An Adaptive Simplex Cut-Cell Method for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations

Krzysztof J. Fidkowski *
David L. Darmofal †
Massachusetts Institute of Technology, Cambridge, MA 02139

A cut-cell adaptive method is presented for high-order discontinuous Galerkin discretizations in two and three dimensions. The computational mesh is constructed by cutting a curved geometry out of a simplex background mesh that does not conform to the geometry boundary. The geometry is represented with cubic splines in two dimensions and with a tesselation of quadratic patches in three dimensions. High-order integration rules are derived for the arbitrarily-shaped areas and volumes that result from the cutting. These rules take the form of quadrature-like points and weights that are calculated in a pre-processing step. Accuracy of the cut-cell method is verified in both two and three dimensions by comparison to boundary-conforming cases. The cut-cell method is also tested in the context of output-based adaptation, in which an adjoint problem is solved to estimate the error in an engineering output. Two-dimensional adaptive results for the compressible Navier-Stokes equations illustrate automated anisotropic adaptation made possible by triangular cut-cell meshing. In three dimensions, adaptive results for the compressible Euler equations using isotropic refinement demonstrate the feasibility of automated meshing with tetrahedral cut cells and a curved geometry representation. In addition, both the two and three-dimensional results indicate that, for the cases tested, $p = 2$ and $p = 3$ solution approximation achieves the user-prescribed error tolerance more efficiently compared to $p = 1$ and $p = 0$.

I. Introduction

Computational Fluid Dynamics (CFD) is used regularly in industry to reduce design cycle costs and to improve product design. The accessibility, relatively fast turnaround time, and almost arbitrary test conditions offered by CFD make it an attractive tool, especially for sensitivity studies, optimization, and preliminary vehicle design. Although the use of CFD is prevalent, recent evidence suggests that it is not yet a fully-mature field. In the most recent AIAA Drag Prediction Workshop (DPW), the spread in the on-design drag values computed by some of the current state-of-the-art CFD codes was over 30 counts [1, 2]. Given that 1 drag count translates to several passengers for a typical long-range jet, such a spread is unacceptable for engineering analysis and design.

The recent DPW results demonstrate that the risk of unacceptably large errors is high for current CFD practices. Typically, such risks are managed by practitioners who are knowledgeable about the assumptions and limitations of the models. However, even very experienced users cannot quantify the error in a discrete approximation of a complex flowfield. As a result, current CFD practices are not robust across the wide variety of existing applications, including ones such as the DPW case, for which many of the codes are tuned.

Lack of automation is another key problem that plagues CFD analysis and design. Current industry practices require heavy "person-in-the-loop" involvement, especially during mesh generation, which may take days or weeks of user involvement and is often the bottleneck in CFD analysis. This meshing bottleneck not only extends the design cycle time but also hinders the application of mesh adaptation methods and design optimization. Removing the user completely out of the design loop is neither possible nor advisable;

*Research Assistant, Department of Aerospace Engineering, Building 37, Room 442. Student Member AIAA.
†Associate Professor, Department of Aerospace Engineering, Building 37, Room 401. Senior Member AIAA.

however, improving automation in areas such as meshing is expected to reduce design cycle time and to allow for techniques such as solution-based adaptation and optimization.

In an effort to improve the automation and robustness of CFD, this paper proposes a novel adaptive cut-cell method that uses a simplex (triangular/tetrahedral) background mesh and a high-order finite element discretization. The motivation for cut cells is to remove the meshing bottleneck common to boundary-conforming methods: mesh generation is greatly simplified when the boundary of a complex geometry does not need to be taken into account. Removing this bottleneck then makes an automated adaptive solution method possible.

The idea of using cut cells to automate meshing has been used successfully for finite volume discretizations on lattice-bound meshes [3–7]. These "Cartesian methods" differ from the present work in two ways. First, the present cut-cell method is applied to a high-order discontinuous Galerkin (DG) finite element method in which accurate volume as well as surface integrals need to be computed on irregularly-shaped regions. The need for high-order accuracy is motivated in previous works [8–10] in the interest of practical high-fidelity engineering calculations. Second, the use of simplex background mesh elements in the present work allows for anisotropic adaptation in general directions. Lack of such capability has been a roadblock in applying Cartesian methods to the Navier-Stokes equations.

The present work extends the two-dimensional cut-cell method presented in Ref. [11] into three dimensions, where the intersection and integration problems are of increased complexity. As this work is an initial proof-of-concept study, adaptation in three dimensions is performed with isotropic elements. Anisotropic results are therefore only shown in two dimensions. The outline for the remainder of this paper is as follows. Section II describes the mechanics of intersecting a curved surface representation with a simplex background mesh and the generation of high-order integration rules on the resulting irregular areas and volumes. Section III outlines the error estimation and mesh adaptation procedure. Section IV presents adaptive results in two and three dimensions. Finally, Section V discusses conclusions and ongoing work.

II. Cut-cell Mesh Generation

A. Geometry Definition

In two dimensions, a geometry definition consisting of cubic-splined points is sufficient. Cubic splines can approximate curved geometries very well and possess first and second derivative continuity at spline knots. Geometric corners, where the tangent vector is discontinuous, can be represented using multiple splines. On the other hand, three-dimensional surface modeling is more complicated, with many geometry representation techniques. Computer-Aided Design packages typically employ one or more of a variety of spline representations, including bivariate splines and non-uniform rational B-splines (NURBS) [12]. For CFD purposes, the surface representation should be watertight, which means that no gaps should be present at surface junctures. While general CAD models are not always watertight, robust post-processing tools are available for generating watertight descriptions [13]. However, because of the variability in CAD representations, for this proof-of-concept study, the CAD model is not used directly in the cut-cell method. Rather, an intermediate surface representation is used for which the cut-cell implementation is simplified. This intermediate representation consists of a surface tessellation of quadratic patches. Sample quadratic patches are illustrated in Fig. 1. Each patch is obtained by mapping six nodes from reference space to physical space, and using quadratic interpolation between the nodes: $\mathbf{x} = \sum_{j=1}^{6} \phi_j(\mathbf{X})\mathbf{x}_j$, where the $\mathbf{x}_j$ vectors are physical-space coordinates of the six patch nodes, and the $\phi_j(\mathbf{X})$ are quadratic Lagrange interpolating functions in reference space. The convention for the ordering in physical space is such that the vector obtained via the right-hand rule in traversing nodes 1,2,3 (i.e. $\vec{12} \times \vec{13}$) points into the computational domain.

Quadratic patches were chosen instead of the more common linear patches in order to more efficiently resolve curved geometries to a level required for high-order finite element computations. Specifically, a linear patch tesselation exhibits relatively large geometry slope discontinuities between the patches. If resolved, these corners will lead to non-physical singular features in the solution. A quadratic patch tesselation also exhibits slope discontinuities, but to a much lesser extent than linear patches. For example, for a hemispherical geometry, a linear tesselation would require over 1.8 million patches in order to achieve a slope error comparable to a tesselation with 210 quadratic patches [14]. Thus, for high-order computations, in which the geometry slope continuity is important, quadratic patches are expected to be much more efficient compared to linear patches.

In practice, quadratic patches are obtained from linear tesselations by adding extra nodes on the geometry

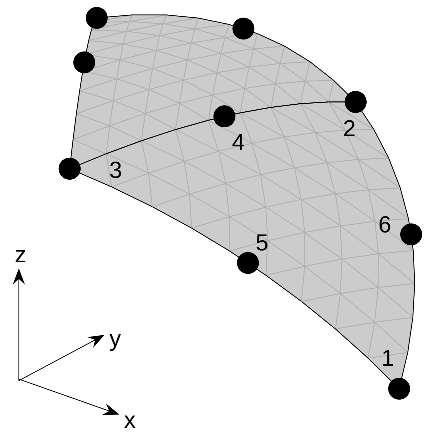


Figure 1. Two adjacent quadratic patches; the local node numbering for one of the patches is shown.

at the midpoints of the surface triangle edges. They are watertight because the interpolation of an edge depends only on the locations of the three nodes on that edge, so that the surface interpolations from two adjacent patches always match. While the nodes of the patches can be chosen to lie exactly on the geometry, the quadratically-interpolated coordinates of the patch interior and edges in general will not coincide with the geometry. Thus, quadratic patches serve only as an approximation to the true geometry. A more accurate or even exact geometry representation with tractable cut-cell intersection and integration algorithms is expected to perform better and is an area of possible future work.

