Three–Dimensional Turbulent RANS Adjoint–Based Error Correction

Michael A. Park *
NASA Langley Research Center, Hampton, VA 23681

Abstract

Engineering problems commonly require functional outputs of computational fluid dynamics (CFD) simulations with specified accuracy. These simulations are performed with limited computational resources. Computable error estimates offer the possibility of quantifying accuracy on a given mesh and predicting a fine grid functional on a coarser mesh. Such an estimate can be computed by solving the flow equations and the associated adjoint problem for the functional of interest. An adjoint-based error correction procedure is demonstrated for transonic inviscid and subsonic laminar and turbulent flow. A mesh adaptation procedure is formulated to target uncertainty in the corrected functional and terminate when error remaining in the calculation is less than a user-specified error tolerance. This adaptation scheme is shown to yield anisotropic meshes with corrected functionals that are more accurate for a given number of grid points then isotropic adapted and uniformly refined grids.

Introduction

Engineering problems commonly require computational fluid dynamics (CFD) solutions with functional outputs of specified accuracy. The computational resources available for these solutions are limited in practice, and errors in solutions and outputs are unknown. CFD solutions may be computed with an unnecessarily large number of grid points (and associated high cost) to ensure that the outputs are within a required accuracy. A method which estimates the error present in a computed functional offers the possibility to avoid the use of overly refined grids in order to guarantee accuracy.

Unstructured grid technology promises easier initial grid generation for novel complex three-dimensional (3D) configurations compared with structured grid techniques. The use of unstructured grid technology for CFD simulations allows more freedom in adapting the discretization of the meshes to improve the fidelity of the simulation. Many previous efforts attempted to tailor the discretizations of unstructured meshes to increase solution accuracy while reducing computational cost.[1–8]

*Research Scientist, Computational Modeling and Simulation Branch, m.a.park@larc.nasa.gov

Most of these adaptive methods focus on modifying discretizations to reduce local equation errors. These local errors are not guaranteed to directly impact error in global output functions. These methods, often referred to as feature-based adaptation, focus on resolving discontinuities or strong gradients in the flow field. Unfortunately, flow features (e.g., shocks) can be in the incorrect location due to errors elsewhere in the flow field. Also, resolving the flow in a location with large local error may have a minimal effect on the output function (e.g., a downstream shock). However, locations with small local errors (e.g., along a stagnation streamline) may have a large effect on forces.

If the flow equations are linearized about the existing primal solution, a linear dual problem can yield a direct measure of the impact of local primal (flow equation) residual on a selected functional output. The combination of the primal and dual problems can also yield a correction to a specified functional on a given mesh. There are many examples of these techniques in the finite element communities.[9,10] Pierce and Giles[11] applied these methods to finite-volume discretizations. Venditti and Darmofal[12–15] demonstrated these methods for compressible two-dimensional (2D) inviscid and viscous flow solutions. Müller and Giles[16] also presented results for a similar approach. Park[17] applied the methods of Venditti and Darmofal[13] to 3D inviscid problems.

Adaptation can reduce error in a output function by refining a discretization in locations where the local equation error weighted with the dual solution is large.[16] This methodology targets refinement where it will have the most impact on reducing the error in an output functional. An alternative method is to refine the mesh to reduce uncertainty in the dual correction term because this correction can be computed to high accuracy.[14,17]

Adaptation to reduce the error prediction uncertainty is a robust and effective method for tuning a discretization to efficiently calculate a specific functional.[13–15] Park[17] demonstrated 3D isotropic adaptation coupled to a computer-aided design (CAD) description of the model for transonic, subsonic, and incompressible inviscid flow. The current study focuses on defining the requirements for extending Park's methodology to anisotropic adaptation for turbulent flow.

The adjoint variable approach (solution of the dual problem) is an efficient method for computing derivatives of a functional of interest for gradient-based design methods. Some examples of discrete adjoint design methods are given in Nielsen and Anderson.[18,19] The discrete dual problem for adjoint variables can be expensive (on the order of the primal problem). However, the adjoint solution is already available for use during the aerodynamic design process, so it could be employed for simultaneous design, error prediction, and grid adaptation.

The combination of adjoint-based grid adaptation and design techniques can yield an attractive tool for the aerodynamic design of new configurations. Adjoint-based error prediction and adaptation can yield meshes with fewer points than traditional feature-based schemes with computable error estimates on output functions. Design processes require analysis and derivative evaluation tools that operate with minimal human interaction. Robust, automatic adaptation techniques enable the increased use of nonlinear flow calculations in larger multidisciplinary design frameworks. These new techniques will enable efficient analysis for existing configurations and expanded exploration of design spaces for new configurations.

This work is part of the Fast Adaptive Aerospace Tools[20] (FAAST) program. The goal of the FAAST program is to improve the robustness of high-fidelity CFD analysis and reduce total time for analysis and design. This study describes ongoing work toward error correction, grid adaptation for function outputs, and CAD-to-grid methods.

Flow Equations

The Fully Unstructured Navier-Stokes Three-Dimensional[21–23†] (FUN3D) suite of codes is employed in this study. The compressible flow solver employs an unstructured finite-volume tetrahedral method for conserved variables, $\mathbf{Q}$, i.e.,

$$\mathbf{Q} = [\rho \ \rho u \ \rho v \ \rho w \ E]^T \tag{1}$$

where ρ is density, u, v, and w are velocity, and E is total energy per unit volume. The node-based variables $\mathbf{Q}$ are computed by driving the flow equation residual $\mathbf{R}$ to steady-state with an implicit point-iterative method. FUN3D is able to solve incompressible, Euler, and Reynolds-averaged Navier-Stokes (RANS) flow equations, either tightly or loosely coupled to the Spalart-Allmaras[24] (S-A) one-equation turbulence model. When the S-A model is included, the turbulence model quantity $\tilde{\nu}$ is included in $\mathbf{Q}$. The turbulent viscosity μ_t is also computed and stored to reduce execution time. The present study employs the Euler and RANS equations coupled to the S-A turbulence model.

†`http://fun3d.larc.nasa.gov`

The solution of $\mathbf{Q}$ allows the calculation of integral quantities f (e.g., lift, drag). The skin friction component of the forces can be computed with element- or flux-based methods (these are the nonconservative and conservative methods of Venditti,[14,15] respectively).

To speed execution, the global problem domain is decomposed into multiple subdomains and the flow and the adjoint problems are solved with a parallel execution scheme, which communicates via the message passing interface (MPI) standard. The FUN3D suite of codes is being extended to the High Energy Flow Solver Synthesis[20] (HEFSS) modular framework of FORTRAN 90 libraries.

Adjoint Equations

After the flow solution is known, the discrete adjoint equations[18,22,23] are solved to complete the dual problem. The first step is to linearize the flow equation residual (including the turbulence model) $\mathbf{R}$ and output function f with respect to the flow solution $\mathbf{Q}$. After this linearization, an adjoint variable λ is solved for each of the flow equations and the turbulence model.

An abbreviated derivation, adapted from Taylor et al.,[25] is given below. The chain rule for the linearized output function is

$$\frac{\partial f}{\partial \mathbf{Q}} = \left(\frac{\partial f}{\partial \mathbf{R}}\right)^T \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \tag{2}$$

The adjoint variable λ is defined as the effect of the flow residual on the output function:

$$\frac{\partial f}{\partial \mathbf{R}} = \lambda \tag{3}$$

A set of linear equations is solved to find λ:

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^T \lambda = \left(\frac{\partial f}{\partial \mathbf{Q}}\right)^T \tag{4}$$

After the flow solution is known, this set of linear equations is solved with a point-iterative time-marching method.[26–29] An earlier implementation of the adjoint solver used GMRES.[30] See Refs. 18, 19, and 23 for details.

Error Correction

The error prediction and correction scheme is taken from Refs. 13–15. With a solution on a mesh of reasonable size $\mathbf{Q}^0$, it is desirable to predict the value of an output function evaluated with a solution on a much finer mesh $f(\mathbf{Q}^*)$. This prediction can be computed without the solution on this finer mesh when the adjoint solution on this reasonable mesh λ^0 is used. The full derivation of the error correction term is available in Refs. 11,13,14, and 16. An abbreviated derivation is

presented by expanding the Taylor series about $f(\mathbf{Q}^0)$, i.e.,

$$f(\mathbf{Q}^*) = f(\mathbf{Q}^0) + \left(\left.\frac{\partial f}{\partial \mathbf{R}}\right|_0\right)^T \left(\mathbf{R}(\mathbf{Q}^*) - \mathbf{R}(\mathbf{Q}^0)\right) + \ldots \tag{5}$$

Employing Eq. (3) and assuming that the residual on the much finer mesh is zero yields an improved estimate for the functional of interest f_{est}:

$$\left.\frac{\partial f}{\partial \mathbf{R}}\right|_0 = \lambda^0 \tag{6}$$

$$\mathbf{R}(\mathbf{Q}^*) = 0 \tag{7}$$

$$f(\mathbf{Q}^*) \approx f_{est} = f(\mathbf{Q}^0) - (\lambda^0)^T \mathbf{R}(\mathbf{Q}^0) \tag{8}$$

