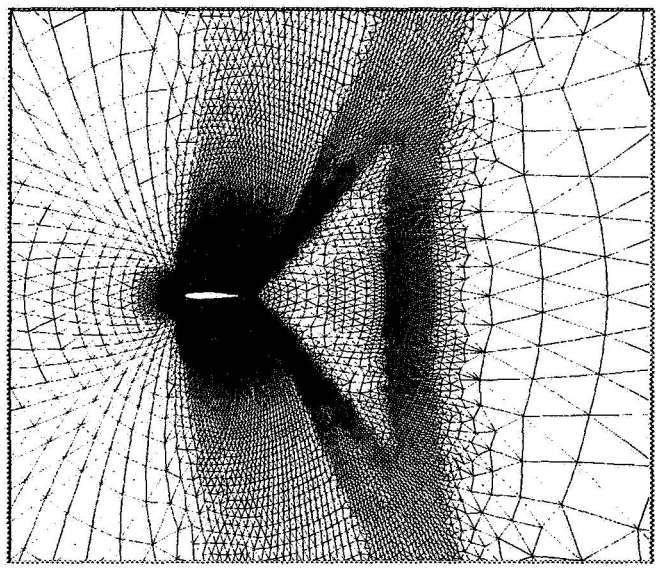
NASA Conference Publication 3316

ICASE/LaRC Workshop on Adaptive Grid Methods

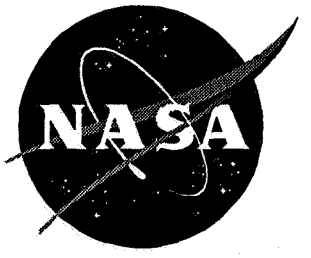
Edited by
Jerry C. South, Jr., James L. Thomas, and John Van Rosendale



Proceedings of a workshop sponsored by the National Aeronautics and Space Administration, Washington, D.C., and the Institute for Computer Applications in Science and Engineering (ICASE), Hampton, Virginia, and held in Hampton, Virginia
November 7–9, 1994

October 1995

NASA Conference Publication 3316



ICASE/LaRC Workshop on Adaptive Grid Methods

Edited by
Jerry C. South, Jr. and James L. Thomas
Langley Research Center • Hampton, Virginia

John Van Rosendale
Institute for Computer Applications in Science and Engineering (ICASE) • Hampton, Virginia

Proceedings of a workshop sponsored by the National Aeronautics and Space Administration, Washington, D.C., and the Institute for Computer Applications in Science and Engineering (ICASE), Hampton, Virginia, and held in Hampton, Virginia November 7–9, 1994

National Aeronautics and Space Administration
Langley Research Center • Hampton, Virginia 23681-0001

October 1995

Cover photograph:

Unstructured adaptive grid for transonic flow over a NACA 0012 airfoil (figure 9 from *Multigrid Solution Strategies for Adaptive Meshing Problems* by Dimitri J. Mavriplis).

This publication is available from the following sources:

NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

PREFACE

Solution-adaptive grid techniques are essential to the attainment of practical, user-friendly, computational fluid dynamics (CFD) applications. In this 3-day workshop, experts gathered together to describe state-of-the-art methods in solution-adaptive grid refinement, analysis, and implementation; assess the current practice; and discuss future needs and directions for research. This was accomplished through a series of invited and contributed papers. The workshop focused on a set of two-dimensional test cases designed by the organizers to aid in assessing the current state of development of adaptive-grid technology. These test cases are listed and described in the following section. In addition, a panel of experts from universities, industry, and government research laboratories discussed their views of needs and future directions in this field.

The invited and contributed papers, as well as the transcript of the panel discussion, are included herein. In this preface, several observations regarding the general results of the workshop follow, condensed mainly from the panel discussion. One general observation is that not many of the "benchmark" cases were attempted by all the participants, so it is difficult to rate the efficiency among the many current approaches.

The second general observation is that the state of the art is characterized by approaches that refine meshes only in high-gradient regions, while ignoring some very important regions of the flow which are smoother but nonetheless crucial to obtaining the correct solution. The transonic airfoil case with the "fishtail" shock, wherein the location of the normal shock in the wake is extremely sensitive to the smooth supersonic flow over the airfoil ahead of the oblique shock at the trailing edge, illustrates this difficulty.

The third general observation is that there is still a dearth of research regarding the analysis of accuracy and convergence of adaptive methods, especially for problems with embedded hyperbolic regions of flow. The vast majority of current research and demonstrations must resort to comparing results with those of a globally-refined numerical solution (the "old fashioned way"), which still seems to be the only trustworthy method for assessing numerical convergence in practical nonlinear boundary-value problems.

Finally, there was considerable discussion among the panelists and participants on the need for a "black box" CFD code in the future, meaning that the design engineer would be able to use a CFD code as a research tool without requiring intimate knowledge of the workings of the code itself. Certainly, solution-adaptive grids will be a key element in such a code of the future. One imagines a code which performs a preliminary grid generation based on the geometry and flow parameters input to the code, followed by adaptive refinement of the grid without intervention of the designer. If such "black box" CFD codes become a

reality, they should provide an estimate for the engineer as to how much numerical error remains in the result.

There was a feeling that a follow-on workshop would be highly profitable to the research community. A number of the algorithms presented at this workshop have the capability to treat three-dimensional flows. Thus, it is expected that another workshop will be organized in the near future, including one or more three-dimensional test cases.

Our thanks go to Ms. Emily Todd for managing the workshop and to Ms. Lori Rowland and Ms. Lisa Kitchen for transcribing the panel discussion.

Jerry C. South, Jr., NASA Langley Research Center

James L. Thomas, NASA Langley Research Center

John Van Rosendale, ICASE

CONTENTS

Adaptive Grid Workshop Airfoil Cases

1.0 AGARD 01

1.1 Geometry

The airfoil geometry for this case is that of a NACA 0012 airfoil with a closed trailing edge. Coordinates and cubic spline coefficients will be given. Because of the solution sensitivity of the outer boundary location, either of the following outer boundary definitions is recommended.

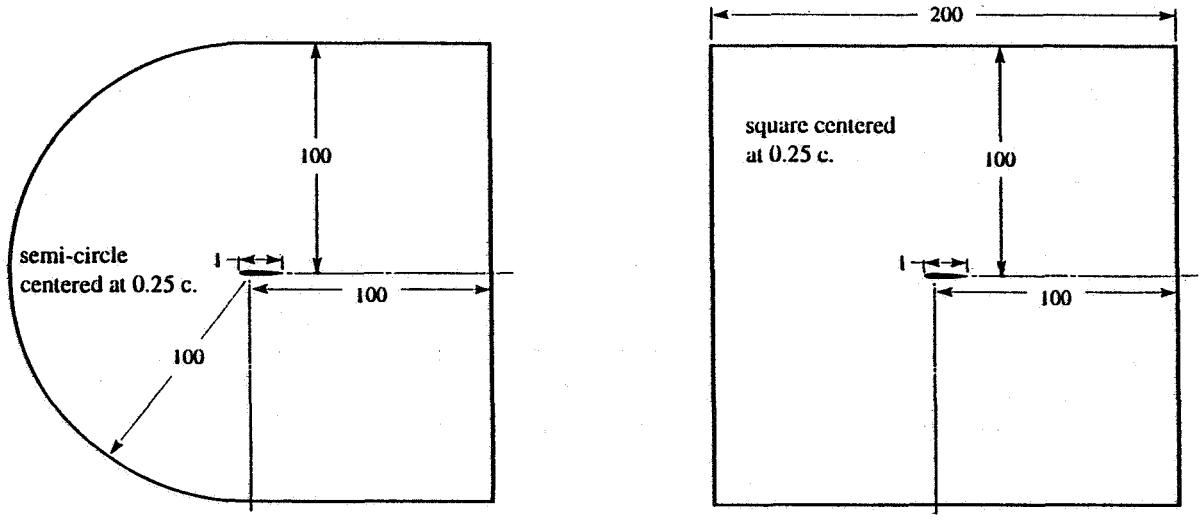


Figure 1. Outer boundary definitions for NACA 0012 airfoil grid.

1.2 Flow conditions

The flow conditions for the AGARD 01 test case are $M_\infty = 0.8$ and $\alpha = 1.25^o$. The fluid is assumed to be a perfect gas.

1.3 Output

The AGARD 01 test case has an upper and lower surface shock. The locations of these two shocks, along with the shape of the sonic line, will be compared between solutions.

2.0 AGARD 03

2.1 Geometry

The airfoil geometry for this case is that of a NACA 0012 airfoil with a closed trailing edge. Coordinates and cubic spline coefficients will be given. Because of the solution sensitivity of the outer boundary, the adaptive methods should account for this sensitivity. Grid convergence studies will be conducted for a series of outer boundary diameters to eliminate the outer boundary effect. If the adaptive calculation does not eliminate the outer boundary effects, then one of the outer boundary definitions from case 1 should be used. One way to eliminate the outer boundary effect would be to employ an adaptively movable outer boundary.

2.2 Flow conditions

The flow conditions for the AGARD 03 test case are $M_\infty = 0.95$ and $\alpha = 0^0$. The fluid is assumed to be a perfect gas.

2.3 Output

The AGARD 03 test case has a fish-tail shock structure that is shown below. The distance X_s from the trailing edge of the airfoil to the normal shock in the wake will be measured. Additionally, the shape of the sonic line will be compared between solutions.

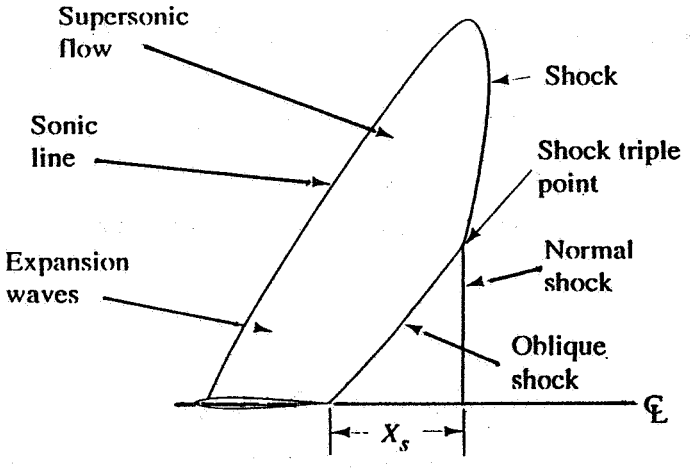


Figure 2. Shock structure for AGARD 03.

3.0 Suddhoo and Hall four-element airfoil

3.1 Geometry

The geometry for the four-element airfoil case of Suddhoo and Hall is obtained by the applying the Karman-Trefftz mapping function. For this workshop, coordinates and spline coefficients for each of the elements will be given. Additionally, a coarse, block-structured grid which consists of 14 blocks, and an unstructured triangular grid are available upon request.

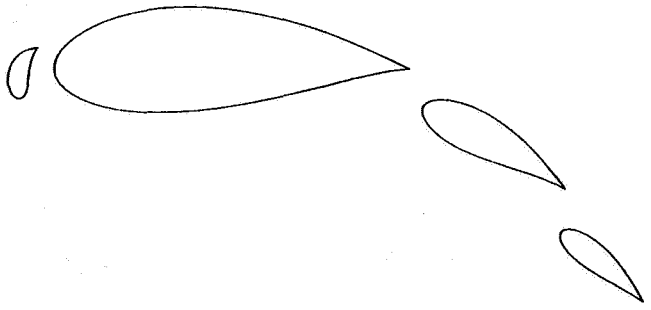


Figure 3. Suddhoo and Hall four-element airfoil configuration.

3.2 Flow conditions

The flow conditions for the Suddhoo and Hall four-element airfoil are $M_\infty = 0.2$ and $\alpha = 0^o$. The fluid is assumed to be a perfect gas.

3.3 Output

The calculated coefficient of pressure for each of the elements will be required.

4.0 Douglas three-element airfoil (viscous)

4.1 Geometry

The geometry for this case is that of a Douglas three-element airfoil in a wind tunnel. Coordinates and spline coefficients for each of the elements will be given at a later date.

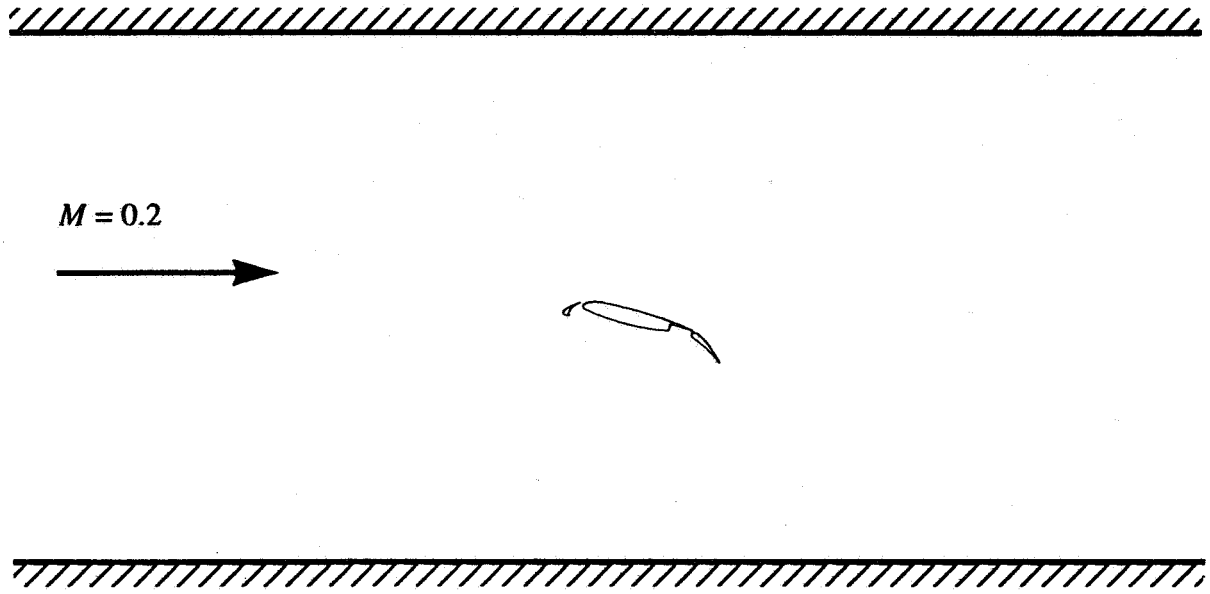


Figure 4. Configuration of three-element airfoil in wind tunnel.

4.2 Flow Conditions

The flow conditions for this case are $M = 0.2$, $\alpha = 16^{o}$, $Re = 9 \times 10^{6}$. The fluid is assumed to be a perfect gas. Inflow and outflow boundary conditions are specified at the left and right of the domain, and inviscid, solid-wall boundary conditions are specified at the top and bottom of the domain.

4.3 Turbulence Model

The calculations for this case should be done with the assumption that the boundary layer is always turbulent on the airfoil; the flow is assumed to be inviscid on the tunnel walls. The turbulence model used should be the Spalart-Allmaras turbulence model as described in AIAA paper 92-0439, "A One-Equation Turbulence Model for Aerodynamic Flows."

4.4 Output

The calculated coefficient of pressure for each of the elements will be required.

5.0 Jameson airfoil with nonunique solution

5.1 Geometry

The airfoil geometry for this case is described in AIAA paper 91-1625, "Airfoils Admitting Nonunique Solution of the Euler Equations" by A. Jameson. Coordinates and spline coefficients will be supplied at a later date.

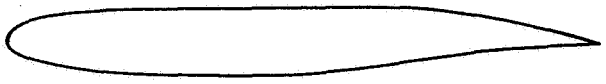


Figure 5. Jameson airfoil geometry.

5.2 Flow Conditions

This case should be calculated with an inviscid solver. As noted in Jameson's report, this airfoil geometry exhibits nonunique solutions for a given angle of attack. Nonunique solutions occur within a specified angle-of-attack range when the new solution is started from the previous solution while the angle of attack is first increased and then decreased. Two types of calculations are encouraged.

1. Steady-state spatially adaptive solutions that exhibit the nonuniqueness described by Jameson.

2. Unsteady, spatially adaptive solutions in the limit as the reduced frequency goes to 0.

The angle of attack range is from -1.2° to -0.8°.

5.3 Output

Plots of lift versus angle of attack will be required.

6.0 Shock reflection from a double wedge (time dependent)

6.1 Geometry

Given the relative dimensions shown in the schematic, the geometry is fixed by the wedge angles α and β, which are set to 20^0 and 55^0, respectively.

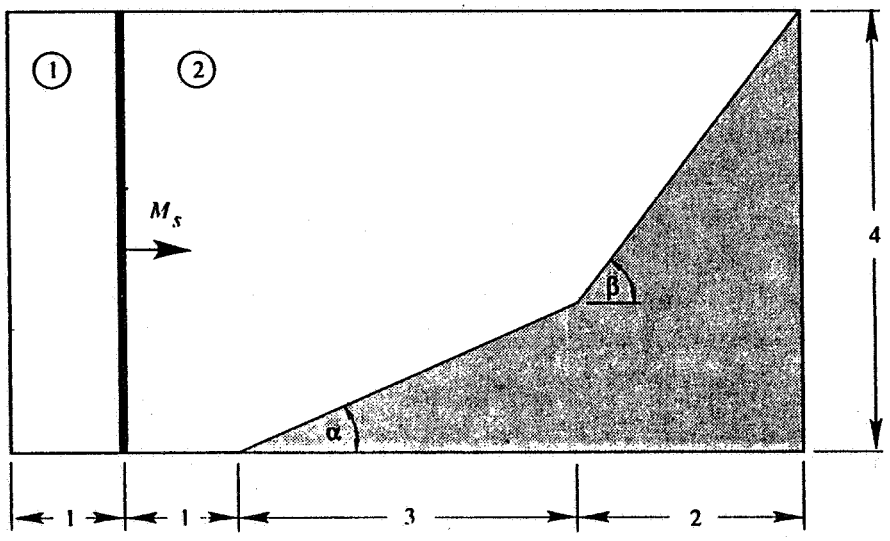


Figure 6. Double wedge geometry.

6.2 Initial conditions

A shock is prescribed at the location shown in the schematic. The fluid is assumed to be a perfect gas for which the ratio of specific heats is taken to be 1.4. The strength of the shock is determined from the shock Mach number M_s (the ratio of the shock speed to the sound speed of the quiescent fluid), which should be taken as 2.16. To provide a common frame of reference, the density and the pressure of the quiescent fluid (state 2) should both be set to 1. Given these initial conditions, the simulation is performed by integrating the Euler equations forward in time.

6.3 Output

Details in regard to required output quantities will be given at a later date.

hp-ADAPTIVITY AND ERROR ESTIMATION FOR HYPERBOLIC CONSERVATION LAWS

Kim S. Bey
NASA Langley Research Center
Hampton, VA

SUMMARY

This paper presents an hp-adaptive discontinuous Galerkin method for linear hyperbolic conservation laws. *A priori* and *a posteriori* error estimates are derived in mesh-dependent norms which reflect the dependence of the approximate solution on the element size (h) and the degree (p) of the local polynomial approximation. The *a posteriori* error estimate, based on the element residual method, provides bounds on the actual global error in the approximate solution. The adaptive strategy is designed to deliver an approximate solution with the specified level of error in three steps. The *a posteriori* estimate is used to assess the accuracy of a given approximate solution and the *a priori* estimate is used to predict the mesh refinements and polynomial enrichment needed to deliver the desired solution. Numerical examples demonstrate the reliability of the *a posteriori* error estimates and the effectiveness of the hp-adaptive strategy.

INTRODUCTION

Adaptive methods, based on successive refinement of an existing mesh or a complete re-meshing of the computational domain, have become invaluable tools in computational fluid dynamics. The amount of refinement or the clustering of grid points is often determined by an element refinement indicator, θ_K, of the form

$$\theta_K = h^\alpha |D^n u| \tag{1}$$

where h is a measure of the element size, α is an exponent, and $D^n u$ represents some higher-order derivative of a key variable. Since these indicators are based on interpolation or truncation error estimates, they are applicable to a large class of problems and are independent of the numerical method used to obtain an approximate solution. Although these indicators detect certain flow features, they may not relate to the actual error in the solution. Often, these indicators only provide a relative measure of the error and do not provide a criteria for stopping the adaptive process.

This paper summarizes some of the work presented in [4] aimed at developing hp-adaptive strategies based on reliable error estimates for hyperbolic conservation laws. The work focuses on

the discontinuous Galerkin method applied to a model class of linear hyperbolic conservation laws for which it is possible to develop mathematically rigorous *a priori* and *a posteriori* error estimates.

The notion of discontinuous Galerkin methods for hyperbolic problems originated in the classical work of Lesaint and Raviart [1] over two decades ago. Johnson and Pitkaranta [2] generalized the theory of discontinuous Galerkin methods by introducing mesh-dependent norms and were able to derive *a priori* error estimates in such norms for linear hyperbolic problems. Discontinuous Galerkin methods were extended to nonlinear hyperbolic conservation laws by Cockburn, Hou, and Shu [3] who developed a local projection strategy to provide nonlinear stability.

The local nature of the discontinuous Galerkin method makes it ideally suited for adaptive strategies which combine local mesh refinement (h) with local enrichment of the polynomial approximation (p) in an element to improve solution accuracy. The theory of discontinuous Galerkin methods was extended to *hp*-finite element approximations by Bey in [4] for a class of linear hyperbolic conservation laws. In [4], very high accuracies and convergence rates were observed in applying discontinuous Galerkin methods to representative test problems.

The *a posteriori* error estimates developed in [4] are based on the element residual method and provide bounds on the global error. Error estimates are combined with an *hp*-adaptive strategy that predicts the mesh required to deliver a solution with the specified level of error.

The theoretical developments of [4] are summarized in this paper. The discontinuous Galerkin method, the *a priori* error estimate which establishes the accuracy and convergence of the method, and the *a posteriori* error estimate used to assess the accuracy of the numerical solution are presented. The reliability of the *a posteriori* error estimate is assessed by solving two examples problems with known discontinuous solutions. The effectiveness of the adaptive strategy at delivering a solution with the specified level of error is also demonstrated using the numerical examples.

THE DISCONTINUOUS GALERKIN METHOD

Consider the following hyperbolic conservation law

$$\boldsymbol{\beta} \cdot \boldsymbol{\nabla} u + a u = f \quad \text{in } \Omega \subset \mathcal{R}^2 \tag{2}$$

$$\boldsymbol{\beta} \cdot \mathbf{n}\, u = \boldsymbol{\beta} \cdot \mathbf{n}\, g \quad \text{on } \Gamma_- \tag{3}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2)^T$ denotes a constant unit velocity vector, $\mathbf{n}$ denotes the unit normal vector pointing outward to the domain boundary $\partial\Omega$, $\Gamma_- = \{\mathbf{x} \in \partial\Omega \mid \boldsymbol{\beta} \cdot \mathbf{n}(x) < 0\}$ denotes the inflow boundary, $a = a(\mathbf{x})$ is a bounded measurable function on Ω such that $0 < a_0 \leq a(\mathbf{x})$, $f \in L^2(\Omega)$, and $g \in L^2(\Gamma_-)$. While this is the simplest of hyperbolic conservation laws, solutions to (2) may

contain discontinuities along characteristic lines $\mathbf{x}(s)$ defined by $\frac{\partial \mathbf{x}}{\partial s} = \boldsymbol{\beta}$. Solutions to (2) belong to the space of functions $V(\Omega) = \{v \in L^2(\Omega) \mid v_\beta \in L^2(\Omega)\}$ where $v_\beta = \boldsymbol{\beta} \cdot \boldsymbol{\nabla} v$.

The starting point for the discontinuous Galerkin methods is to develop an appropriate weak formulation of (2) defined on a partition of Ω into elements, denoted by $\mathcal{P}_h$. Here the elements $K \in \mathcal{P}_h$ are general quadrilaterals of diameter h_K with outward unit normals $\mathbf{n}_K$. The element boundaries ∂K have an inflow boundary $\partial K_- = \{x \in \partial K : \boldsymbol{\beta} \cdot \mathbf{n}_K < 0\}$ and an outflow boundary $\partial K_+ = \partial K \setminus \partial K_-$. The space of admissible solutions is extended to the partition using the broken space $V(\mathcal{P}_h) = \Pi_{K \in \mathcal{P}_h} V(K)$. The standard conventions in finite element meshing are assumed to be in force: $\mathcal{P}_h$ is a family of partitions $\mathcal{F}_h$ and each element K of $\mathcal{P}_h$ is the image of an invertible map F_K of a master element $\hat{K} = [-1, 1]^2$. The partitions $\mathcal{P}_h \in \mathcal{F}_h$ are regular and, in the present study, it is sufficient to take F_K as affine maps. For each partition $\mathcal{P}_h$, approximate solutions are sought in the subspace $V_p(\mathcal{P}_h) = \{v \in L^2(\Omega) \mid v|_K \circ F_K^{-1} \in Q^{p_K}(\hat{K})\}$ where $Q^{p_K}(\hat{K})$ denotes the space of functions formed by tensor products of Legendre polynomials of degree p_K on the master element $\hat{K}$. Note that the polynomial degree, p_K, may vary over different elements in the mesh and that functions $v_h^p \in V_p(\mathcal{P}_h)$ are discontinuous across element interfaces. The approximation properties of such spaces are typified by local interpolation estimates of the following type (see [5]): if $u \in H^s(K)$, there exists a constant C, independent of $h_K = \mathrm{diam}(K)$ and p_K (the minimal order of the polynomial shape functions for K), and a polynomial w of degree p_K, such that

$$||u - w||_{r,K} \leq C \frac{h_K^{\min(p_K+1,s)}}{p_K^{s-r}} ||u||_{s,K} \quad ; \quad r = 0, 1 \tag{4}$$

where $|| \cdot ||_{r,K}$ denotes the usual Sobolev norm.

The following notation is used for functions $v \in V(\mathcal{P}_h)$:

$$\left.\begin{aligned} v^{\pm} &= \lim_{\epsilon \to 0} v(\mathbf{x} \pm \epsilon \boldsymbol{\beta}) \\ v^{\text{int } K} &= v|_K(x), \quad x \in \partial K \\ v^{\text{ext } K} &= v|_L(x), \quad x \in \partial K \cap \partial L \\ (v, w)_K &= \int_K v\, w\, d\mathbf{x} \quad ; \quad ||v||_K = \sqrt{(v, v)_K} \\ \langle v, w \rangle_{\partial K} &= \int_{\partial K} v\, w\, |\boldsymbol{\beta} \cdot \mathbf{n}_K| ds \quad ; \quad \langle\langle v \rangle\rangle_{\partial K} = \sqrt{\langle v, v \rangle_{\partial K}} \end{aligned}\right\} \tag{5}$$

The discontinuous Galerkin method applied to (2) is written in the following abstract form:

Find $\hat{u} \in V_p(\mathcal{P}_h)$ such that

$$\sum_{K \in \mathcal{P}_h} B_K(\hat{u}, \hat{v}) = \sum_{K \in \mathcal{P}_h} L_K(\hat{v}) \ , \text{ for every } \hat{v} \in V_p(\mathcal{P}_h) \tag{6}$$

where (see [4])

$$B_K(\hat{u},\hat{v}) \stackrel{\text{def}}{=} (\hat{u}_\beta + a\hat{u}, \hat{v} + \delta\frac{h_K}{p_K^2}\hat{v}_\beta)_K + (1+\delta\frac{h_K}{p_K^2})\langle \hat{u}^+ - \hat{u}^-, \hat{v}^+\rangle_{\partial K_- \setminus \Gamma_-}$$
$$+ (1+\delta\frac{h_K}{p_K^2})\langle \hat{u}^+, \hat{v}^+\rangle_{\partial K_- \cap \Gamma_-} \quad (7)$$

$$L_K(\hat{v}) \stackrel{\text{def}}{=} (f, \hat{v} + \delta\frac{h_K}{p_K^2}\hat{v}_\beta)_K + (1+\delta\frac{h_K}{p_K^2})\langle g, \hat{v}\rangle_{\partial K_- \cap \Gamma_-} \quad (8)$$

and δ is a parameter with a value of 0 or 1. The method with $\delta = 1$ in (7) and (8) is the so-called streamline-upwind discontinuous Galerkin method. The additional term $\frac{h_K}{p_K^2}\hat{v}_\beta$ in the element integrals adds diffusion in the streamline direction without compromising the accuracy of the approximation. The method with $\delta = 0$ is the standard discontinuous Galerkin method which can be viewed as a higher-order extension of a cell-centered finite volume method where the coefficients of the higher-order terms in the polynomial approximation of the solution in an element are obtained from the conservation law and not by reconstruction. Integrating the first-order terms by parts in (6) with $\delta = 0$ and manipulating the result yields the familiar numerical flux formulation of the finite volume method

$$B_K(\hat{u},\hat{v}) = (\hat{u}, a\hat{v} - \hat{v}_\beta)_K + \int_{\partial K} \hat{q}(\hat{u}^{\text{int } K}, \hat{u}^{\text{ext } K})\hat{v}ds \quad (9)$$

where

$$\hat{q}(\hat{u}^{\text{int } K}, \hat{u}^{\text{ext } K}) = \frac{1}{2}\left(\hat{u}^{\text{int } K} + \hat{u}^{\text{ext } K}\right) - \frac{1}{2}|\boldsymbol{\beta}\cdot\mathbf{n}_K|\left(\hat{u}^{\text{ext } K} - \hat{u}^{\text{int } K}\right) \quad (10)$$

The error in the discontinuous Galerkin solution satisfies the following *a priori* estimate [4]:

Theorem 1 *Let $u \in H^s(\Omega)$ be a solution to (2), let $\hat{u}$ be a solution to (6), and let (4) hold. Then there exists a positive constant C, independent of h_K, p_K, and u, such that the approximation error, $e = u - \hat{u}$, satisfies the following estimate*

$$|||e|||_{hp,\beta} \leq C\left\{\sum_{K\in\mathcal{P}_h}\left[\frac{h_K^{2\mu_K}}{p_K^{2\nu_K}}\max\left(1, \frac{h_K}{p_K^2}\right)||u||_{s,K}^2\right]\right\}^{\frac{1}{2}} \quad (11)$$

where $\mu_K = \min(p_K + 1, s) - \frac{1}{2}$, $\nu_K = s - 1$, and

$$|||e|||_{hp,\beta} = \left\{\sum_{K\in\mathcal{P}_h}\left[\frac{h_K}{p_K^2}||e_\beta||_K^2 + ||e||_K^2 + \langle\langle e^+ - e^-\rangle\rangle_{\partial K_- \setminus \Gamma_-}^2 + \langle\langle e\rangle\rangle_{\partial K\cap\partial\Omega}^2\right]\right\}^{\frac{1}{2}} \quad (12)$$

The *a priori* estimate (11) establishes convergence of the method and is useful for predicting how the error in numerical solutions behaves with h-refinement or p-enrichment. Unfortunately,

its usefulness in assessing the accuracy of a given numerical solution is limited since the estimate involves unknown constants and the exact solution.

A POSTERIORI ERROR ESTIMATION

A posteriori error estimates used here are based on extensions of an element residual method of Ainsworth and Oden [6]. Element error indicators are computed by solving a suitably-constructed local problem with the element residual as data. These local indicators are used in the adaptive strategy to assess the accuracy of the solution in an element. Moreover, they contribute to a global error estimate which is accurate enough to provide a reliable assessment of the quality of the approximate solution. Detailed derivation of the *a posteriori* estimate can be found in [4].

The local problem is constructed to result in an upper bound on the error. Let ψ_K be the solution to the following local problem,

$$A_K^U(\psi_K, v_K) = B_K(e_K, v_K) = L_K(v_K) - B_K(\hat{u}_K, v_K) \quad \forall\, v \in V(\mathcal{P}_h) \tag{13}$$

where

$$A_K^U(\psi_K, v_K) \stackrel{\text{def}}{=} \frac{h_K}{p_K^2}(\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\psi_K, \boldsymbol{\beta}\cdot\boldsymbol{\nabla}v_K)_K + \bar{a}(\psi_K, v_K)_K \tag{14}$$

and $\bar{a} > 0$ is a constant. Note that the local problem differs from the conservation law, in particular, it is symmetric and induces a norm on the space $V(K)$. The solution to the local problem, measured in the norm,

$$\|\psi_K\|_{A^U(K)} = \sqrt{A_K^U(\psi_K, \psi_K)} \tag{15}$$

serves as an element error indicator in the adaptive strategy. The global error estimate is a sum of element contributions given by

$$\|\psi\|_{A^U} = \sqrt{\sum_{K\in\mathcal{P}_h} \|\psi_K\|^2_{A^U(K)}} \tag{16}$$

The solution to the local problem (13) provides an upper bound on the global error in the following sense [4]:

Lemma 1 *Let $\psi \in V(\mathcal{P}_h)$ be the solution to the following problem:*

$$\sum_{K\in\mathcal{P}_h} A_K^U(\psi_K, v) = \sum_{K\in\mathcal{P}_h} B(e, v) \qquad \forall v \in V(\mathcal{P}_h) \tag{17}$$

Then there exists a positive constant k such that

$$\|\psi\|_{A^U} \geq k|||e|||_{hp,\beta} \tag{18}$$

An approximate solution to the local problem (13) in the corresponding norm serves as a local error indicator for the element. Since the discontinuous Galerkin solution satisfies the orthogonality condition,

$$B_K(e, v) = 0 \quad \forall v \in Q^{p_K}(K) \tag{19}$$

the error indicator must be approximated with a polynomial of degree $p_K + \sigma_K$ where $\sigma_K \geq 1$ in order for the discrete local problem to have a non-trivial solution. If a complete polynomial of degree $p_K + \sigma_K$ (on the master element) is used to approximate the solution to the local problem, then the discrete local problem requires the solution of a system of order $(p_K + \sigma_K + 1)^2$. This system can be fairly large compared to the system of $(p_K + 1)^2$ equations used to obtain the approximate solution for which we are estimating the error. Since $(p_K + 1)^2$ terms on the right hand side of the discrete local problem (corresponding to (19)) are zero, a simplification is made by approximating the solution to the local problem in the space $Q^{p_K+\sigma_K}(K) \setminus Q^{p_K}(K)$. In other words, the solution to the local problem is approximated with incomplete polynomials of degree $p_K + \sigma_K$ by neglecting the terms associated with polynomials of degree p_K. This simplification results in a system of $\sigma_K(\sigma_K + 2p_K + 2)$ equations for each element.

THE *hp*-ADAPTIVE STRATEGY

The *hp*-adaptive strategy used here is an extension of the 3-step strategy developed by Oden, Patra, and Feng [7] for a large class of elliptic problems and, in several applications, was shown to yield exponential rates of convergence with respect to both CPU time and the number of unknowns.

The goal of the adaptive strategy is to deliver a solution with the specified level of error in three adaptive steps: (1) estimate the error in the solution obtained on an initial mesh (2) construct a new mesh using *h*-refinement of the initial mesh, solve the problem on the new mesh, and estimate the error, and (3) enrich the approximation in regions where the solution is smooth by increasing the spectral order of the elements in the mesh from step (2), and if necessary, perform *h*-refinement in regions where the solution is of low regularity. If the level of error after step (3) exceeds the specified level, it is necessary to repeat steps (2) and (3) until the desired error is attained.

The *hp*-adaptive strategy is based on the assumption that the *a posteriori* estimate is a reasonable approximation to the actual error in a particular solution. The *a priori* estimate (11) and some additional assumptions (see [4]) lead to expressions for estimating the local regularity of the solution and for predicting the mesh required to reduce the error to the specified level. The entire procedure is outlined below. Detailed development of the *hp*-adaptive strategy can be found in [4].

(i) Specify a target normalized error, η_T. The target error is normalized by the solution in the same norm. Specify the parameter α to determine the intermediate target error,

$\eta_I = \alpha\eta_T$. Specify the parameters α_s and α_D which establish reduction factors for the error in smooth and non-smooth regions as described below. Specify the parameters μ_K and ν_K in the *a priori* estimate (11). Formally these parameters depend on the global regularity of the solution. While there is little theoretical justification, local values can be used by computing the rate of convergence of the local error for a uniform h-refinement and p-enrichment of a coarse mesh.

(ii) Construct an initial mesh $\mathcal{P}_0$ containing $N(\mathcal{P}_0)$ elements. The elements in $\mathcal{P}_0$ have uniform $p_K = p_0$ and essentially uniform $h_K \approx h_0$. Find the approximate solution $\hat{u}_0 \in V_{p_0}(\mathcal{P}_0)$. Estimate the error θ_0 where

$$\theta_0 = \sqrt{\sum_{K\in\mathcal{P}_0} \theta_{0,K}^2} = \sqrt{\sum_{K\in\mathcal{P}_0} ||\psi_K||_{A^U}^2} \tag{20}$$

and ψ_K is the solution to the local problem (13).

(iii) Construct a mesh $\mathcal{P}_1$ by subdividing each element in $\mathcal{P}_0$ into the number of elements, n_K, required to equally distribute the error and reduce it to $\theta_I = \eta_I(||\hat{u}_0||_{A^U} + \theta_0)$. The number of elements, n_K, is obtained by iteratively solving the following two equations:

$$n_K = \left(\frac{\theta_{0,K}^2}{\theta_I^2} N(\mathcal{P}_1)\right)^{\frac{1}{\mu_K+1}} \tag{21}$$

$$N(\mathcal{P}_1) = \sum_{K\in\mathcal{P}_0} n_K \tag{22}$$

Find the approximate solution $\hat{u}_1 \in V_{p_0}(\mathcal{P}_1)$ and estimate the error θ_1.

(iv) Estimate the local regularity of the solution by computing the rate of convergence of the local error

$$\mu_K = \frac{\log\theta_{0,K} - \log\sqrt{\sum_{L=1}^{n_K}\theta_{h,L}^2}}{\log h_K - \log\frac{h_K}{\sqrt{n_K}}}, \quad K = 1, \cdots, N(\mathcal{P}_0) \tag{23}$$

The value of μ_K given by (23) is associated with an element K in the initial mesh and is simply inherited by the new elements generated by subdividing the element K. The expected rate of convergence for smooth solutions is $p_K + \frac{1}{2}$, according to the *a priori* estimate (11). Divide the error into two contributions according to the value of μ_K:

$$\theta_D = \sqrt{\sum_{K\in\Omega_D} \theta_{1,K}^2}\,; \quad \Omega_D = \{K \in \mathcal{P}_1 : \mu_K < p_K + \frac{1}{2}\} \tag{24}$$

$$\theta_S = \sqrt{\sum_{K\in\Omega_S} \theta_{1,K}^2}\,; \quad \Omega_S = \mathcal{P}_1 \setminus \Omega_D \tag{25}$$

Subdivide the elements in Ω_D into the number of elements required to equally distribute the error and reduce it to $\alpha_D\theta_D$. Enrich the approximation in Ω_S to equally distribute the error and reduce it to $\alpha_S\theta_S$ by increasing p_K according to

$$p_K = p_0 \left(\frac{\theta_{1,K}\sqrt{N(\Omega_S)}}{\alpha_S\theta_S} \right)^{\frac{1}{\nu_K}} \tag{26}$$

Find the approximate solution on the new mesh and estimate the error.

(v) If the estimated error in (iv) is larger than the target error, repeat step (iii) and (iv) until the target error is reached.

In the current implementation, h-refinement is accomplished by successive bisection of an element and is limited to two levels for a particular adaptive step. The h-refinement in (iv) is necessary only when the error θ_D exceeds the target error.

NUMERICAL EXAMPLES

The discontinuous Galerkin method with $\delta = 1$ in (6) is used to solve the model problem (2) to assess the reliability of the error estimate and to investigate the performance of the hp-adaptive strategy for problems with discontinuous solutions.

Example 1

We solve the linear model problem (2) with the following data:

(i) $\Omega = (-1,1) \times (-1,1)$

(ii) $\boldsymbol{\beta} = (1.0, 0.0)^T$

(iii) $a(\mathbf{x}) = 1.0$

(iv) $g = \begin{cases} 3e^{-5(1+y^2)} & \text{if } y < 0 \\ -3e^{-5(1+y^2)} & \text{otherwise} \end{cases}$

The source term f in (2) is chosen so that the exact solution is the discontinuous function given by

$$u(x,y) = \begin{cases} 3e^{-5(x^2+y^2)} & \text{if } y < 0 \\ -3e^{-5(x^2+y^2)} & \text{otherwise} \end{cases} \tag{27}$$

and shown in Fig. 1. The discontinuity is aligned with element interfaces at $y = 0$ to illustrate the advantage of using a discontinuous method to capture discontinuities, particularly if the adaptive scheme includes some shock fitting which aligns the grid with the discontinuity.

The problem was solved using a variety of uniform meshes with h-refinements, p-enrichments, and the hp-adaptive strategy. The error history for an hp-adaptive solution starting from an initial 8×8 mesh of $p = 1$ elements is listed in Table 1. The target error was nearly achieved at each step in the adaptive process. Recall that the target error is a global quantity obtained as the square root of the sum of the squares of the element error indicators (16). Therefore the error in a single element cannot exceed the target error. The global effectivity index, the ratio of the estimated error to the actual error, is also listed in Table 1. An effectivity index close to unity indicates that the error estimate is reliable and provides a good approximation to the actual error. The effectivity indices in Table 1 are slightly less than unity, indicating that the actual error is larger than the estimated error. However, the estimated error is sufficiently close to the actual error to result in an effective adaptive strategy.

Adaptive step	Target error	Achieved error	Effectivity index
initial (8×8 mesh, $p = 1$)	—	15.4%	0.998
h-refinement	7.5%	3.3%	0.996
p-enrichment	5.0%	5.5%	0.901

Table 1: Example 1 - Error history for an hp-adaptive solution

The rate of convergence of the estimated and exact error is compared in Fig. 2. The exact error (denoted by a solid line in the figure) and the estimated error (denoted by a dashed line) are in close agreement, indicating the reliability of the estimate. Note that with the discontinuity aligned with element interfaces, the error behaves as if the solution is smooth; that is, algebraic rates of convergence are achieved with respect to mesh refinement, and exponential rates of convergence are achieved with respect to p-enrichment. When the discontinuity is aligned with the element interfaces, the most significant error reduction with fewest degrees of freedom results by specifying a target error for the h-step which is closer to the initial error than to the final target error. This is verified by the curves corresponding to two hp-adaptive solutions in Fig. 2. The error corresponding to the hp-adaptive solutions in Fig. 2 exhibits super-linear rates of convergence.

The element residual method gives a global error estimate which bounds the actual global error, however, nothing in the theory indicates the reliability of the local element indicators. Since the local error indicators are used in the adaptive strategy, it is important that the indicators give an accurate approximation of the actual element error. The hp-adapted mesh resulting from an initial 8×8 mesh of $p = 1$ elements and the local effectivity index for the element error indicators, η_K, are shown in Fig. 3. Although there are some elements with low effectivity indices (indicating that that

actual error is much larger than the estimate), the local effectivity index for most of the elements falls between 0.8 and 1.2 indicating that the local error indicators are sufficiently accurate for use in the adaptive strategy.

Example 2

The following data is used in (2):

(i) $\Omega = (-1,1) \times (-1,1)$

(ii) $\boldsymbol{\beta} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})^T$

(iii) $a(\mathbf{x}) = 1.0$

(iv) $g(x,y) = \begin{cases} 5e^{-[\frac{1}{4}+y^2]} + 3e^{-[1+(y-\frac{1}{2})^2]} & x = -1 \\ -1 - 8e^{-5[(x-\frac{1}{2})^2+\frac{1}{4}]} & y = -1 \end{cases}$

The source term f in (2) is chosen so that the exact solution is a function which is discontinuous along the domain diagonal given by

$$u(x,y) = \begin{cases} 5e^{-[(x+\frac{1}{2})^2+y^2]} + 3e^{-[x^2+(y-\frac{1}{2})^2]} & \text{if } y > x \\ -1 - 8e^{-5[(x-\frac{1}{2})^2+(y+\frac{1}{2})^2]} & \text{otherwise} \end{cases} \tag{28}$$

and shown in Fig. 4.

The global estimated error for a sequence of uniform refinements and for several adaptive *hp*-meshes is shown in Fig. 5. The labels *hp*-adaptive in Fig. 5 refer to the adaptive strategy with only *p*-enrichment in the third adaptive step. The labels *hhp*-adaptive refer to the strategy with both *h*-refinement and *p*-enrichment in the third adaptive step. The *hp*-adaptive strategy delivers nearly linear rates of convergence with respect to the number of degrees of freedom. The rates of convergence (the slope of the lines in Fig. 5) for the adaptive strategy are higher than the rates of convergence for uniform refinement, indicating that a more accurate solution is obtained with far fewer degrees of freedom when using the *hp*-strategy. The rate of convergence obtained with the adaptive strategy determines the efficiency of the overall process, and as seen in Fig. 5, the rate of convergence depends significantly on the target and intermediate error specified.

The error history for the *hp*-adaptive solution denoted by the solid triangles in Fig. 5 is listed in Table 2. The target error was nearly achieved at each step in the adaptive process. The effectivity index (the ratio of the estimated error to the exact error) is on the order of 0.6, quite good for a discontinuous solution, but indicating that the actual error is larger than the estimated error.

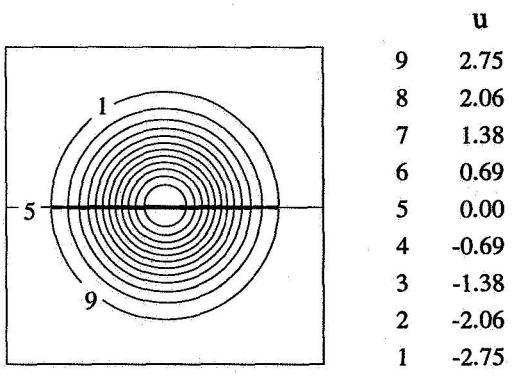


Figure 1: Example 1 - Exact solution.

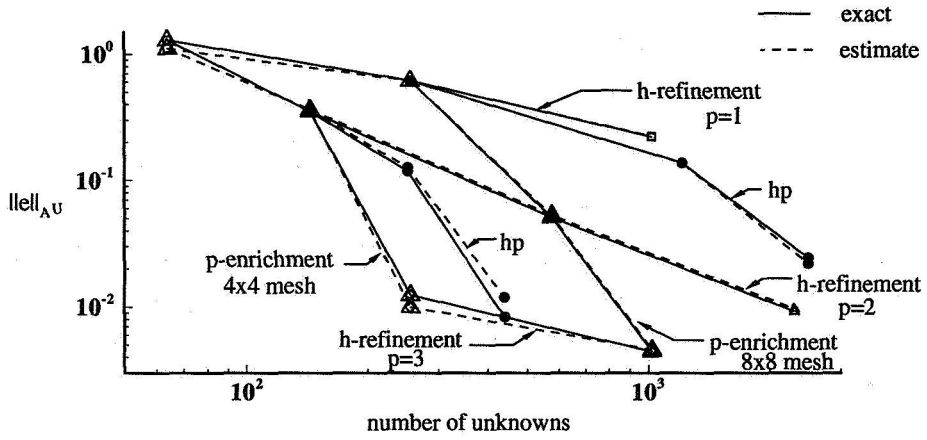


Figure 2: Example 1 - Rates of convergence of the global error with respect to the total number of unknowns.

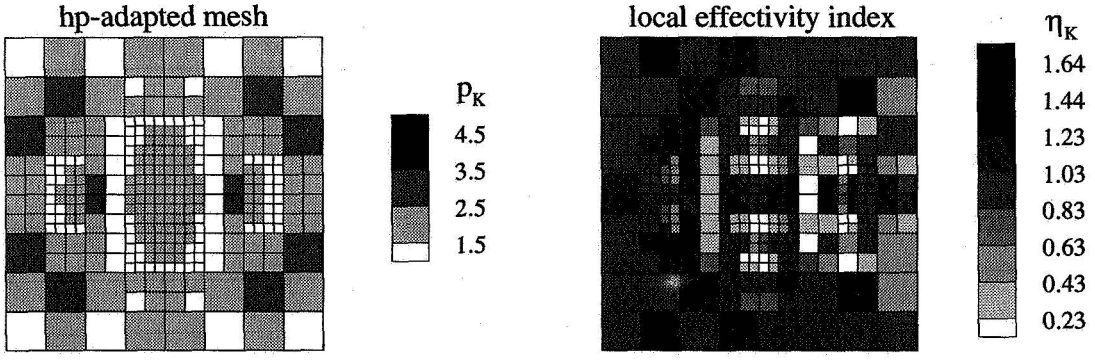


Figure 3: Example 1 - Reliability of the local error indicators for an *hp*-adapted mesh.

Adaptive step	Target error	Achieved error	Effectivity index
initial (16×16 mesh, $p = 1$)	—	7.22%	0.62
h-step	3.6%	4.1%	0.58
hp-step	2.2%	2.8%	0.57

Table 2: Example 2 - Error history for an hp-adaptive solution

Recall that the global error is a sum of element error indicators. The primary source of the under-estimation of the global error is the under-estimation of the element error indicators near the discontinuity as shown in Fig. 6. Although the local error estimate provides a qualitative measure of the error at the discontinuity, the low local effectivity index indicates some severe under-estimation of the error in that region. Note, however, that the local error estimate in smooth regions is very accurate with effectivity indices near unity.

CONCLUDING REMARKS

The development of an hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws is presented in this work. The emphasis of the work is on a model class of linear hyperbolic conservation laws for which it is possible to develop *a priori* error estimates and reliable *a posteriori* estimates which provide bounds on the actual error. These estimates are obtained using a mesh-dependent norm which reflects the dependence of the error on the local element size and the local order of the approximation.

The hp-adaptive strategy is designed to deliver solutions to a specified error level in an efficient way. This is accomplished using a three-step procedure in which the *a posteriori* estimate is used to determine the error in the solution at a particular adaptive step and the *a priori* estimate is used to predict the mesh required to deliver a solution with the specified level of error. The hp-adaptive strategy makes further use of the *a priori* estimate to provide detection of discontinuities in the solution thereby identifying regions where h-refinement and p-enrichment are appropriate.

Numerical experiments demonstrate the effectiveness of the *a posteriori* estimates in providing reliable estimates of the actual error in the numerical solution. Although local error estimates near discontinuities under-estimate the actual error, the local error estimates are very accurate in smooth regions. The numerical examples also illustrate the ability of the hp-adaptive strategy to deliver a final solution with the specified error. While the hp-adaptive strategy provides super-linear convergence rates with respect to the number of unknowns in the problem, the rate of convergence depends on the level of error requested at each step in the adaptive process. More numerical experiments are needed to provide guidelines for selecting the optimum user-specified parameters.

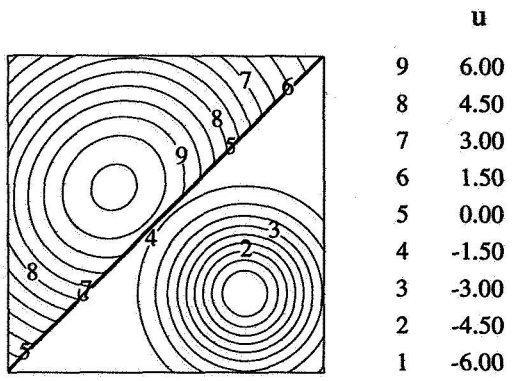


Figure 4: Example 2 - Exact solution.

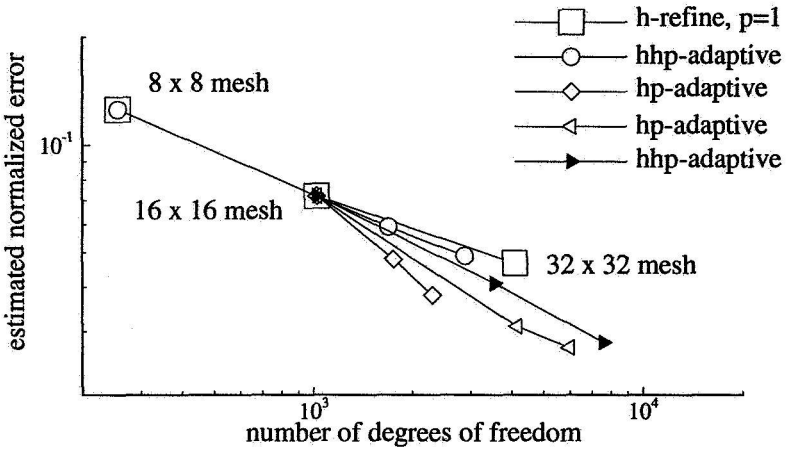


Figure 5: Example 2 - Rates of convergence of the estimated global error with the number of unknowns.