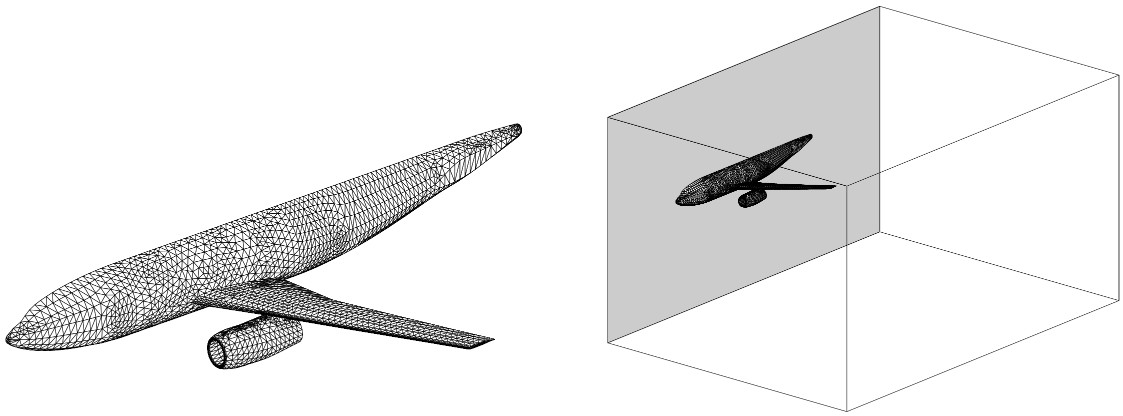
B. Cutting Algorithm

The cutting algorithm takes as input a spline or quadratic-patch surface representation of the geometry of interest and a linear area/volume mesh of the background domain. An example of a quadratic-patch surface representation of a wing-body-nacelle geometry is shown in Fig. 2a. Due to symmetry, only half of the geometry is modeled. Shown in Fig. 2b is one possible choice for the background domain. In this case, it is a box; on five sides of the box, farfield boundary conditions are imposed, and the remaining side is a symmetry plane.

The output of the cutting algorithm is a cut-cell mesh of the computational domain obtained from the original background mesh by removing elements completely contained in the geometry and by appropriately cutting elements that intersect the geometry. The resulting cut cells are portions of the original triangles/tetrahedra that lie inside the computational domain. Fig. 3a illustrates a two-dimensional example in which a background mesh of triangles is cut by an airfoil spline geometry. An embedded edge and two cut edges are identified for one resulting cut cell. Embedded edges consist of contiguous portions of the spline geometry inside the background mesh triangles, whereas cut edges consist of portions of background triangle edges inside the computational domain. Fig. 4a shows a three-dimensional example: an intersection between a background-mesh tetrahedron and a quadratic patch surface. The upper portion of the tetrahedron lies inside the computational domain and forms a cut cell. Ultimately, for use in the solver, integration rules are required on the interior and on the boundary of each cut cell. However, generating these rules first requires identification and description of the intersections that produce the cut cells.

In two dimensions, the relevant intersections are between the cubic splines and the edges of the background triangles. These intersections are identified analytically by solving cubic equations. Each background triangle thus yields a set of embedded and cut edges that form the boundaries of the cut cells. Edges sharing common intersections are joined into loops that enclose disjoint cut cells. Note, multiple cut cells arising from a single background triangle are certainly possible, as shown in Fig. 3b for a triangle straddling an airfoil trailing edge. In such cases, each disjoint area is associated with a distinct cut cell equipped with separate sets of polynomial basis functions.

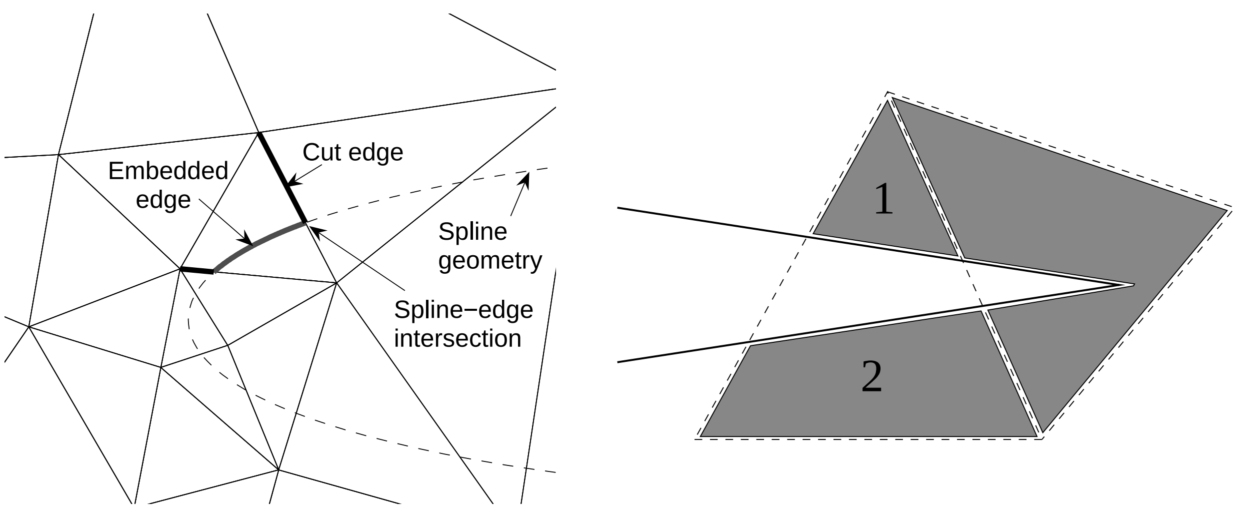
In three dimensions, the intersection problem is more complex. Figure 4b illustrates the basic intersection



(a) Quadratic patches

(b) Sample background domain

Figure 2. Quadratic patch surface representation of a wing-body-nacelle geometry (a) and one possible choice for the background domain (b). The shaded side of the background domain indicates a symmetry boundary condition.



(a)

(b)

Figure 3. Intersection between a background mesh and an airfoil spline geometry. (a) illustrates embedded and cut edges for one cut cell, and (b) presents an example of a multiply-cut background triangle.

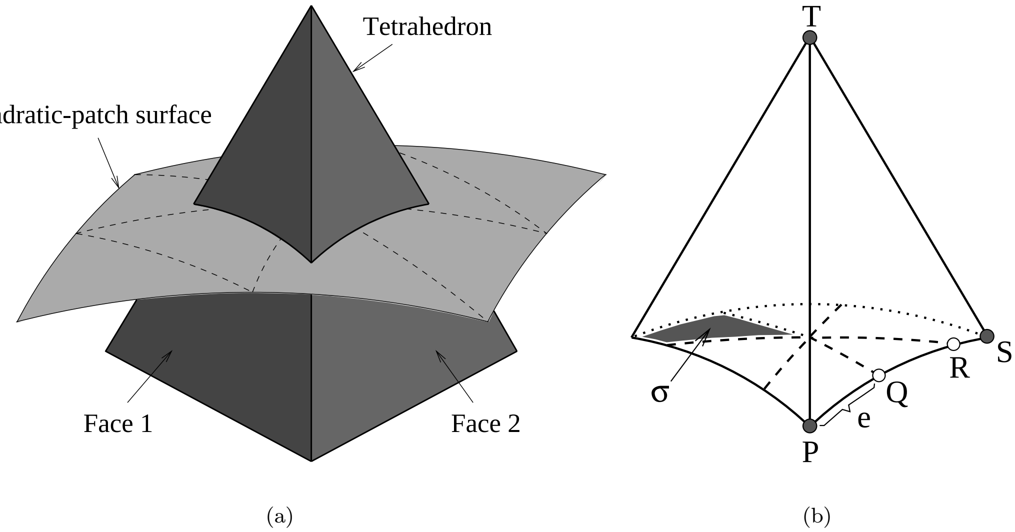


Figure 4. A background-mesh tetrahedron intersecting a quadratic-patch surface (a). The upper portion of the tetrahedron lies inside the computational domain. A wire-frame of the resulting cut-cell (b) is obtained by joining various intersection points (e.g. P and Q) into edges (e.g. e). Loops of edges enclose faces, such as the shaded one labeled by σ.