To improve the prediction of the functional f_{est}, $\mathbf{Q}^0$ and λ^0 can be interpolated to an embedded mesh. Interpolation is performed in two ways for this study: a linear interpolation ($\mathbf{Q}^L$,λ^L) and a higher order interpolation ($\mathbf{Q}^H$,λ^H). Details of this interpolation and the construction of this embedded mesh are in the section "Error Correction and Adaptation Process." Substituting the higher order interpolant into Eq. (8) yields the higher order functional estimate f^H_{est}:

$$f^H_{est} = f(\mathbf{Q}^H) - (\lambda^H)^T \mathbf{R}(\mathbf{Q}^H) \tag{9}$$

Adaptation Parameter

The adaptation parameter, also from Ref. 13, is intended to specify a grid spacing modification to reduce the uncertainty in the computed error prediction. The grid is not modified to directly reduce the computed error prediction (as in Ref. 16) because an accurate estimate for the functional including this error term can be computed with Eq. (8). Instead, targeting the uncertainty in this computed quantity is more effective and improves the robustness of the adaptive process. The error correction (Eq. (8)) including the uncertainty in the dual solution is

$$f(\mathbf{Q}^0) - f(\mathbf{Q}^*) \approx (\lambda^0)^T \mathbf{R}(\mathbf{Q}^0) + (\lambda^* - \lambda^0)^T \mathbf{R}(\mathbf{Q}^0) \tag{10}$$

The uncertainty in the computed error correction is

$$f_{est} - f(\mathbf{Q}^*) \approx (\lambda^* - \lambda^0)^T \mathbf{R}(\mathbf{Q}^0) \tag{11}$$

The relation of the primal and dual problems[11,13] yields another expression for the error correction uncertainty

$$(\lambda^* - \lambda^0)^T \mathbf{R}(\mathbf{Q}^0) = \mathbf{R}_\lambda(\lambda^0)(\mathbf{Q}^* - \mathbf{Q}^0)^T \tag{12}$$

where $\mathbf{R}_\lambda(\lambda)$ is the residual of the dual problem:

$$\mathbf{R}_\lambda(\lambda) = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^T - \left(\frac{\partial f}{\partial \mathbf{Q}}\right)^T \lambda \tag{13}$$

A computable term is found by using the interpolation error of λ to replace $(\lambda^* - \lambda^0)$ and the interpolation error of $\mathbf{Q}$ to replace $(\mathbf{Q}^* - \mathbf{Q}^0)$. The higher order interpolate for $\mathbf{Q}^0$ and λ^0 is employed for the residual calculations to improve prediction in place of the linear interpolate in Ref. 13. The interpolation error is expressed as the difference in the high-order and linear interpolated values:

$$(\lambda^* - \lambda^0) \approx (\lambda^H - \lambda^L) \tag{14}$$

$$(\mathbf{Q}^* - \mathbf{Q}^0) \approx (\mathbf{Q}^H - \mathbf{Q}^L) \tag{15}$$

The average of the absolute values of the two uncertainty terms in Eq. (12) yields the adaptation intensity $\mathbf{I}$, which is computed for each equation on each embedded node:

$$\mathbf{I} = \frac{\left|(\lambda^H - \lambda^L)^T \mathbf{R}(\mathbf{Q}^H)\right| + \left|\mathbf{R}_\lambda(\lambda^H)(\mathbf{Q}^H - \mathbf{Q}^L)^T\right|}{2} \tag{16}$$

The intensity $\mathbf{I}$ is therefore the average of the absolute values of two terms. The first term is a dual solution interpolation error weighted with the primal residual. The second term is the dual problem residual weighted with a primal solution interpolation error. This form of the adaptation intensity (which includes weighed interpolation errors) focuses on the nonlinear contributions to the function error, which increases robustness of the adaptation method.

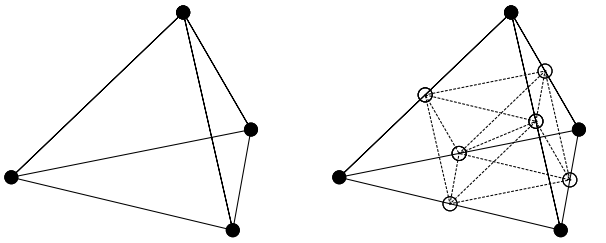
Error Correction and Adaptation Process

The error correction process begins with an initial tetrahedral mesh, which can come from any mesh generation system. The state variables are computed as the nonlinear solution to the flow equations on the initial mesh. The adjoint variables are then computed with the linearized flow equations at the flow solution. These flow and adjoint solution procedures employ a parallel execution scheme. Then the global problem domain is reconstructed to facilitate the creation of a finer, embedded grid with interpolated primal and dual solutions.

Embedded Mesh

To compute the error prediction and the adaptation parameter, a globally embedded or h-refined mesh is created. To construct the embedded mesh, a new node (represented by an open circle in Fig.1) is inserted at the midpoint of each existing edge (solid line) that connects existing nodes (closed circles); see Fig. 1(a) and 1(b). Each existing tetrahedron is subdivided to reconnect these new nodes with eight interior tetrahedra. (Each of the existing boundary faces is also divided into four triangles.)

The four new tetrahedra constructed at the corners of the original tetrahedron have the same shape as the original but are smaller in size. The construction of the four corner tetrahedra leaves an interior volume with eight faces, which is subdivided into four tetrahedra.



a) Original tetrahedron. **b) Embedded tetrahedra.**

Fig. 1 Tetrahedron embedding process.

The four interior tetrahedra have three unique configurations. The configuration with the lowest maximum dihedral angle is selected.

The new nodes are placed at the midpoints of edges during the mesh embedding process. The embedded nodes on the boundaries of the mesh may no longer remain on the original surface definition of the model. In Refs. 13 and 14, the newly created embedded boundary nodes are repositioned to coincide with the domain boundary and adjacent nodes are smoothed to maintain quality and integrity of the mesh. In the current approach, newly inserted boundary nodes are only projected to the domain boundary for the cylinder case (shown later in the "Results" section). All boundary nodes are projected during grid adaptation.

Embedding High Aspect-Ratio Cells

Performing global embedding on a 2D triangular mesh will yield smaller cells with shape measures identical to their parent cells. In 3D, global embedding of high aspect-ratio tetrahedra results in nearly collapsed sliver cells (excluding the four corner tetrahedra). However, if the high aspect ratio tetrahedra are arranged in the semistructured form of prismatic elements divided into three tetrahedra, the original shape measures can be maintained.

To regain the original shape measures, 4-for-4 edge swapping[31] is only performed on edges connecting newly created nodes. The subset of edges selected for swapping are shown as dashed lines in Fig. 1(b). The edge swapping technique chooses alternative cell connectivities that have a lower maximum dihedral angle. After this limited edge swapping, the mesh is equivalent to reconstructing the prismatic elements, refining the prisms, and dividing the embedded prismatic elements into tetrahedra. Advancing layer meshers[6,32–35] often employ the semistructured form of prismatic elements divided in to three tetrahedra for boundary layer elements.

Interpolation Preprocessing for Adjoint on Strong Boundaries

The primal equations change character on boundaries with strong enforcement of Dirichlet data. Instead of solving for the conservation of momentum and energy, the velocity and temperature are explicitly set for these boundary nodes. Therefore, the adjoint solution also changes character.[14,15,27] Enforcement of the continuity equation remains the same on these strong boundaries as in the interior. The interpolation of the adjoint solution to the embedded mesh should not be performed for the first layer of interior embedded nodes near strong boundaries because this interpolation will be between nodes with different dual properties. If the original mesh boundary adjoint solution is replaced with values extrapolated from the interior of the domain, the solution will be consistent between strong boundaries and the interior. Interpolation can then be performed between the consistent interior and extrapolated boundary adjoint solution.

To perform this extrapolation, least-squares gradients[21] are computed at the existing nodes by using the existing mesh and adjoint solution. The edges incident to strong boundary nodes and the strong boundary nodes themselves are excluded from this gradient calculation. The existing boundary adjoint solution is discarded and replaced with values extrapolated from the interior. This extrapolation is performed along each edge connecting the interior nodes to the boundary nodes. The extrapolated value at the boundary is computed with the interior value of the adjoint and its gradient.

If boundary nodes receive more than one edge contribution from the interior, those contributions are averaged. The original mesh adjoint solution is then consistent between the extrapolated strong boundaries and interior, so the same interpolation technique for problems with weak boundary conditions is applied. After the adjoint is interpolated to the embedded mesh, it is postprocessed to estimate the boundary adjoint for the error prediction calculation. See Ref. 14 for details.

Interpolation Techniques

Each of the solution and adjoint variables is interpolated in two ways to form the linear and the higher order reconstruction for the new nodes on the embedded mesh. The higher order reconstruction of the solution for the new nodes requires the computation of least-squares gradients[21] at the existing nodes by using the existing mesh and solution. Previous work[17] used a compact edge-based scheme to compute the linear and higher order reconstruction. The current work employs the same scheme for linear reconstruction; however, an improved element-based scheme, similar to Ref. 14, is used for the higher order reconstruction.

