An Output-based Adaptive and Higher-Order Method for a Rotor in Hover

by

James M. Modisette

B.S., Aerospace Engineering (2005)
Massachusetts Institute of Technology

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Masters of Science in Aerospace Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[February 2008]
January 2008

Author .....
Department of Aeronautics and Astronautics
January 30, 2008

Certified by .....
David L. Darmofal
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
David L. Darmofal
Associate Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

An Output-based Adaptive and Higher-Order Method for a Rotor in Hover

by

James M. Modisette

Submitted to the Department of Aeronautics and Astronautics
on January 30, 2008, in partial fulfillment of the
requirements for the degree of
Masters of Science in Aerospace Engineering

Abstract

A high-order discontinuous Galerkin finite element discretization and output-based adaptation scheme for the compressible Euler equations are presented and applied to an isolated rotor in hover. A simplex cut-cell mesh generation technique is used to support robust and autonomous creation of higher-order meshes. The calculations are performed using a parallel implementation of the DG discretization and the results are compared to experimental data. As accurate simulation of rotorcraft wakes and blade-vortex interactions continues to be a challenge, the output-based adaptation scheme is used with thrust as the output of interest to refine the mesh. The result is a solution with less than three million degrees of freedom that is capable of preserving a rotor tip vortex for three and a half revolutions.

Thesis Supervisor: David L. Darmofal
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

I would like to express my gratitude to the many people who have made this thesis possible. First, I would like to thank my advisor, Professor David Darmofal, for giving me the opportunity to work with him. I am grateful for his guidance, inspiration and encouragement, and I look forward to continuing my studies under his supervision. Of course, this work would not have been possible without the tireless efforts of the entire ProjectX team (Garrett Barter, Laslo Diosady, Krzysztof Fidkowski, Bob Haimes, Josh Krakos, Todd Oliver, Mike Park, and David Walfisch). There is no way this work would ever have been completed without their help. Of course, I would like to thank my family for their constant support, without which I am sure I would not have gotten this far. And, last but not least, I would like to thank Mora for her constant support and always being there to make sure my head was on straight.

This work was partially supported by funding from The Boeing Company with technical monitor Dr. Mori Mani.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The compressible Euler equations have been used extensively to model an isolated rotor in hover [2, 6, 21, 31, 41–43, 49]. Computations based on the Euler equations have matched experimental results for static pressure and sectional thrust and have been used to optimize a rotor geometry to reduce horsepower, vibrations, and rotor weight[1]. Accurate computations for an isolated rotor in hover are feasible because the influence of the rotor tip vortex on loading is small.

For more complex problems such as full rotorcraft simulations or rotors in descent, accurate load estimates require preservation of the rotor tip vortex over multiple revolutions. The importance of the successful modeling of the tip vortex is due to the significance of wake modeling for predicting acoustics, performance, and vibrational loading. For a typical finite volume flow solver, the element size required to accurately model the small scale vortex core demands a grid which is impractically large [10].

An alternate approach is to explicitly include a vortex core model in the numerical scheme[10, 35, 47, 48]. Examples of this approach include vortex embedding and vortex confinement. Vortex embedding was developed from potential methods and allows for rotational flow only on the wake structure which is then convected downstream using the potential. While vortex embedding has been shown to provide good results for hover, it is hard to implement for forward flight. The goal with vorticity confinement is to achieve a wake-preserving solution through the addition of an acceleration term

in the momentum equations. The acceleration term is non-zero in the defined vortical regions of the flow and acts in the tangential direction of the flow mimicking vorticity. Therefore, inner vortical regions exist only insofar as they contain a non-dissipating circulation due to the acceleration term in small regions, but they are not resolved through small mesh elements. The inner vortical regions have no physical meaning other than their circulations. Vorticity confinement has been shown to successfully model wake around the body of the helicopter [48].

Higher-order and adaptive methods have also been applied to an isolated rotor in hover and have been able to improve the preservation of the tip vortex. Hariharan et al. [20] relied on an overset grid technique to support their fifth-order or seventh-order finite difference method, while Boelens et al. [6] used linear polynomials to model the solution in a second-order accurate Discontinuous Galerkin (DG) method with adaptation based on gradients of vorticity to preserve the vortex core.

This work also utilizes a DG discretization and extends previous work by applying higher-order elements (quadratic polynomials) and an output-based adaptive method to the rotor in hover problem. This represents the first application of output-based adaptation to rotor flow computations. Output-based adaptation estimates the impact of local discretization errors on an output and then adapts the mesh to reduce these errors [19, 36]. Engineering applications have included lift and drag for two-dimensional and three-dimensional flows [16, 32, 45], sonic boom problems [23, 25], and forces on re-entry vehicles [30]. Output error estimation techniques require the solution of an adjoint problem, where the adjoint relates the output error to the local residual. Chapter 5 presents this technique in more detail.

To employ adaptation, the ability must exist such that, given an error estimate, a new mesh can be reliably and autonomously generated. In order to support the DG discretization, the output mesh must have higher-order geometry. To this end, a simplex cut-cell technique first presented by Fidkowski [14] and detailed in Chapter 4 is used. With the cut-cell technique, a higher-order mesh is cut out of a simplex background mesh that does not conform to the geometry. The cut-cell technique

relies on the ability to accurately and autonomously intersect the geometry with the background mesh and generate integration rules over the arbitrarily-shaped elements that result from the intersection problem. Although the discretization is more costly when using the cut-cell technique, cut-cells permit easy adaptation since elements do not conform to the geometry.

Output-based adaptation results are presented in Chapter 6. The results demonstrate the benefit of combining a higher-order discretization with output-based adaptation. The adaptation cycle is begun on a $p = 0$ mesh with ten thousand degrees of freedom and autonomously a $p = 2$ solution is produced containing less than three million degrees of freedom and capable of preserving three and a half revolutions of the rotor tip vortex. Successful matching of experimental data [11] for an extruded NACA 0012 rotor is achieved.

Chapter 2

Model Equations

2.1 Euler Equations for a Rotating Reference Frame

The Euler equations, which govern the three-dimensional, inviscid, unsteady, compressible problem of interest, can be formulated in the rotating frame of the rotor such that the rotor in hover problem is steady. Two common choices for flow variables exist in the rotating reference frame. The first choice is to solve for the velocities relative to the rotating reference frame, $\vec{u}_r$. In this work, the x and y axes are in the chordwise and spanwise directions rotating around the z axis at a constant rate Ω. In this case, the Euler equations are,

$$\begin{aligned} \frac{\partial U_r}{\partial t} &+ \nabla \cdot F_r = S_r \\ F_r &= f_r \hat{i} + g_r \hat{j} + h_r \hat{k} \end{aligned} \tag{2.1}$$

where,

$$U_r = \begin{bmatrix} \rho \\ \rho u_r \\ \rho v_r \\ \rho w_r \\ \rho E \end{bmatrix} \quad f_r = \begin{bmatrix} \rho u_r \\ \rho u_r^2 + p \\ \rho u_r v_r \\ \rho u_r w_r \\ \rho u_r H \end{bmatrix} \quad g_r = \begin{bmatrix} \rho v_r \\ \rho v_r u_r \\ \rho v_r^2 + p \\ \rho v_r w_r \\ \rho v_r H \end{bmatrix} \quad h_r = \begin{bmatrix} \rho w_r \\ \rho u_r w_r \\ \rho w_r v_r \\ \rho w_r^2 + p \\ \rho w_r H \end{bmatrix}$$

$$\begin{aligned} E &= \frac{p}{(\gamma-1)\rho} + \frac{1}{2}(u_r^2 + v_r^2 + w_r^2) - \frac{1}{2}(u_\Omega^2 + v_\Omega^2) \\ H &= E + \frac{p}{\rho} \\ u_\Omega &= -\Omega y \\ v_\Omega &= \Omega x \end{aligned}$$

The source term, S_r, comes from the Coriolis accelerations, $-\rho\Omega \times (\Omega \times \bar{x}) - 2\rho\Omega \times \bar{u_r}$, appearing in the momentum equation and is

$$S_r = \begin{bmatrix} 0 \\ \rho\Omega^2 x + 2\rho\Omega v_r \\ \rho\Omega^2 y - 2\rho\Omega u_r \\ 0 \\ 0 \end{bmatrix}$$

An inconvenience with this formulation of the Euler equations is the inclusion of $\frac{1}{2}(u_\Omega^2 + v_\Omega^2)$ in the relationship between pressure and energy. A more convenient approach is to solve for the absolute velocities (i.e. the velocities relative to a fixed frame). In this case, the solution velocities are $\vec{u} = \vec{u_r} + \Omega \times \vec{x}$,

$$u = u_r - \Omega y, \quad v = v_r + \Omega x, \quad w = w_r. \tag{2.2}$$

Combining Equations (2.1) and (2.2), the Euler equations solving for absolute velocities in a rotating reference frame are

$$\begin{aligned} \frac{\partial U}{\partial t} &+ \nabla \cdot \mathcal{F} = S \\ \mathcal{F} &= \left(f\hat{i} + g\hat{h} + h\hat{k} \right) - (\Omega \times \vec{x})\, U \end{aligned} \tag{2.3}$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{bmatrix} \quad g = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{bmatrix} \quad h = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ \rho wH \end{bmatrix} \quad S = \begin{bmatrix} 0 \\ \rho \Omega v \\ -\rho \Omega u \\ 0 \\ 0 \end{bmatrix}$$

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

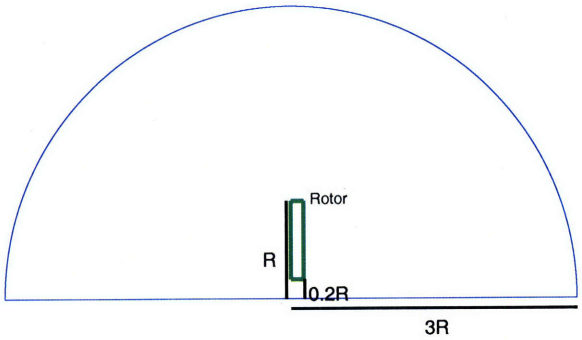
This form of the Euler equations is applied in this work.

2.2 Outer Boundary Condition for a Rotor in Hover

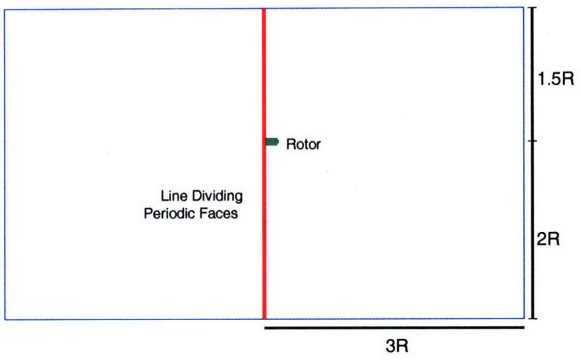
The background domain utilized in this work is shown in Figure 2-1. The boundary conditions on the outer boundaries are modeled by a jet combined with a potential sink (see Figure 2-2). This model has been shown to allow for much smaller domains [2, 41–43].

The jet velocity at the outflow is given by

$$u_e = -2M_t \sqrt{\frac{C_t}{2}},$$



(a) Top view of domain



(b) Side view of domain

Figure 2-1: Computational domain

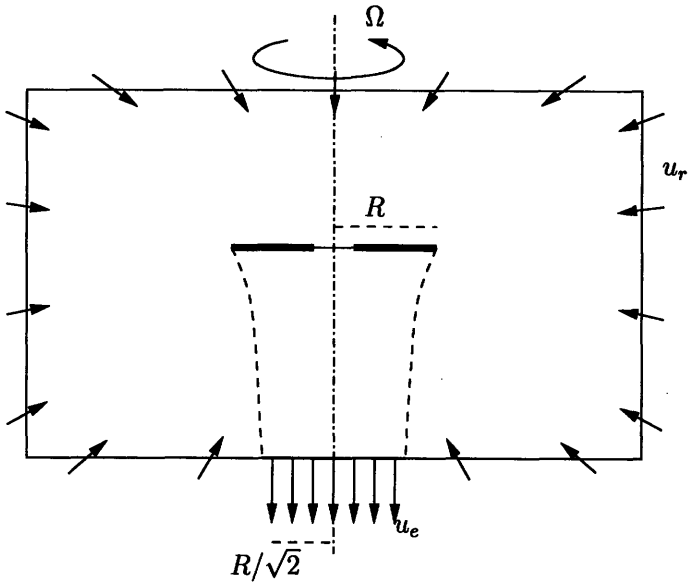


Figure 2-2: Jet-sink boundary conditions for an idealized hovering rotor system

where M_t is the tip Mach number and C_t is the coefficient of thrust. The radius of outflow velocity is $R/\sqrt{2}$ [41, 43]. A point sink centered at the rotor hub is used to model the inflow. The strength of the sink is set to conserve mass with the radial inflow velocity given by

$$u_r = -\frac{M_t}{4}\sqrt{\frac{C_t}{2}}\left(\frac{R}{r}\right)^2$$

To improve the jet-sink model for higher-order discretizations, the velocity discontinuity in the outflow boundary conditions at $R/\sqrt{2}$ where u_e goes from $u_e = -2M_t\sqrt{\frac{C_t}{2}}$ to 0, must be removed. The discontinuity can be smoothed by making the transition continuous using a sine curve over a 2Δ wide region surrounding $\frac{R}{\sqrt{2}}$. In the range $\frac{R}{\sqrt{2}} - \Delta < r < \frac{R}{\sqrt{2}} + \Delta$ u_e can be defined as,

$$u_e = M_t\sqrt{\frac{C_t}{2}}\left[\sin\left(\frac{r - \frac{R}{\sqrt{2}}}{\Delta}\frac{\pi}{2}\right) - 1\right].$$

The resulting jet velocity can be seen in Figure 2-3 for $\Delta = 0.1R$.