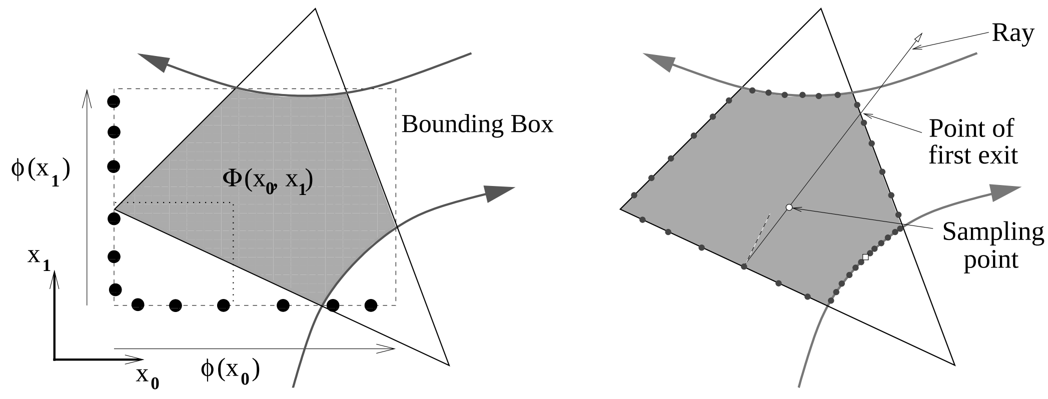
structures for the cut cell from Fig. 4a. As shown, the cut cell is enclosed by a wire-frame of edges, which are possibly-curved line segments that join pairs of intersection points. These intersection points consist of tetrahedron vertices lying inside the computational domain, intersections between tetrahedron edges and patches (e.g. point P in the figure), and intersections between patch edges and tetrahedron faces (e.g. point Q in the figure). Loops of edges enclose faces of the new cut cell. As in two dimensions, a single background element can yield multiple cut cells. Again, in such cases, each distinct cut cell is equipped with separate basis functions.

A key component of the three-dimensional cutting algorithm is the identification of the intersection points and edges that form the wire frame of each cut cell. For quadratic patches, the intersection edges can be identified analytically. Specifically, the intersection between a quadratic patch and a tetrahedron face is a portion of a conic section (ellipse, parabola, hyperbola, etc.) in the reference space of the patch. An example of an edge resulting from such an intersection, mapped into physical space, is edge e in Fig. 4b. Periodic intersection edges, arising from completely-contained ellipses, are split along the major axis to form two edges. The remaining edges in the wire frame consist of portions of the straight tetrahedron edges lying inside the computational domain and portions of the quadratic patch edges lying inside the tetrahedron.

Once available, the edges comprising the wire frame can be connected together into faces. Specifically, on each original face and intersecting quadratic patch, edges are connected into loops according to their shared intersection points. For example, on the cut face arising from face 2 of the tetrahedron in Fig. 4, five edges (e being one of them) are connected into a single loop according to the intersection points P, Q, R, S, and T. Such loops of edges enclose the new faces of the cut cells. Note, each intersecting quadratic patch yields a separate face. The new faces are then tied together into surfaces that enclose disjoint volumes. Connections between faces are determined by the edges: two faces sharing the same edge enclose the same volume.

C. Integration

Irregularly-shaped cut cells require a modified integration technique since triangular and tetrahedral-based quadrature rules are no longer valid. In addition, in three dimensions, cut faces of the background tetrahedra and embedded faces on the geometry surface also require a modified integration technique, as they are in general no longer triangular. In this work, these two and three-dimensional integrations are performed using a direct application and extension of the two-dimensional technique introduced in [11]. This technique consists of “speckling” sampling points in the area or volume of interest and applying the divergence theorem



(a) Tensor-product Lagrange functions, $\Phi_{\mathbf{i}}(\mathbf{x})$ (b) Sampling point selection

Figure 5. (a) Definition of the $\Phi_{\mathbf{i}}(\mathbf{x})$ functions for use in defining the basis, $\zeta_{\mathbf{i}}(\mathbf{x})$, for integrating on the irregularly-shaped shaded area. (b) Illustration of the ray-casting procedure for sampling point selection.

to pre-compute an integration weight for each sampling point.

As will be shown shortly, deriving volume integration rules on three-dimensional cut cells requires the ability to integrate on the enclosing faces. As these faces may be irregularly-shaped, integration rules must also be derived on them, thus requiring the ability to integrate on the enclosing loops of edges. On each edge, high-order accurate integration is achieved by mapping the edge to a reference interval and using Gaussian quadrature on the interval. This mapping is trivial for straight segments. For curved segments, a well-conditioned parametrization is required. In two dimensions, the arclength parameter is sufficient for curved spline segments. In three dimensions, curved conic segments are parametrized by angle from a point in reference space which is usually chosen as the closest focus to the conic segment. When the focus is far from the segment of interest, a suitably-chosen well-conditioned point (e.g. one of the reference triangle vertices) is used for the origin point. Details on the parametrization are given in [14].

Integration rules for irregularly-shaped two-dimensional areas bounded by possibly-curved edges, such as the one pictured in Fig. 5, consist of sampling points x_q and associated weights w_q, Note, for curved embedded faces in three dimensions, the integration is performed in reference space of the face. Of interest is the calculation of the integral $\int_A f(\mathbf{x})dA$, where $f(\mathbf{x})$ is an arbitrary integrand and A is the enclosed two-dimensional area. The approach in this work consists of projecting the integrand onto a set of N_{basis} high-order basis functions, $\zeta_{\mathbf{i}}(\mathbf{x})$, that are specially-constructed to allow for simple computation of the integral $\int_A \zeta_{\mathbf{i}}(\mathbf{x})dA$. Specifically, the $\zeta_{\mathbf{i}}$ are

$$\zeta_{\mathbf{i}} \equiv \nabla \cdot \left(\mathbf{x}\Phi_{\mathbf{i}}(\mathbf{x})\right), \tag{1}$$

where the $\Phi_{\mathbf{i}}(\mathbf{x})$ are high-order tensor-product Lagrange basis functions on the bounding box, as illustrated in Fig. 5a. For anisotropically-shaped sliver areas, the axes are rotated to produce a tightly-fitted bounding box. The divergence operator in Eq. 1 allows for application of the divergence theorem to convert the interior integral to a boundary integral: $\int_A \zeta_{\mathbf{i}}(\mathbf{x})dA = \int_{\partial A} \mathbf{n}\cdot\mathbf{x}\ \Phi_{\mathbf{i}}(\mathbf{x})ds$, where ∂A is the one-dimensional boundary of A, with outward pointing normal $\mathbf{n}$. The boundary integral is then computed using the edge integration rules.

