AIAA JOURNAL
Vol. 32, No. 4, April 1994

Automated Adaptive Time-Discontinuous Finite Element Method for Unsteady Compressible Airfoil Aerodynamics

B. E. Webster,* M. S. Shephard,† Z. Rusak,‡ and J. E. Flaherty§
Rensselaer Polytechnic Institute, Troy, New York 12180

An adaptive finite element methodology for unsteady compressible aerodynamics is presented. The automated adaptive environment consists of the finite quadtree automatic mesh generator, a time-discontinuous Galerkin least-squares finite element flow solver, an error indicator based on interpolation estimates, and a mesh enrichment procedure. Specifically, a linear space-time wedge element formulation was developed and implemented for two-dimensional problems with relative body motion. The mesh enrichment procedure is developed to enable the solution process to be spatially adaptive (*h* refinement), is edge based, and stores no mesh enrichment history. Unsteady transonic airfoil calculations show good correlation with available experimental data.

I. Introduction

THE accurate treatment of modern aerodynamic calculations of both fixed and rotary wing vehicles operating in various flight conditions requires the capability to efficiently resolve the widely varying length and time scales that are present in their complex flowfields. Recognition of this requirement has lead to the integration of mesh generators, flow solvers, error indicators, and mesh update procedures into adaptive computational fluid dynamics (CFD) solution methodologies. This paper focuses on the development of an automated adaptive finite element method[1] suitable for unsteady and compressible airfoil aerodynamic calculations, whereas Ref. 2 focuses on the application of the automated adaptive method to various compressible airfoil aerodynamic calculations.

Compressible aerodynamic problems have been successfully computed in the past with finite difference, finite volume, and finite element methods.[3–8] In particular, efforts to develop numerical solutions for unsteady flows with time-dependent boundaries are found in Refs. 1, 2, and 9–21. Adaptive procedures using unstructured meshes have also been developed and successfully applied to both two-dimensional and three-dimensional steady compressible aerodynamic problems using finite volume and finite element methods.[7,8,22,23] In fact, adaptive procedures have also been applied during the past few years to unsteady transonic aerodynamic problems using finite volume techniques.[22,24] On the other hand, during the past five years, the time-discontinuous Galerkin least-squares (GLS) finite element method for calculating compressible flows, which is ideally suited for problems with multiple relative motion and adaptive procedures, has come to maturity.[6,23,25,26]

This paper presents an automated adaptive technique that uses the time-discontinuous GLS finite element method as its flow solver to calculate unsteady transonic airfoil aerodynamics. In addition to the flow solver, the automated adaptive environment is composed of an automatic mesh generator, a discretization error estimator or indicator, and a mesh enrichment procedure. A schematic description of the automated adaptive environment developed here is given in Fig. 1. This adaptive methodology discretizes the flow domain (i.e., develops a valid mesh for the given problem) and adjusts this discretization in time (i.e., adaptively modifies the mesh) so that the unsteady and spatial physics of the problem can be resolved accurately and efficiently.

Selection of a flow solver based on finite elements allows for unstructured meshes. Therefore, unstructured mesh generators and mesh enrichment procedures are used in the adaptive environment developed. The finite quadtree mesh generator,[27] an interpolation-based error indicator,[28] and an edge-based mesh enrichment procedure[1] are the specific components used in the present methodology.

Section II provides a brief description of the time-discontinuous GLS finite element flow solver.[1,6] Section III provides a brief discussion of the automated adaptive environment.[1] Finally, Sec. IV describes an unsteady transonic airfoil calculation based on this automated adaptive finite element method. The calculated results show a good agreement with experimental data and substantiate that this type of adaptive procedure based on a time-discontinuous GLS finite element flow solver can accurately resolve unsteady compressible flow characteristics for maneuvering airfoils.

II. Time-Discontinuous GLS Finite Element Method

The time-discontinuous GLS finite element method was developed over the last decade by Shakib,[6] Johnson,[25] Hughes et al.,[26] Hansbo and Johnson,[29] Hughes,[30] and Johnson and Szepessy[31] to be a general numerical method for the solution of compressible flow problems including shock waves. In these

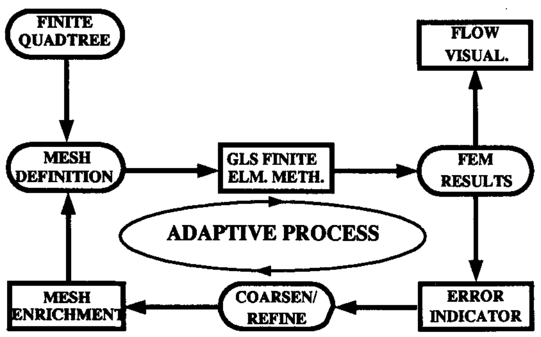

Fig. 1 Automated adaptive environment.

Presented as Paper 93-0339 at the AIAA 31st Aerospace Sciences Meeting, Reno, NV, Jan. 11–14, 1993; received March 1, 1993; revision received Sept. 30, 1993; accepted for publication Oct. 6, 1993. Copyright © 1992 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Army Research Office Rotorcraft Fellow; currently Manager, Modeling and Mesh Generation, Centric Engineering Systems, Inc., 3801 East Bayshore Road, Palo Alto, CA 94303. Member AIAA.

†Director, Scientific Computation Research Center, and Johnson Chair of Engineering. Associate Fellow AIAA.

‡Assistant Professor, Mechanical Engineering, Aeronautical Engineering, and Mechanics. Senior Member AIAA.

§Amos Eaton Professor, Department of Computer Science. Member AIAA.

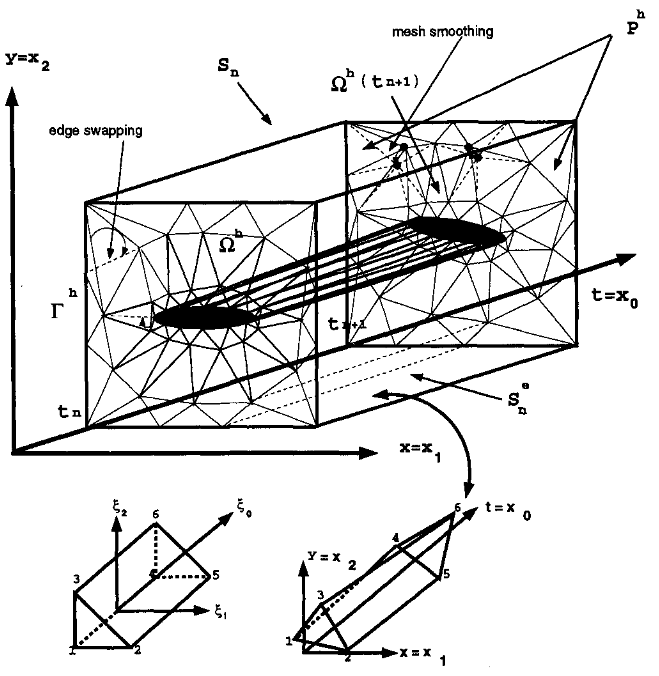

Fig. 2 Illustration of space-time slab with linear wedge elements.

efforts, the GLS method was shown to provide solutions that are consistent with the second law of thermodynamics, i.e., the entropy inequality. Recently, the GLS method has been successfully applied to two-dimensional fluid flow problems involving arbitrary relative body motion.[1,18,20,21,32] For time-dependent problems, the GLS method uses basis functions that are continuous in space but discontinuous in time. Figure 2 illustrates the nth space-time "slab" $S_n = \Omega \times I_n$ in two dimensions and how space-time wedge finite elements are used to discretize S_n in such a way that the temporal motion of a body can be handled. Specifically, all temporal effects due to the motion of the body are calculated directly by the Jacobian that relates the physical space-time coordinates with the local finite element space-time coordinates. In this way, the unsteady effects of multiple bodies with different rigid-body and elastic motions can also be simulated.