The higher order reconstruction begins with a quadratic fit of the solution over each coarse grid element. The quadratic function has ten terms: a constant, three first derivatives, three second derivatives, and three cross derivatives. The quadratic function is fit to the coarse grid value and its derivatives at the nodes of the coarse grid element. This fit

yields a 10x16 system of equations that is solved in a least-squares sense with LU decomposition to find the quadratic function terms. The least-squares 10x16 system is ill-conditioned for highly skewed elements. Venditti[14] employ singular value decomposition to handle highly stretched elements. A simple LU is sufficient to invert the equations for the meshes examined in this study.

The quadratic fit is evaluated at each new and existing node. Contributions from all surrounding elements are averaged at these node locations. At the completion of the grid embedding and interpolation step, the linear and higher order interpolated solutions to the primal and dual problems $\mathbf{Q}^L$, λ^L, $\mathbf{Q}^H$, and λ^H are available to compute the error correction and adaptation criteria.

Error Correction and Adaptation Parameter

Once the new embedded grid is constructed with $\mathbf{Q}^H$ and λ^H, it is partitioned to allow parallel calculation of the functional and residuals. The flow and the adjoint equations are not iterated or solved on this embedded grid; instead, the flow state and adjoint variables are interpolated from the original mesh. Therefore, the only computational costs on this larger embedded grid are function evaluations, flow and adjoint residual evaluations, and dot products of vectors. The higher order error correction term, Eq. (9), is computed at each node on the embedded mesh and summed over the entire mesh for all flow equations. The adaptation intensity, Eq. (16), is also computed at each embedded mesh node.

To specify the grid adaptation on the original mesh, the adaptation intensities must be reduced from the embedded mesh to the original mesh ($\mathbf{I}^0$). The new nodes on the embedded mesh all lie on existing edges of the original mesh (see Fig. 1(b)). Therefore, to construct $\mathbf{I}^0$, the original mesh is examined one edge at a time. One half of the intensity computed at each new node (which is at the midpoint of an original edge) is added to each existing node at the endpoints of the new node's parent edge. The intensities are also summed over the equations at this step, resulting in one intensity value for each original node.

The adaptation parameter, which has been reduced to the original mesh, is summed to find the global intensity $I_g = \sum \mathbf{I}^0$. The number of nodes in the original mesh n and the user-specified error tolerance t are combined to scale the adaptation intensity; that is,

$$\mathbf{I}_s^0 = \frac{I_g}{t}\frac{n}{t}\mathbf{I}^0 \tag{17}$$

Previous work[17] refined the mesh when $\mathbf{I}_s^0$ was greater than unity. An anisotropic adaptation metric is applied in the current study to improve the adaptive scheme.

Anisotropic Metric

Venditti and Darmofal[15] incorporated the adjoint adaptation parameter into an anisotropic Hessian-based framework. This combined approach sets the anisotropy of mesh elements by using the Mach Hessian, and it scales the element size so that the tightest spacing is dictated by the adjoint adaptation parameter. Hessian-based adaptation[1,15,36] is formulated to equally distribute the interpolation errors for linear elements throughout the mesh.

The Mach Hessian $\mathbf{H}$ is symmetric and can be decomposed into Eigenvalues Λ and Eigenvectors $\mathbf{R}$; that is,

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 M}{\partial x^2} & \frac{\partial^2 M}{\partial x \partial y} & \frac{\partial^2 M}{\partial x \partial z} \\ \frac{\partial^2 M}{\partial x \partial y} & \frac{\partial^2 M}{\partial y^2} & \frac{\partial^2 M}{\partial y \partial z} \\ \frac{\partial^2 M}{\partial x \partial z} & \frac{\partial^2 M}{\partial y \partial z} & \frac{\partial^2 M}{\partial z^2} \end{bmatrix} = \mathbf{R}\Lambda\mathbf{R}^t \tag{18}$$

A symmetric metric tensor $\mathbf{M}$ is employed to describe a transformation where grid elements are ideal (isotropic with unit length edges). The matrix $\mathbf{M}$ has the same Eigenvectors as matrix $\mathbf{H}$ and absolute value of the corresponding Eigenvalues:

$$\mathbf{M} = \mathbf{R}\begin{bmatrix} |\lambda_1| & & \\ & |\lambda_2| & \\ & & |\lambda_3| \end{bmatrix}\mathbf{R}^t = \mathbf{R}\,|\Lambda|\,\mathbf{R}^t \tag{19}$$

The element spacing h_i in the three principle directions $\mathbf{R}$ is given as

$$\mathbf{M} = \mathbf{R}\begin{bmatrix} \left(\frac{1}{h_1}\right)^2 & & \\ & \left(\frac{1}{h_2}\right)^2 & \\ & & \left(\frac{1}{h_3}\right)^2 \end{bmatrix}\mathbf{R}^t \tag{20}$$

The Jacobian $\mathbf{J}$ is employed to map physical space $\mathbf{x} = [x\ y\ z]^t$ to a transformed space $\mathbf{x}' = [x'\ y'\ z']^t$ where a triangle or tetrahedron becomes equilateral with unit length edges; that is,

$$\mathbf{x}' = \mathbf{J}\mathbf{x} \tag{21}$$

The metric tensor $\mathbf{M}$ is related to $\mathbf{J}$ by

$$\mathbf{M} = \mathbf{J}^t\mathbf{J} \tag{22}$$

$$\mathbf{J} = |\Lambda|^{\frac{1}{2}}\,\mathbf{R}^t \tag{23}$$

If a vector $\mathbf{x}$ describes an edge in physical space, the length l in mapped space is

$$l = \sqrt{\mathbf{x}'^t\mathbf{x}'} \tag{24}$$

Employing Eq. 21, this expression of length becomes

$$l = \sqrt{(\mathbf{J}\mathbf{x})^t\,(\mathbf{J}\mathbf{x})} \tag{25}$$

$$l = \sqrt{\mathbf{x}^t \mathbf{M} \mathbf{x}} \tag{26}$$

Equation (26) is employed by the adaptive node insertion and removal mechanics to compute edge lengths. Equation (22) is applied to the physical coordinates of element nodes to map them to the transformed space before computing shape metrics during edge swapping, face swapping, and node smoothing.

The adjoint adaptation parameter is incorporated into the Hessian framework by scaling the three Eigenvalues so that the largest Eigenvalue corresponds to the adjoint adaptation spacing requirement. The Eigenvalues and corresponding Eigenvectors are sorted so that $|\lambda_1| > |\lambda_2| > |\lambda_3|$. The largest Eigenvalue is computed with the relationship

$$\lambda_1 = \left(\frac{1}{h_1} \right)^2 \tag{27}$$

The specified element length h_1, corresponding to the largest Eigenvalue, is computed with an estimate of the spacing on the original mesh h^0 and the adaptation intensity Eq. (17); that is,

$$h_1 = h^0 \left(\mathbf{I}_s^0 \right)^{0.2} \tag{28}$$

In previous studies, the original spacing h^0 was computed with an implied metric. However, in the current study, h^0 is estimated by the length of the shortest edge incident to a node. The implied metric is described in Ref. 14, 15, and 37. The exponent value of 0.2 is used to under-relax the adaptive procedure.

The Eigenvalue ratios $|\lambda_2| / |\lambda_1|$ and $|\lambda_3| / |\lambda_1|$ can also be increased toward unity to reduce anisotropy of the mesh. Limits are placed on the minimum Eigenvalue to reduce the maximum element size in the mesh. The completed anisotropic adaptation metric is employed to describe the requested grid modifications to the adaptation mechanics.

CAD-to-Grid Methods

The GridEx[38, 39] framework is currently being developed at NASA Langley Research Center to link various grid generation and adaptation strategies to geometry through Computational Analysis Programing Interface[40, 41] (CAPRI). CAPRI is a CAD vendor-neutral package that provides a common interface to many native CAD kernel application programing interfaces (API). Interrogating CAD parts with their native kernel removes the difficulties of translation and associated pitfalls.

The isotropic mesh generator used in this study is the FELISA[4] mesher. FELISA is a Delaunay mesh generator with an advancing-front method for inserting nodes. It is used to compute both surface and volume meshes in this study.

The GridEx application provides various visualization tools and allows interactive adjustment of mesh spacing specifications. It also has a batch mode for generating meshes without visualization. The underlying GridEx framework facilitates data persistence between various applications and application instances. This framework enables meshing research into mesh adaptation and viscous grid generation.

Adaptation Mechanics

The adaptation mechanics are separated into multiple modules. These modules are written in C and wrapped with the Ruby[42, 43‡] script language. The C code was unit tested with the Ruby `Test::Unit` package. The relative ease of testing with this package enabled test first development.[44] GNU Autotools[45] are used to configure, build, and install this grid refinement package.

The first module inserts or removes nodes of an existing mesh and locally reconnects tetrahedra and boundary faces to maintain a valid tessellation. The second module employs face and edge swapping to improve the mesh quality. The final two modules perform node smoothing and boundary node projection operations. These modules are incorporated into a common framework that stores auxiliary data such as adjacency of elements and CAD parameterizations and allows translation between the GridEx framework and internal data structures.