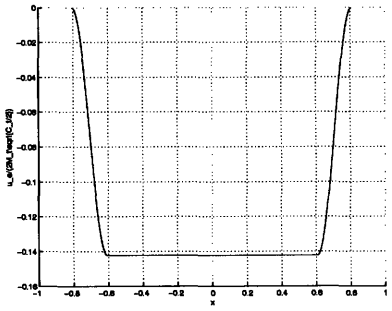


Figure 2-3: Plot of $u_e(r)$ showing smoothing at $R/\sqrt{2}$.

In order to specify all five components of the state vector at the outer boundaries, the pressure and density are determined using the isentropic relationships between the flow at the boundary and the stagnation flow at infinity. Specifically, setting $p_t, \rho_t = 1$, the pressure and density are,

$$\begin{aligned} p_b &= \left(\frac{1}{1 - \frac{\gamma-1}{2}\frac{u_r^2}{\gamma}} \right)^{\frac{\gamma}{1-\gamma}} \\ \rho_b &= p_b^{\frac{1}{\gamma}} \end{aligned}$$

The boundary condition is applied through the inviscid flux function, $\mathcal{H}_b(u_h^b, u_h^+, \hat{n})$ in the same manner as the internal faces except that u_h^b is the state vector composed from the radial velocity, p_b, and ρ_b.

2.3 Flow Tangency Boundary Condition

The boundary condition used in this work for the rotor itself is flow tangency. For a flow tangency boundary condition, the flux is computed by setting the relative velocity components normal to the wall equal to zero. Thus,

$$\vec{u}_r^b \cdot \hat{n} = 0,$$

where $\vec{u}_r^b$ is the velocity relative to the rotating reference frame computed from the interior state. Therefore the boundary state from which the flux, $\mathcal{F}$, is computed as

$$u_h^b = \begin{pmatrix} \rho^b \\ \rho u^b - \rho^b \left(\vec{u}_r^b \cdot \hat{n}\right) n_1 \\ \rho v^b - \rho^b \left(\vec{u}_r^b \cdot \hat{n}\right) n_2 \\ \rho w^b - \rho^b \left(\vec{u}_r^b \cdot \hat{n}\right) n_3 \\ \rho E^b \end{pmatrix}.$$

Chapter 3

Discretization and Solution Method

3.1 Discontinuous Galerkin Discretization

The discontinuous Galerkin (DG) discretization of Equation (2.3) is created by choosing a tessellation, T_h, of the computational domain, Ω. The tessellation is comprised of tetrahedral elements, κ. At any instant in time, a solution is sought in $\mathcal{V}_h^p$, the space of piecewise polynomials of order p, which satisfies a weak form of the equations. Using the approximate solution u_h in $\mathcal{V}_h^p$ and an arbitrary test function $v_h \in \mathcal{V}_h^p$, the weak form is:

$$\sum_{\kappa \in T_h} \int_\kappa v_h^T \frac{\partial u_h}{\partial t} + \mathcal{R}(v_h, u_h) = 0 \tag{3.1}$$

where,

$$\begin{aligned} \mathcal{R}(v_h, u_h) = \; & - \sum_{\kappa \in T_h} \int_\kappa \nabla v_h^T \cdot \mathcal{F}(u_h) + \int_{\Gamma_i} (v_h^+ - v_h^-)^T \mathcal{H}_i(u_h^+, u_h^-, \hat{n}) \\ & + \int_{\partial\Omega} v_h^{+^T} \mathcal{H}_b(u_h^b, u_h^+, \hat{n}) - \sum_{\kappa \in T_h} \int_\kappa \mathcal{S}(u_h)^T v_h \end{aligned} \tag{3.2}$$

In Equation (3.2), Γ_i is the combination of all interior faces while $\partial\Omega$ is the domain boundary. $()^+$ and $()^-$ represent values taken from either side of a face and $\hat{n}$

represents the unit normal pointing outward from the $()^+$ element to the $()^-$ element. $\mathcal{H}_i(u_h^+, u_h^-, \hat{n})$ and $\mathcal{H}_b(u_h^b, u_h^+, \hat{n})$ are the flux functions for the interior and boundary faces, respectively. In this work, Roe numerical flux functions [37] approximating $\mathcal{F} \cdot \hat{n}$ are used on the interior faces and all outer boundary faces.

For this work, a nodal Lagrange basis which spans the space $\mathcal{V}_h^p$ is chosen. Further details of the basis may be found in Fidkowski et al. [18]. The solution vector, $u_h(x,t)$, is then given as a linear combination of the bases,

$$u_h(x,t) = \sum_i \mathbf{u}_{h_i}(t) v_{h_i}(x). \tag{3.3}$$

Then, Equation (3.1) can be written in semi-discrete form as

$$\mathcal{M}_h \frac{d\mathbf{u}_h}{dt} + \mathcal{R}_h(\mathbf{u}_h) = 0, \tag{3.4}$$

where $\mathcal{M}_h$ is the mass matrix given by

$$\mathcal{M}_{h_{ij}} = \int_\Omega v_{h_i}^T v_{h_j} dV. \tag{3.5}$$

An important feature of the mass matrix is that it is block-diagonal because the basis functions are zero outside of each element.

3.2 Backward Euler Method

To integrate Equation (3.4) in time, a Newton-like method, equivalent to taking a single linear step at each iteration of a backward Euler scheme, is used. A single iteration is given by

$$\mathbf{U}_h^{m+1} = \mathbf{U}_h^m - \left(\frac{1}{\Delta t} \mathcal{M}_h + \frac{\partial \mathcal{R}_h}{\partial \mathbf{U}_h} \right)^{-1} \mathcal{R}_h(\mathbf{U}_h^m). \tag{3.6}$$

A steady state solution is obtained using this scheme and increasing the time step,

Δt, such that $\Delta t \to \infty$. Directly setting $\Delta t = \infty$ is equivalent to using Newton's method to solve $\mathcal{R}_h(\mathbf{U}_h) = 0$. This approach is unlikely to succeed if the initial guess is too far from the actual solution. Thus, Equation (3.6) is used so that intermediate solutions approximate physical solutions in a time evolution of the flow and convergence is more likely.

3.3 Linear Solution Method

The time-marching integration scheme given in Equation (3.6) involves the solution to a large system of linear equations in the form $A\mathbf{x} = \mathbf{b}$ at every time step, where

$$A = \frac{1}{\Delta t}\mathcal{M}_h + \frac{\partial \mathcal{R}_h}{\partial \mathbf{U}_h}, \qquad \mathbf{x} = \Delta \mathbf{U}_h^m, \qquad \mathbf{b} = \mathcal{R}_h(\mathbf{U}_h^m). \tag{3.7}$$

The matrix A is commonly referred to as the Jacobian matrix. For the DG discretization, the Jacobian matrix has a block-sparse structure with N_e block rows of size n_b, where N_e is the number of elements in the tesselation T_h and n_b is the number of unknowns per element. In this case $n_b = n_s * n_m$, where n_s is the size of the state vector and n_m is the number of modes per state. n_m is a function of the solution order, p, and the spatial dimension, as shown in Table 3.1. The block rows of the Jacobian matrix contain a nonzero diagonal block, corresponding to the coupling between states within each element, and n_f off-diagonal blocks, corresponding to the coupling between states of neighboring elements. n_f is the number of faces per element (for 2D and 3D, n_f is 3 and 4 respectively). When the time step, Δt, in Equation (3.6) is small the Jacobian matrix is dominated by the block diagonal and the linear system is relatively easy to solve iteratively. Unfortunately, as the time step increases, the coupling between elements becomes increasingly important and the linear system becomes more difficult to solve.

Due to the size of the Jacobian matrix and its block-sparse structure, an iterative method is used to solve the linear system. As the Jacobian is non-symmetric, a restarted GMRES algorithm is used [12, 38, 44]. Restarted GMRES finds an ap-

p	n_m, 2D	n_m, 3D
0	1	1
1	3	4
2	6	10
p	$\frac{(p+1)(p+1)}{2}$	$\frac{(p+1)(p+2)(p+3)}{6}$

Table 3.1: Number of modes per element, n_m, for a given solution order and spatial dimension

proximated solution, $\tilde{\mathbf{x}}$, in the Krylov subspace, $\mathcal{K} = \{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots A^{n-1}\mathbf{b}\}$, that minimizes the L^2 norm of the linear residual $\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{x}}$. The convergence of the GMRES algorithm depends greatly on the eigenvalues of A [24, 38, 44]. Therefore, to improve the convergence a preconditioner is used to transform the linear system from $A\mathbf{x} = \mathbf{b}$ to a related system $P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$. In this work, an in-place Block-ILU(0) factorization with line reordering [13] of the Jacobian matrix is used as the preconditioner for the GMRES algorithm. Based on the success of implicit line solvers for both finite volume and DG discretizations [15, 18, 28, 29], a matrix reordering algorithm based on lines of maximum coupling was developed by Diosady [13]. The reordering algorithm involves the creation of lines of maximum coupling in the flows as presented by Fidkowski et. al [18]. The elements within A are then reordered in the order that they are traversed along each line.

In order to support the large number of elements used in 3D modeling and the resulting linear system, a parallel implementation is necessary. The parallel implementation [12] involves partitioning the computational grid across multiple processors. As each processor maintains all elements in a partition, the partitioning is based on breaking up the grid to minimize the number of faces dividing partitions, while attempting to balance the number of elements on each processor. In particular, it is important to balance the number of cut-cell elements due to their larger number of quadrature points. Each processor also maintains ghosted data, corresponding to neighboring elements on other partitions that are required for the local computation of the residual and Jacobian matrix. The ghosted states are updated from the ap-

propriate partition at the beginning of each residual evaluation. Communication is performed using the Message Passing Interface (MPI).

Chapter 4

Simplex Cut Cells

4.1 Introduction

For a higher order discretizations, the geometry must be represented at a higher order. Furthermore, DG discretizations of the Euler equations are known to behave poorly even for $p = 1$ solutions when the geometry is only represented linearly [4]. Many issues arise in the generation of grids with higher-order geometry information [14]. One of the most common problems when attempting to create a higher-order mesh out of a linear mesh occurs when the higher-order surface pushes through an opposing face as in Figure 4-1. A method to tackle the problem of generating higher-order meshes is the cut cell method. Purvis and Burkhalter [34] were the first to consider a cut-cell method. They started with a structured Cartesian mesh that did not conform to the geometry and simply "cut" the geometry out. Cut cells allow the grid generation process to become automated, taking a process which previously dominated the solution procedure and making it a preprocessing step. However, with the cut-cell method the body must be intersected with the background mesh and the solver must be able to discretize on the arbitrarily shaped cells that result from this intersection. Purvis and Burkhalter's method used rectangular/box shaped cells from which the geometry was cut out in a piecewise linear fashion. Although the linear intersections did not provide higher-order geometry, Purvis and Burkhalter laid down the building blocks for future

work with Cartesian cut cells.

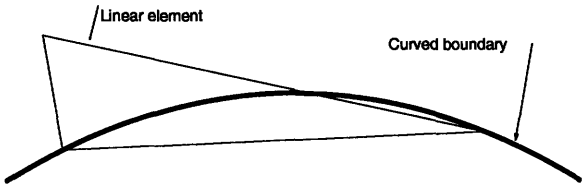


Figure 4-1: Example of a curved boundary pushing threw the interior edge of a boundary element. Attempting to curve the boundary edge of the element results in a negative Jacobian in the mapping of the reference triangle to the curved element and leads to an invalid mesh.

Unfortunately, Cartesian meshes add the constraint that the background mesh must be isotropic and aligned to a fixed coordinate system. This results in a mesh which may be far from ideal for many fluid dynamics problems due to the presence of boundary layers, shocks, wakes, etc. To introduce anisotropic meshes, the background mesh can be changed to a mesh of simplices (triangles or tetrahedra). Fidkowski [14] introduced the idea of using an unstructured simplex mesh for the background mesh. As in the Cartesian method, the computational mesh is built in a preprocessing step by intersecting the background mesh with the geometry.

4.2 Geometry Definition

A discussion of the simplex cut-cell method is included here, but for more details consult Fidkowski[14]. Three obstacles exist for the simplex cut-cell method: the geometry definition, the intersection problem between the geometry definition and the background mesh, and the ability to integrate over arbitrarily shaped cut cells.