The general form of a conservation law used for unsteady inviscid flows is[33]

$$\int_{V(t+\Delta t)} U \,\mathrm{d}V - \int_{V(t)} U \,\mathrm{d}V + \int_t^{t+\Delta t} \int_{S(t)} \boldsymbol{F}\cdot\boldsymbol{n} \,\mathrm{d}S \,\mathrm{d}t = 0 \tag{1}$$

where $\boldsymbol{F} = (\boldsymbol{u} - \boldsymbol{b})\boldsymbol{U}$ is the flux, $\boldsymbol{u}$ is the fluid velocity, $\boldsymbol{b}$ is the surface element velocity, $\boldsymbol{U}$ is any conserved quantity, $\boldsymbol{V}$ is the volume in space of the element, S is the spatial surface area of the element, t is time, and Δt is a time increment of the element. The differential form of Eq. (1) is given by

$$\frac{\partial}{\partial t}\bigg|_x U + \mathrm{div}(U\boldsymbol{u}) = 0 \tag{2}$$

which yields the Euler equations and can be rewritten as

$$\boldsymbol{U}_{,t} + \boldsymbol{F}_{i,i} = 0 \tag{3}$$

or

$$\boldsymbol{U}_{,t} + \boldsymbol{A}_i\boldsymbol{U}_{,i} = 0 \tag{4}$$

Here for a two-dimensional space

$$\boldsymbol{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ e \end{bmatrix}$$

$$e = \frac{p}{\gamma - 1} + \frac{\rho}{2}(u_1^2 + u_2^2) \tag{5}$$

$$p = (\gamma - 1)\rho i$$

$$i = c_v\Theta$$

$$\boldsymbol{A}_i = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}}$$

and

$$\boldsymbol{F}_j = u_j\boldsymbol{U} + p\begin{bmatrix} 0 \\ \delta_{j1} \\ \delta_{j2} \\ u_j \end{bmatrix} \tag{6}$$

where δ_{ij} is Kronecker delta, $\boldsymbol{F}_j$ is the flux vector in the jth direction, and the $\boldsymbol{A}_i$ matrices are referred to as the ith Euler Jacobian matrix.

Following Ref. 6, Eq. (4) can be rewritten using entropy variables as

$$\tilde{\boldsymbol{A}}_0\boldsymbol{V}_{,t} + \tilde{\boldsymbol{A}}_i\boldsymbol{V}_{,i} = 0 \tag{7}$$

where

$$\begin{aligned} \tilde{\boldsymbol{A}}_0 &= \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{V}} \\ \tilde{\boldsymbol{A}}_i &= \boldsymbol{A}_i\tilde{\boldsymbol{A}}_0 \\ \boldsymbol{A}_i &= \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}} \\ \boldsymbol{V} &= \frac{\partial H}{\partial \boldsymbol{U}} \\ H = H(\boldsymbol{U}) &= -\rho(s - s_0) \\ s &= \ell n\left(\frac{p}{\rho^\gamma}\right) + \text{const} \end{aligned} \tag{8}$$

For a detailed definition of the previous quantities see Appendix A of Ref. 6 or Ref. 34.

The finite element trial and weight functions are defined in the following spaces:

$$\mathcal{V}_n^h = \{\boldsymbol{V}^h | \boldsymbol{V}^h(\boldsymbol{x}, t) \in [C^0(S_n)]^m, \boldsymbol{q}(\boldsymbol{V}^h) = \boldsymbol{g}(t) \text{ on } \boldsymbol{P}_n\} \tag{9}$$

$$\mathcal{W}_n^h = \{\boldsymbol{W}^h | \boldsymbol{W}^h(\boldsymbol{x}, t) \in [C^0(S_n)]^m, \boldsymbol{q}'(\boldsymbol{W}^h) = 0 \text{ on } \boldsymbol{P}_n\} \tag{10}$$

where $\boldsymbol{q}$ and $\boldsymbol{q}'$ are the nonlinear boundary condition transformation functions on the domain boundaries $\boldsymbol{P}_n$ of the space-time slab $\boldsymbol{S}_n$, $\boldsymbol{g}$ is a prescribed boundary condition, m is the number of degrees of freedom (i.e., $m = 4$ for two-dimensional problems), and $\boldsymbol{V}^h$ and $\boldsymbol{W}^h$ are the finite element trial and weight functions.

To formulate an approximate solution to Eq. (3) using the GLS finite element method for a two-dimensional space, we used the following discretized weighted-residual statement for a space-time domain $\boldsymbol{S}_n$: Find $\boldsymbol{V}^h \in \mathcal{V}_n^h$ such that for all $\boldsymbol{W}^h \in \mathcal{W}_n^h$ the following integral equation is satisfied[6]:

$$\begin{aligned}
&\int_{S_n} [\boldsymbol{U}(\boldsymbol{V}^h)_{,t} + \boldsymbol{F}_{i,i}(\boldsymbol{V}^h)]\cdot \boldsymbol{W}^h \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\
&+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{S_n^e} [\tilde{\boldsymbol{A}}_0 \boldsymbol{V}^h_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{V}^h_{,i}]\cdot \boldsymbol{\tau}(\boldsymbol{V}^h)[\tilde{\boldsymbol{A}}_0 \boldsymbol{W}^h_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{W}^h_{,i}] \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\
&+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{S_n^e} \nu^h(\boldsymbol{V}^h)\hat{\nabla}_\xi \boldsymbol{W}^h \cdot \begin{bmatrix} \nwarrow & & \\ & \tilde{\boldsymbol{A}}_0 & \\ & & \searrow \end{bmatrix} \hat{\nabla}_\xi \boldsymbol{V}^h \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\
&+ \int_{\Omega(t_n)} \boldsymbol{W}^h(t_n^+)\cdot[\boldsymbol{U}(\boldsymbol{V}^h)] \,\mathrm{d}\boldsymbol{x} = 0
\end{aligned} \tag{11}$$

Here

$$[\boldsymbol{V}^h] = \boldsymbol{V}^h(t_n^+) - \boldsymbol{V}^h(t_n^-) \tag{12}$$

and the nth space-time slab $\boldsymbol{S}_n$ is subdivided into $(n_{\mathrm{el}})_n$ space-time wedge elements, S_n^e, $e = 1, \ldots, (n_{\mathrm{el}})_n$ (Fig. 2). Also,

$$\hat{\nabla}_\xi = \begin{bmatrix} \dfrac{\partial}{\partial \xi_1} \boldsymbol{I}_n \\ \dfrac{\partial}{\partial \xi_2} \boldsymbol{I}_n \\ \dfrac{\partial}{\partial \xi_0} \boldsymbol{I}_n \end{bmatrix} \tag{13}$$

where ξ_0 is the isoparametric time coordinate, and ξ_1 and ξ_2 are the isoparametric space coordinates for a linear space-time wedge finite element (see Fig. 2). Also, $\tau(\boldsymbol{V}^h)$, $\nu^h(\boldsymbol{V}^h)$, $\tilde{\boldsymbol{A}}_0$, and $\hat{\nabla}_\xi$ are defined in similar fashion as in Refs. 1 and 6. The various terms in Eq. (11) can be identified in order as the standard Galerkin term, the least-squares term, the shock-capturing term, and the jump term.

The standard Galerkin term is defined as the integral over $\boldsymbol{S}_n$ of Eq. (3) or the Euler equations multiplied by the finite element weight function $\boldsymbol{W}^h$. The jump term is a consequence of enforcing continuity between the space-time slabs in a weak sense.

The time-discontinuous GLS finite element method introduces artificial dissipation into the weighted residual problem (11) through the least-squares and discontinuity-capturing terms. The motivation for these two terms is to increase stability without substantially reducing accuracy, i.e., $\mathcal{O}(\boldsymbol{h}^{p+1/2})$ for smooth solutions with polynomials of degree p.[35] The least-squares term is defined as

$$\sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{S_n^e} [\tilde{\boldsymbol{A}}_0 \boldsymbol{W}^h_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{W}^h_{,i}]\cdot \boldsymbol{\tau}(\boldsymbol{V}^h)[\tilde{\boldsymbol{A}}_0 \boldsymbol{V}^h_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{V}^h_{,i}] \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \tag{14}$$

and gives control of the residual of the finite element solution [cf. Eq. (7)]. The discontinuity-capturing term is defined as

$$\sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{S_n^e} \nu^h(\boldsymbol{V}^h)\hat{\nabla}_\xi \boldsymbol{W}^h \cdot \begin{bmatrix} \nwarrow & & \\ & \tilde{\boldsymbol{A}}_0 & \\ & & \searrow \end{bmatrix} \hat{\nabla}_\xi \boldsymbol{V}^h \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \tag{15}$$

and gives control of the first derivatives. Including these two terms in Eq. (11) provides the necessary numerical stability to guarantee entropy consistency, error localization, and monotone shock resolution.[35] It may also provide the basis to develop a strong a posteriori error estimate useful for adaptive analyses.[35] Although the construction of both the least-squares and the discontinuity-capturing terms is straightforward, understanding and defining appropriate expressions for τ and ν^h are quite complex. References 1, 6, 23, 35, and 36 describe the theoretical details concerning the development of τ and ν^h.