Node Insertion and Removal

The node insertion and removal module allows for the local density of the mesh to be modified. Nodes may be inserted into an element or be created by splitting an edge or face. Nodes are removed by edge collapse.

An adaptation sweep is performed by iterating over nodes in the mesh. For each node, the longest and shortest incident edge in the anisotropic metric mapped space is computed with Eq. (26). If the longest edge is found to be longer than the spacing metric by a user-defined percentage, it is split. If all edges are found to be shorter than the spacing metric by a defined percentage then the shortest edge is collapsed. No edges are modified if the edge lengths are all within tolerance of the spacing metric. After a successful edge split or collapse, a cloud of local elements is examined for improved configurations with the face and edge swapping routines.

Face and Edge Swapping

The connectivity improvement scheme employs face and edge swapping[31] in metric mapped space, Eq. (22). The swapping algorithm maximizes a shape function (aspect ratio). Reconnections of tetrahedra with undesirable shape measures are investigated, and new local tetrahedra configurations with more desirable shape measures are selected. Edges on boundary

‡ `http://ruby-lang.org/`

faces can also be swapped. The actual locations of nodes are not modified in this module; that modification is performed by the node smoothing and projection modules.

Node Smoothing

The locations of the nodes in the mesh are modified to improve the overall quality (shape measure) of the mesh elements. This smoothing method is from Freitag.[46] The elements are mapped with the anisotropic metric, Eq. (22), during node smoothing, so that well-shaped elements in mapped space yield anisotropic elements in physical space.

If a node is connected to a poorly shaped element, it is selected for node smoothing. The first step is to differentiate the shape function of all incident elements with respect to the current node location. The search direction and step size are calculated with the nonsmooth optimization method of Freitag.[46] Smart-Laplacian smoothing is also used on the interior nodes as in the combined approach of Freitag.[46]

Boundary node smoothing is coupled to CAPRI, which uses native CAD evaluation routines. Nodes on the boundary are smoothed by moving the nodes in (u, v) CAD parametric space to improve the shape measure of adjacent tetrahedra and boundary faces. It is much faster and more robust to evaluate points with a (u, v) CAD parameter than to project the points to the CAD surface in physical space.

To optimize a node in (u, v) space, the search direction in physical space is first calculated. This search direction is simply the gradient of the shape measure of the worst quality element adjacent to a node. The local derivatives of the CAD model parameterization are evaluated at the node location and the chain rule is used to express the physical search direction in (u, v). A line search is performed in the (u, v) search direction to optimize the boundary node location. Nodes constrained to CAD edge entities are optimized with a line search for the single (t) parameter.

Node Projection

Inserted boundary nodes may not be located on the surface geometry of the model to be simulated because they were linearly inserted at the midpoints of existing edges. A CAD model is employed to describe the actual model surface. To regain the surface fidelity of the mesh, the newly inserted boundary nodes are projected to the model surface with CAPRI. The projection of these new nodes to their location on the CAD surface can result in inverted, invalid tetrahedral elements, especially for anisotropic elements or highly curved geometries.

The grid smoothing and swapping algorithms are employed to facilitate boundary projection without generating invalid elements. As the nodes are projected, the neighboring tetrahedra are tested for validity. If negative volume tetrahedra result from the projection, the projection displacement of the boundary nodes is iteratively reduced until the neighboring tetrahedra are valid. Then the nodes in the neighborhood of the projected node are smoothed to improve a quality measure of the adjacent tetrahedra. Connectivity is also swapped. The boundary points are then moved into the fully projected position in a number of iterative cycles.

It is anticipated that grid smoothing in the neighborhood of projected nodes may not adequately regain surface fidelity of highly anisotropic 3D meshes. A grid-movement scheme may be required as in Ref. 19. Another possibility is a 3D version of mesh restructuring as in Ref. 47.

Adaptation Module Development

These three adaptation modules were originally developed independently to facilitate a quick initial implementation and to investigate the strengths and weaknesses of each technique. They were then refactored or rewritten to merge the abilities of these separate modules into a common framework that allowed for more flexible modifications of grids (e.g., point insertion, point removal, and anisotropic elements).[4,6] This framework has also been extended to generate advancing layer viscous grids.

Viscous Grid Generation

Advancing layer[4,6,32–35] mesh generation is used by many researchers to generate strongly anisotropic regions in unstructured meshes for viscous fluid simulations. This technique has been implemented in the current adaptation framework to generate initial viscous meshes for this study. The advancing layer capability enables research into anisotropic mesh generation. This research includes a mixed element capability, but only tetrahedral meshes were analyzed in this study.

Mixed elements can be generated with extruded prisms that can be optionally subdivided into tetrahedra. Growth curves can be terminated by a GridEx framework spacing metric, a maximum number of layers, or maximum growth curve length. Transition elements (pyramids or tetrahedra) are created to span layers which have growth curves terminated at different levels.

Multiple normals (blend elements) are optionally created at surface discontinuities (e.g., sharp corners such as a wing trailing edge). Blends can be extruded to create a zero thickness layer for wake structures. The incremental length, growth rate, and maximum length of each normal growth curve can be set independently or described with user-defined polynomial function. Growth curves may be constrained to CAD face or edge entities. These constrained curves create new boundary face and edge elements in the path of the layer as it advances.

Once all the growth curves have been terminated,

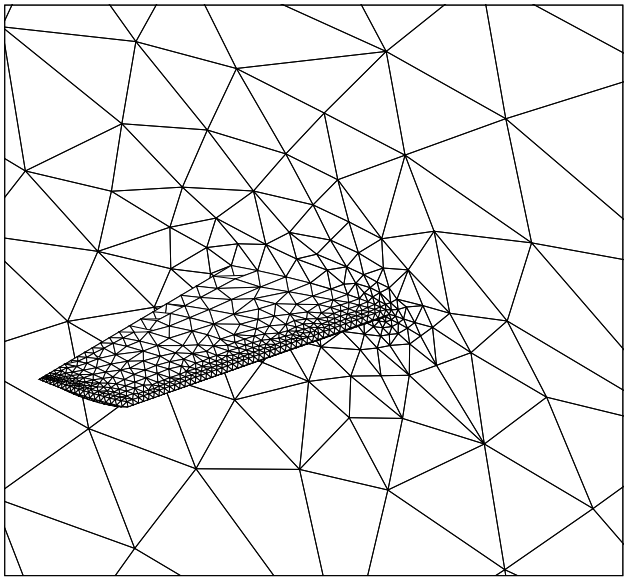


Fig. 2 Initial ONERA M6 mesh.

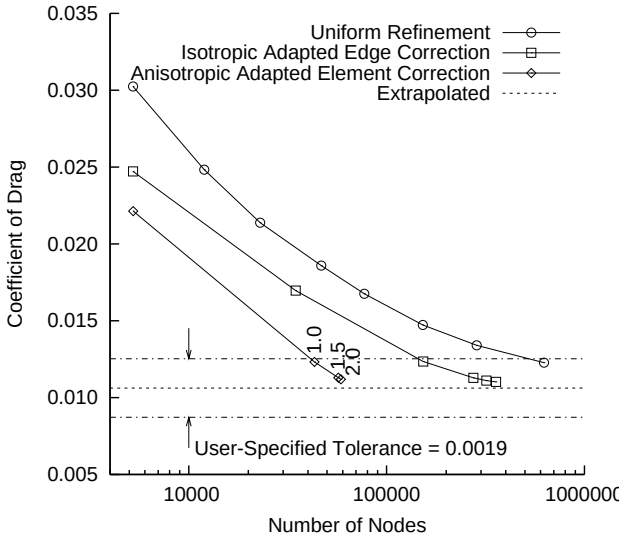


Fig. 3 Adapted ONERA M6 drag.

isotropic surface and volume meshes are created with the GridEx framework and merged with the viscous layers. The isotropic volume mesh is bounded by the final layer, rebuilt faces, and any exposed original faces of the model. The final layer may not exactly match the inviscid spacing specification, so the FELISA mesher was modified to increase robustness when the initial boundary grid of the volume is of poor quality. The gain in robustness was a result of lowering the minimum quality requirement of the initial tessellation in order to increase the likelihood of recovering the boundary faces. Subsequent smoothing, as a part of normal FELISA operation, is sufficient to recover good grid quality.

Results

Adaptation results are shown for a wing configuration. Error prediction results are shown for a cylinder and an extruded airfoil configurations. The wing is simulated with inviscid transonic flow conditions, the cylinder is simulated with low Reynolds number laminar flow, and the extruded airfoil configuration is simulated with turbulent subsonic flow.

Inviscid Transonic ONERA M6 Adaptation

An Office National d'Etudes et de Recherches Aérospatiales (ONERA) M6 wing configuration is simulated at 0.84 Mach and an angle of attack of 3 deg. This case is presented as a continuation of a study in Ref. 17. A uniform refinement study of the configuration was performed for comparison with adjoint drag adaptation schemes. The geometry for this model is represented with the CAPRI native kernel. The native part was created from a IGES surface definition. The initial mesh for the wing adaptation and error prediction study is generated with the FELISA mesher connected to geometry by CAPRI and the GridEx framework.