The geometry definition must be:

- Watertight
- Easy to construct from a CAD geometry definition

- Feasible to intersect with tetrahedra
- Accurate enough to efficiently support higher-order solutions

Many choices of geometry definitions exist which meet the above requirements, but the method chosen was to subdivide the geometry into quadratic patches. Quadratic patches were selected because they greatly reduce the number of patches needed to accurately represent the geometry when compared to linear patches and allow for the intersections of the patch and tetrahedron to be solved for analytically. Figure 4-2 shows an example of a quadratic surface representation of the rotor.

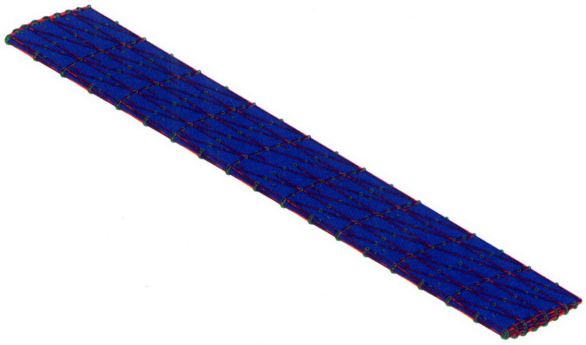


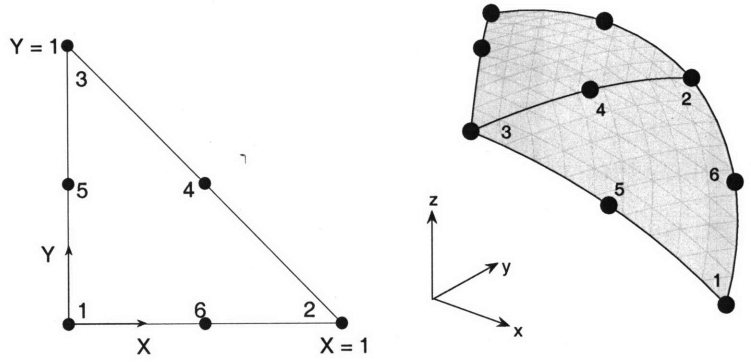
Figure 4-2: Example showing the quadratic patch representation of the rotor studied in this work.

Evaluating the surface of a quadratic patch involves storing the 3D coordinates of the six nodes of a quadratic patch and using quadratic interpolation to represent the surface. Figure 4-3 shows a pair of quadratic patches in 3D space as well as the reference space of a single patch and the node numbering used.

The surface of the quadratic patch can be written as

$$\mathbf{x}(\mathbf{X}) = \sum_{j=1}^{6} \phi_j(\mathbf{X})\mathbf{x}_j, \tag{4.1}$$

where $\mathbf{x} = [x, y, z]^T$ is a vector in 3D coordinates on the surface of the patch, $\mathbf{X} = [X, Y]^T$ is a vector in the 2D reference space of the patch, $\mathbf{x}_j$ represents the 3D, physical



(a) Reference triangle (b) Two adjacent patches

Figure 4-3: Example of two patches in physical space with a patch in reference space showing the node numbering (taken from [14]).

space, coordinates of the six nodes and the $\phi_j(\mathbf{X})$ are quadratic Lagrange interpolation functions. To simplify the evaluation of the surface, the Lagrange interpolation functions can be written as $\phi_j = R^T P_j R$, where $R = [X, Y, 1]^T$ and P_j is a 3×3 matrix representing the Lagrange polynomials. For example ϕ_1, initially written as

$$\phi_1(\mathbf{X}) = 1 - 3X - 3Y + 2X^2 + 4XY + 2Y^2$$

can also be written as

$$\phi_1(\mathbf{X}) = \begin{bmatrix} X & Y & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & -3/2 \\ 2 & 2 & -3/2 \\ -3/2 & -3/2 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

The output of the intersection problem is a cut-cell mesh for the computational domain computed from the background mesh, where elements completely contained within the geometry are removed, elements completely contained in the computational domain are untouched, and elements intersecting the geometry are appropriately cut. Figure 4-4 shows an example of an element cut by the surface geometry. The upper half of

the tetrahedron is within the computational domain and forms the cut cell. The lower half of the element is discarded since it lies within the geometry.

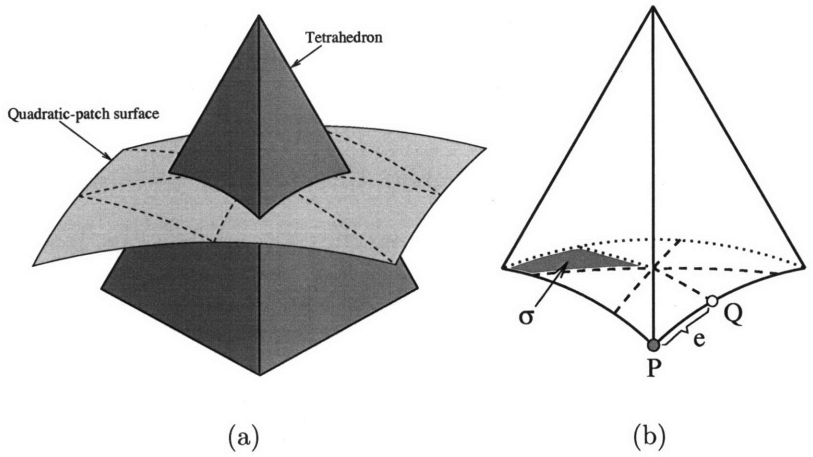


Figure 4-4: Example of a single cut tetrahedron and the resulting "wire frame" construction of the cut cell (taken from [14]).

Figure 4-4 also alludes to a key portion of the cutting procedure. In Figure 4-4 (b) a "wire frame" of a cut element exists. This "wire frame" is composed of zero-dimensional nodes and one-dimensional curves which result from the intersection of the patches and the background tetrahedron. The one-dimensional curves are built from the zero-dimensional intersection nodes, while the two-dimensional surfaces are assembled from the one-dimensional wire frames. Finally, the two-dimensional surfaces are assembled to make the cut cell.

4.3 Geometry-Mesh Intersection

One of the reasons quadratic patches were selected is that the intersection between them and a tetrahedron face is a portion of a conic section (ellipse, parabola, hyperbola, etc.) within the reference space of the patch. Consider the face of a tetrahedron. For a given point $\mathbf{x}$ to be inside the tetrahedron or on a face of the tetrahedron, it

must satisfy

$$(\mathbf{x} - \mathbf{q}_f) \cdot \mathbf{n}_f \leq 0, \tag{4.2}$$

where $\mathbf{q}_f$ is a point on the tetrahedron face f and $\mathbf{n}_f$ is the outward pointing normal again for the face f, for all four of the tetrahedron faces. If the above inequality is not satisfied for any of the faces of a tetrahedron the point lies outside the tetrahedron. This test can be used on the quadratic patches themselves as, from Equation (4.1), the points on the quadratic patches can be expressed as

$$\mathbf{x} = \sum_{j=1}^{6} \phi_j(\mathbf{X})\mathbf{x}_j = \sum_{j=1}^{6} \left(R^T P_j R\right) \mathbf{x}_j. \tag{4.3}$$

If Equation (4.3) is substituted in Equation 4.2 the result is

$$R^T \left(\sum_{j=1}^{6} (\mathbf{x}_j \cdot \mathbf{n}_j) P_j \right) R - \mathbf{q}_j \cdot \mathbf{n}_j \leq 0$$

Then using the fact that $R = [X, Y, 1]^T$ and defining $E_1 = [0,0,0;0,0,0;0,0,1]$ the above inequality can be written in quadratic form in the reference space of the patch as

$$\begin{aligned} R^T S_f R &\leq 0, \\ S_f &\equiv \sum_{j=1}^{6} (\mathbf{x}_j \cdot \mathbf{n}_j) P_j - (\mathbf{q}_f \cdot \mathbf{n}_f) E_1. \end{aligned} \tag{4.4}$$

Equation (4.4) must again be satisfied for all four of the faces of the tetrahedron for a reference patch point to lie inside the tetrahedron or on a face. Figure 4-5 shows the conic sections that result from mapping the faces of a tetrahedron into the reference space of a patch.

Since Equation (4.4) is an inequality there is one side of each conic section which

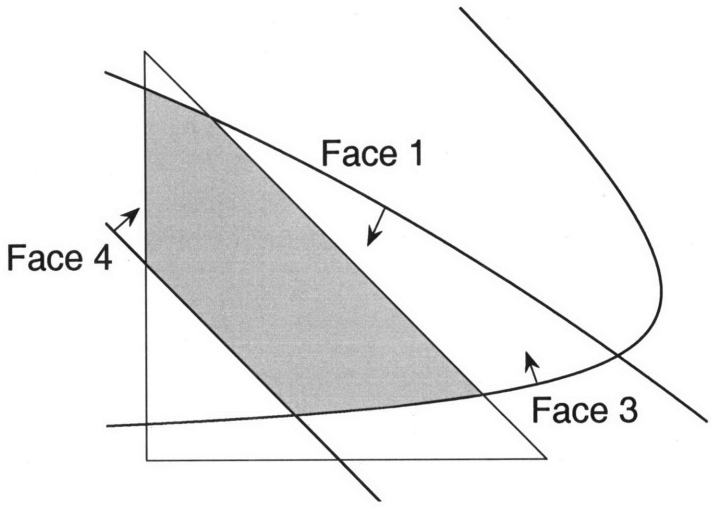


Figure 4-5: Example of a patch in reference space showing the conics which result from mapping the tetrahedron faces from physical space to the reference space of the patch (taken from [14]).

represents the valid portion of the patch in reference space. In Figure 4-5, this is represented with arrows attached to the conic sections pointing into the valid region of the patch. Also, note that any area outside the reference triangle is not on the patch. Therefore, the shaded region represents the portion of the patch that intersects the tetrahedron.

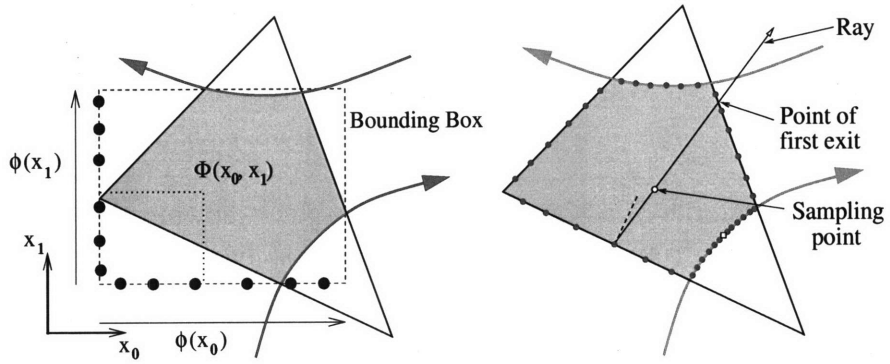
4.4 Cut-Cell Integration

The final obstacle for implementing the cut-cell mesh generation technique is the ability to accurately integrate on the arbitrarily cut elements which result from the intersection problem. The higher-order DG discretization requires accurate integration over the volume of the elements and area of the faces. The idea for integrating over the cut element's faces is to "speckle" sampling points in the area and apply the divergence theorem to compute integration weights associated with each point. These "speckled" points are a randomly chosen set of N_{quad} points, $\mathbf{x}_q$, which are given weights, w_q,

therefore enabling quadrature like evaluation of integrals, such that

$$\int_{\Omega} f(\mathbf{x}) d\Omega \approx \sum_{q=1}^{N_{quad}} w_q f(\mathbf{x}_q),$$

where $f(\mathbf{x})$ is an arbitrary integrand and Ω is an arbitrary enclosed area or volume, like the one in Figure 4-6.



(a) Tensor-product Lagrange functions, $\Phi_i(x)$

(b) Sampling point selection

Figure 4-6: (a) Definition of the $\Phi_i(x)$ functions for use in defining the basis, $\zeta_i(x)$, for integrating on the irregularly-shaped shaded area. (b) Illustration of the ray-casting procedure for sampling point selection. (taken from [14]).

The quadrature weights are determined by projecting the integrand onto a set of N_{basis} higher-order basis functions $\zeta_i(\mathbf{x})$. These new basis functions are specifically chosen to allow for the evaluation of the integral $\int_{\Omega} \zeta_i(\mathbf{x}) d\Omega$ to be easy. Specifically, the chosen ζ_i are

$$\zeta_i = \nabla \cdot (\mathbf{x}\Phi_i(\mathbf{x})), \tag{4.5}$$

where the $\Phi_i(\mathbf{x})$ are higher-order Lagrange basis functions on the bounding box of the arbitrary area as seen in Figure 4-6 (a). This specific construction of ζ_i allows for the integral to be easily evaluated by applying the divergence theorem. Thus $\int_{\Omega} \zeta_i(\mathbf{x}) d\Omega$

becomes $\int_{\partial\Omega} \mathbf{n} \cdot \mathbf{x}\Phi_i(\mathbf{x})dS$, where $\partial\Omega$ is the boundary of the arbitrary area or volume, Ω, and $\mathbf{n}$ is the outward pointing normal. The boundary integral is then computed using the edge integrals. For the 2D surfaces, the 1D edge integral rules come directly from Gauss Quadrature points distributed over the edge, as can be seen in Figure 4-6(b).