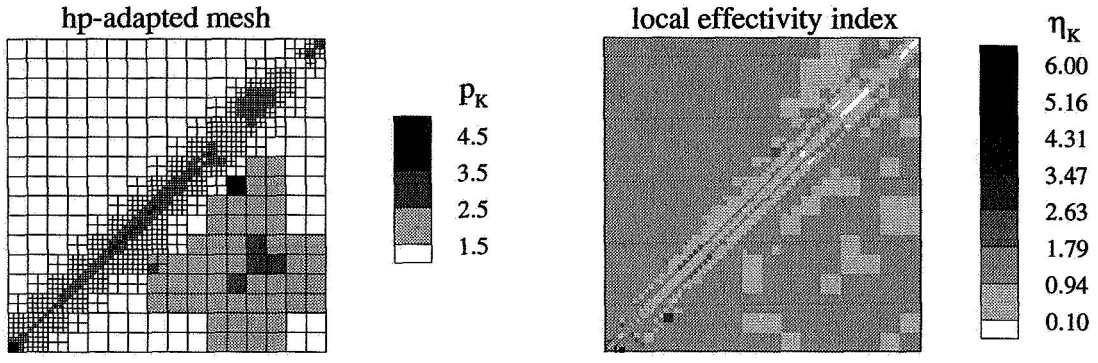


Figure 6: Example 2 - Reliability of the local error indicators for an *hp*-adapted mesh.

References

[1] Lesaint, P.; and Raviart, P.A.: On a Finite Element Method for Solving the Neutron Transport Equation. *Mathematical Aspects of Finite Elements in Partial Differential Equations*, Edited by C. de Boor, Academic Press, 1974, pp. 89–123.

[2] Johnson, C.; and Pitkaranta, J.: An Analysis of the Discontinuous Galerkin Method for a Scalar Hyperbolic Equation. *Mathematics of Computations*, Vol. 46, 1986, pp. 1–26.

[3] Cockburn, B.; and Shu, C. W.: TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Methods for Conservation Laws II: General Framework. *Mathematics of Computations*, Vol. 52, 1989, pp. 411–435.

[4] Bey, K. S.: An *hp*-Adaptive Discontinuous Galerkin Method for Hyperbolic Conservation Laws. Phd Dissertation, The University of Texas at Austin, May 1994.

[5] Babuška, I.; and Suri, M.: The hp-Version of the Finite Element Method with Quasi-uniform Meshes. *Mathematical Modeling and Numerical Analysis*, Vol. 21, 1987, pp. 199–238.

[6] Ainsworth, M.; and Oden, J. T.: A Unified Approach to A Posteriori Error Estimation using Element Residual Methods. *Numerische Mathematik*, Vol. 65, 1993, pp. 23-50.

[7] Oden, J. T.; Patra, A.; and Feng, Y. S.: An *hp* Adaptive Strategy. *Adaptive, Multilevel, and Hierarchical Computational Strategies*, A. K. Noor (ed.), AMD-Vol. 157, 1992, pp. 23-46.

MULTIGRID SOLUTION STRATEGIES FOR ADAPTIVE MESHING PROBLEMS

Dimitri J. Mavriplis
ICASE
NASA Langley Research Center
Hampton, VA

ABSTRACT

This paper discusses the issues which arise when combining multigrid strategies with adaptive meshing techniques for solving steady-state problems on unstructured meshes. A basic strategy is described, and demonstrated by solving several inviscid and viscous flow cases. Potential inefficiencies in this basic strategy are exposed, and various alternate approaches are discussed, some of which are demonstrated with an example. Although each particular approach exhibits certain advantages, all methods have particular drawbacks, and the formulation of a completely optimal strategy is considered to be an open problem.

INTRODUCTION

Although most work on adaptive meshing methods has concentrated on the logistics of refining the mesh and the formulation of suitable refinement criteria, efficient solution techniques for the resulting discrete equations are also required in order to enable both fast and accurate solutions. The use of multigrid methods as fast solvers for computational fluid dynamics problems on both structured and unstructured meshes is now well established. Adaptive meshing in particular provides a natural setting for the use of multigrid solvers. The various refined meshes generated from the adaptive process can be used to form the set of coarse and fine meshes of the multigrid sequence. The multigrid algorithm can then be used to accelerate the convergence to steady-state of the discrete equations on the finest adaptive mesh. In fact, the synergy between the two techniques is greater than may be initially apparent, and has roots in the ideas of multi-resolution (see Figure 1). The role of the adaptation process is to identify regions of the domain where the resolution of smaller scales is required and to generate these required new mesh levels, while the role of the multigrid solver is to eliminate the various high and low frequency errors of the solution on the grid level which best represents them. This has led to the development of methods such as the FAC (Full Adaptive Composite) method, [1], and to the notion of the dealgebraization of multigrid, as described by Brandt [2], where the multigrid procedure is no longer viewed as simply a fast solver for discrete equation sets, but rather as part of a complete strategy for approximating the solution to the continuous partial differential equation. Spatial convergence is achieved by

the adaptation process, while temporal or numerical convergence is achieved by the multigrid procedure. Additionally, the multigrid defect-correction (*i.e.* coarse grid source term in the multigrid formulation) can be used to devise a refinement criterion.

Although these ideas are appealing, their application to systems of non-linear equations such as those found in computational fluid dynamics is still a relatively unexplored research area. In the present work, various adaptive-meshing multigrid strategies are proposed, and evaluated both in practical terms (*i.e.* speed of convergence, complexity of V or W cycle), and in terms of how well they obey the principles of multi-resolution.

DESCRIPTION OF BASE STRATEGY

The first adaptive-meshing multigrid strategy employed is denoted as the "basic strategy". This method has been found to perform well in practice, and has been used to solve a number of inviscid and viscous steady-state cases. The approach relies exclusively on the use of unstructured meshes which greatly simplifies the task of adaptation.

Single Grid Solver

The Euler (inviscid) or Navier-Stokes (viscous) equations are discretized using a Galerkin finite element approach [3]. In the inviscid case, this reduces to a finite-volume scheme where the flow variables are stored at the vertices of the mesh, and the control volumes are formed by the union of all triangles which touch the considered vertex. This corresponds to a central difference scheme, and additional dissipative terms must be added in order to preserve stability. These are constructed as a blend of an undivided Laplacian and biharmonic operator, with the Laplacian terms used to suppress oscillations near shocks, and the biharmonic terms used to prevent odd-even decoupling in regions of smooth flow. These discrete equations are integrated in time using a five-stage time-stepping scheme devised specifically to damp high frequency error modes (as is required in a multigrid scheme). Integration to steady-state is accelerated by the use of local time-stepping and residual averaging [3,4,5].

Adaptive Meshing Procedure

Adaptively refined meshes are generated by inserting new points into the existing mesh in regions of large gradients, and connecting them to existing mesh points by Delaunay triangulation. The refinement criterion is based on simple undivided differences of one or more flow variables. The difference of the flow variables across each mesh edge is compared to the average difference across all edges of the mesh. When the difference along a given edge is larger than some fraction of the average difference, a new mesh point is added midway along

the edge. If one or more edges of a given triangle are flagged for refinement in this manner, then all three edges are refined. This ensures an isotropic refinement strategy, which is necessary to guarantee high quality meshes when using Delaunay triangulation. Once all the new mesh points have been determined, they are inserted into the existing mesh sequentially using Bowyer's algorithm for Delaunay triangulation [6]. Given an initial Delaunay triangulation, this method enables the insertion of a point anywhere in the mesh, and determines the reconnection of this point to the existing points, which is the Delaunay triangulation of this newly augmented point set. As illustrated in Figure 2, Bowyer's algorithm first identifies all triangles whose circumcircle is intersected by the new point. These triangles are then removed creating a polygonal cavity, and the new triangulation is formed by joining the new point to all vertices of the polygonal cavity. New boundary points are repositioned onto the spline curves which define the geometry of the boundaries. After all points have been inserted, the mesh is smoothed and edges are swapped in order to preserve the Delaunay property [4,5,7]. Several passes of smoothing and swapping are usually performed. The use of Bowyer's algorithm in this manner is ideally suited for adaptive meshing problems, since new meshes are constructed through local modifications of an existing mesh, which is much more efficient than global mesh regeneration. Furthermore, the Delaunay construction of the adaptive meshes prevents the appearance of degenerate connectivities which can arise with simple refinement schemes such as triangle subdivision. Although a reverse Bowyer's algorithm is simple to formulate in two-dimensions, provisions for point removal have not been implemented, since the applications here concern exclusively steady-state problems. For transient problems, point removal capabilities are essential.

Multigrid Approach

There are various possible strategies for implementing a multigrid method with adaptive meshing techniques for unstructured meshes. One approach consists of using the adaptively refined meshes as the multigrid levels themselves [4,5]. If, for example, adaptively refined meshes are created by simply subdividing the appropriate mesh triangles into four finer nested triangles, multiple adaptive refinement passes result in a sequence of fully nested adaptive meshes to which multigrid can be applied in a straight-forward manner using simple restriction and prolongation (inter-grid) operators. This approach has been pursued by several authors in the literature [8,9]. One of the drawbacks of this approach is to restrict the type of adaptive refinement strategies which may be employed, and to tightly couple the multigrid process with the adaptive mesh generation procedure. Furthermore, if the initial

unadapted grid is relatively fine (which is most often required to resolve initial flow features), multigrid efficiency will be limited by the ability to efficiently solve the discrete equations of this mesh.

The multigrid approach adopted in this work relies on a sequence of coarse and fine meshes which are essentially independent from one another [3,4,5,11]. The various meshes of the sequence are not required to be nested, or even to have common points. They simply must discretize the same physical domain. Linear interpolation is used to transfer flow variables, residuals and connections between the various meshes of the sequence. The intergrid transfer operators must be formed in a preprocessing operation, where for each vertex of a given grid, the enclosing triangle on the next coarser (or finer) grid must be determined. Once this information has been determined, grid transfer addresses and weights can be determined and stored for later use in the multigrid solution cycles. This multigrid strategy enables the adaptively refined meshes to be constructed by any means available, even global mesh regeneration. The Delaunay construction employed here, and described in the previous section, generally results in non-nested meshes, and meshes with no coincident points (due to the mesh smoothing operation which displaces the mesh points). Furthermore, additional coarser grids may be utilized to accelerate the solution of the initial grid itself. These are generated using the same global mesh generation procedure as the initial mesh, but with lower resolution throughout the domain. The basic procedure consists of generating the initial mesh, and several coarser meshes. The flow solution on the initial mesh is then obtained using this sequence of meshes in the multigrid procedure. A new adaptively refined mesh is then constructed, based on the solution on the initial mesh, and this mesh is then added as a new finer mesh to the current stack of multigrid levels. The restriction and prolongation operators between the new and the initial mesh are then computed and stored. The flow solution is interpolated from the initial mesh to the new finer mesh using these operators, and multigrid cycling resumes, using the newly augmented sequence of meshes. This procedure can be repeated, each time adding a new finer mesh to the sequence, until the desired level of accuracy is obtained, as depicted in Figure 3.

A third multigrid approach for unstructured meshes consists of constructing the sequence of coarse level meshes automatically, given a fine grid. This approach is embodied in algebraic multigrid methods [12], agglomeration strategies [13,14,15], as well as automated coarsening methods used in conjunction with the independent-mesh multigrid approach described above. Thus, in the context of adaptive meshing, each time a new finer mesh is generated, the history of adaptive refinement which resulted in this mesh is ignored, and an automated algorithm is used to generate a complete set of coarse mesh levels based on the new mesh. The philosophy in this approach is to employ multigrid simply as a fast solver for discrete equation sets, in

the same manner as an implicit method or direct solver may be used to solve the fine grid equations. Since the history of refinement is not utilized as part of the solution strategy, the multi-resolution concepts discussed previously are not exploited. Such methods have, however, proved to be advantageous, and will be discussed in more detail in the section on Adaptive Multigrid Issues.

RESULTS

The finite-volume method described above, combined with the non-nested multigrid strategy and the Delaunay point-insertion adaptive mesh-refinement technique has been used to solve various inviscid and viscous flow cases. These techniques have been implemented in a single FORTRAN code, which takes as input a sequence of coarse initial meshes, the desired number of adaptive levels, the number of cycles on each level and the refinement criteria for each level, and outputs the sequence of adaptive meshes generated and the solution obtained on the finest mesh.

Inviscid Flow Case 1

The first case consists of the inviscid transonic flow over a NACA 0012 airfoil at Mach number 0.8 and 1.25 degrees incidence. For this case, the outer boundary was approximately circular and placed at a distance of 100 chords from the airfoil. The initial mesh contained 2,112 points, and 5 coarser mesh levels were generated to accelerate the solution on this mesh. The coarsest mesh of this sequence contains only 40 points. Three levels of adaptivity were employed for this calculation. The final mesh is shown in Figure 4, and the solution in terms of Mach contours is depicted in Figure 5. This mesh contains a total of 14,219 points. Mesh refinement is evident in the region of expansion near the leading-edge, and in the vicinity of both the upper and the weak lower shock. The slip line at the trailing edge of the airfoil is however poorly resolved. The undivided gradient of density was used as the refinement criterion. Figures 6 and 7 depict the computed surface pressures and entropy for this case. The shocks are well resolved, and the lift coefficient of 0.3587 is in agreement with previously reported values [16]. Entropy, computed as

$$s = \frac{\frac{P}{P_\infty}}{\frac{\rho}{\rho_\infty}} - 1$$

should be zero for inviscid flow ahead of the shock waves. As can be seen from Figure 7, the computed values near the leading edge are well below 1%, a good indication of the local accuracy of this solution. The convergence rate of the entire adaptive process is shown in Figure 8. At each state of adaptivity, 25 W-multigrid cycles were used to converge

the solutions, and 100 W-cycles were used on the final level, in order to demonstrate the asymptotic convergence rate of this method. The residuals were reduced by 6 orders of magnitude in 100 cycles, which corresponds to an average reduction rate of 0.89. This case was performed in about 45 minutes of CPU time on an SGI Indigo R4000 workstation.

Inviscid Flow Case 2

The second case consists of transonic flow over a NACA 0012 airfoil at a freestream Mach number of 0.95 and 0 degrees incidence. For this case, two oblique shock waves and a normal shock wave are set up downstream of the airfoil. The position of the normal shock is very sensitive to the accuracy of the solution. A similar strategy to that discussed for the previous case is employed; *i.e.*, five initial meshes, three levels of adaptivity, undivided difference of density as a refinement criterion. The final mesh and solution are depicted in Figures 9 and 10 respectively. This mesh contains approximately 16,000 points. The normal shock shown in Figure 10 is located 3.06 chord lengths downstream of the trailing edges, which is slightly ahead of that reported elsewhere [17]. In this case, the outer boundary was located 130 chords away from the airfoil leading edge. A previous run on a similar mesh with the outer boundary located at 42 chords yielded a normal shock position of 2.6 chords. This highlights the sensitivity of the solution to the position of the outer boundary. The use of a simple undivided difference as refinement criterion may also be partly responsible for the inexact shock location in this case.

This case should be perfectly symmetric about the $y = 0$ axis, since the NACA 0012 profile is symmetric, and the flow incidence is zero. An appealing feature of the present adaptation strategy is that, in such cases, given an initial symmetric grid, the adaptively refined grids remain perfectly symmetric, as can be seen from Figure 9. In the final solution, the lift coefficient remained zero, to 6 significant figures.

Inviscid Flow Case 3

The third test case involves the inviscid subsonic flow over the Sudhoo-Hall four element airfoil. The freestream Mach number is 0.2, and the incidence is 0 degrees. Three levels of adaptivity were used for this case, beginning with an initial mesh of 6,466 points. Four coarser meshes were employed to accelerate the convergence on the initial mesh. Thus, a total of 8 mesh levels were used in the final phase of the calculations. The final mesh contained a total of 22,792 points, and is depicted in Figure 11. Figures 12 and 13 depict the computed surface pressures and surface entropy on the finest mesh. As can be seen, the entropy is less than 0.1% over the entire configuration, indicating a good level of local accuracy in the solution. The lift and drag coefficients for this case were 4.9245 and -0.0038 respectively.

For inviscid isentropic flows, the overall drag should vanish. Thus the drag value of -38 counts is a good indication of the global accuracy of the solution. Figure 14 depicts the convergence rate for this case, where 100 multigrid W-cycles were performed at each level of adaptivity. The slopes of the various multigrid convergence histories are nearly identical on the four different mesh levels, demonstrating the mesh independent convergence property of the multigrid algorithm. Convergence on the final mesh is only slightly slower than that on the initial levels, resulting in an average reduction rate of 0.925. The convergence history of the computed lift coefficient is also plotted. On each mesh level, the lift coefficient comes very close to its final value in less than 50 cycles. The effect of grid convergence can also be seen by the diminishing differences between the final lift values on consecutively finer meshes. Figure 14 thus illustrates the concept of using adaptive-multigrid as a method of solving for the continuous set of partial differential equations, with the lift coefficient converging to the infinite resolution value, and the multigrid procedure driving the numerical solution on each level. This entire run, including all mesh adaptivity, was achieved in approximately 2 hours on an SGI Indigo R4000 workstation.

Viscous Flow Case

This case consists of viscous turbulent flow over a three-element high-lift airfoil section. The far-field boundary was placed at a distance of 50 chords away from the airfoil (wind-tunnel walls were not modeled in this case). The finite-element discretization of the Navier-Stokes equations described previously was employed, and the single equation turbulence model of Spalant-Allamaras [18] was implemented to account for turbulence effects. The same multigrid strategy described previously was employed to solve both the flow equations and the turbulence equation in a loosely coupled approach. The mesh refinement procedure required some modification for the highly-stretched meshes which are typically used for viscous flows. The Delaunay in-circle criterion described above is used in a mapped space, (resulting in a Delaunay in-ellipse criterion) for both the initial mesh construction, and subsequent adaptive refinement operations [19]. When new boundary points are generated by the refinement procedure, these must be displaced in order to coincide with the surface splines which define the body shape. Whereas in the inviscid case this was easily achieved, in the viscous case, this displacement can require the restructuring of many layers of grid cells near the boundary. This is due to the possibility of the boundary point displacement being much larger than the local normal grid spacing for highly stretched meshes. Thus, a system of pointers is managed, in order to enable local mesh reconstruction near the boundary [19].

For this case, the freestream Mach number is 0.2, the incidence is 16 degrees, and the Reynolds number is 9 million. Three levels of adaptivity were employed. The initial mesh

contained approximately 25,000 points, while the final adaptive mesh which is depicted in Figure 15, contains 120,307 points. This mesh exhibits very high resolution in the regions of rapid expansions and in boundary layer and wake regions. A combination of (undivided) pressure and Mach number gradients were employed to identify inviscid and viscous phenomena for refinement. The solution in terms of computed surface pressure, is depicted in Figure 16. This case involved a total of 7 multigrid levels (three adaptive levels, four initial levels). The solution was obtained by running 100 multigrid W-cycles on each mesh, and 300 cycles on the final mesh. The residuals were reduced by 2.5 orders of magnitude on the finest mesh in 300 cycles. This rate is substantially slower than for the inviscid cases, and is primarily due to the stiffness associated with high grid stretching. For the viscous flow cases, the mesh adaptivity operations are run as a separate job with a stand-alone code.

This case has been computed previously on non-adapted meshes of high resolution (up to 240,000 points) and compared extensively with experimental data [20]. Although the solution in Figure 16 appears well resolved, there are certain features, (such as the wake of the slat element for example), which are lost prematurely when compared with the results of [20], due to inadequate grid resolution. This illustrates the difficulty in applying adaptive meshing to viscous flows, where features such as wakes are both spatially hyperbolic and nonisotropic, and highlights the need for better refinement criteria.

ADAPTIVE MULTIGRID ISSUES

Although the previous examples demonstrate the effectiveness of multigrid as an efficient solution strategy for adaptive meshing problems, certain characteristics of adaptive problems can degrade the overall efficiency of the above multigrid approach. These manifest themselves, not as degradations of the observed convergence rates, but rather as unwanted increases in complexity (number of operations) of the multigrid cycle. For example, in the non-adaptive two dimensional case, the complexity of a V-cycle is bounded by 4/3 work units, and that of a W-cycle by 2 work units, where a work unit is defined as the equivalent work of one fine grid iteration (see Figure 17 for the definition of these cycles). Here, the meshes are not generated adaptively, and the above bounds are computed assuming each coarser mesh level contains 1/4 the number of points of the previous level. In the case of adaptively generated meshes, where such relations between the complexities of the various mesh levels no longer hold, the V-cycle complexity becomes equal to the sum of the complexities of all meshes in the sequence, while the W-cycle complexity can become so high as to make it impractical.

Even the V-cycle complexity is much higher than it need be. For adaptively refined meshes, refinement only occurs in localized regions of the mesh, and there are large regions of the domain where the mesh resolution is essentially unaltered between mesh levels. Repeatedly time-stepping in these regions of the mesh on various levels represents a waste of computational effort. In this section, two strategies which overcome this increase in complexity for V-cycles are described. A third approach which results in optimum complexity, thus enabling the use of V or W cycles, is finally discussed.

The Zonal Fine Grid Scheme

The basic idea behind this scheme [21] is to omit time-stepping in regions of the mesh which have not been refined with regards to the previous level. A crude implementation consists of making use of the same multigrid strategy as described previously, but blanking out the appropriate vertices on each mesh level. In actual fact, the fine mesh consists only of the regions which have been refined, with possibly some extra buffer layers. The method can be implemented by only storing these regions at each level in order to save memory (although this has not been done in this work).

As an example, consider the adaptive mesh used to compute the inviscid flow over a tandem airfoil configuration, shown in Figure 18. This mesh is the result of 6 levels of adaptivity. For the zonal fine grid scheme, the 3rd and 4th adaptive levels are depicted in Figure 19. Figure 20 compares the convergence rates of the zonal-fine grid scheme with that of the global multigrid scheme described previously for this case. There are in fact 8 mesh levels in both multigrid cases, 2 initial global levels, and 6 adaptively generated levels. (The global levels are identical for both schemes). The freestream Mach number is 0.7, and the incidence is 3 degrees. The resulting transonic flow solution is qualitatively depicted in Figure 21. Both multigrid schemes converge at nearly identical rates, in terms of residual reduction per cycle. This result verifies the fact that multigrid time-stepping in regions where no change in resolution occurs is unnecessary. The advantage of the zonal fine grid scheme is the result of the reduction in complexity of the multigrid cycle, as shown in Figure 20. For this case, the zonal fine grid scheme is seen to be roughly twice as efficient as the global multigrid approach.

This so-called zonal fine grid scheme developed in [21] is the unstructured mesh equivalent of the fast-adaptive-composite scheme (FAC) [1], and as such embodies the multi-resolution principles outlined in the introduction. Each mesh level is responsible for resolving a particular range of scales, and highly disparate length scales are not found on any common mesh, as is the case in a global mesh with localized regions of adaptive refinement.

One of the drawbacks of this method is that the final solution lies on a composite mesh which is spread over various multigrid levels. Aside from practical difficulties involved in postprocessing the solution, this complicates other issues, such as the requirement of constructing a conservative discretization in the final solution, as well as the use of different schemes on fine and coarse mesh levels.

Zonal Coarse Grid Schemes

The idea of the zonal coarse grid scheme is to overcome the difficulties encountered in the zonal fine grid scheme due to the composite nature of the final solution, by maintaining a global fine grid upon which the final solution is based. In order to maintain favorable complexity, time-stepping is omitted on the coarser meshes in regions of the domain where no mesh refinement takes place between two consecutive levels. This strategy is illustrated in Figure 22, using one-dimensional linear meshes, and compared to the zonal fine-grid and global multigrid strategies. The overall complexity of the zonal fine grid and coarse grid schemes are necessarily equivalent. As can be inferred from the figure, the zonal fine grid and coarse grid schemes are equivalent, except that in the former case the non refined mesh regions are represented on the coarse level meshes, whereas in the latter, these are assigned to the finest possible mesh level. Hence, the zonal coarse grid scheme simply corresponds to a reordering of the local unrefined and refined mesh levels.

The convergence rate of the zonal coarse grid scheme is compared with that of the zonal find grid scheme and the global multigrid scheme for the transonic tandem-airfoil case on the mesh of Figure 18. As expected, all three methods yield similar convergence rates on a per cycle basis, while the zonal fine and coarse grid schemes achieve a factor two gain in efficiency over the global multigrid scheme in this case due to the reduction in complexity, as shown in Figure 20. Thus the zonal coarse grid scheme is equivalent to the zonal fine grid scheme in terms of efficiency, but enables the final solution to be computed on a global fine grid. The disadvantage of this approach is that each time a new adaptively refined mesh is generated, the zonal coarse meshes must be reassigned to the appropriate levels.

Aggressive Coarsening Strategies

While the zonal fine and coarse grid schemes achieve substantial reduction in the complexity of a multigrid cycle for adaptively generated meshes, the use of a W-cycle with such schemes is still unpractical, due to the relative complexities of the various mesh levels. Since the W-cycle performs frequent visits to the coarse level meshes within a single cycle, the mesh complexity must be reduced by at least a factor of four when going to the next coarser level in order to guarantee a bound on the overall W-cycle complexity, as the number of

mesh levels increases. Another characteristic of the zonal multigrid schemes described above is that they rely on the adaptive refinement history in order to identify the coarse and fine mesh levels. Such methods cannot be used effectively in the cases where this information is not available, or in the case of a mesh of arbitrary construction.

Automated coarsening strategies can be employed to overcome these difficulties. Given a fine mesh, these methods automatically generate coarser level meshes for use in the multigrid algorithm. Algebraic multigrid [12], and agglomeration multigrid [13,14,15] are examples of automated coarsening strategies. Automated coarsening algorithms have also been devised for use with the fully nested multigrid approaches [10] and the non-nested approach [22]. These methods are attractive because they are fully automated and can be applied to any given grid, regardless of its construction. These methods represent a philosophy in which multigrid is decoupled form the adaptive process, and employed simply as a fast solver for a discrete fine grid problem, much in the same manner as an implicit or direct solver would be employed.

Aggressive coarsening relates to the attempt in an automated coarsening process to optimize the complexity of the generated coarse mesh levels. For a multigrid smoother which is designed to damp high-frequency errors (as is usually the case), the optimal reduction in coarse grid complexity between two successive levels is 4:1 in two dimensions, and 8:1 in three dimensions. Aggressive coarsening strategies can be devised which result in such reductions of mesh complexity, thus resulting in an overall multigrid cycle of near optimal complexity, and enabling the use of V or W-cycles. Although the complexity of the multigrid cycle may be optimal, the overall solution efficiency can only be competitive provided the multigrid convergence rate does not degrade substantially. Figure 23 provides a comparison between the coarse mesh level obtained by two passes of aggressive coarsening on the fine mesh of Figure 18, and the equivalent mesh from the global multigrid sequence (6th level out of 8). Because each cell of the original grid is forced to "grow" at the same rate, the large outer boundary cells are seen to grow much more rapidly throughout the coarsening process than the small refined cells in the shock region of the fine mesh. This results in large discontinuities in cell size which become even more pronounced on the coarser levels. This in turn may degrade the observed convergence rate of a multigrid scheme based on these mesh levels. A similar behavior is observed for agglomeration multigrid methods [15]. Aggressive coarsening strategies are evidently in complete violation of the multi-resolution principle associated with adaptive multigrid methods, where each mesh level is responsible for a given range of scales. Not only does each mesh level contain a wide range of scales in the present approach, but the bandwidth of this range increases on the coarser mesh levels.

Nevertheless, for many problems, aggressive coarsening strategies are highly desirable, both due to their fully automatic nature, and their low complexity. Such methods could obviously be improved by trading off complexity for more regularity in the coarse mesh levels, and thus better multigrid efficiency. However, this task generally requires global information about the current fine mesh construction (*i.e.* in the adaptive mesh case the history of refinement). This has important implications for the future design of automated coarsening techniques, since at present, most of these methods (including algebraic multigrid methods) rely exclusively on local information for constructing coarser levels.

CONCLUSION

Multigrid methods and adaptive meshing techniques have been shown to be complimentary strategies which, when combined in the appropriate manner, can lead to a powerful method which enables rapid convergence, both numerically and spatially, to the continuous partial differential equation. Such methods naturally embody the principle of multiresolution where each mesh level is responsible for the spatial and numerical resolution of given length scales. In practice, strict adherence to these principles is not always possible or desirable. Successful methods must achieve a balance between complexity, convergence efficiency, practicality, and ease of implementation.

A non-nested multigrid approach which utilizes each new adaptively refined mesh as an additional multigrid level has been shown to work well in practice for a range of fluid dynamics problems. The simple refinement criterion based on gradients in the flow solution is not sufficiently reliable for application to all types of flows, particularly in the viscous case. Improved refinement criteria and/or better error estimates are sorely needed before adaptive meshing can be routinely used with confidence for complex viscous flows.

REFERENCES

[1] McCormick, S., and Thomas, J., "The Fast Adaptive Composite Grid (FAC) Method for Elliptic Equations," Mathematics of Computations, Vol. 46, 1986, pp. 439-456.

[2] Brandt, A., "Multigrid Techniques with Applications to Fluid Dynamics: 1984 Guide" Lecture Notes for the Computational Fluid Dynamics Lecture Series, von-Karmen Institute for Fluid Dynamics, Belgium, March 1984.

[3] Mavriplis, D. J., "Turbulent Flow Calculations Using Unstructured and Adaptive Meshes," Int. J. Numer. Methods Fluids, Vol. 13, No. 9, November 1991, pp. 1131-1152.

[4] Mavriplis, D. J., and Jameson, A., "Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," ICASE Report No. 87-53, Proceedings of the Third Copper Mountain Conference on Multigrid Methods, Lecture Notes in Pure and Applied Mathematics, Ed. S. F. McCormick, Marcel Dekker Inc., April 1987, pp. 413-430.

[5] Mavriplis, D. J., "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes, AIAA Journal, Vol. 28, No. 2, February 1990, pp. 213-221.

[6] Bowyer, A., "Computing Dirichlet Tessalations," The Computer Journal, Vol. 24, No. 2, 1981, pp. 162-166.

[7] Lawson, C. L., "Transforming Triangulations," Discrete Mathematics, Vol. 3, pp. 365-372, 1972.

[8] Perez, E., "Finite Element and Multigrid Solution of the Two-Dimensional Euler Equations on a Non-Structured Mesh," INRIA Report No. 442, September 1985.

[9] Connell, S. D., and Holmes, D. G., "Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Euler Equations," AIAA Journal, Vol. 32, No. 8, pp. 1626-1632, 1994.

[10] Parthasarathy, V., and Kallinderis, Y., "New Multigrid Approach for Three-Dimensional Unstructured Adaptive Grids," AIAA Journal, Vol. 32, No. 5, May 1994.

[11] Mavriplis, D. J., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," AIAA Journal, Vol. 26, No. 7, July 1988, pp. 824-831.

[12] Ruge, J. W., and Stuben, K., Algebraic Multigrid, in Multigrid Methods, McCormick, S. F., Editor, SIAM Frontiers in Applied Mathematics, SIAM, Philadelphia, 1987, pp. 73-131.

[13] Lallemand, M. H., Steve, H., and Dervieux, A., "Unstructured Multigridding by Volume Agglomeration: Current Status," Computers and Fluids, Vol. 21, pp. 397-433, 1992.

[14] Smith, W. A., "Multigrid Solution of Transonic Flow on Unstructured Grids," Recent Advances and Applications in Computational Fluid Dynamics, Proceedings of the ASME Winter Annual Meeting, Ed. O. Baysal, November 1990.

[15] Venkatakrishnan, V. and Mavriplis, D., "Agglomeration Multigrid for the 3D Euler Equations," AIAA Paper 94-0069, January 1994.

[16] Pulliam, T. H., and Barton, J. T., "Euler Computations of AGARD Working Group 07 Airfoil Test Cases," AIAA Paper 85-0018, January 1985.

[17] Warren, G., Anderson, W. K., Thomas, J., and Krist, S., "Grid Convergence for Adaptive Methods," AIAA Paper 91-1592-CP, June 1991.

[18] Spalart, P. R., and Allmaras S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, January 1992.

[19] Mavriplis, D. J., "Unstructured and Adaptive Mesh Generation for High-Reynolds Number Viscous Flows," Proc. of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Eds. A. S. Arcilla, J. Hauser, P. R. Eiseman, J. F. Thompson, North Holland, June 1991.

[20] Valarezo, W. O., and Mavriplis, D. J., "Navier-Stokes Applications to High-Lift Airfoil Analysis," AIAA Paper 93-3534, AIAA 11th Applied Aerodynamics Conference, Monterey CA, August 1993.

[21] Mavriplis, D. J., "Zonal Multigrid Solution of Compressible Flow Problems on Unstructured and Adaptive Meshes," ICASE Report No. 89-35, NASA CR-181848, May 1989.

[22] Guillard, H., "Node Nested Multigrid with Delaunay Coarsening," INRIA Report No. 1898, 1993.

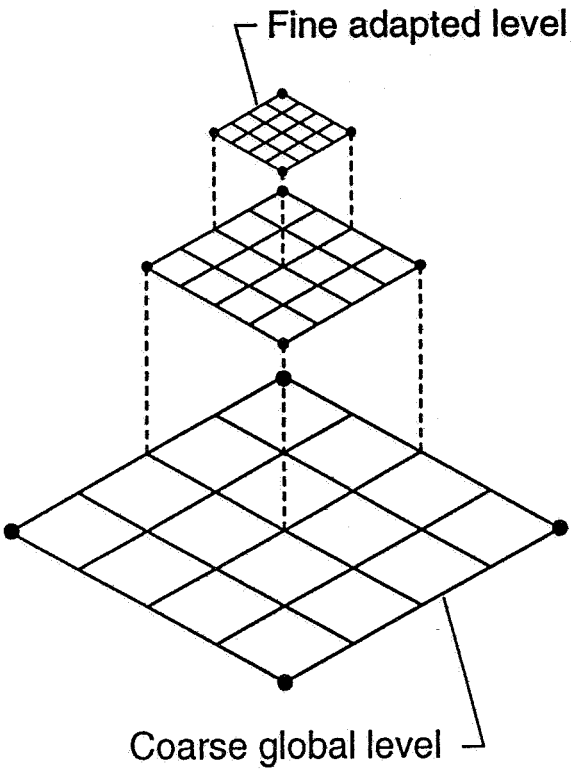


Figure 1: Illustration of the Ideal Multi-resolution Principle of Adaptive Meshing Combined with Multigrid where Each Mesh Level of the Multigrid Sequence Represents a Unique Resolution Scale.

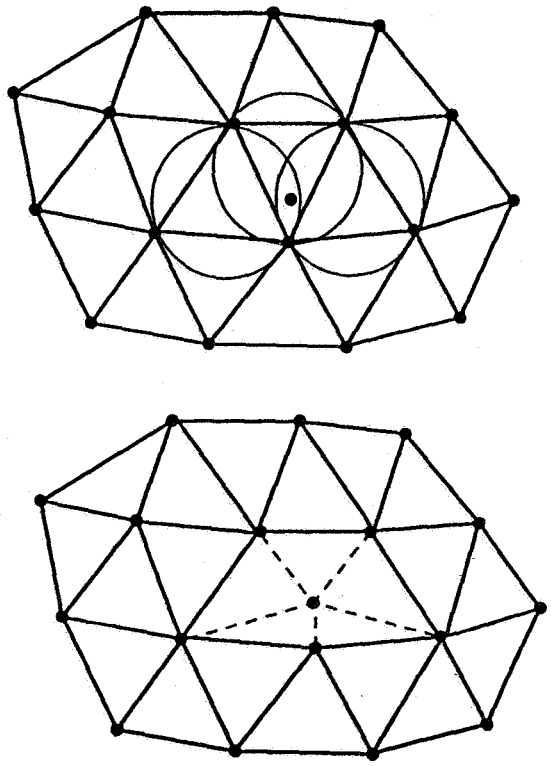


Figure 2: Illustration of Bowyer's Algorithm for Delaunay Triangulation New Point is Inserted into Existing Mesh By Removing all Triangles whose Circumcircles Contain the New Point, and Rejoining the New Point to All Vertices of the Resulting Cavity.

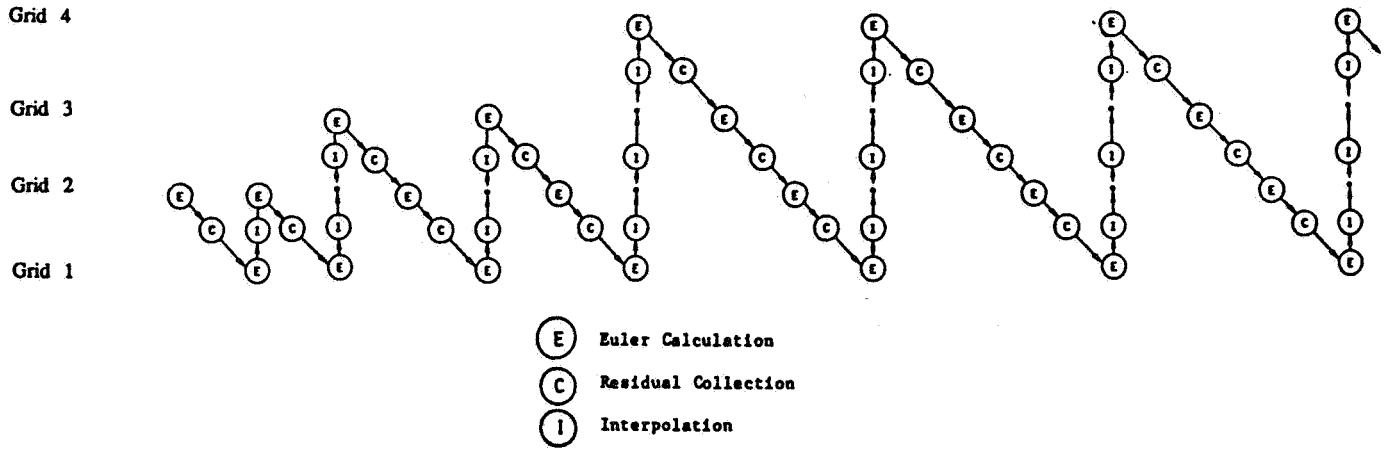


Figure 3: Full Multigrid Strategy Used in Conjunction with Adaptive Meshing Each New Adaptive Mesh is Added onto the Stack, the Solution is Interpolated onto the New Mesh, and Multigrid Cycling Resumed.

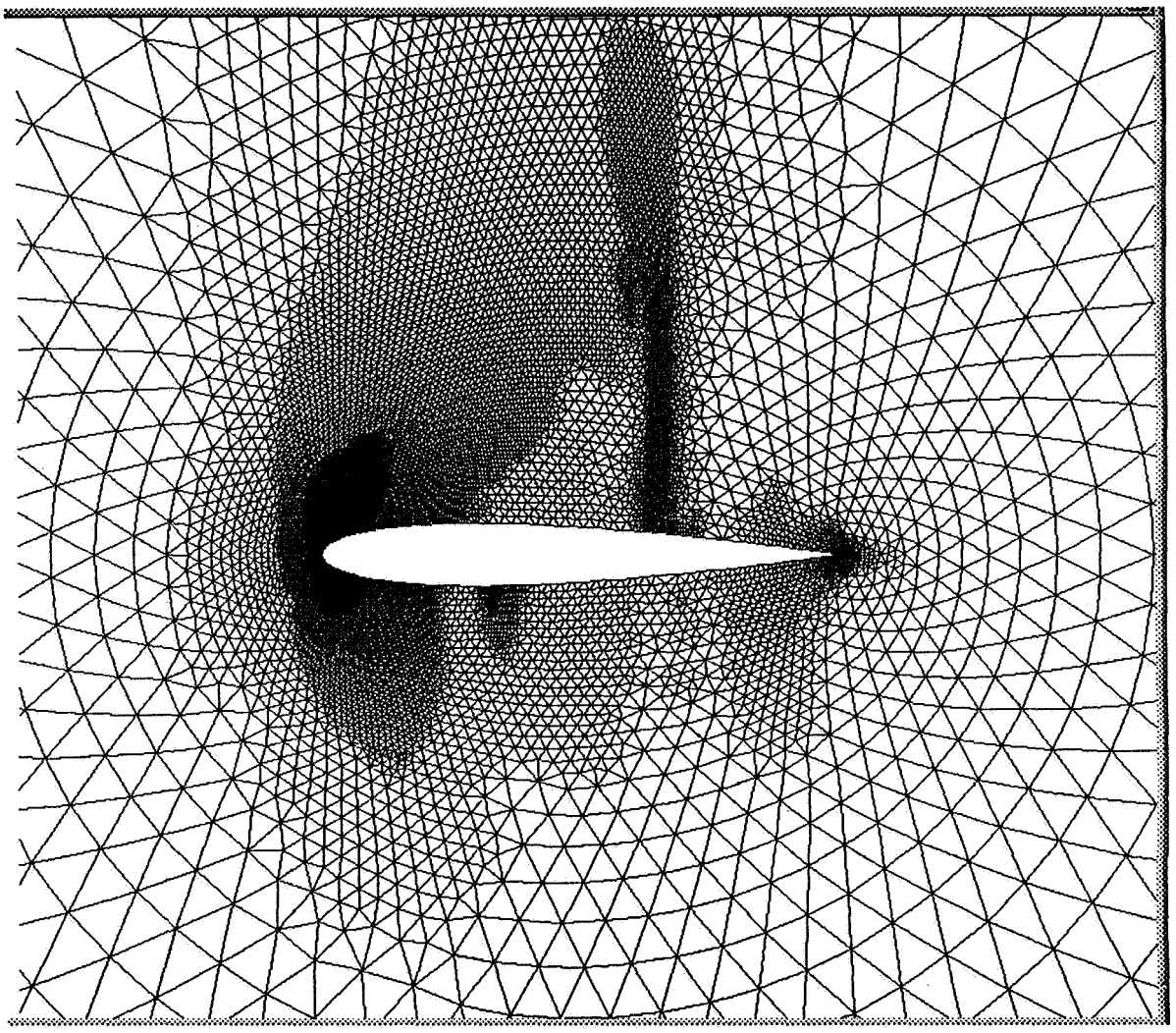


Figure 4: Final Adapted Mesh for Flow Over NACA 0012 Airfoil
(Number of Points: 14,219)

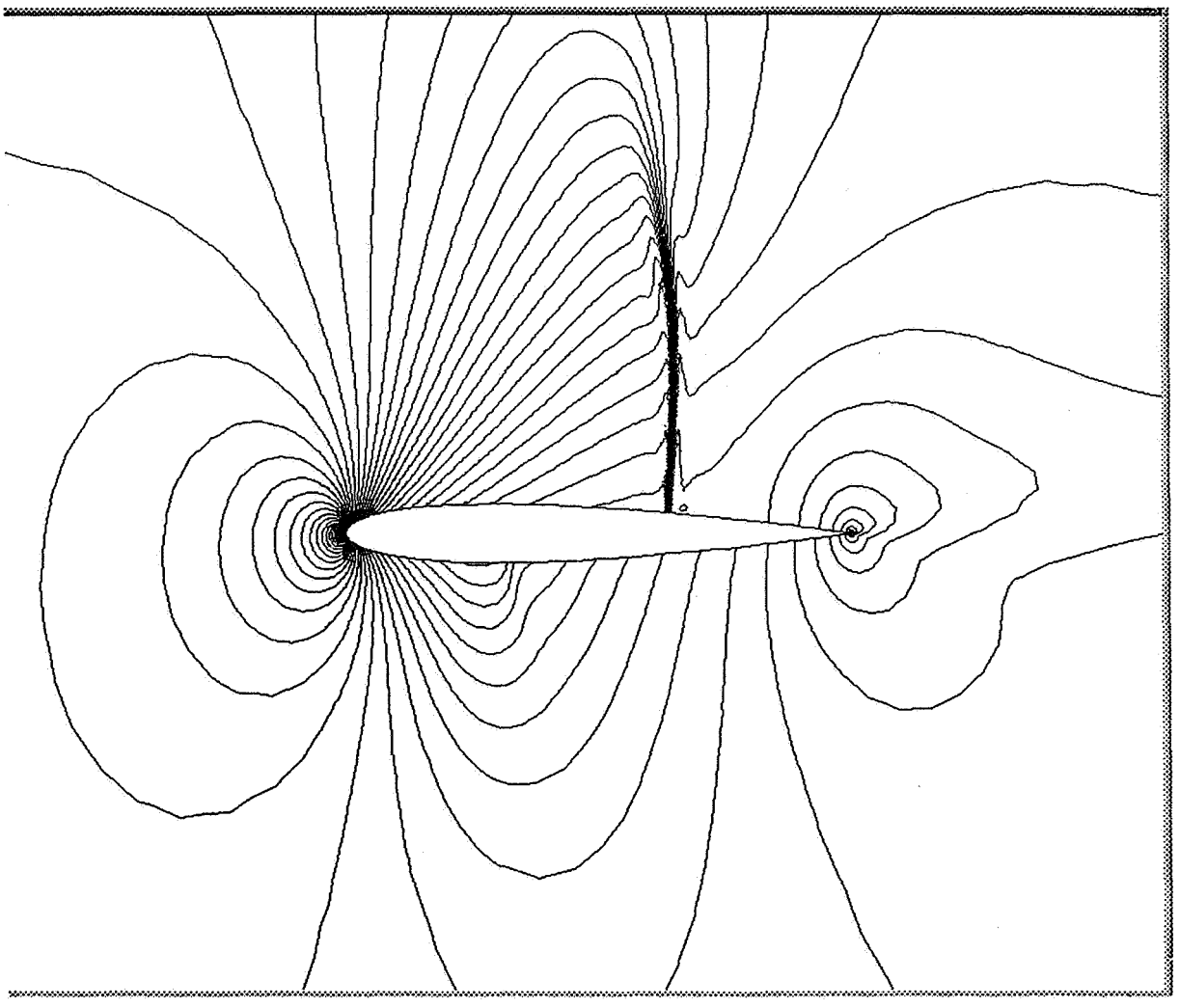


Figure 5: Computed Mach Contours on Adapted Mesh over NACA 0012 Airfoil (Mach = 0.8, Incidence = 1.25 degrees)

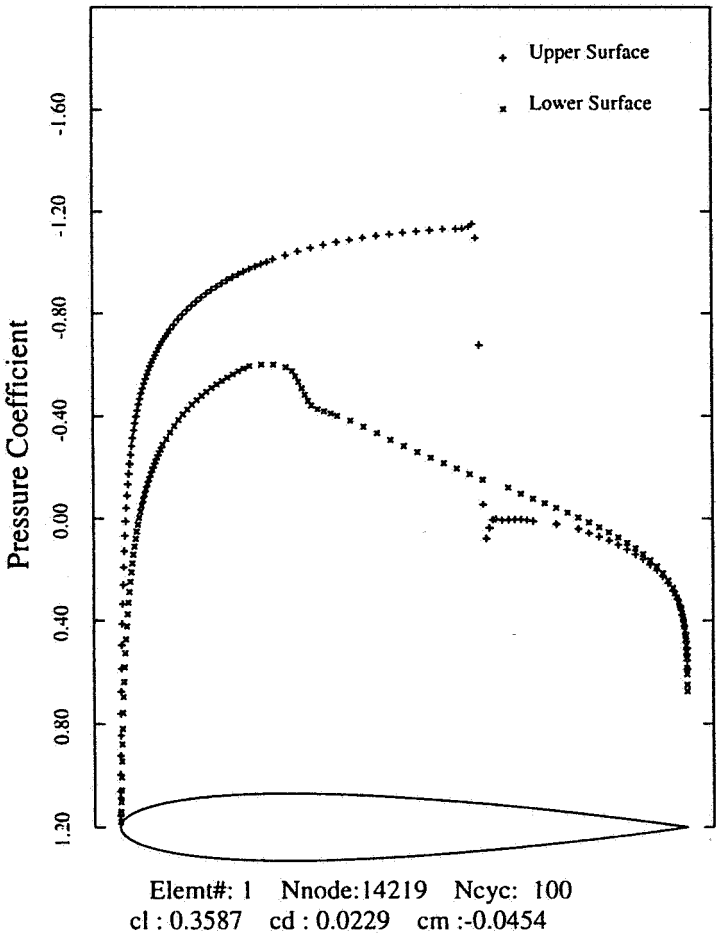


Figure 6: Computed Surface Pressure Distribution for Flow over NACA 0012 Airfoil (Mach = 0.8, Incidence = 1.25 degrees)

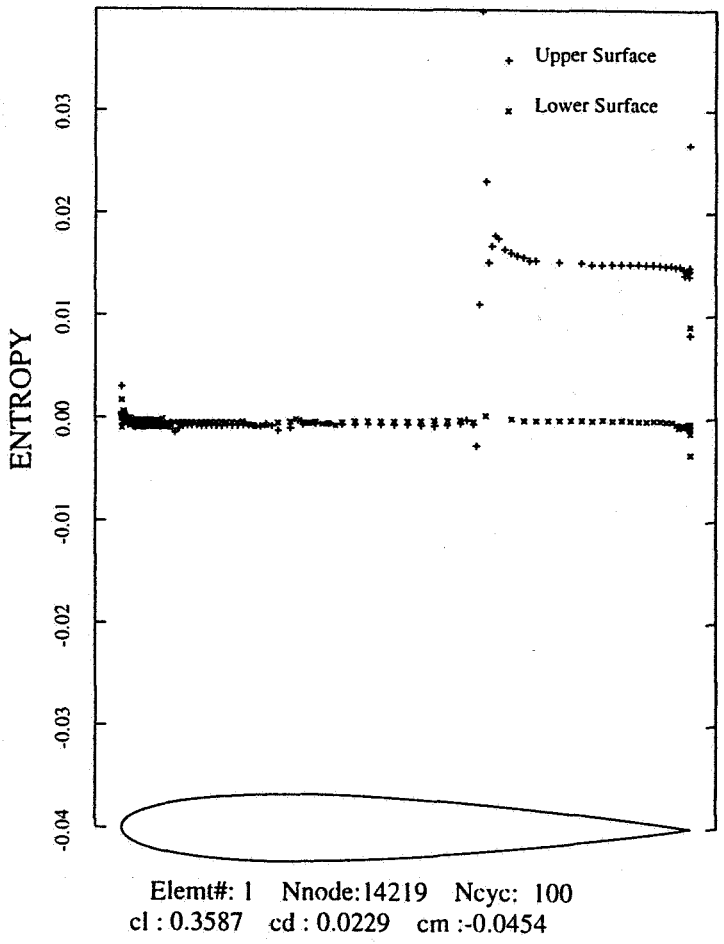


Figure 7: Computed Surface Entropy Distribution for Flow over NACA 0012 Airfoil (Mach = 0.8, Incidence = 1.25 degrees)

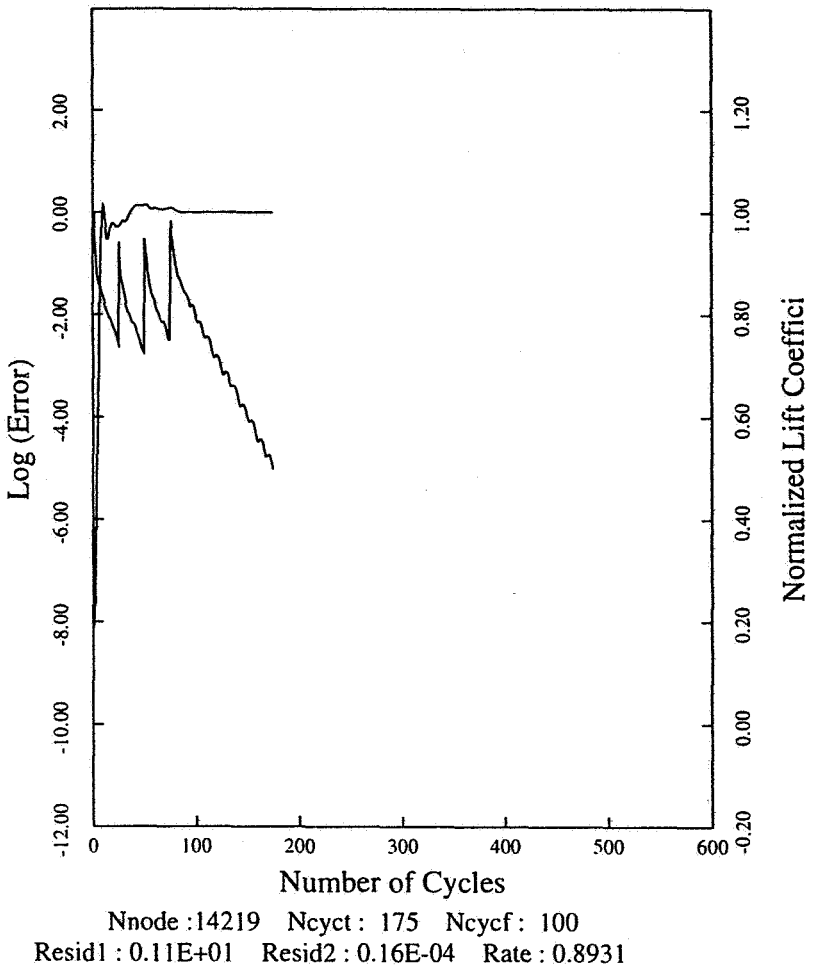


Figure 8: Full Multigrid Convergence Rate on Initial and Three Adapted Meshes for Flow over NACA 0012 Airfoil (Mach = 0.8, Incidence = 1.25 degrees)

