aerodynamics using CFD methods, Paper 2005-1223, AIAA (2005).

[15] M. Nemec, M. Aftosmis, S. Murman, T. Pulliam, Adjoint formulation for an embedded-boundary Cartesian method, Paper 2005-0877, AIAA (2005).

[16] M. J. Aftosmis, M. J. Berger, J. J. Alonso, Applications of a Cartesian mesh boundary-layer approach for complex configurations, Paper 2006-0652, AIAA (2006).

[17] K. J. Fidkowski, D. L. Darmofal, Output-based adaptive meshing using triangular cut cells., M.I.T. Aerospace Computational Design Laboratory Report. ACDL TR-06-2 (2006).

[18] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247–264.
[19] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, in: A. Iserles (Ed.), Acta Numerica, Cambridge University Press, 2001.
[20] J. D. Müller, M. B. Giles, Solution adaptive mesh refinement using adjoint error analysis, Paper 2001-2550, AIAA (2001).
[21] M. B. Giles, E. Süli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, in: Acta Numerica, Vol. 11, 2002, pp. 145–236.
[22] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, Journal of Computational Physics 183 (2) (2002) 508–532.
[23] T. Barth, M. Larson, A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes, in: R. Herban, D. Kröner (Eds.), Finite Volumes for Complex Applications III, Hermes Penton, London, 2002.
[24] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22–46.
[25] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: application to two-dimensional inviscid flows, Journal of Computational Physics 176 (1) (2002) 40–39.
[26] P. Houston, E. Süli, A note on the design of hp-adaptive finite element methods for elliptic partial differential equations, Comput. Methods Appl. Mech. Engrg 194 (2005) 229–243.
[27] J. Peraire, M. Vahdati, K. Morgan, O. C. Zienkiewicz, Adaptive remeshing for compressible flow computations, Journal of Computational Physics. 72 (1987) 449–466.
[28] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, M.-G. Vallet, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles, Int. J. Numer. Meth. Fluids 32 (2000) 725–744.
[29] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaptation for flow simulations, International Journal for Numerical Methods in Fluids 25 (1997) 475–491.
[30] P. Solin, L. Demkowicz, Goal-oriented hp-adaptivity for elliptic problems, Comput. Methods Appl. Mech. Engrg. 193 (2004) 449–468.
[31] O. Zienkiewicz, J. Z. Zhu, Adaptivity and mesh generation, International Journal for Numerical Methods in Engineering 32 (1991) 783–810.
[32] R. Rannacher, Adaptive Galerkin finite element methods for partial differential equations, Journal of Computational and Applied Mathematics 128 (2001) 205–233.
[33] A. Brandt, Guide to Multigrid Development, Springer-Verlag, 1982.
[34] L. Formaggia, S. Micheletti, S. Perotto, Anisotropic mesh adaptation with

applications to CFD problems, in: H. A. Mang, F. G. Rammerstorfer, J. Eberhardsteiner (Eds.), Fifth World Congress on Computational Mechanics, Vienna, Austria, 2002.

[35] D. A. Venditti, Grid adaptation for functional outputs of compressible flow simulations, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2002).

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