The remaining piece, projection of the integrand onto the basis $\zeta_{\mathbf{i}}(\mathbf{x})$, is performed by a least-squares minimization of integrand values sampled at a randomly-chosen set of points inside the cut cell. The end result is a set of N_{quad} sampling points, x_q, and weights, w_q, that enable a quadrature-like sum expression for the integral,

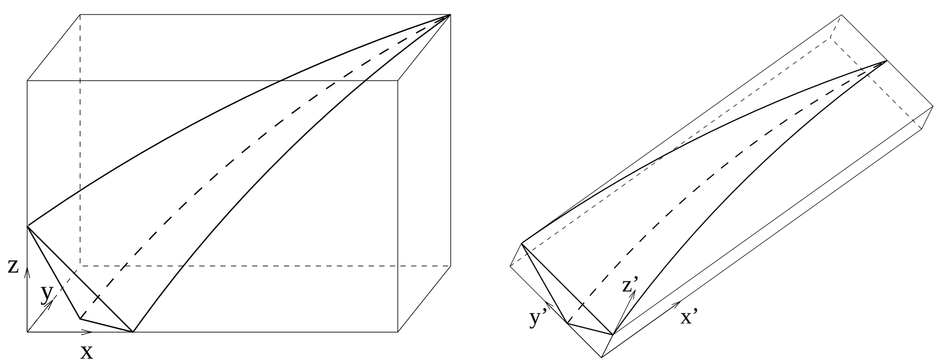
$$\int_A f(\mathbf{x})dA \approx \sum_q w_q f(\mathbf{x}_q).$$

The weights are given by

$$w_q = Q_{q\mathbf{j}}(R^{-T})_{\mathbf{ji}} \int_A \zeta_{\mathbf{i}}(\mathbf{x}) dA, \tag{2}$$

where $\zeta_{\mathbf{i}}(\mathbf{x}_q) = Q_{q\mathbf{j}} R_{\mathbf{ji}}$ is a QR factorization and summation is implied on repeated indices. Since the expression for w_q does not involve the integrand, $f(\mathbf{x})$, the weights can be calculated in a pre-processing step and stored. The random sampling points, x_q are determined by casting interior-bound rays from the enclosing edges and randomly chosing a point between the ray origin and the point of first exit. This procedure is illustrated in Fig. 5b. To minimize the probability of an ill-conditioned set of sampling points, oversampling is used in which N_{quad} is set to $4N_{\text{basis}}$.

In three dimensions, the face integration rules are used directly for constructing the residual during solution iteration. They are also used in preprocessing for creating volume integration rules. The volume integration rules are calculated using a straightforward extension of the sampling point speckling procedure used in two dimensions. In particular, the sampling point weights are given by Eq. 2 with a volume integral instead of an area integral. The integrand basis functions are still given by Eq. 1, with the tensor product functions, $\Phi_{\mathbf{i}}(\mathbf{x})$, now defined on the bounding box of the volume. To improve conditioning of the integration, a tightly-fitted oriented bounding box is used, as indicated in Fig. 6. By the divergence theorem, volume



(a) Axis-aligned bounding box (b) Oriented bounding box

Figure 6. Numerical conditioning improvement of element-interior integration via bounding-box rotation for the case of a sliver element.

integrals of the $\zeta_{\mathbf{i}}(\mathbf{x})$ are converted into area integrals on the volume boundary, which are computed using the face integration rules. The sampling points in three dimensions are also chosen using a ray-casting procedure. Interior bound rays are cast from the bounding faces, and random points are chosen between the origin of each ray and the points of first exit. The volume integration points and weights are computed and stored in a pre-processing step.

The $\zeta_{\mathbf{i}}(\mathbf{x})$ and $\Phi_{\mathbf{i}}(\mathbf{x})$ basis functions are used only for construction of the integration rules. The order of these functions is the maximum required order of integration (e.g. for residual construction). For the Euler equations with conservative state variables, an integration order of $2p+1$ has been found to be sufficient. On cut cells the solution is approximated using a standard Lagrange or Hierarchical basis defined on a right tetrahedron taken from the oriented bounding box in Fig. 6. In DG, the choice of approximation basis is not constrained by any inter-element continuity requirements. Using the oriented bounding box improves the approximation conditioning for sliver elements.

III. Error Estimation and Adaptation

This work employs an error estimator and indicator based on an engineering output, such as a force or moment on the geometry. Such an indicator identifies all areas of the domain that are important for the accurate prediction of the output, properly accounting for propagation effects of convection-dominated problems. The technique used requires solution of an adjoint problem associated with the output, where the adjoint links local residuals to the output error. This approach has been studied extensively in the literature [15–21]. In particular, the error estimate used in this work is nearly identical to that used by Hartmann and Houston and by Lu. On each element κ, the local error indicator is given by

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

In the above expression, $\mathbf{u}_H$ and $\boldsymbol{\psi}_H$ are the flow and adjoint solutions, and $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$ are higher-order approximations obtained via a patch-reconstruction of $\mathbf{u}_H$ and $\boldsymbol{\psi}_H$ on an enriched, order $p+1$ space. $\mathcal{R}_h$ and $\mathcal{R}_h^\psi$ are the flow and adjoint residuals evaluated on the enriched space, and $|_\kappa$ denotes restriction to element κ. The global output error estimate, $\epsilon = \sum_\kappa \epsilon_\kappa$, is not a bound on the actual error in the output; however, its validity is expected to increase as $\mathbf{u}_H$ and $\boldsymbol{\psi}_H$ approach the exact solutions $\mathbf{u}$ and $\boldsymbol{\psi}$.

Given a localized error estimate, an adaptive method modifies the computational mesh in an attempt to decrease and equidistribute the error. 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 additional complexity involved in making the regularity estimation decision. This simplification also allows for a straightforward comparison of the adaptive performance at different interpolation orders. However, extension to hp-adaptation is one of the areas of possible future work.

Inputs to the h-adaptation method are the current mesh, the solution on the mesh, an element-local error indicator, and a user-specified error tolerance, e_0. The output is a mesh-size request in the form of a metric associated with each element for use in re-meshing. Adaptation stops when the error tolerance is met. A flowchart for the adaptive solution process is shown in Fig. 7. In practice, the most expensive step is the flow and adjoint solution on each mesh.

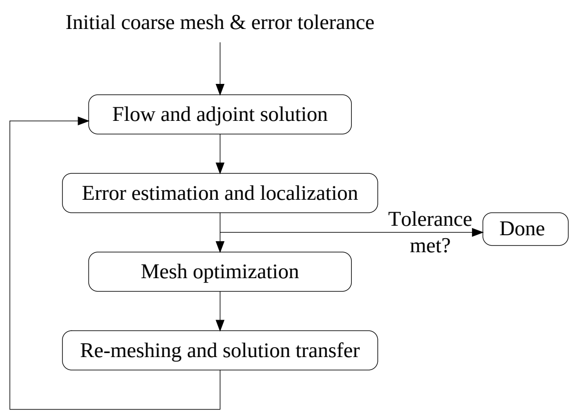


Figure 7. Adaptive solution process flowchart

Mesh optimization is performed at every adaptation iteration following output-error estimation. In this step, the principle of error equidistribution is used in conjunction with anisotropy detection based on high-order interpolation of the Mach number. Details of the optimization procedure are given in [11]. In the present work, anisotropy detection is only performed in two dimensions. Metric-driven meshing of the background domain is performed via the Bi-dimensional Anisotropic Mesh Generator (BAMG) [22] in two dimensions and TetGen [23] in three dimensions. One of the advantages of using cut cells in an adaptive method lies in being able to re-mesh the domain at each iteration according to some prescribed metric without requiring the mesh to conform to an intricate geometry. Such re-meshing requires specification of the metric request everywhere in the background domain, including inside the geometry. Since triangles and tetrahedra completely contained within the geometry are removed from the cut-cell data structure, they do not possess an error estimate or associated metric. On these elements, a grid-implied metric is used, under which the element size and stretching remain approximately constant.