The projection is performed using a least squares minimization of the projection error,

$$E^2 = \sum_{q=1}^{N_{quad}} \left[\sum_{i=1}^{N_{basis}} F_i \zeta_i(\mathbf{x}_q) - f(\mathbf{x}_q) \right]^2 .$$

F_i is found using a QR factorization of the matrix $\zeta_i(\mathbf{x}_q)$ (which is $N_{quad} \times N_{basis}$),

$$F_i = \left(R^{-1}\right)_{ij} \left(Q^T\right)_{jq} f(\mathbf{x}_q), \quad \text{where } \zeta_i(\mathbf{x}_q) = Q_{qj} R_{ji}$$

Finally, the integral of an arbitrary integrand $f(\mathbf{x})$ becomes

$$\int_\Omega f(\mathbf{x})d\Omega \approx \sum_{i=1}^{N_{basis}} F_i \int_\Omega \zeta_i(\mathbf{x})d\Omega = \sum_{q=1}^{N_{quad}} f(\mathbf{x}_q) Q_{qj} \left(R^{-}T\right)_{ji} \int_\Omega \zeta_i(\mathbf{x}_q)d\Omega. \quad (4.6)$$

$$\Longrightarrow w_q = Q_{qj} \left(R^{-}T\right)_{ji} \int_\Omega \zeta_i(\mathbf{x})d\Omega$$

Therefore, for each arbitrarily cut element the quadrature-like weights can be evaluated once, in a preprocessing step, and can then be used to evaluate any integrand $f(\mathbf{x})$.

As noted, the sampling points are chosen randomly in the interior of the area or volume, however, their selection greatly influences the conditioning of the QR factorization of $\zeta_i(\mathbf{x}_q)$ used to solve for w_q. Many methods exist for selecting the sample points as the only real requirement is that they must be inside the area/volume. One approach would be to uniformly fill the bounding box and then determine if each point was inside or outside the area/volume. This would have the benefit of preventing clumping of sampling points which directly leads to poor conditioning. Unfortunately, this method becomes extremely inefficient when the area/volume is small compared to

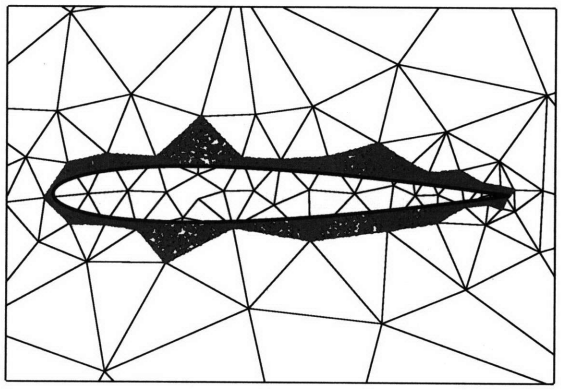
that of the bounding box. Therefore, in this work the method used is to cast rays from the boundary sampling points which are the 1D quadrature points bounding areas or 2D sampling points bounding volumes. Figure 4-6 (b) shows an example of one of these rays. The sampling point is then randomly cast along the ray between its origin and where it exits the area/volume. Example sampling points can be seen in Figure 4-7 for a 2D cut-cell computation for a NACA 0012 airfoil. Figure 4-7 also exemplifies the over sampling that is currently used to ensure accurate integration within the cut elements by spreading points throughout the element. The selection of the sampling points is an area of future work for the simplex cut-cell method.

Integration over the cut element volumes is simply an extension of the integration procedure over the cut element faces. Although, the sampling points over the area faces must be derived first so as to give origins to the ray casting method for "speckling" points within each element's volume.

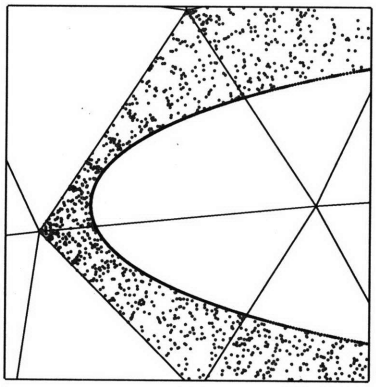
4.5 Merging Cut Cells

During the course of this research, poor iterative convergence was observed in regions of the mesh where small cut cells neighbor larger cells. Small cut cells are typically formed when just the corner of a background element remains in the computational domain, like the 2D example in Figure 4-8(a). In 3D cut-cell cases, the ratio of a small cut cell volume to the volume of one of its neighbors can be $\mathcal{O}(10^{-12})$.

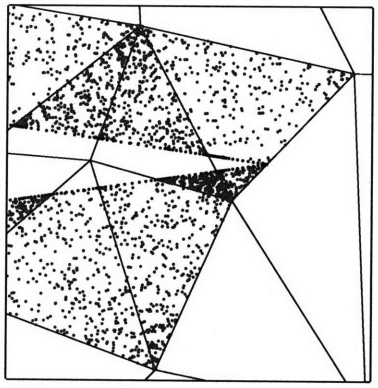
The small cut cells adversely manifest themselves by generating large updates from the solution to the linear system, $A\mathbf{x} = \mathbf{b}$. Even when the residual is $\mathcal{O}(10^{-6})$, it is common for some components of $\mathbf{x} = \Delta\mathbf{U}_h$ to be $\mathcal{O}(1)$. When these large updates are used to advance the state, limits are placed on the fraction of the update to prevent the state from going unphysical. In another attempt to prevent the solution from becoming unphysical, Δt can be lowered. The result of lowering Δt is that Newton-like convergence is never achieved and the simulations become extremely computationally expensive.



(a) Entire airfoil

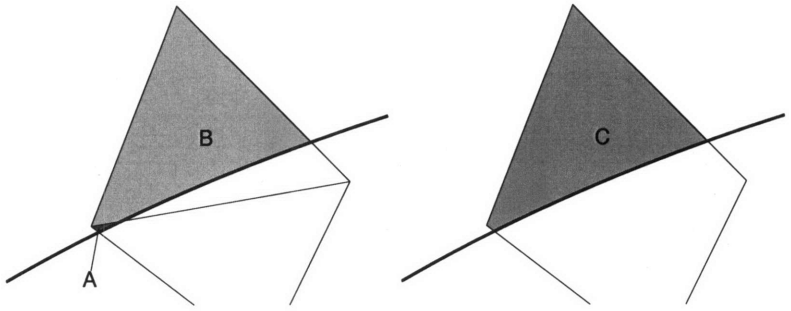


(b) Leading edge



(c) Trailing edge

Figure 4-7: Example of "speckled" points used for integration rules for a NACA 0012 airfoil (taken from [14]).



(a) Example small cut cell (b) Example merged cut cell

Figure 4-8: Example of a small cut cell, A, and the merged cut cell, C.

An approach to improve the discretization for small cut cells is to merge the small cut cell with a large neighbor. For example, returning to Figure 4-8(a), if the internal face between elements A and B is removed, the result is the new cut element C in Figure 4-8(b).

In practice using this method of merging small cut-cell elements has improved the convergence of 3D Euler cases. Figure 4-9 shows the residual history for a $p = 2$ solution for the rotor in hover case of interest in this work. With no merging the residual drops to around 5×10^{-7} and then hovers there for quite a while and in this case happens to eventually converge. If the volume ratio is defined as

$$VR = \frac{\text{volume of element } \kappa}{\text{volume of largest neighbor to } \kappa},$$

then, as Figure 4-9 shows, if every element with a volume ratio less than 10^{-6} is merged into its neighbors the residual converges to machine zero with few problems. For the extent of this work all elements with a volume ratio less than 10^{-6} are merged.

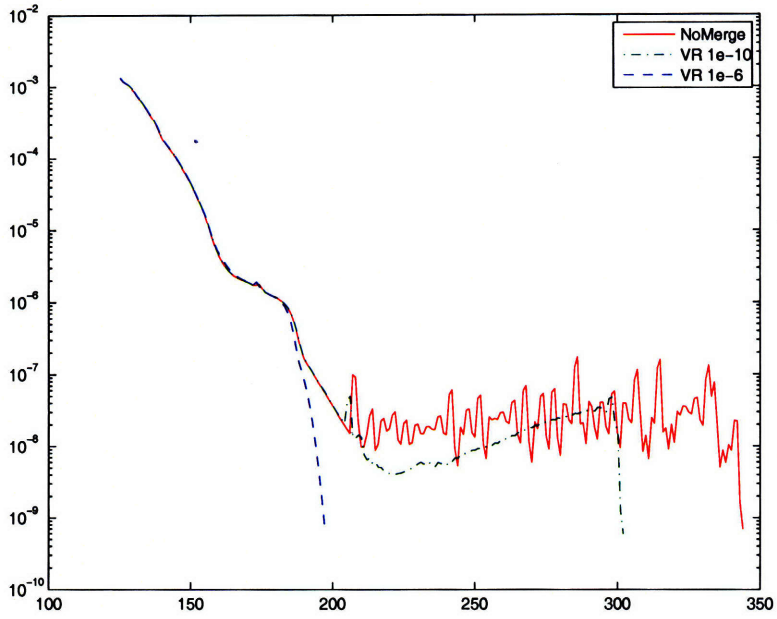


Figure 4-9: Plot of convergence history for solutions using different volume ratios to base merging on.

Chapter 5

Output-based Adaptation

5.1 Output-based Error Estimation

The purpose of adaptation is to minimize the cost of a computational simulation by locally enriching the computational domain in regions which most adversely affect the accuracy of the final solution while coarsening the grid in more benign regions. One of the difficulties in executing adaptation is a lack of reliable error indicators. Many schemes rely on adapting on a variety of physical flow features, such as stagnation points, shock waves, or vortex cores. These schemes often look for large flow gradients and assume that areas of large gradients correspond to regions of large errors in the final solution. One problem with this approach is that refinement of a specific flow feature does not guarantee any level of global error.

This work relies on an output-based error estimation method that has been studied extensively in the literature [3, 5, 19, 22, 26, 27, 33, 46]. The method uses an error estimation and grid adaptation scheme specifically designed for improving the accuracy of a functional output, in this case the lift caused by the rotor. Output-based error estimation improve upon local error criteria because it captures propagation effects, by linking local residuals to outputs through the use of the adjoint solution.

The adjoint can be interpreted as a Green's function that relates the residual of a PDE to an output derived from the solution to the PDE. Defining the PDE solution

symbolically as $\mathcal{R}(\mathbf{u}) = \mathbf{f}$ then for a perturbed solution $\mathbf{u} + \delta\mathbf{u}$, the change in the output $\mathcal{J}(\mathbf{u})$ is

$$\mathcal{J}(\mathbf{u} + \delta\mathbf{u}) - \mathcal{J}(\mathbf{u}) = \int_{\Omega} \psi \left(\mathcal{R}(\mathbf{u} + \delta\mathbf{u}) - \mathcal{R}(\mathbf{u}) \right)$$

where ψ is the adjoint. Through integration by parts, a PDE can be defined for the adjoint. For application to the DG discretization, a system given by a semi-linear form can be considered to find $\mathbf{u}_H \in \mathcal{V}_H$, such that

$$\mathcal{R}_H(\mathbf{u}_H, \mathbf{v}_H) = 0, \quad \forall \mathbf{v}_H \in \mathcal{V}_H,$$

where $\mathcal{V}_H$ is an appropriate finite dimensional functional space. Letting $\mathbf{u} \in \mathcal{V}$ be the exact solution of the underlying problem of interest, then for a general output of interest, $\mathcal{J}(\cdot)$, the adjoint or dual problem is: find $\psi \in \mathcal{V}$ such that

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

where $\bar{\mathcal{R}}_H$ and $\bar{\mathcal{J}}$ are mean-value linearizations given by

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

and $\mathbf{v}, \mathbf{w} \in \mathcal{V}_H + \mathcal{V}$. Thus, using $\mathbf{v} = \mathbf{u} - \mathbf{u}_H$,

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

Assuming that $\mathcal{R}_H(\mathbf{u}, \mathbf{w}) = 0,\ \forall \mathbf{w} \in \mathcal{V}_H + \mathcal{V}$, the output error can be expressed as

$$\begin{aligned} \mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) &= \bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H) \\ &= \bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \psi) \\ &= 0 - \mathcal{R}_H(\mathbf{u}_H, \psi) \\ &= -\mathcal{R}_H(\mathbf{u}_H, \psi - \psi_H), \end{aligned} \tag{5.1}$$

for all $\psi_H \in \mathcal{V}_H$. Therefore, if the exact adjoint solution is known, the error can be computed exactly by evaluating the primal residual. On the other hand, by defining the adjoint residual,

$$\bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \mathbf{w}) \equiv \bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{v}, \mathbf{w}) - \bar{\mathcal{J}}(\mathbf{u}, \mathbf{u}_H; \mathbf{v}), \quad \mathbf{v}, \mathbf{w} \in \mathcal{V}_H + \mathcal{V}, \tag{5.2}$$

the output error can also be expressed in terms of the exact primal solution as

$$\begin{aligned} \mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) &= \bar{\mathcal{R}}_H(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \psi_H) - \bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \psi_H) \\ &= \mathcal{R}_H(\mathbf{u}, \psi_H) - \mathcal{R}_H(\mathbf{u}_H, \psi_H) - \bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \psi_H) \\ &= -\bar{\mathcal{R}}_H^{\psi}(\mathbf{u}, \mathbf{u}_H; \mathbf{u} - \mathbf{u}_H, \psi_H), \end{aligned} \tag{5.3}$$

for all $\psi_H \in \mathcal{V}_H$.