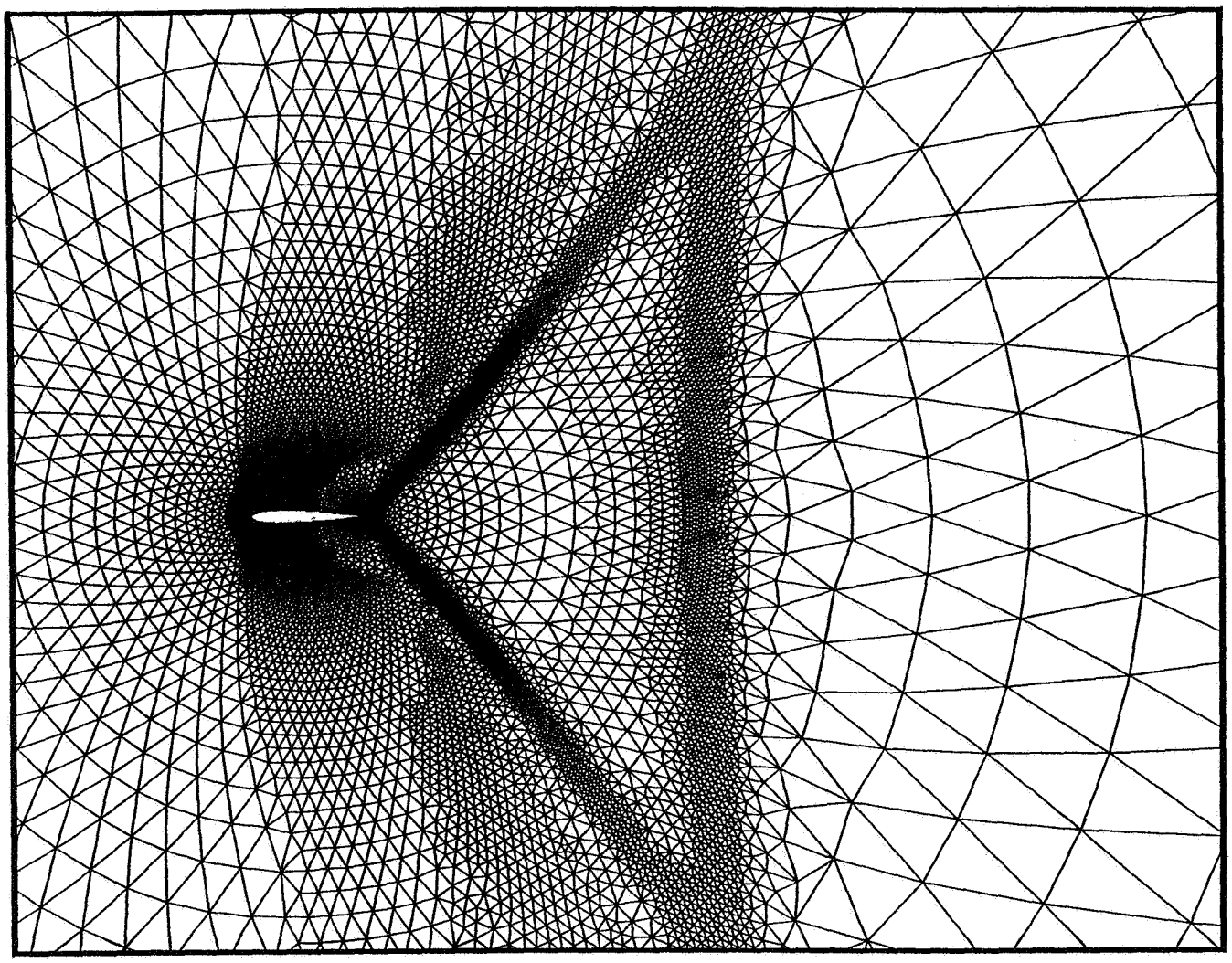


Figure 9: Final Adapted Mesh for Flow over NACA 0012 Airfoil
(Mach = 0.95, Incidence = 0 degrees, Number of Points = 16,000)

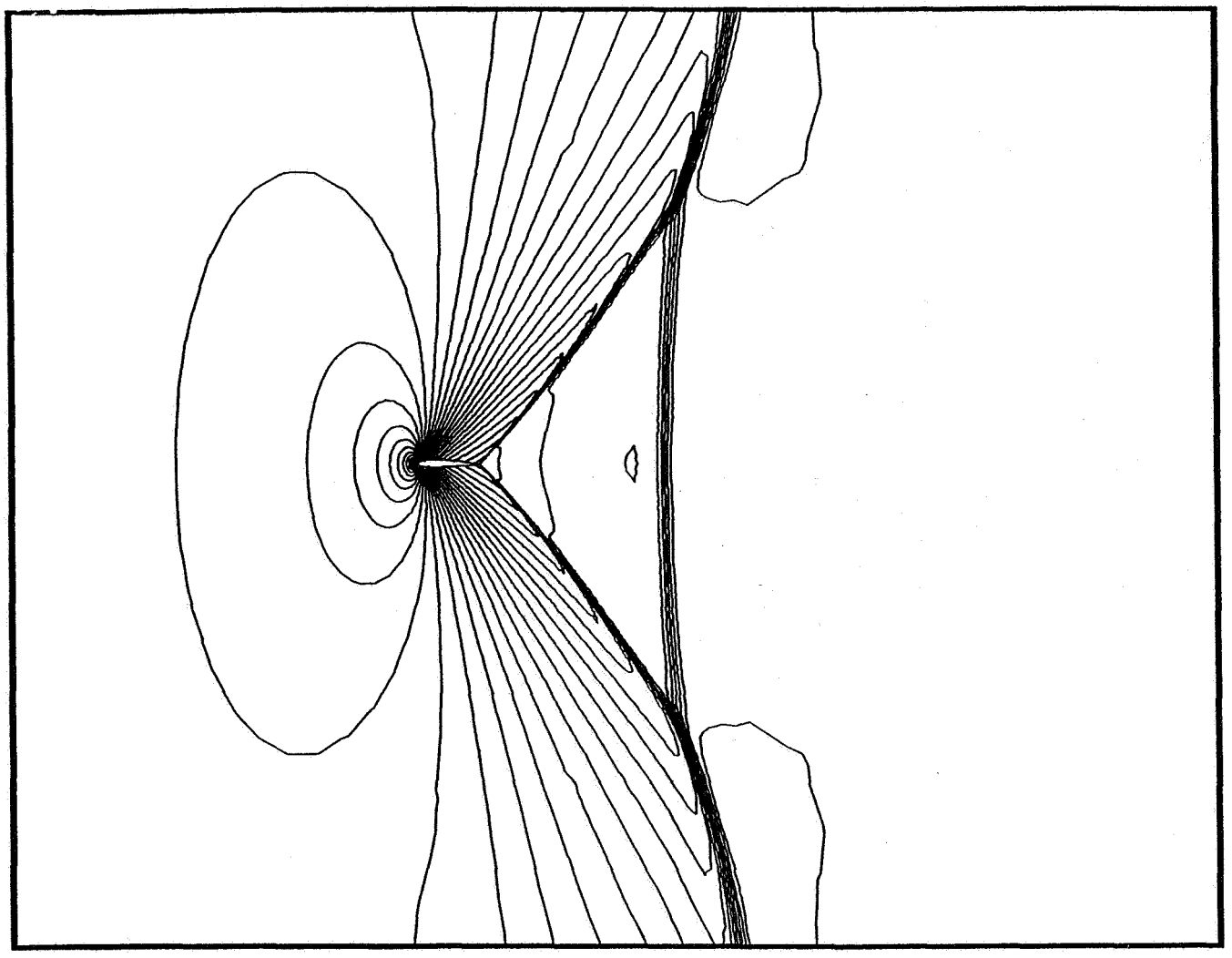


Figure 10: Computed Mach Contours on Adapted Mesh for Flow over NACA 0012 Airfoil (Mach = 0.95, Incidence = 0 degrees, Number of Points = 16,000)

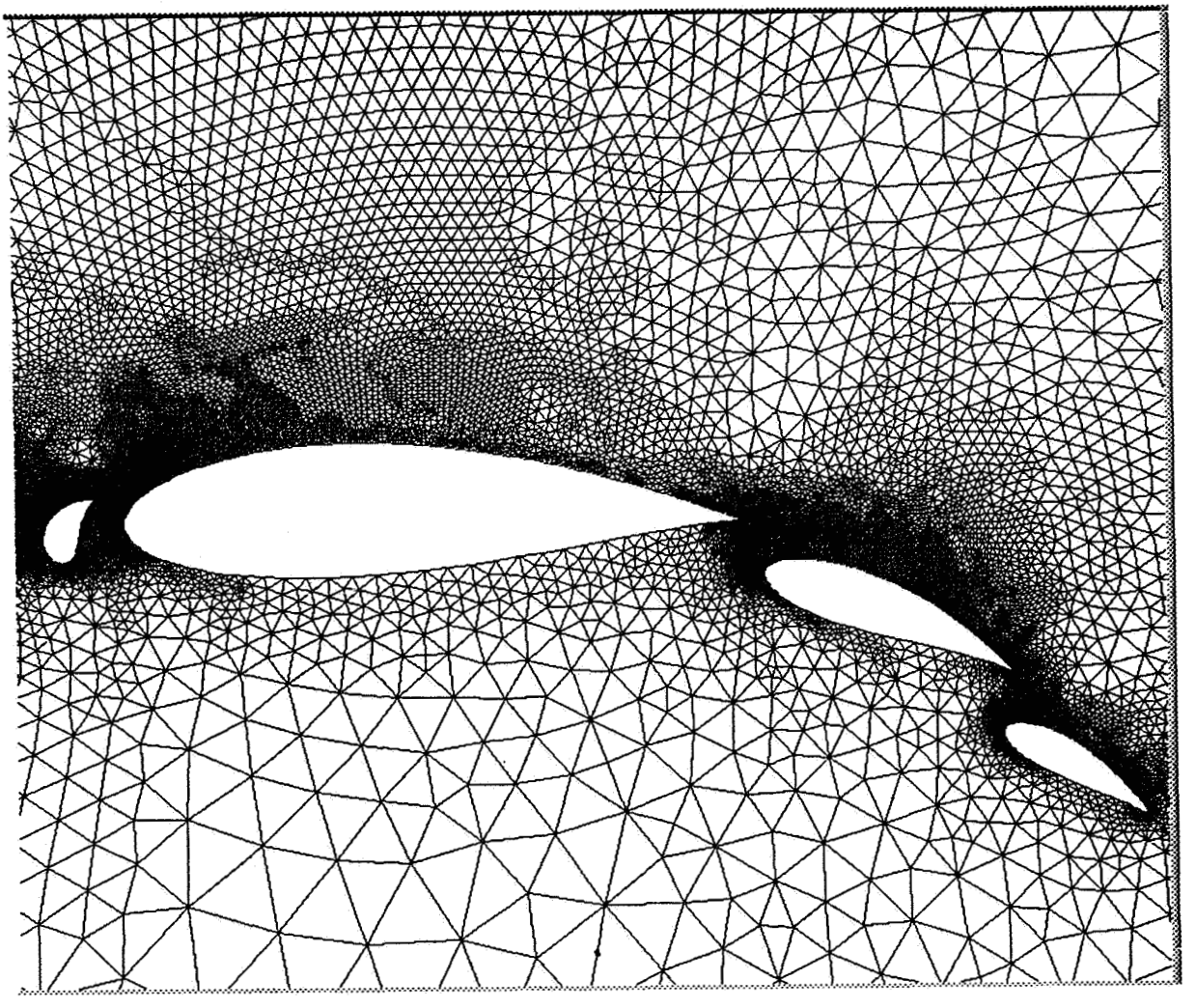


Figure 11: Final Adapted Mesh Employed for Computation of Inviscid Flow over Four Element Airfoil (Mach = 0.2, Incidence = 0 degrees, Number of Points = 22,792)

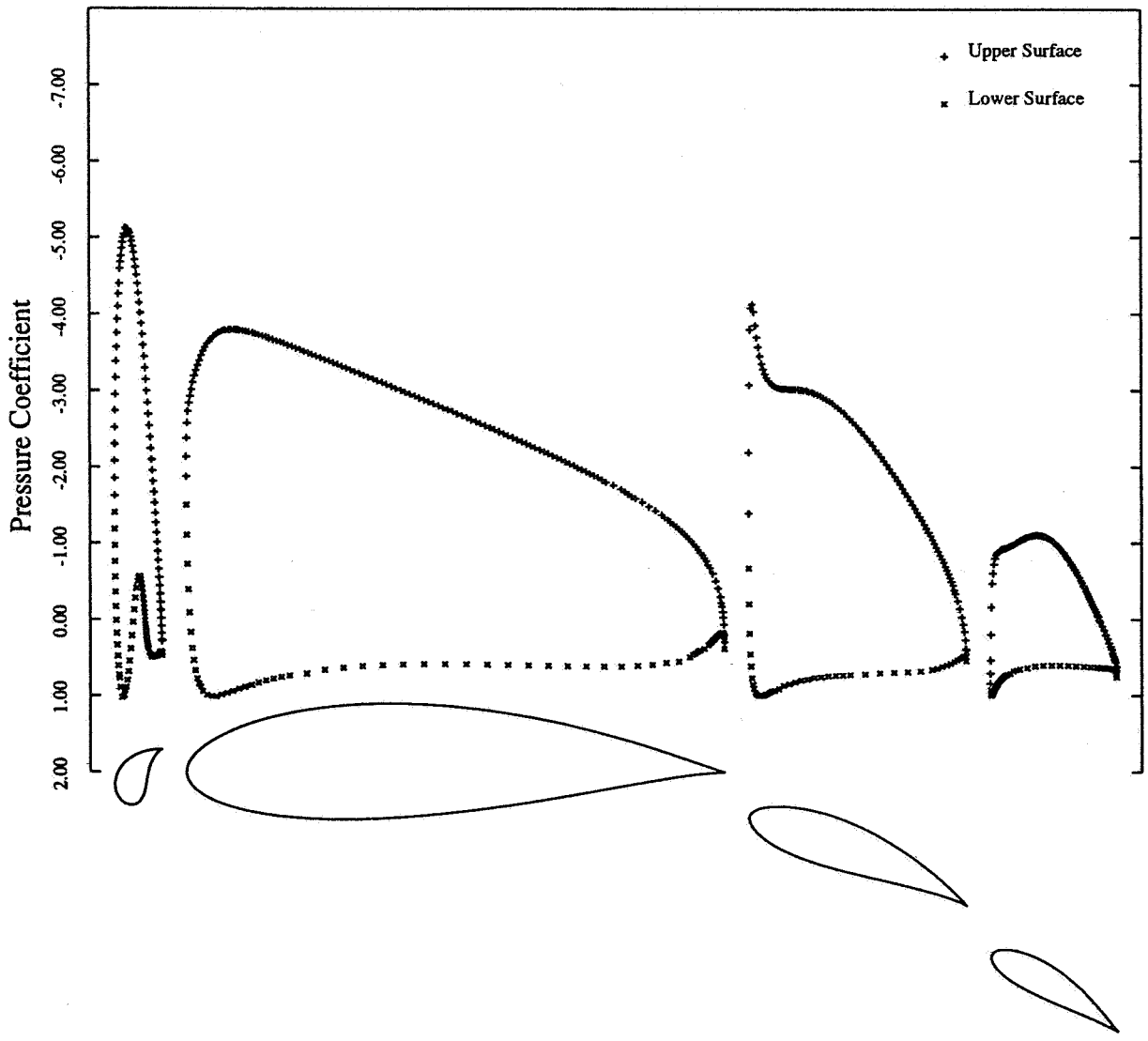


Figure 12: Computed Surface Pressure Distribution for Inviscid Flow over Four Element Airfoil (Mach = 0.2, Incidence = 0 degrees, Number of Points = 22,792)

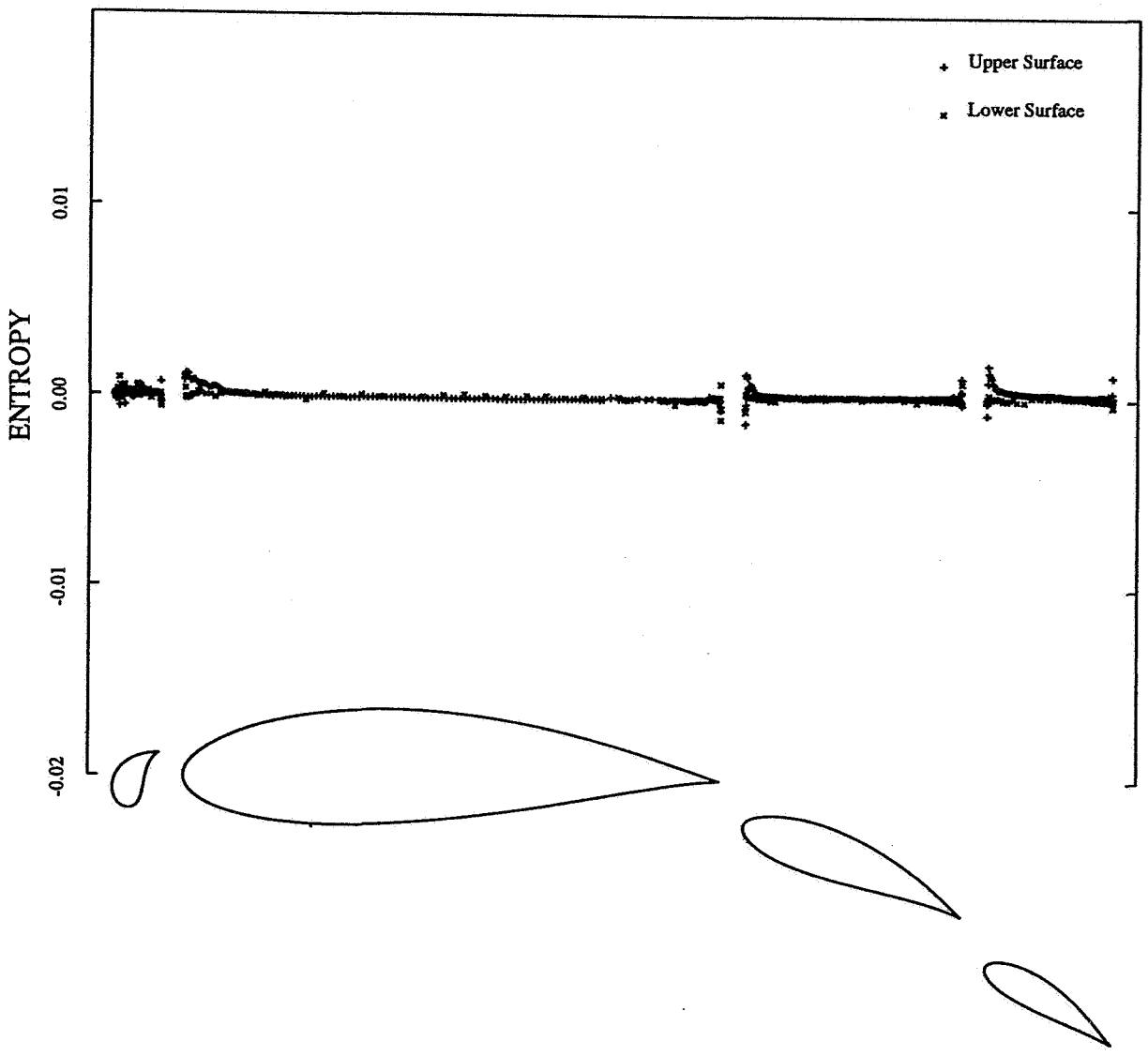


Figure 13: Computed Surface Entropy Distribution for Inviscid Flow over Four Element Airfoil (Mach = 0.2, Incidence = 0 degrees, Number of Points = 22,792)

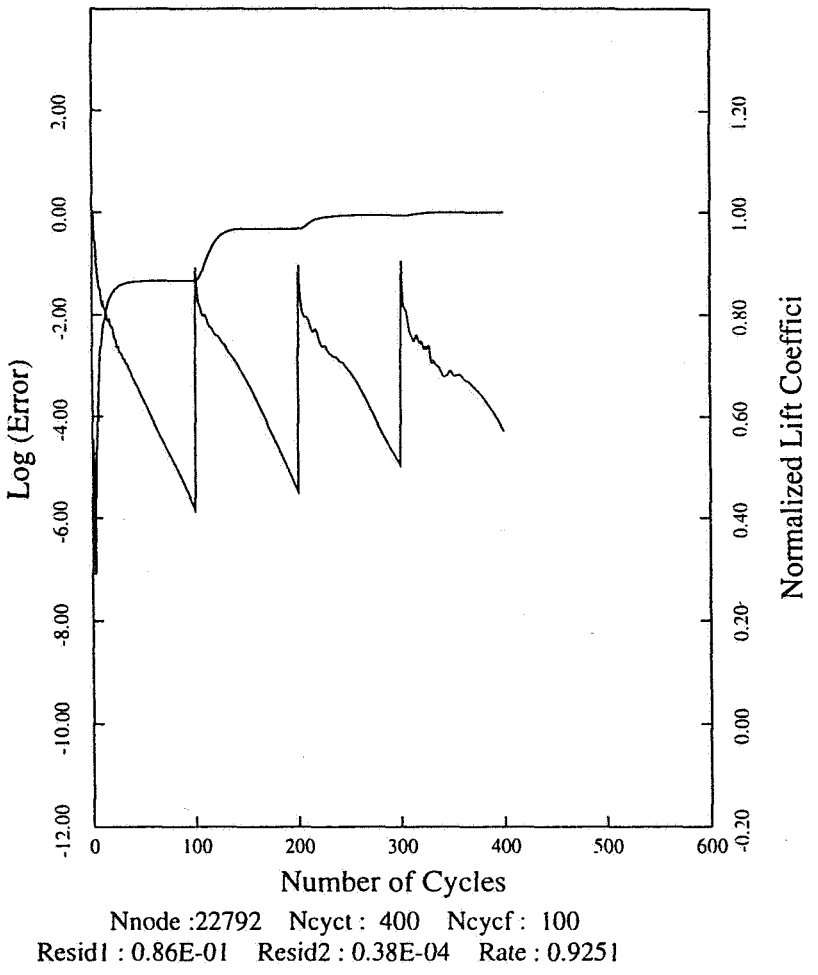


Figure 14: Full Multigrid Convergence Rate on Initial and Three Adapted Meshes for Flow over Four Element Airfoil (Mach = 0.2, Incidence = 0 degrees)

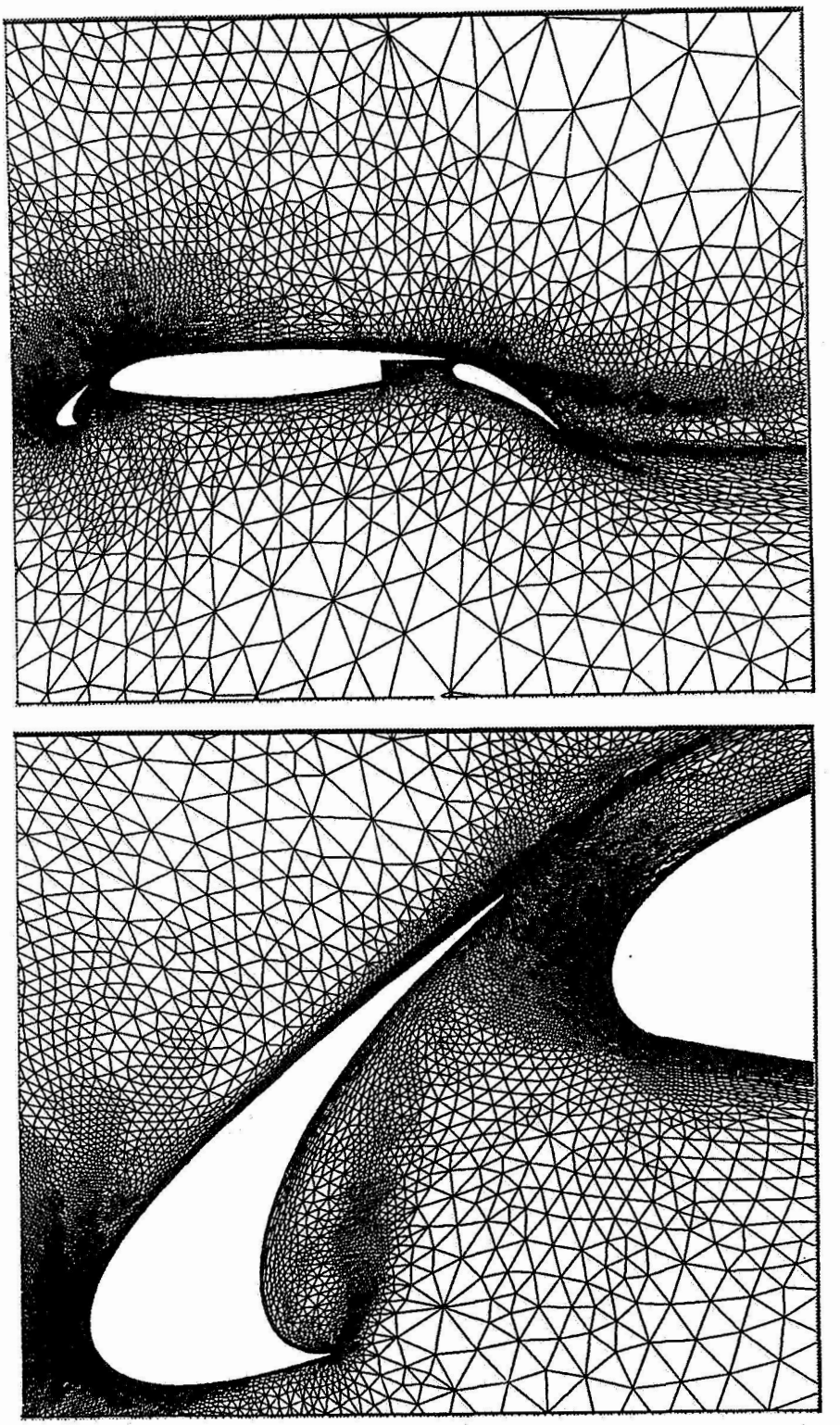


Figure 15: Final Adapted Mesh for Computation of Viscous Turbulent Flow Over Three Element Airfoil (Mach = 0.2, Incidence = 16 degrees, Reynolds Number = 9 million, Number of Points = 120,307)

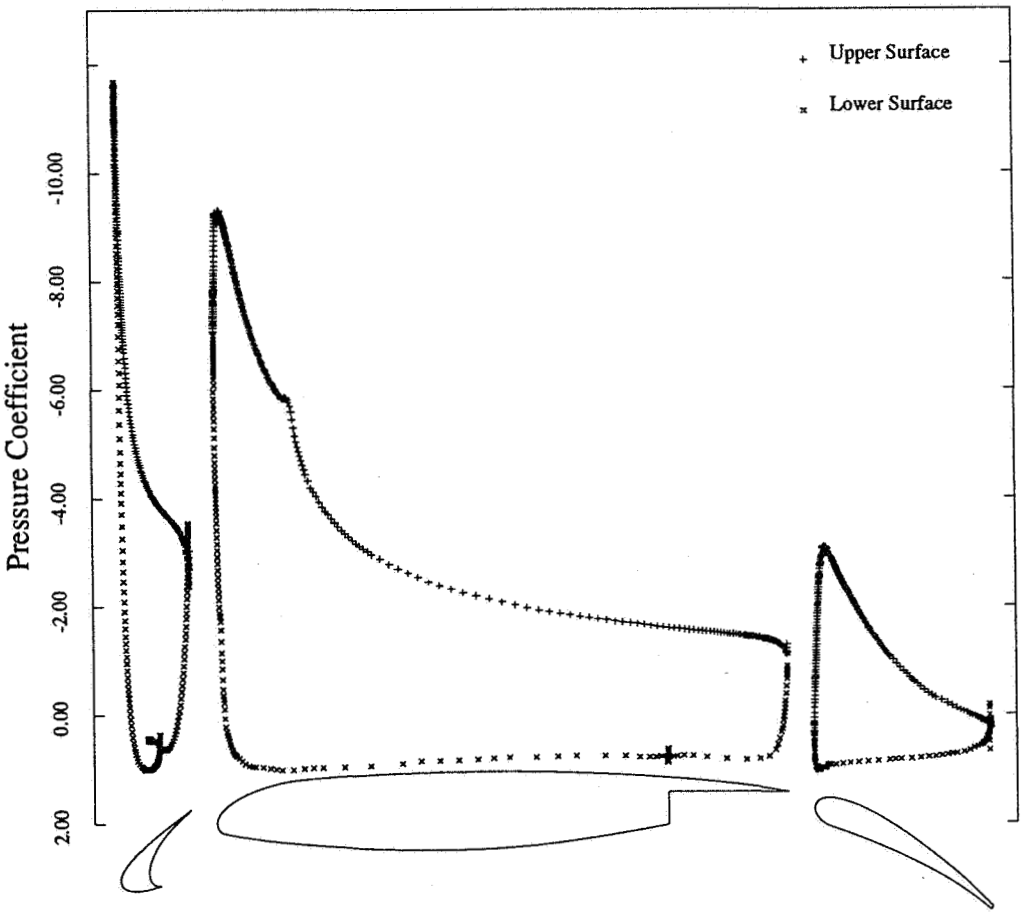
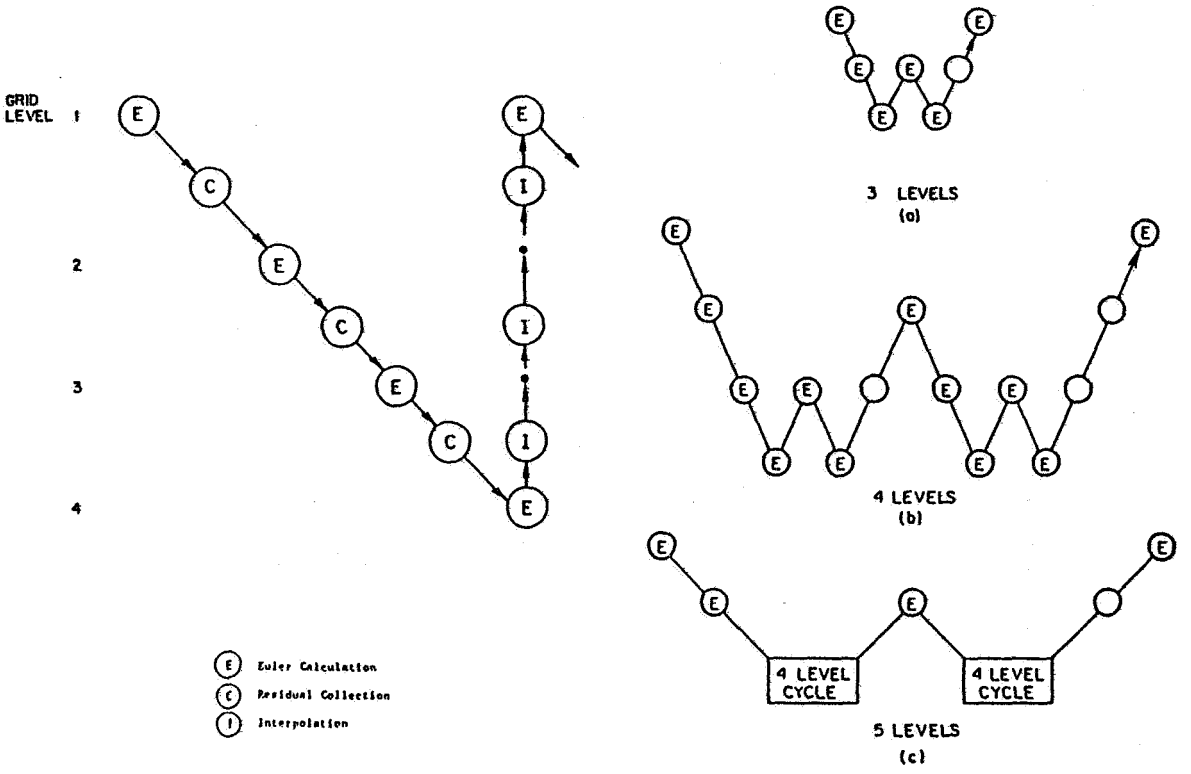


Figure 16: Computed Surface Pressure Distribution on Adapted Mesh for Viscous Turbulent Flow Over Three Element Airfoil (Mach = 0.2, Incidence = 16 degrees, Reynolds Number = 9 million)



Multigrid V-Cycle

Multigrid W-Cycle

Figure 17: Illustration of Multigrid V and W cycles

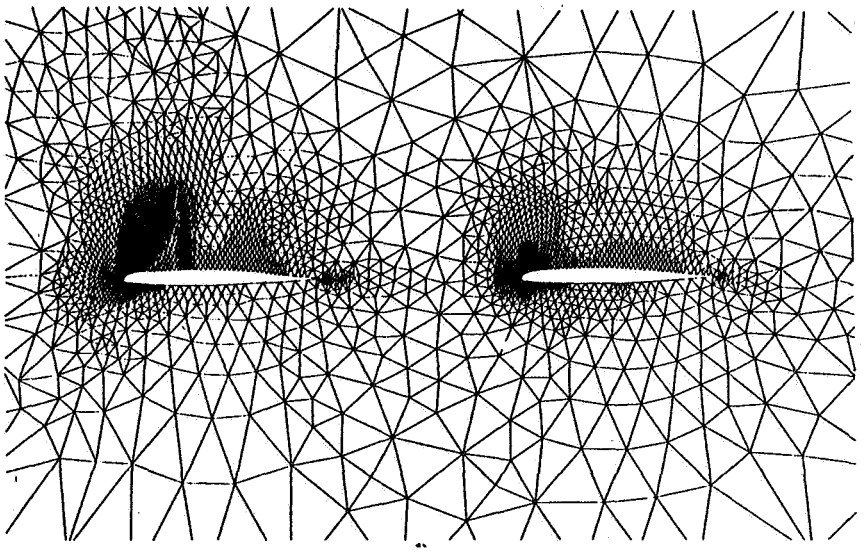


Figure 18: Adapted Mesh Employed for Computation of Flow over Tandem Airfoil Configuration (Mach = 0.7, Incidence = 3 degrees)

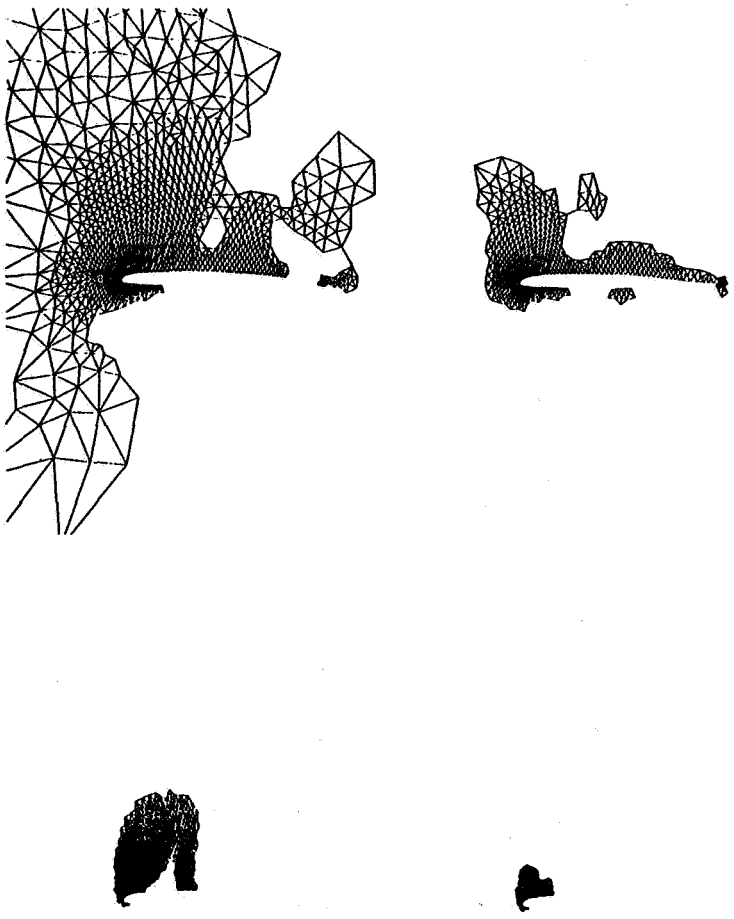


Figure 19: Fifth and Sixth Level Meshes Employed in the Zonal-Fine Grid Scheme for Computation of Flow over Tandem Airfoil Configuration

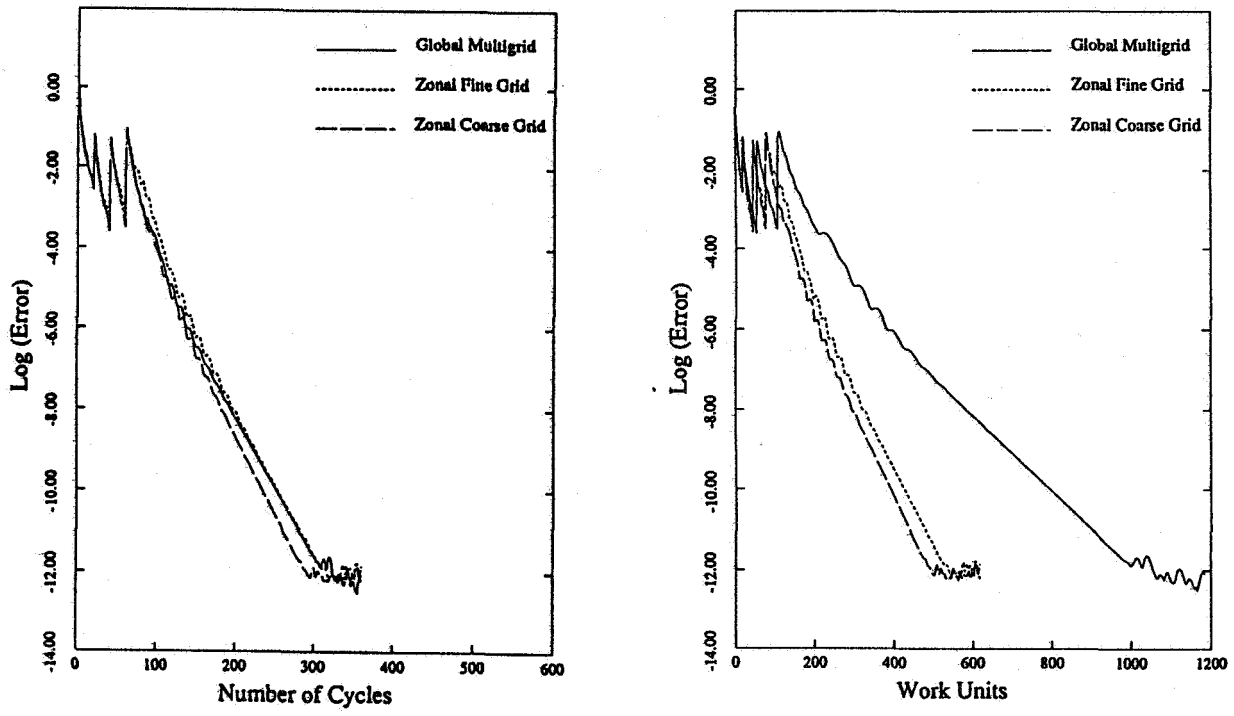


Figure 20: Convergence Rates of Various Multigrid Methods in Terms of Multigrid Cycles and Work Units for Flow over Tandem Airfoil Configuration

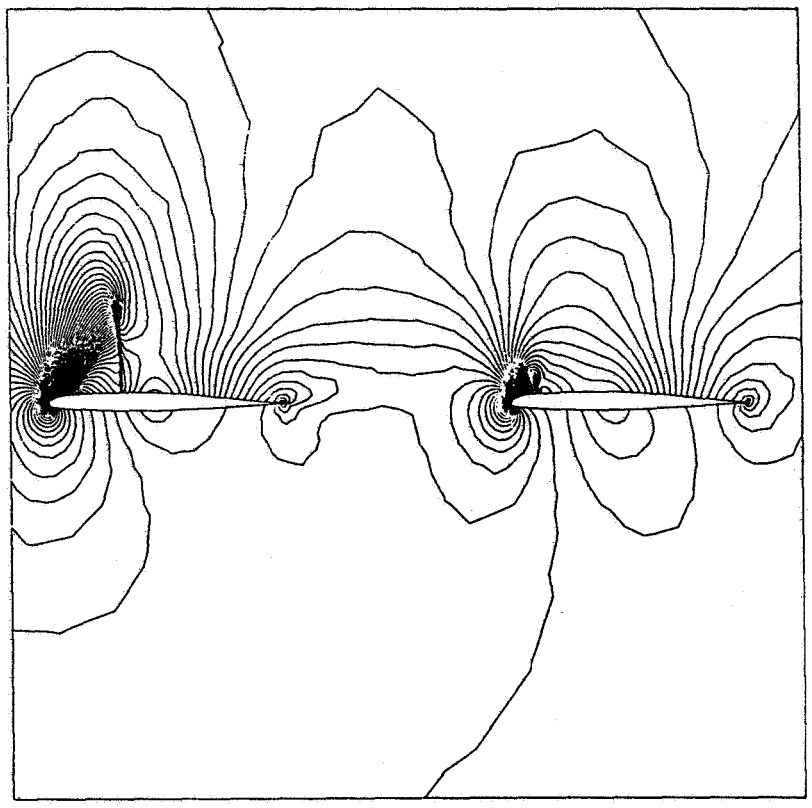


Figure 21: Computed Mach Contours on Adapted Grid for Flow over Tandem Airfoil Configuration

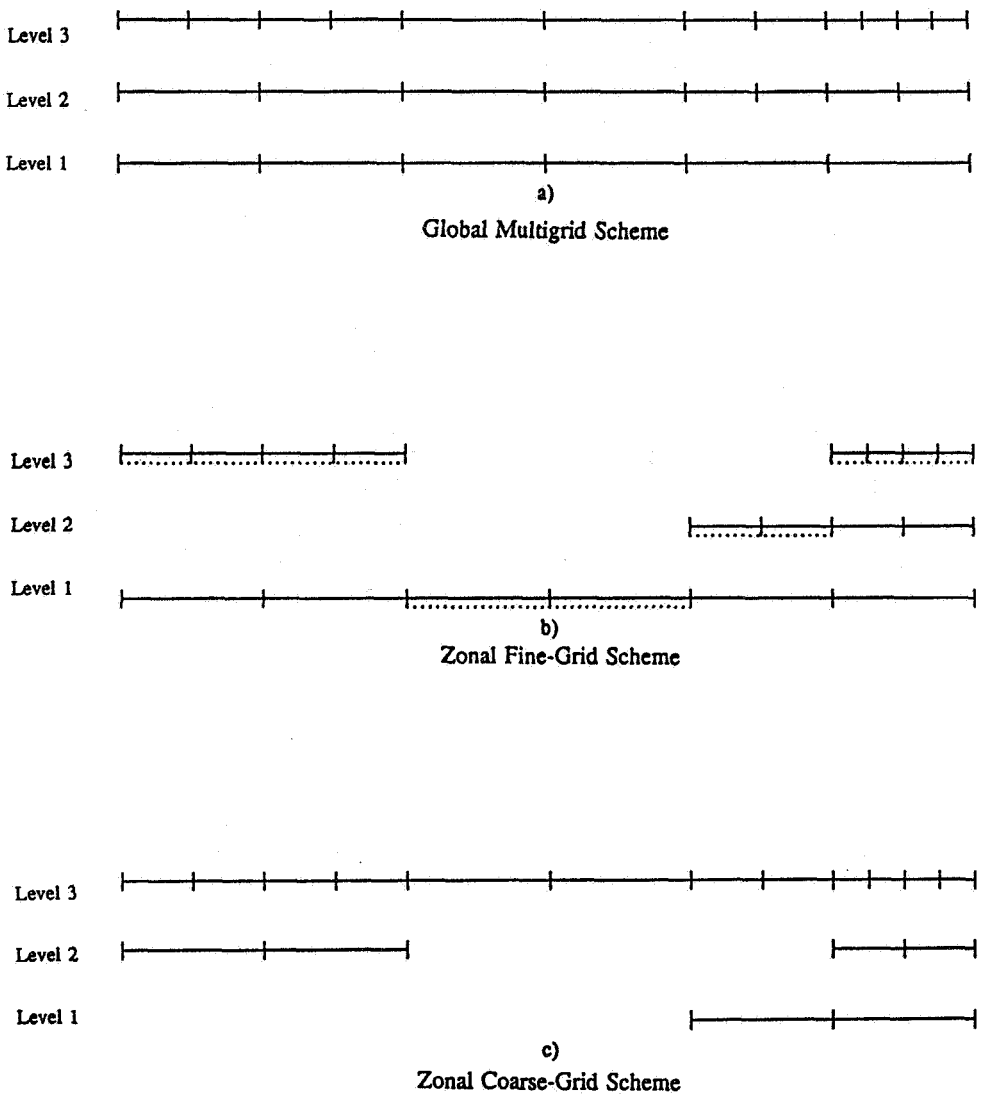


Figure 22: Illustration of the Relationship Between the Zonal-Fine Grid Scheme, the Zonal-Coarse Grid Scheme, and the Global Multigrid Scheme Using Linear One-Dimensional Meshes

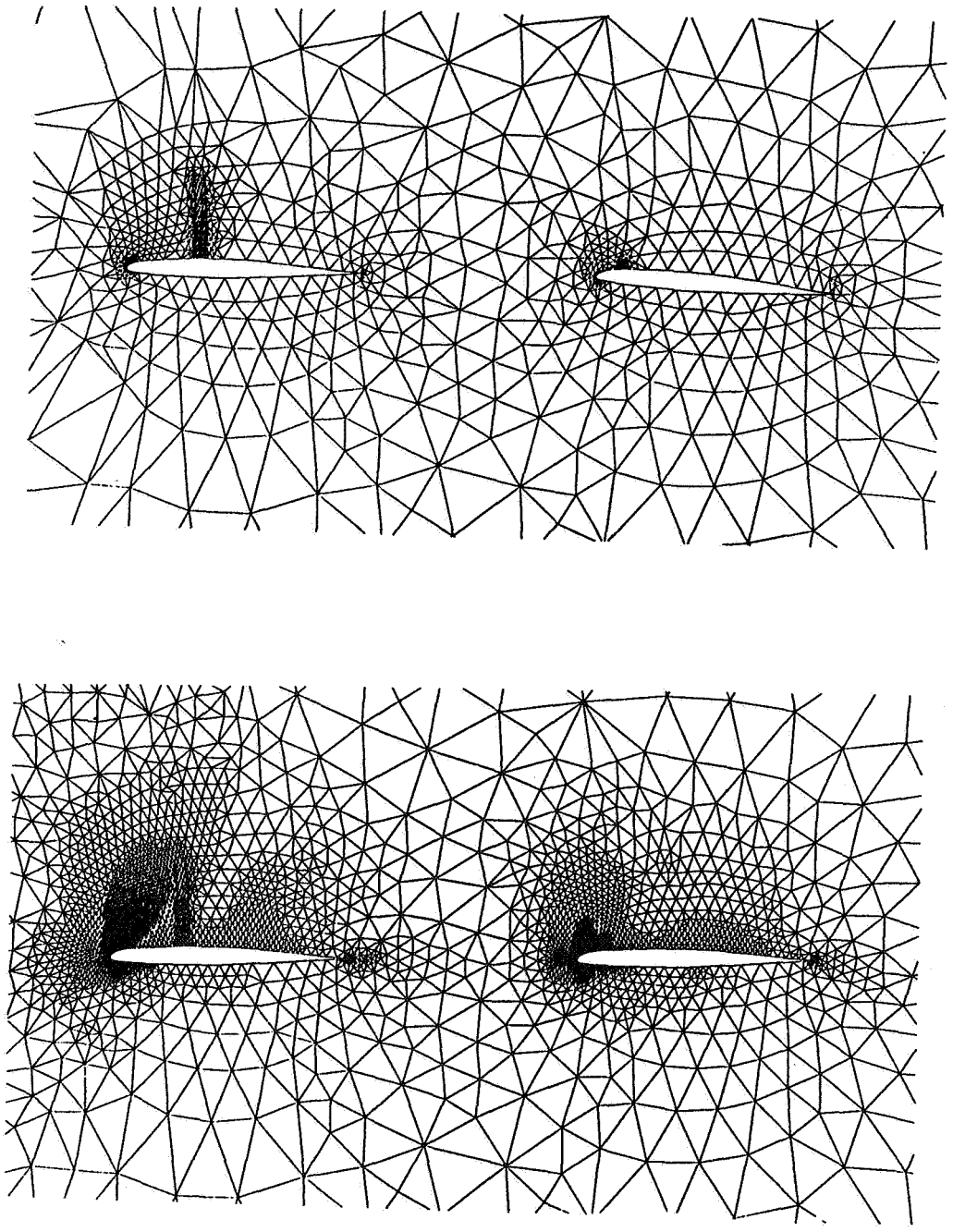


Figure 23: Resulting Coarse Mesh using Two Passes of Aggressive Coarsening on Fine Mesh of Figure 19, and Equivalent Mesh used in Global Multigrid Sequence.

THE ADAPTIVE, CUT-CELL CARTESIAN APPROACH (WARTS AND ALL)

Kenneth G. Powell
The W. M. Keck Foundation CFD Laboratory
Department of Aerospace Engineering
The University of Michigan
Ann Arbor, MI

INTRODUCTION

Solution-adaptive methods based on cutting bodies out of Cartesian grids are gaining popularity now that the ways of circumventing the accuracy problems associated with small cut cells have been developed. Researchers are applying Cartesian-based schemes to a broad class of problems now, and, although there is still development work to be done, it is becoming clearer which problems are best suited to the approach (and which are not). The purpose of this paper is to give a candid assessment, based on applying Cartesian schemes to a variety of problems, of the strengths and weaknesses of the approach as it is currently implemented.

BASIC ELEMENTS OF THE APPROACH

In the adaptive, cut-cell Cartesian approach, as in many adaptive-grid methods, the grid-generation and flow-solution algorithms are strongly linked. The basic pieces of the grid-generation are:

- A cell-based tree data structure;
- A geometry-based adaptive refinement scheme for generating an initial grid;
- A solution-based adaptive refinement/coarsening scheme for generating the final grid.

The basic pieces of the flow-solution algorithm are:

- A limited linear reconstruction scheme;
- A flux function based on an approximate Riemann solver;
- A multi-stage time-stepping scheme.

Each of the basic pieces of the grid-generation and the flow-solution algorithms are described briefly in the following paragraphs. Additional pieces, such as a viscous-term discretization, cell-merging for moving boundary problems, and multigrid acceleration have also been implemented and used in

obtaining results presented in this paper; readers are directed to other papers for details on these techniques.

The data structure

In the work presented in this paper, a cell-based tree data structure is used. In this approach, the grid is represented as a tree in which cells in the grid correspond to nodes of the tree. The root of the tree is a large cell that covers the entire solution domain. This root cell is then divided, yielding a number of children cells which depends on the type of tree being used: two children cells for a binary-tree-based grid; four for a quadtree-based grid; eight for an octree-based grid. In the work presented in this paper, a quadtree data structure was used for the two-dimensional Euler results, a binary-tree data structure was used for the Navier-Stokes results, and an octree data structure was used for the three-dimensional Euler results.

Tree-based structures are well-suited to an adaptive Cartesian-grid approach for several reasons. One reason is that a tree-based structure is memory-efficient; connectivity information relating cells to neighboring cells is unnecessary, as this information can be inferred from the tree structure. Another reason is the ease with which the grid can be adapted: local refinement simply adds children cells to one of the nodes of the tree; local coarsening simply deletes the children cells of one of the nodes of the tree. It should be noted that cell-based trees are not the only data structure suited to the adaptive Cartesian approach. While the current work and the work of the TRANAIR group at Boeing [1] are based on tree structures, the work of Berger [2] and of Quirk [3] are based on local patches of refined grids, with each patch addressed in a structured-grid manner.

Geometry-Based Adaptive Refinement

Unquestionably the primary attraction of the cut-cell Cartesian approach arises from the quest for "hands-off" grid generation. In the geometry-based adaptive-refinement scheme used to generate the initial grids for calculations, both the body-surface discretization and the solution-domain volume discretization are carried out automatically, based on a minimum of user input. The inputs to this step are:

- A length scale for the root cell (typical value 100 chords);
- A maximum length scale for cells that intersect a body (typical value 0.01 chords);
- A maximum angle deviation between successive faces on a body (typical value 5°);
- A minimum length scale of interest (typical value 0.001 chords).

In addition, a location for the centroid of the root cell (typically the origin) is required, as is a spline representation of the bodies in the flow.

The grid-generation algorithm makes use of these input parameters by carrying out the following steps;

1. A root cell is constructed based on the input size and location;

2. The root cell is recursively refined until each body has at least one cell intersecting it;
3. Cells that intersect bodies are recursively refined until the maximum body length-scale constraint is met (or the minimum length-scale is hit);
4. Cells that intersect bodies are recursively refined until the angle-deviation constraint is met (or the minimum length-scale is hit).