IV. Numerical Results

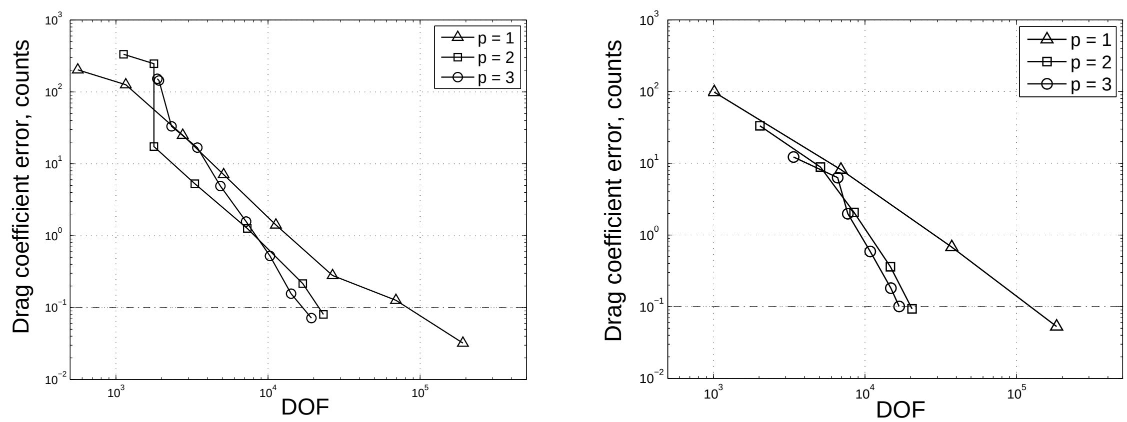
The cut-cell adaptive method is applied to several aerodynamic cases to demonstrate the accuracy and adaptive convergence of simplex cut cells with curved embedded boundaries. The discontinuous Galerkin discretization is applied to the compressible Navier-Stokes equations in two dimensions and to the Euler equations in three dimensions. The Roe-averaged flux [24] is used for the inviscid term, and the second form of Bassi and Rebay [25] is used for the viscous term. The resulting non-linear system of equations is solved via a preconditioned Newton-GMRES technique [26].

Comparisons of the adapted meshes and the error convergence histories are given in terms of degrees of freedom (DOF) using interpolation orders $p = 0$ to $p = 3$ ($p = 2$ for three dimensions). The DOF count does not include the equation-specific multiplier, which is 4 in two dimensions and 5 in three dimensions. In addition, computational work estimates are given with the DOF results. The work estimate is of the form $W \sim N_e(n(p))^a = \mathrm{DOF}(n(p))^{a-1}$, where N_e is the number of elements, $n(p) = \prod_{i=1}^{d}(1 + p/i)$ is the DOF count per element, and a is a measure of the computational complexity. From observations in practice, the work is assumed to be dominated by matrix-vector products, so that $a \sim 2$.

A. Viscous flow over a NACA 0012 airfoil

In this case, a Navier-Stokes solution is computed around a NACA 0012 in a freestream Mach number of 0.5, Reynolds number of 5000, and angle of attack of 2^o. The initial boundary-conforming and cut-cell meshes are isotropic and adapted to the geometry with roughly 250 elements. The farfield is a square, 100 chord lengths away from the airfoil. Mesh optimization is performed with anisotropic elements to efficiently resolve the boundary layer and wake.

The adaptation algorithm was tested using drag as the output, with a tolerance of $e_0 = 0.1$ counts. 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 error convergence histories from the adaptation are plotted in Fig. 8. Overall, the cut-cell and boundary-conforming results are similar.



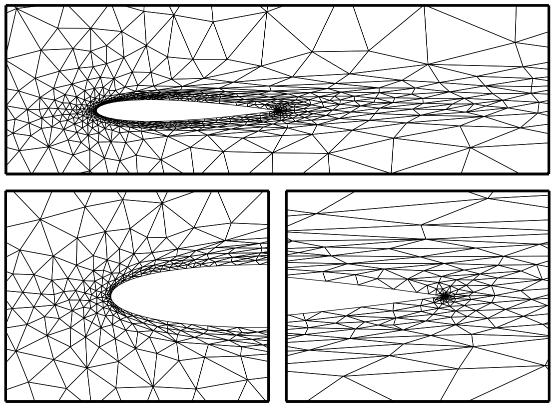
(a) Boundary-conforming

(b) Cut-cell

Figure 8. NACA 0012: $M_\infty = 0.5, Re = 5000, \alpha = 2^o$. Drag error versus degrees of freedom. Dashed line indicates prescribed tolerance of $e_0 = 0.1$ drag counts.

For both the boundary-conforming and the cut-cell cases, $p = 3$ requires the fewest degrees of freedom at the error tolerance, although $p = 2$ does not require much more. 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 4-5 more degrees of freedom than $p = 2$, which translates to an estimated work increase of about a factor of 2.

Figure 9 shows the final adapted meshes for $p = 3$. For comparison, the $p = 2$ meshes contain 4068 (boundary-conforming) and 3403 (cut-cell) elements, while the $p = 1$ meshes contain 98808 (boundary-conforming) and 61163 (cut-cell) elements. 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. Elements in the boundary layer and in the wake are stretched in the flow direction, correctly capturing the anisotropy in this viscous solution. The similarity in element size and anisotropic stretching between the cut-cell meshes and their boundary-conforming counterparts is evident. Finally, the skin friction coefficient distributions for

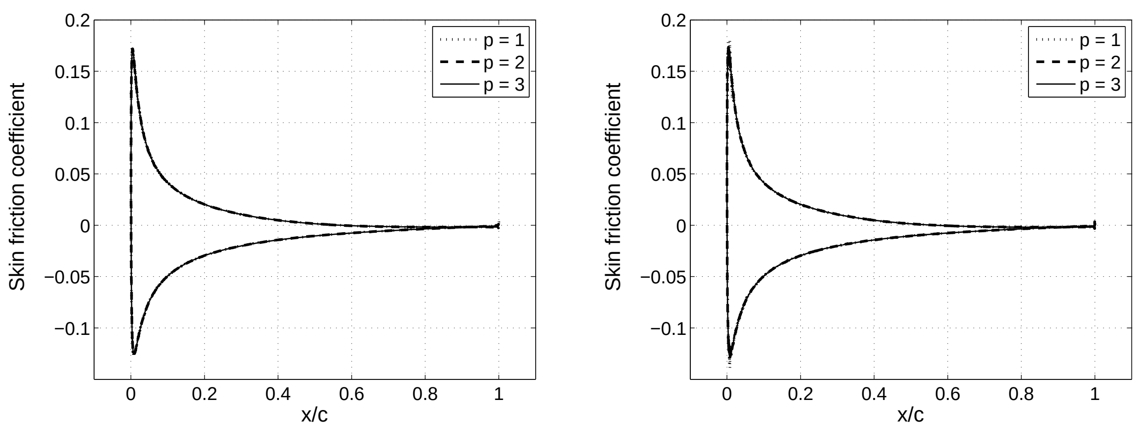


(a) Boundary-conforming: 1781 elements

(b) Cut-cell: 1680 elements

Figure 9. NACA 0012: $M_\infty = 0.5, Re = 5000, \alpha = 2^o$. **Final** $p = 3$ **meshes adapted on drag with tolerance** $e_0 = 0.1$ **counts.**

solutions on the final adapted meshes are shown in Fig. 10. As shown, the plots are practically identical for the boundary-conforming and the cut-cell cases and for the different orders, p.



(a) Boundary-conforming

(b) Cut-cell

Figure 10. NACA 0012: $M_\infty = 0.5, Re = 5000, \alpha = 2^o$. **Surface skin friction coefficient distributions on the final adapted meshes.**

B. Channel Flow Over a Gaussian Perturbation

This case considers inviscid flow through a channel, the floor of which has a Gaussian perturbation in the streamwise (x) direction. The output of interest for the adaptive runs is the drag, which is the x-component of the force on the channel floor. The true value of the drag for this flow is not exactly zero due to the proximity of the inflow and outflow boundaries. Thus, the true value is computed using $p = 3$ interpolation on a structured, 18432-element, boundary-conforming mesh, shown in Fig. 11. This mesh was generated manually, and elements with faces or edges on the bottom surface were curved to cubic order [27]. Comparing the adapted cut-cell results to a boundary-conforming true value tests the accuracy of the cut-cell method.