Since $\mathbf{u}$ and ψ are not known in general, two approximations are used to ease the computational expense of the output error estimates. The first approximation replaces the exact errors, $\mathbf{u} - \mathbf{u}_H$ and $\psi - \psi_H$, with $\mathbf{u}_h - \mathbf{u}_H$ and $\psi_h - \psi_H$, where $\mathbf{u}_h$ and ψ_h come from a reconstruction of $\mathbf{u}_H$ and ψ_H in an enriched space $\mathcal{V}_h$. In this work, the mesh is fixed while $\mathcal{V}_h$ is constructed from $\mathcal{V}_H$ by increasing the interpolation order to $p+1$. See Lu [26] and Fidkowski[17] for more information on the reconstruction.

The second approximation replaces the exact mean value linearizations with linearalizations about $\mathbf{u}_H$. To help this approximation, ψ_H is set to the discrete adjoint solution. In more detail, ψ_H satisfies $\mathcal{R}_H^{\psi}(\mathbf{u}_H; \mathbf{v}_H, \psi_H) = 0$ for all $\mathbf{v}_H \in \mathcal{V}_H$ where

$\mathcal{R}_H^{\psi}(\mathbf{u}_H; \mathbf{v}, \mathbf{w})$ is the adjoint residual based only on linearizations about $\mathbf{u}_H$:

$$\mathcal{R}_H^{\psi}(\mathbf{u}_H; \mathbf{v}, \mathbf{w}) = \mathcal{R}'_H[\mathbf{u}_H](\mathbf{v}, \mathbf{w}) - \mathcal{J}'[\mathbf{u}_H](\mathbf{v}), \quad \mathbf{v}, \mathbf{w} \in \mathcal{V}_H + \mathcal{V}.$$

Therefore, the error can be approximated with either the primal or adjoint residual from the enriched space $\mathcal{V}_h$ as:

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) \approx -\mathcal{R}_h(\mathbf{u}_H; \psi_h - \psi_H) \tag{5.4}$$

$$\mathcal{J}(\mathbf{u}) - \mathcal{J}(\mathbf{u}_H) \approx -\mathcal{R}_h^{\psi}(\mathbf{u}_H; \mathbf{u}_h - \mathbf{u}_H, \psi_H). \tag{5.5}$$

Using these two estimates for the error, a local error indicator can be constructed on each element by averaging Equation (5.4) and (5.5). Thus, for each element κ, the error is

$$\epsilon_\kappa = \frac{1}{2}\left(\left|\mathcal{R}_h\left(\mathbf{u}_H, (\psi_h - \psi_H)|_\kappa\right)\right| + \left|\mathcal{R}_h^{\psi}\left(\mathbf{u}_H; (\mathbf{u}_h - \mathbf{u}_H)|_\kappa, \psi_H\right)\right|\right). \tag{5.6}$$

The global error is approximated as $\epsilon = \sum_\kappa \epsilon_\kappa$. This error estimate is not a bound on the global error but it has proven sufficient to drive adaptation.

5.2 Grid Adaptation

The adaptation strategy takes the localized error estimate and modifies the computational mesh in an attempt to decrease and equally distribute the error. This work uses *h*-adaptation at a constant *p*. Using this strategy does not take advantage of the benefits of *hp*-adaptation but avoids the added complexity of making regularity estimates involved in *p*-adaptation.

For three dimensional flows, the *h*-adaptation strategy consists of mesh optimization, referring to the decision on which elements to refine or coarsen and/or the amount of refinement or coarsening. The mesh optimization used in this work is based on Fidkowski [17].

Mesh optimization has critical significance for flow solutions. Too little refinement at each adaptation iteration results in an unnecessary number of iterations. On the other hand, too much refinement may result in an overly-refined mesh with an unnecessary computational cost. In order to equally distribute the error on the adapted mesh, the number of fine mesh elements, N_f, or the sum of n_κ, the number of fine mesh elements contained in element κ, over all the elements in the current mesh, must be predicted using the local error estimate. n_κ does not need to be an integer and $n_\kappa > 1$ corresponds to mesh refinement. If the current size of element κ is denoted by the reference length h^c and the requested element size is given by the length h, then n_κ can be approximated as

$$n_\kappa = \left(\frac{h^c}{h}\right)^{dim}, \tag{5.7}$$

where *dim* is the mesh dimension. The current size, h^c, is computed from the singular values of the mapping of the unit equilateral tetrahedron to the element κ.

In order to satisfy error equidistribution, each fine-mesh element is given an error of e_0/N_f, where e_0 is the global error tolerance. This means that each element κ is allowed an error of $n_\kappa e_0/N_f$. An expression for n_κ can then be obtained by relating the changes in element size to expected changes in the local error. An *a priori* estimate for the output error gives

$$\frac{\epsilon_\kappa}{\epsilon_\kappa^c} = \left(\frac{h}{h^c}\right)^{r_\kappa}, \tag{5.8}$$

where ϵ_κ^c is the current error indicator, ϵ_κ is the expected error indicator and r_κ is the expected convergence rate of the error. In this work, r_κ is assumed to be at least $p+1$ away from geometric singularities (i.e. corners). For cut-cell elements that contain geometric singularities, such as the trailing edge, and adjacent elements, the convergence rate is limited to 1. This results in the isolation of the geometric singularities in fewer adaptation iterations.

If the allowable error and the expected error are equated the resulting relationship,

$$\underbrace{n_\kappa \frac{e_0}{N_f}}_{\text{allowable error}} = \underbrace{\epsilon_\kappa^c \left(\frac{h}{h^c}\right)^{r_\kappa}}_{\textit{a priori}\text{ error estimate}}, \tag{5.9}$$

expresses $\frac{h}{h^c}$ in terms of n_k and yields a correlation between N_f and n_κ. For example, in two dimensions Equation (5.7) becomes

$$\frac{h}{h^c} = \left(\frac{1}{n_\kappa}\right)^{\frac{1}{3}}.$$

Substituting this into Equation (5.9),

$$n_\kappa \frac{e_0}{N_f} = \epsilon_\kappa^c \left(\frac{1}{n_\kappa}\right)^{\frac{r_\kappa}{3}} \quad \Rightarrow \quad n_\kappa^{1+\frac{r_\kappa}{3}} = \frac{\epsilon_\kappa^c}{e_0/N_f}.$$

Given $N_f = \sum_\kappa n_\kappa$,

$$N_f = \sum_\kappa \left[\left(\frac{\epsilon_\kappa^c}{e_0/N_f}\right)^{\frac{3}{r_\kappa+3}}\right]. \tag{5.10}$$

If all the r_κ are equal Equation (5.10) can be solved directly for N_f. Otherwise, it can be solved iteratively. With N_f known, Equation (5.9) yields n_κ, from which the h are calculated using Equation (5.7). Thus, the output error estimate in conjunction with the mesh optimization step provides the remaining piece of information necessary to fully specify the reference length, h, for all the elements in the computational domain. Since tetrahedra completely contained within the geometry are removed from the cut-cell data structure, they do not possess an error estimate or an associated metric. On these elements, a grid implied metric is used, which attempts to keep element size approximately constant.

During each adaptation iteration, mesh optimization is performed after each output-based error estimate. The adaptation cycle stops when the total error estimate is less than the requested tolerance, $\epsilon \equiv \sum_\kappa \epsilon_\kappa < e_0$.

When the error estimate is far from the target estimate, e_0, as would arise with a coarse mesh where the solution and the error estimate may have significant errors, attempting to achieve an error of e_0 in one step can be overly aggressive leading to an adaptation process with poor robustness. To alleviate this problem, the desired error can be bounded from below by $\eta_a \epsilon$ where η_a is an aggressiveness parameter. Thus, when the estimated error (scaled by η_a) is larger than the desired error, adaptation will be slowed. However, if $\eta_a \epsilon$ bounds the desired error, the convergence of the adaptive cycle could stall as the error estimate, ϵ, approaches the requested error tolerance, e_0.

To address these issues the requested error level, e_0, in Equation (5.9) is modified to $\tilde{e_0}$,

$$\tilde{e_0} = \max(\eta_a \epsilon, \eta_t e_0),$$

with both $0 < \eta_a < 1$ and $0 < \eta_t < 1$.

A value of η_a close to one indicates unaggressive adaptation and typically leads to a large number of adaptation interations to converge, while a value close to zero indicates aggressive adaptation, but has the danger of over-refinement. η_t controls the targeted error level and prevents stalling of the adaptation convergence. The values used for these parameters in this work are $\eta_t = 0.7$ and $\eta_a = 0.2$

During each iteration of the adaptation cycle TetGen [40] preforms the re-meshing. Currently, TetGen only supports isotropic mesh refinement.

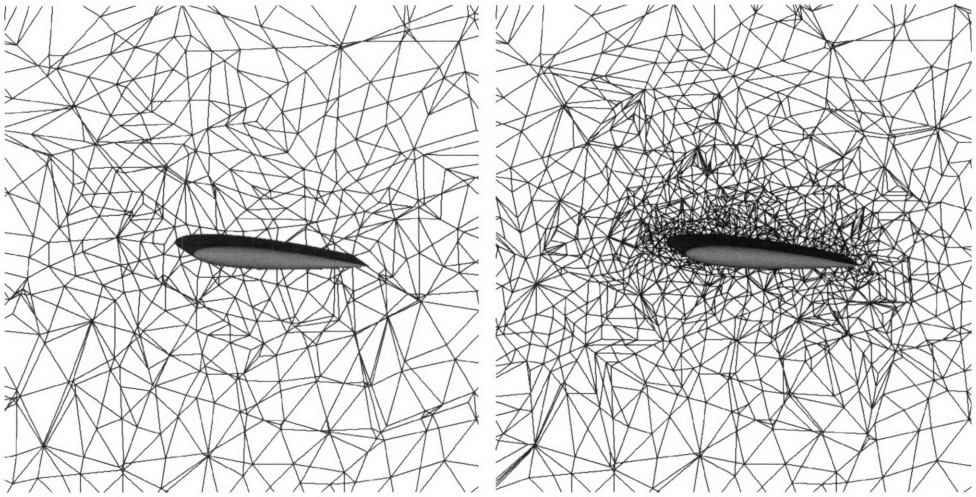
Chapter 6

Results

The output-based adaptive higher-order method was validated by comparing results for an isolated rotor in hover against the standard Caradonna and Tung experiments for an extruded NACA 0012 rotor[11]. The specific conditions are $M_{tip} = 0.439$ and $\theta_c = 8^o$. The goal of this is to demonstrate the ability to accurately compute the thrust of the rotor by matching the loading distribution along the chord and span. Of secondary interest is the ability to capture the rotor tip vortex through output-based adaptation targeting thrust. Comparisons of the adapted meshes and the estimated error convergence histories are given in terms of degrees of freedom (DOF) using interpolation orders $p = 0$ to $p = 2$. The degrees of freedom count does not include the equation set specific unknowns, typically 5 for the 3D Euler equations.

Figure 4-2 shows the quadratic patch representation of the wing geometry. 6,976 patches are used, spaced evenly along the span, and distributed along the chord to favor the leading and trailing edges. The initial background mesh in Figure 6-1(a) is created using TetGen. An initially arbitrary set of elements filling the half-cylinder background volume are adapted on. The adaptation indicator used to refine the mesh is the number of quadratic patches from the geometry representation intersecting or contained in each element. This geometric adaptation is performed to ensure sufficient refinement of the mesh around the geometry to provide a valid initial solution. As the output of interest is thrust, refinement is expected primarily along the leading and

trailing edges of the rotor. The adjoint solution in Figure 6-2 (a) shows the adaptation scheme's desire to refine the mesh not only on the surface of the geometry but also in the region upstream of the rotor surface. Also of note, in Figure 6-2 (b), is the presence of an adjoint wake that lies above (and thus upstream) of the rotor. The result of three adaptation cycles can be seen in Figure 6-1 (b) where many more elements have been placed on the rotor surface.

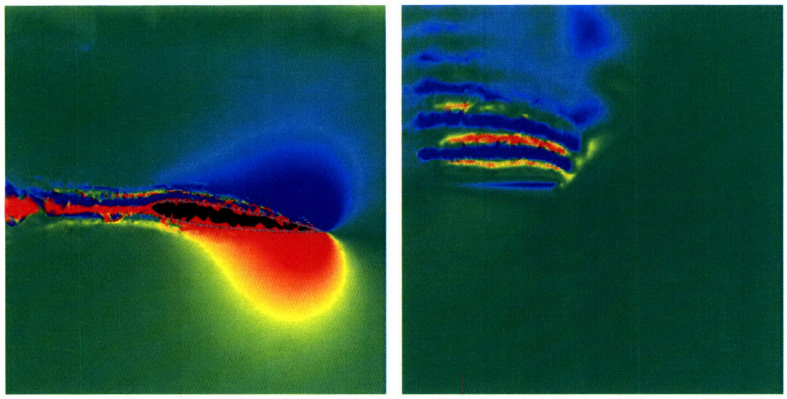


(a) Initial mesh with $N_{elem} = 11,664$

(b) Mesh with $N_{elem} = 34,831$

Figure 6-1: Initial mesh and a mesh resulting from three adaptation cycles viewed at $r/R = 0.80$.