This approach is robust, and simple to code, although it relies on recursion. Most of the time is spent querying the spline representations of the bodies, and computing intersections of cells in the grid with the splines. Efficient and robust coding of those intersection calculations is extremely important. Quirk [4] uses integer arithmetic to compute these intersections, which guarantees robustness and can resolve arbitrarily fine geometric details.

Solution-Based Adaptive Refinement

Solution-based adaptation is carried out by flagging cells for refinement or coarsening based on heuristic criteria. The criteria used in this work are tuned to capturing regions in which compressibility and/or vorticity are appreciable, and are scaled in the manner suggested by Warren et al [5] so as to avoid over-refining high-gradient regions at the cost of smooth regions. The criteria are

$$\tau_c = |\nabla \cdot \mathbf{u}|\, h_i^{1.5} \qquad \sigma_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \tau_{c_i}^2} \tag{1}$$

$$\tau_v = |\nabla \times \mathbf{u}|\, h_i^{1.5} \qquad \sigma_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \tau_{v_i}^2} \tag{2}$$

where n is the number of cells in the grid, and h_i is a length-scale associated with cell i. Based on these criteria, cells are flagged for refinement if

$$|\tau_c| > \sigma_c \text{ or } |\tau_v| > \sigma_v \tag{3}$$

and flagged for coarsening if

$$|\tau_c| < \frac{1}{10}\sigma_c \text{ and } |\tau_v| < \frac{1}{10}\sigma_v \,. \tag{4}$$

Limited Linear Reconstruction

In order for the scheme to be more than first-order accurate, a local reconstruction must be done; in order for the scheme to yield oscillation-free results, the reconstruction must be limited. The limited linear reconstruction here is due to Barth [6]. A least-squares gradient is calculated, using neighboring cells, by locally solving the following non-square system for the gradient of the primitive variable vector W by a least-squares approach

$$\mathcal{L}\nabla W^{(k)} = f \tag{5}$$

$$\mathcal{L} = \begin{pmatrix} \Delta x_1 & \Delta y_1 \\ \vdots & \vdots \\ \Delta x_N & \Delta y_N \end{pmatrix} \qquad f = \begin{pmatrix} \Delta W_1^{(k)} \\ \vdots \\ \Delta W_N^{(k)} \end{pmatrix} \tag{6}$$

where

$$\Delta x_i = x_i - x_0 \quad (7)$$

$$\Delta y_i = y_i - y_0 \quad (8)$$

$$\Delta u_i = u_i - u_0 \quad (9)$$

and the points are numbered so that 0 is cell in which the gradient is being calculated, and i is one of N neighboring cells used in the reconstruction. A cell-based minmod-type limiter is implemented by reconstructing the solution as

$$\mathbf{W}(\mathbf{x}) = \bar{\mathbf{W}} + \phi(\mathbf{x} - \bar{\mathbf{x}}) \cdot \nabla \mathbf{W} \quad (1)$$

where ϕ is given by

$$\phi = \min \begin{cases} 1 \\ \min_k \left(\frac{\left| \bar{W}^{(k)} - \max_{neighbors}\left(W^{(k)}\right) \right|}{\left| \bar{W}^{(k)} - \max_{cell}\left(W^{(k)}\right) \right|} \right) \\ \min_k \left(\frac{\left| \bar{W}^{(k)} - \min_{neighbors}\left(W^{(k)}\right) \right|}{\left| \bar{W}^{(k)} - \min_{cell}\left(W^{(k)}\right) \right|} \right) \end{cases} \quad (2)$$

Flux Function

The flux function used is Roe's approximate Riemann solver, described in more detail in many references, including [7]. The inputs to the flux function are the left and right states resulting from the limited reconstruction step described above.

Time-Stepping

A multi-stage scheme is used to advance the solution in time (in the unsteady calculations) or to a steady state (in the steady calculations). Local time stepping is used in the steady calculations; a cell-merging procedure is used in the unsteady calculations.

RESULTS AND OUTLOOK FOR VARIOUS CLASSES OF FLOWS

Steady, Inviscid Flows

The original drawbacks to cut-cell Cartesian approaches for solving the Euler equations were two-fold: stability and accuracy problems associated with the small cut cells; and resolution problems due to the regularity of Cartesian grids. With the introduction of local refinement and coarsening, the resolution problem for these flows is basically solved for two-dimensional flows. For three-dimensional flows, resolution problems remain, as will be described later in this section.

The accuracy and stability problems associated with the small cut cells that occur in the Cartesian approach have been the topic of several papers. Several techniques have been developed, all of which

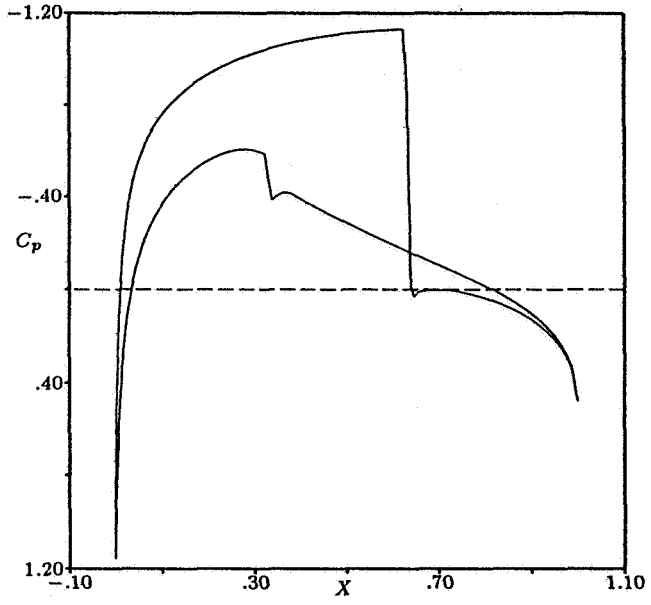


Figure 1: AGARD 01 Benchmark Case — C_p

alleviate the problems. Berger and Leveque [8] use a wave-propagation-based technique that allows the time step in small cells to be computed from a CFD criterion based on the entire uncut cell. Chern and Colella [9] use a simpler but less accurate technique based on similar concepts. DeZeeuw and Powell [7] treat the mesh with cut cells as an unstructured mesh, and use linear reconstruction to ensure accuracy and local time stepping to ensure stability in steady flows. For unsteady flows, Quirk [4] and Bayyuk, Powell and Van Leer [10] use a cell-merging technique that alleviates the small time-step problem, and also lends itself well to computing flows in which the boundaries are moving.

Results for the AGARD-1 and AGARD-3 benchmark cases are shown in Figures 1–7. The first was run on a grid of 11,366 cells; the second on a grid of 15,234 cells. As can be seen, the adaptation criteria capture the shocks and wakes, without overresolving other flow regions. The C_p on the wing for the first case, and the Mach number on the wing for the second case, show that the cut cells on the wing do not lead to non-smooth values of the flow variables there.

Results for the four-element Suddhoo-Hall benchmark case, are shown in the paper by Coirier and Powell in this volume. The comparison of computed and exact C_p show that, once sufficient adaptation has been done to resolve the flow features, the solution is smooth and matches the analytical solution well.

Results for the "Jameson non-unique" benchmark case are shown in Figures 8–11. The grid, shown in Figure 8, was used for all of the calculations; it has 13,613 cells. The free-stream Mach number was set at $M_\infty = 0.78$, and the angle of attack was varied incrementally, converging the code to machine zero residuals at each angle of attack. The hysteresis effect is shown in Figure 9; the two solutions obtained at $\alpha = -0.45°$ are shown in Figures 10 (obtained by decreasing the angle of attack incrementally) and 11 (obtained by increasing the angle of attack incrementally). While the two solutions highly resemble those originally obtained by Jameson, and the hysteresis effect is evident, the range of M and α in which the solution was non-unique was different from that reported

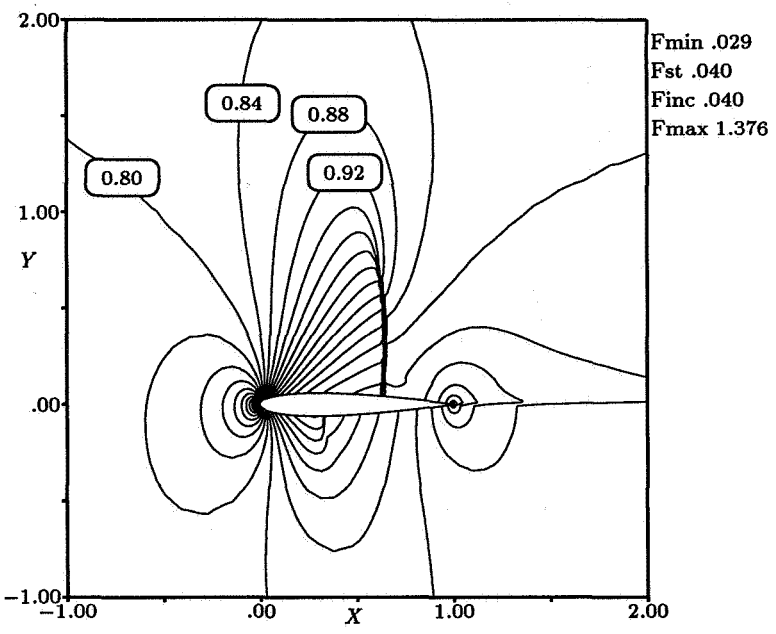


Figure 2: AGARD 01 Benchmark Case — M contours

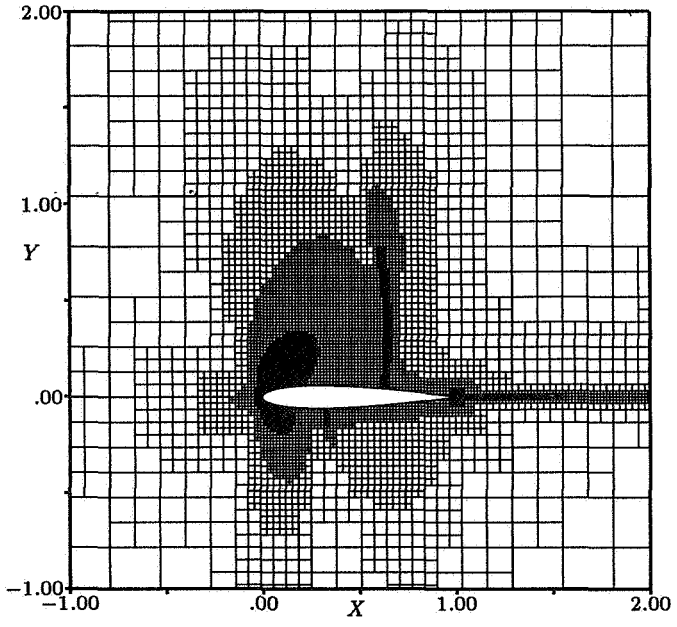


Figure 3: AGARD 01 Benchmark Case — Grid

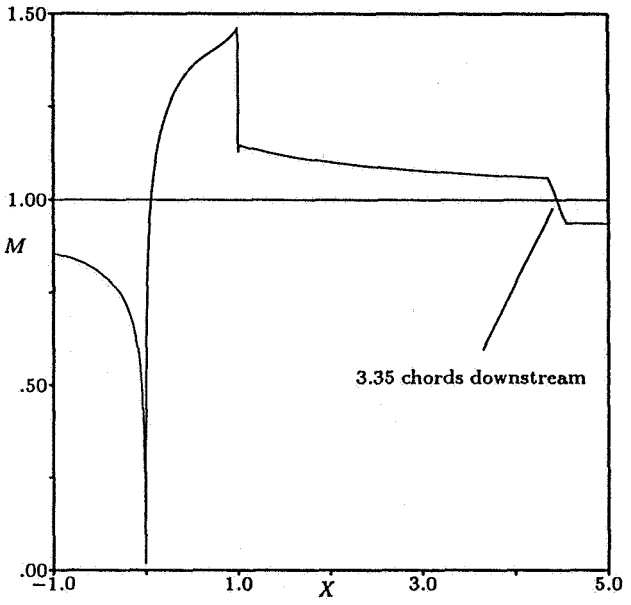


Figure 4: AGARD 03 Benchmark Case — Mach Number on axis

by Jameson. Running this case raised more questions than it answered for us. Further studies of the sensitivity of the non-unique range to computational parameters will have to be carried out before anything definitive can be said about this airfoil.

Extension of the two-dimensional scheme above to three dimensions adds complexity to the geometric algorithms (such as calculating the locations at which the cells are cut by the bodies) but adds very little to the flow solver. It is in the three-dimensional cases that the advantages of the automated grid-generation procedure are truly seen. Preliminary results for the double-ellipsoid benchmark case are shown in Figures 12–14.

The simplicity of the double-ellipsoid geometry hides an important inefficiency of the three-dimensional Cartesian-based scheme, however. For problems in which, due to the geometry, gradients are higher in one direction than another, the cost of resolving the one direction is the over-resolution of the other direction. An example is a high-$\mathcal{AR}$ wing: resolving the chordwise direction well leads to gross over-resolution of the spanwise direction, due to the isotropic nature of the grid refinement. Allowing directional refinement will in general not help; for instance, a swept-back high-$\mathcal{AR}$ wing require a large number of cells to resolve, regardless of the refinement approach used.

Memory and CPU usage for the steady-flow codes are presentied in Table 1. The times reported are for one single-grid iteration of a four-stage time-stepping scheme. The data structure used for the cut cells in the 3D code is a preliminary one; a code with lower memory overhead could be easily implemented.

With the incorporation of local refinement/coarsening into Cartesian codes, and with any of the various solutions to the small-cell problem, adaptive cut-cell Cartesian codes have reached a point where they can compete favorably with other approaches for two-dimensional steady, inviscid flows. The grid-generation is as automatic as totally unstructured approaches, and in some cases more automatic, since the surface discretization of the boundaries is carried out at the same time as the

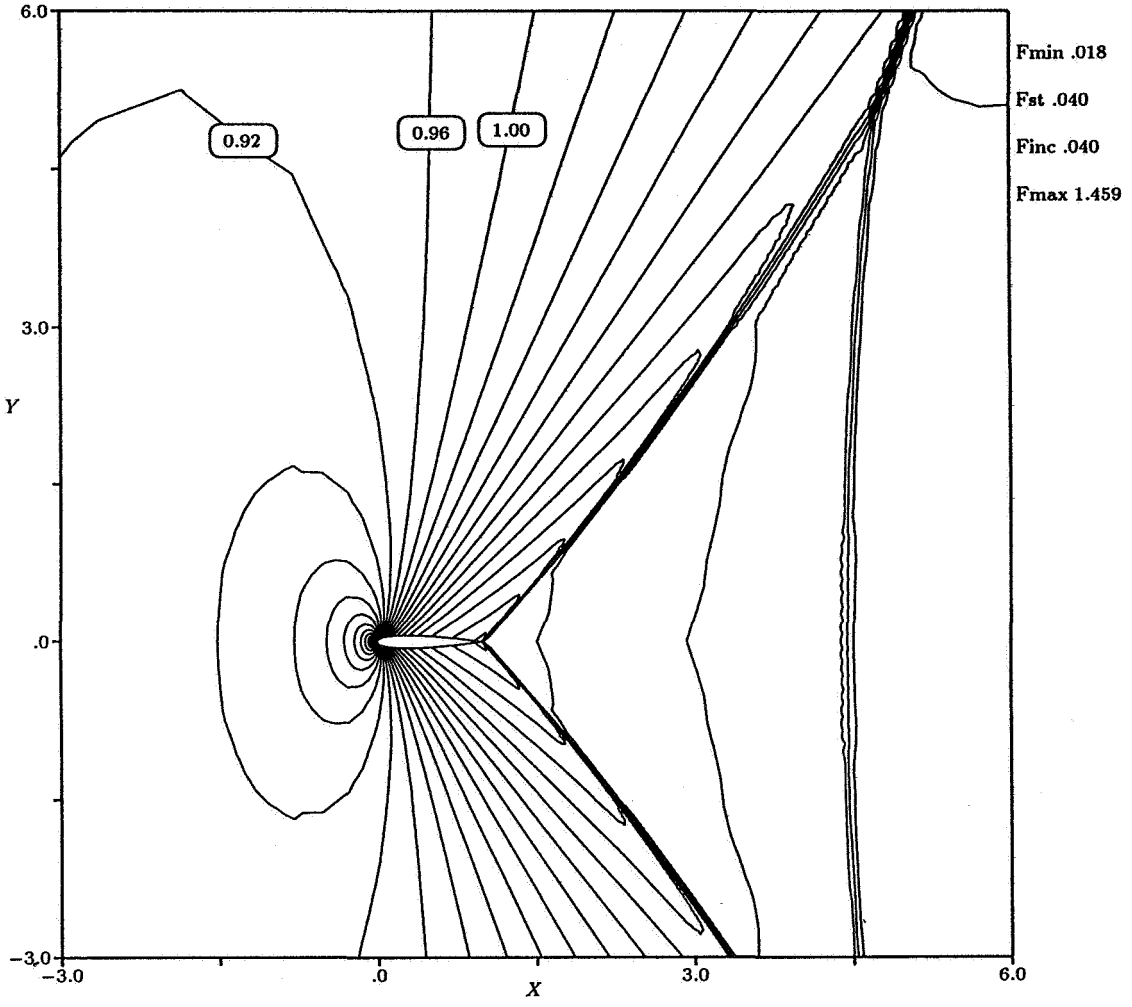


Figure 5: AGARD 03 Benchmark Case — M contours

	Memory Usage	CPU usage (μseconds/cell/iteration)
2D	(30 reals + 8 ints)*nCells + (2 reals + 3 ints)*nCutCells	370 (HP 735/99)
3D	(35 reals + 16 ints)*nCells + (38 reals + 35 ints)*nCutCells	470 (IBM RS 6000/590)

Table 1: Memory and CPU usage for Cut-Cell Cartesian Euler Codes (steady)

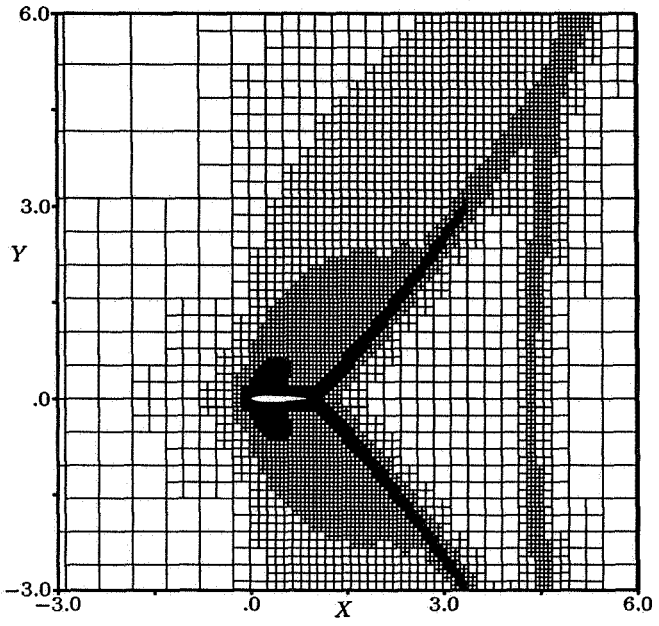


Figure 6: AGARD 03 Benchmark Case — Grid

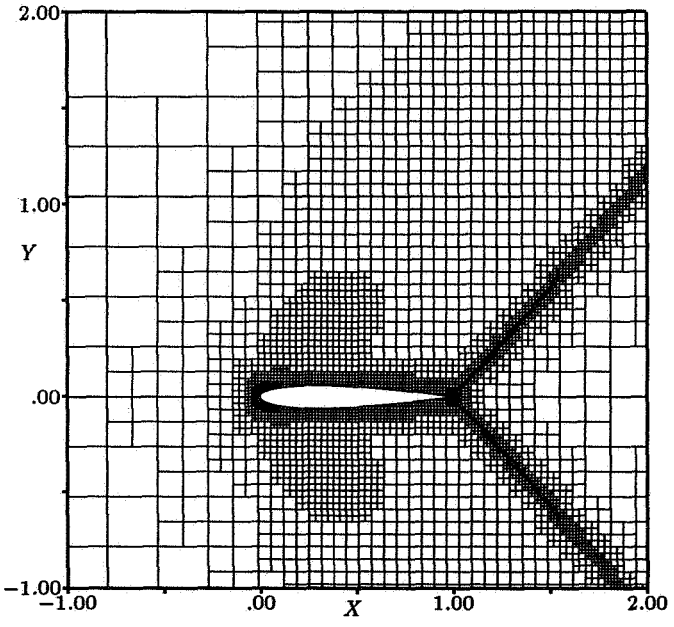


Figure 7: AGARD 03 Benchmark Case — Grid Close-up

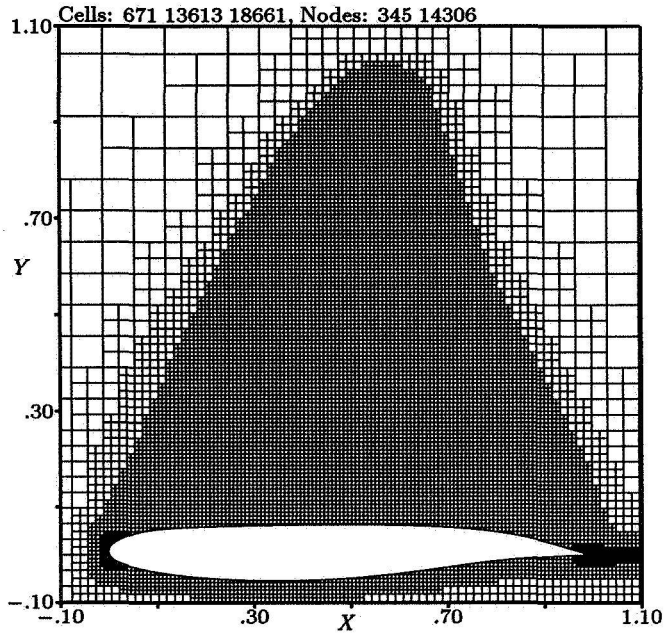


Figure 8: Jameson Airfoil Benchmark Case — Grid

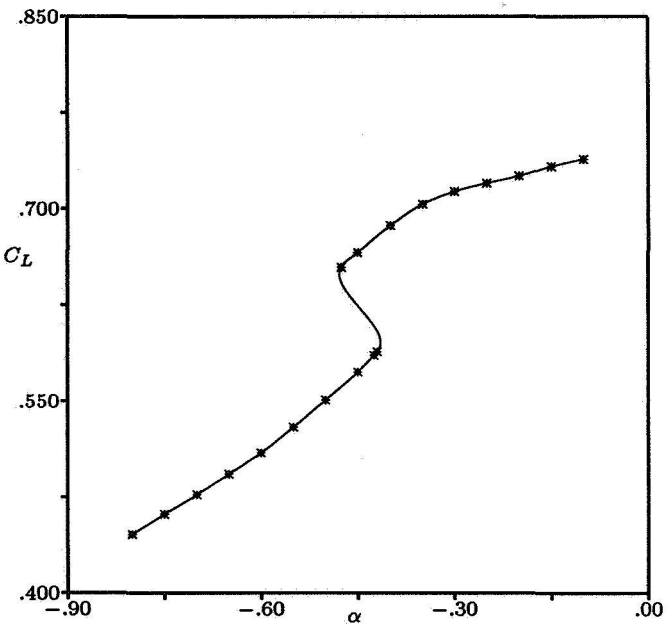


Figure 9: Jameson Airfoil Benchmark Case — C_L versus α curve

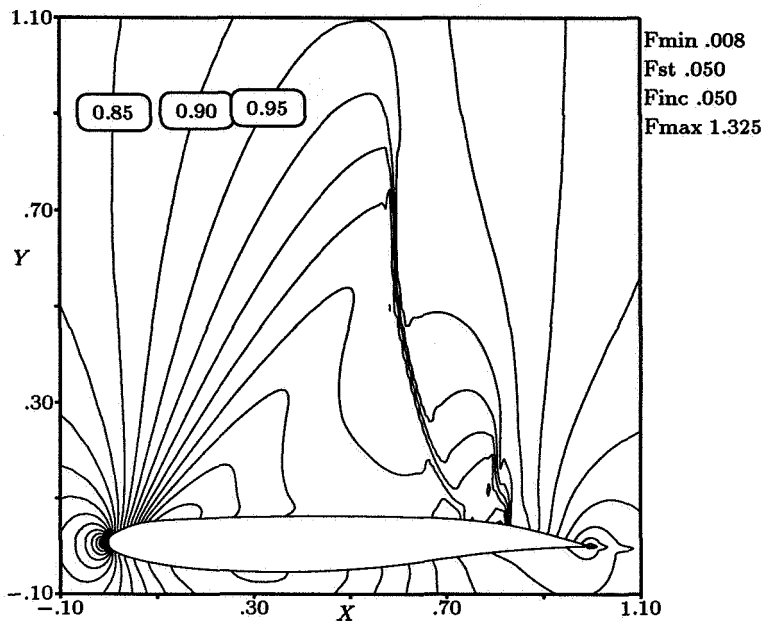


Figure 10: Jameson Airfoil Benchmark Case — Mach contours

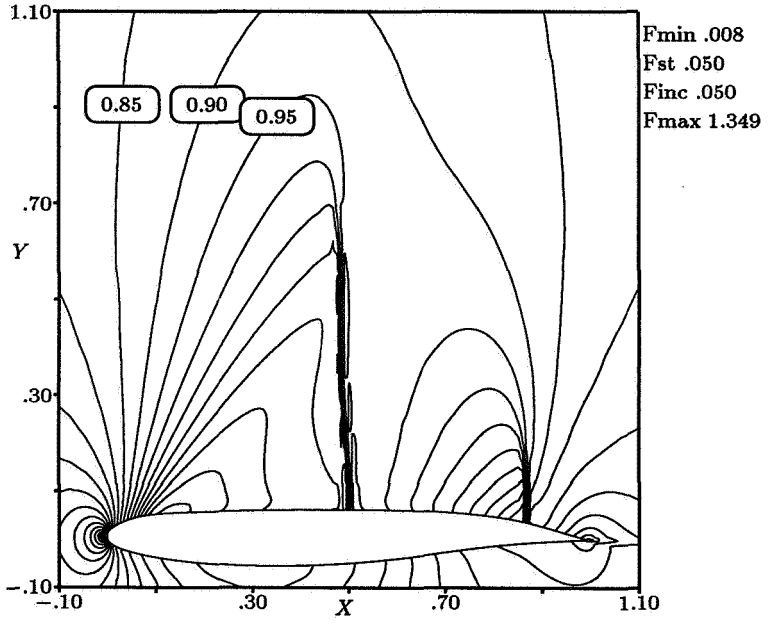


Figure 11: Jameson Airfoil Benchmark Case — Mach contours

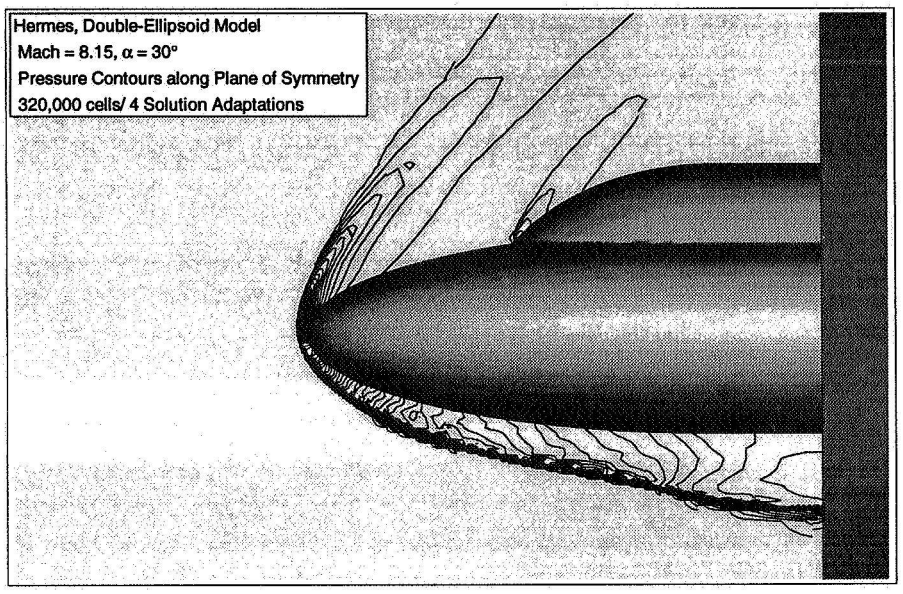


Figure 12: Solution for Hermes Problem

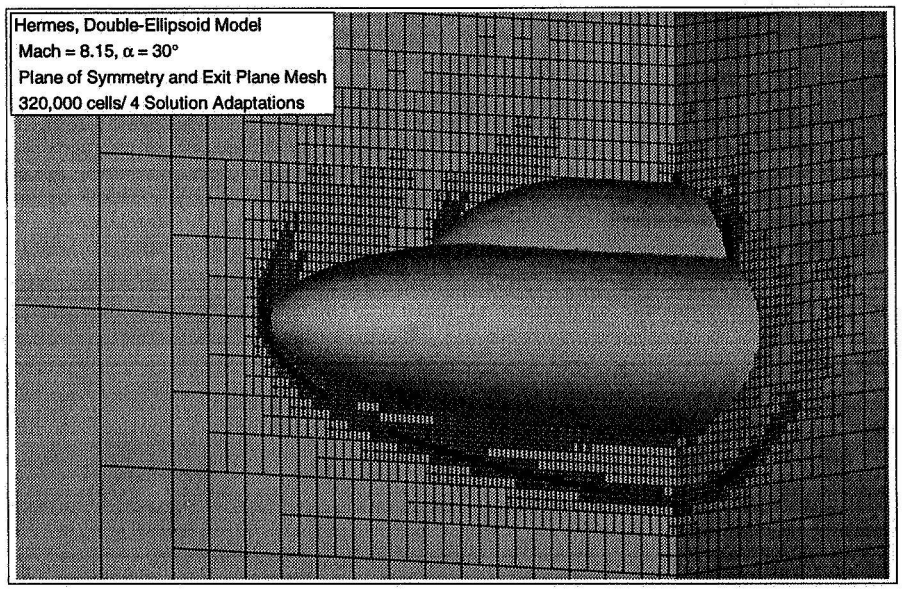


Figure 13: Grid for Hermes Problem

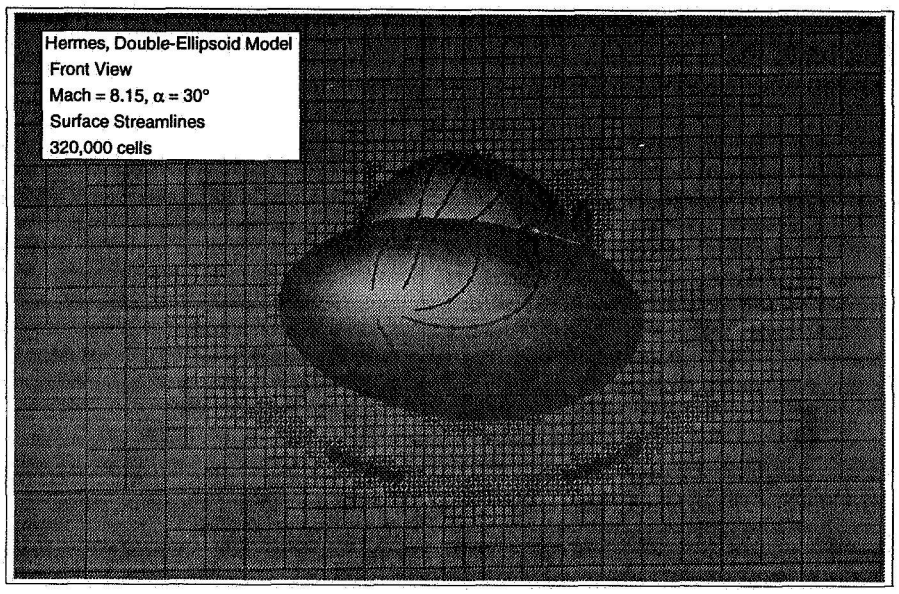


Figure 14: Streamlines for Hermes Problem

volume discretization of the solution domain. Typically, approximately 50% more cells are required to get the same accuracy out of a Cartesian scheme as on a body-conforming grid (see, for example [11]).

In three dimensions, the payoff of automated grid generation is even greater. However, since the Cartesian approach is geared towards solving isotropic problems, many geometries are difficult to resolve properly without an extremely high number of cells. This trade-off of ease of grid generation versus efficient use of computational resources is one that is almost certainly worthwhile in the early stages of a design process; very high-caliber calculations for use in detailed analysis of three-dimensional flows will probably always have to be done on body-conforming grids.

Viscous Flows

The difficulties in applying Cartesian-based schemes to viscous flows are detailed in the paper by Coirier and Powell in this volume. The two fundamental issues are:

- The difficulty in defining a viscous discretization that is positive and consistent (let alone accurate) on the very non-smooth meshes produced by the cut-cell Cartesian approach;
- The inherent inefficiency in isotropic refinment of one cell into four for highly anisotropic (e.g. high-Reynolds-number) problems.

It is interesting to note that the first problem is most acute when the Reynolds number is low; the second is most acute when the Reynods number is high. At moderate Reynolds number, a stable, reasonably accurate scheme that makes reasonably efficient use of the computational resources can be constructed. Flow quantities such as density, pressure and velocities can be obtained to good accuracy; derivative quantities such as pressure gradient and C_f can not.

Taking these issues into account, Cartesian Navier-Stokes codes will probably never play more than a very preliminary design role in low-to-moderate Reynolds number flows, and probably have no role in high-Re flows. For high Reynolds number flows, the best route is a viscous-inviscid coupling approach, in which a Cartesian Euler code is coupled to an integral or finite-difference boundary-layer code (this approach is currently under development in collaboration with Marsha Berger). Another approach, that will require much more development but seems very promising, is to rewrite the Navier-Stokes equations as a first-order hyperbolic system, and solve the resulting equations by methods similar to those used for the Euler equations [12].

Unsteady Flows

The ability of the Cartesian approach to model shock physics accurately and efficiently has been shown most impressively by Berger and Colella [13], Berger and Leveque [8] and Quirk [4, 3]. One very promising arena for Cartesian approaches is that of problems with moving boundaries. In this approach, the boundary is "cut" from the Cartesian mesh at each time step. An unsteady Euler solver is implemented, that accounts for the changing areas of the cut cells of the mesh. In order to alleviate the small time-step problem, and the problems occuring when the body covers or uncovers a cell in one time step, a merging procedure is used [4, 10].

Results from an idealized inlet case are presented in Figures 15–17. This case is a time-accurate simulation of an evolving flow pattern. The evolution is induced by a continuous and smooth deformation of boundaries resembling a supersonic inlet. The figures show the Mach number contours and the associated grids at three points during the geometric excursion. The inflow Mach number is 2.54 throughout. Bewteen the first and second positions, the inlet sides open up at the back and the front and rear edges of these sides become more tapered. The sides also move apart and the spike moves forward and contracts in the vertical direction. The effect of this change on the flow-field can clearly be seen. Between the second and third positions, the spike decreases in length and its geometry changes as shown in the figures. Also the sides flatten and they move towards each other. The final flow pattern achieves the required objective of the inlet (decelaration of the flow to approximately Mach 1.4 with minimal losses in stagnation pressure). The location and shape of the rear of the spike in the third position is critical: the impinging reflected shock must fall in a region of increasing cross-sectional area, otherwise shock system will be disgorged. The number of cells in the mesh ranges from 17,000 at the initial time to 83,000 at the final time.

Results from a case of a body expelled from a high-pressure chamber are presented in Figures 18 and 19. In this case, the two rigid bodies and the gas are all initially stationary. The gas in the enclosure of the larger body has a density and a pressure ten times those of the outside gas. The boundary separating the high-pressure gas from the low-pressure gas is initially half-way down the channel in the larger body and coincides with a plane of symmetry of the smaller body. The simulation shows the evolution of the flow and the motion of the bodies as the compressed gas flows through the channel. The motion of each body is computed by integrating the accelaration due to the net (inviscid) aerodynamic force acting on it. The grid begins with 17,000 cells; that number has increased to 175,000 by the final time.

The stretching and shearing that a body-conforming mesh would undergo during these calculations would necessitate frequent remeshing. In some sense, in the Cartesian calculation remeshing is carried out every time step; this procedure is inexpensive in the Cartesian case, however.

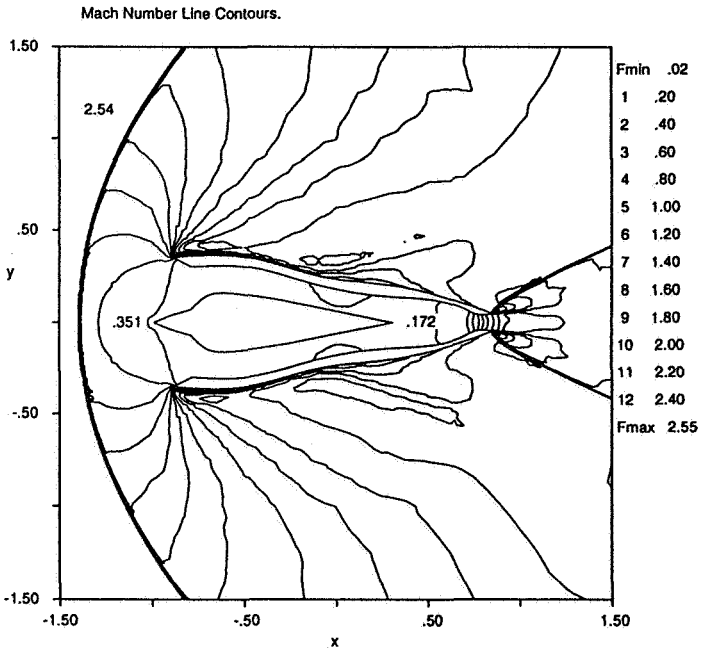


Figure 15: Mach Contours for Inlet Problem — Initial Time

	Memory Usage	CPU usage (μseconds/cell/iteration)
2D	(15 reals + 11 ints)*nCells	300 (2 stage scheme, HP 735/99)

Table 2: Memory and CPU usage for Cut-Cell Cartesian Euler Code (unsteady)

Memory and CPU usage for the unsteady code are reported in Table 2.

THE BOTTOM LINE

The siren-song of totally automated grid-generation has lured a growing number of researchers to Cartesian-based schemes.

Two-dimensional Euler solvers, both steady and unsteady, have reached a level of sophistication and maturity comparable to that of structured quadrilateral- and unstructured triangular-mesh schemes. For these problems, the Cartesian approach is competitive with the best structured and unstructured schemes. Grid generation is automated, user input to the codes is minimal, and the only penalty is that more cells must be used than in a more traditional approach.

One set of problems for which the Cartesian approach promises to show real advantages over more traditional approaches is that of inviscid flows with moving boundaries. The preliminary results shown here, and those of Quirk [4] and Pember et al [14], for moving-boundary flows suggest that

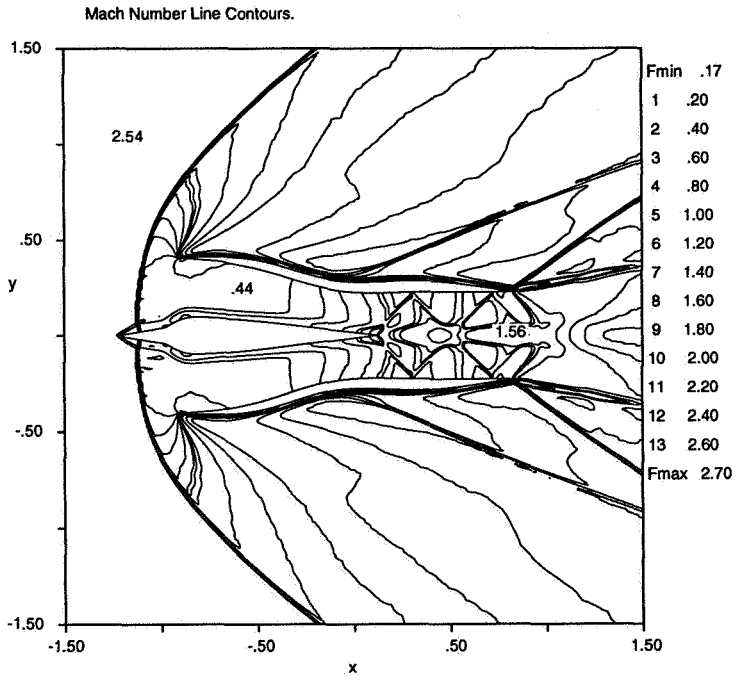


Figure 16: Mach Contours for Inlet Problem — Intermediate Time

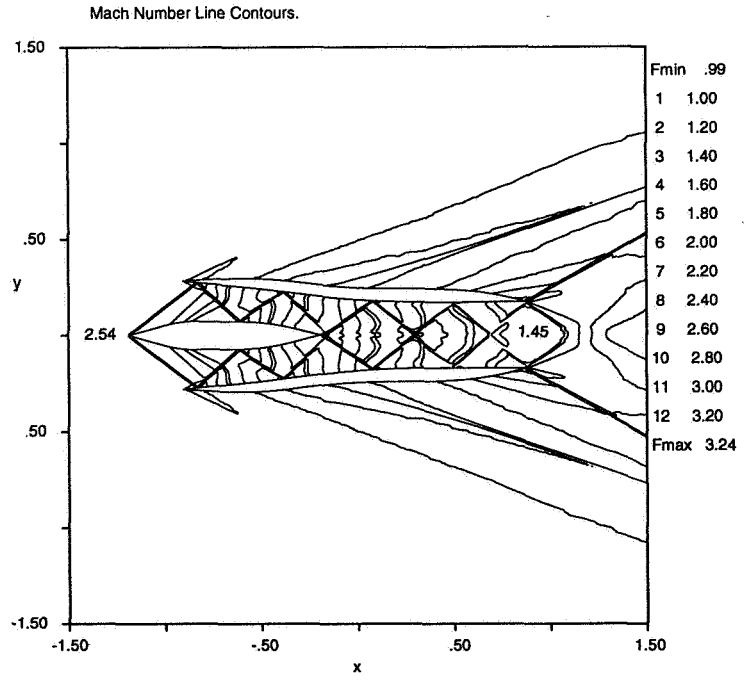


Figure 17: Mach Contours for Inlet Problem — Final Time

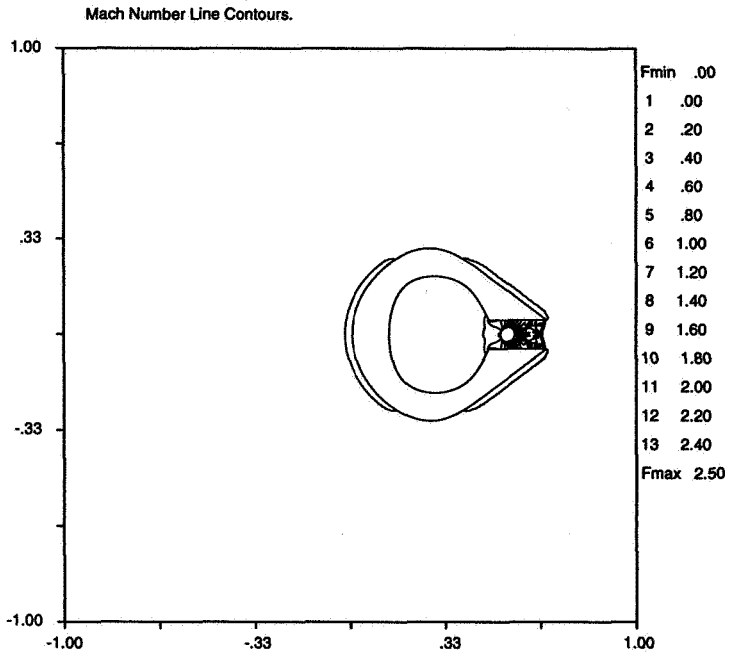


Figure 18: Mach Contours for Propulsion Problem — Initial Time

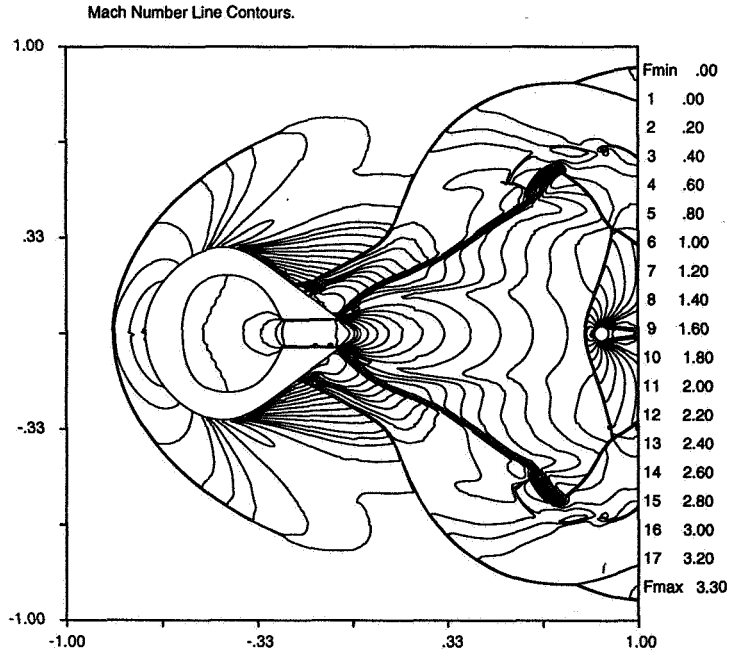


Figure 19: Mach Contours for Propulsion Problem — Final Time

this class of problems may be one for which the Cartesian approach is the best suited.

For three-dimensional inviscid flows, the automated grid generation of the Cartesian approach is even more attractive, but for one drawback. That drawback is that resolving one coordinate direction along a body typically means over-resolving another coordinate direction, for all but the simplest bodies. Judging by the popularity of Cartesian potential-flow solvers, however, for which this drawback also exists, the benefit of automating the grid generation may outweigh the inefficiency when computing geometrically complex cases.

Application of the Cartesian approach to solution of the Navier-Stokes equations is more problematic. At low Reynolds number, the difficulty in constructing an accurate, positive approximation to the viscous terms is the primary problem. At high Reynolds number, the inherent inefficiency of adapting what started out as a Cartesian mesh to non-coordinate-aligned features is the primary problem. At moderate Reynolds numbers, Cartesian Navier-Stokes codes may be useful in preliminary design, but high-caliber results will rely on a method that uses a body-conforming grid with special treatment of the boundary-layer. For high Reynolds number flows, Cartesian Euler coupled with a boundary-layer solver may be the most useful approach.

More work remains to be done. Issues of accuracy and efficiency for Cartesian-based schemes are still being resolved. However, while there are some classes of flows for which these codes will never compete favorably with more traditional approaches, it is clear that, for other classes of flows, Cartesian codes are not only competitive, but offer unique advantages.

ACKNOWLEDGMENTS

Most of the work presented in this paper was carried out by four of the author's graduate students:

- Darren DeZeeuw
- Sami Bayyuk
- Bill Coirier
- Jeff Benko

The work was funded by NASA Langley Research Center, the McDonnell Aircraft Corporation, and the National Science Foundation.

REFERENCES

[1] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, and S. S. Samant. A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 62:1–66, 1991.

[2] M. J. Berger. Adaptive finite difference methods in fluid dynamics. Technical Report DOE/ER/0377-277, Courant Mathematics and Computing Laboratory, 1987.

[3] J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* PhD thesis, Cranfield Institute of Technology, 1991.

[4] J. J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrariliy complex two-dimensional bodies. ICASE Report 92-7, 1992.

[5] G. Warren, W. K. Anderson, J. Thomas, and S. Krist. Grid convergence for adaptive methods. In *AIAA 10th Computational Fluid Dynamics Conference*, 1991.

[6] T. J. Barth. On unstructured grids and solvers. In *Computational Fluid Dynamics*. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[7] D. De Zeeuw and K. G. Powell. An adaptively-refined cartesian mesh solver for the Euler equations. *Journal of Computational Physics*, 104(1):56–68, 1993.

[8] M. J. Berger and R. J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. In *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[9] I. L. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical Report UCRL-97200, Lawrence Livermore National Laboratory, 1987.

[10] S. Bayyuk, K. G. Powell, and B. van Leer. A simulation technique for two-dimensional unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry. AIAA Paper 93-3391-CP, 1993.

[11] W. J. Coirier and K. G. Powell. An accuracy assessment of Cartesian-mesh approaches for the Euler equations. Aiaa paper, 1995.

[12] T. I. Gombosi, C. P. T. Groth, P. L. Roe, and S. L. Brown. 35-moment closure for rarefied gases: Derivation, transport equations, and wave structure. submitted to *Physics of Fluids*, 1994.

[13] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. Technical Report UCRL-97196, Lawrence Livermore National Laboratory, 1987.

[14] R. B. Pember, J. G. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. Adaptive cartesian grid methods for representing geometry in inviscid compressible flow. In *AIAA 11th Computational Fluid Dynamics Conference*, 1993.

SOME OBSERVATIONS ON MESH REFINEMENT SCHEMES APPLIED TO SHOCK WAVE PHENOMENA*

James J. Quirk†
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center, Hampton VA 23681-0001, USA.

SUMMARY

This workshop's double-wedge test problem is taken from one of a sequence of experiments which were performed in order to classify the various canonical interactions between a planar shock wave and a double wedge. Therefore to build up a reasonably broad picture of the performance of our mesh refinement algorithm we have simulated three of these experiments and not just the workshop case. Here, using the results from these simulations together with their experimental counterparts, we make some general observations concerning the development of mesh refinement schemes for shock wave phenomena.

INTRODUCTION

For problems governed by disparate physical scales, the potential savings to be gained from using local mesh refinement are often so large that any strategy will pay handsome dividends: a poor refinement scheme is better than none. Consequently the literature is littered with examples where some form of mesh refinement capability has been botched in a problem specific manner. Superficially the 'quick and dirty' approach appears attractive because the development costs are considerably less than those for a general scheme. In practice, however, the development costs of a general scheme can be recouped across a wide range of projects, and over time the cost/project becomes negligible. On the other hand, with the one-off approach the effective costs accumulate with each passing project and can become unexpectedly large over time. Moreover since one-off schemes rarely reach maturity, they tend to be needlessly expensive to run. Therefore, taken overall, we feel there is no merit in pursuing one-off refinement strategies.

Nevertheless, since an algorithm has to strike a balance between that which is desirable and that which is practicable, an element of 'horses for courses' remains even amongst general purpose mesh refinement schemes. Therefore a method, say, which was designed to provide the cheapest medium-accuracy solution to a steady flow problem might not be competitive when it comes to producing the most accurate solution to a time-dependent problem, and vice versa. Thus some care should be

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

†New address: Graduate Aeronautical Laboratories, Mail Code 205-45, California Institute of Technology, Pasadena CA 91125.

taken in choosing the most appropriate form of mesh refinement before embarking on what might be an arduous exercise in software development. Given our research interests[10], in 1988 after much deliberation we plumped for a form of embedded mesh refinement first developed by Berger and co-workers[1, 2, 3].

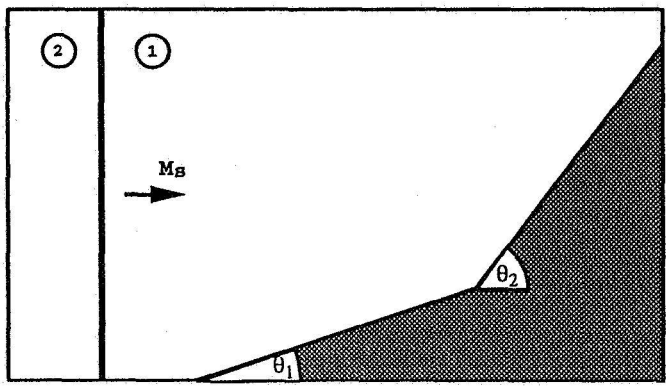
Our formulation[10, 13] has now matured well beyond the development stage and so will not be described here. Instead we wish to engender some discussion as to the strengths and weaknesses of different refinement strategies as applied to investigations of shock wave phenomena. Our aim is not to promote one scheme over another, but to reveal some pitfalls which await the unwary. Therefore to place this discussion in the right context we will first present our numerical results for the workshop double-wedge test problem. This problem was inspired by a series of experiments performed by Takayama *et al.*[15] at Tohuku University to clarify the various types of reflection processes that can occur when a planar shock wave interacts with a double wedge. In view of this, it is worthwhile considering more than just the case chosen for the workshop and we will in fact present results from three different cases.