The initial mesh contains 5227 nodes and is shown in Fig. 2. The initial spacing function for this mesh is specified manually and is intended to be representative of an automated curvature or maximum chord-height specification. The mesh has extremely coarse spacing, especially at the trailing edge, and is intended to resolve the surface curvature of the leading edge and the wingtip and not any particular flow features.

Figure 3 shows drag and estimates of drag as a function of mesh size for the ONERA M6 wing. A refinement study of the mesh was performed to describe the convergence of drag calculated with uniformly refined grids (represented by circles in Fig. 3). These meshes have the same spacing function as the original mesh globally modified with a scalar to uniformly reduce the element spacing. The extrapolated value for drag (dashed line) was calculated with second-order Richardson extrapolation of the two finest embedded grids.[17] These grids were generated with the batch version of GridEx.

The uniformly refined grid drag values are compared to adjoint (output) adaptation from previous work[17] and the current approach, represented in Fig. 3 by squares and diamonds respectively. The user specified error tolerance for final adapted drag error is ± 0.0019 for both adaptive schemes. Drag values within this tolerance are within the borders of the dot-dash line. The previous scheme only allowed isotropic refinement, and an edge-based interpolation scheme was employed to prolongate the solution to the embedded mesh for error prediction. The current anisotropic refinement and coarsening scheme uses an improved element-based interpolation. The numbers next to the diamond symbols denote the maximum allowed metric length ratio (h_2/h_1 and h_3/h_1), which is controlled by limiting the Eigenvalue ratios of the adaptation metric. A limit of one yields an isotropic metric.

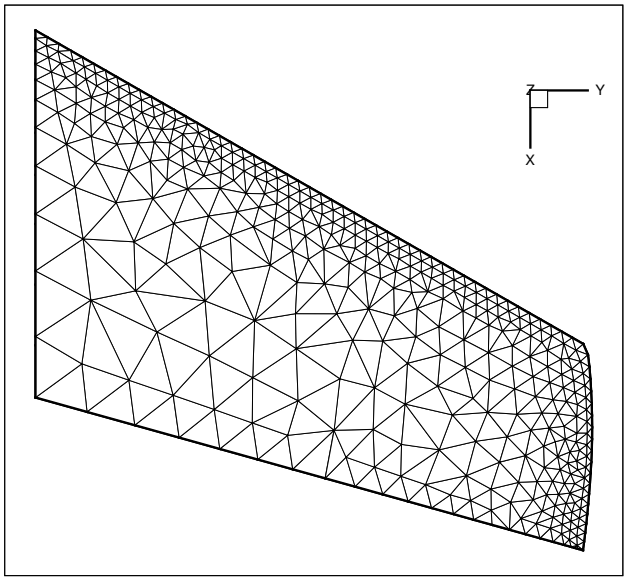


Fig. 4 Original ONERA M6 upper wing surface mesh.

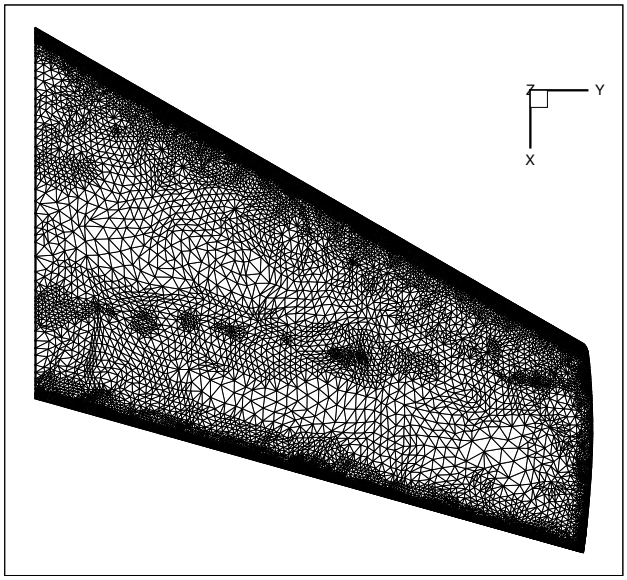


Fig. 5 Isotropic adapted ONERA M6 upper wing surface mesh.

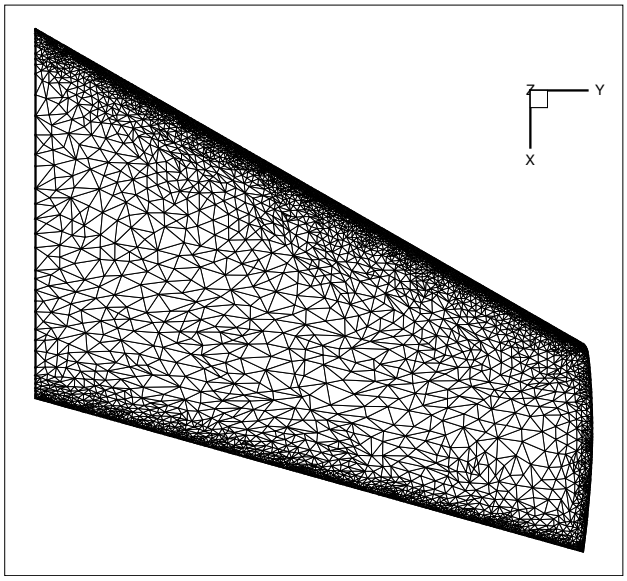


Fig. 6 Anisotropic adapted ONERA M6 upper wing surface mesh.

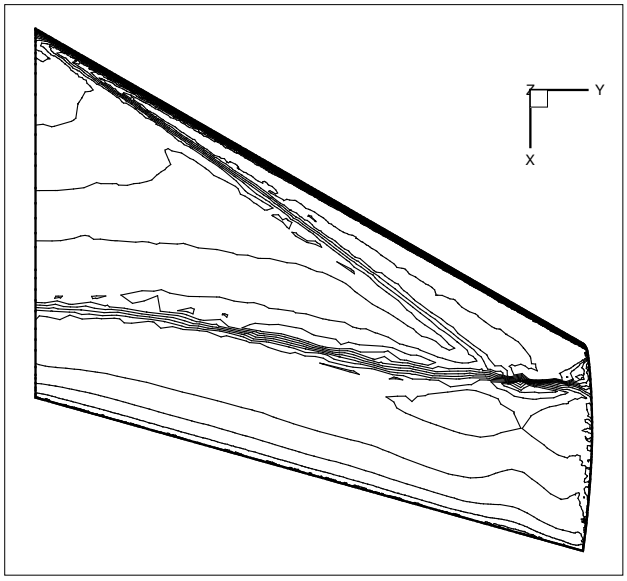


Fig. 7 Anisotropic adapted ONERA M6 upper wing surface Mach contours.

The element-based error correction provides the best approximation for the extrapolated value on the initial mesh. This improvement over the edge-based method is expected due to the smoother interpolant produced by the element-based interpolation. The anisotropic refinement and coarsening adaptation scheme yields a smaller mesh than the isotropic refinement-only scheme for an equivalent error level. The final adapted anisotropic mesh size may also be reduced further with the use of higher aspect-ratio meshes. The current approach is limited by the inherent difficulty of projecting nodes to geometry with high curvatures. This difficulty is exacerbated by highly anisotropic elements near these bounding geometries with high curvature.

The initial upper wing surface mesh is shown in Fig. 4. The final adapted upper wing surface mesh for the previous isotropic adaptation scheme is shown in Fig. 5, and the current anisotropic adaptation scheme is shown in Fig. 6. The integrated smoothing and swapping yields a smoother mesh size distribution. The incidence of high-degree nodes is lower for the anisotropic scheme than the isotropic refinement-only scheme. Both methods have a high degree of mesh clustering at leading and trailing edges.

The Mach contours on the final anisotropic adapted upper wing surface are shown in Fig. 7. A lambda shock structure is visible in these Mach contours. The anisotropic mesh has larger elements in the neighborhood of the lambda shock structure than the isotropic

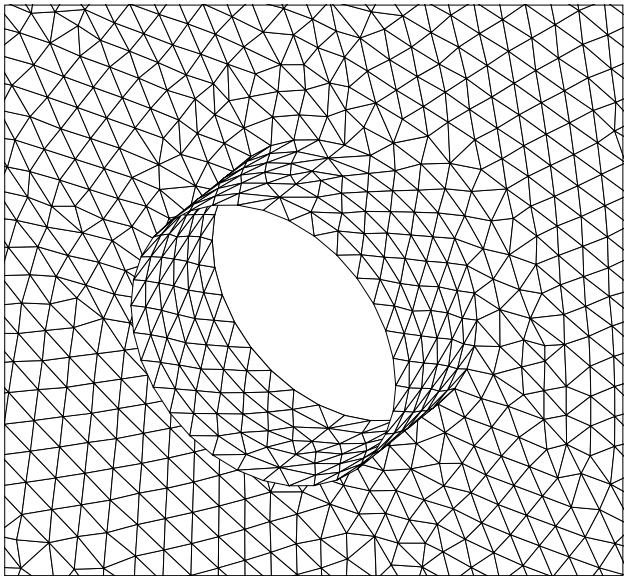


Fig. 8 Initial cylinder mesh with the near symmetry plane removed.