Adaptation runs were performed for $p = 0, 1, \&2$. Figure 6-3 shows the convergence of the estimated output error versus degrees of freedom for all three of the solution orders. Convergence with $p = 0$ is very slow and would take many orders of magnitude more degrees of freedom to match the final error estimates of either $p = 1$ or $p = 2$. The magnitude of the final error estimate per degree of freedom points towards $p = 2$ being the better solution procedure than $p = 1$, but this does not take into account the robustness of the process which currently favors $p = 1$. Due to the impact of the small cut cells, even when merging is used, Newton-like convergence is rare for $p = 2$



(a) Slice at $r/R = 0.89$ of the adjoint solution for energy

(b) Adjoint solution for mass directly behind the trailing edge

Figure 6-2: Adjoint solution for the $p = 2$, $DOF = 2,524,710$ mesh.

solutions. Figure 6-3 shows that the $p = 1$ solution with $N_{elem} = 750,537$ has a similar error to the $p = 2$ solution with $N_{elem} = 58,636$, but as Figure 6-4 shows the $p = 1$ solution demonstrates better convergence characteristics.

Figure 6-5 shows the convergence of the computed coefficient of thrust versus the degrees of freedom. Figure 6-5 (a) shows that the $p = 1$ and $p = 2$ solutions have not yet converged to the same value for coefficient of thrust. However, when the output error estimate is included forming an estimated bound on the coefficient of thrust, seen in Figure 6-5 (b), it is still reasonable to assume that the $p = 1$ and $p = 2$ solutions are converging to the same value.

The improvement of surface pressure due to adaptation and polynomial order at $r/R = 0.80$ and $r/R = 0.96$ is shown in Figure 6-6 and 6-6, respectively. Although the finest $p = 1$ mesh results in more degrees of freedom than the finest $p = 2$ mesh, the $p = 2$ solution does a better job resolving the flow, particularly at the suction peak at the leading edge.

In order to compare the final output of the adaptation cycle to experimental data, Figure 6-8 shows the computed coefficient of pressure results from the finest $p = 2$

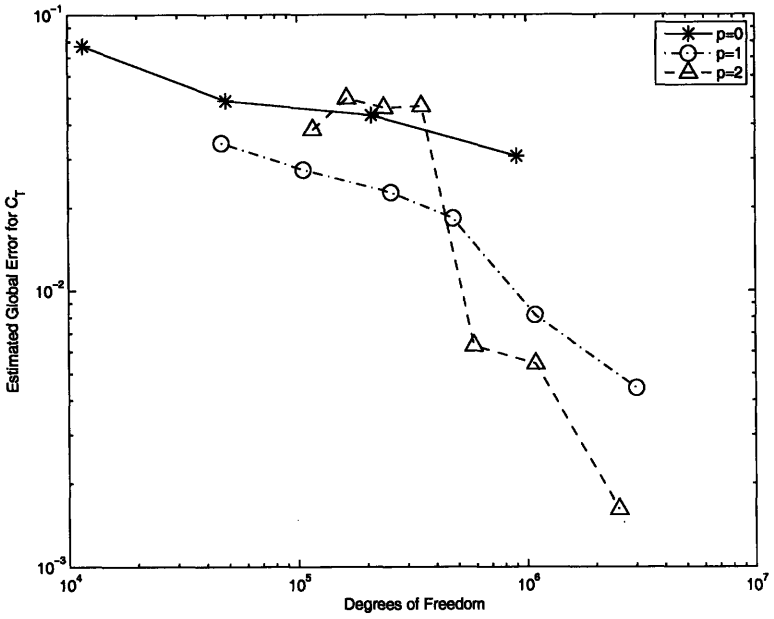


Figure 6-3: Plot of the estimated output error for the coefficient of thrust versus degrees of freedom.

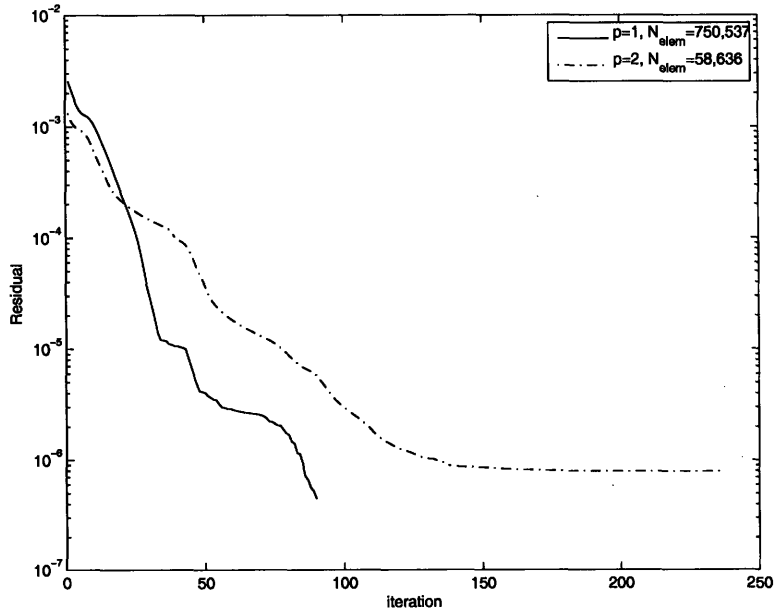
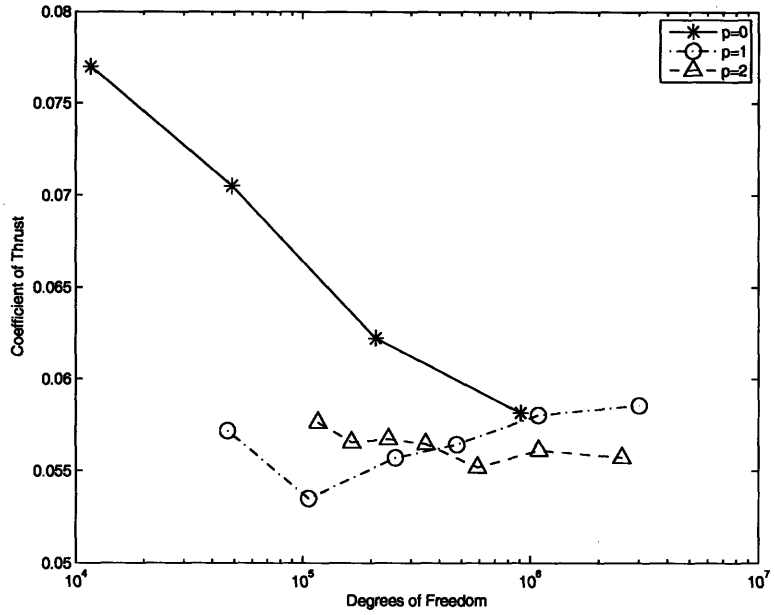
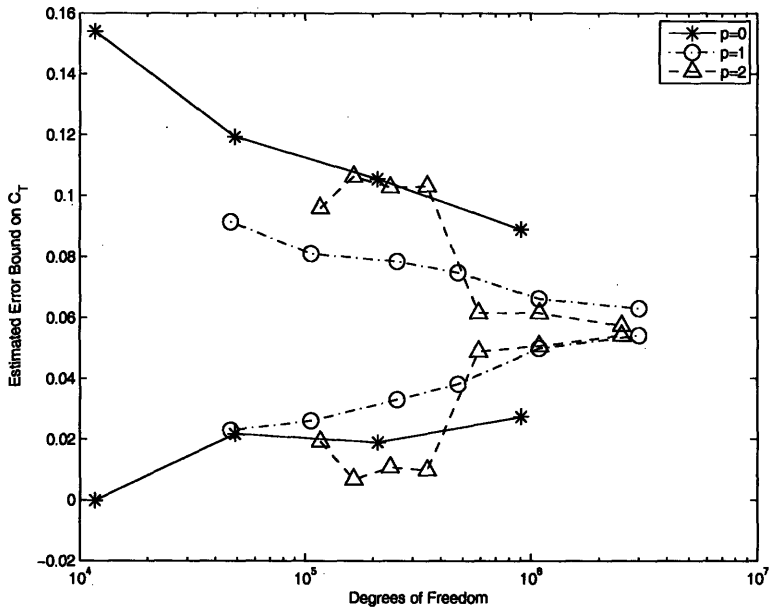


Figure 6-4: Convergence characteristics of a $p = 1$ and a $p = 2$ solution with similar error. Demonstrates the inefficiencies which still exist with $p = 2$ cut-cell solutions.

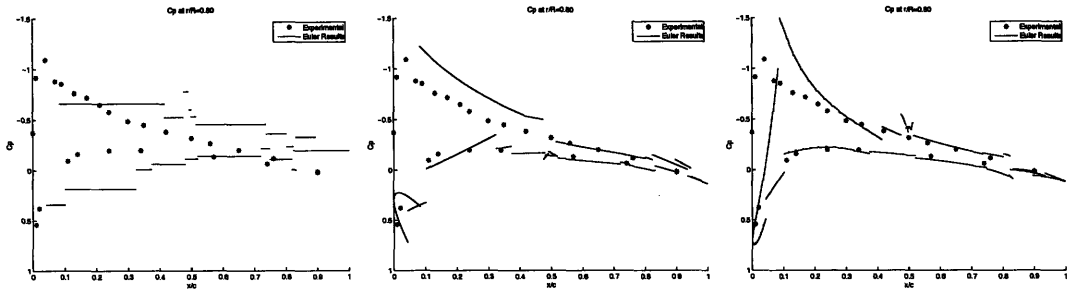


(a) Computed C_T



(b) Estimated bound on C_T

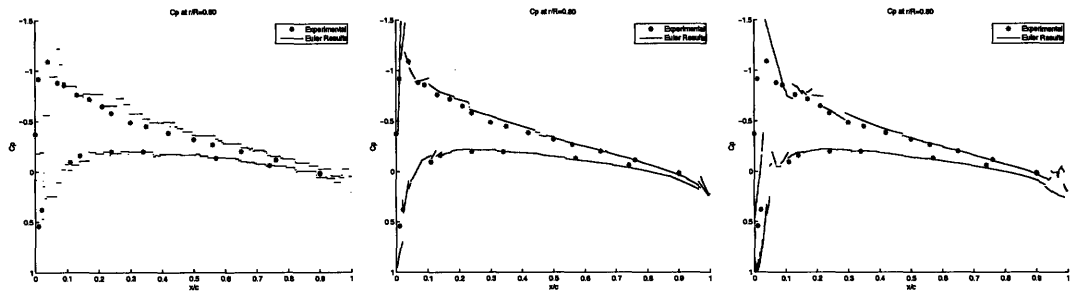
Figure 6-5: Computed coefficient of thrust and the estimated bound of coefficient of thrust versus degrees of freedom.