SHOCK DOUBLE-WEDGE INTERACTIONS

With reference to the schematic shown in Figure 1, we have simulated the interaction of a planar shock wave with three different double-wedge configurations (θ_1, θ_2). These configuration were chosen to match those in the experiments of Takayama *et al.*[15]. Given the instructions for the workshop, the flow was modelled using the two-dimensional Euler equations, taking the equation of state to be that of a perfect gas with ratio of specific heats (γ) set to 1.4 . The Mach number for the incident shock (M_S) was taken to be 2.16 giving a pressure ratio p_2/p_1 of 5.28.

The computational method used for our simulations is the same as in [11] i.e. a non-body-fitted grid was used in conjunction with a two-step finite-volume integration scheme. The effective resolution of the grid was equivalent to that of a uniform mesh of 2240 by 1280 cells. This was obtained using two levels of dynamic refinement, each by a factor of 4, on a uniform base grid of 140 by 80 cells.

Since this paper contains a number of interferograms and schlieren images whose sizes have been reduced solely to keep the length of this paper within acceptable limits we have also placed them on the World Wide Web at URL **http://www.icase.edu/~jjq/flowviz/gal_dwedge.html** so that they might be viewed at their original quality.



Expt.	θ_1	θ_2
#1	15°	35°
#2	20°	55°
#4	60°	30°

Figure 1: Double-wedge configurations used in the experiments of Takayama *et al.*[15] and our numerical simulations.

Experiment #1

A sequence of schlieren snapshots from our simulation of this interaction are shown in Figure 2. At early times, frames (a) and (b), there is single Mach reflection (SMR) of the incident wave from the first ramp. At intermediate times, frames (c) and (d), the Mach stem from this primary reflection interacts with the second wedge giving rise to a secondary reflection which is also of type SMR. At late times, frames (e) to (h), the secondary reflection interferes with the primary reflection. In Figure 3 a numerical interferogram is shown with its experimental counterpart. The two images are in good agreement, hence there is a reasonable quantitative agreement between simulation and experiment. Nevertheless, there are some clear discrepancies on the small scale. For example, in the experiment the base of the primary reflected shock has a small lambda foot due to its interaction with the boundary layer on the bottom wall of the shock tube (see bottom-left corner of image). This feature is missing in the numerical image since the simulation assumed that the flow was inviscid. In principle, adding viscous terms to the simulation is not difficult. However, a much finer grid would have to be used so as to resolve the relevant viscous scales. Thus the cost of the simulation would be increased dramatically and in this instance it is debatable whether the small improvements to be gained by adding physical viscosity would prove cost effective

Experiment #2

A sequence of schlieren snapshots from our simulation of this interaction are shown in Figure 4. At early times, frames (a) and (b), there is SMR of the incident wave from the first ramp as in Experiment #1. However, at intermediate times, frames (c) and (d), the reflection of the Mach stem is now complex Mach reflection (CMR) rather than SMR. At late times, frames (e) to (h), the secondary reflection again interferes with the primary reflection. In Figure 5 a numerical interferogram is shown with its experimental counterpart. The two images are in reasonable agreement, but the tie-up is noticeably poorer than in Experiment #1. Again the discrepancies are due to the lack of physical viscosity in the flow model. For example, in the experimental image there is a recirculation zone at the apex of the first ramp, and the base of the secondary reflected shock has a lambda foot due to its interaction with the boundary layer on the wedge. But these features cannot be reproduced by an inviscid simulation. Here the shock-boundary layer interactions are stronger than in Experiment #1 and have had quite a pronounced affect on the curvature with which both the primary and secondary reflected shocks run in to the wall. Consequently there would be some justification for switching to a viscous simulation for this experiment.

Experiment #4

A sequence of schlieren snapshots from our simulation of this interaction are shown in Figure 6. At early times, frames (b) to (c), the slope of the first wedge is sufficient that there is regular reflection (RR) and not SMR as in the other two experiments. At late times, frames (d) to (h), the incident shock diffracts around the convex corner formed by the two wedges. In Figure 7 a numerical interferogram is shown with its experimental counterpart. The two images are in good agreement except for those regions where viscous effects are expected to be important. Namely, the vortex core near the convex corner, and the foot of the reflected shock where it interacts with the boundary layer on the wall of the shock tube. This interaction affects the curvature of the reflected shock and would seem to account for the difference in the curvature of the fringes between the computational and experimental interferograms. However, the tie-up is sufficiently good that, as in Experiment #1, it is not clear that a viscous simulation would be worth the extra effort involved.

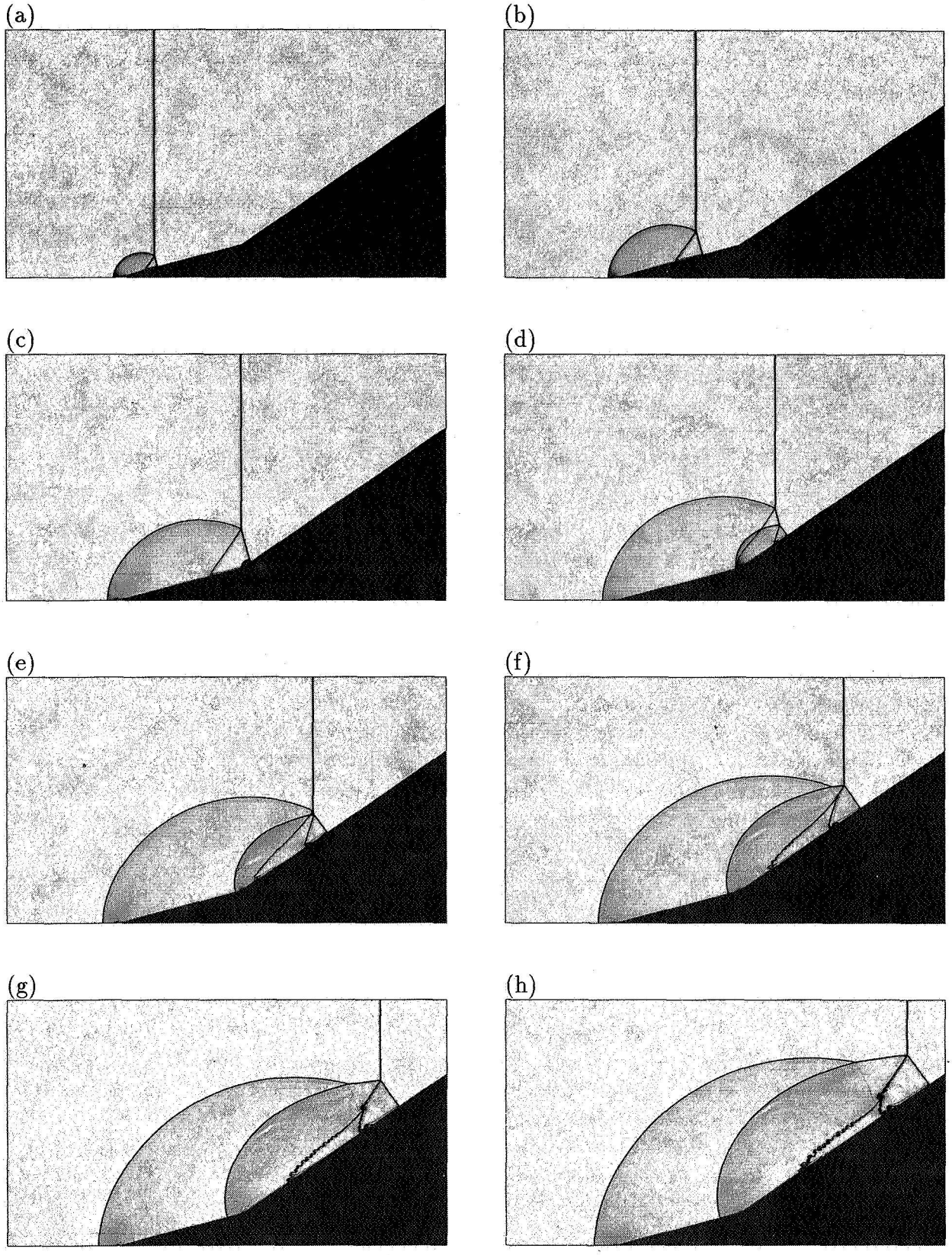
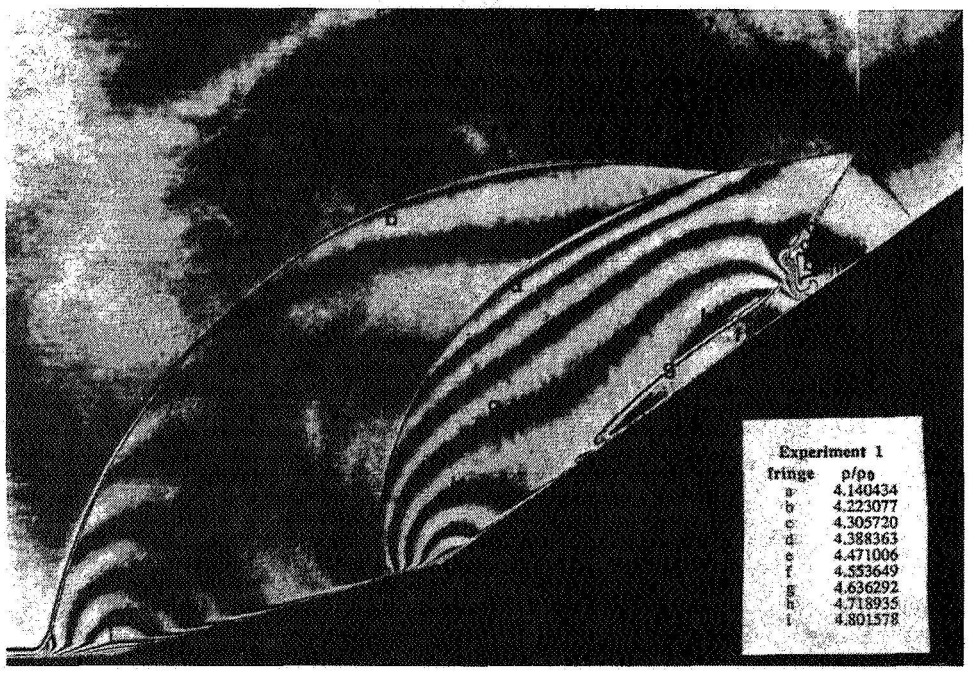


Figure 2: Sequence of schlieren snapshots from the simulation of Experiment #1.

(a) Experimental Interferogram, courtesy of Prof. Takayama



(b) Numerical Interferogram

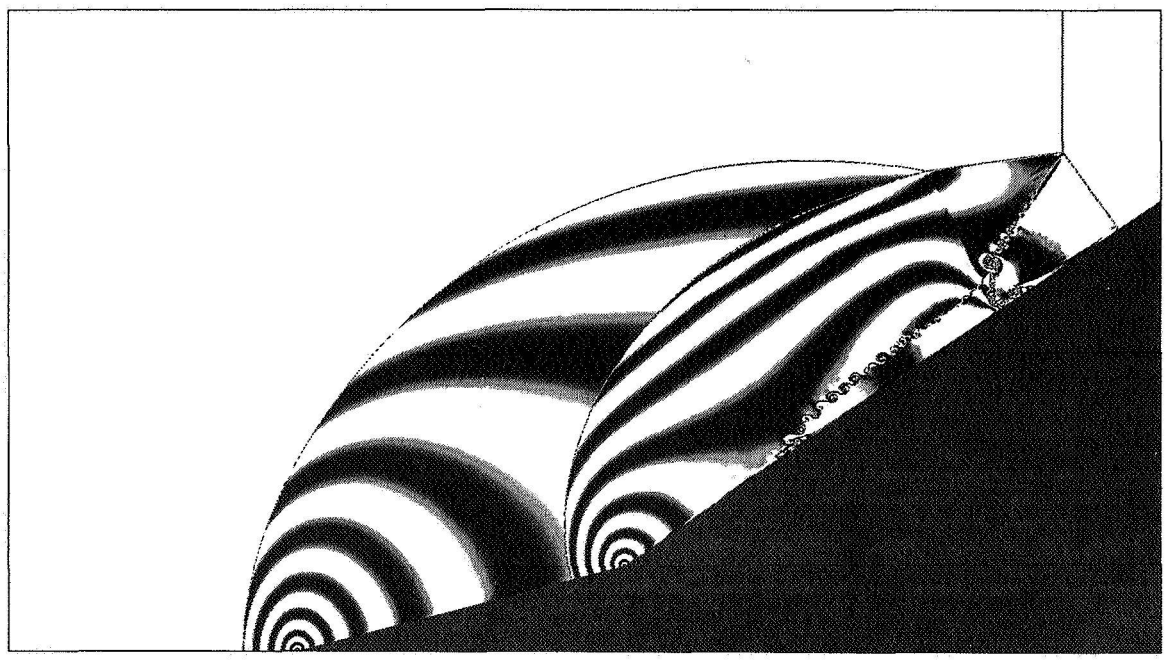


Figure 3: Comparison between numerical and experimental interferograms for Experiment #1.

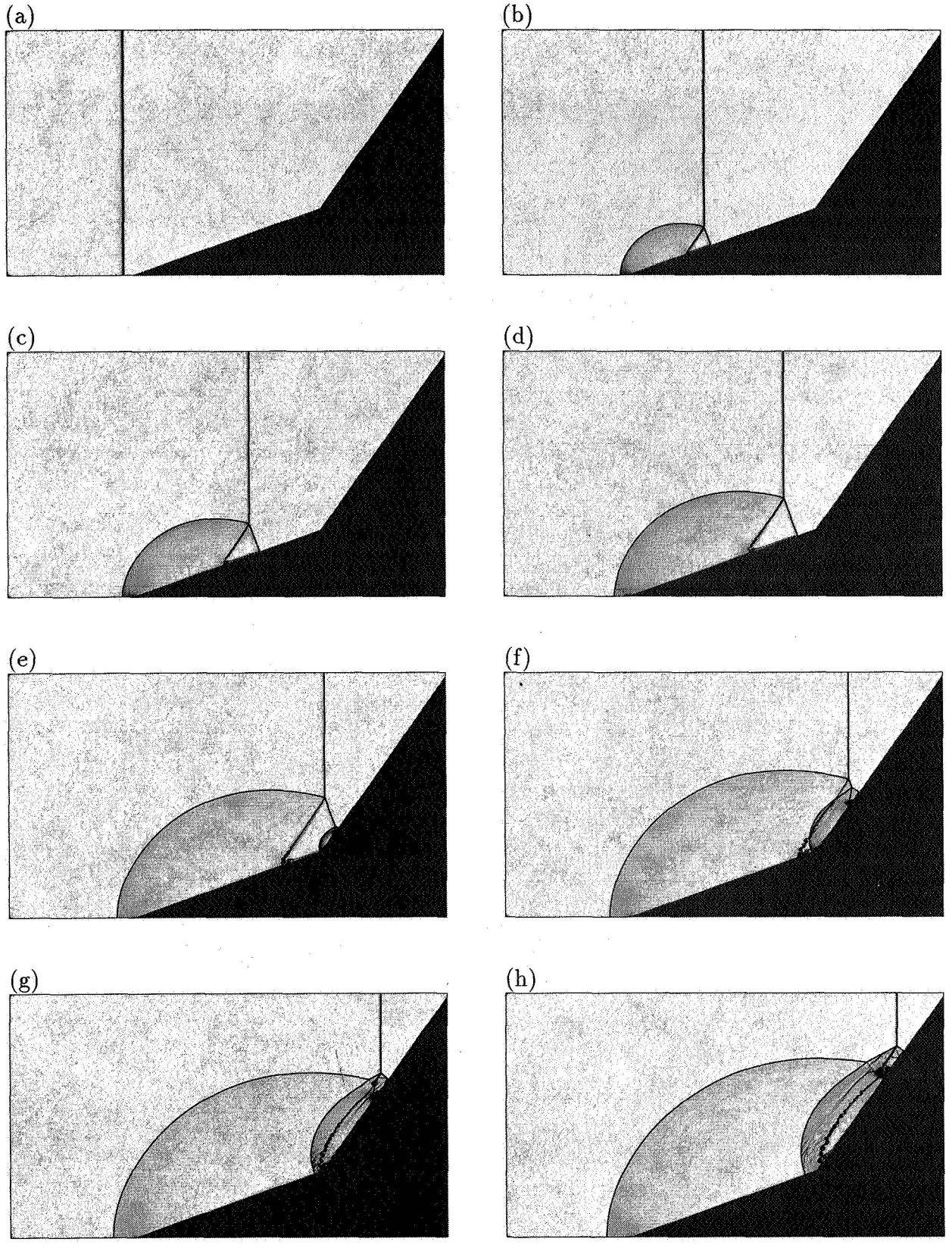
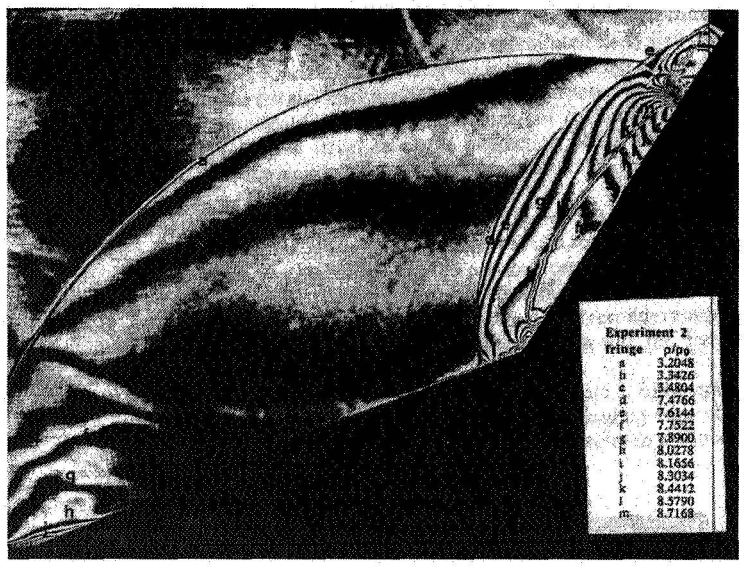


Figure 4: Sequence of schlieren snapshots from the simulation of Experiment #2.

(a) Experimental Interferogram, courtesy of Prof. Takayama



(b) Numerical Interferogram

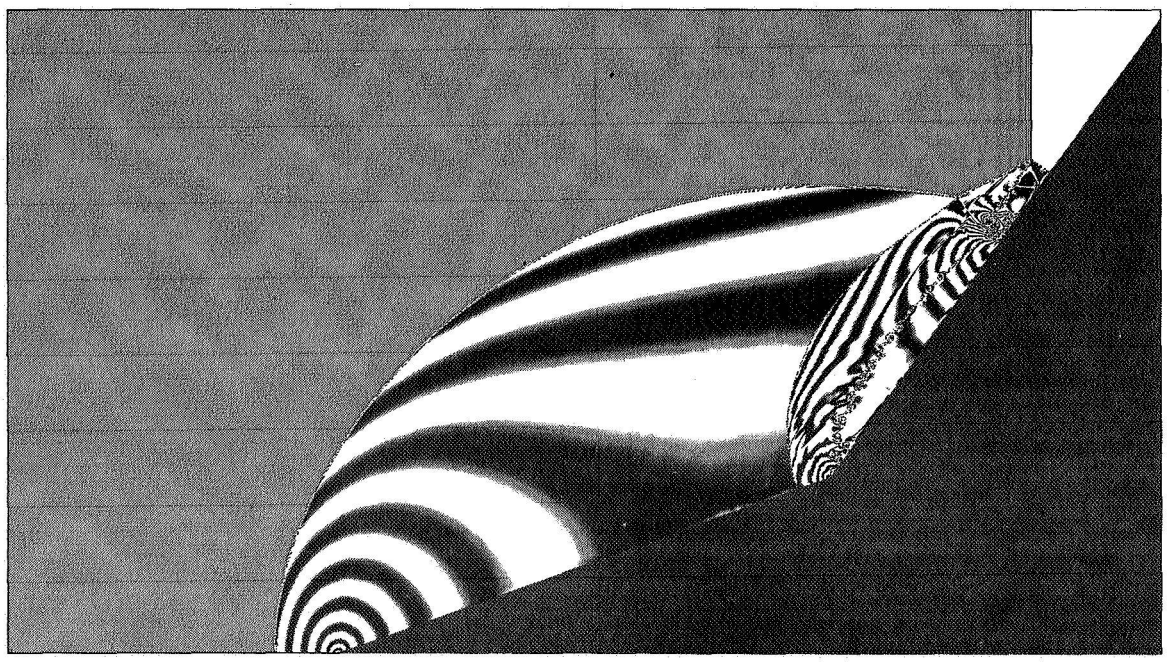


Figure 5: Comparison between numerical and experimental interferograms for Experiment #2.

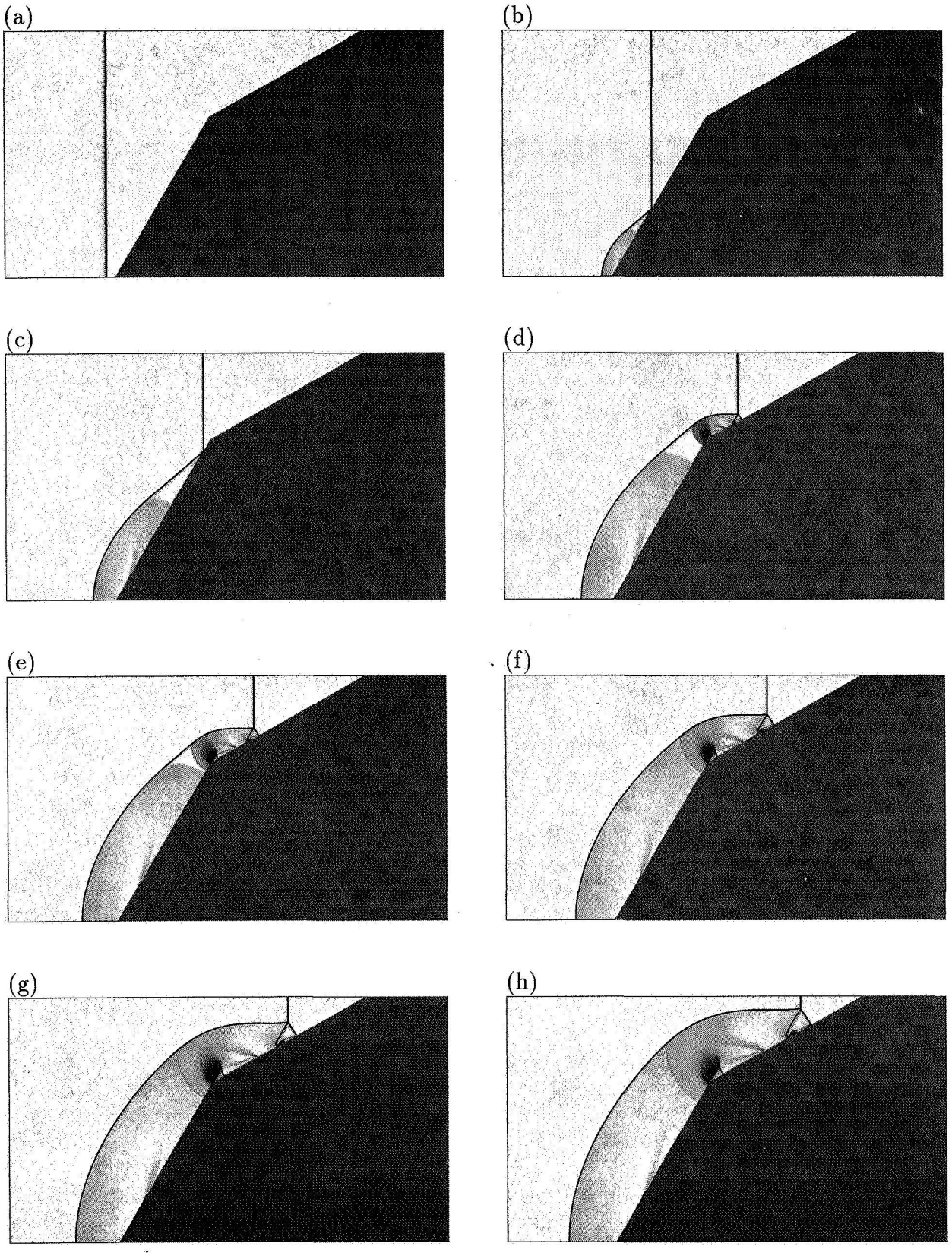


Figure 6: Sequence of schlieren snapshots from the simulation of Experiment #4.

(a) Experimental Interferogram, courtesy of Prof. Takayama

(b) Numerical Interferogram

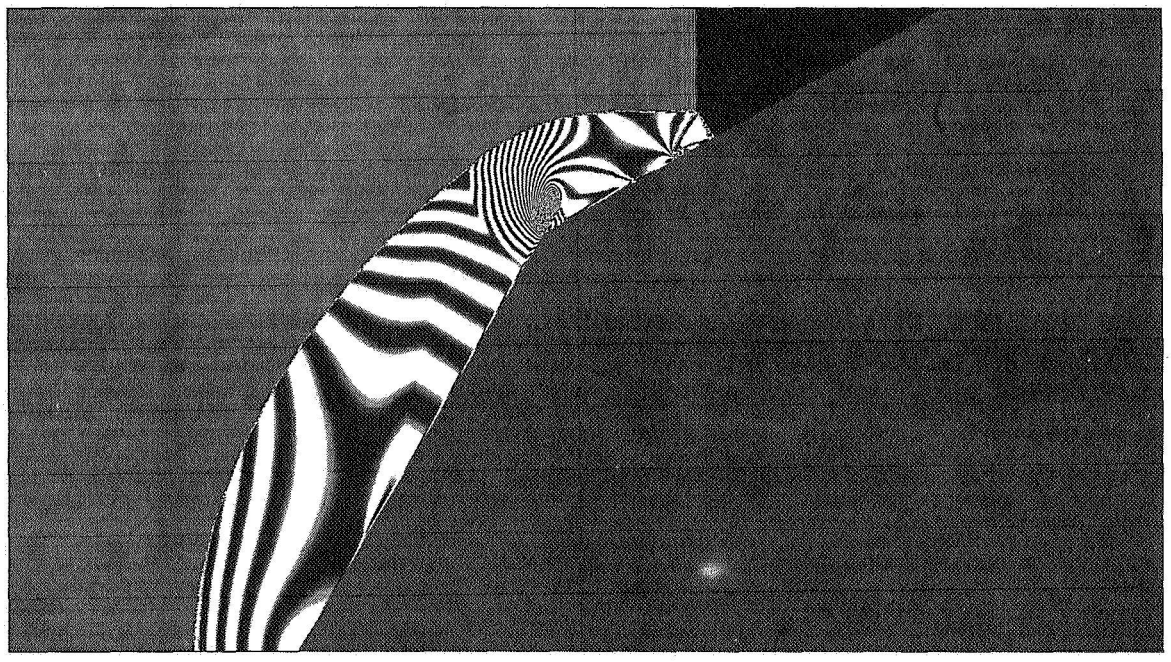


Figure 7: Comparison between numerical and experimental interferograms for Experiment #4.

DISCUSSION

Here we restrict ourselves to making some specific observations about the development of mesh refinement methods for investigations of unsteady shock wave phenomena, and the reader who is unfamiliar with the basic techniques of mesh refinement is directed to[9, 17].

The majority of mesh refinement schemes give the impression of having been designed solely to minimize the number of grid cells that are required to compute a solution of a given resolution or accuracy. This design philosophy is presumably based on the notion that the effort required to integrate a discretized flow solution decreases as the number of grid cells decreases. But the following example demonstrates that the number of grid cells can have surprisingly little bearing on the cost of performing a time-dependent simulation and so this particular design philosophy is flawed.

Consider the propagation of a shock down a uniform mesh of N cells, each of width Δx. If a uniform time step is chosen such that the Courant number based on the speed of the shock is one (hence the shock traverses one cell per time step), it will take N integrations of N cells for the shock to pass through the domain i.e N^2 cell integrations. Now halve one cell in the grid such that there are $N-1$ cells of width Δx and two of width $\Delta x/2$. Again if a uniform time step is used to propagate the shock through this domain, without violating the CFL condition it will take $2N$ integrations of $N+1$ cells to propagate the shock through the domain i.e. $2N^2+2N$ integrations. Therefore although but a single cell has been added to the grid the cost of the simulation has more than doubled. Thus for time-dependent problems it is desirable to refine in time as well as space[10]. Here, using temporal refinement, the two small cells would be integrated $2N$ times and the other $N-1$ cells would be integrated N times as in the uniform mesh case i.e. a total of N^2+3N integrations. Thus, for N reasonably large, the cost of the refinement becomes negligible. As an alternative to temporal refinement one could conceivably opt for an integration scheme which was stable for large Courant numbers, but for highly non-linear problems the loss in temporal accuracy would probably prove unacceptable.

A temporal refinement strategy is easily incorporated into hierarchical refinement schemes such as those based on quad-trees (e.g. [4]) or embedded patches (e.g. [3, 10]) since it is possible to avoid ever having to interpolate across discontinuities[10]. However, a temporal refinement strategy seems ill-suited to refinement schemes based on unstructured triangular meshes (as typified by[6]), at least when combined with a shock-capturing methodology, since one cannot avoid having to perform awkward non-linear interpolations at discontinuities. Such interpolations are unlikely to satisfy a shock-capturing scheme's unique smeared shock profile and so would result in spurious oscillations[10]. One convenient way around this difficulty would be to employ an integration scheme based on floating shock-fitting[7, 16] rather than shock-capturing. Then there would be no smeared discontinuities and the cause of the problem disappears. This strategy illustrates an important feature of the design of mesh refinement methods. It is often better to work around difficulties than to attempt to effect a cure. A refinement scheme contains many components and the best schemes seem to be those whose components work symbiotically.

Leaving aside the issue of temporal refinement, minimizing the number of grid cells will not automatically lead to an efficient method of refinement. Consider the case of an isolated discontinuity which runs oblique to the grid. It is clear that cellular quad-tree refinement (say [4]) is more efficient than embedded patch refinement (say [10]) in terms of the number of cells each method requires to

C-2

tile the discontinuity. However, it also has the larger storage overheads per mesh cell of the two associated data structures. For inert shock wave simulations which need only a small number of levels of refinement the storage overheads from quad-tree refinement are easily tolerated, but this might not be the case if the flow contained chemical reaction. Instead of a shock oblique to the grid consider a detonation wave which in addition to a shock front has some internal structure, albeit on a very fine scale, which must be resolved and cannot be captured. In this instance one would need a wide swathe of cells to cover the reaction zone which might be ten or more levels of refinement down in the quad tree because of the disparateness between the width of the reaction zone and the distance over which the detonation wave needs to be propagated. Therefore although the cells in the swathe are close to one another spatially they could lie far apart in the quad tree structure, which might impact on a parallel implementation of the scheme, and each cell would introduce a large overhead due to the accumulation of pointers down to its level in the data structure. Consequently embedded patch refinement might now prove to be more efficient because its storage overheads would be so much lower and it would better preserve the proximity of cells within the reaction zone.

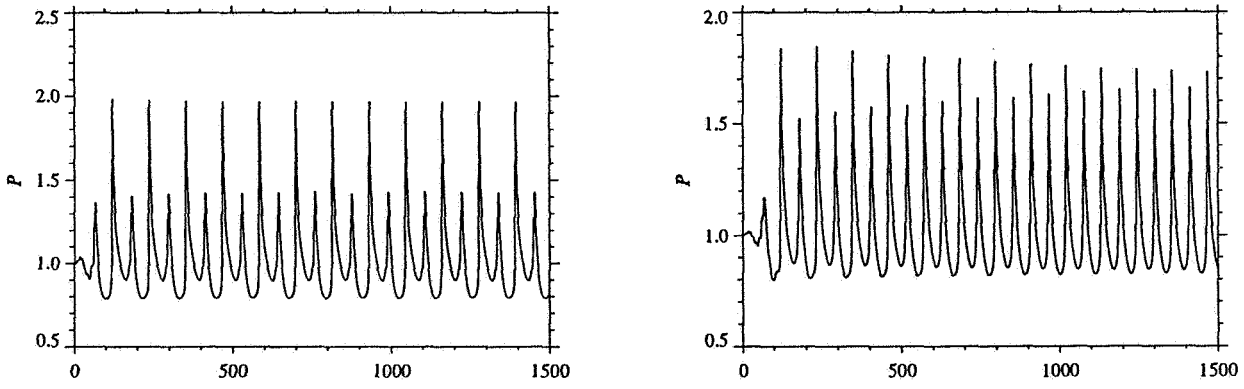
Adaptive mesh refinement algorithms, unlike classical numerical methods, entail quite sophisticated software. Therefore arguments such as the one above must be tempered by the realization that specific implementation details can make or break an algorithm in terms of its practical performance. In particular the grid data structure needs to be well crafted. For example, the data storage needs to be flexible enough to cope with dynamic allocation and deallocation as local refinement is added and removed, and data accesses have to be efficient so as not to impact on performance. Now since it is all too easy to underestimate the level of commitment required to write, test and debug a pukka mesh refinement code, any newcomer would be well advised to take his or her own software skills in to account before choosing to code up any one particular method.

In fact the number of considerations that must be taken in to account before choosing a mesh refinement strategy are legion, even when one's needs are fairly specific. For example, our interests lie in investigating complex shock-wave phenomena, and given the results from the previous section it would appear that our refinement algorithm is well suited to our purposes. But suppose we were dissatisfied with the quality of our results for Experiment #2 (Figure 5) and wanted to perform a viscous simulation, would our scheme cope as well as in the inviscid case?

In the past the scheme has been used to perform viscous simulations of shock-boundary layer interactions[10], and so there is no reason to believe that it could not cope with a viscous simulation of Experiment #2. However, since viscous flow features tend to be anisotropic in nature, such a simulation would expose a weakness of our refinement scheme: it does not cope very well with anisotropic refinement. The method used[10] is basically limited to features like boundary layers which are affixed to solid surfaces. To refine a free shear layer which might happen to lie oblique to the mesh we would be forced to use isotropic refinement which would be needlessly expensive. This is an example where a change in the flow model can have a significant impact on the refinement efficiency, even though the application remains unchanged. Thus the correct choice of refinement strategy is never straightforward. To complicate matters even further, one cannot ignore the interplay between the method of refinement and the method of flow integration. For example, a triangular unstructured mesh has the geometric flexibility to allow for efficient anisotropic refinement but a certain amount of care must still be taken to generate meshes that are suitable for viscous simulations[8]. In general, depending on the application, one might wish to compromise the refinement efficiency so as to avoid compromising the accuracy of the flow integration (or vice versa). Of course the accuracy of a refinement scheme is, for the most part, ordained by the monitor functions which determine where refinement does or does not take place.

As is common practice we employ heuristic functions to determine where to refine, and coarsening takes place naturally by choosing not to refine and so involves no additional criteria[10]. For the present double wedge problems we used a combination of two monitor functions: density gradients were used to locate shocks and a local comparison between density and pressure gradients was used to locate contact discontinuities[13]. Now there are numerous reasons why this type of heuristic approach is unsatisfactory, not least of which is that it introduces tunable parameters and so increases the experience factor needed to operate a refinement scheme reliably. As Warren *et al.*[18] have shown, a poorly constructed heuristic monitor function can cause a mesh refinement scheme to home in on an incorrect solution. But this can happen with any refinement function, heuristic or not, which provides estimates for the local error without also providing estimates for how the local error affects the global error i.e. every refinement function in common use. To a large extent the mesh refinement community has been lulled into a false sense of security by the general experience that local errors are often benign. The test case discussed in [18] is a gentle reminder that small local errors can sometimes tip the balance and result in large global errors, but other more pathological examples are not difficult to find especially where chemical reaction is involved.

Figure 8 (a) shows a trace of the pressure behind the lead shock front of a one-dimensional detonation wave which exhibits a galloping instability[14]. By normal standards this computation would have been thought to be well resolved since 160 mesh points covered the so-called reaction half-length (giving some 256,000 cells over the time period shown) when contemporary simulations have ten or less points in the reaction half-length. However, when the simulation was repeated with the grid spacing halved, the dynamic behaviour of the detonation wave altered dramatically, see Figure 8 (b). At first glance one might assume that Figure 8 (b) came from the coarser computation since it looks more dissipative in that a two mode pulsation is decaying to a single mode pulsation. But in fact it is the extra dissipation in Figure 8 (a) that sustains a spurious two mode pulsation whereas the correct behaviour should be that of a two mode pulsation with a time-attractor limit cycle[14] i.e. Figure 8 (b). Interestingly the difference in behaviour arises not from an error in resolving the detonation shock front, but from a failure to resolve an innocuous part of the reaction zone which is smooth.



(a) 160 pts/$L_{\frac{1}{2}}$ (b) 320 pts/$L_{\frac{1}{2}}$

Figure 8: Variation in the computed pressure history trace for a galloping detonation wave when the mesh spacing is halved[14].

Clearly there is much room for improvement in the current crop of criteria used to control refinement. However, any attempts at devising rigorous mathematically based refinement criteria should not ignore certain practicalities. For example, in our simulations it can be necessary to adapt the grid tens of thousands of times[13] and so the method of determining where to refine must be reasonably cheap otherwise it would cripple the simulation. Also, the physical scales involved are so disparate that one cannot afford the luxury of periodically comparing the solution computed with refinement against that computed on a uniform mesh of the same high resolution, as is effectively done in[5], because this would require an unrealistic amount of storage.

For practical purposes the lack of a fool-proof refinement criteria does not undermine the usefulness of adaptive mesh refinement schemes for investigating shock wave phenomena, but it does complicate matters. Whenever we start to investigate a new problem we perform a sensitivity study to see how the computed results vary with, amongst other things, the effective resolution of the computational grid as controlled by our chosen refinement criteria. Thus we tool-up to a position where we think we can produce a reliable simulation. Note we would do more or less the same thing even if we were not employing mesh refinement, as past performance is no real guide as to how a numerical scheme will fair on a new problem.

For serious investigations the cost of tooling is generally spread over a parameter study and so is not excessive. The only drawback we find is that the results from sensitivity studies are rarely as conclusive as we would like. Many shock wave phenomena exhibit physical instabilities and so the notion of a grid converged solution is not always clear, or even appropriate since the flow model might preclude the possibility of having a sensible solution in the limit of the mesh spacing going to zero. For example, in [12] we presented results for the vortex sheet produced by a shock wave diffracting over a knife edge. These results show that an inviscid simulation can reproduce the correct behaviour and yet provide no limiting solution since the numerical dissipation which controls the fine scale structure of the vortex sheet, in the absence of physical viscosity, never bottoms out as the grid is refined. On the other hand, in some of our simulations of detonation phenomena it is clear that we are incapable of reaching a fully converged solution either because the physical scales are too disparate for our computing resources or the physical behaviour of the system is non-deterministic in that variations in discretization errors, no matter how small, lead to significant variations in dynamical behaviour.

Most CFD simulations are performed with the aim of producing quantitative answers to well understood problems, in which case the above vagaries are abhorrent. However, much of our work is performed in an attempt to fathom behaviour which is not known and simulations are used as a qualitative diagnostic and so a certain amount of subjectivity cannot be avoided. In short we use our mesh refinement algorithm to perform simulations which are more detailed than would otherwise be possible. Consequently we close this discussion without making any attempt to sell our scheme in terms of how efficiently it was able to compute the workshop double wedge problem. While this might be viewed as contrary we would argue that any results we could present would have little practical value: by comparison to our recent studies[13] the present simulations are so cheap as to be almost inconsequential. Moreover it should be appreciated that the cost of performing a time-dependent simulation can pale into insignificance when compared to the time taken to decipher the results, and to bandy performance figures would lose sight of the fact that our scheme has progressed well beyond the development stage and is used as an everyday tool.

CLOSING COMMENTS

At times adaptive mesh refinement appears to be more of an art than a science, therefore on a self-indulgent note we close with two quotations that sum up our thoughts on this branch of computational fluid dynamics.

The first quotation is taken from Shakespeare's *Twelfth Night* and explains why we feel there will always be a plethora of refinement schemes: "some [methods] are born great" in that they are so well suited to a particular class of problem they do not deserve to be replaced by some monolithic refinement scheme; "some [methods] achieve greatness" when they leapfrog the field by virtue of being able to exploit some new generation hardware feature and so methods tend to pass in and out of fashion; "and some [methods] have greatness thrust upon 'em" in that many fluids researchers cannot develop their own refinement code and must make do with whatever is available, so schemes that should be put to rest will not die by dint of their users.

Our second quotation is attributed to the son of the author Alexandre Dumas: "All generalizations are dangerous, even this one." In this paper we have tried to emphasize that context is all important where mesh refinement is concerned. Therefore whilst the subject is sorely in need of some formalism to guide us out of the present heuristic quagmire, there needs to be a realization that not all needs are the same. As we have shown, following rigorous criteria which are misplaced can prove disastrous. Therefore if some of our observations appear provocative it is only because we are attempting to correct an imbalance, as we see it, in current thinking. Grid convergence is a case in point. While rigorous Mathematical concepts of convergence are unambiguous, the practical concept of a grid converged solution to an unsteady problem, where the flow might be physically unstable, is hazy to say the least. And so common ground must be found between theoreticians and the practical exponents of mesh refinement before any real progress can be made in eliminating the heuristic elements from today's algorithms.

Acknowledgements

I would like to thank Prof. K. Takayama for his generosity in allowing me to reproduce the experimental interferograms shown in Figures 3, 5 and 7, and I am happy to acknowledge the efforts of Dr. H. Babinsky in this matter. I am also very grateful to Dr. T.W. Roberts who had the thankless task of standing in for me at the workshop. Lastly, I would like to thank Dr. J. Van Rosendale who held the boat while I scribed this contribution.

REFERENCES

[1] M. J. BERGER, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations.* Ph.D. thesis, Computer Science Dept., Stanford University (1982).

[2] M. J. BERGER AND J. OLIGER, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations.* J. Comput. Phys., **53**(1984), pp. 482–512.

[3] M. J. BERGER AND P. COLELLA, *Local Adaptive Mesh Refinement for Shock Hydrodynamics.* J. Comput. Phys., **82**(1989), pp. 67–84.

[4] Y.-L. CHIANG, B. VAN LEER AND K.G. POWELL, *Simulation of Unsteady Inviscid Flow on an Adaptively Refined Cartesian Grid.* AIAA Paper 92-0443, 1992.

[5] A. HARTEN, *Multiresolution Algorithms for the Numerical Solution of Hyperbolic Conservation Laws.* UCLA CAM Report 93-03, 1993.

[6] R. LÖHNER, K. MORGAN AND O. ZIENKIEWICZ, *Adaptive Grid Refinement for the Compressible Euler Equations.* In "Accuracy Estimates and Adaptivity for Finite Elements", Wiley, 1994.

[7] M. PARASCHIVOIU, J.-Y. TRÉPANIER, M. REGGIO AND R. CAMARERO, *A Conservative Dynamic Discontinuity Tracking Algorithm for the Euler Equations.* AIAA Paper 94-0081, 1994.

[8] S. PIRZADEH, *Unstructured Viscous Grid Generation by the Advancing-Layers Method.* AIAA J., **32**(1994), pp. 17–19.

[9] K. G. POWELL, P. L. ROE AND J. J. QUIRK, *Adaptive-mesh Algorithms for Computational Fluid Dynamics.* pp. 303–337, Algorithmic Trends in Computational Fluid Dynamics, Edited by M. Y. Hussaini, A. Kumar and M. D. Salas. Springer-Verlag, New York, 1993.

[10] J. J. QUIRK, *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* Ph.D. thesis, College of Aeronautics, Cranfield Institute of technology (1991).

[11] J. J. QUIRK, *An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two-Dimensional Bodies.* Computers & Fluids, **23**(1994), pp. 125–142.

[12] J. J. QUIRK, *A Cartesian Grid Approach with Hierarchical Refinement for Compressible Flows.* pp. 200–209, Computational Fluid Dynamics '94,(Invited Lectures and Special Technological Sessions of the Second European Computational Fluid Dynamics Conference 5-8th September 1994, Stuttgart, Germany), Edited by S.Wagner et al., John Wiley and Sons Ltd., Chichester, 1994.

[13] J. J. QUIRK, *A Parallel, Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* To appear in Applied Numerical Mathematics.

[14] M. SHORT AND J. J. QUIRK, *Evolution of a One-dimensional Pulsating Detonation Wave Driven by a Model 3-step Chain-branching Reaction.* Submitted.

[15] K. TAKAYAMA, O. ONODERA AND G. BEN-DOR, *Holographic Interferometric Study of Shock Transition over Wedges.* SPIE, **491**(1984), pp. 976-983.

[16] J. VAN ROSENDALE, *Floating Shock Fitting via Lagrangian Adaptive Meshes.* ICASE Report No. 94-89, 1994.

[17] J. F. THOMPSON AND N. P. WEATHERILL, *Aspects of Numerical Grid Generation: Current Science and Art.* AIAA Paper 93-3539, 1993.

[18] G. WARREN, W. K. ANDERSON, J. THOMAS AND S. KRIST, *Grid Convergence for Adaptive Methods.* AIAA Paper 91-1592, 1991.

ADAPTIVELY–REFINED OVERLAPPING GRIDS FOR THE NUMERICAL SOLUTION OF SYSTEMS OF HYPERBOLIC CONSERVATION LAWS *

Kristi D. Brislawn, David L. Brown, Geoffrey S. Chesshire and Jeffrey S. Saltzman
Los Alamos National Laboratory
Los Alamos, NM

SUMMARY

Adaptive mesh refinement (AMR) in conjunction with higher-order upwind finite-difference methods has been used effectively on a variety of problems in two and three dimensions. In this paper we introduce an approach for resolving problems that involve complex geometries in which resolution of boundary geometry is important. The complex geometry is represented by using the method of overlapping grids, while local resolution is obtained by refining each component grid with the AMR algorithm, appropriately generalized for this situation. The CMPGRD algorithm introduced by Chesshire and Henshaw is used to automatically generate the overlapping grid structure for the underlying mesh.

INTRODUCTION

Over the past decade, the Adaptive Mesh Refinement (AMR) algorithm pioneered by Berger and Oliger [1] has proven to be a successful, efficient strategy for obtaining high-resolution solutions to partial differential equations. Using AMR combined with high-order upwind finite-difference methods, one has the ability to simulate shock hydrodynamics problems, including those with multiple materials, in both 2-D and 3-D [2, 3, 4, 5, 6] not otherwise possible within the limitations of present computers. To date most of the developmental work done on the AMR method has concentrated on the perfection of the adaptive algorithm, and not on the development of the capability to represent complex geometry. A notable exception is the "Cartesian grid" method introduced by Berger and Leveque in [7] in which complex geometry is represented by cutting holes in an otherwise rectangular grid, and using special flux formulas in the resulting odd-shaped grid cells at the boundary.

More recent work on this method is reported in [8]. The overlapping grid approach introduced in the present paper uses a more accurate representation of boundary surfaces than the Cartesian

*This work performed under the auspices of the U.S. Department of Energy by Los Alamos National Laboratory under Contract W-7405-ENG-36.

grid method, although with correspondingly more work by the user required in order to construct the initial grid. A set of curvilinear component grids is used, the union of which completely covers the computational region. There are small regions of overlap between the individual component grids. Each component grid is logically rectangular with some of the cells possibly "blanked" out and unused. With this approach, a complex structure can be represented by combining many separate pieces, each represented by its own curvilinear grid. The potential of the overlapping grid method was first demonstrated by Starius [9, 10], Kreiss [11] and Steger et. al. [12]. Successful three-dimensional aerodynamic simulations involving configurations as complex as the space shuttle [13, 14], and with moving components [15], validated the usefulness of this technique as a practical engineering tool. A fully automatic grid overlapping procedure for two- and three-dimensional grids grids (CMPGRD) was developed by Chesshire and Henshaw which forms the basis for the current work [16, 17, 18]. The use of overlapping grids to represent a complex geometry was dubbed the "Chimera" method by the late Joe Steger.

The overlapping grid approach allows a great deal of flexibility in the placement of the component grids. Since the component grids may overlap, rather than being required to match exactly along an interface as with the block-structured grid method [19], they are relatively unconstrained. This additional freedom allows generation of smoother component grids. This is ideal for applying higher-order upwind finite-difference methods, since they perform best on grids whose transformation to the unit square or cube are smooth. The finite-difference method used in this paper is introduced in [20] and is an unsplit Godunov method based on the methods introduced by Colella [21].

Since CMPGRD produces sets of logically rectangular grids, the extension of the AMR method to this framework is natural. The AMR method developed in this paper follows closely the technique discussed by Berger and Colella in [3]. Each component grid of the overlapping grid structure is refined separately by the AMR algorithm. As in [3], the nested refinement grids are constrained to have boundaries coinciding with the underlying "parent" component grid, i.e. none of the refinements are allowed to be rotated with respect to that parent grid. The differences are in the treatment of the cutout and overlap regions of the underlying overlapping-grid.

THE OVERLAPPING GRID AMR ALGORITHM

The adaptive grid construction and solution procedures on an overlapping grid are straightforward extensions of the AMR procedures on a single grid. Modifications are made, as necessary, to accomodate the special requirements of the overlapping grid structure. In order to describe the grid construction and problem solution methods for overlapping grid AMR, we first briefly describe the relevant parts of the procedures for the non-overlapping AMR and the non-adaptive overlapping cases. For simplicity, in both cases we assume that the grid covers a two-dimensional region.

Solution Procedure Using AMR on a Single Grid

The basic AMR mesh construction and solution procedures are a recursive generalization of the basic two-level procedure that we will describe here. The reader is referred to [3] for a more detailed description. At each timestep in the two-level AMR procedure for a single underlying "base" grid, the grid hierarchy consists of the base grid and patches of *properly aligned* refinement grids that have been automatically placed by the AMR algorithm in regions where additional solution accuracy is required. Properly aligned grids have the property that a grid at some level n always has its boundary cells aligned with cell edges of the level $n-1$ grids. In the general n-level algorithm, the grids must have the additional property that they are *properly nested*, which means that a grid at some level n is always found embedded in some subset of the level $n-1$ grids. A level n grid is not allowed to be all or partially embedded in parts of the grid structure that contain only grids at lower levels $(1, 2, \ldots, n-2)$.

Returning to the two-level case, all refinement grids at the fine level are automatically replaced with new refinement grids after every m timesteps, where m is a user–specified interval. The solution data is interpolated onto the new grid hierarchy before the calculation continues. For the purposes of this discussion, assume that solution values are available in all cells of all grids in the current grid hierarchy. The grid refinement regeneration is done automatically by estimating the error in the calculation at the current timestep on all grids, and then defining new refinement grids in regions where the error is estimated to be higher than some user-specified tolerance. In practice, the error estimation is done either by a Richardson-extrapolation procedure that compares solutions on grids of different overall resolution [3, 4], or by measuring the size of local solution gradients and refining in regions where the gradients have become unacceptably large relative to the grid [6]. The latter approach was employed for the computations presented in the present paper. Using one of these procedures, the error is estimated in each cell on the grid, and cells with unacceptably high estimated error are "flagged." Since the grid will not be refined again for m timesteps on the base grid level, it is necessary to expand the refinement region somewhat before a refinement grid is constructed. A simple domain-of-dependence argument requires that an additional row of cells be added around each group of flagged cells for each timestep that the computation will proceed without re-refinement of the grid. This is referred to as the "cell-diffusion" step of the AMR grid construction procedure. Once this is done, the flagged cells are grouped into "boxes", or rectangular regions using a procedure described in [3]. Refined grids with a user-specified refinement factor n_{ref} relative to the base grid are then constructed in each of the boxes. The solution on the previous adaptive grid hierarchy is then interpolated onto the new grid hierarchy. In regions where the solution is defined on more than one grid refinement level, the solution values on the finest grid available are used.

Once the data is available on all of the grids in the new grid hierarchy, the solution procedure can continue. First the coarse grid solution is advanced by one timestep in all interior cells of the coarse grid. This includes coarse grid cells that are covered by a refined grid. Boundary conditions for this step are assumed to be provided as part of the original problem specification. Once the coarse grid solution has been advanced, the solution on the refinement grid patches can be advanced. Since the solution method is explicit, the refinement grid solutions are advanced using the same CFL timestep restriction as on the coarse grid. This means that n_{nref} timesteps must be taken on the fine grids for each single timestep on the coarse grid. Boundary conditions for each of the refinement patches are obtained again following the principle that the "best" values available should be taken. At fine grid boundaries where there is a refinement grid at the same refinement

level n immediately adjacent, values from the adjacent grid are used to provide the boundary conditions. At fine grid boundaries where the adjacent grid is at a coarser level, boundary values are obtained by interpolating the coarse-grid solution in space and time. When advancing the solution of a system of hyperbolic conservation laws with a conservative method, an additional "conservative update" step is taken in which values in coarse grid cells at the coarse-fine grid interface are recomputed using fluxes available from the fine grid calculations. Details of this procedure are found, for example, in [3]. Finally, once fine-level values are available at the new timestep of the coarse grid, the coarse grid solution values in cells covered by refinement patches are replaced by transferring or interpolating fine grid values to the coarse grid cells. This assures that the best possible values are always used in the solution procedure for the succeeding timestep.