The numerical implementation of the time-discontinuous GLS finite element method in two dimensions based on Eq. (11) is described in detail in Ref. 1 and is based on extending the Euler/Navier-Stokes Analysis (ENSA) code described in Ref. 6. The space-time finite element interpolation functions enable one to solve the Euler equations for a given space-time slab (see Fig. 2) through an implicit nonlinear Newton iteration scheme using, for example, a generalized minimum residual (GMRES) solver.[6,37] The present capability with the automated adaptive environment described in Sec. III is now being extended to four-dimensional space-time problems.

III. Automated Adaptive Environment

Reference 1 describes in detail the development of the automated adaptive environment (see Fig. 1) used in this effort to simulate unsteady and compressible flows around a pitching airfoil. The following is a brief description of the mesh generation, error indicator, and mesh enrichment components used within the adaptive environment.

Mesh Generation

The mesh generator used within the automated adaptive environment is finite quadtree.[27] Its basic function is to spatially decompose a problem domain into a set of nested discrete cells or quadrants that are tied together through a hierarchic tree structure. Based on this decomposition of the domain into quadrants, a finite element mesh is then created (see Fig. 3).

Initially the problem domain is placed completely within a square universe called the root quadrant. The root quadrant is then subdivided, forming four new quadrants that can be viewed as its children. The remainder of the tree is then defined by further subdivision of the quadrants according to the problem geometry and specified mesh control levels. After the initial decomposition (Figs. 3b and 3c), the quadtree is

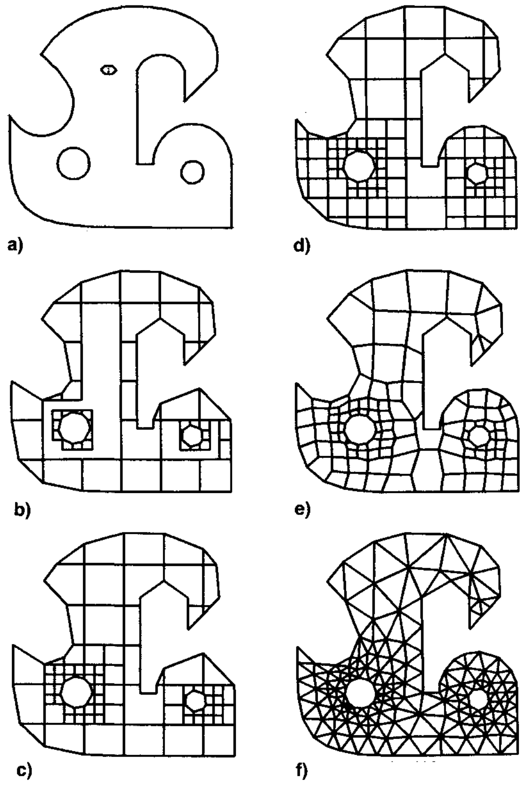

Fig. 3 Example for finite quadtree meshing: a) geometric model; b) boundary quadrants; c) boundary and interior quadrants; d) one level difference enforced; e) smoothed quadrants; and f) final mesh.

processed further to provide a better representation for mesh generation. The steps in this process insure that only one level of difference exists between each terminal quadrant and its neighbors, so that the finite elements will have relatively good aspect ratios (Fig. 3d), and collapse and smooth boundary and interior quadrants to improve the shape of the finite elements that are formed based on the individual quadrants (Fig. 3e). The decomposition procedure requires interaction between the quadtree data structure and the topological and geometric databases describing the model problem. Finite quadtree performs this communication through a set of operators.[27]

After the problem domain is discretized into a quadtree, the finite element mesh (Fig. 3f) is created on a quadrant-by-quadrant basis. Meshing templates are applied to interior quadrants, whereas an element removal procedure is required for boundary quadrants due to their complex geometric shape. After the mesh has been determined, the node points are smoothed to improve the element shapes.

The finite quadtree mesh generator is an efficient procedure that has been shown to possess computation rates of $\mathcal{O}(n \log n)$ where n is the number of quadrants. This efficiency, as well as its ability to perform local multiple level mesh updates,[38] is due to the design of the quadtree and mesh topology data structures and is the means by which these various data structures interact. Both the geometric model and the mesh topology databases are based on the radial edge data structure designs defined in Ref. 39.

Error Indication

Since solution of the Euler equations may include hyperbolic as well as elliptic regions, error estimation techniques developed for elliptic problems only are not applicable for the present problem. Consequently, error indication methods have been developed based on modified forms of classical interpolation estimates.[28,40,41] Typically for smooth problems, an interpolation estimate based on linear functions indicates the error measured in $\boldsymbol{H}^1(\Omega)$ the seminorm as the following:

$$\|e\|_{\boldsymbol{H}^1(\Omega)} \leq C \left[\sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{\Omega^e} (h_e D_h^2 u^h)^2 \, \mathrm{d}\Omega \right]^{1/2} \tag{16}$$

where $|D_h^2 u^h| = \sup_{i,j} |\partial^2 u / \partial x_i \partial x_j|$. As a consequence of using linear basis functions in the present finite element formulation, which are C^0 between elements, methods to reconstruct good approximations to $|D_h^2 u^h|$ are required. Using this type of error estimate for nonsmooth problems is not strictly valid, particularly since the nonsmooth solution does not exist in the $\boldsymbol{H}^1(\Omega)$ seminorm. This point is intentionally overlooked when using this type of error estimate as an error indicator for compressible flows with discontinuities.

For this effort, an error indicator is used that is based on modifying the interpolation estimate of Eq. (16) in the following fashion[28,42]:

$$E_I = \frac{h^2 |\text{second derivative}|}{h |\text{first derivative}| + \varepsilon |\text{mean value}|} \tag{17}$$

where E_I is the indicated error at the finite element node I. Dividing by the first derivative weakens the value of E_I for large gradient flow features like shocks, while still trying to look for the second derivative error contributions for weak flow features in areas where the first derivative is relatively small. The addition of the term $\varepsilon |\text{mean value}|$ in the denominator of Eq. (17) is a means to further tune the range of magnitude of the indicated errors detected. Specifically, ε is a small number less then 1 with a typical value being 0.12.

Implementation of Eq. (17) requires calculating second derivatives using the finite element solution derived by linear basis functions. For two- or three-dimensional problems with variable element size, Refs. 1, 28, and 42 calculate E_I through a variational method, yielding the following expression:

$$E_I = \sqrt{\frac{\sum_{k,l} \left(\int_\Omega N_{I,k} N_{J,l} \, \mathrm{d}\Omega U_J \right)^2}{\sum_{k,l} \left\{ \int_\Omega |N_{I,k}| [|N_{J,l} U_J| + \varepsilon(|N_{J,l}| |U_J|)] \, \mathrm{d}\Omega \right\}^2}} \tag{18}$$

Here U is the key variable to be used to indicate the error, N are the finite element interpolation functions, k and l are the indexes referring to the spatial coordinates, I is the finite element node where one wishes to calculate the error indicator, and J is the index of the finite element nodes that surround node I.

Equation (18) was used to calculate the indicated error for each finite element node. Both local Mach number and entropy were used as the key variable U in Eq. (18). Local Mach number was used primarily to identify large gradient flow features associated with shocks and stagnation points. Entropy was used to identify regions with high numerical entropy production as well as vortical flow regions. In addition, the Mach number scaled by multiplying by h/chord was also used to identify significant errors in the second derivative associated with small gradient flow features.

Edgewise values for the error indicators were calculated by averaging the two bounding nodal values. As part of the mesh enrichment procedure, described in the following section, the edgewise error indicator values for the key variables are used to determine if a specific edge should be collapsed, split, or left unchanged (i.e., acceptable as is). For each distinct key variable, an upper and lower indicated error tolerance is defined. Consequently, an edge should be collapsed if all edgewise values are below their lower error tolerance level, it should be split if any of the edgewise values are above their upper error tolerance level, or it should remain unchanged. In this way, the size of the mesh edges are adjusted in an attempt to make their indicated error levels fall within an acceptable range. In addition, since the error indicated along a discontinuity line (shock or slipstream) will never approach zero, regardless of how small the local element size becomes, a minimum size for h is specified, and edges are not allowed to be split if their length is below this limit.

Mesh Enrichment

The mesh enrichment strategy is based on multilevel derefinement and single level refinement procedures that are edge based and independent of mesh enrichment history. The shapes of the elements are controlled by edge swapping and Laplacian mesh smoothing. Figures 4 and 5 illustrate the major idea behind the derefinement and refinement processes, respectively.