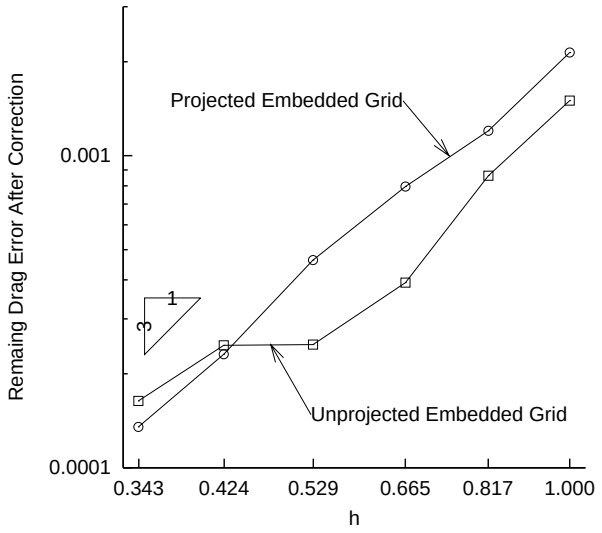


Fig. 9 Remaining drag error after correction on the embedded grid.

refinement-only scheme. The shock location may have a larger impact on drag calculation than shock resolution. The grid enrichment upstream of the shocks near the stagnation location and sonic acceleration region has a larger impact on shock location and therefore drag than grid resolution at the shock.

Laminar Cylinder

A 3D cylinder is simulated in laminar flow at 0.38 Mach and a Reynolds number of 10. The cylinder is capped at each end by a symmetry plane and has a height of half its diameter. The outer boundary for the problem is placed at a radius of 40 diameters. The geometry for this case is generated from Parasolid cylinder primitives and is accessed with CAPRI. A suite of isotropic, uniformly refined FELISA meshes were created for this case with the GridEx application. The initial grid for the configuration is shown in Fig. 8 with the near symmetry plane removed. The skin friction component of drag is computed with the flux-based skin friction calculation of Refs. 14 and 15.

Figure 9 depicts the remaining error in the corrected functional. This remaining error is the difference between the corrected drag value and the converged solution on the embedded grid. A characteristic length h for each grid is estimated as the cube root of the number of nodes in the mesh. The characteristic lengths are normalized by the characteristic length of the coarsest mesh $\sqrt[3]{31343}$. The error corrections were performed on projected (circles) and unprojected (squares) embedded grids. The converged solutions were calculated on projected embedded grids.

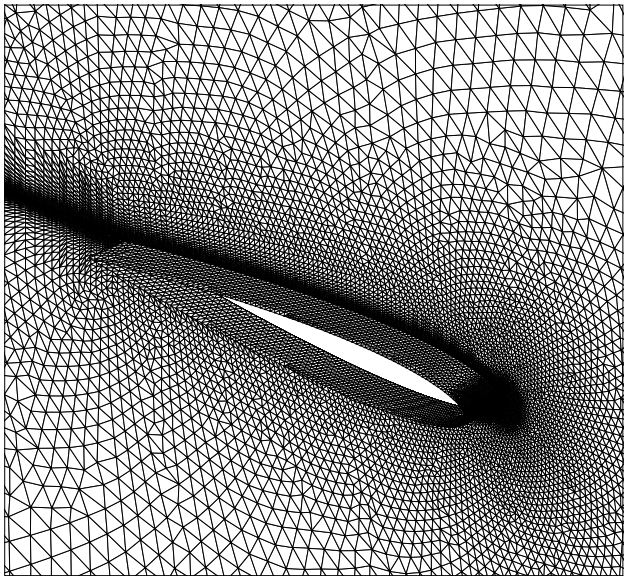
The corrected drag value computed on the unprojected grid better predicts the drag value calculated on the embedded grid for coarse meshes. The embedded drag calculation on the finest mesh is better predicted by the projected grid error prediction. The projected grid error correction appears to converge with a more consistent slope than the unprojected grid error correction. The slope for the projected mesh steepens for each consecutive mesh after the coarsest mesh, indicating an increasing rate of convergence. References 13–15 employed projected embedded grids to compute error-corrected functionals. Projecting embedded meshes has been avoided in 3D due to increased cost, complexity, and robustness issues. The embedded mesh size increases by factors of eight in 3D.

Turbulent Extruded Airfoil

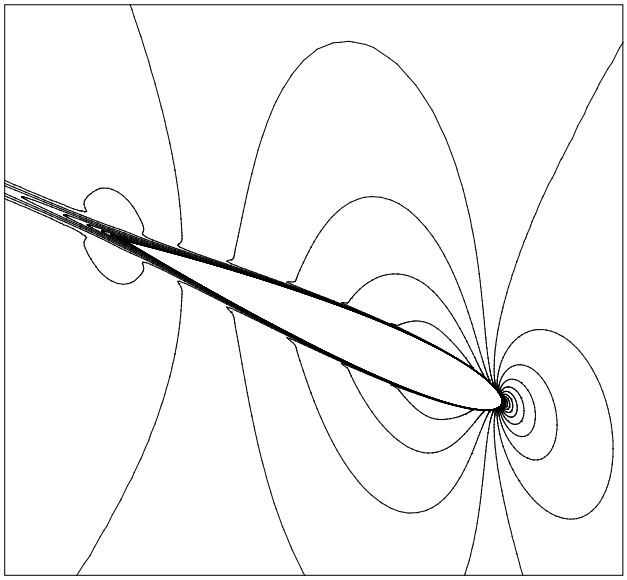
The initial mesh for the the extruded airfoil configuration is shown with the near symmetry plane removed in Fig. 10(a). The airfoil is a NACA 0012 extruded 0.1 chord lengths. Viscous grids were generated from a Parasolid part with the GridEx framework and the adaptation framework with the advancing layer viscous extension. The isotropic portion of the mesh was completed with the FELISA mesher. To create this mesh, the geometry was split along the airfoil centerline. Half of the mesh is created (including the wake structure) and then mirrored to complete the mesh.

The extruded NACA 0012 configuration is simulated at 0.70 Mach, 0 deg angle of attack, and 9 million Reynolds number. Mach contours are presented for the far symmetry plane in Fig. 10(b). The contour lines range from 0.0 to 0.95 Mach and are shown in 0.05 Mach increments. The flow is subcritical, and airflow is from lower right to upper left. The wake and the extent of the boundary layer are clearly visible.

Close-up views of the symmetry plane grid and the drag adaptation intensity, Eq. (17), are shown for the initial mesh in Fig. 11. Airflow is from left to right. The dark gray areas in Fig. 11(b) are regions with a large adaptation intensity, which indicates a need for

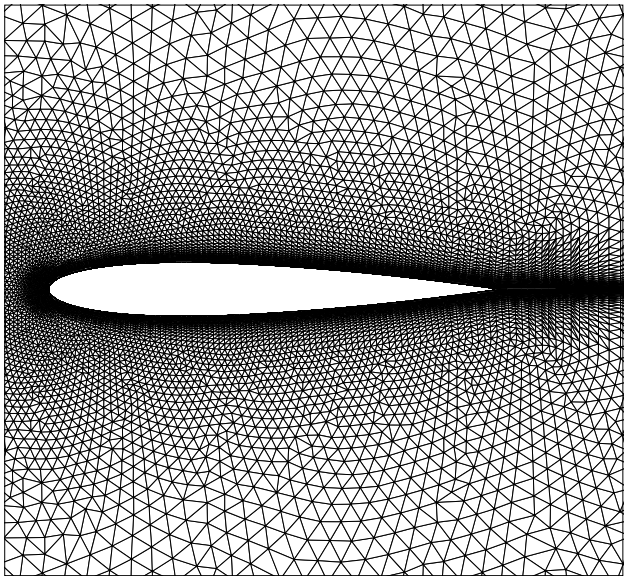


a) Initial NACA 0012 mesh with the near symmetry plane removed.

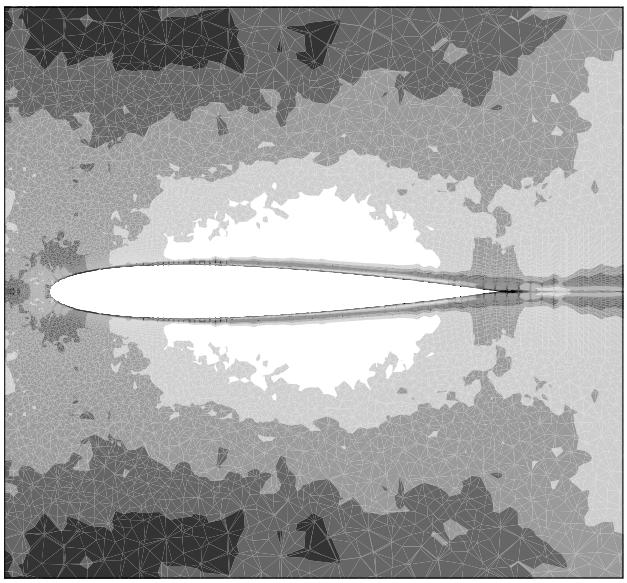


b) Initial NACA 0012 Mach contours on symmetry plane.