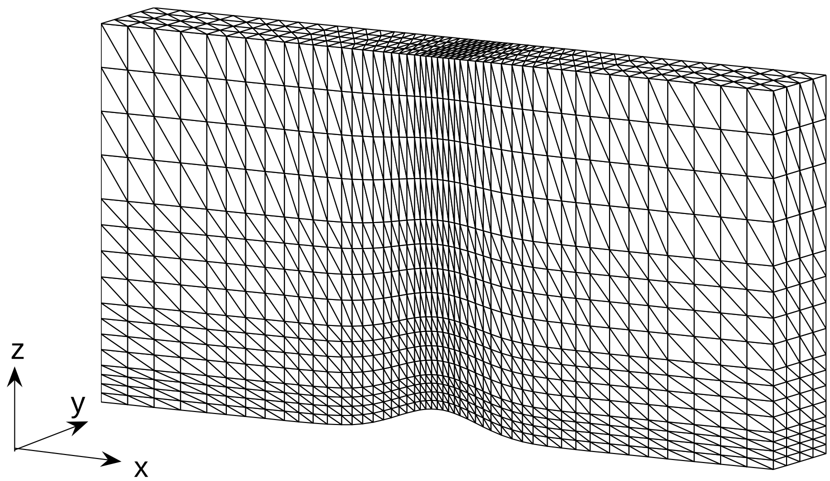


Figure 11. Manually-generated, 18432-element, boundary-conforming mesh of a channel with a Gaussian bump perturbation.

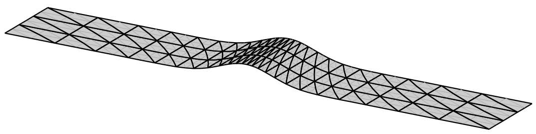
For the cut-cell runs, the floor of the channel is represented with 144 quadratic patches. Figure 12 shows the embedded surface along with an initial 576-element background mesh that was generated manually by subdividing rectangular parallelepipeds of a structured mesh. Adaptive runs were performed for $p = 0, 1, 2$, using $e_0 = 0.15$ counts for the drag tolerance. The drag coefficient was computed using the floor planform as the reference area. Figure 13 shows the results of the adaptation runs.

Adaptation for $p = 0$ was not continued down to the drag tolerance as computational costs became prohibitive. However, assuming that the $p = 0$ convergence rate continues, over 10^{10} DOF would be necessary to reach the error tolerance. On the other hand, both $p = 1$ and $p = 2$ converge to the desired error tolerance, at which point $p = 2$ requires an order of magnitude fewer degrees of freedom than $p = 1$. In terms of estimated work at the error tolerance, $p = 2$ is cheaper by a factor of four compared to $p = 1$.

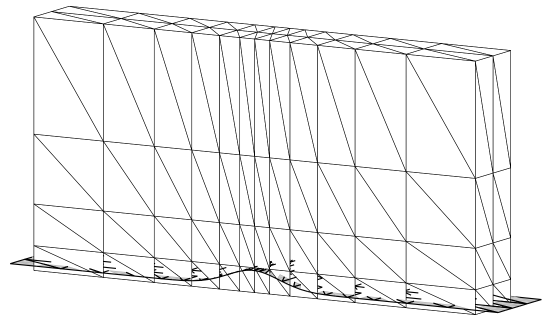
The final adapted meshes for $p = 1$ and $p = 2$ are shown in Fig. 14. These are the background meshes prior to cutting. As such, tetrahedra outside the computational domain are still shown. As expected, refinement is concentrated in the vicinity of the bump, inside the computational domain. The $p = 2$ mesh is much coarser than the $p = 1$ mesh, requiring only 1787 elements as opposed to 51264 elements for $p = 1$. For comparison, the finest $p = 0$ mesh contains 377042 elements and yields an error of over 20 counts. Mach number contours for solutions on the adapted $p = 1$ and $p = 2$ meshes are shown in Fig. 15, for a planar cut parallel to the $x - z$ plane taken down the middle of the channel. Since the solution is on a cut-cell mesh, some of the contours extend through the geometry. This is a by-product of the visualization, as the solution is rendered on tetrahedra of the background mesh. For the flow solver, the solution is only physically valid inside the computational domain. The contours are similar for $p = 1$ and $p = 2$. The $p = 2$ contours appear slightly more rugged, although this is in part due to the fact that the $p = 2$ solution is plotted using linear rendering on a uniformly-refined mesh.

C. Wing-body Configuration

In this case, a more complex geometry, consisting of the DLR-F6 wing-body configuration from the DPW workshop [2], is used to demonstrate the robustness of the cut-cell, adaptive method for practical problems of interest. Figure 16 shows the geometry, tiled with 9368 quadratic patches placed using curvature-based spacing. A coarse, geometry-adapted mesh of 20447 elements served as the initial mesh for adaptive runs at



(a) 144 surface patches



(b) 576-element initial mesh

Figure 12. Quadratic patch representation of the bump surface (a) and the initial background mesh (b).

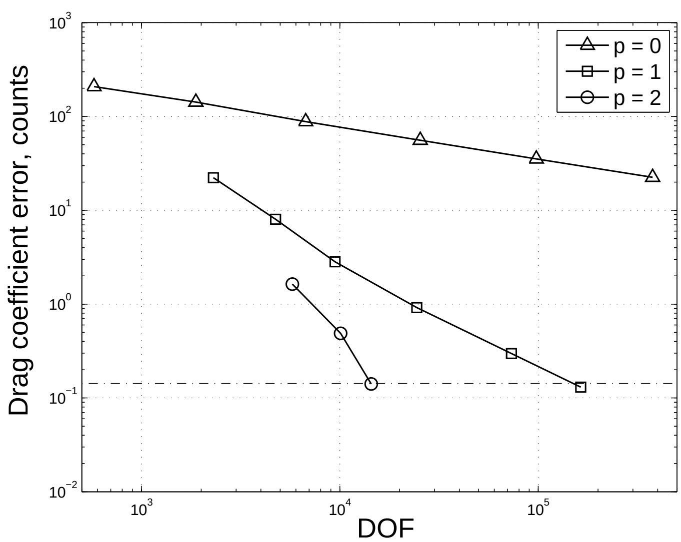
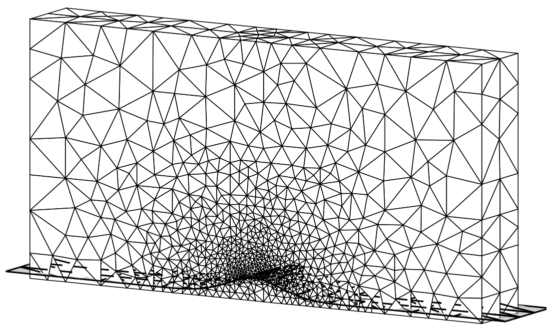


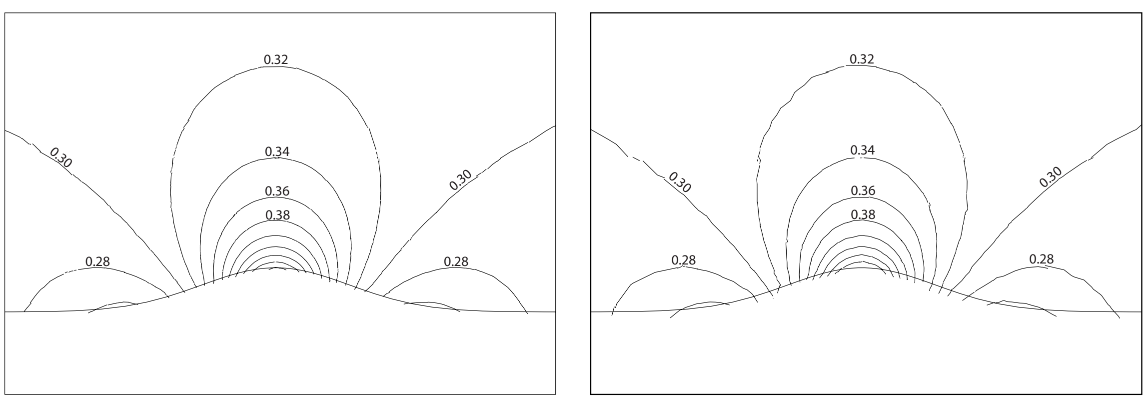
Figure 13. Gaussian bump channel: $M = 0.3$. Drag output error vs degrees of freedom. Dashed line indicates prescribed tolerance of $e_0 = 0.15$ counts.