The coarsening process (Fig. 4) proceeds by traversing the mesh edge link list and then determining if the edge under consideration can be collapsed based on the error indication procedure described in the previous section. If the edge should be collapsed, the procedure determines which bounding vertex of the edge should be the target vertex (i.e., the vertex that the edge collapses into) based on shape considerations. Once an edge is collapsed, each edge connected to the target vertex is examined. If there are two connected faces associated with the edge under consideration, then the diagonal of the polygon defined by these two faces is switched if the shape of the faces would be improved.

The coarsening process is multiple level in nature and is continued until no edges within the mesh are marked to be coarsened based on error indications. Practically, this process is accomplished by reinserting at the end of the mesh edge link list all edges that are affected by the collapsing procedure. Therefore, when the entire mesh edge link list is completely

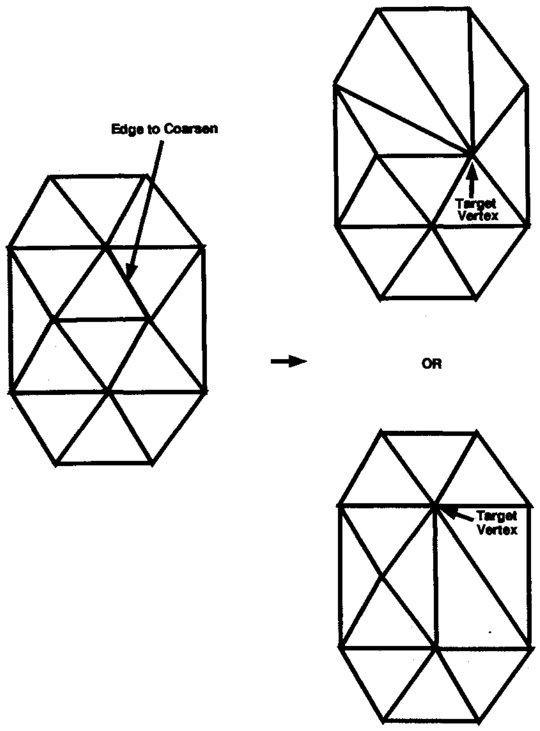

Fig. 4 Edge collapsing.

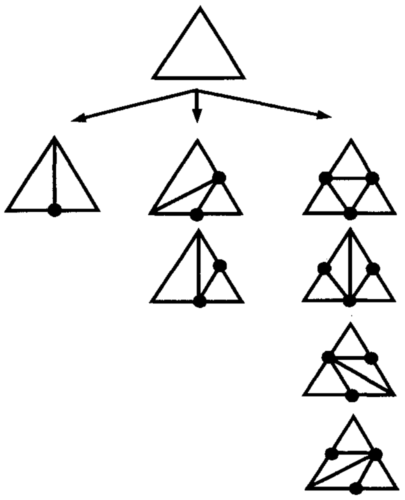

Fig. 5 Various refinement templates based on edges to be split.

traversed, no edges remain that desire to be coarsened. This type of coarsening procedure introduces no new finite element nodes. Therefore, it has been implemented with no nodal interpolation or mapping of the finite element solution variables.

The refining process follows the coarsening process and is based on bisection of edges. New vertices are introduced at the midpoint of edges that have been identified to be refined based on the error indication procedure described in the previous section. In addition, special care is taken when refining an edge that is classified on the boundary of the control volume of interest. Instead of simply bisecting the boundary edge, the new vertex is placed on the geometric boundary itself.

The algorithm proceeds by traversing the mesh face link list and determining which edges, if any, that bound the face should be split based on indicated error considerations and if the edge to be split is larger than the minimum edge length allowable. Based on this information, the face is classified as either a one, two, or three edge-split case as illustrated in Fig. 5. Then the appropriate splitting template is applied where the template with the best worst shape is selected. All newly created faces are inserted at the beginning of the mesh face link list so that by traversal of the link list each original face can be refined only once.

Interpolation errors due to refinement are minimized by restricting that any edge can be split only once. Therefore, a new vertex is placed at the midpoint of an original mesh edge. New solution variables at this point are based on using linear interpolation of vertex values of the original mesh edge. This same technique is also used when refining an edge that is classified on the boundary of the control volume of interest even though the new vertex is placed not at the midpoint of the original mesh edge but on the geometric boundary itself.

After the edge-based derefinement and refinement procedures are completed, the element shapes are improved by edge swapping and mesh smoothing (Figs. 6 and 2). The edge-swapping procedure is based on Lawson's algorithm.[43] However, instead of the maxmin criteria, the shape parameter to be maximized is

$$\text{shape} = \frac{6.9282 * A}{\sum_{i=1}^{3} l_i^2}$$

where A is the face area, l_i is the length of the ith edge, and the constant 6.9282 is a normalization factor that scales an equilateral triangle to a value of 1.0. This type of measure behaves qualitatively like shape = 2(inscribed radius/circumscribed radius). Edge swapping is applied to the mesh defined at $\Omega^h(t_n)$ (see Fig. 2) and does not move the position of any vertices. During edge swapping, nodal solution values are preserved, and any changes in the interpolation field caused by swapping is ignored. It should be pointed out that under some circumstances an edge cannot be swapped due to concavity of its bounding polygon. The bounding polygon is defined by the two adjacent triangles that share a given edge under consideration.

The mesh-smoothing procedure implemented is called Laplacian smoothing.[27,44] Laplacian smoothing moves each interior vertex to the centroid of its connecting neighbors in an iterative fashion. Specifically, for each space-time slab S_n, the vertices at the end of the slab [i.e., the triangulation at $\Omega^h(t_{n+1})$] are smoothed (Fig. 2). Smoothing only the nodes at the end of the space-time slab is advantageous because the nodal values of the solution variables for these vertices are determined as part of the solution process. Thus, interpolation of solution variables between space-time slabs is avoided. In addition, it should be stated that Laplacian smoothing should be used with caution, since it can fail under some circumstances for two-dimensional meshes, and in general it does not work well at all for three-dimensional meshes. Alternative smoothing procedures that explicitly consider the shape of the elements are described in Ref. 45.

In summary, the mesh enrichment procedure developed does not maintain mesh history information, and it is edged based. Recent mesh enrichment procedures for three dimensions can be found in Refs. 46 and 47. The procedure can

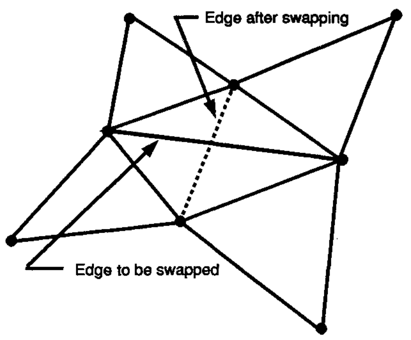

Fig. 6 Edge-swapping process.

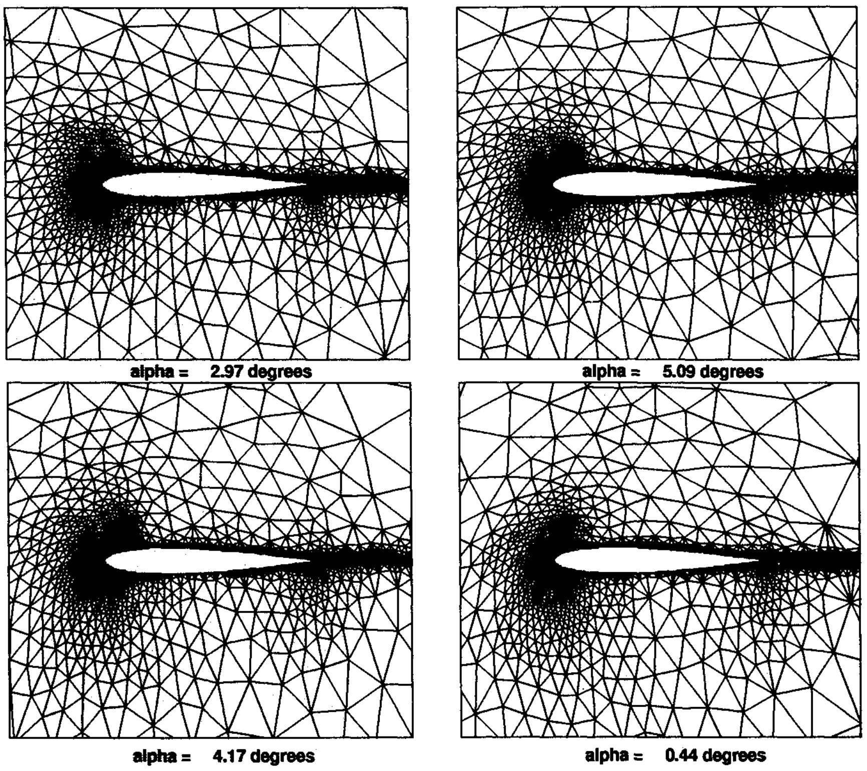

Fig. 7 Near-field meshes for $M = 0.6$ case.

derefine multiple levels, whereas it currently refines a single level each time it is executed. The coarsening process is achieved by edge collapsing and edge swapping, whereas the refinement process is based on the bisection of edges identified to be refined. Control of element shapes is provided by edge swapping and Laplacian mesh smoothing. Discretization error due to interpolation of solution variables to new vertices is not completely eliminated but minimized in a certain sense.