Solution Procedure Using Overlapping Grids Without AMR

An overlapping grid in two space dimensions consists of a set of logically rectangular curvilinear component grids that overlap where they meet and whose union completely covers the computational domain for a system of partial differential equations. The cells on each component grid are classified according to their function during the PDE solution procedure. "Interior" or "discretization" cells are cells on a component grid that can be updated using an interior discretization formula for the PDE. This means that each discretization cell has a buffer zone of cells around it of sufficient width that the interior discretization formula can be applied. Near physical boundaries of the domain, "fictitious" or "ghost" cells are added to the component grid outside the physical boundary. Boundary conditions derived from the physical boundary conditions for the problem, or using considerations based on numerical analysis, are used to update the solution values in these cells. In regions of overlap between the component grids, "interpolation" cells are included in the grid. Solution values in these cells are updated using an interpolation formula applied to a stencil of cells on an adjacent component grid. As with the ghost cells, interpolation cells are included in the grid to provide the necessary buffer zone around every discretization cell so that the interior discretization formula may be applied.

The overlapping grids used in this paper are constructed using the CMPGRD overlapping grid software developed by Chesshire and Henshaw [17], [22]. Overset grids constructed using this software have the property that they overlap the minimum amount necessary in order that essentially centered interpolation formulas can be used to transfer values between adjacent grids during a PDE solution procedure. CMPGRD automatically generates an overlapping grid along with all the data necessary for communicating data between the component grids given a set of user-specified "component" grids, each of which is a logically-rectangular grid in general curvilinear coordinates. If the original user-specified component grids overlap more than this minimum required amount, the un-needed cells are marked as "inactive" and are not used in the PDE solution procedure. In the computations presented in this paper, CMPGRD was used both as an interactive package for the construction of the initial underlying overlapping grid, and also as part of the AMR PDE solver, where it is called as a subroutine and used for embedding AMR refinement grids within the underlying overlapping grid. For this project, we modified CMPGRD to be able to insert refinement grids adaptively during a PDE solution process. The basic principles and algorithm details for overlapping grid construction are described in more detail in [17] and [22].

Because of space considerations, the detailed modifications needed for the overlap algorithm for adaptive grids will be discussed in an archival paper.

Adaptively-refined Overlapping Grids

We discuss here the procedure for advancing a PDE solution by one coarse-level timestep on an adaptively-refined overlapping grid. An adaptively-refined overlapping grid structure consists of a set of underlying curvilinear component grids $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$ that make up the "base grid" for the problem, each of which may contain embedded refinement grids.

The embedded refinement grids have the property that they are both properly nested and properly aligned with their parent component grid just as in the single grid case discussed above in the section entitled "Solution procedure using AMR on a single grid". If a region of refinement is needed that extends beyond the boundary of active cells of one of the parent component grids and into a region covered by the active cells of another parent component grid, each of the parent component grids is refined separately rather than attempting to construct a single refinement patch that covers the entire refinement region. This is an important point in our adaptive mesh procedure, since it greatly simplifies the grid construction algorithm compared to what would be required if general adaptive refinement grids were allowed. The interior cells of a refinement grid patch must lie completely within the interior cell region of its parent grid(s) at the next coarser level. If a refinement patch is created adjacent to an overlap boundary of the parent grid, its overlap interpolation cells will lie completely within the set of overlap interpolation cells for the parent grid(s) as well.

The overlap interpolation rules for overlap regions on an adaptively-refined overlapping grid specify that values are interpolated from adjacent component grids preferentially from grids at the same level, followed in preference by grids at the next coarsest level. The implicit assumption here is that cell size and aspect ratio of grids at the same refinement level on adjacent component grids is roughly the same. Thus it is most appropriate to interpolate values from adjacent grids at the same refinement level unless such a grid is not available, in which case the best possible values should be used. The proper-nesting assumption implies that in the latter case, the values will come from coarser refinement levels on the adjacent grid. While it could occur that the adjacent component grid would have values available at *finer* levels, interpolation from the finer level adjacent grids is not necessary since the values would be approximately equally degraded by interpolating directly from the fine grids as they would be by first transferring values to the adjacent coarse grid and then interpolating.

We now discuss the procedure for constructing the new refinement grid patches in the two-level refinement case where the adaptive grid hierarchy consists of the base overlapping grid together with refinement grid patches one level finer than the base grid. The solution data is assumed to be available in all cells on all grids in the adaptive overlapping grid hierarchy at the beginning of the coarse-grid timestep. As in the single-grid case, all the refinement grids at the fine level are replaced at user-specified timestep intervals using the adaptive procedure. The procedure for constructing the new refinement grid patches is essentially the same as for the one-grid case. An

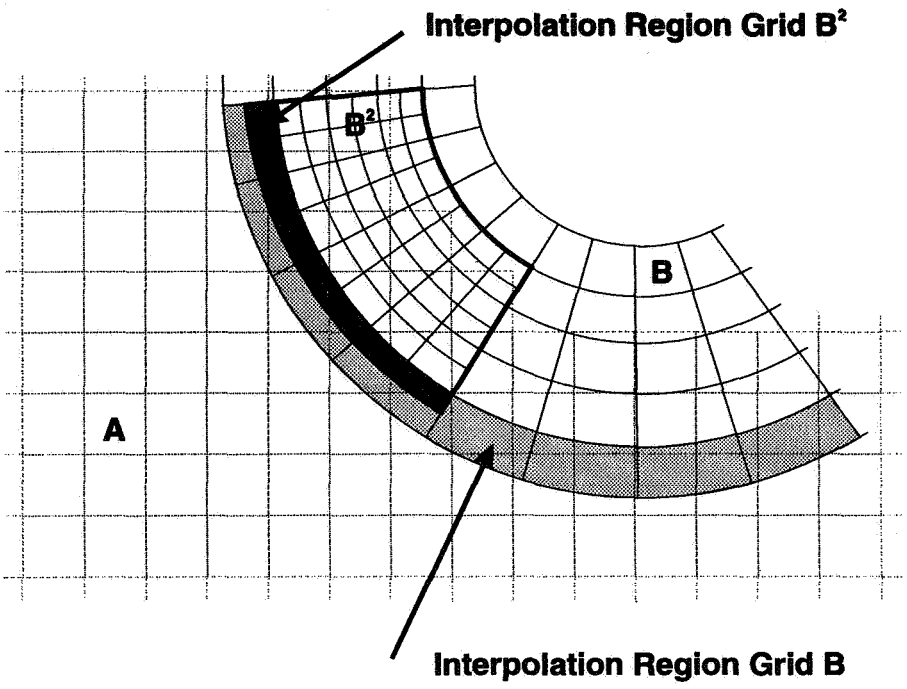


Figure 1: The interpolation cells for refinement grid B^2 lie within the interpolation region for grid B. Interior cells for grid B^2 may not lie inside the interpolation region for the coarser grid.

error estimation procedure is used on each parent grid in all interior cells, and cells of high estimated error are flagged. The set of flagged cells is then "diffused" as in the single-grid case. A difference in the overlapping grid case, however, is that we do not permit interpolation cells on the coarse grid to be flagged by either the error estimation or diffusion procedure. This is disallowed to simplify the AMR solution procedure on an overlapping grid. If the diffusion procedure indicates that an interpolation cell should be flagged, the "interpolee" cells [23] on the adjacent grid are flagged instead (If a cell on grid $\mathcal{A}$ interpolates from cells on grid $\mathcal{B}$, the interpolee cells for an interpolation cell on grid $\mathcal{A}$ are defined to be those cells on grid $\mathcal{B}$ that are included in the interpolation stencil used for determining the solution value in the interpolation cell on grid $\mathcal{A}$). This procedure of flagging interpolee cells is a logical extension to the overlapping grid case of the basic domain-of-dependence argument for the cell-diffusion process. It also provides for the expansion of a refinement region across overlap boundaries during the course of a time-dependent PDE solution, which otherwise would not take place. Figures 2–4 illustrate the procedure with the curved grid representing grid $\mathcal{A}$ and the rectilinear grid represents grid $\mathcal{B}$.

The solution on an adaptively-refined overlapping grid is updated as follows. Values are first transferred from the old adaptively-refined grid hierarchy. For interior discretization cells, this involves either copying values from grids at the same level with the same parent grid, or transferring values from grids at the next coarser level on the same parent grid. Values in interpolation cells are transferred in a different way. These values must either be copied from an old grid at the same level with the same parent component grid *or* they must be interpolated from data on an adjacent component grid according to the interpolation rules given above. It is important never to transfer coarse grid interpolation values to fine interpolation cells since a degradation of

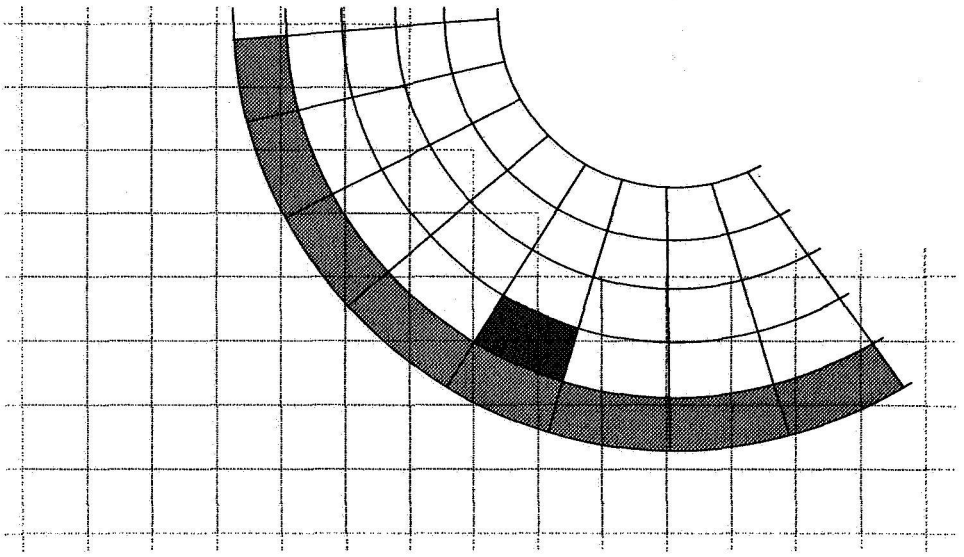


Figure 2: The cell on the curved grid shaded dark dark gray is flagged as a high error cell and will be refined.

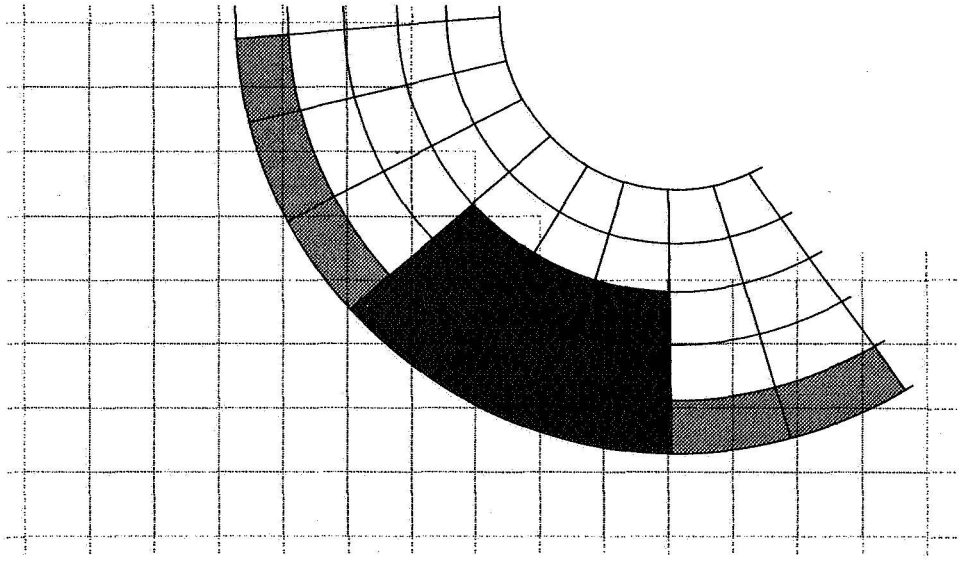


Figure 3: A single "diffusion" step of flagging surrounding cells is done. Note that interpolation cells (lighter gray) are touched by the diffusion. These cells are only tagged for the purpose of flagging the underlying "interpolee" cells and will not be refined.

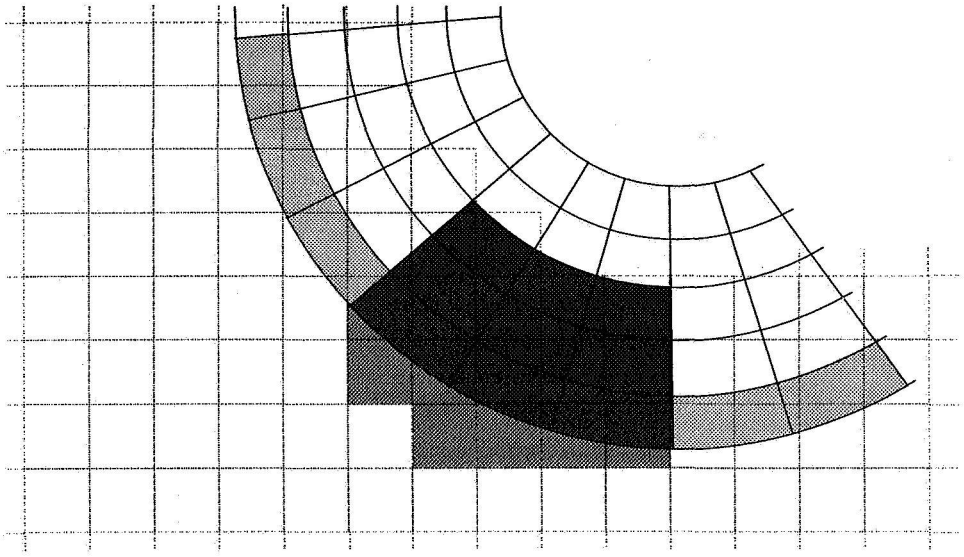


Figure 4: The cells are on the rectilinear grid are flagged (shaded an intermediate gray) because they are used for interpolation data by the flagged interpolation points on the curvilinear grid.

the computed solution can result.

Away from interpolation boundaries, the solution advance procedure is identical to the single–grid AMR procedure. The only difference in the solution advance procedure for interpolation cells is that no conservative update procedure is used. This is largely due to the fact that we feel that satisfactorily efficient methods for conservative update of overlap boundary values on general overlapping grids have not been developed. Some research has been done in this area, however, cf. [23, 24].

SOME IMPLEMENTATION DETAILS

AMR codes have greater code complexity than single or even block structured logically rectangular grid methods. The primary reason for the programming complexity is the demands made on the programming environment for dynamic allocation and deallocation of data structures. Even communication between grids at the same or different levels is nontrivial. To handle the programming complexity, we have begun to move to a programming language more capable in the manipulation of complex data structures. We use C++ [25] to handle the dynamic memory management of the data structures and FORTRAN for the numerical parts of the algorithm such as the integration. This is the first step in moving towards a C++ based programming environment that not only abstracts out the data structures but hides details of a parallel implementation as well.

C++

C++ is a superset (with some very minor exceptions) of the programming language C. For those who appreciate C, the postfix operator "++" is used to increment by one the variable it follows. Therefore C++ is literally an addon to C. C++ adds new capabilities that bring it into the realm of what is called Object Oriented Programming (OOP). OOP is a technique, discipline or style of writing programs. Here algorithms are organized around data structures called objects that both hold data and supply the functions needed to manipulate the data in safe ways. Encapsulation is often used to describe this process. The goal of OOP is to generate reusable program modules with few side effects. The idea seems a good and simple concept from a common sense viewpoint but the implementation of flexible and reusable object libraries requires a great deal of forethought and design. Practice shows us that several design iterations are often required to "get it right".

The primary way C is augmented to become a language that supports OOP is through the introduction of the data structure called a *class*. A *class*, in its most simple form, is a structure which in turn is much like a common block in FORTRAN. However, classes are used to instantiate objects by associating member functions with the class. Member functions are simply functions that operate on the data within the *class/structure*. Once classes are introduced C++ can handle two basic OOP programming paradigms. The first is called *inheritance* and the second *polymorphism*.

Inheritance is a mechanism of reuse of objects. New objects can be created from currently available objects by inheritance. The inheriting objects will have all the properties of the inherited objects plus whatever is added (data or more member functions). Inheritance facilitates a rich structure of objects through multiple inheritance (inheriting an inherited class which may in turn inherit other classes) yet allows the developer to encapsulate data at all levels. However, no programming paradigm will prevent people from writing sloppy code.

Inheritance is further enhanced by Polymorphism. Polymorphism literally means many shapes. In C++ the same function name or even operators such as "+", "*", ... can be used for many purposes. A simple example is to consider the type double, a double precision number. Doubles can be added, subtracted, multiplied, along with a host of arithmetic operations. A simple example of polymorphism is to define a new class that would represent complex numbers as a pair of real numbers. Many of the same arithmetic operators that are used to manipulate doubles can be "overloaded" to manipulate complex numbers as well. The array class library we will describe below is another example of polymorphism.

Data Structures

The implementation of the adaptive overlapping code heavily uses classes to manipulate and manage user data. In addition, inheritance is also incorporated to make the code much more ammenable to modification and/or reuse. Polymorphism is used very little at this point. However, our current direction is to use a C++ array class called A++/P++ [26] with syntax similar to F90 to develop newer versions of this and other overlapping grid codes. Here polymorphism will come into play as we are overloading the arithmetic operations found in C or C++ to manipulate multidimensional arrays.

The design of the adaptive overlapping code splits the overall algorithm into two sets of pieces. The first set is a collection of objects and functions that implement an "abstract" adaptive overlapping code. The second piece is supplied by a code developer who wants to make his or her own adaptive overlapping mesh code. Within the first collection are objects that describe the logical layout of the overlapping grids. These objects don't perform any approximation of the solution of PDEs but can be modified so that they do. Here inheritance is used. The abstract objects are inherited by developer defined objects that contain the necessary data to perform useful computations. Other objects within the abstract set develop a skeleton set of functions that give a roadmap that can be used by code developers to modify or rebuild a new algorithm. These functions are called *virtual*. When objects are inherited, the inherited object functions that are labeled virtual can be replaced by the inheriting object. This defines a clean interface between the developer and the abstract code. The developer has the flexibility to design the right functions for his or her needs and the abstract interface does not have to be changed.

The primary data structure within the abstract code is a list of objects that represent individual grids. All logical information about a component grid or any refinement is stored in what is called a *Patch* class. Logical information includes logical coordinates of a grid, interpolation information and boundary condition information. A *patchNode* is derived from a Patch so that it can be put in a linked list class called a *patchList*. This linked list contains only patches at the same refinement level on a single overlapping grid component. A *levNode* is derived from patchList to be added to a list of refinement levels on a single grid. This list class is called a *levList*. In addition, there are pointers within a levNode to point to list of patches on other components at the same level. Finally a class called *compNode* is derived from levList to be contained in a list of refinements at all levels on all components called a *compList*.

Although complex, implementation of this list structure is greatly simplified by using C++ inheritance. This allows the developer to concentrate on numerical algorithms rather than the data management aspects. Figure 5 illustrates the complete hierarchy.

The actual grid generation is performed by calling as a subroutine the CMPGRD mesh generation code [17] modified to generate mesh refinements as discussed in the previous section. Finally, many smaller objects and functions are used to help in regridding and managing refinements within components. These objects came from a very useful and reusable C++ library from the Center for Computational Sciences at Lawrence Livermore National Laboratory called BoxLib [27].

NUMERICAL RESULTS

The computational examples presented in this paper are two dimensional simulations of compressible fluid flow as described by a numerical approximation to the compressible Euler equations. The code runs both two- and three dimensional problems but three-dimensional results will be described in a separate journal article because of space limitations. In general curvlinear coordinates, these are given by

$$\partial_t \mathbf{u} + J_\xi \sum_{\ell=1}^{n_d} \partial_{\xi^\ell} \mathbf{F}^\ell(\mathbf{u}) = 0. \tag{1}$$

Grid Data Structure Hierarchy

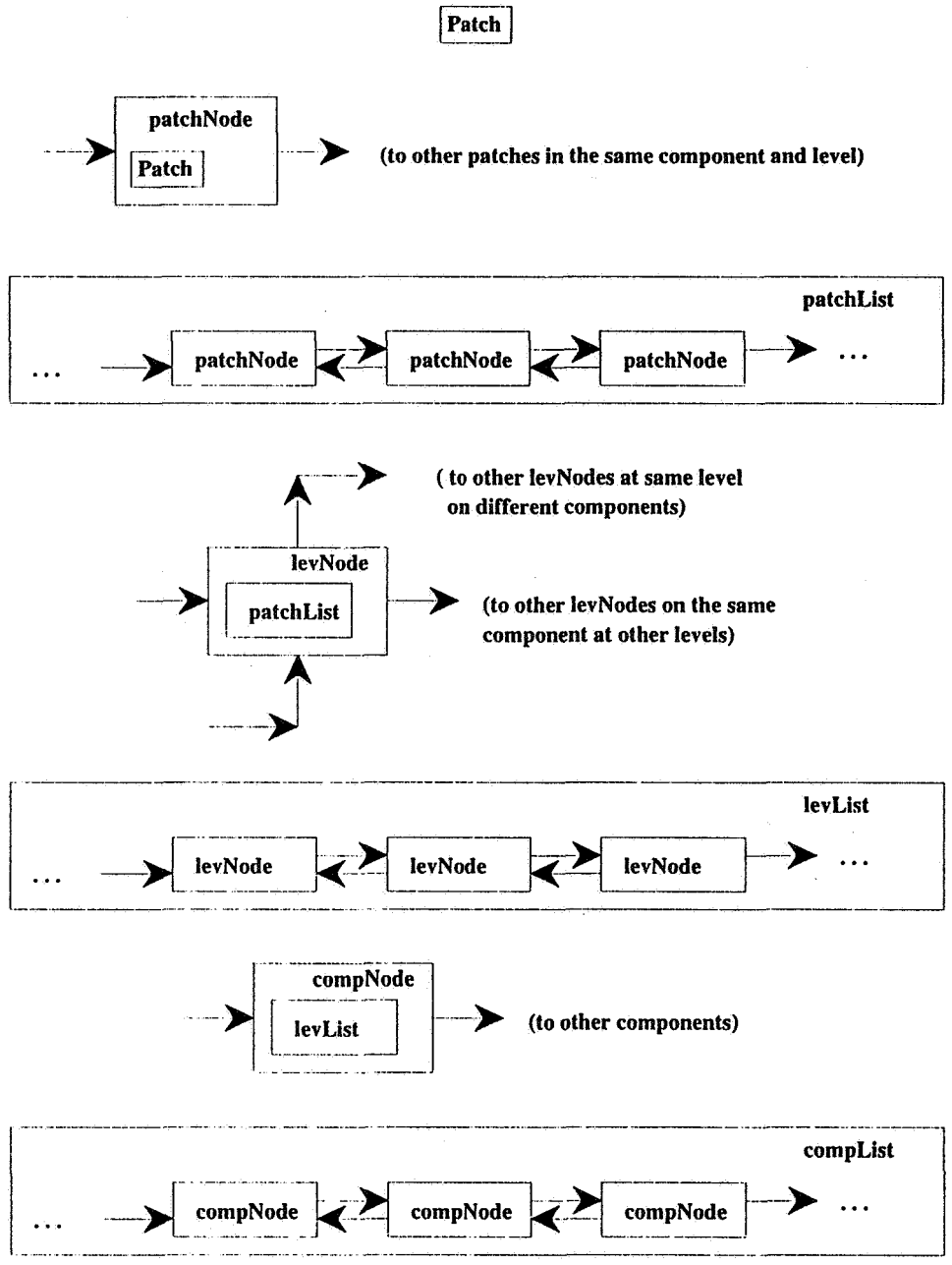


Figure 5: The list elements from the top are incorporated into more complex data structures further down.

Here

$$\mathbf{F}^{\ell}(\mathbf{u}) =: \begin{pmatrix} \rho U^{\ell} \\ u\rho U^{\ell} + \tilde{\xi}^{\ell}_{x} p \\ v\rho U^{\ell} + \tilde{\xi}^{\ell}_{y} p \\ w\rho U^{\ell} + \tilde{\xi}^{\ell}_{z} p \\ U^{\ell}(\rho E + p) \end{pmatrix}, \ell = 1,2,3, \tag{2}$$

$$U^{\ell} := \sum_{i=1}^{n_d} \tilde{\xi}^{\ell}_{x^i} u^i$$

are the contravariant velocities, and $J_{\xi} := det|\frac{\partial \xi}{\partial x}|$ is the determinant of the coordinate transformation from Cartesian coordinates $\mathbf{x} := (x^1, x^2, x^3)$ to curvilinear coordinates $\xi := (\xi^1, \xi^2, \xi^3)$. The dependent varibles are the density ρ, the three components of velocity $u^i, i = 1,2,3$, and the energy E. The pressure, p is related to the other variables through the equation of state for an ideal gas: $p = (\gamma - 1)(\rho E - \frac{1}{2}\rho \sum_i (u^i)^2)$. The finite difference method used for the computations is based on the conservative cell-centered upwind-centered Godunov method in [20]. It has been modified to use a linearized approximate Riemann solver described in unpublished work by Colella, Glaz and Ferguson. (The original method in[20] used a computationally more expensive approximate flux function based on [28].)

Several test problems were outlined, in advance, by the conference organizers to stimulate discussion at the workshop. We chose to compute the double wedge geometry where an oncoming Mach 2.16 shock hits a wedge at an angle of 20 degrees followed by a wedge at an angle of 50 degrees three horizontal units later. Quiescent preshock values are unity in the pressure and density. The ratio of the specific heats (γ) is 1.4. Because the geometry was simple enough, two computations were preformed. The first computation used a single grid that was deformed to fit the double wedge geometry. The second computation uses two component grids — the first grid being a cartesian grid and the second grid conforms to the wedge boundary cutting away the cartesian grid.

Figures 6 and 7 show the grids and density for times near 2.5 time units. The solutions are very nearly the same in structure. However, single grid computation ran in twice the number of cycles that the two grid computation did. This is primarily caused by the timestep in the single grid case being unnecessarily CFL limited in the upper right hand corner of the computation. The two grid case has uniform cell sizes throughout the computational region. Another issue brought out by these computations is conservation. The two grid computation is not conservative yet still matches the single conservative grid case. This reflects our experience that if care is taken in making sure that cells sizes don't vary greatly from component to component then nonconservative interpolation is sufficient for computational purposes.

The last subject to discuss is overall performance of the adaptive algorithm. At this point the time spent in the integrator is less than 50% of the entire run time. This is primarily due to the current implementation of the composite grid generation package. This package handles each point on each grid separately instead of processing a list of points in a vectorized manner. We are

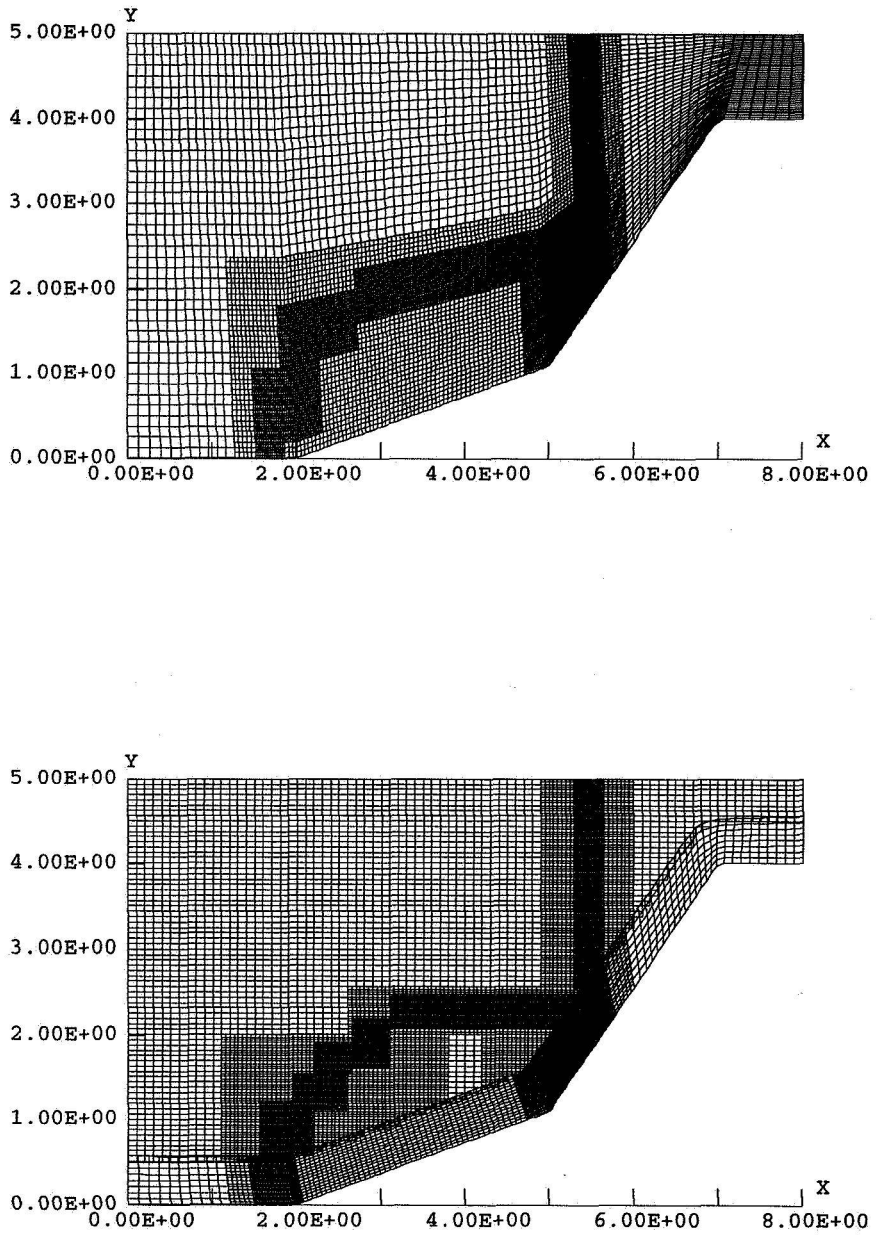


Figure 6: The adaptive grids for a single base grid (top) and two base grid (bottom) computation near time 2.5.

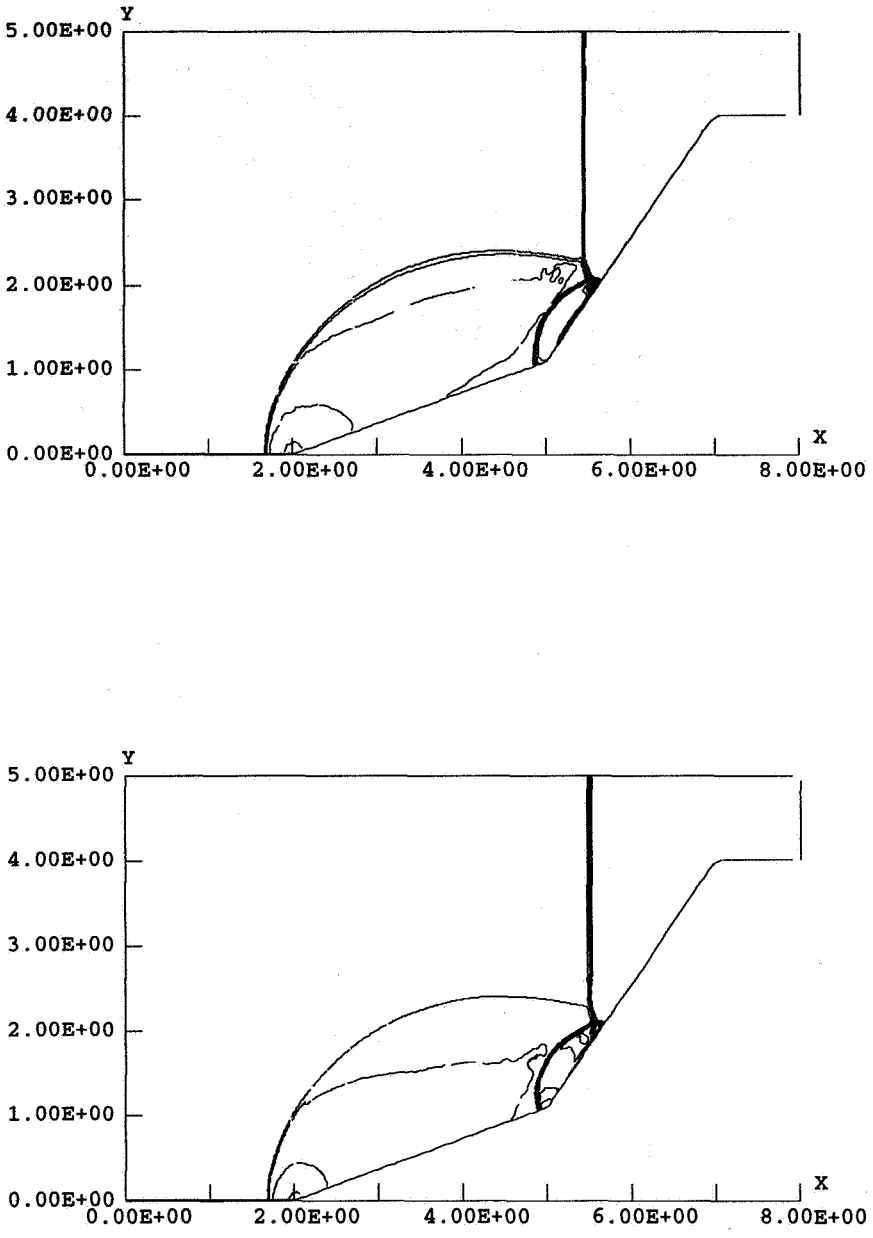


Figure 7: The density contours for a single base grid (top) and two base grid (bottom) computation near time 2.5.

currently rewriting the mesh generation package so that it can achieve vector performance.

*

References

[1] M. J. Berger and J. Oliger. Adaptive mesh refinement for hypberbolic partial differential equations. *J. Comp. Phys.*, 53:561-568, March 1984.

[2] M. J. Berger and A. Jameson. Automatic adaptive grid refinement for the Euler equations. *AIAA Journal*, 23:4:561-568, 1985.

[3] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64-84, 1989.

[4] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.*, 15:127-138, 1994.

[5] E. G. Puckett and J. S. Saltzman. A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics. *Physica D*, 60:84-93, 1992.

[6] M. J. Berger and J. Saltzman. AMR on the CM-2. *Applied Numerical Mathematics*, 14:239-253, 1994.

[7] M. J. Berger and R. Leveque. Cartesian meshes and adaptive mesh refinement for hyperbolic partial differential equations. In B. Engquist and B. Gustafsson, editors, *Third International Conference on Hyperbolic Problems*, pages 67-73. Chartwell-Bratt, 1991.

[8] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. Adaptive Cartesian grid methods for representing geometry in inviscid compressible flow. In *Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference*, pages 530-539. AIAA, June 1993.

[9] G. Starius. Composite mesh difference methods for elliptic boundary value problems. *Num. Math.*, 28, 1977.

[10] G. Starius. On composite mesh difference methods for hyperbolic differential equations. *Numer. Math.*, 35:241-255, 1980.

[11] B. Kreiss. Construction of a curvilinear grid. *SIAM J. Sci. Stat. Comput.*, 4:270-279, 1983.

[12] J. A. Benek, J. L. Steger, and F. C. Dougherty. A flexible grid embedding technique with application to the Euler equations. *AIAA paper 831944*, 1983.

[13] J. A. Benek, P. G. Buning, and J. L. Steger. A 3-D Chimera grid embedding technique. In *Proceedings of the 7th AIAA Computational Fluid Dynamics Conference, Cincinnati*, pages 322-331. AIAA, 1985.

[14] P. G. Buning, I. T. Chiu, S. Obayashi, Y. M. Rizk, and J. L. Steger. Numerical simulation of the integrated space shuttle vehicle in ascent. AIAA paper 88-4359-CP, AIAA, 1988.

[15] F. C. Dougherty and J-H Kuan. Transonic store separation using a three-dimensional Chimera grid scheme. AIAA paper 89-0637, AIAA, 1989.

[16] G. S. Chesshire. *Composite Grid Construction and Applications.* PhD thesis, California Institute of Technology, 1986.

[17] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comp. Phys.*, 90(1):1–64, 1990.

[18] D.L. Brown, G. Chesshire, W.D. Henshaw, and H.O. Kreiss. On composite overlapping grids. In *Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems*, 1989.

[19] J. F. Thompson. A composite grid generation code for general 3D regions - the Eagle code. *AIAA J.*, 26:271, 1988.

[20] D. L. Brown. An unsplit Godunov method for systems of conservation laws on curvilinear overlapping grids. *Mathl. Comput. Modelling*, 20(10):29–48, 1994.

[21] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comp. Phys.*, 87:171–200, 1990.

[22] D. L. Brown, G. Chesshire, and W. D. Henshaw. Getting started with CMPGRD, introductory user's guide and reference manual. LANL unclassified report LA-UR-90-3729, Los Alamos National Laboratory, 1990.

[23] G. Chesshire and W. D. Henshaw. A scheme for conservative interpolation on overlapping grids. *SIAM J. Sci. Comput.*, 15(4):819–845, July 1994.

[24] M. J. Berger. On conservation at grid interfaces. *SIAM J. of Numer. Anal.*, 24:967–984, 1987.

[25] Bjarne Stroustrup. *The C++ Programming Language.* Addison Wesley, second edition, 1992.

[26] R. Parsons and D. Quinlan. Run-time recognition of task parallelism within the p++ parallel array class library. Technical Report LA-UR-93-3856, Los Alamos National Laboratory, October 1993. proceedings of the IEEE Conference on Scalable Parallel Libraries at Mississippi State University, Oct. 6-8, 1993.

[27] M. Welcome, W. Crutchfield, C. Rendleman, J. Bell, L. Howell, V. Beckner, and D. Simkins. *BoxLib User's Guide and Manual.* Lawrence Livermore National Laboratory, Lawrence Livermore National Laboratory, Livermore, CA 94550, version 0.02 beta edition, September 1994. A Library for Managing Rectangular Domains.

[28] J. Bell, P. Colella, and J. Trangenstein. Higher order Godunov methods for general systems of hyperbolic conservations laws. *J. Comp. Phys.*, 82:362–397, 1989.

MOVING AND ADAPTIVE GRID METHODS FOR COMPRESSIBLE FLOWS

Jean-Yves Trépanier and Ricardo Camarero
Department of Mechanical Engineering
École Polytechnique de Montréal, C.P. 6079, Succ. Centre-Ville,
Montréal, Québec, Canada, H3C 3A7

SUMMARY

This paper describes adaptive grid methods developped specifically for compressible flow computations. The basic flow solver is a finite-volume implementation of Roe's flux difference splitting scheme on arbitrarily moving unstructured triangular meshes. The grid adaptation is performed according to geometric and flow requirements. Some results are included to illustrate the potential of the methodology.

INTRODUCTION

A large number of engineering flow problems are concerned with the numerical simulation of unsteady compressible flows in complex geometries with moving boundaries. Examples are internal gas dynamics with pistons, external flows with bodies in relative motion (store separation, etc..). Our own motivation was related to the prediction of the internal flow in a circuit-breaker, which involves electrodes and piston in relative motion [1].

The computational tools required to tackle these type of problems are still a research area. Only from the grid point of view, different schools can be found ranging from overset structured grids to global unstructured grid remeshing at each time step, and the research is still very active in this domain.

Our own approach is to use an unstructured triangular grid. This choice was driven by many factors. First, triangular grids offers a great flexibility in gridding complex geometries with various length scales; second, their potential for automation and adaptation is clear; third, it simplifies the coding of the flow solver which has no special cases to handle. From this choice, we also select to perform adaptation by modifying thegrid with local actions because in many problems, only a small portion of the grid need to be modified when adaptation is done. The flow solver also need to take into account properly the grid motion and this was assured using an ALE version of Roe's flux difference splitting scheme.

This technology has enable the investigation of various adaptation strategies, including a novel shock fitting approach where the discontinuities are captured at the interfaces of two triangles.

The paper is organised as follows : we first give a description of the grid management algorithm, followed by a few words about the moving grid flow solver. We then present various adaptive strategies using grid relocation and grid enrichment. Finally, some conclusions are drawn.

Temporal Evolution of the Grid

A set of curves serves to describe the geometry and its evolution is described in terms of the velocity of these curves. The temporal evolution of the grid is performed in two steps. First, one computes the effect of the moving curves on the grid and second, a smoothing term is added. According to this, the velocity of the grid nodes can be represented by:

$$w = w_g + w_s$$

where w_g is the geometric grid velocity and w_s the smoothing grid velocity. Both of these terms must respect the boundary conditions defined by the movement of the curves.

The geometric term w_g depends on the two types of curve-node interaction considered: Dirichlet and Neumann. In a Dirichlet curve-node interaction, the velocity of the grid nodes that lie on a moving curve is set equal to the velocity of the curve. There is no relative motion of the nodes with respect to the curve. In a Neumann curve-node interaction, the nodal velocity is set equal to the normal component of the curve velocity at the position of the grid node. This is the minimal constraint which can be imposed on the grid node in order to remain on the curve.

The last situation to be considered is the curve-curve interaction. This happens when a grid node is located at the intersection of two curves. This type of node will be constrained to remain on the intersection of the two curves, and its velocity is simply set equal to the velocity of the intersection of the two curves.

In addition, a smoothing of the grid velocity is performed which consists in assigning to the internal nodes the mean velocity of their direct neighbors. This procedure is repeated for a few iterations which normally is also limited to some selected nodes located near the moving curves. The final result is a diffusion-like operator which smoothes out the large variations in grid velocity and that was found effective for the type of computations that were conducted.

The purpose of the smoothing term w_s is to produce an additional smoothing of the transient grid evolution by improving the grid quality by consideringthe node displacement. The new position of the grid nodes is obtained as the average of the position of their neighbors. The velocity of the grid nodes is then calculated by dividing the node translation by a time interval, which is chosen to be the non-dimensional time scale of the problem. When this action is applied on nodes located on a Neumann-type boundary curve, the resulting smoothing grid velocity w_s must be tangential to it. Consequently the normal component of w_s is dropped out and the nodes will slide on this curve.

Grid Generation

The generation of stretched triangular grids will be performed using an incremental algorithm which uses local actions on the grid to obtain, from a given triangulation, a new triangulation with the required properties. The different local actions on the grid are driven by a definition of the quality of the triangles and the different procedure will have different roles towards the reaching of the objective. For the purpose of clarity, the quality will first be defined for an isotropic, or non-stretched, grid and then generalized to an arbitrarily stretched triangulation.

Let us assume that we have a list of nodes $\mathbf{N}$ and assume that we also have an element list $\mathbf{T}$ giving the connectivity of the triangulation. A triangular element is defined by three points $\boldsymbol{r}_1$, $\boldsymbol{r}_2$

and $\boldsymbol{r}_3$ in a counter-clockwise direction, while its side vectors are defined by $\Delta\boldsymbol{r}_1$, $\Delta\boldsymbol{r}_2$ and $\Delta\boldsymbol{r}_3$. We then define:

$$\begin{aligned} A &= \sqrt{3}\Delta\boldsymbol{r}_1 \times \Delta\boldsymbol{r}_2 \\ B &= \tfrac{1}{2}\sum_{i=0}^{3} \Delta\boldsymbol{r}_i \cdot \Delta\boldsymbol{r}_i \end{aligned} \tag{1}$$

A is proportional to the Jacobian of the triangle (twice its area) while B is the so-called potential energy of the triangle [2]. The dimensionless quantity

$$Q = \frac{A}{B} \tag{2}$$

varies from zero to one can be used as a measure of the equilarity of the triangle.

Definition of Stretching

The stretching of any triangle can be simply defined by considering its transformation into an equilateral triangle. Such a transformation is built from a rotation of the system of axis to a new axis (x', y') followed by a scaling by a factor $1/E$ in the direction x'. This couple (E, θ) can thus be used as the definition and measure of triangle stretching.

Quality of Stretched Triangles

The quality of a stretched triangle can now be computed. First, one has to define an objective which, in the isotropic case, was implicitely an equilateral triangle. One thus needs a spatial distribution of stretching amplitude and orientation which is considered as data from the mesh generation point of view. The quality of stretched grids can now be measured: we first apply the transformation with the couple (E_o, θ_o) to a triangle and compute the quality of the resulting triangle in the transformed plane as:

$$\overline{Q} = \frac{\overline{A}}{\overline{B}}.$$

Local Actions on the Triangulation

Several basic actions on the triangulation are performed to make the grid closer to the objective. A remeshing algorithm based on the successive application of these operators is described in ref. [3] for isotropic grids.

The refinement of the grid is obtained through triangle subdivision. A triangle requiring to be refined is branched into two triangles by cutting it on it's longest side. The lengths of the side of the triangle are measured in the transformed plane.

The coarsening of the grid is performed through node removal followed by a local remeshing. The removal of a node in the triangulation leaves an open polygon. This polygon is then retriangulated by recursively removing from it the triangle with the highest quality, until only four nodes are left. The placement of the last diagonal is performed according to the algorithm of diagonal swapping, described below. This process is influenced by the stretching requirements by retriangulating the open polygon in the transformed plane.

The swapping of diagonal is a well known technique to obtain a Delaunay triangulation from an existing triangulation [4]. It consist in examining each pair of adjacent triangles and to select from the two possible configurations for the diagonal side, the one which maximize the minimum angle of the triangulation. This procedure is repeated until no more swapping occurs. Generalized Delaunay triangulation have been proposed which introduces some notion of space transformation [5, 6, 7, 8]. In the current implementation, each valid diagonal side is examined for swapping and the configuration which maximize the minimum quality is choosed, where the qualities are measured in the transformed plane.

A coarse-cure procedure has been implemented, which examines the triangulation and marks triangles with a small quality (below 0.4) as "bad" triangles. Then, for each bad triangle, the node opposed to the longest side(measure in the transformed plane) is deleted.

The Remeshing Algorithm

The goal of the remeshing algorithm is to produce a triangulation meeting these required area and this, starting from the current triangulation and using the basic tools previously described. The proposed remeshing algorithm is described in ref. [9]. The first step determines which triangles requires refinement or coarsening and a corresponding code is attributed to each triangle. In practice, these actions are discrete operations on the grid and some care must be taken in setting the triangle code to avoid possible oscillations in the remeshing process, i.e. to insure a quasi-smooth grid convergence. To do so, one has first to evaluate the average performance of the two basic operators, the refinement and the coarsening. The refinement operator produces triangles of area half of their parent. The coarsening operator which, in the average, will operate on nodes surrounded by 6 triangles, produces new triangles areas of about 1.5 times the average parent area. From this basic data, the code on triangles have been set according to the following inequalities:

IF actual area $>$ 3/2 required area THEN set a refinement code
IF actual area $<$ 3/4 required area THEN set a coarsening code

Geometric Requirements

The computation of flows in complex geometries with moving boundaries must also take into acount the geometric requirements. As discussed in ref.[10], these requirements are governed by two different aspects of the computational domain: the curvature of the bounding curves and the proximity of the various parts of the domain.

An automatic method for computing these requirements has been described in ref [10]. The method defines a reference grid density which respects th geometric requirements from both the curvature and proximity point of view.

Adaptivity and Flow Coupling

Error Estimation

The principle of estimating the error by projection of the solution in a higher order subspace has been used in the present work. Starting from an existing flow solution, which is piecewise constant in each triangle, a projection to a piecewise linear solution is performed using the technique of Barth [11]. The error in the solution in each triangle is then estimated to be the integration of the difference

between the linear and constant solutions. In addition, since the error is estimated to be proportional to the grid size (for the first order implementation of the Roe scheme), the required areas are obtained by scaling.

Some peculiarities of compressible flow solutions must nevertheless be taken into account when one tries to use directly this type of error estimation. In practice, and due to the intrinsic nature of compressible flow solutions, very high ratios of minimum and maximum required areas for triangles will be obtained. This results in extremely small triangles in regions of high gradients of the solution and these will reduce strongly the convergence of the computation. To overcome this problem, some limits on smaller and larger triangles in the computational domain must be imposed.

Grid Control Strategy

We propose to start by devising a initial grid for the solution process, which will be called the "reference grid". The limits on the smallest and the largest triangle area are then specified in terms of a fraction of the reference grid, and are thus locally defined. For computations in geometries which need very high ratios of initial grid sizes, this approach is more flexible than the specification of absolute minimum and maximum sizes. Another advantage of this approach is the ability to deal with both the geometric and flow grid requirements during a transient solution process.

The frequency of remeshing is determined by two conditions: the geometric requirement and the flow requirement. A geometric remeshing is performed each time that the Δt_{Grid}, which is defined as the minimum time interval needed to reduce one of the triangle areas by one half [12], is reached. In this grid adaptation step, very few triangles are normally involved. A fluid remeshing is carried out after a certain number of iterations on the flow solution. The frequency of this action is determined by a user controlled variable, as are the minimum and maximum area ratio limits.

Details about the grid management algorithms can be found in refs. [3, 9, 13, 14, 10].

FLOW SOLVER

The mathematical model describing an inviscid thermally nonconducting perfect gas is given by the Euler system, which can be written for a general moving (or non-moving) reference frame in integral form as:

$$\frac{\partial}{\partial t}\int_{V(t)} U dV + \oint_{S(t)} \boldsymbol{n}\cdot\boldsymbol{F} dS = \int_{V(t)} f dV \tag{3}$$

where $U^T = [\rho, \rho\boldsymbol{u}, \rho E]$ is the vector of dependent variables, with ρ, the density, $\boldsymbol{u}$,the fluid velocity, and E the specific energy. The term $\boldsymbol{F}^T = [\rho(\boldsymbol{u}-\boldsymbol{w}), \rho(\boldsymbol{u}-\boldsymbol{w})\boldsymbol{u} + \boldsymbol{I}p, \rho(\boldsymbol{u}-\boldsymbol{w})E + \boldsymbol{u}p]$ is the flux tensor,where $\boldsymbol{w}$ is the mesh velocity, p is the pressure, and $\boldsymbol{I}$ is the unit tensor. The variable $\boldsymbol{n}$ indicates the outward unit vector normal to the boundary. The symbol f denotes external sources from the physics or from the axisymmetric formulation. In this case, $f^T = [0, \boldsymbol{e}_y p/y, 0]$. These conservation laws are completed by the equation of state $p = (\gamma-1)\rho E$.

The associated discrete approach to the above integral equations is referred to as a Finite-Volume method. For the case of non-moving meshes, with no source terms, and using an explicit procedure, the variables U^{n+1} are updated by:

$$U^{n+1} = U^n - \frac{1}{V^n}\left(\sum F_i(Q_i)\right) \tag{4}$$

where F_i represents the discrete flux through a face B_i during a time interval Δt and V^n, the volume of the cell.

Among the different possibilities to obtain the flux vectors on the cell faces, the methods based on the solution of a set of local Riemann problems are used frequently. As their exact solution is costly, several approximate alternatives have been proposed, one of these being introduced by Roe [15].

Roe's Scheme for Moving Grids

The ingenuity of Roe's procedure relies on the definition of an average state A^* which approximates the Jacobian $A = \frac{\partial F}{\partial u}$ of the equation $\frac{\partial U}{\partial t} = \frac{\partial F}{\partial x} = A\frac{\partial U}{\partial x}$. This average state can be obtained on the basis of the quadratic character of the variables: $\sqrt{\rho}, \sqrt{\rho u}, \sqrt{\rho v}, \sqrt{\rho h}$. Using this information, averaged right eingenvectors e_k, eigenvalues λ_k,and wave strengths α_k can be obtained. Then it is possible to define the flux at a face, say, i+1/2 as:

$$F_{i+\frac{1}{2}} = \frac{1}{2}[F_{i+1} + F_i - |\Delta F|_{i+\frac{1}{2}}] \quad \text{with} \quad |\Delta F|_{i+\frac{1}{2}} = \sum \alpha_k |\lambda_k| e_k \tag{5}$$

This method can be extended to moving grids in a simple manner. For example, for a grid node moving with a velocity w,the wave speed $\lambda_1 = (u - a)$ (where a is the speed of sound), becomes: $\lambda_1 = (u - a - w)$.On the other hand the flux $F(u)$ now transforms to $F(u - w)$. In this respect there are two fundamental remarks to be done. First, this modification *only affects the convective terms.* Second, the grid motion is characterized by the face velocity which is defined by:

$$w = \frac{\Delta V}{S\Delta t} \tag{6}$$

where S represents the face area at a given time, and ΔV the volumetric increment along a face. Details of this fundamental approach are given in [12].

Applying these ideas, the updated variable U^{n+1} can be computed by:

$$U^{n+1} = \frac{V^n}{V^{n+1}}\left[U^n - \frac{1}{V^n}\left(\sum F_i(Q_i - \Delta V_i)\right)\right] \tag{7}$$

It can be realized that the term in brackets corresponds to the advanced flow variable U^{n+1} computed after Eq. 4, with the term Q_i modified to $Q_i - \Delta V_i$. More details concerning the extended Roe's scheme for moving grids can be found in ref. [16].