(a) $p = 1$: 51264 elements

(b) $p = 2$: 1787 elements

Figure 14. Gaussian bump channel: $M = 0.3$. Final adapted meshes for $p = 1$ and $p = 2$.



(a) $p = 1$

(b) $p = 2$

Figure 15. Gaussian bump channel: $M = 0.3$. Mach number contours on the final adapted meshes for $p = 1$ and $p = 2$.

orders $p = 0, 1, 2$. Adaptation was based on drag, with 1 error count as the tolerance. The wing planform area was used to non-dimensionalize the drag. Figure 17 shows the convergence of the drag coefficient for the three orders. For this case, no "true" drag value was available due to computational limitations. Nevertheless, Fig. 17 indicates that $p = 0$ is converging much more slowly compared to $p = 1$ and $p = 2$. After the initial adaptation iterations, $p = 2$ appears to be converging more quickly than $p = 1$. More importantly, the successful application of cut cells and adaptation in this case demonstrates the robustness and automation possible for a practical geometry configuration.

Figure 18 compares the finest $p = 1$ and $p = 2$ meshes. The meshes are plotted on the symmetry plane, and the quadratic-patch surface is overlaid. As expected, areas of refinement on the symmetry plane include the nose and the tail. Away from the symmetry plane, the leading and trailing edges of the wing also exhibit high refinement. Finally, Fig. 19 shows the Mach number contours for the finest $p = 1$ and $p = 2$ mesh solutions, at a section of the wing 50% along the half-span. Both contour plots exhibit a slight dissipation wake off the trailing edge, indicating that the flow is not very highly resolved. The $p = 1$ and $p = 2$ contours are very similar, with $p = 2$ exhibiting slightly smoother features compared to $p = 1$.

V. Conclusions

This paper presents an automated adaptation procedure for high-order discontinuous Galerkin discretizations in two and three dimensions. The key components of this method are output-based mesh adaptation for high-order solutions and simplex, cut-cell meshing. Together, these ideas target two shortcomings in current CFD practices: insufficient automation and insufficient robustness in the geometry-to-solution process.

The two-dimensional result presented in this paper is representative of a more comprehensive set of results given in Ref. [11]. The present result illustrates the feasibility of anisotropic adaptation in general directions using simplex cut-cells. In such situations, when mesh anisotropy is required near a curved boundary, cut-cells are often more robust in that they do not require curving boundary elements, a step that can potentially introduce negative areas. In addition, the adaptive convergence rates on the cut-cell meshes are similar to those observed on the boundary-conforming meshes.

While preliminary, the three-dimensional results demonstrate the feasibility of using tetrahedral cut cells with curved geometry representations. Quadratic patches allow for accurate and efficient representation of curved geometries and thereby make possible the use of high-order solution approximation. As indicated in the results, higher order is desirable for high accuracy: $p = 2$ converges at a steeper rate compared to $p = 1$ and $p = 0$. While the optimal approximation order will in general depend on the problem and on the desired error tolerance, the results suggest that $p > 1$ (i.e. higher than second-order solution convergence) is likely

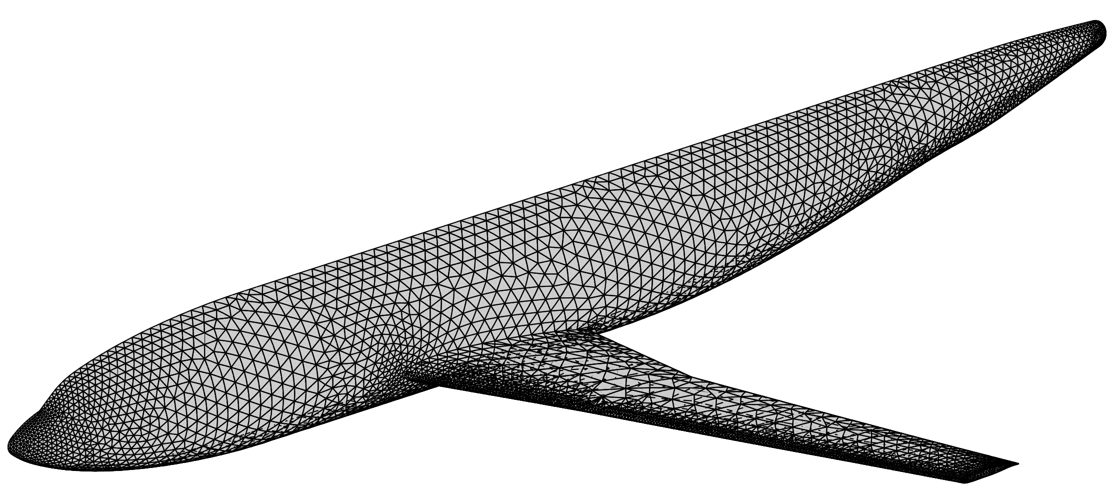


Figure 16. Wing-body: $M = 0.1, \alpha = 0^o$. Surface representation with 9368 quadratic patches.

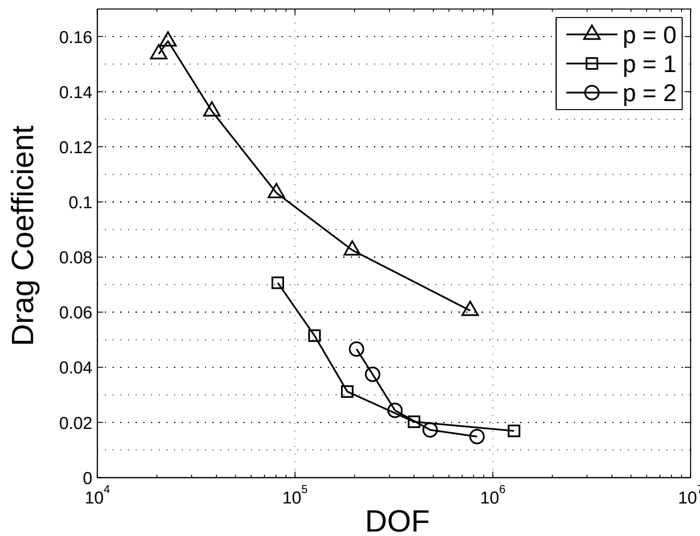
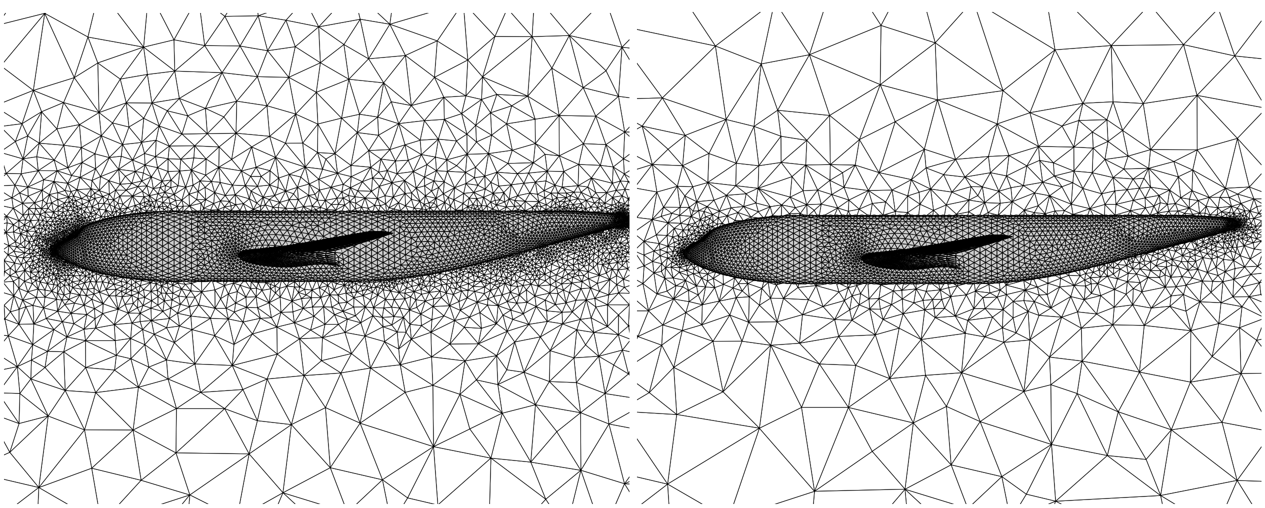