IV. Results

To verify the potential benefits of the adaptive time-discontinuous GLS finite element method for transonic airfoil aerodynamic calculations, we investigated a NACA 0012 airfoil oscillating about the quarter chord at $M_\infty = 0.6$ and an angle of attack defined by $\alpha = 2.89$ deg + 2.41 deg sin $k\tau$ [where $k = 0.1616$ and $\tau = (U_\infty t/c)$]. Examples of other steady and unsteady compressible rotor airfoil calculations using this automated adaptive procedure can be found in Refs. 1 and 2. The previous case is one from among several described in Ref. 48 where typical flows around rotor airfoils were experimentally investigated. It is particularly difficult to calculate due to the unsteady formation and motion of shocks along the upper surface of the airfoil as it oscillates. A steady calculation with an initial mesh of 2486 node points was run for 100 time steps, thus reducing the residual norm eight orders of magnitude for a total of 1625 CPU seconds using an IBM RISC 6000 workstation (320H). Then, using the converged steady solution with the same initial mesh, the adaptive finite element methodology was applied with the limit on the size that an edge can be split to 0.01 of a chord length. Specifically, the mesh was enriched every 30 time steps for 4 cycles of oscillatory motion. Each cycle was divided into 900 space-time slabs. The resulting time step corresponds to a CFL number of approximately 10 when compared with the size of the smallest edge in the mesh or $\Delta\tau = 0.0432201218$. This $\Delta\tau$ was selected based on a global time step that satisfies[22,23,29]

$$\Delta\tau = c \min_x h(x)$$

The constant c for this example was determined by running $\Delta\tau/2$ and seeing that the computed results were comparatively equal to $\Delta\tau$ results.

Figure 7 illustrates how the mesh (near field) changes adaptively during the fourth cycle of oscillation (the outer boundary of the control volume was 200 chords in length, which is still large enough to use freestream prescribed boundary conditions with no far-field point vortex correction). Figure 8 gives results for the pressure coefficient vs x_{airfoil}/chord for various values of $k\tau$ during the oscillatory motion of the airfoil. These results are compared with the experimental data found in Ref. 48 and are in good agreement during the entire cycle of motion. The instantaneous angle of attack labeled on the figures corresponds to the values given by the experimental data even though the numerical angle of attack is actually defined by the value of $k\tau$. The formation, motion, and disappearance of the shocks as a function of time seem to be captured particularly well. As was done in Ref. 3 for various steady transonic airfoil cases, shock jump values based on the Rankine-Hugoniot relations were computed for $k\tau$ = 90 deg where quasisteady conditions exist using the predicted values of the coefficient of pressure ahead of the shock (see Fig. 8). The good correlation in the shock jump of pressure helps to confirm that the overshoot of pressure followed by the expansion or acceleration of the flow is not a numerical artifact but an actual inviscid flow feature of the solution. Additional confirmation for such a result is given in Ref. 49 where asymptotic analytical methods were used to describe the shock wave shape at a curved surface and the flow behind it (Zierep solution). It is believed that the inclusion of viscous effects will improve the correlation between the numerical and experimental results around the shock region and near the trailing-edge region where flow separation occurs.

Figure 9 provides a time history of the total number of finite element nodes during the four cycles of motion. The variation is periodic and is found to be related to the periodicity of the intensity in the strength of the shed vortices from the trailing edge of the airfoil. This periodic change in shed vorticity strength is reflected in the size and number of elements in the trailing-edge wake region as can be seen in

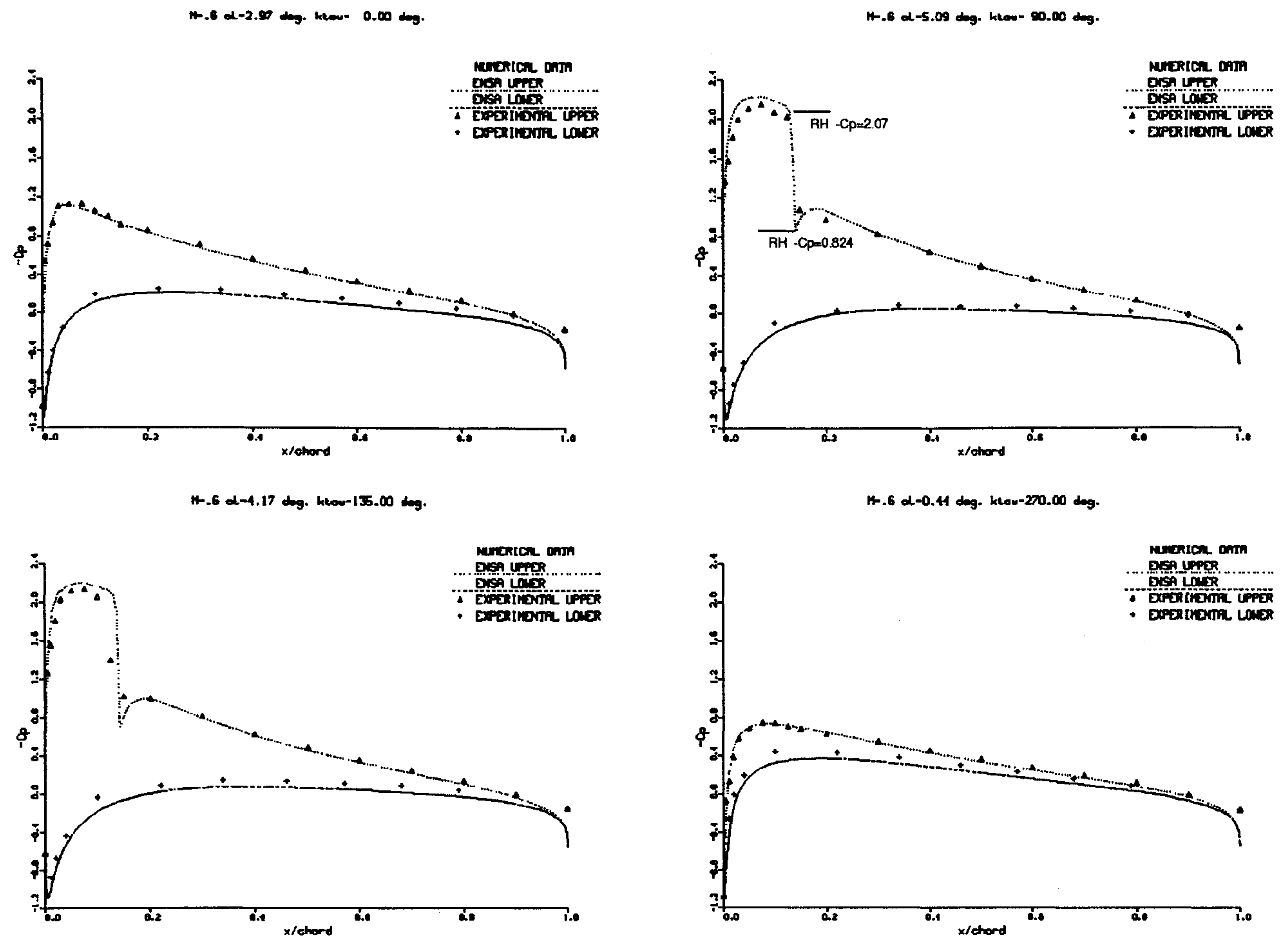

Fig. 8 c_p vs x/chord for M = 0.6 case.