ADAPTIVE METHODS

Grid Optimization

A spring smoothing scheme is derived by minimizing the function that represents the total potential energy of the triangulation given by:

$$\Phi = \sum_{\mathbf{T}} \kappa B \tag{8}$$

where κ is a penalty for the spring system. The Laplacian smoothing scheme is obtained by using $\kappa = 1$. However, the penalty can be accomplished differently for a better control of the grid. Such a penalty was introduced by Kennon and Anderson [2] to treat the case of non-convex domains and

$\kappa = 1/A$ was used. In the current work, the spring constants are choosed as a function of the qualities, such that:

$$\kappa = \phi(Q) \tag{9}$$

where $\phi(Q)$ is a function that depends only on the quality of the triangle. Since the Eq. 8 is continuous with respect to the position of the nodes $\boldsymbol{r}$, a minimum of the function will be found when the gradient of Eq. 8 with respect to $\boldsymbol{r}$ is zero.

Optimization of Stretched Triangles

Following the previous discussions and definitions, a generalized form of the non-linear spring system given by Eq. 8 and 9 can be obtained by the minimization of the function:

$$\Phi = \sum_{\mathbf{T}} \phi(\overline{Q})\overline{B} \tag{10}$$

Minimization Methodology

The previously defined optimization problem is then solved using the gradient method of steepest descent. It is well known that this simple algorithm can converges very slowly but it is sufficient to test the formulation of the problem and the implementation of some more sophisticated optimization strategies are reported to a future work.

Details about the various adaptive strategies can be found in refs. [17, 18]

Example

The optimization strategy is applied to the computation of a shock reflection problem. The initial grid and solution are reproduced on Fig. 1. The shock is diffused over two or three cells. Starting from this solution, requirements on grid stretching and orientations have been set according to the gradient of the density.

Figure 2 illustrate the final grid and solution. A comparison of the optimized and initial grid is presented on Fig. 3 where it beomes evident that this type of adaptation is a serious alternative to grid refinement.

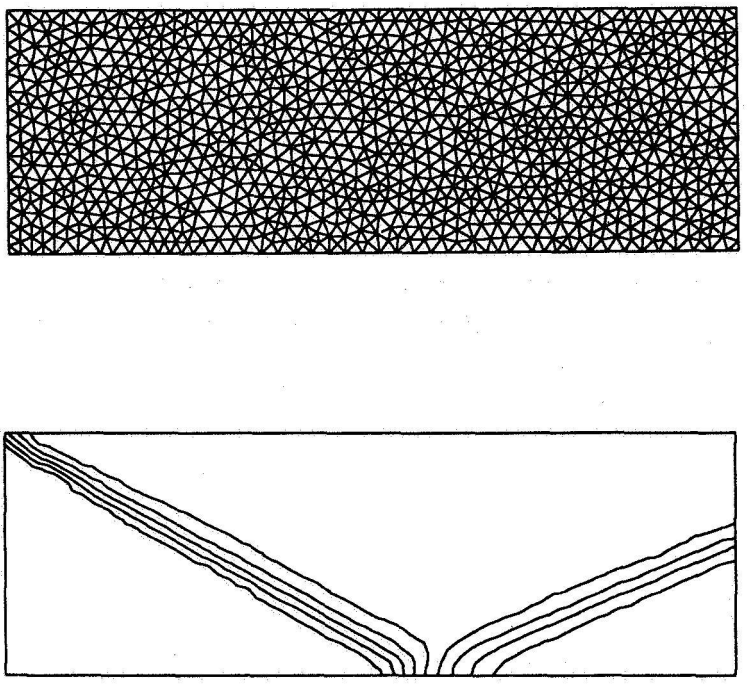


Figure 1: Initial grid and isoMach lines.

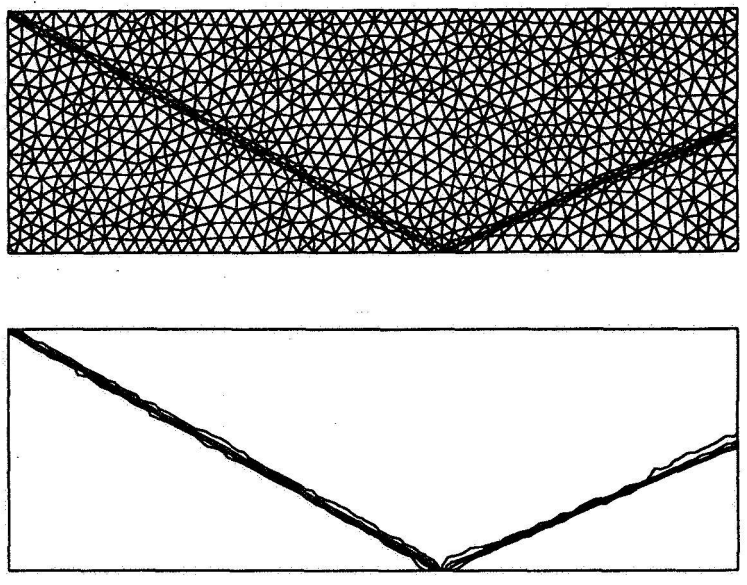


Figure 2: Optimized grid and isoMach lines.

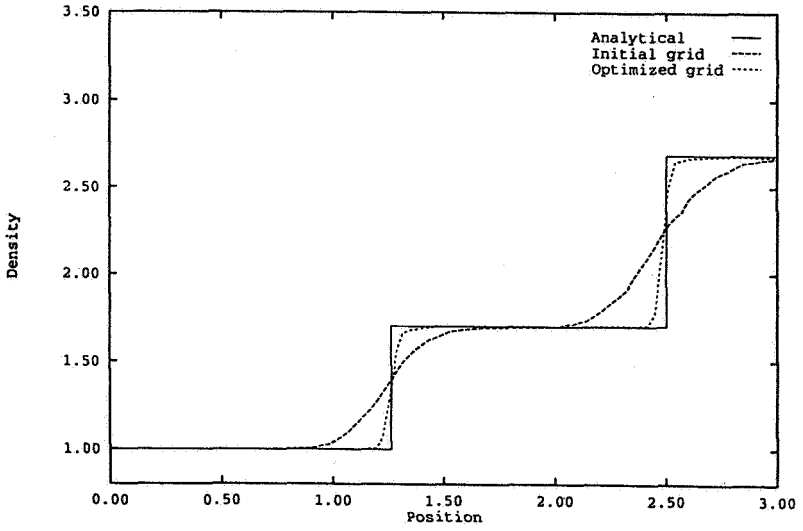


Figure 3: Density profiles for the initial and optimized grids.

Shock Fitting

The adaptation to discontinuities such as shock waves has been a persistent problem in the numerical simulation of compressible flows. The weakness of the shock capturing approach is that the shocks are captured over several grid points. While the sharpness of these discontinuities can be improved by refining the grid in the vicinity of these regions, this leads to computing costs due the large number of elements and the decrease of the global time step particularly in 2 or 3D space dimension. The shock-fitting approach and recent variants such as the λ-scheme have fared better on the second count with some severe limitations on the topology of interaction patterns.

Resolving these problems requires addressing some fundamental and technological issues relating to the correct computation of shock discontinuities and their detection. Until recently it was felt that applying the Rankine-Hugoniot relations was the only way to achieve the exact jump conditions. With the Roe scheme, it is possible, although this is not generally appreciated, to obtain the exact jump provided the shock and the cell face are aligned. The problem of shock detection and tracking does not have rigourous foundations and is still largely based on heuristics. However a recent study [19] has proposed a model for the wave propagation phenomena. In this model three basic waves are identified and relations to compute the directions and strengths of these from the basic variables are given. It is possible, to extract information from the flow field this model to align locally the mesh. Coupled with a grid adaptivity algorithm, it is felt this model can produce the local grid alignment to allow the correct jump calculation and sharp shock resolution by the Roe scheme.

In the present section, a methodology is described to perform these tasks. The basic idea in the present method is to adapt dynamically the mesh to fit discontinuities in the flow. This involves

two fundamental capabilities: the first one is to detect accurately the various wave patterns of the flow; the second one is to perform the required actions on the grid to align it with discontinuities. The adjusment of the grid must be carried out without perturbing the solution because the method is to be applied to unsteady flows and because the convergence of the process will be improved if unphysical perturbations are avoided.

In the present work, feature detection is performed using the wave model proposed by Roe. This model has demonstrated its accuracy to capture oblique shock waves as well as contact discontinuities. A full description of this wave model is found in Ref. [19]. Application of this wave model requires the flow gradients. In the present finite-volume scheme, the flow properties are piecewise constants for the first order scheme and piecewise linear for the second order scheme and for both schemes are stored at the centers of the triangles. The flow gradients are computed at the triangle center using a standard Gauss quadrature involving all the triangles sharing a common node with the considered triangle.

Identification of the Flow Features

The wave model used in this study is based on a superposition of linear waves and is not capable of representing genuinely non-linear waves and discontinuities. But this is not critical because it is not used for that purpose. What is required is the detection of a dominant wave and its angle. On the other hand, when the model is applied in regions of discontinuities such as shocks or slip lines, a correct physical behaviour will be captured by the model. More specifically, a shock wave will be seen as a strong acoustic wave, a slip line will be represented by a shear wave and a moving contact discontinuity as an entropy wave. This correspondance is at the basis of the detection algorithm.

The detection process involves the filtering of the waves which comprises two operations. First, the weak waves are discarded, based on the relative strength of each wave. The criteria for this step has been fixed at ten percent of the maximum wave intensity over the whole domain. Second, only one wave needs to be selected for each triangle. In this case, a wave is retained if it has a strength of an order of magnitude greater that the other waves in the same element.

After this process, most of the triangles will have their waves discarded, except triangles near discontinuities, dividing the whole triangulation in two groups: the active group comprising triangles with only one strong wave and the non-active group comprising triangles without a dominant wave.

Adaptation

The grid management is a critical aspect of the algorithm. It is performed with three basic actions: i) orientation of some edges of the triangulation to align them perpendicular to the wave direction; ii) translation of the edges to follow moving discontinuities; and iii) removal ill shaped triangles. In addition a grid adaptivity procedure can be superimposed on these algorithms.

Orientation of the edges The orientation of the edges is obtained from the output of the feature detection phase of the method based on the wave model. As described in section 3.2, this is a set of triangles for which a dominant wave has been identified. Only the orientation of this wave is used to modify the orientation of the grid.

The list of triangles in the active group is converted into a list of active edges. For each triangle, an edge is selected which is the most perpendicular to the main wave; and the an edge becomes active only if it is selected by its two neighbor triangles. With this list of active edges, an attempt is made to orient these edges perpendically to the dominant. As conflicting requirements can result from different wave directions, this is carried out globally through an optimisation procedure. A

function that represents the vector product between the normalized wave orientation vector and the normalized side vector is constructed. The minimization is performed using a gradient method based on the steepest descent technique.

Translation of the edges The translation is performed to move the edges directly on the shock or slip lines. One is reminded that the third condition on Roe's average state matrix is another form of the Rankine- Hugoniot condition. This is obtained by adding at each node the velocity of the main wave in the direction detected by the wave model. For this action, the velocity of the main wave is taken from the flux eigenvalues provided by Roe's scheme which are more accurate than the wave speeds computed by the wave model, because of the Gauss quadrature required for the latter. For steady discontinuities the movement converges to an accurate positioning of the edges directly on the discontinuities. For unsteady flows, the velocity obtained at each node is the sum of two velocities, one of which follows the normal movement of the discontinuity and the other that rotates the edge about the discontinuity.

Flow Over a Wedge

This first test case will be used to illustrate how the method works. It consists of a Mach 2 flow incident over a 10 degree wedge. The effect of the various actions involved in the process of grid adaptation will be investigated in a systematic way. The starting point of the adaptation process is the grid and solution represented on fig. 4. The oblique shock wave is captured by the scheme and extends over approximately two to three cells. In a first computation, the method was used without cure and adaptation. This means that the grid connectivity remains unchanged as the grid nodes move. After a few time steps, the grid motion and the optimization phase of the algorithm have almost succeeded in aligning the grid with the shock wave. This is illustrated on fig. 5 together with the current grid velocity, as computed by the algorithm.

After a few hundred time steps, some triangles tend to degenerate along the shock line, as shown in fig. 6. This is attributable to the translation grid velocities which attempts to bring grid lines from both side of the shock to the shock position directly. At this point, the algorithm almost stops due to the time step limitation given by the CFL criteria. The solution obtained is represented on fig. 6 in the form of a step function of the Mach number. It can be appreciated on this figure that even if the algorithm stops because of a degenerated triangle, the overall solution is improved compared to the initial solution. However, further improvements are straigtforward if one now allows for some cure action of the grid.

In a second computation, grid cure was allowed while adaptation was still unused. The actions on the grid are thus limited to node removal and the result is that the number of grid nodes will be reduced as the grid is cured.

The results obtained with this procedure are shown on Fig. 7. One can see the ability of the method to align grid lines with the shock. However, as the grid becomes coarser, the ability of the wave model to correctly indicate the wave angle becomes problematic. It is thus suggested to complement the coarse cure method by a local remeshing including refinement.

The third result to be presented thus allows some kind of refinement of the grid. However, to simplify the analysis, the control of the remeshing is based purely on geometrical data, i.e. the flow has no influence on this refinement. The refinement criteria was the following: a triangle is refined if its area is 1.5 times the reference area value, which is the value of the triangle over of the initial grid at the same spatial location. The refinement is performed by disecting the triangle along its longest side. This insures that the grid size distribution will remains almost identical to the initial grid. The

resulting grid and Mach number distribution are presented on Fig. 8. The shock is clearly identified on the grid itself and the Mach number distribution is sharply discontinuous.

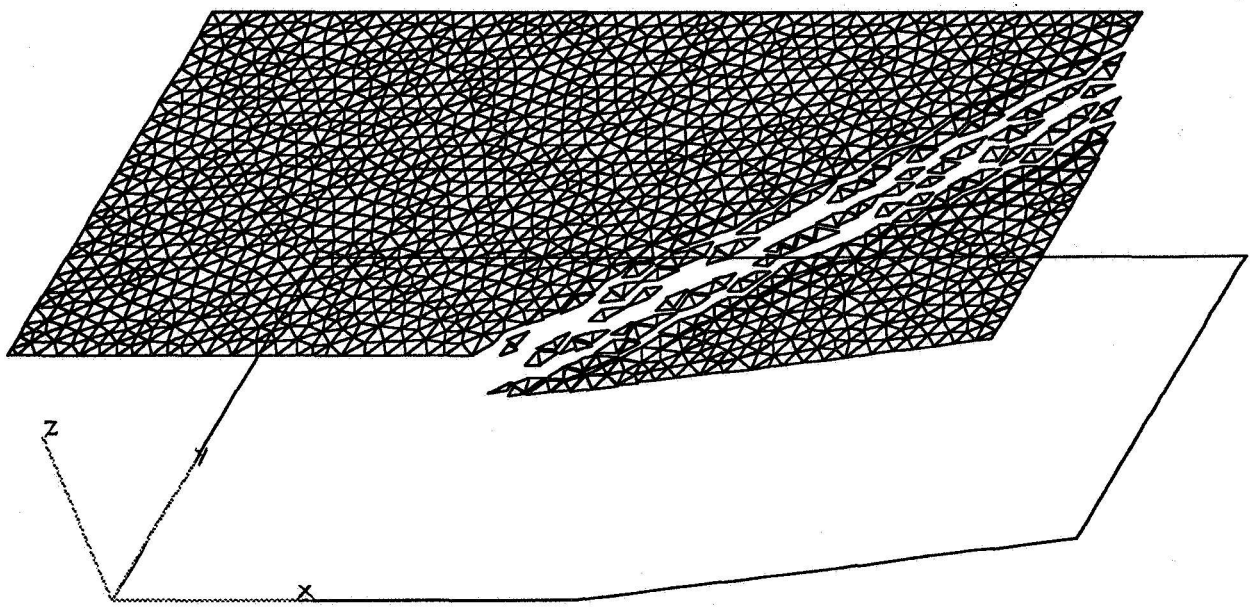


Figure 4: Initial grid and Mach graph.

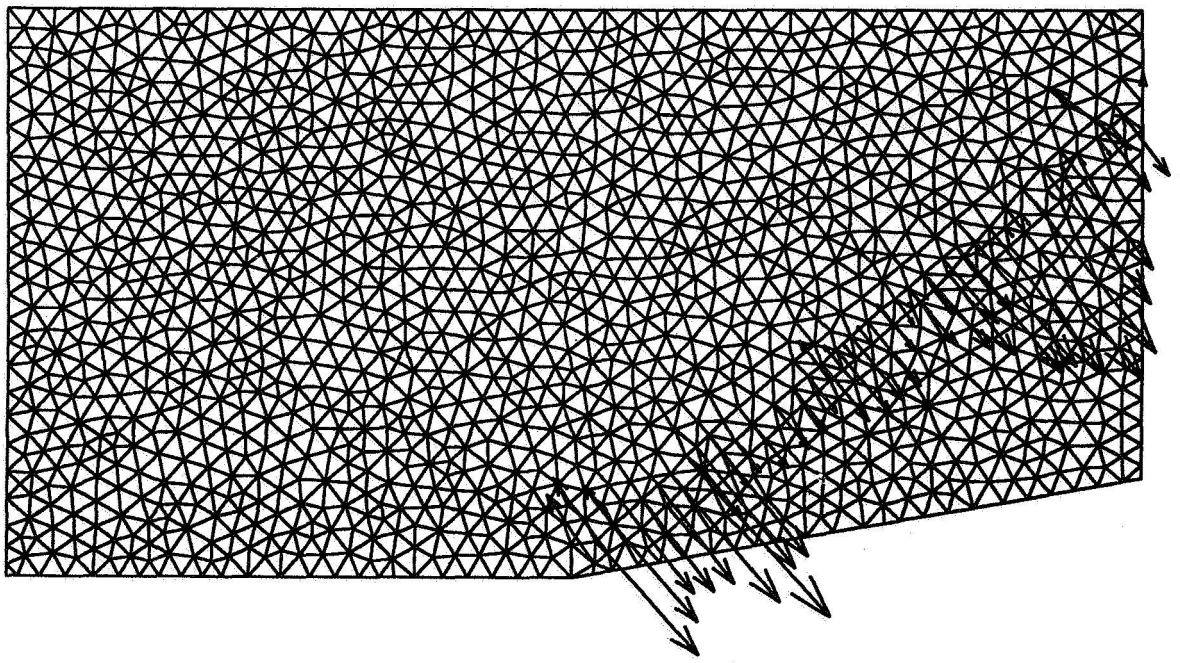


Figure 5: Grid and grid velocities after a few time steps.

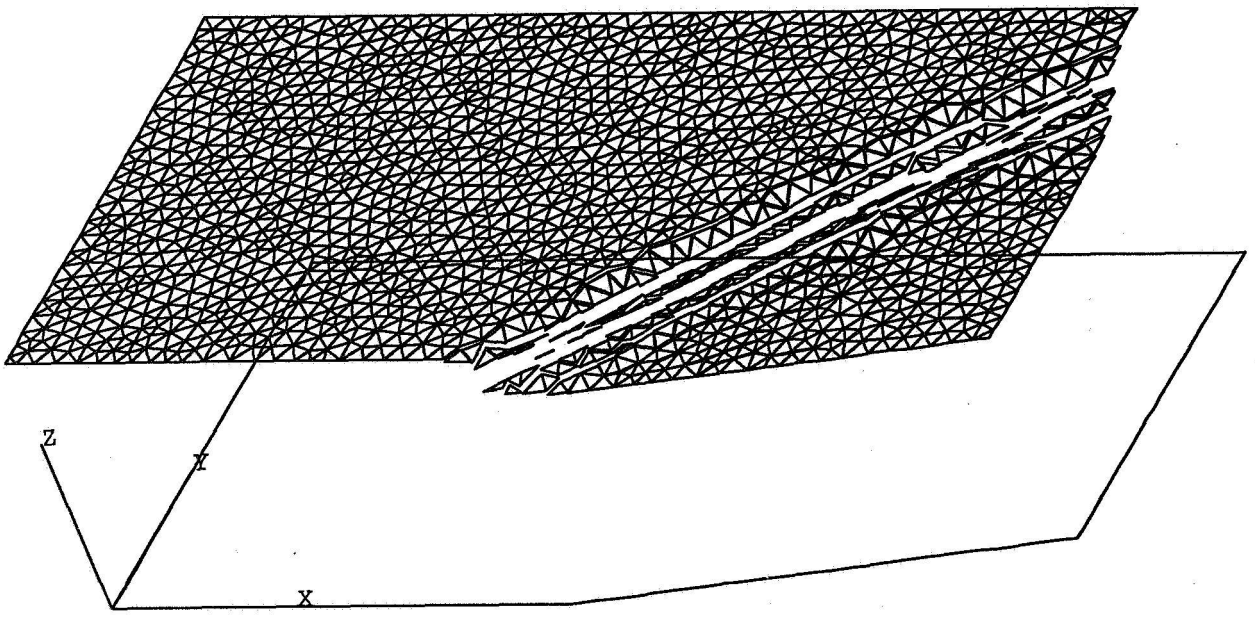


Figure 6: Grid and Mach graph without cure and adaptation.

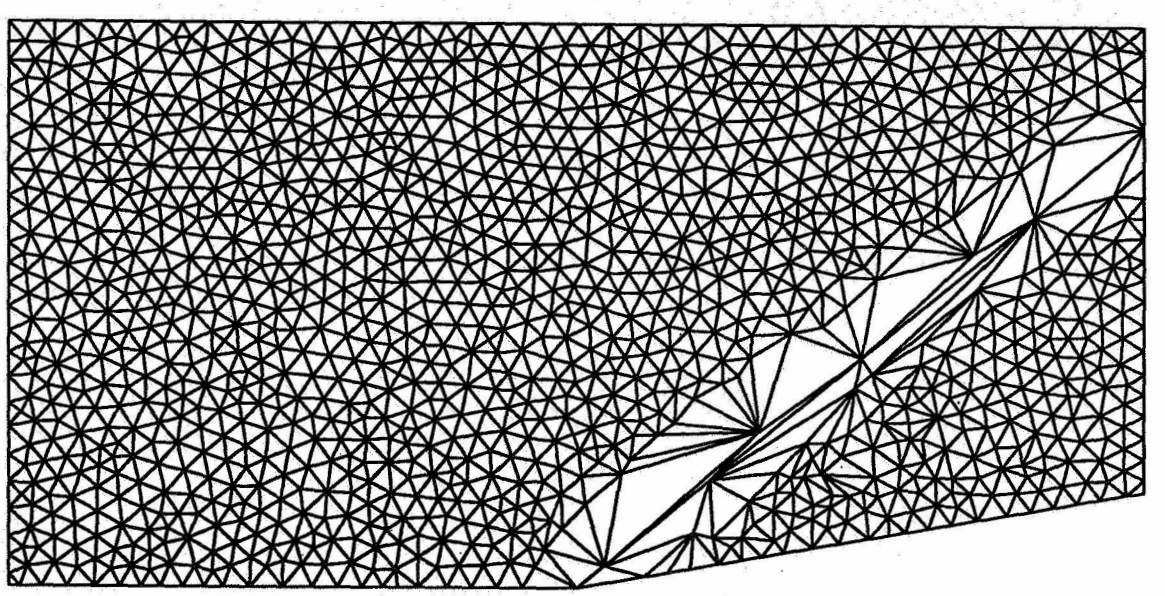


Figure 7: Grid obtained using the cure action.

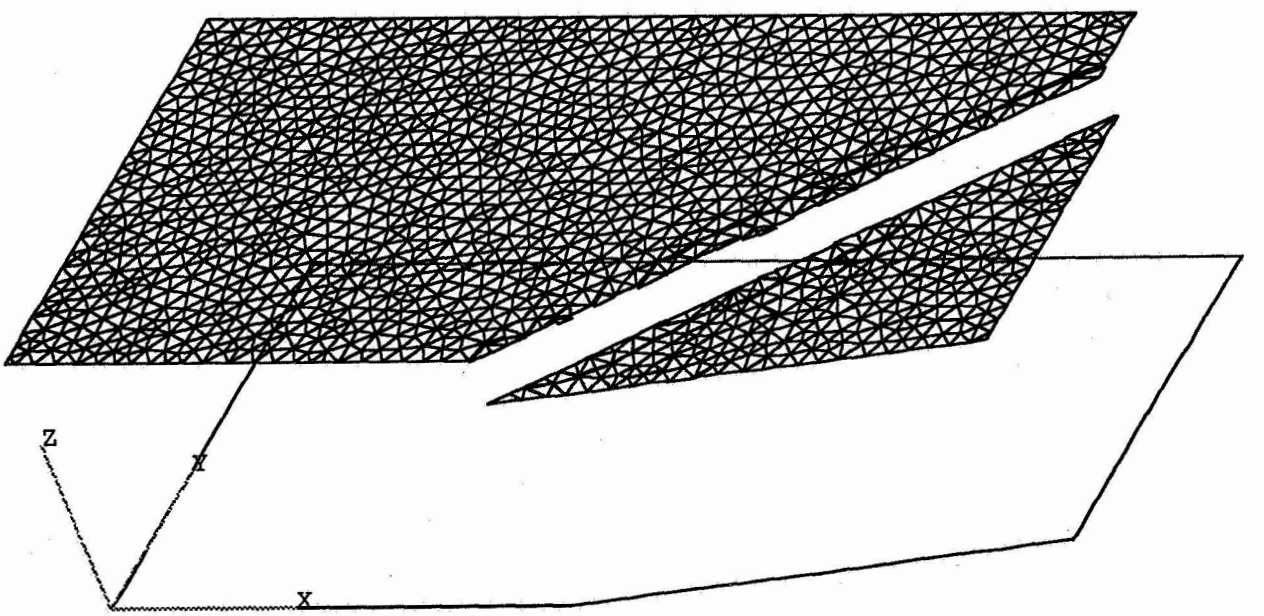


Figure 8: Grid and Mach graph with cure and adaptation.

CONCLUSION

This paper has presented a set of methods for the computation of complex 2D compressible flows in domains with moving boundaries. It has been shown that the complete methodology provides a comprehensive tool for the solution various problems of engineering.

However, more work still need to be done on specific aspects to improve the accuracy and reliability of the method. More specifically, our future work will be concentrated on :

- develop a conservative interpolation algorithm for coarsening operations
- develop a more rigourous error estimator to drive the adaptation process
- quantify the performance of our adaptive methods

ACKNOWLEDGMENTS

The authors thanks the National Science and Engineering Research Council (NSERC) of Canada for their financial support of this work through operational grants.

REFERENCES

[1] X.D. Zhang, J.Y. Trépanier, and R. Camarero. Modelling and computation of arc-flow interaction in circuit-breakers. *International Journal of Computational Fluid Dynamics*, 2:41–64, 1994.

[2] S.R. Kennon and D.A. Anderson. Unstructured grid adaptation for non-convex domains. In S. Sengupta et al., editors, *The second Int. Conf. on Num. Grid generation in CFD.* Pineridge Press Limited, 1988.

[3] J.Y. Trépanier, M. Reggio, M. Paraschivoiu, and R. Camarero. Unsteady Euler solutions for arbitrarily moving bodies and boundaries. In *The AIAA 30th Aerospace Science meeting*, 1992. AIAA Paper 92-0051, January 6-9, Reno, USA.

[4] C.L. Lawson. Software for c1 interpolation. In *Mathematical software III*, pages 166–194. Academic Press, 1977.

[5] D.J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *Journal of Computational Physics*, 90:271–291, 1990.

[6] M.G. Vallet. Generation de maillages anisotropes adaptés, application à la capture de couches limites. Technical report, INRIA, Rocquencourt, 1990.

[7] E.F. D'azevedo and R.B. Simpson. On optimal interpolation triangle incidences. *Siam J. Sci. Stat. Comput.*, 10(6):1063–1075, 1989.

[8] T.J. Barth. On unstructured grids and solvers. In *Computational fluid dynamics.* VKI Lectures Series 1990-03, 1990.

[9] J.Y. Trépanier and H. Yang. Algorithms for adaptive discretization based on triangular grids. Technical report, Ecole Polytechnique de Montréal, 1992.

[10] J.Y. Trépanier, M. Reggio, and R. Camarero. Automated geometric-based mesh requirement for adaptive flow computations. In *The AIAA 31th Aerospace Science Meeting*, 1993. AIAA Paper 93-0674, January 11-14, Reno, USA.

[11] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. In *AIAA Paper 89-0366*, 1989.

[12] H. Zhang, M. Reggio, J.Y. Trépanier, and R. Camarero. Discrete form of the GCL for moving meshes and its implementation in CFD schemes. *Computers and Fluids*, 22(1):9–23, 1993.

[13] J.Y. Trépanier, H. Zhang, M. Reggio, and R. Camarero. Adaptive and moving meshes for the computation of unsteady compressible flows. In A.S. Arcilla et al., editors, *The Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Barcelona, Spain, June 3-7 1991. North-Holland.

[14] Y. Lauzé, R. Camarero, and H. Yang. Interactive generation of structured/unstructured surface meshes with adaptivity. In A.S. Arcilla et al., editors, *The third Int. Conf. on Num. Grid generation in CFD and related fields*, Barcelona, Spain, June 3-7 1991. North-Holland.

[15] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, October 1981.

[16] J.Y. Trépanier, M. Reggio, H. Zhang, and R. Camarero. A finite volume method for solving the Euler equations on arbitrary Lagrangian-Eulerian grids. *Computers and Fluids*, 20(4):399–409, 1991.

[17] J.Y. Trépanier, M. Reggio, and R. Camarero. Adaptation and optimization of triangular meshes for inviscid and viscous flow computations. In *7th IMACS International Conference on Computer Methods for Partial Differential Equations*, 1992. June 22-24, New-Brunswick, NJ, USA.

[18] H. Zhang and J.-Y. Trépanier. An algorithm for the optimization of directionally stretched triangulations. *International Journal for numerical methods in engineering*, 37:1481–1497, 1994.

[19] P.L. Roe. Discrete models for the numerical analysis of time-dependent multidimensional gas dynamics. *Journal of Computational Physics*, 63:458–476, 1986.

FLOATING SHOCK FITTING VIA LAGRANGIAN ADAPTIVE MESHES*

John Van Rosendale
Institute for Computer Applications in Science and Engineering
M.S. 132 C, NASA Langley Research Center
Hampton, VA 23681

SUMMARY

In recent work we have formulated a new approach to compressible flow simulation, combining the advantages of shock-fitting and shock-capturing. Using a cell-centered Roe scheme discretization on unstructured meshes, we warp the mesh while marching to steady state, so that mesh edges align with shocks and other discontinuities. This new algorithm, the **S**hock-fitting **L**agrangian **A**daptive **M**ethod (SLAM) is, in effect, a reliable shock-capturing algorithm which yields shock-fitted accuracy at convergence.

INTRODUCTION

One of the principal difficulties in computing compressible flows is that such flows are generally only piecewise smooth. The solutions are smooth, except along a sequence of arcs or surfaces at which the solution or its derivatives have jump discontinuities. In the vicinity of these discontinuities, difference approximations are problematic. Moreover, errors at shocks can contaminate the solution everywhere.

There are two basic approaches to computation of compressible flows, shock-capturing and shock-fitting. Shock-capturing, in which one applies a well chosen difference scheme throughout the flow field, is effective and reliable, but is usually only first order accurate near shocks. Such schemes smear shocks over several mesh cells, limiting the accuracy and resolution obtainable.

The alternative is shock-fitting, in which the shocks are treated as internal boundaries in the flow across which one applies the Rankine-Hugoniot jump conditions. Shock-fitting algorithms can achieve an arbitrarily high order of accuracy, though properly locating shocks is difficult, especially for flows containing complex embedded shocks.

In recent work we have formulated a new approach to compressible flow simulation, combining the advantages of shock-fitting and shock-capturing. The fundamental difficulty in shock-fitting has always been that of unambiguously detecting and locating shocks. In simple cases, such as that of a strong bow shock, one has enough apriori knowledge of the shock location that fitting schemes are highly successful. However, in more complex situations, shock-fitting becomes difficult and unreliable. For this reason, given the simplicity and effectiveness of modern shock-capturing

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480.

schemes, the latter have come to dominate computational aerodynamics, despite their limited resolution.

The new approach we are exploring begins with a cell-centered Roe discretization, on unstructured meshes [1]. Roe's scheme is a popular and effective method, which has an interesting property: at steady state, this scheme imposes the exact Rankine-Hugoniot jump conditions on any cell face which is oriented to and lying along a shock or other flow discontinuity. Thus if we warp the mesh while marching to steady state, so that shocks and other discontinuities lie along cell faces, Roe's scheme gives virtually exact answers there. This is the basic idea of the **S**hock-fitting **L**agrangian **A**daptive **M**ethod. SLAM is, in effect, a reliable shock-capturing algorithm, which incidentally yields shock-fitted solutions at convergence, with the attendant improvement in accuracy and resolution.

RELATED WORK

While shock-fitting has existed for decades [2, 3, 4], the idea of warping an unstructured grid in a shock-capturing code to effectively fit shocks is new. The basic idea of using a conservative shock-capturing scheme on unstructured meshes, and warping the mesh to fit shocks during iteration to convergence, was independently developed by several groups, including ourselves.

All three of the groups we are aware of used algorithms based on Roe-scheme discretizations, but beyond that the details of these approaches differ. Parpia and Parikh [5] use the waves occuring in a six-wave multidimensional Riemann solver to align mesh edges with shocks, without actually fitting shocks. The multidimensional Riemann solver, due originally to Roe, is described in references [6, 7]. Aligning the grid allows them to achieve true "one-point" shocks, free of the "splitting error" that occurs when shocks cross the mesh at an oblique angle. The other group, Trepanier et al., also used the six-wave multidimensional Riemann solver to control mesh warping [8, 9]. However, unlike Parpia and Parikh, they also move mesh edges to coincide with shocks, thus obtaining shock-fitting accuracy in the final solution.

Our algorithm is similar to that of Trepanier et al., differing in that we warp the mesh using only density gradients, rather than using the waves occuring in a multidimensional Riemann solver. Thus unlike these other groups, we do not need a separate discretization to control mesh movement. In effect, we are reusing information from the Roe scheme discretization.

ALGORITHM DESIGN

The discretization used here is the cell centered Roe scheme. This is an effective and heavily used scheme, whose occasional failings are now well understood and easily overcome [10]. In our algorithm we march to steady state using the Roe scheme coupled to a "locally implicit" time-stepping scheme [1]. For the first 30 or 40 iterations, we keep the mesh frozen, allowing initial transients to dissipate. After that, we allow the mesh to warp at each time step to fit the developing shocks.

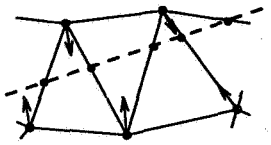


Figure 1: Attraction of Vertices to Shock

Our scheme for warping the mesh consists of two separate components:

1. A shock detector
2. A vertex attraction force

The latter is applied only at points detected as shocks. There is no direct consideration of either attracting or orienting edges, we simply attract vertices to shocks. In effect, we are making use of the following principle:

> *When two vertices of a triangle (or three of a tetrahedron) lie on a straight shock, the intervening cell face exactly fits the shock.*

Our current shock detector uses density gradients at the vertices, computed, for example, by Green's theorem path integrals. Let g_v^n denote the density gradient at vertex v and time step n. For all neighboring vertices, a of v, define a weight

$$w_a^n \;=\; |\,(a-v)\cdot g_v^n\,|$$

where $\cdot$ denotes the normal inner product. Then we take the weighted average of gradient norms, with
respect to these weights:

$$c_v^n \;=\; \frac{\sum_{neighbors} w_a^n\;||g_a^n||}{\sum_{neighbors} w_a^n}$$

We flag points at which the density gradient exceeds this average of gradients at surrounding points. In particular, we threshold the quantity

$$\frac{max(0,\;||g_v^n||\;-\;c_v^n)}{||g_v^n||\;+\;\epsilon},$$

where $\epsilon > 0$ is needed to avoid division by zero in smooth regions. With both numerator and denominator proportional to the density gradient (neglecting epsilon), this detector is equally effective at detecting weak and strong shocks.

Once we have flagged the vertices along the shock, we attract vertices to the shock, by applying forces to each shocked vertex. The force at vertex v in our scheme is of the form:

$$s_v^n\;\frac{g_v^n}{||g_v^n||}$$

That is, a scalar multiple, of the unit vector in the direction of the density gradient. The scale factor s_v^n:

$$s_v^n \;=\; h_v^n\;(\rho_v^n - \bar{\rho}_v^n)\;\delta_v^n$$

with:

h_v^n ..... local mesh size

ρ_v^n ..... angle-weighted vertex average

$\bar{\rho}_v^n$ ..... local ambient density

δ_v^n ..... magnitude of local density range

We take h_v^n to be the minimum length of edges incident on v, while ρ_v^n is the angle-weighted vertex average, as before.

Quantities δ_v^n and $\bar{\rho}_v^n$ both depend on the range of density in a small surrounding region. Define the "local maximum density" $\rho_{1,v}^n$ as the maximum density in cells touching vertices neighboring v. Thus $\rho_{1,v}^n$ is the maximum density in the 20 or so surrounding cells. Similarly, define the "local minimum density" $\rho_{0,v}^n$ as the minimum density in these surrounding cells. Then the local density range is:

$$\delta_v^n = \rho_{1,v}^n - \rho_{0,v}^n$$

Similarly, the local "ambient density" is

$$\bar{\rho}_v^n = (\rho_{1,v}^n + \rho_{0,v}^n)/2$$

Note that this is an average of density extremes, as opposed to a direct average of densities.

The motivation behind all of this is the following. A vertex along a shock is correctly located when the density value there is midway the high and low density in a surrounding region. Thus we want to satisfy the equation

$$\rho_v^n = \bar{\rho}_v^n$$

at vertices along shocks. The scale factor s_v^n approximates the amount of the correction needed to satisfy this equation. Since we adjust the grid at every iteration, the precise scale factor used is unimportant; in effect it is just a relaxation parameter.

MESH CONTROL

In our scheme, vertices are rapidly attracted to shocks. However, without constraints on mesh movement, one rapidly produces undesirably thin cells, or negative cell areas. To avoid this two things are needed:

1. mesh control forces, partially counteracting the forces attracting shocks to vertices.
2. mesh movement step-size control.

We are currently using two forces, one proportional to the change in cell area, another based on angles at vertices. The latter, which applies torques on edges, prevents angles from approaching either 0 or 180 degrees. As angles approach either 0 or 180 degrees, the torques it produces become infinite. Thus if the ODE governing mesh movement is properly integrated, degenerate triangles

cannot occur. By contrast, using "springs" on edges is not effective, since they cannot prevent angles from approaching 0 or 180 degrees.

There are a number of ways to control the step-size in the mesh movement scheme. Our approach is to compute a maximum step at each vertex, designed to prevent degenerate triangles. For every triangular cell R let $h_{min}(T)$ be the minimum of the three altitudes. Then for each vertex define

$$h_{min}(v) \;=\; \min_{\textit{neighboring triangles}} h_{min}(R)$$

If no vertex v moves further than $\frac{1}{2}h_{min}(v)$, degenerate triangles cannot occur.

Note that controlling step-size alone suffices to prevent degenerate triangles, but grid quality may be quite poor. The combination of these mesh control forces and step-size control suffices to maintaining mesh quality, while still allowing effective fitting of shocks.

GENERALIZED VAN ALBADA LIMITER

The first order scheme just described works well, but provides inadequate resolution in smooth regions. Second or third order accuracy can be achieved with a MUSCL-style scheme [11], in which one reconstructs a polynomial in every cell via an appropriate "limiter." One way of doing this is to adapt the stencils, following the ENO approach. However ENO is complex and expensive on unstructured meshes [12].

Our approach is, instead, to use a multidimensional generalization of the Van Albada limiter [13]. This limiter is simple, reliable, and has the attractive property of not clipping extrema. Thus it can, in principle, achieve perfectly sharp approximations of N-waves on very coarse meshes.

The goal in a MUSCL scheme is to replace the constant value in each cell by a linearly varying distribution

$$q^n(\eta) \;=\; q_i^n \;+\; (\eta - \eta_i)(\delta q)_i^n$$

where $(\delta q)_i^n$ is an approximation to the gradient. The Van Albada limiter takes this gradient as a nonlinear average of the gradients computed by forward and backward differencing:

$$(\delta q)_i^n \;=\; \text{ave}\,(q_{i+1}^n - q_i^n,\, q_i^n - q_{i-1}^n),$$

using the averaging function

$$\text{ave}\,(a,b) \;=\; \frac{(b^2+\epsilon^2)\;a \;+\; (a^2+\epsilon^2)\;b}{a^2 \;+\; b^2 \;+\; 2\;\epsilon^2},$$

where epsilon is a small positive constant designed to provide smooth transitions. This kind of averaging was used in [13] for all quantities except density, which was handled slightly differently to avoid negative overshoots in strong astrophysical flows.

The Van Albada limiter generalizes easily to unstructured meshes. To see this, rewrite the above formulas as

$$(\delta q)_i^n \;=\; w_a\,(q_{i+1}^n - q_i^n) \;+\; w_b\,(q_i^n - q_{i-1}^n),$$

with:

$$w_a \;=\; \frac{(b^2 \;+\; \epsilon^2)}{a^2 \;+\; b^2 \;+\; 2\;\epsilon^2},$$

$$w_b \;=\; \frac{(a^2 \;+\; \epsilon^2)}{a^2 \;+\; b^2 \;+\; 2\;\epsilon^2}$$

Now in a similar way, for triangular mesh cells, assume one has gradients g_a^n, g_b^n, g_c^n at the vertices of a triangle, obtained, as usual, by Green's theorem path integrals. Then one can compute the cell centered gradient as

$$g^n = w_a\, g_a^n + w_b\, g_b^n + w_c\, g_c^n$$

for suitable weights w_a, w_b, w_c. Constraining these weights by

$$w_a + w_b + w_c = 1$$

$$w_a,\ w_b,\ w_c \in [0,1].$$

yields second order consistency of the overall scheme, assuming the nodal gradients are first order accurate. We also want to preserve the Van Albada property of not clipping extrema. The particular choice we used was

$$w_a = \frac{(b\,c + \epsilon^2)}{a\,b + b\,c + c\,a + 3\,\epsilon^2},$$

$$w_b = \frac{(c\,a + \epsilon^2)}{a\,b + b\,c + c\,a + 3\,\epsilon^2},$$

$$w_c = \frac{(a\,b + \epsilon^2)}{a\,b + b\,c + c\,a + 3\,\epsilon^2},$$

with $a = ||g_a||^2$, $b = ||g_b||^2$, $c = ||g_c||^2$. Other choices work about as well. In particular, one can chose

$$a = ||g_a||, \quad b = ||g_b||, \quad c = ||g_c||,$$

in closer analogy with the original Van Albada scheme. We prefer the stronger switching that occurs in using the squares.

This generalized Van Albada limiter has the property that near strong jumps the reconstructed gradients use information entirely from one side of the jump, thus achieving second-order consistency while avoiding spurious oscillations. Thus one can think of this as an inexpensive approach to ENO, avoiding the use of complex adaptive stencils and to some extent the convergence difficulties they create.

EXPERIMENTAL RESULTS

Results obtained with SLAM, while preliminary, seem quite promising. We have, in general, no trouble with strong shocks, including attached and detached bow shocks, fish tail shocks, and standing shocks on transonic airfoils. Similar techniques can be used to resolve slip lines and contacts [9], though we have not yet studied this.

Figure 2 shows an un-adapted mesh of 8,000 points around a 10% circular arc airfoil, while Figure 3 shows the same mesh after modification by SLAM to align mesh edges with shocks. Figure 4 shows the density field on this 8,000 point adapted mesh. The shocks are sharp all the way to the far-field boundary, and the limiter is also producing an accurate solution in the smooth region between shocks.

One can judge the solution more accurately by taking cross sections. Figures [6-7] show density contours on cross sections one and four chords from the axis, computed using the the 8,000 point mesh in Figure 3. As can be seen, the sonic-boom profile is well captured, even four chords from the axis. These solutions are mesh-converged to graphical accuracy in the smooth regions. However, there is a slight anticipation of the bow shock, due to small errors in the shock location. This is caused by several numerical effects, which create second-order errors in the shock location.

For this simple problem, one can obtain qualitatively reasonable solutions via SLAM, using as few than 1,000 mesh points. However, accuracy is lacking until the smooth regions are resolved. By contrast, Figures [8-9] show the solution on the un-adapted 8,000 point mesh of Figure 2, one and four chords from the axis. With the Lagrangian mesh adaptivity turned off, the sonic boom profiles are now badly distorted. Figures [10-11] show the same solution on a 32,000 point mesh. The solution is still quite smeared, even though this is a second order accurate scheme, and we made a real effort to locate mesh in regions where the sonic boom was expected.

On the 8,000 point mesh, at four chords from the axis, Figure 9, the bow and tail shocks are separated by about 15 mesh widths. Thus significant smearing is inevitable. This smearing is not as severe on the 32,000 point mesh, which has four times the mesh density throughout the flow field, but the answer there is still much poorer than the SLAM solutions on the 8,000 point mesh. In particular, comparing Figures 11 and 7, notice that the extrema are substantially blurred on the 32,000 point un-adapted mesh. The combination of Lagrangian adaptivity and our generalized Van Albada limiter is particularly effective at getting the extrema right.

Figure 5 shows a more complicated example, flow over an airfoil with a perfectly sharp nose. Since the interior angle at the nose of this airfoil is zero, there is no shock there. Instead, a lambda shock forms in the free stream, some distance away, where the acoustic waves coalesce. Figures 12 show the density cross section just off the body (0.1 chords from the axis). The smooth profile at the nose in Figure 12 rapidly steepens into a shock. By 0.4 chords, shown in Figure 13, the eventual N-wave solution is beginning to form.

The point of this second example is that, unlike most shock-fitting schemes, the Lagrangian adaptive scheme has no effect on the underlying conservative discretization. Thus examples like this, with coalescing waves, intersecting shocks, and so on, present no difficulty, at least in principal.

We are in the process of comparing the SLAM algorithm with standard mesh-enrichment schemes. SLAM achieves shock-fitted accuracy without addition of mesh points, while enrichment strategies substantially increase the number of mesh points, and still produce diffused shocks. Thus SLAM should, in general, require about one tenth as many mesh points as mesh-enrichment schemes in 2D, and should be relatively even better in 3D. The results of such a comparison will be reported in a sequel.

CONCLUSIONS

Lagrangian adaptive grid methods, like SLAM, have great potential for resolving shocks and other flow discontinuities. Unlike mesh-enrichment strategies, which put much finer mesh along shocks, fitting strategies can resolve discontinuities without increasing the number of mesh cells. This advantage is especially important in three dimensions, where the cost of tiling shocks with fine mesh is great. This improved resolution is achieved at little cost, and without loss of robustness, since we retain a Roe scheme-based shock capturing scheme. Thus even if the fitting scheme fails, we still have a robust and effective shock-capturing algorithm.

We have demonstrated the efficacy of SLAM in 2D, and are beginning work on an analogous 3D code. The latter is intended to be applied to the problem of predicting the sonic boom profiles of supersonic aircraft. Current CFD codes cannot adequately resolve the complex shock waves emanating from a supersonic vehicle, since one cannot afford a sufficiently fine grid extending several body-lengths from the aircraft. Fitting schemes, like SLAM, will be able to do much better, and should be able to reproduce the complex shock patterns observed in wind-tunnel experiments.

ACKNOWLEDGMENTS

The author would like to thank K. Anderson for providing the robust unstructured grid flow solver used [2]. We also had help from R. Rausch and D. Mavriplis in grid generation. In particular, our high-quality initial grids were created using Mavriplis' advancing front grid generator [5], which allows clustering of points near the expected shocks, and had no trouble with the sharp-nosed airfoil with zero thickness at the nose. Kwan-Liu Ma helped produce the high-resolution graphics images, and finally, V. Venkatakrishnan provided sound advice on the design of limiters.

REFERENCES

[1] R. ABGRALL, *Design of an essentially non-oscillatory reconstruction procedure on finite-element type meshes*, tech. rep., ICASE, Dec. 1991.

[2] W. K. ANDERSON, *A grid generation and flow solution method for the Euler equations on unstructured grids*, Journal of Computational Physics, 110 (1994), pp. 23–38.

[3] P. M. HARTWICH, *Fresh look at floating shock fitting*, AIAA Journal, 29 (1991), pp. 1084–1091.

[4] J. HASE AND I. PARPIA, *Flux limiters in a rotated upwind scheme for the Euler equations*, AIAA paper 93-0066, (1993).

[5] D. J. MAVRIPLIS, *An advancing front delaunay triangulation algorithm designed for robustness*, tech. rep., ICASE, Oct. 1992.

[6] G. MORRETI, *A technique for integrating two-dimensional Euler equations*, Computers and Fluids, 15 (1987).

[7] J. PARASCHIVOIU, *Une méthod adaptive pour la résolution exact des ondes de chocs et des discontinuités de contact*, Master's thesis, École Polytechnique de Montréal, 1993.

[8] J. PARASCHIVOIU, J.-Y. TREPANIER, M. REGGIO, AND R. CAMARERO, *A conservative dynamic discontinuity tracking algorithm for the Euler equations*, no. AIAA 94-0081, Jan. 1994.

[9] I. PARPIA AND P. PARIKH, *A solution adaptive mesh generatinon method with cell-face orientation control*, Jan. 1994.

[10] J. QUIRK, *A contribution to the great Riemann solver debate*, tech. rep., ICASE, Nov. 1992.

[11] P. ROE, *Discrete models for the numerical analysis of time-dependent multidimensional gas dynamics*, Journal of Computational Physics, 63 (1986).

[12] M. D. SALAS, *Shock-fitting method for complicated two-dimensional supersonic flows*, AIAA Journal, 14 (1976), pp. 583–588.

[13] G. D. VAN ALBADA, B. VAN LEER, AND W. W. ROBERTS, *A comparative study of computational methods in cosmic gas dynamics*, Astronomy and Astrophysics, 108 (1982), pp. 76–84.

[14] B. VAN LEER, *Towards the ultimate conservative difference scheme. V. a second-order sequel to Gudonov's method*, Journal of Computational Physics, 32 (1979).