(a) $p = 0$,
$DOF = 11,664$,
$N_{elem} = 11,664$

(b) $p = 1$,
$DOF = 46,656$,
$N_{elem} = 11,664$

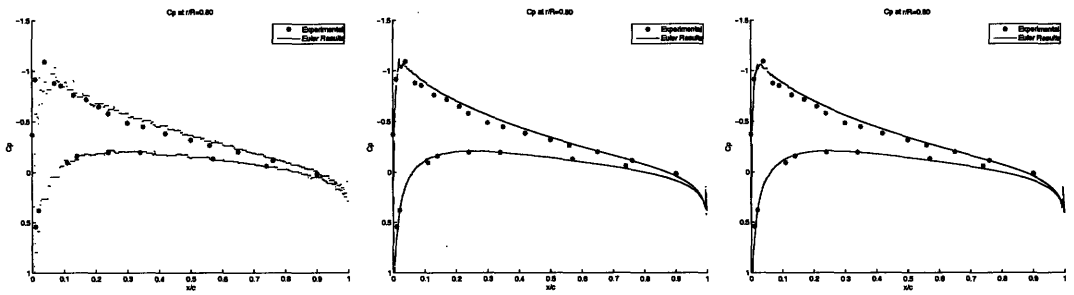
(c) $p = 2$,
$DOF = 116,640$,
$N_{elem} = 11,664$



(d) $p = 0$,
$DOF = 209,816$,
$N_{elem} = 209,816$

(e) $p = 1$,
$DOF = 255,464$,
$N_{elem} = 63,866$

(f) $p = 2$,
$DOF = 239,060$,
$N_{elem} = 23,906$

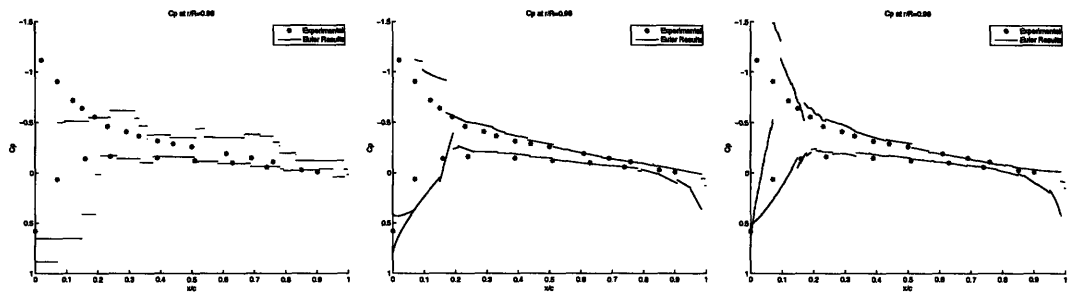


(g) $p = 0$,
$DOF = 906,436$,
$N_{elem} = 906,436$

(h) $p = 1$,
$DOF = 3,002,148$,
$N_{elem} = 750,537$

(i) $p = 2$,
$DOF = 2,524,710$,
$N_{elem} = 252,471$

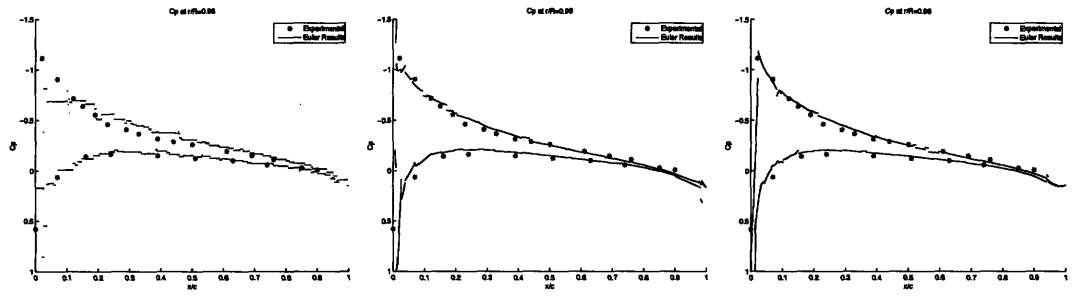
Figure 6-6: Selected coefficient of pressure plots at $r/R = 0.80$ for the $p = 0$, $p = 1$, and $p = 2$ solution.



(a) $p = 0$,
$DOF = 11,664$,
$N_{elem} = 11,664$

(b) $p = 1$,
$DOF = 46,656$,
$N_{elem} = 11,664$

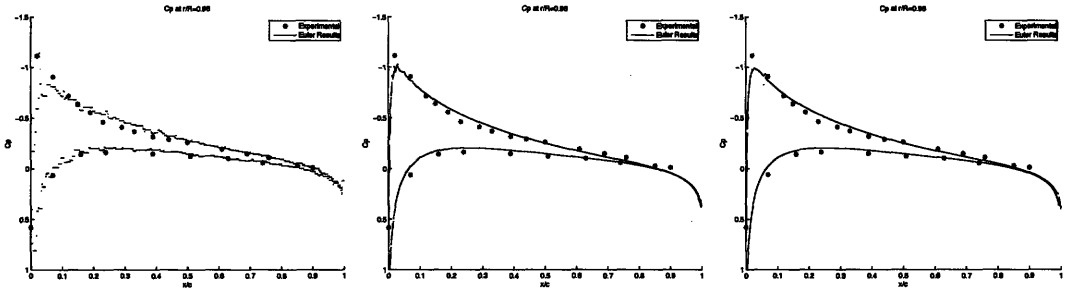
(c) $p = 2$,
$DOF = 116,640$,
$N_{elem} = 11,664$



(d) $p = 0$,
$DOF = 209,816$,
$N_{elem} = 209,816$

(e) $p = 1$,
$DOF = 255,464$,
$N_{elem} = 63,866$

(f) $p = 2$,
$DOF = 239,060$,
$N_{elem} = 23,906$



(g) $p = 0$,
$DOF = 906,436$,
$N_{elem} = 906,436$

(h) $p = 1$,
$DOF = 3,002,148$,
$N_{elem} = 750,537$

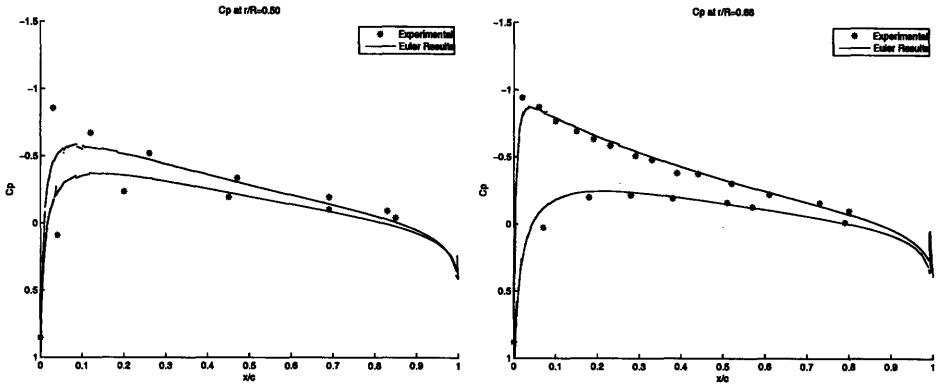
(i) $p = 2$,
$DOF = 2,524,710$,
$N_{elem} = 252,471$

Figure 6-7: Selected coefficient of pressure plots at $r/R = 0.96$ for the $p = 0$, $p = 1$, and $p = 2$ solution.

solution at all five cross-section locations where experimental data exists. The agreement between the experimental data and numerical results is generally good for the four outboard stations along the span and compares well with results for the same case presented by Agarwal and Deese [2] and Strawn and Barth[43]. There is a general lack of agreement in the C_p's on the leading edge at $r/R = 0.5$. This is very different than the results from Agarwal and Deese who for the same case actually over predicted the ΔC_p at $r/R = 0.5$. The cause of this underprediction is unknown and an area for future work.

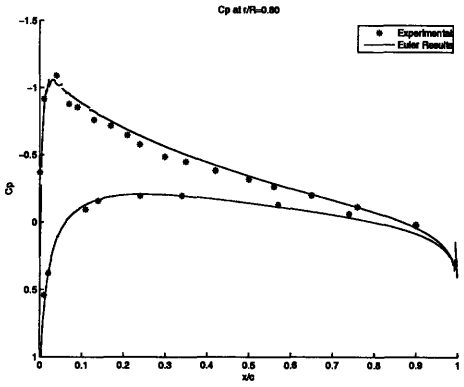
Also of interest is the ability for the higher-order DG discretization and output-based adaptation to capture the rotor tip vortex. Figure 6-9 and 6-10 show the evolution of the rotor tip vortex, represented using an iso-surface of entropy, as the adaptation cycle progresses. The higher-order, $p = 2$, solution does a much better job capturing the tip vortex. For around three million degrees of freedom the $p = 2$ captures three and a half revolutions of the tip vortex while the $p = 1$ solution captures about two revolutions of the vortex. The results of the output-based adaptation cycle are comparable to results presented in the literature. Canonne et al. [9] used an adaptation indicator based on the norm of the vorticity to preserve 380 degrees of the tip vortex using 600,000 degrees of freedom with a finite volume discretization. This is better than what the $p = 2$ solution can do for 586,360 degrees of freedom. Figure 6-10 indicates that only about 120 degrees can be preserved. On the other hand, Shaw et al. [39] captured three turns of the tip vortex again using a finite-volume discretization on a mesh with three million degrees of freedom and an adaptation scheme driven by an analytic description of the tip vortex path based on C_T, while the $p = 2$ solution with 2,524,710 degrees of freedom appears to capture three and a half revolutions.

It appears that the integrity of the tip vortex is preserved mainly via the higher-order method as opposed to an increase in element density in the tip vortex region. Figure 6-11 and 6-12 show the mesh and contours of entropy half a chord downstream of the trailing edge of the rotor. In neither figure is it possible to see an increase in element density around the contours that surround the tip vortex.

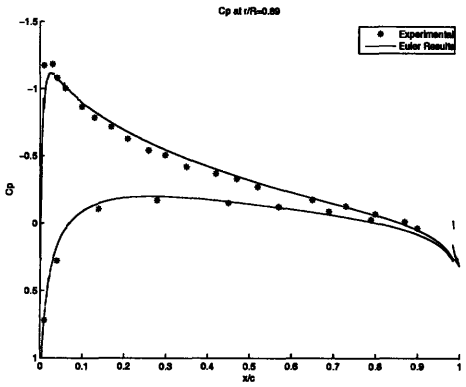


(a) $\frac{r}{R} = 0.50$

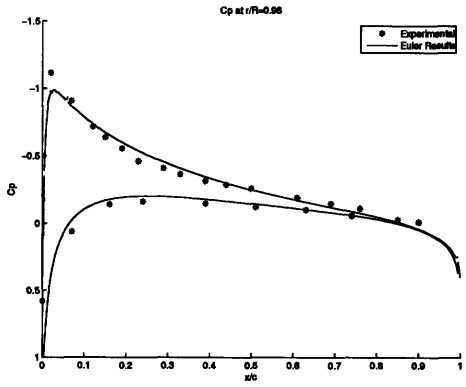
(b) $\frac{r}{R} = 0.68$



(c) $\frac{r}{R} = 0.80$

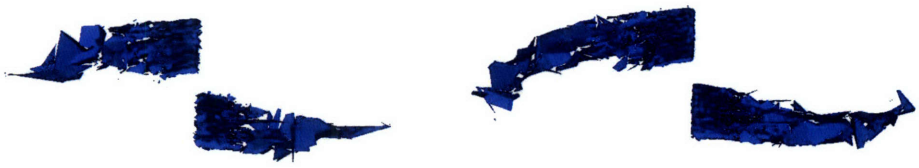


(d) $\frac{r}{R} = 0.89$



(e) $\frac{r}{R} = 0.96$

Figure 6-8: Numerical results and experimental data for the coefficient of pressure on the rotor surface for $M_{tip} = 0.439$ and $\theta_c = 8^o$. Results are taken from finest $p = 2$ solution with $2,524,710$ degrees of freedom from the adaptation cycle.



(a) $DOF = 46,656$, $N_{elem} = 11,664$

(b) $DOF = 106,120$, $N_{elem} = 26,530$

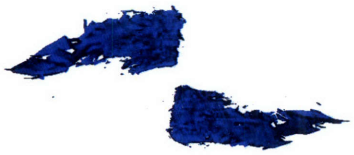
(c) $DOF = 255,464$, $N_{elem} = 63,866$

(d) $DOF = 476,312$, $N_{elem} = 119,078$

(e) $DOF = 1,085,396$, $N_{elem} = 271,349$

(f) $DOF = 3,002,148$, $N_{elem} = 750,537$

Figure 6-9: Evolution of the rotor tip vortex, viewed as iso-surfaces of entropy, for the $p = 1$ adaptation cycle.



(a) $DOF = 116,640$,
$N_{elem} = 11,664$

(b) $DOF = 239,060$,
$N_{elem} = 23,906$

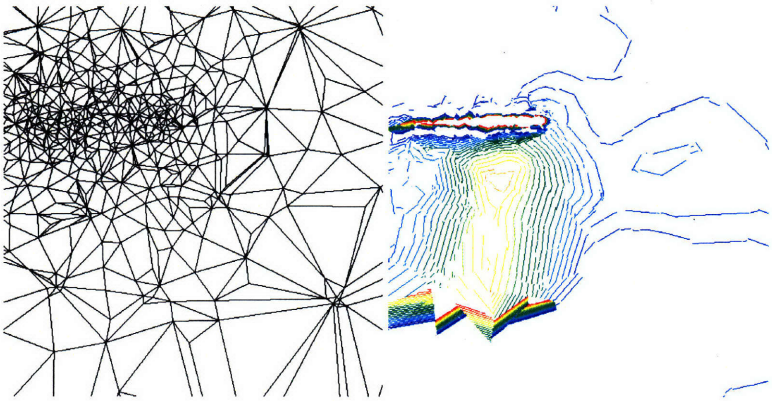
(c) $DOF = 348,310$,
$N_{elem} = 34,831$

(d) $DOF = 586,360$,
$N_{elem} = 58,636$

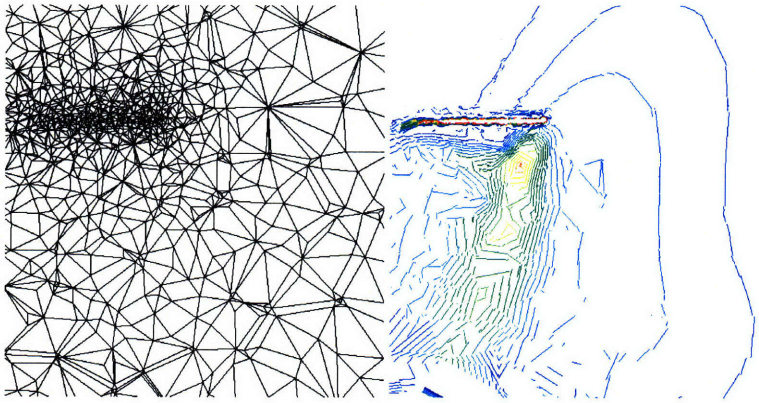
(e) $DOF = 1,090,820$,
$N_{elem} = 109,082$

(f) $DOF = 2,524,710$,
$N_{elem} = 252,471$

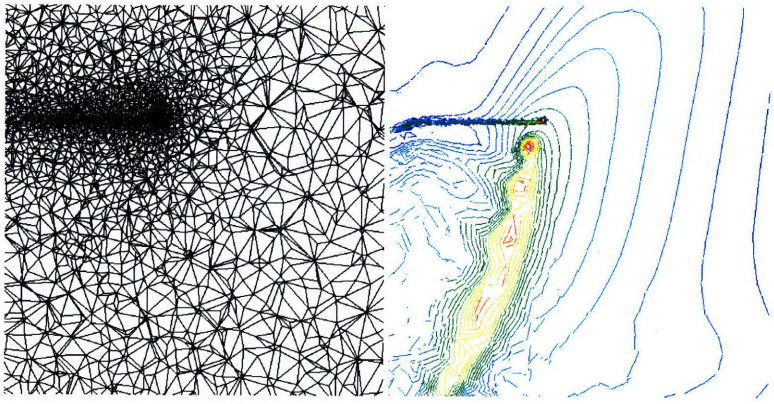
Figure 6-10: Evolution of the rotor tip vortex, viewed as iso-surfaces of entropy, for the $p = 2$ adaptation cycle.