Fig. 7. The decrease in node numbers near $k\tau$ = 90 deg is less than the decrease at $k\tau$ = 270 deg because that part of the cycle refinement is adding new nodes due to the presence of the shock on the upper surface. Even after four cycles of oscillations, the total number of finite element nodes is still increasing slightly. This further increase is due to new nodes still being introduced downstream from the airfoil as the numerical information from the slipstream propagates toward the outflow boundary.

Figure 9 can also be used to estimate the computational advantage of using the present adaptive CFD methodology for the unsteady transonic flow under discussion. The difference in the minimum peak values at $k\tau$ = 90 and 270 deg is approximately 600 node points. The peak value of approximately 8500 mesh points at $k\tau$ = 190 deg is predominately due to the shed vorticity from the trailing edge. Therefore, it is estimated that without adaptivity the computational mesh would need at a minimum the 600 mesh points for the shock and the 8500 mesh points for a total of 9100 mesh points for all four cycles. It is estimated that the difference in CPU time using a single mesh of 9100 mesh points vs the adaptive procedure is about 20% more. Of course, this savings is conservative because it assumes that one's a priori knowledge concerning the flow features of the problem is able to develop the 9100 node mesh. For the case under consideration, the motion of the shock and the size of the supersonic region is small. Therefore, for flows with shocks that have substantial range of motion as well as larger supersonic regions, the computational savings are much larger. References 1 and 2 also discuss the case of a NACA 0012 airfoil at M_∞ = 0.755 and angle of attack defined by α = 0.016 deg + 2.51 deg sin $k\tau$ where k = 0.1628 and $\tau = (U_\infty t/c)$, where the total CPU time required for the adaptive method was estimated to be only one-third that required for a static mesh that would yield similar quality results. Another way of stating the inherent advantage of adaptive vs nonadaptive methods is that, for a given number of nodes, the adaptive technique will distribute the points to capture the temporally evolving length scales whereas the static mesh of a nonadaptive method reflects only the length scales chosen by an analyst's a priori knowledge of the flow. Therefore, for a given number of points, an adaptive solution may be more accurate.

Figure 9 also provides a time history of the residual norm. Initially the residual norm starts lower than its final mean value of about 1.3e-4. Only 1 N iteration step was performed per space-time slab. Increasing this number would decrease the value of the residual norm but significantly increase the computational cost.

Figure 9 provides a CPU time history for this case using an IBM RISC 6000 340 workstation. In general, the increase in CPU seconds per time step for an unsteady problem compared with a steady problem is due to the fact that space-time wedge elements for unsteady problems require six nodes and six Gauss points for numerical integration, whereas the linear in space and constant in time triangular elements for steady problems require three nodes and three Gauss points. Approximately 8% of the total CPU time for this case was spent performing mesh enrichment. The CPU cost can be broken down to each of the 120 mesh enrichment steps, averaging approximately 2.5 times the average CPU time to run the flow solver for one time step. Mesh coarsening or derefinement accounts for the majority of the CPU cost for each mesh enrichment step. Specifically, what is most expensive is the edge-swapping procedure executed every time an edge is collapsed. This procedure could be significantly improved by swapping edges only in a local region near the collapsed edge.

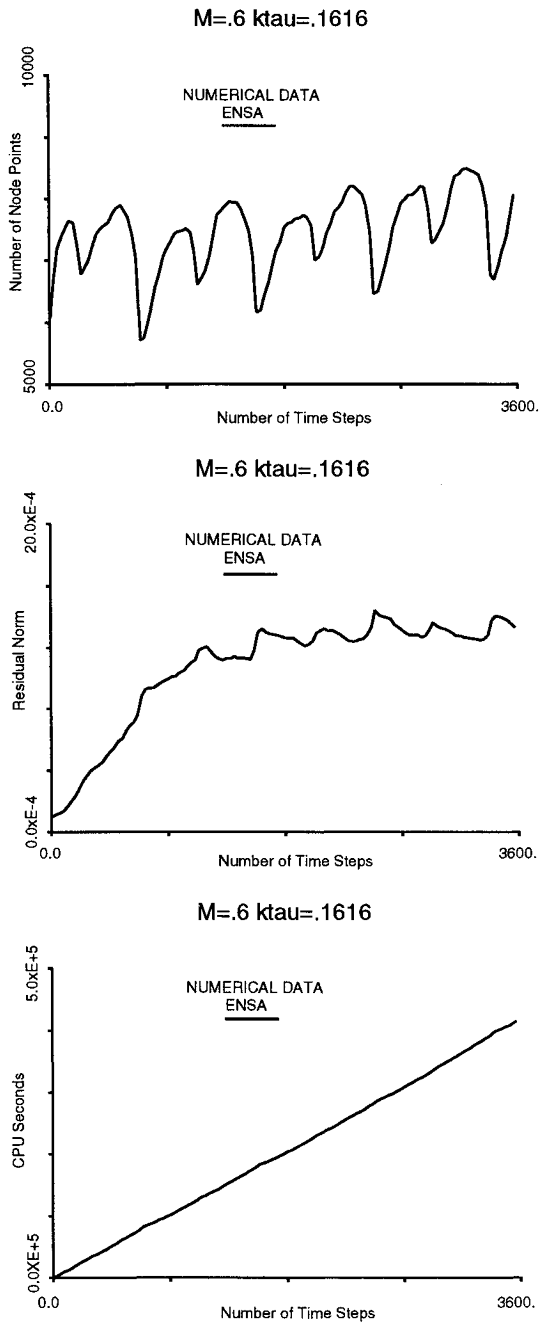

Fig. 9 Various quantities vs number of time steps.

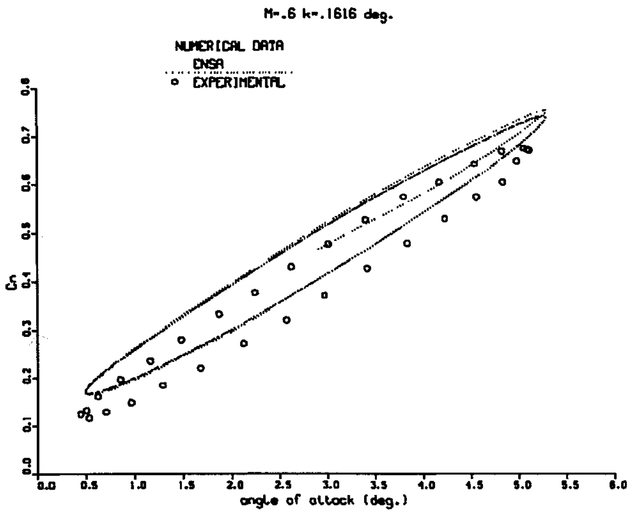

Fig. 10 c_n vs angle of attack.

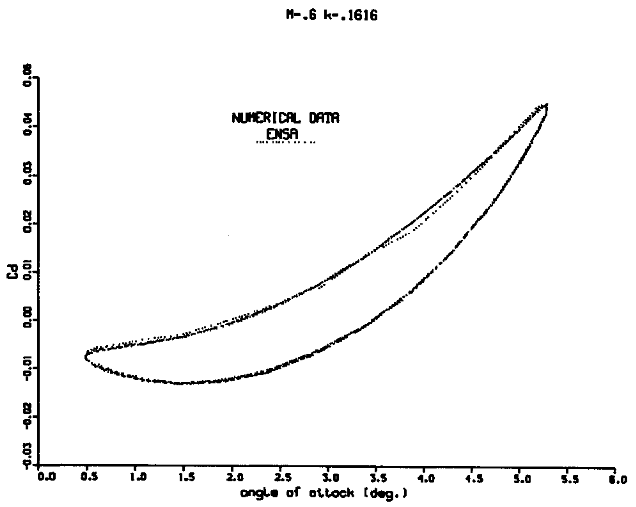

Fig. 11 c_d vs angle of attack.

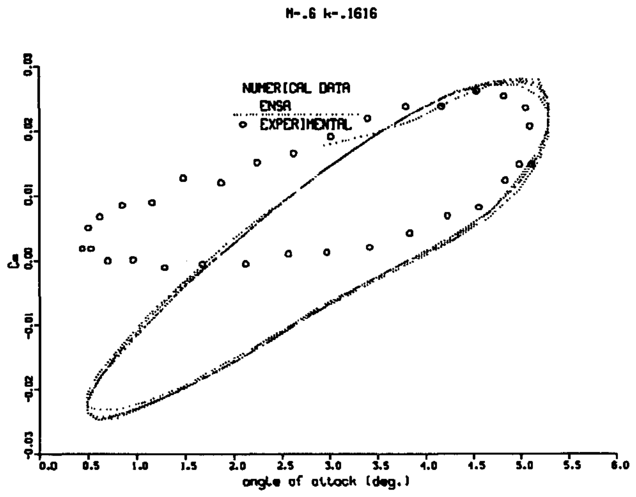

Fig. 12 c_m vs angle of attack.