Fig. 10 Initial NACA 0012.



a) Symmetry plane mesh.



b) Drag adaptation intensity (dark areas have large intensities).

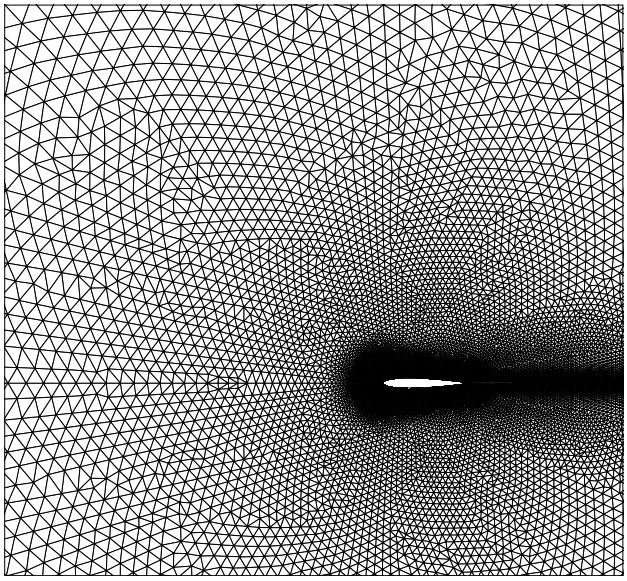
Fig. 11 Initial NACA 0012 close-up.

increased grid resolution in those regions. White areas in Fig. 11(b) are sufficiently resolved (possibly over-resolved) and have an adaptation intensity less than or equal to unity. The area of large adaptation intensity (dark gray) at the edge of the boundary layer (Fig. 11(b)) appears to correspond to high curvature of the Mach contour lines in Fig. 10(b).

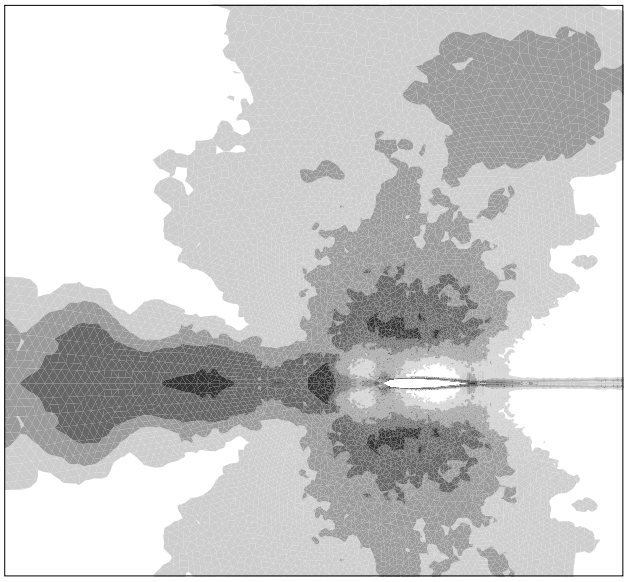
The mechanics to automatically adapt 3D grids with highly anisotropic regions is still under development. Therefore, the advancing layer structure near the airfoil and in the wake can currently only be affected by indirect, manual modification to the advancing normal initial heights and growth rates. Improved mesh adaptation mechanics would allow a more efficient allocation of grid spacing to satisfy the adaptation intensity.

A close-up view of these high intensity areas is available in Fig. 11. Resolving the wake and the outer edge of the boundary layer is clearly important to drag calculation, although these areas are often neglected when constructing unstructured meshes for drag prediction investigations.

The symmetry plane grid and the drag adaptation intensity, Eq. (17), are shown for the initial mesh in Fig. 12. The dark gray areas in Fig. 12(b) are regions with a large adaptation intensity, which indicates a need for increased grid resolution in those regions. The adaptation intensity is employed to manually modify

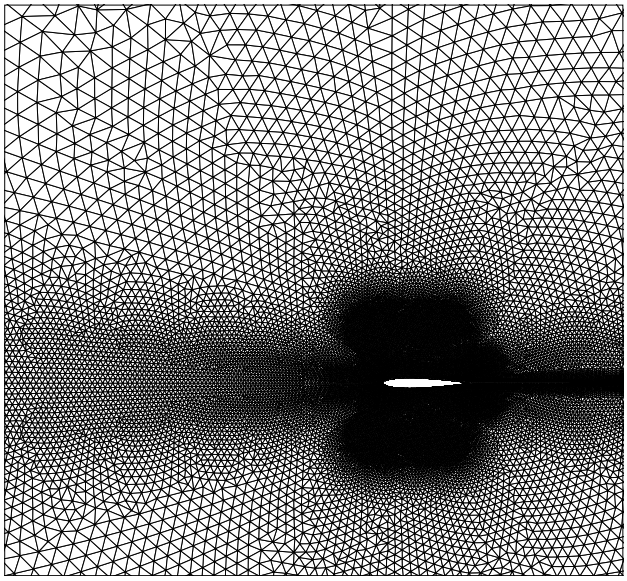


a) Symmetry plane mesh.

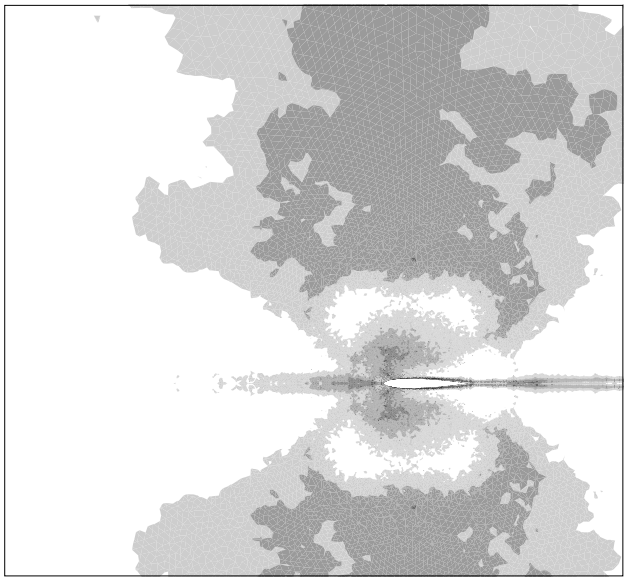


b) Drag adaptation intensity (dark areas have large intensities).

Fig. 12 Initial NACA 0012 (211733 nodes).



a) Symmetry plane mesh.



b) Drag adaptation intensity (dark areas have large intensities).

Fig. 13 Manually adapted NACA 0012 (369417 nodes).

the inviscid spacing specification of the grid. The dark areas upstream, above, and at the tail of the airfoil were targeted for adaptation. A new grid and resulting adaptation parameter are shown in Fig. 13. Figure 13(a) depicts the enriched grid areas upstream, above, and at the tail of the airfoil. The adaptation intensity is reduced from dark gray to light gray or white in the areas of grid enrichment, as shown in Fig. 13(b).

Drag and corrected drag estimates are shown as a function of grid size in Fig. 14. The computed drag values are shown with open symbols and the corrected drag estimates are depicted with closed symbols. Second-order Richardson extrapolation is performed for the two finest mesh drag values. The Richardson extrapolation values is denoted approximate because the drag value has not reached asymptotic convergence. The drag and the drag adjoint solution are computed with the flux-based skin friction. Drag on the embedded grid, which is part of the estimated corrected drag value, is computed with element-based skin friction. The flux-based skin friction calculation did not give reasonable values for the interpolated solution on the embedded grid. These unpredictable results may be due to high frequency noise in the embedded solution. The turbulence model quantity $\tilde{\nu}$ is floored at zero on the embedded mesh to prevent negative turbulent viscosity components.

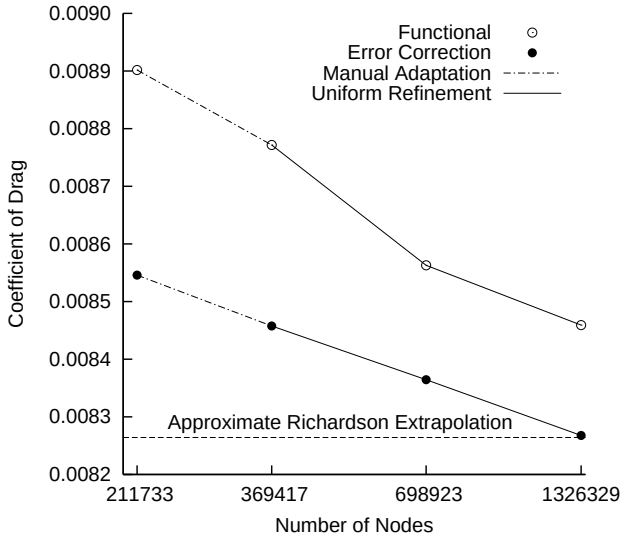


Fig. 14 NACA 0012 drag convergence.