(a) $DOF = 46,656$, $N_{elem} = 11,664$



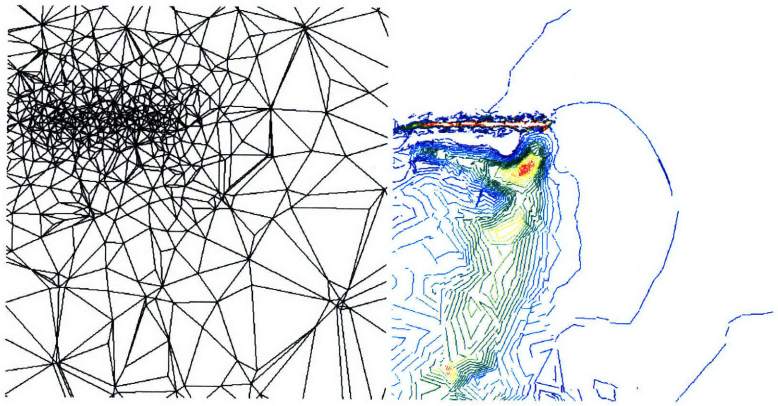
(b) $DOF = 255,464$, $N_{elem} = 63,866$



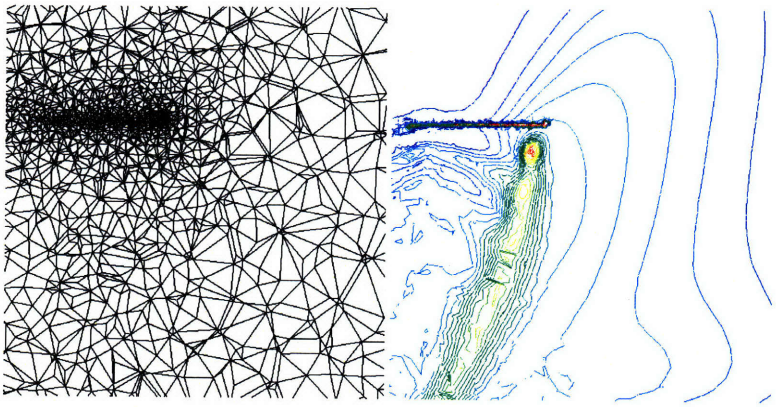
(c) $DOF = 3,002,148$, $N_{elem} = 750,537$

Figure 6-11: Selected $p = 1$ meshes and entropy contours on a slice half a chord downstream from the trailing edge of the rotor.

(a) $DOF = 116,640$, $N_{elem} = 11,664$



(b) $DOF = 239,060$, $N_{elem} = 23,906$



(c) $DOF = 2,524,710$, $N_{elem} = 252,471$

Figure 6-12: Selected $p = 2$ meshes and entropy contours on a slice half a chord downstream from the trailing edge of the rotor.

Chapter 7

Conclusions

This work demonstrates the use of an output-based mesh adaptation method for high-order discontinuous Galerkin discretizations. In order to make the adaptation procedure more autonomous while supporting the higher-order needs of the DG discretization, a simplex cut-cell method is used to provide meshes which contain higher-order geometry information. The inclusion of the cut-cell technique is an enabler for the adaptation process, allowing the generation of higher-order meshes without user involvement based on the requested mesh size parameters from the output-based error estimation.

The results demonstrate that the DG discretization is a good foundation to model rotary-wing aerodynamics. Even though the Euler equations are used in this work, good agreement between the computed and experimental results is seen. Using the output-based adaptation method convergence of a bound on the coefficient of thrust is established. Compared solely on a degree of freedom count the $p = 2$ adaptation cycle demonstrates better convergence than the $p = 1$ solution, however convergence issues remain for $p = 2$ solutions with small cut cells.

Also explored in this work was the capability of a higher-order method combined with output-based adaptation to preserve the tip vortex over large distances. Although the output of interest used in this work, thrust, gives no guarantees on the importance of the rotor tip vortex, the adjoint solution appears to highlight the vortex wake and

with adaptation the tip vortex is increasingly preserved. With the adaptation cycle, a final $p = 2$ solution is produced that contains less than three million degrees of freedom and is capable of preserving the rotor tip vortex for three and a half revolutions.

Much work remains to be done to make the cut-cell mesh generation technique presented in this work more functional. The impact of small elements on convergence and solution quality needs to be studied. As the robustness of the cut-cell technique improves, a logical next step would be to expand the output-based adaptation process to include anisotropic adaptation and solve the Navier-Stokes equations for an isolated rotor in hover.

Bibliography

[1] Adelman, H. M. and Mantay, W. R. "Integrated Multidisciplinary Design Optimization of Rotorcraft." Journal of Aircraft 1, 1991.

[2] Agarwal, R. R. and Dees, J. E. "Euler Calculations for Flowfiled of a Helicopter Rotor in Hover." Journal of Aircraft 24-4, 1987.

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

[4] Bassi, F. and Rebay, S. "High-order accurate discontinuous finite element solution of the 2D Euler equations." *Journal of Computational Physics*, 138(2):251–285, 1997.

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

[6] Boelens, O., van der Ven, H., Oskam, B., and Hassan, A. "Accurate and Efficient Vortex-Capturing for a Helicopter Rotor in Hover." National Aerospace Laboratory NLR-TP-2000-420, August 2000.

[7] Bottasso, C. L. and Shephard, M. S. "Parallel Adaptive Finite Element Euler Flow Solver for Rotary Wing Aerodynamics." AIAA Paper 1995-1661, June 1995.

[8] Bottasso, C. L. and Shephard, M. S. "Parallel Adaptive Finite Element Euler Flow Solver for Rotary Wing Aerodynamics." *AIAA Journal*, (1997-0112), June 1997.

[9] Canonne, E., Benoit, C., and Jeanfaivre, G. "Cylindrical Mesh Adaptation for Isolated rotors in Hover." Aerospace Science and Technology 1, January 2004.

[10] Caradonna, F. X. "Developments and Challenges in Rotorcraft Aerodynamics." AIAA Paper 2000-0109, January 2000.

[11] Caradonna, F. and Tung, C. "Experimental and Analytical Studies of a Model Helicopter Rotor in Hover." NASA Technical Memorandum 81232, September 1981.

[12] Diosady, L. T. "A Linear Multigrid Preconditioner for the Solution of the Navier-Stokes Equations using a Discontinuous Galerkin Discretization." Masters thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, May 2007.

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

[14] Fidkowski, K. J. *A Simplex Cut-Cell Adaptive Method for High-Order Discretizations of the Compressible Navier-Stokes Equations.* Phd thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2007.

[15] Fidkowski, K. J. and Darmofal, D. L. "Development of a higher-order solver for aerodynamic applications." AIAA Paper 2004-0436, January 2004.

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

[17] Fidkowski, K. J. and Darmofal, D. L. "A Triangular Cut-Cell Adaptive Method for Higher-Order Discretizations of the Compressible Navier-Stokes Equations." *Journal of Computational Physics*, 225:1653–1672, 2007.

[18] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L. "p-Multigrid Solution of High-Order Discontiunous Galerkin Discretizations of the Compressible Navier-Stokes Equations." *Journal of Computational Physics*, 207(1):92–113, 2005.

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

[20] Hariharan, N., Ekaterinaris, J., Liu, Y., Gupta, R., and L.N.Sankar. "An Evaluation of Higher Order Spactial Accuracy Algorithms for Modeling Fixed and Rotary Wing Tip Regions." In *Proceedings of the American Helicopter Society Specialists Meeting on Aerodynamics and Aeroacoustics*, San Francisco, January 2002.

[21] Hariharan, N., Sankar, L. N., and Tadghighi, H. "Evaluation of High Order Upwind Schemes for Rotors in Hover." AIAA Paper 2003-49, January 2003.

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

[23] Jones, W. T., Nielsen, E. J., and Park, M. A. "Validation of 3D Adjoint Based Error Estimation and Mesh Adaptation for Sonic Boom Prediction." AIAA Paper 2006-1150, 2006.

[24] Kelley, C. T. and Keyes, D. E. "Convergence Analysis of Pseudo-Transient Continuation." *SIAM Journal of Numerical Analysis*, 35(2):508–523, 1998.

[25] Lee-Rausch, E. M., Park, M. A., Jones, W. T., Hammond, D. P., and Nielsen, E. J. "Application of Parallel Adjoint-Based Error Estimation and Anisotropic Grid Adaptation for Three-Dimensional Aerospace Configurations." AIAA Paper AIAA-2005-4842, 2005.

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

[27] Machiels, L., Peraire, J., and Patera, A. "A Posteriori Finite-Element Output Bounds for the Incompressible Navier-Stokes Equations: Application to a Natural Convection Problem." *Journal of Computational Physics*, 172:401–425, 2001.

[28] Mavriplis, D. J. "Multigrid Strategies for Viscous Flow solvers on anisotropic unstructured meshes." *Journal of Computational Physics*, 145:141–165, 1998.

[29] Mavriplis, D. J. "Large-scale parallel viscous flow computations using an unstructured multigrid algorithm." ICASE XSReport TR-99-44, 1999.

[30] Nemec, M., Aftosmis, M. J., and Wintzer, M. "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries." AIAA Paper 2008-725, 2008.

[31] O'Brien, D. M. *Analysis of Computational Modeling Techniques for Complete Rotorcraft Configurations.* PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, 2006.

[32] Park, M. A. "Adjoint-based, Three-dimensional Error Predication and Grid Adaptation." *AIAA Journal*, 42(9):1854–1862, 2004.

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

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

[35] Ramachandran, K., Tung, C., and Caradonna, F. X. "Rotor Hover Performance Prediction Using a Free-Wake, Computational Fluid Dynamics Method." Report 12, 1989.

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

[37] Roe, P. L. "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes." *Journal of Computational Physics*, 43(2):357–372, 1981.

[38] Saad, Y. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 1996.

[39] Shaw, S. T., Hill, J. L., and Villamarin, C. E. "A Priori Grid Adaptation For Helicopter Rotor Wakes Using Unstructured and Structured-Unstructured Hybrid Grids." AIAA Paper 2005-0463, January 2005.

[40] 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`.

[41] Srinivasan, G. R., Raghavan, V., Duque, E. N., and McCroskey, W. J. "Flowfield Analysis of Modern Helicopter Rotors in Hover by Navier-Stokes Method." American Helicopter Society Journal 38-39, October 1993.

[42] Strawn, R. and Ahmad, J. "Computational Modeling of Hovering Rotors and Wakes." AIAA Paper 2000-0110, January 2000.

[43] Strawn, R. C. and Barth, T. J. "A Finite-Volume Euler Solver for Computing Rotary-Wing Aerodynamics on Unstructured Meshes." *American Helicopter Society*, 48(1):419–428, 1992.

[44] Trefethen, L. N. and Bau, D. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

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

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

[47] Wenren, Y., Fan, M., Dietz, W., Hu, G., Braun, C., Steinhoff, J., and Grossman, B. "Efficient Eulerian Computation of Realistic Rotorcraft Flows Using Vorticity Confinement." AIAA Paper 2001-0996, January 2001.

[48] Wenren, Y. and Steinhoff, J. "Application of Vorticity Confinement to the Prediction of the Wake of Helicopter Rotors and Complex Bodies." AIAA Paper 1999-3200, June 1999.

[49] Yeshala, N., Egolf, T. A., Vasilescu, R., and Sankar, L. N. "Application of Higher Order Spactially Accurate Schemes to Rotors in Hover." AIAA Paper 2006-2818, June 2006.