Calculations indicate that with this type of change the CPU cost per mesh enrichment step can be reduced from 2.5 times to 1.0 times the average CPU time to run the flow solver for one time step. However, this kind of change only reduces the overall CPU time by 3%.

Figures 10–12 provide time histories of c_n, c_d, and c_m, respectively. All of the rotor airfoil sectional properties were calculated by numerically integrating the pressure distributions along the airfoil surface. These figures of rotor airfoil sectional properties confirm that after four cycles the flow simulation seems to be nearly periodic. The majority of the discrepancies between the numerical values and the experimental values for c_n, c_d, and c_m can be accounted for by careful examination of Fig. 8. Specifically, the numerical value of c_p along the lower surface of the rotor airfoil is systematically about 0.05 higher then the experimental value. Therefore, integrating this amount over the entire chord may produce an increase in c_n of about 0.05 seen in Fig. 10. In addition, when $0.44 < \alpha < 3.0$ deg or $180 < k\tau < 360$ deg, this small difference in c_p produces a larger percentage of the total integrated area for c_n. This situation is exacerbated between 50–100% chord, producing a relatively large nose down moment for this part of the cycle as seen in Fig. 12. This type of correlation difference between the numerical and experimental results could possibly be due to the fact that the experimental instantaneous angle of attack does not seem to correspond exactly with the given formula $\alpha = 2.89 + 2.41 \sin k\tau$ deg.

Figure 11 does illustrate an increase in c_d due to the wave drag associated with the shock on the upper surface being present when $60 < k\tau < 160$ deg. However, a small negative drag is predicted when $160 < k\tau < 270$ deg, which is probably not realistic at all. It should be noted that the numerical values for c_n, c_d, and c_m were obtained based on integrating along the rotor airfoil surface. This type of calculation is known to be inaccurate specifically for c_d. Reference 50 illustrates how this type of calculation can be corrected using integral momentum considerations. Another possible explanation is that the c_m calculation is sensitive to the position about which the rotor airfoil oscillates, and since the physical chord was only 4 in., a mere 0.08-in. shift would make the center of oscillation at the 27.3% chord position instead of 25.0%. Reference 13 illustrates how such a small change may substantially improve the c_m time history.

Finally, it should be mentioned that correlation with experimental values is only one way to validate the numerical results for the previous case. Repeating the adaptive procedure with decreasing values for the upper indicated error tolerance, minimum values for smallest mesh edge length, and the lower indicated error tolerance set to zero (i.e., no coarsening) would show at what point the numerical results are converged with respect to mesh attributes. In consideration of the substantial amount of computing resources required to execute the previous exercise, using mesh convergence results for a steady transonic airfoil calculation[1,2] to help set the upper and lower indicated error levels and then comparing the results with experimental data were considered acceptable. In fact, the upper indicated error tolerance using entropy as the key variable was set quite low based on the criteria that for the steady transonic calculation this tolerance level was necessary to ensure that numerical entropy is nearly eliminated. Using such a low error tolerance for entropy causes the meshes in Fig. 7 to be quite refined along the whole surface of the airfoil.

V. Closing Remarks

The adaptive compressible airfoil results described in the previous section substantiate that the time-discontinuous GLS finite element method holds promise to be able to accurately calculate unsteady compressible flow characteristics, specifically when multiple relative body motion is important. This is a crucial requirement for any comprehensive treatment of the aerodynamic flowfield around an aeroelastic rotor-body combination of a helicopter in flight as was stated in recent review papers on the state of the art of CFD.[51,52] It is planned to extend our application of the time-discontinuous GLS finite element method to three-dimensional unsteady elastic wing problems and ultimately to the treatment of an aeroelastic rotor-body combination of a helicopter in flight.

Acknowledgments

The first two authors would like to acknowledge the Army Research Office (DAAL03-88-C-0004, G. Anderson, project monitor) for funding this research through the ARO Rotorcraft Technology Center at Rensselaer Polytechnic Institute. In addition, the authors would like to thank T. J. R. Hughes for access to the ENSA space/time finite element code for solving Euler/Navier-Stokes problems developed by his group at Stanford University and Roger Strawn at the Aeroflightdynamics Directorate at NASA Ames Research Center for his continued interest in this research and for providing computer time on a Cray Y-MP.

References

[1]Webster, B. E., "Adaptive Finite Element Method for Unsteady Compressible Rotor Airfoil Aerodynamics," Ph.D. Thesis, Rensselaer Polytechnic Inst., Mechanical Engineering, Aeronautical Engineering, and Mechanics, Troy, NY, May 1993.

[2]Webster, B. E., Shephard, M. S., and Rusak, Z., "Unsteady Compressible Rotor Airfoil Aerodynamics Using an Automated Adaptive Finite Element Method," *AHS 49th Forum Proceedings*, St. Louis, MO, May 19–21, 1993, pp. 631–648.

[3]Yoshihara, H., "Test Cases for Inviscid Flow Field Methods," AGARD-AR-211, 1985.

[4]Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1388.

[5]Barth, T., "Aspects of Unstructured Grids and Finite Volume Solvers for the Euler and Navier-Stokes Equations," AGARD Rept. 787 Ref. 6, Neuilly Sur Seine, France, May 1992.

[6]Shakib, F., "Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations," Ph.D. Thesis, Dept. of Mechanical Engineering, Stanford Univ., Stanford, CA, Nov. 1988.

[7]Peiro, J., "A Finite Element Procedure for the Solution of the Euler Equations on Unstructured Meshes," Ph.D. Thesis, Dept. of Civil Engineering, Univ. College of Swansea, Wales, UK, Sept. 1989.

[8]Luo, H., Baum, J. D., Loehner, R., and Cabello, J., "Adaptive Edge-Based Finite Element Schemes for the Euler and Navier-Stokes Equations on Unstructed Grids," Science Applications International Corp., Technical Rept., McLean, VA, 1993; also AIAA Paper 93-0336, Jan. 1993.

[9]Jameson, A., and Venkatakrishnan, V., "Transonic Flows about Oscillating Airfoils Using the Euler Equations," Princeton Univ., Technical Rept., Princeton, NJ, 1985; also AIAA Paper 85-40949, July 1985.

[10]Smith, G. E., Whitlow, W., and Hassan, H. A., "Unsteady Transonic Flows Past Airfoils Using the Euler Equations," North Carolina State Univ., Technical Rept., Raleigh, NC, 1986; also AIAA Paper 86-1764, 1986.

[11]Venkatakrishnan, V., "Computation of Unsteady Transonic Flows Over Moving Airfoils," Ph.D. Thesis, Princeton Univ., Princeton, NJ, Jan. 1987.

[12]Farhat, C., and Lin, T. Y., "Transient Aerolastic Computations Using Multiple Moving Frames of Reference," Univ. of Colorado at Boulder, Dept. of Aerospace Engineering Studies, Technical Rept., Boulder, CO, 1990; also AIAA Paper 90-3053-CP, 1990.

[13]Gaitonde, A. L., and Fiddes, S. P., "A Moving Mesh System for the Calculation of Unsteady Flows," Univ. of Bristol, Technical Rept., Bristol, England, UK, 1993; also AIAA Paper 93-0641, 1993.

[14]Loehner, R., and Baum, J. D., "Three-Dimensional Store Separation Using a Finite Element Solver and Adaptive Remeshing," George Washington Univ., CMEE, SEAS, Technical Rept., Washington, DC, 1991; also AIAA Paper 91-0602, Jan. 1991.

[15]Davis, G. A., and Bendiksen, O. O., "Unsteady Transonic Euler Solutions Using Finite Elements," Univ. of California, Technical Rept., 1992; also AIAA Paper 92-2504-CP, 1992.

[16]Probert, E. J., Hassan, O., and Morgan, K., "An Adaptive Finite Element Method for Transient Compressible Flows with Moving Boundaries," *International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 751–765.

[17]Tezduyar, T. E., Liou, J., and Behr, M., "A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces—The DSD/ST Procedure I," Army High Performance Computing Research Center, Dept. of Aerospace Engineering and Mechanics and Minnesota Supercomputer Inst., Univ. of Minnesota, Minneapolis, MN, Preprint 90-34, Aug. 1990.