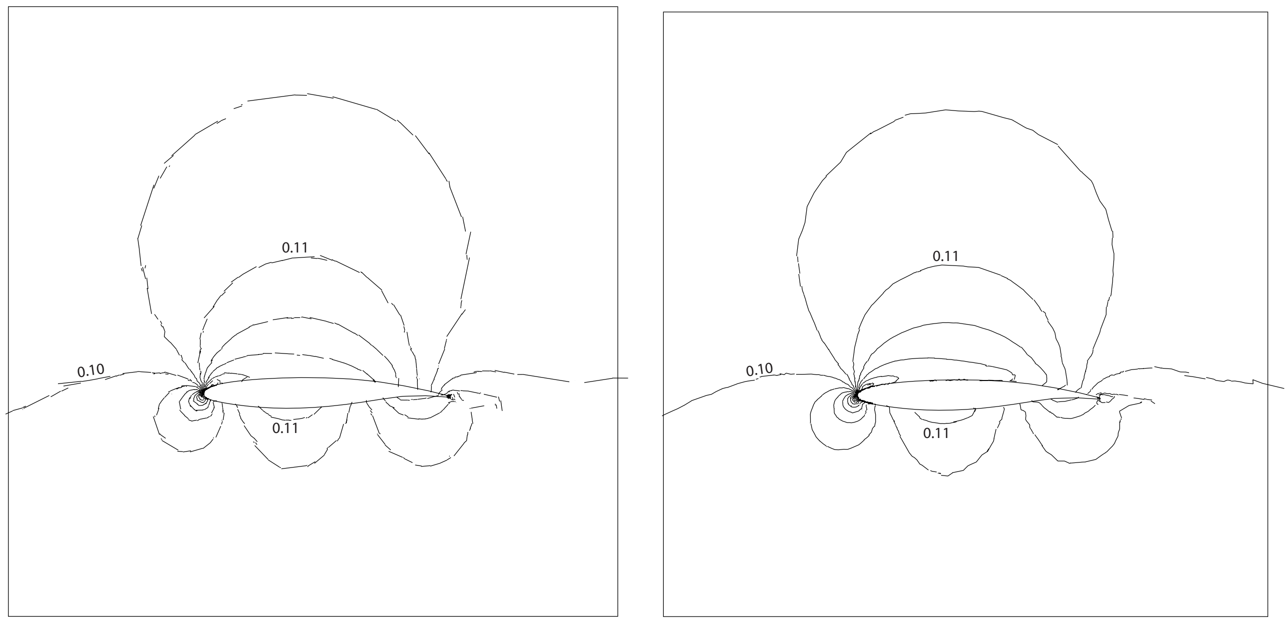
Figure 17. Wing-body: $M = 0.1, \alpha = 0^o$. Drag output versus degrees of freedom.



(a) $p = 1$: 320245 elements

(b) $p = 2$: 83193 elements

Figure 18. Wing-body: $M = 0.1, \alpha = 0^o$. Finest adapted meshes for $p = 1$ and $p = 2$.



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

Figure 19. Wing-body: $M = 0.1, \alpha = 0^o$. Mach number contours on the finest meshes for $p = 1$ and $p = 2$. The cut is parallel to the $x-z$ plane and is situated at 50% of the half-span.

appropriate for practical engineering computations. Finally, the use of tetrahedra for the background mesh paves the path for anisotropic adaptation in three dimensions, which is the subject of ongoing work.

VI. Acknowledgments

K. Fidkowski's work was supported by the Department of Energy Computational Science Graduate Fellowship, under grant number DE-FG02-97ER25308. The authors also thank the Boeing Company, technical monitor Mori Mani.

References

[1]Frink, N. T., "Test Case Results from the 3rd AIAA Drag Prediction Workshop," NASA Langley, 2007, `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/ Workshop3/final_results_jm.tar.gz`.

[2]Morrison, J. H. and Hemsch, M. J., "Statistical Analysis of CFD Solutions from the Third AIAA Drag Prediction Workshop," AIAA Paper 2007-254, 2007.

[3]Rubbert, P. E., Bussoletti, J. E., Johnson, F. T., Sidwell, K. W., Rowe, W. S., Samant, S. S., SenGupta, G., Weatherill, W. H., Burkhart, R. H., Everson, B. L., Young, D. P., and Woo, A. C., "A New Approach to the Solution of Boundary Value Problems Involving Complex Configurations," *Computational Mechanics – Advances and Trends*, edited by A. K. Noor, 1986, pp. 49–84.

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

[5]Pember, R., Bell, J. B., Colella, P., Crutchfield, W. Y., and Welcome, M. L., "An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions," *Journal of Computational Physics*, Vol. 120, 1995, pp. 278–304.

[6]Coirier, W. J. and Powell, K. G., "Solution-adaptive cut-cell approach for viscous and inviscid flows," *AIAA Journal*, Vol. 34, No. 5, 1996, pp. 938–945.

[7]Aftosmis, M. J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," *von Karman Institute for Fluid Dynamics, Lecture Series 1997-02*, Rhode-Saint-Genése, Belgium, Mar. 3-7, 1997.

[8]Venkatakrishnan, V., Allmaras, S. R., Kamenetskii, D. S., and Johnson, F. T., "Higher Order Schemes for the Compressible Navier-Stokes Equations," AIAA Paper 2003-3987, 2003.

[9]Mavriplis, D. J., "An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers," *Journal of Computational Physics*, Vol. 175, 2001, pp. 302–325.

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

[11]Fidkowski, K. J. and Darmofal, D. L., "A Triangular Cut-Cell Adaptive Method for High-Order Discretizations of the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, 2007, doi:10.1016/j.jcp.2007.02.007.

[12]Dawes, W. N., Dhanasekaran, P. C., Demargne, A. A. J., Kellar, W. P., and Savill, A. M., "Reducing Bottlenecks in the CAD-to-Mesh-to-Solution Cycle Time to Allow CFD to Participate in Design," *Journal of Turbomachinery*, Vol. 123, No. 11, 2001, pp. 552–557.

[13]Haimes, R., "CAPRI: Computational Analysis Programming Interface, a Solid Modeling Based Infra-structure for Engineering Analysis and Design." CAPRI user's guide, MIT, Revision 1.00, 2000.

[14]Fidkowski, K. J., *A Simplex Cut-Cell Adaptive Method for High-Order Discretizations of the Compressible Navier-Stokes Equations*, Ph.D. thesis, M.I.T., Department of Aeronautics and Astronautics, June 2007.

[15]Pierce, N. A. and Giles, M. B., "Adjoint recovery of superconvergent functionals from PDE approximations," *SIAM Review*, Vol. 42, No. 2, 2000, pp. 247–264.

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

[17]Hartmann, R. and Houston, P., "Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations," *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532.

[18]Barth, T. and Larson, M., "A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes," *Finite Volumes for Complex Applications III*, edited by R. Herban and D. Kröner, Hermes Penton, London, 2002.

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

[20]Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.

[21]Lu, J., *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.

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

[23]Si, H., "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator," Weierstrass Institute for Applied Analysis and Stochastics, 2005, `http://tetgen.berlios.de`.

[24]Roe, P. L., "Approximate Riemann solvers, parametric vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[25]Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

[26]Diosady, L. and Darmofal., D., "Discontinuous Galerkin Solutions of the Navier-Stokes Equations using Linear Multigrid Preconditioning," AIAA Paper 2007-3942, 2007.

[27]Fidkowski, K. J., *A High-Order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications*, MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.