Conclusion

The initial three-dimensional implementation of an adjoint-based error correction method is demonstrated for compressible Euler, laminar, and turbulent Navier-Stokes flow. With a given flow and adjoint solution, the error correction for a functional is described. The treatment of high aspect-ratio meshes and strong boundary conditions are discussed.

The spatial convergence for drag and error corrected drag is shown for a wing in transonic inviscid flow. A refinement and coarsening scheme was combined with an anisotropic adaptation framework to calculate accurate functionals on smaller meshes than refinement only meshes.

The drag computed by this error correction method was shown to be as accurate as direct flow calculations using uniformly refined grids for a cylinder in laminar flow, and an extruded airfoil in subcritical turbulent flow. The adaptation intensity is shown for a turbulent extruded airfoil configuration. Manual adaptation and uniform refinement is performed on this airfoil configuration. The need for continued research into automatic, highly anisotropic mesh adaptation mechanics is highlighted.

Acknowledgments

The author wishes to thank Dr. Eric Nielsen for his help in employing the FUN3D flow and adjoint solvers, Dr. David Venditti and Dr. David Darmofal for their suggestions and guidance in error prediction and adaptation, William Jones and William Kleb for aid in developing the adaptation mechanics and advancing layer mesher, Elizabeth Lee-Rausch for discussions on drag prediction and viscous boundary layers, Dr. James Thomas and Long Yip for their support of this work, and the Fast Adaptive AeroSpace Tools (FAAST) development team.

References

[1]Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C., "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, Vol. 72, 1987, pp. 449–266.

[2]Baker, T. J., "Mesh Adaptation Strategies for Problems in Fluid Dynamics," *Finite Elements in Analysis and Design*, Vol. 25, 1997, pp. 243–273.

[3]Mavriplis, D. J., "Adaptive Meshing Techniques for Viscous Flow Calculations on Mixed Element Unstructured Meshes," AIAA Paper 97–0857, 1997.

[4]Peraire, J. and Morgan, K., "Unstructured Mesh Generation Including Directional Refinement for Aerodynamic Flow Simulation," *Finite Elements in Analysis and Design*, Vol. 25, 1997, pp. 343–356.

[5]Braack, M. and Rannacher, R., "Adaptive Finite Element Methods for Low-Mach Flows with Chemical Reactions," Vol. 3 of *30th Computational Fluid Dynamics*, von Karman Institute, 1999, pp. 1–93.

[6]Löhner, R., "Generation of Unstructured Grids Suitable for RANS Calculations," AIAA Paper 99–0662, 1999.

[7]Pirzadeh, S. Z., "A Solution-Adaptive Unstructured Grid Method by Grid Subdivision and Local Remeshing," *AIAA Journal of Aircraft*, Vol. 37, No. 5, September–October 2000, pp. 818–824, See also AIAA Paper 99–3255.

[8]Park, M. T. and Kwon, O. J., "Unsteady Flow Computations Using a 3-D Parallel Unstructured Dynamic Mesh Adaptation Algorithm," AIAA Paper 2001–0865, 2001.

[9]Monk, P. and Suli, E., "The Adaptive Computation of Far-Field Patterns by A Posteriori Error Estimation of Linear Functionals," *SIAM Journal on Numerical Analysis*, Vol. 8, 1998, pp. 251–274.

[10]Paraschivoiu, M., Peraire, J., and Patera, A., "A Posteriori Finite Element Bounds for Linear-Functional Outputs of Elliptic Partial Differential Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 150, 1997, pp. 289–312.

[11]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.

[12]Venditti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal Computational Physics*, Vol. 164, 2000, pp. 204–227, see also AIAA Paper 99–3292.

[13]Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal Computational Physics*, Vol. 176, 2002, pp. 40–69, see also AIAA Paper 2000–2244.

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

[15]Venditti, D. A. and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows," *Journal Computational Physics*, Vol. 187, 2003, pp. 22–46.

[16]Müller, J. D. and Giles, M. B., "Solution Adaptive Mesh Refinement Using Adjoint Error Analysis," AIAA Paper 2001–2550, 2001.

[17]Park, M. A., "Adjoint–Based, Three–Dimensional Error Prediction and Grid Adaptation," AIAA Paper 2002–3286, 2002.

[18]Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," AIAA Paper 2001–0596, 2001.

[19]Nielsen, E. J. and Anderson, W. K., "Aerodynamic Design Optimization On Unstructured Meshes Using the Navier-Stokes Equations," AIAA Paper 98–4809, 1998.

[20]Fast Adaptive Aerospace Tools (FAAST) development team, "Opportunities for Breakthroughs in Large-Scale Com-

putational Simulation and Design," NASA TM 211747, NASA Langley Research Center, June 2002.

[21]Anderson, W. K. and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No. 1, 1994, pp. 1–22.

[22]Anderson, W. K., Rausch, R. D., and Bonhaus, D. L., "Implicit/Multigrid Algorithm for Incompressible Turbulent Flows on Unstructured Grids," *Journal of Computational Fluids*, Vol. 128, 1996, pp. 391–408.

[23]Nielsen, E. J., *Aerodynamic Design Sensitivities on an Unstructured Mesh Using the Navier-Stokes Equations and a Discrete Adjoint Formulation*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 1998.

[24]Spalart, P. R. and Allmaras, S. R., "One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92–0429, 1992.

[25]Taylor, A. C., Green, L. L., Newman, P. A., and Putko, M. M., "Some Advanced Concepts in Discrete Aerodynamic Sensitivity Analysis," AIAA Paper 2001–2529, 2001.

[26]Giles, M., "On the Use of Runge-Kutta Time-Marching and Multigrid for the Solution of the Steady Adjoint Equations," Technical Report NA00/10, 2000.

[27]Giles, M., "Adjoint Code Developments Using the Exact Discrete Approach," AIAA Paper 2001–2596, 2001.

[28]Lu, J. and Darmofal, D. L., Private Communication, Massachusetts Institute of Technology.

[29]Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An Exact Dual Discrete Adjoint Solution Method for Turbulent Flows on Unstructured Grids," AIAA Paper 2003–0273, 2003.

[30]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856–869.

[31]Freitag, L. A. and Ollivier-Gooch, C., "Tetrahedral Mesh Improvement Using Swapping and Smoothing," *International Journal for Numerical Methods in Engineering*, Vol. 40, 1997, pp. 3979–4002.

[32]Pirzadeh, S. Z., "Three-Dimensional Unstructured Viscous Grids by the Advancing-Layers Method," *AIAA Journal*, Vol. 34, No. 1, September–October 1996, pp. 43–49.

[33]Garimella, R. V., *Anisotropic Tetrahedral Mesh Generation*, Ph.D. thesis, Rensselaer Polytecnic Institute, 1998.

[34]Garimella, R. V. and Shephard, M. S., "Boundary Layer Meshing for Viscous Flows in Complex Domains," 7th International Meshing Roundtable, Sandia National Lab, October 1998, pp. 107–118.

[35]Marcum, D. L., "Advancing–Front/Local–Reconnection (AFLR) Unstructured Grid Generation," *Computational Fluid Dynamics Review*, 1998, pp. 140.

[36]Castro-Diáz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic Unstructured Mesh Adaptation for Flow Simulations," *International Journal of Numerical Methods in Fluids*, Vol. 25, 1997, pp. 475–491.

[37]Labbé, P., Dompierre, J., Vallet, M.-G., Guibault, F., and Trépanier, J.-Y., "A Measure of the Conformity of a Mesh to an Anisotropic Metric," 10th International Meshing Roundtable, Sandia National Lab, October 2001, pp. 319–326.

[38]Jones, W. T., "An Open Framework for Unstructured Grid Generation," AIAA Paper 2002–3192, 2002.

[39]Jones, W. T., "GridEx – An Integrated Grid Generation Package for CFD," AIAA Paper 2003–4129, 2003.

[40]Haimes, R., "Automatic Generation of CFD-Ready Surface Triangulations from CAD Geometry," AIAA Paper 99–0776, 1999.

[41]Haimes, R., "CAPRI: Computational Analysis PRogramming Interface," *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, July 1998.

[42]Matsumoto, Y., *Ruby in a Nutshell*, O'Reilly, Sebastopol, CA, 2002.

[43]Thomas, D. and Hunt, A., *Programming Ruby: The Pragmatic Programmer's Guide*, Addison-Wesley, New York, 2001.

[44]Beck, K., *Extreme Programming Explained*, Addison-Wesley, Reading, MA, 2000.

[45]Vaughan, G. V., Elliston, B., Tromey, T., and Taylor, I. L., *GNU Autoconf, Automake, and Libtool*, New Riders, Indianapolis, IN, 2001.

[46]Freitag, L. A., "On Combining Laplacian and Optimization-Based Smoothing Techniques," *ASME*, Vol. 220, July 1997, pp. 37–43.

[47]Mavriplis, D., "Turbulent Flow Calculations Using Unstructured and Adaptive Meshes," ICASE Report No. 90-61, NASA CR-182102, 1990.