[18]Tezduyar, T. E., Liou, J., and Behr, M., "A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces—The DSD/ST Procedure II," Army High Performance Computing Research Center, Dept. of Aerospace Engineering and Mechanics and Minnesota Supercomputer Inst., Univ. of Minnesota, Minneapolis, MN, Preprint 90-35, Nov. 1990.

[19]Mittal, S., and Tezduyar, T. E., "Space-Time Finite Element Computation of Incompressible Flows with Emphasis on Flows Involving Oscillating Cylinders," Army High Performance Computing Research Center, Dept. of Aerospace Engineering and Mechanics and Minnesota Supercomputer Inst., Univ. of Minnesota, Minneapolis, MN, Preprint 91-41, July 1991.

[20]Mittal, S., and Tezduyar, T. E., "Finite Element Study of Incompressible Flows Past Oscillating Cylinders and Airfoils," Army High Performance Computing Research Center, Dept. of Aerospace Engineering and Mechanics and Minnesota Supercomputer Inst., Univ. of Minnesota, Minneapolis, MN, Preprint 91-120, Dec. 1991.

[21]Aliabadi, S. K., and Tezduyar, T. E., "Space-Time Finite Element Computation of Compressible Flows Involving Moving Boundaries and Interfaces," Army High Performance Computing Research Center, Dept. of Aerospace Engineering and Mechanics and Minnesota Supercomputer Inst., Univ. of Minnesota, Minneapolis, MN, Preprint 92-043, April 1992.

[22]Rausch, R. D., Batina, J. T., and Yang, H. T. Y., "Spatial Adaptation of Unstructured Meshes for Unsteady Aerodynamic Flow Computations," *AIAA Journal*, Vol. 30, No. 5, 1992, pp. 1243–1251.

[23]Hansbo, P., "Adaptivity and Streamline Diffusion Procedures in the Finite Element Method," Ph.D. Thesis, Dept. of Structural Mechanics, Chalmers Univ. of Technology, Goeteborg, Sweden, April 1989.

[24]Rausch, R. D., Batina, J. T., and Yang, H. T. Y., "Spatial Adaptation Procedures on Tetrahedral Meshes for Unsteady Aerodynamic Flow Calculations," Purdue Univ., Technical Rept., West Lafayette, IN 1993; also AIAA Paper 93-0670, 1993.

[25]Johnson, C., *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge Univ. Press, Cambridge, England, UK, 1987.

[26]Hughes, T. J. R., Franca, L. P., and Hulbert, G. M., "A New Finite Element Formulation for Computational Fluid Dynamics: Viii. The Galerkin/Least-Squares Method for Advective-Diffusive Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 73, 1989, pp. 173–189.

[27]Baehmann, P. L., Wittchen, S. L., Shephard, M. S., Grice, K. R., and Yerry, M. A., "Robust Geometrically Based, Automatic Two-Dimensional Mesh Generation," *International Journal for Numerical Methods in Engineering*, Vol. 24, No. 6, 1987, pp. 1043–1078.

[28]Loehner, R., "Finite Element Methods in CFD: Grid Generation, Adaptivity, and Parallelization," AGARD Rept. 787 Ref. 8, Neuilly Sur Seine, France, May 1992.

[29]Hansbo, P., and Johnson, C., "Adaptive Streamline Diffusion Methods for Compressible Flow Using Conservation Variables," *Proceedings from the Symposium on Large Scale Computations in Fluid Dynamics*, Minnesota Super Computer Inst., April 1990.

[30]Hughes, T. J. R., "Recent Progress in the Development and Understanding of SUPG Methods with Special Reference to the Compressible Euler and Navier-Stokes Equations," *International Journal for Numerical Methods in Fluids*, Vol. 7, 1987, pp. 1261–1275.

[31]Johnson, C., and Szepessy, A., "A Shock-Capturing Streamline Diffusion Finite Element Method for a Non-Linear Hyperbolic Conservation Law," Chalmers Univ. of Technology, Preprint 86-09/ISSN 0347-2809, Goeteborg, Sweden, 1986.

[32]Hansbo, P., "The Characteristic Streamline Diffusion Method for the Time-Dependent Incompressible Navier-Stokes Equations," Univ. of Goeteborg, Dept. of Computer Sciences, Preprint 91-14/ ISSN 0347-2809, Goeteborg, Sweden, 1991.

[33]Vinokur, M., "An Analysis of Finite-Difference and Finite-Volume Formulations of Conservations Laws," *Journal of Computational Physics*, Vol. 81, 1989, pp. 1–52.

[34]Chalot, F., Hughes, T. J. R., and Shakib, F., "Symmetrization of Conservation Laws with Entropy for High-Temperature Hypersonic Computations," *Computing Systems in Engineering*, Vol. 1, Nos. 2–4, 1990, pp. 495–521.

[35]Johnson, C., "Finite Element Methods for Flow Problems," AGARD Rept. 787, Ref. 1, Neuilly Sur Seine, France, 1992.

[36]Mallet, M., "A Finite Element Method for Computational Fluid Dynamics," Ph.D. Thesis, Dept. of Civil Engineering, Stanford Univ., Stanford, CA, Nov. 1986.

[37]Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," Yale Univ., Dept. of Computer Science, Research Rept. YALE/DCS/RR-254, New Haven, CT, 1983.

[38]Baehmann, P. L., and Shephard, M. S., "Adaptive Multiple Level h-Refinement in Automated Finite Element Analyses," *Engineering with Computers*, Vol. 5, No. 3/4, 1989, pp. 235–247.

[39]Weiler, K. J., "The Radial-Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Representations," *Geometric Modeling for CAD Applications*, edited by M. J. Wozny, H. W. McLaughlin, and J. L. Encarnacao, North Holland, Amsterdam, The Netherlands, 1988, pp. 3–36.

[40]Morgan, K., Peraire, J., and Peiro, J., "Unstructured Grid Methods for Compressible Flows," AGARD Rept. 787, Ref. 5, Neuilly Sur Seine, France, May 1992.

[41]Warren, G., Anderson, W., Thomas, J., and Krist, S., "Grid Convergence for Adaptive Methods," NASA Langley Research Center, Technical Rept., Hampton, VA, 1991; also AIAA Paper 91-1592-CP, 1991.

[42]Loehner, R., "An Adaptive Finite Element Scheme for Transient Problems in CFD," *Computer Methods in Applied Mechanics and Engineering*, Vol. 61, 1987, pp. 323–338.

[43]Lawson, C. L., "Properties of n-Dimensional Triangulations," *Computer Aided Geometric Design*, Vol. 3, 1986, pp. 231–246.

[44]Frey, W. H., "Mesh Relaxation: A New Technique for Improving Triangulations," *International Journal for Numerical Methods in Engineering*, Vol. 31, 1991, pp. 1121–1133.

[45]de Cougny, H. L., Shephard, M. S., and Georges, M. K., "Explicit Node Point Smoothing within the Finite Octree Mesh Generator," Scientific Computation Research Center, Rensselaer Polytechnic Inst., Technical Rept. SCOREC 10-1990, Troy, NY, 1990.

[46]Biswas, R., and Strawn, R., "A New Procedure for Dynamic Adaptation of Three-Dimensional Unstructured Grids," AIAA Paper 93-0672, Reno, NV, Jan. 1993.

[47]de Cougny, H. L., "Refinement/Derefinement of 3-D Geometric Triangulations," Army Research Office Site Visit for Rensselaer Rotorcraft Technology Center, Scientific Computation Research Center, May 1993.

[48]Landon, R. H., "Naca 0012 Oscillatory and Transient Pitching," Data Set 3 in AGARD-R-702, *Compendium of Unsteady Aerodynamic Measurements*, Technical Rept., Aug. 1982.

[49]Cole, J. D., and Cook, L. P., *Transonic Aerodynamics*, Elsevier, Amsterdam, The Netherlands, 1986.

[50]Varma, R. R., and Caughey, D. A., "Evaluation of Navier-Stokes Solutions Using the Integrated Effect of Numerical Dissipation," Cornell Univ., Sibley School of Mechanical and Aerospace Engineering, Technical Rept., Ithaca, NY, 1993; also AIAA Paper 93-0539, Jan. 1993.

[51]Singleton, R. E., "CFD Joins the Army," *Aerospace America*, Feb. 1992.

[52]Steger, J. L., and Hafez, M. M., "CFD Goes to School," *Aerospace America*, Jan. 1992.