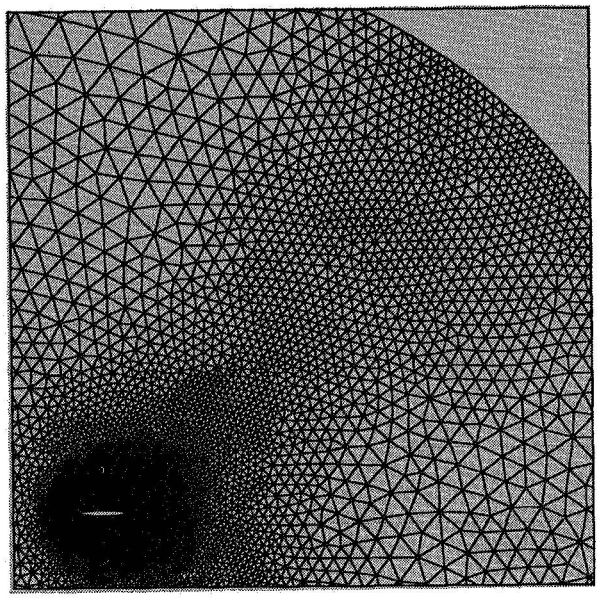


Figure 2. Initial Mesh for Circular Arc Airfoil

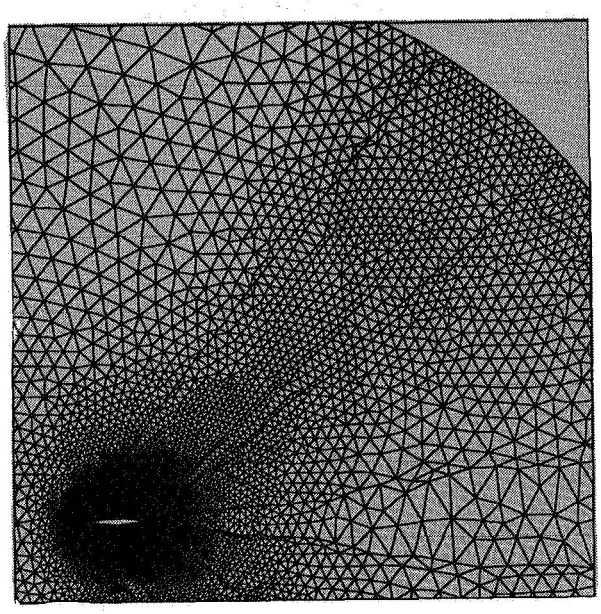


Figure 3. Adapted Mesh for Circular Arc Airfoil

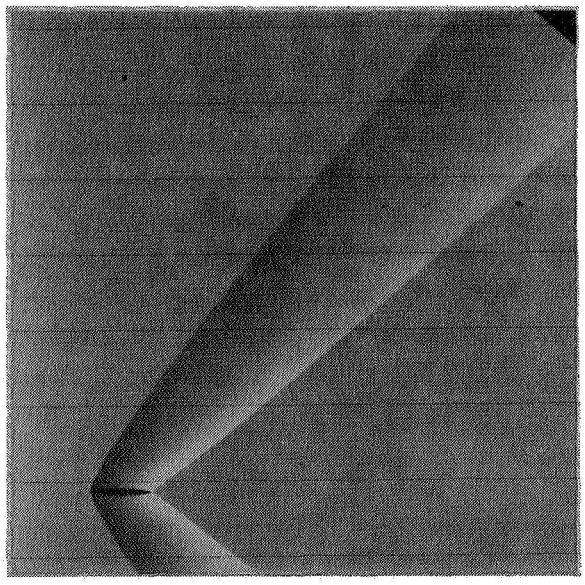


Figure 4. Flow Field, Circular Arc Airfoil, Mach 1.4, 8000-point Adapted Mesh

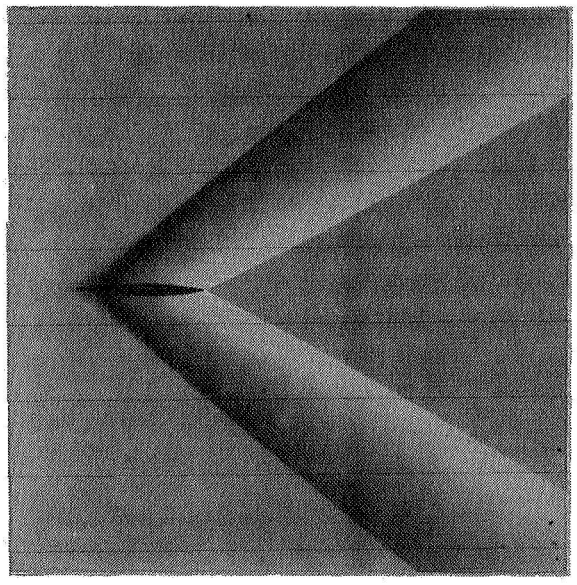


Figure 5. Flow Field, Pointed-nose Airfoil Mach 1.8, 9000-point Adapted Mesh

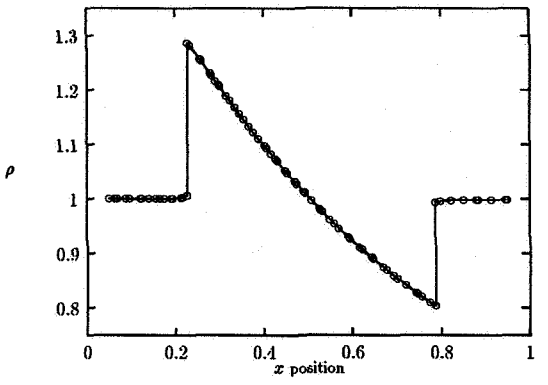


Figure 6. Density One Chord from Axis, Circular Arc Airfoil, Mach 1.4, 8000-point Adapted Mesh

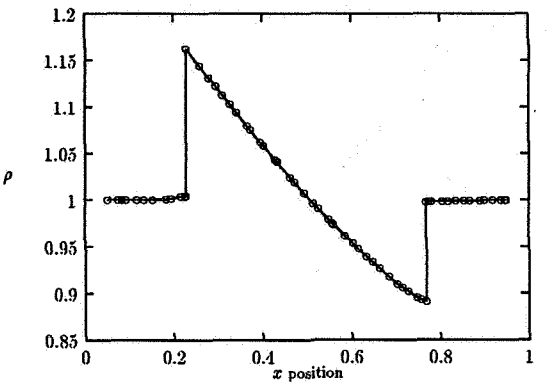


Figure 7. Density Four Chords from Axis, Circular Arc Airfoil, Mach 1.4, 8000-point Adapted Mesh

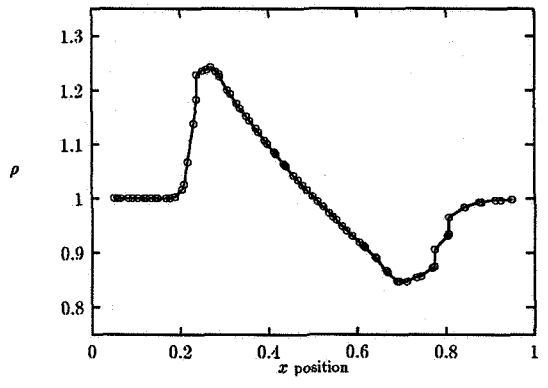


Figure 8. Density One Chord from Axis, Circular Arc Airfoil, Mach 1.4, 8000-point Un-adapted Mesh

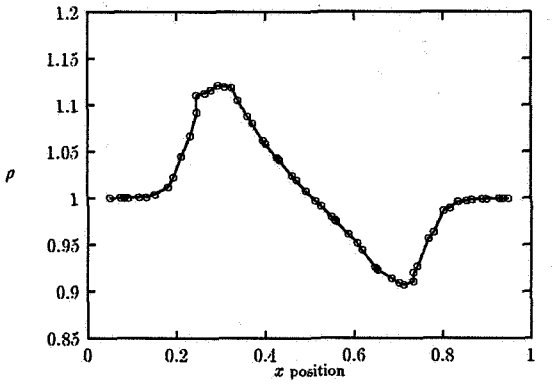


Figure 9. Density Four Chords from Axis, Circular Arc Airfoil, Mach 1.4, 8000-point Un-adapted Mesh

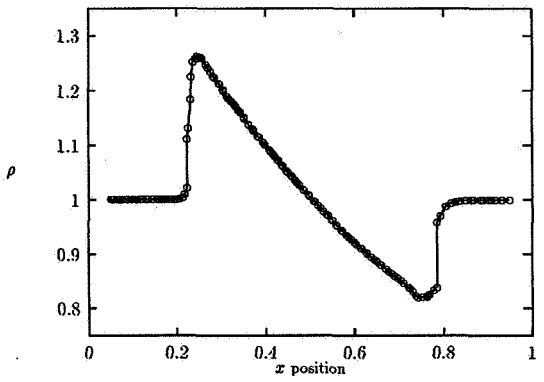


Figure 10. Density One Chord from Axis, Circular Arc Airfoil, Mach 1.4, 32000-point Un-adapted Mesh

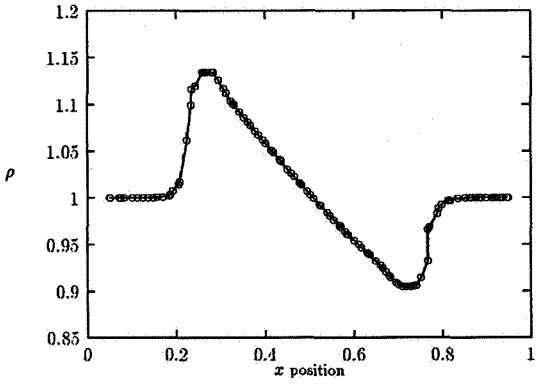


Figure 11. Density Four Chords from Axis, Circular Arc Airfoil, Mach 1.4, 32000-point Un-adapted Mesh

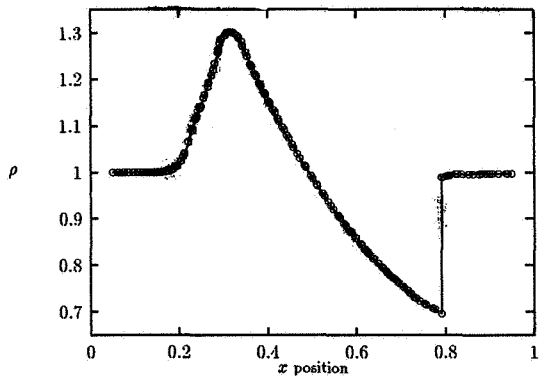


Figure 12. Density 0.1 Chords from Axis, Pointed-nose Airfoil, Mach 1.8, 9000-point Adapted Mesh

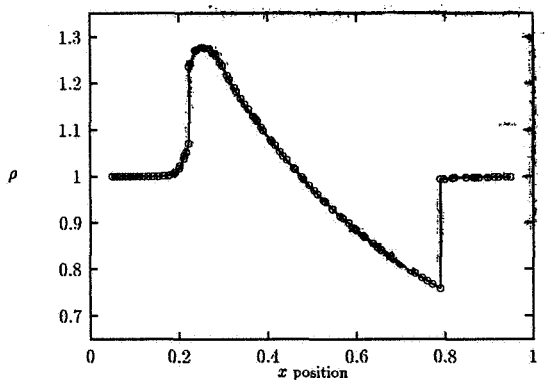


Figure 13. Density 0.4 Chords from Axis, Pointed-nose Airfoil, Mach 1.8, 9000-point Adapted Mesh

ADAPTIVE UNSTRUCTURED TRIANGULAR MESH GENERATION AND FLOW SOLVERS FOR THE NAVIER-STOKES EQUATIONS AT HIGH REYNOLDS NUMBER

Gregory A. Ashford* and Kenneth G. Powell†

W. M. Keck Foundation Laboratory for Computational Fluid Dynamics
Department of Aerospace Engineering
The University of Michigan
Ann Arbor, MI 48109-2118, U.S.A.

Abstract

A method for generating high quality unstructured triangular grids for high Reynolds number Navier-Stokes calculations about complex geometries is described. Careful attention is paid in the mesh generation process to resolving efficiently the disparate length scales which arise in these flows. First the surface mesh is constructed in a way which ensures that the geometry is faithfully represented. The volume mesh generation then proceeds in two phases thus allowing the viscous and inviscid regions of the flow to be meshed optimally. A solution-adaptive remeshing procedure which allows the mesh to adapt itself to flow features is also described. The procedure for tracking wakes and refinement criteria appropriate for shock detection are described. Although at present it has only been implemented in two dimensions, the grid generation process has been designed with the extension to three dimensions in mind. An implicit, higher-order, upwind method is also presented for computing compressible turbulent flows on these meshes. Two recently developed one-equation turbulence models have been implemented to simulate the effects of the fluid turbulence. Results for flow about a RAE 2822 airfoil and a Douglas three-element airfoil are presented which clearly show the improved resolution obtainable.

1 Introduction

The desire in the engineering community to simulate numerically flows about increasingly complex geometries has fueled interest in the development of unstructured grid methods. These methods provide great flexibility in dealing with the complex geometries encountered in practice and offer a natural framework for solution-adaptive mesh refinement. As attention has turned to solutions of the full Navier-Stokes equations, several methods have been recently developed to address the special requirements imposed on the grid generator. At the high Reynolds numbers encountered in typical aerodynamic applications, the viscous effects are felt predominantly in the very thin boundary layers adjacent to solid surfaces and in the wakes. In these regions, the normal length scale can be many orders of magnitude smaller than the tangential length scale. For

*Doctoral Candidate, Aerospace Engineering and Scientific Computing
†Associate Professor, Aerospace Engineering

efficiency this requires that the mesh be highly stretched in the viscous regions while local isotropy is desired in the mainly inviscid regions of the flow.

The generation of an unstructured triangular mesh for a complex geometry is usually accomplished in two phases. In the first, the boundaries of the domain are discretized to form the surface mesh. During this phase, one desires a method which renders a faithful discretization of the geometry by taking into account effects such as curvature and proximity to nearby bodies [1]. In the next phase of the process, the volume mesh is generated filling the domain with triangles. A number of special techniques have been developed to generate the highly stretched elements which are desired in the thin boundary layers and wakes encountered in high Reynolds number flows. Some of these methods use portions of a structured or semi-structured mesh in the viscous regions which are then matched up with an unstructured mesh in the inviscid regions [2]. Another approach is to generate the mesh using prismatic elements [3]. Others are based on modifications to the Delaunay triangulation [4, 5] or to the advancing front method [6, 7].

A variety of algorithms have appeared for solving the compressible Navier-Stokes equations on unstructured meshes. They range from implicit solvers employing upwind methods [8, 9] to multigrid solvers based on a Galerkin finite element technique [10]. Progress has also recently been made in the development of turbulence models which are well-suited to implementation on unstructured meshes [11, 12].

In this work, a method for generating unstructured meshes suitable for high Reynolds number Navier-Stokes flows is described. First the viscous regions are meshed using a node lifting procedure which is a node-based advancing front method. The inviscid regions are then meshed with the traditional face-based advancing front method. A remeshing strategy is also described in which the solution on the current mesh is analyzed and regions which are found to be under-resolved are flagged for refinement when the new mesh is generated. In particular, shock waves and wakes can be well captured in only a few remeshings. An implicit upwind solver is also described for computing solutions to the Reynolds-averaged Navier-Stokes equations.

2 Mesh Generation

The mesh generation process for high Reynolds number Navier-Stokes flows is driven by the need to capture efficiently the thin boundary layers and wakes which occur in these flows. In order to take advantage of the fact that in these regions normal gradients can be many orders of magnitude larger than tangential gradients, the mesh needs to be highly stretched. In addition to control over the stretching, control over the element shape is also desirable. In particular, elements with very large obtuse angles can lead to accuracy problems and should be avoided [13]. It is here that techniques which introduce stretching by means of a mapping [4, 14] can have difficulty.

Ideally, in the anisotropic viscous regions of the flow the elements should be high aspect ratio nearly right triangles while in the mainly isotropic inviscid regions the elements should be nearly equilateral triangles. In this work, this is accomplished by first meshing the viscous regions with a node lifting algorithm and then meshing the remainder of the domain with an advancing front algorithm. A smooth transition between the two regions is assured by dividing the viscous mesh into two layers: a viscous layer and a transition layer. The user has full control over the thickness, the number of points, and the mesh stretching in the viscous layer. The transition layer then serves as a buffer between the edge of the viscous layer and local isotropy at the edge of the viscous mesh. The mesh generation process is detailed in the following sections.

2.1 Geometry Description

The mesh generation procedure begins with a definition of the domain boundaries. In this work a very flexible approach has been taken whereby the boundary curves are given as the union of parametric splines. The particular parameterization chosen here is that of chord length along the spline. Parts of the domain boundary may also be curves describing the locations of wake centerlines in the flow. For example, these may be obtained by streamline traces from an initial solution on a coarse mesh. Each spline is C^2 on the interior while slope discontinuities are permitted at the endpoints. This approach allows for the easy description of very complex multiply-connected domains.

2.2 Surface Mesh Generation

Once the geometry has been defined, the next step is the construction of the surface mesh. This involves the triangulation of the domain boundaries. In 2-D this means the creation of edges and nodes on the boundary curves while in 3-D this would involve the construction of a surface triangulation. Initially each boundary curve is divided into a user-specified number of edges of nearly equal length by specifying a uniform discretization in parameter space. This serves to control the maximum spacing which will be allowed. Next each curve is refined based on a curvature criterion. If the angle formed by the two edges incident to an interior node exceeds a user-specified tolerance (typically 5°), then these edges and their neighbors are flagged for refinement. Flagging neighbors as well as the offending edges expands the region of refinement slightly and produces smoother discretizations. Upon examination of all the nodes, each of the flagged edges is subdivided into two by placing a new node (on the spline) near the edge midpoint. The new distribution of nodes is then smoothed by applying a few sweeps of a Laplacian filter in parameter space. The process is then repeated until the angle criterion is satisfied at every interior node. The process is guaranteed to converge since the splines are at least C^1.

Next the discretization is refined further based on proximity to nearby bodies. The object here is to avoid situations in which the local tangential spacing along the curve is large compared to the distance to a nearby body. In general, if such a situation is allowed to persist, the mesh generator has no choice but to produce badly shaped elements. In fact, in extreme situations, it may fail entirely. Edges which are longer than five times the distance to a nearby body are detected and refinement proceeds recursively as described above. A provision is also made for the user to specify the maximum tangential spacing which can be tolerated at specific locations on the boundary curves. This is useful for clustering points in regions where increased activity is anticipated but is not otherwise apparent from the geometry alone (e.g., trailing edges).

At this point, since each curve has been discretized independently, the tangential spacings on either side of a node at which two curves join may differ substantially. This is then remedied by refining near the endpoint of the curve with the larger spacing until the tangential spacings are comparable. The surface mesh generation concludes with a final smoothing sweep.

2.3 Volume Mesh Generation

The first step in the volume mesh generation procedure is the construction of the background grid. The purpose of the background grid is to specify the local (isotropic) element size throughout the domain. It is constructed by first performing a Delaunay triangulation [15, 16] of the surface nodes and then converting this to a (local) MinMax triangulation via edge swapping [17]. The spacing value at each node is taken to be the average length of the incident boundary edges. Linear

interpolation then provides the spacing function over the whole domain. In most regions of the domain this works quite well resulting in linear variation in element size from the (usually) finely discretized inner boundaries to the coarsely discretized outer boundary. However, this triangulation sometimes produces connections between two widely separated regions of very fine discretization. This then tends to produce an overly fine mesh in regions where it is not desired. This situation is easily remedied by inserting a small number of additional control points into the Delaunay triangulation prior to the edge swapping to break up these unwanted connections. At present this is performed by the user upon examination of the background grid, but a procedure to automate this is under development. This would allow the background grid to be generated automatically from the surface mesh. We believe this is preferable to the traditional approach of first requiring the user to provide the background grid from which the surface mesh is then generated. This is especially true in 3-D where the specification of a background grid which will yield the desired surface resolution can be difficult.

A preliminary step toward the generation of the viscous mesh is the computation of an average surface normal for each boundary node. The surface normals at the nodes are computed by looping over the edges and scattering the contribution due to the edge to each of its two nodes. In order to handle wake cuts, the edges (and nodes) on the wake cut are first duplicated and added to the list of edges but with opposite orientation. This in effect creates a two-sided surface which can then be treated in the standard way. The surface normals are then smoothed with several passes of a Laplacian filter. This smoothing tends to produce better meshes in regions near surface slope discontinuities. At the very end of the mesh generation procedure a clean-up utility is called which removes the duplicate edges and nodes by fusing them with their parents.

With each surface node is associated a local viscous layer thickness, δ_{vl}. When constructing the initial mesh, this is computed by assuming Blasius boundary layer growth (at the given Reynolds number) along the surface of each body starting from a user-specified stagnation point location. Also along any wake curves a similar scaling analysis is used to prescribe an appropriate wake thickness as a function of position. When incorporated within an adaptive remeshing process, the viscous layer thickness can be determined from the existing solution. In particular, since turbulence models are frequently sensitive to the initial y^+ spacing of the first node off the wall, this information can be easily incorporated. Three additional global parameters are asked of the user: n_{vl}, the number of points in the viscous layer, r_{vl}, the stretching ratio in the viscous layer, and r_{tl}, the stretching ratio in the transition layer. Since the stretching ratio is the ratio of the heights of the cells at successive levels, these form a geometric series whose sum is the local viscous layer thickness

$$\delta_1(1 + r_{vl} + r_{vl}^2 + ... + r_{vl}^{n_{vl}-1}) = \delta_{vl}. \tag{1}$$

From this the initial height, δ_1, can then be determined. The heights of cells at successive levels are then given by

$$\delta_n = \delta_{n-1} \begin{cases} r_{vl}, & n \leq n_{vl} \\ r_{tl}, & n > n_{vl} \end{cases} \tag{2}$$

Typical ranges for values of the parameters are $n_{vl} = 10 - 20$, $r_{vl} = 1.3 - 1.7$, and $r_{tl} = 1.4 - 1.8$.

A node lifting process is then used to generate the viscous mesh. This is an advancing front method where the advancement is node-based rather than face-based. The front is initialized to consist of the boundary nodes and edges. These nodes are designated as being at level 0. Advancement begins by selecting a node on the front and marching it out along the local surface

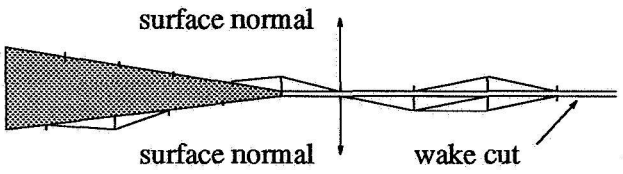


Figure 1: Node lifting example near the trailing edge of an airfoil showing a wake cut.

normal a distance δ_1. In the process, two cells, three edges (two of which form faces on the front), and one node (now at level 1) are created while two faces and one node formally on the front are deleted (see Figure 1). Once all nodes at the current level have been advanced, the process continues with nodes on the next level. Advancement terminates at a node if

- a cell of less than unit aspect ratio would be produced, or
- δ_n exceeds the local background grid spacing, or
- a node forming the base of a new triangle is at a lower level than the node being lifted and the angle between the base and the normal exceeds a threshold (taken to be 120°), or
- an intersection would occur with another edge on the front.

The viscous mesh about the slat for a Reynolds number of 9 million is shown in Figure 2. Note how the transition mesh provides a smooth transition from the highly stretched cells in the boundary layer and wake to local isotropy at the edge of the viscous mesh.

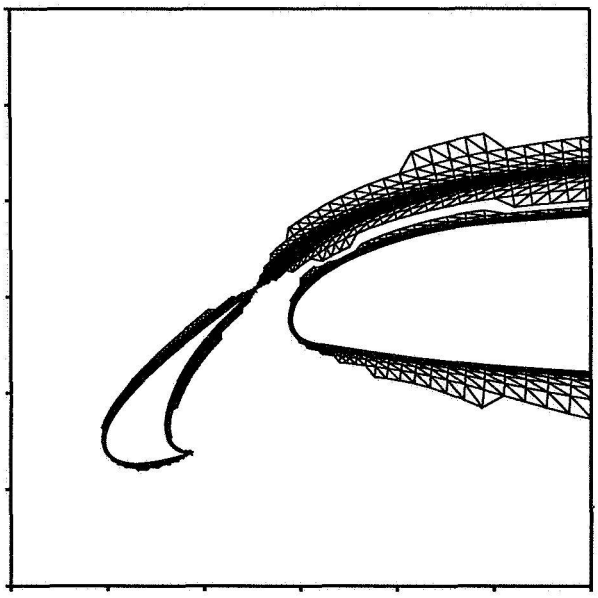


Figure 2: Viscous mesh about a leading-edge slat with a wake.

With the viscous mesh complete, the remainder of the domain is meshed with a traditional face-based advancing front method [14]. This method builds the mesh one element at a time by advancing the boundary of the domain inward. Briefly, the procedure is

- initialize the front
- while(there are faces on the front) do
  - select a face on the front to be the base of the new element
  - obtain the local spacing value from the background grid
  - determine the location of a new 'ideal' point
  - find the nearby nodes and faces on the front
  - decide whether to connect with an existing node on the front or introduce the 'ideal' point
  - form the new element by updating the data structures
- smooth the mesh with a Laplacian filter.

Efficient implementation of this procedure requires the use of dynamic data structures. Typically the front is advanced from its shortest face to help prevent larger cells from overlapping smaller ones. A priority queue of faces which allows efficient insertion and deletion is implemented using a heap. An alternating digital tree [18] is used to locate nearby nodes on the front. Nearby faces on the front are then found using node to face pointers. With these data structures the mesh can be generated in $O(N \log N)$ time while incurring minimal storage overhead. Typical volume mesh generation times are about 35 seconds for a 30,000 node mesh on a 24 MIPS DECstation 5000/200.

The complete mesh consisting of about 31,300 nodes for a Douglas three-element airfoil at a Reynolds number of 9 million is shown in Figure 3. In this example, the locations of the wakes off the leading-edge slat and the main element have been determined by streamline traces from a coarse grid solution. Both wakes have been tracked to slightly downstream of the airfoil at which point the wake grids end.

3 Adaptive Remeshing

One of the advantages of using an unstructured mesh approach is that it provides a natural framework for the incorporation of solution-adaptive mesh refinement. In this process, the mesh is refined locally based on an estimate of the solution error. Since numerical errors tend to be largest in regions where the solution is changing most rapidly, these are generally good candidates for refinement. Conversely, in regions showing little activity, the mesh can often be coarsened with little degradation in solution accuracy. By concentrating mesh points where they are most needed, high quality solutions can be obtained at reasonable computational cost. An unstructured approach facilitates this process because its data structure can easily support local refinement and coarsening of the mesh.

In this work, an adaptive remeshing procedure has been adopted. Guided by the solution on the current mesh, a new mesh is generated which is better suited to capturing the flow features. At the heart of this procedure is the construction of the background grid which will specify the desired spacing throughout the domain. Once this has been established, the new surface mesh can be generated by traversing each spline and discretizing it into line segments whose lengths are given by the background grid. Note that the surface mesh must be derived from the background grid

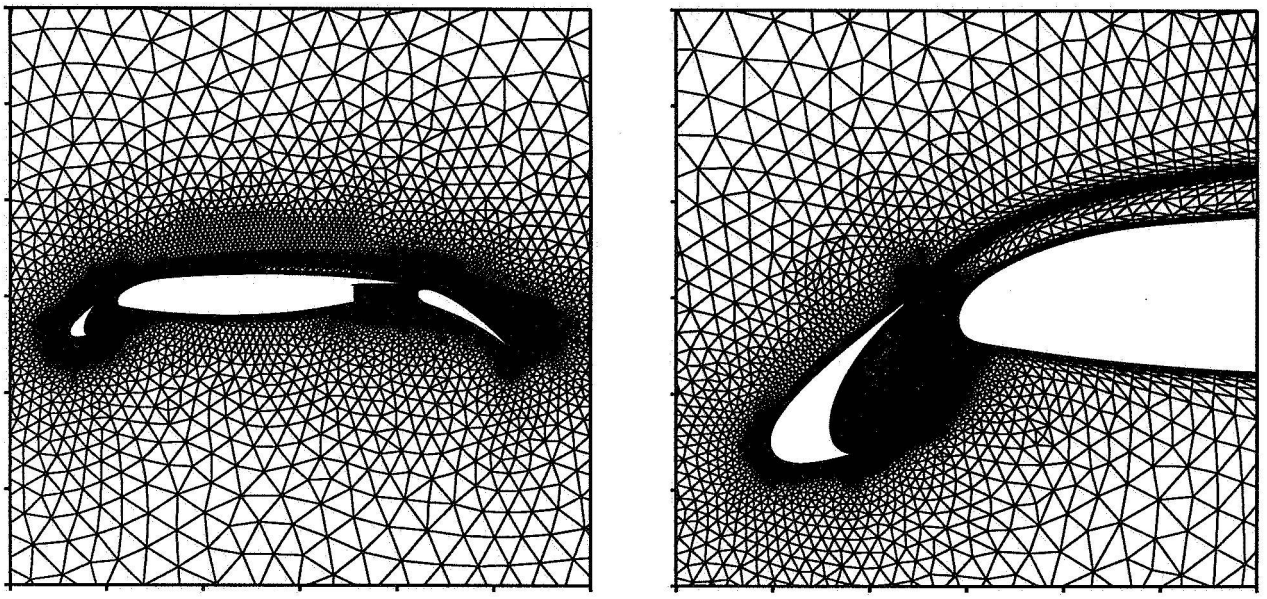


Figure 3: Complete mesh about a Douglas airfoil.

otherwise an inconsistency in the spacing will occur at the domain boundaries. The generation of the volume mesh may then proceed via the node lifting/advancing front algorithm described earlier. The current solution is interpolated onto the new mesh as the initial condition for the flow solver. The solution on the new mesh is then computed and the process repeated if the desired resolution has not been obtained.

By defining appropriate spacing values at the vertices, the current mesh can serve as the background grid. By analyzing the current solution, regions requiring more (or less) resolution are identified. For this purpose, a variety of refinement criteria which detect specific flow features can be used. Next the current mesh spacing is determined by assigning to each mesh vertex a value which reflects the average local element size. This spacing value is then modified in accordance with the selected refinement criteria to produce the background grid. For example, in regions requiring more resolution the spacing value would be reduced to reflect the fact that smaller elements are needed. The adaptive remeshing procedure can be summarized as:

while(solution quality is less than desired) do

- construct the spacing function at the vertices of the current mesh
- compute the refinement parameters
- create the background grid by modifying the spacing function in accordance with the refinement parameters
- generate the new surface mesh using the background grid
- generate the new volume mesh using the background grid
- interpolate the current solution onto the new mesh
- compute the solution on the new mesh.

For inviscid flows a spacing function on the current mesh can be constructed by assigning to each of its vertices the average length of the incident edges. This spacing function can then be smoothed by averaging the value at a vertex with the values at its first-order neighbors. Next the parameters on which the refinement is to based are computed at the mesh vertices. For inviscid flow, two parameters which have been found to work well are related to the local divergence and curl of the velocity field by

$$\tau_{1,i} = |\text{div } \mathbf{u}| \; h_i^p \tag{3}$$

$$\tau_{2,i} = \|\text{curl } \mathbf{u}\| \; h_i^p \tag{4}$$

where h_i is the local spacing value at vertex i and p (taken here to be 1.5) determines how strongly the element size influences the refinement. The divergence criterion measures the local compressibility of the flow and is effective in locating shock waves while the curl criterion measures the local rotationality of the flow and performs well in locating slip layers. The standard deviations from zero of the refinement parameters are then calculated from

$$\sigma_k^2 = \frac{1}{N} \sum_{i=1}^{N} \tau_{k,i}^2 \qquad k = 1, 2 \tag{5}$$

where N is the number of nodes in the mesh.

Refinement can then be effected by decreasing the spacing value wherever $\tau_{k,i}/\sigma_k$ is large. Conversely, the mesh can be coarsened by increasing the spacing value wherever $\tau_{k,i}/\sigma_k$ is small. In general, the desired spacing value can be written

$$h'_{k,i} = h_i \; g_k\left(\frac{\tau_{k,i}}{\sigma_k}\right) \tag{6}$$

where g_k is a non-increasing function which is 1 for intermediate values of $\tau_{k,i}/\sigma_k$ and is bounded away from very large and very small values as $\tau_{k,i}/\sigma_k \to 0$ and ∞ respectively. These bounds are necessary in order to prevent the spacing function, and hence the mesh, from changing too abruptly from one meshing to the next. Typically, g_k can be taken to be a simple piecewise linear function bounded so that $1/4 \leq g_k \leq 2$. Obviously different choices of refinement parameter lead to different values of the desired spacing. It is usually best to take the modified spacing value, h'_i, to be the smallest of these values

$$h'_i = \min_k h'_{k,i}. \tag{7}$$

This ensures that all the selected flow features are detected and resolved by the adaption. Linear interpolation on the triangles is then used to define the modified spacing distribution over the entire domain.

A few modifications are necessary when applying this remeshing procedure to a viscous flow. The spacing value from the current mesh must be constructed so that it is indicative of the local inviscid mesh scale. For example, in the highly stretched cells in the boundary layers and wakes of a high Reynolds number flow it is the local tangential mesh scale which the spacing value should reflect and not the very small normal mesh scale. The reason for this is that near a body or a wake centerline the purpose of the background grid is to specify the local tangential length scale; the local normal length scale is specified explicitly by δ_{vl}. An approach that has been found to work well is to take the spacing to be the average median side length of the triangles incident to the vertex. In the highly stretched cells in the boundary layers and wakes, this results in an appropriate

'streamwise' length scale. In nearly isotropic portions of the mesh, this reverts to a measure of the local average side length. While this has performed quite well, other choices are clearly possible.

Due to the highly anisotropic nature of the flow in the boundary layers and wakes, the refinement criteria must also be modified. In these regions, the refinement criteria should only detect the need for local tangential refinement (e.g., along the surface in the case of shock-boundary layer interaction); the proper normal length scale is accounted for through the specification of the local viscous layer thickness. A parameter which has been found to work well in detecting shocks in these situations is

$$\tau_{3,i} = \frac{p_{max,i} - p_{min,i}}{p_{max,i}} h_i^q \tag{8}$$

where $p_{min,i}$ and $p_{max,i}$ are the minimum and maximum values respectively of the pressure at the vertex and its first-order neighbors and q (taken here to be 0.5) controls the degree to which the local element size influences the refinement.

Other flow features of interest can be captured by similar means. For example, the large vortices which occur in the separated flow behind the slat and in the flap well of the three-element airfoil (and aft of the flap at high angles of attack) are regions where refinement would be beneficial. Since one distinguishing characteristic of these vortices is that they are regions of isotropic rotational flow, one might try using the curl criterion. However, although the curl is relatively large in these areas (on the order of 50), it is much larger in the boundary layer (where it can exceed 100,000). As it stands, refinement based on the curl would tend to flag the entire airfoil surface for refinement while leaving the regions of vortical flow undetected. This can be rectified by replacing h_i in the curl refinement criterion with a measure of the smallest local length scale. This effectively removes the highly stretched cells in the boundary layers and wakes from the computation of the refinement criterion.

4 Solution Algorithm

The Reynolds-averaged Navier-Stokes equations are discretized in space using a finite volume scheme in which the unknowns are associated with the mesh vertices and the median dual mesh is used to define the control volumes. The convective fluxes are evaluated using Roe's flux-difference splitting [19]. Higher-order accuracy is achieved by using a piecewise linear reconstruction within each control volume. A least-squares procedure is used to compute the solution gradients. This procedure is exact whenever the solution varies linearly over the support of the reconstruction. On the highly stretched meshes employed in Navier-Stokes computations, the least-squares procedure is preferable to a Green-Gauss path integration since it appears to be better conditioned. For flows involving discontinuities, it is necessary to limit the reconstructed gradient so that new extrema are not created. Experience has shown that it is sufficient to satisfy this condition at the Gauss points of the flux quadrature (i.e., the edge midpoints). For this purpose the limiter proposed by Barth and Jespersen is used [20]. The viscous terms are evaluated using a Galerkin finite element approximation with piecewise linear elements. On a uniform subdivided quadrilateral mesh this would result in central differencing the viscous terms.

A fully implicit scheme based on a backward Euler linearization of the equations is used to march to the steady state. The backward Euler method is

$$\frac{\Delta \mathbf{u}}{\Delta t} = \mathbf{R}(\mathbf{u}^{n+1}) \tag{9}$$

where $\Delta\mathbf{u} = \mathbf{u}^{n+1} - \mathbf{u}^n$ and $\mathbf{R}(\mathbf{u})$ is an operator representing the spatial discretization. The right hand side is then linearized about $\mathbf{u}^n$ resulting in

$$\left(\frac{I}{\Delta t} - \frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right) \Delta\mathbf{u} = \mathbf{R}(\mathbf{u}^n). \tag{10}$$

This produces a large sparse system of linear equations which needs to be solved at each time step. Since the support of the higher-order scheme is quite large, consisting of the node and its first- and second-order neighbors, typically only the first-order scheme is linearized. This also circumvents the difficulty of linearizing the inherently highly nonlinear limiting procedure. Also due to the complexity of linearizing Roe's flux, generally this is only done approximately. Notice that these approximations do not affect the accuracy at steady state; only the (pseudo) time history is altered. The no-slip and isothermal wall boundary conditions are made implicit by altering appropriate rows in the matrix.

In order to try to reduce memory requirements, a Gauss-Seidel relaxation scheme has been implemented to solve the linear system. This has the advantage of requiring no additional storage beyond that of the matrix itself and is completely vectorizable by using a coloring scheme. In fact, if one is willing to recalculate the matrix each subiteration, the matrix need not be stored at all. However, since the calculation of the matrix is expensive, the required CPU time would increase substantially. Usually 20 subiterations are performed each time step with a CFL number of 300. Typical performance is about 280 MFLOPS on one processor of a Cray Y-MP C90 leading to solution times of about 10-15 minutes for a medium-sized (40,000 node) mesh. The implicit code with the Gauss-Seidel solver currently requires about 300 words of memory per node. This could be reduced further by more frugal memory management.

To simulate the effects of fluid turbulence at high Reynolds numbers, the Baldwin-Barth [11] and Spalart-Allmaras [12] turbulence models have been implemented. These are both one-equation transport models which solve for a working variable related to the eddy viscosity throughout the domain. The turbulence model equation is integrated in time using an implicit method similar to that of the mean flow equations. In order to facilitate the incorporation of different turbulence models, the mean flow equations and turbulence model equation are decoupled in the time integration.

5 Results

The results after one remeshing for a computation at $M_\infty = 0.20$, $\alpha = 16°$, and $Re = 9 \times 10^6$ about the Douglas three-element airfoil are shown in Figure 4. For this calculation the Spalart-Allmaras turbulence model has been used. The wake emanating from the slat is well captured and in fact remains distinct from the boundary layer on the main element over almost all of its length. Although not shown, on the initial mesh (which did not employ wake grids) the boundary layer which develops on the slat merely ends at its trailing edge without any hint of the wake which naturally exists downstream. This is a good example of why a remeshing approach to solution adaption is favored here for these types of flows. If one were to attempt an adaption strategy based on h-refinement for this case, it is likely that many iterations would be required as the wake refinement is gradually propagated downstream from the trailing edge of the slat. Also mesh adaption strategies based on local enrichment have difficulty producing the well-shaped, highly stretched elements desired for the efficient capture of viscous features. A remeshing strategy,

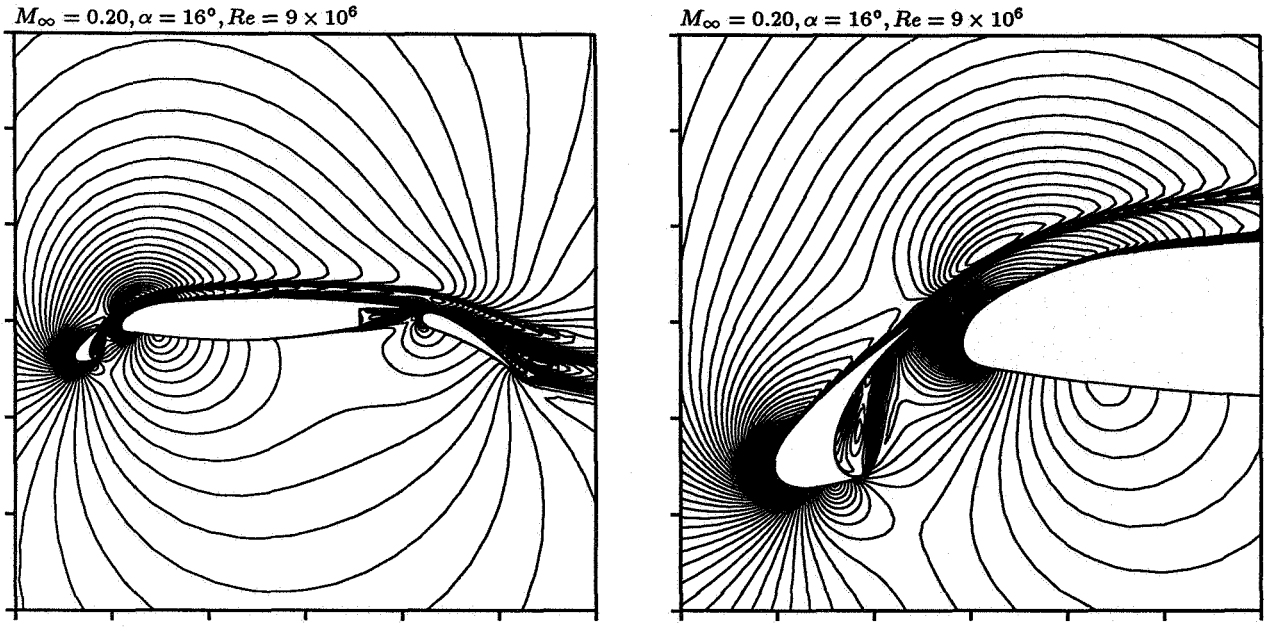


Figure 4: Iso-Mach number contours ($\Delta M = 0.01$) for flow about a Douglas airfoil.

on the other hand, offers the possibility of capturing efficiently many of the flow features in very few iterations.

Figure 5 show the results after two remeshings for flow at $M_\infty = 0.73$, $\alpha = 2.80°$, and $Re = 6.5 \times 10^6$ about the RAE 2822 airfoil. The Baldwin-Barth turbulence model has been used in the computation. The wake centerline has been determined by a streamline trace from the coarse grid solution. The shock-boundary layer interaction region has been well resolved by the two levels of solution adaption based on the local relative pressure change. The calculation predicts slight separation at the base of the shock in agreement with what others have seen for this case with the Baldwin-Barth turbulence model. The surface pressure distribution also shows good agreement with experimental data.

6 Conclusions

A method for generating high quality unstructured triangular grids for high Reynolds number Navier-Stokes calculations about complex geometries has been described. Careful attention has been paid to resolving efficiently the disparate length scales which arise in these problems. By dividing the mesh generation task into two phases, both the viscous and inviscid regions of the flow can be meshed optimally. A solution-adaptive remeshing strategy which allows the mesh to adapt itself to solution features such as wakes and shock waves has also been described. Although at present it has only been implemented in two dimensions, the grid generation process has been designed to be readily extendible to three dimensions. An implicit, higher-order, upwind method has also been presented for computing compressible turbulent flows on these meshes. Two recently developed one-equation turbulence models have been implemented to simulate the effects of the fluid turbulence. High Reynolds number flows about single- and multi-element airfoils have been presented which clearly demonstrate the improved resolution provided by the solution-adaptive remeshing.

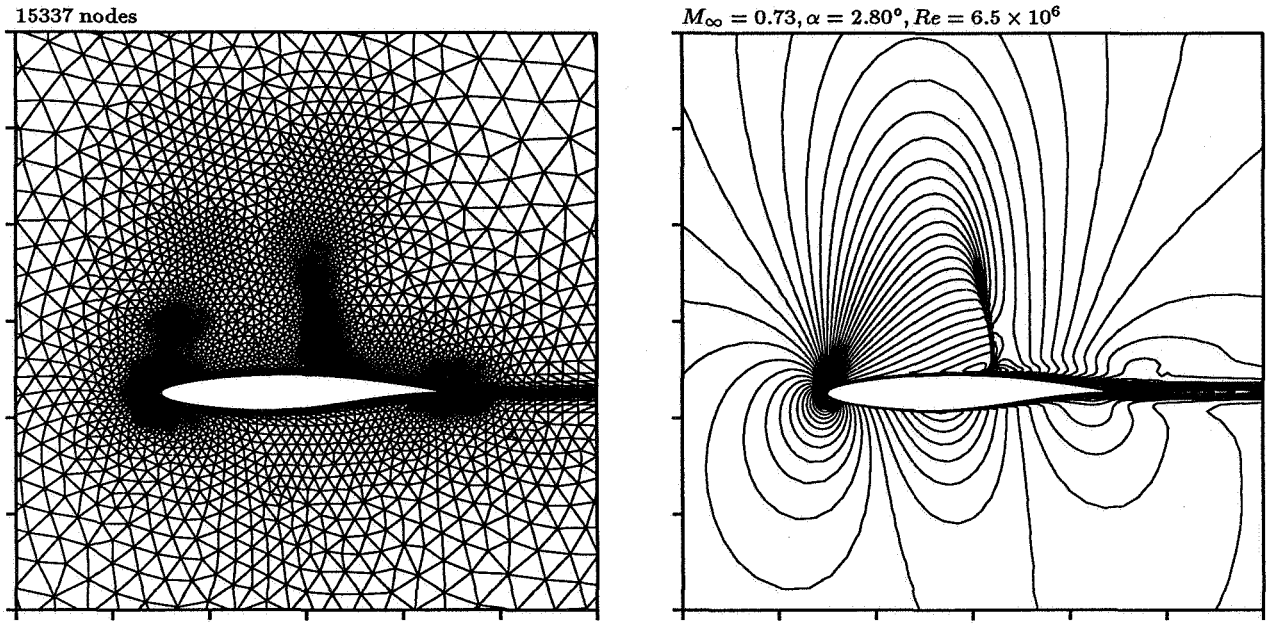


Figure 5: Mesh and Iso-Mach number contours ($\Delta M = 0.02$) about a RAE 2822 after two remeshings.

7 Acknowledgments

This work was funded through an IBM Fellowship in Scientific Computing administered by the Laboratory for Scientific Computation at the University of Michigan.

References

[1] J. Y. Trépanier, M. Reggio, and R. Camarero, "Automated geometric-based mesh requirements for adaptive flow computations," AIAA Paper 93-0674, 1993.

[2] R. Löhner, "Matching semi-structured and unstructured grids for Navier-Stokes calculations," AIAA Paper 93-3348, 1993.

[3] Y. Kallinderis and S. Ward, "Prismatic grid generation for three-dimensional complex geometries," *AIAA Journal*, vol. 31, pp. 1850–6, October 1993.

[4] D. J. Mavriplis, "Adaptive mesh generation for viscous flows using Delaunay triangulation," *Journal of Computational Physics*, vol. 90, pp. 271–291, October 1990.

[5] J.-D. Müller, "Quality estimates and stretched meshes based on Delaunay triangulations," *AIAA Journal*, to appear.

[6] S. Pirzadeh, "Unstructured viscous grid generation by advancing-layers method," AIAA Paper 93-3453, 1993.

[7] S. Pirzadeh, "Viscous unstructured three-dimensional grids by the advancing-layers method," AIAA Paper 94-0417, 1994.

[8] T. J. Barth, "Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes," AIAA Paper 91-0721, 1991.

[9] W. K. Anderson and D. L. Bonhaus, "An implicit upwind algorithm for computing turbulent flows on unstructured grids," *Computers and Fluids*, vol. 23, 1994.

[10] D. J. Mavriplis and L. Martinelli, "Multigrid solution of compressible turbulent flow on unstructured meshes using a two-equation model," AIAA Paper 91-0237, 1991.

[11] B. S. Baldwin and T. J. Barth, "A one-equation turbulence transport model for high Reynolds number wall-bounded flows," NASA TM 102847, 1990.

[12] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," AIAA Paper 92-0439, 1992.

[13] I. Babuška and A. K. Aziz, "On the angle condition in the Finite Element Method," *SIAM Journal on Numerical Analysis*, vol. 13, no. 2, 1976.

[14] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, "Adaptive remeshing for compressible flow computations," *Journal of Computational Physics*, vol. 72, pp. 449–466, 1987.

[15] A. Bowyer, "Computing Dirichlet tessellations," *The Computer Journal*, vol. 24, no. 2, pp. 162–166, 1981.

[16] D. F. Watson, "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes," *The Computer Journal*, vol. 24, no. 2, pp. 167–171, 1981.

[17] T. J. Barth, "Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations," in *Computational Fluid Dynamics*, Von Kármán Institute for Fluid Dynamics, Lecture Series 1994-05, 1994.

[18] J. Bonet and J. Peraire, "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," *International Journal for Numerical Methods in Engineering*, vol. 31, 1991.

[19] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, 1981.

[20] T. J. Barth and D. C. Jespersen, "The design and application of upwind schemes on unstructured meshes," AIAA Paper 89-0366, 1989.

ADAPTIVELY REFINED EULER AND NAVIER-STOKES SOLUTIONS WITH A CARTESIAN-CELL BASED SCHEME

William J. Coirier
NASA Lewis Research Center
Cleveland Ohio

Kenneth G. Powell
The University of Michigan,
Ann Arbor Michigan

SUMMARY

A Cartesian-cell based scheme with adaptive mesh refinement for solving the Euler and Navier-Stokes equations in two dimensions has been developed and tested. Grids about geometrically complicated bodies were generated automatically, by recursive subdivision of a single Cartesian cell encompassing the entire flow domain. Where the resulting cells intersect bodies, N-sided "cut" cells were created using polygon-clipping algorithms. The grid was stored in a binary-tree data structure which provided a natural means of obtaining cell-to-cell connectivity and of carrying out solution-adaptive mesh refinement. The Euler and Navier-Stokes equations were solved on the resulting grids using an upwind, finite-volume formulation. The inviscid fluxes were found in an upwinded manner using a linear reconstruction of the cell primitives, providing the input states to an approximate Riemann solver. The viscous fluxes were formed using a Green-Gauss type of reconstruction upon a co-volume surrounding the cell interface. Data at the vertices of this co-volume were found in a linearly K-exact manner, which ensured linear K-exactness of the gradients. Adaptively-refined solutions for the inviscid flow about a four-element airfoil (test case 3) were compared to theory. Laminar, adaptively-refined solutions were compared to accepted computational, experimental and theoretical results.

INVISCID RESULTS

The solution procedure follows that shown in [4]. The Euler equations are solved upon a Cartesian-cell generated grid using a cell-centered, finite-volume, upwind formulation. The cell primitive variables are reconstructed using a linearly K-exact reconstruction that is slope limited, as in [2] and [3]. For the calculations shown here, Roe's [11] flux difference splitting is used as an approximate Riemann solver at the cell-to-cell interfaces. Solution adaptive mesh refinement is performed by subdividing cells according to the refinement criteria developed in [7]. This procedure computes two parameters, based upon the divergence and curl of the velocity field, which are then weighted by the cell size, l. A simple statistical description of these parameters is then used to determine which cells

to refine and coarsen. That is, letting $\tau_c = |\nabla \bullet u| l^{3/2}$ and $\tau_R = |\nabla \times u| l^{3/2}$ represent parameters that locally describe the compressive and rotational nature of the flow field, cells are refined or coarsened if the variance of these parameters about zero is beyond some specified threshold. For the results shown here, cells are refined if

$$(\tau_C > \sigma_C \quad \text{or} \quad \tau_R > \sigma_R) \quad \text{and} \quad l > l_{min} \tag{1}$$

and cells are coarsened if

$$\tau_C < \frac{\sigma_C}{10} \quad \text{and} \quad \tau_R < \frac{\sigma_R}{10} \tag{2}$$

Experience ([4] and [6]) has dictated the one-tenth scaling of the variance for coarsening, and in practice, the minimum allowable refineable cell size in (1) is typically taken to be 0.001 chords. In all of the adaptively-refined computations shown here, the refinement criteria is set exactly as above. For simplicity, a three-stage, multi-stage scheme is used to advance the equations in pseudo-time, with stage coefficients $\lambda = (0.18, 0.5, 1.0)$. A spatially varying time-step is formed using blended hyperbolic and parabolic stability constraints.

Test Case 3: Suddhoo-Hall Four-Element Airfoil

This test case geometry corresponds to that shown in [13] where successive Karman-Trefftz transformations were applied to a series of circles in the complex plane, resulting in a high-lift-like set of four-element airfoil shapes. The geometry has been approximated using the workshop supplied cubic-splines, and adaptively refined solutions made using the Cartesian, cell-based approach. The free stream Mach number is $M_\infty = 0.2$ and the angle of attack is $\alpha = 0°$. Three levels of adaptive-mesh refinement were made beyond the base grid level. The computed surface pressure coefficients for all the refinement levels are shown along with the geometry in Figure 1. The variation of the computed lift and drag coefficients through the adaptive mesh refinement is shown in Figure 2. Computations using the Cartesian approach on a selection of the inviscid test cases are shown in a companion paper [10].

VISCOUS SOLUTIONS

In [5] and [4] adaptively refined solutions of the Navier-Stokes equations using a Cartesian, cell-based approach are shown for a selection of low and moderate Reynolds number flows. The viscous fluxes are found upon each cell interface using a Green-Gauss type of reconstruction performed about a co-volume located about the interface [4]. The data at the vertices of this co-volume are found in a linearity preserving manner ([4] and [9]), which guarantees the linear K-exactness of the reconstructed gradients. To demonstrate the approach, a selection of the results computed in [4] are shown here.

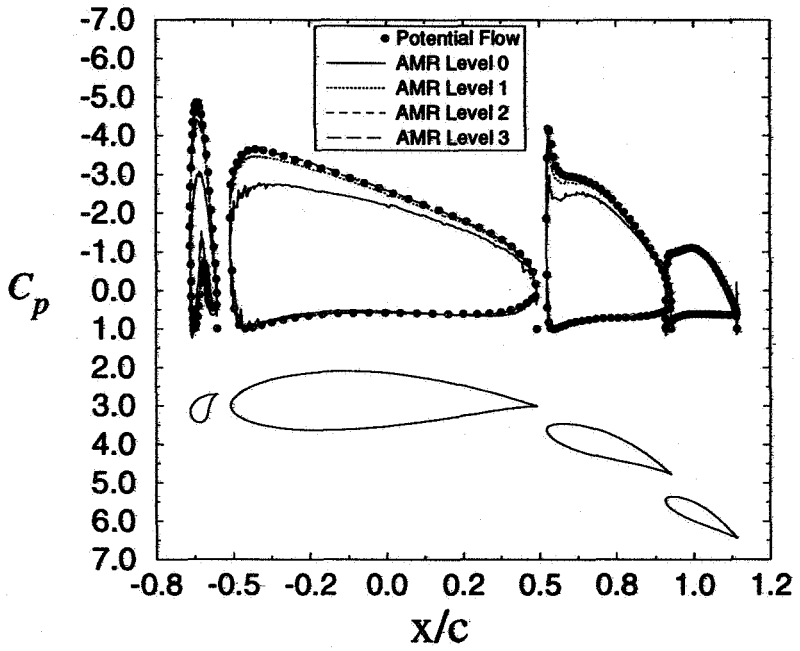


Figure 1 Computed surface pressure coefficient data through adaptive mesh refinement.

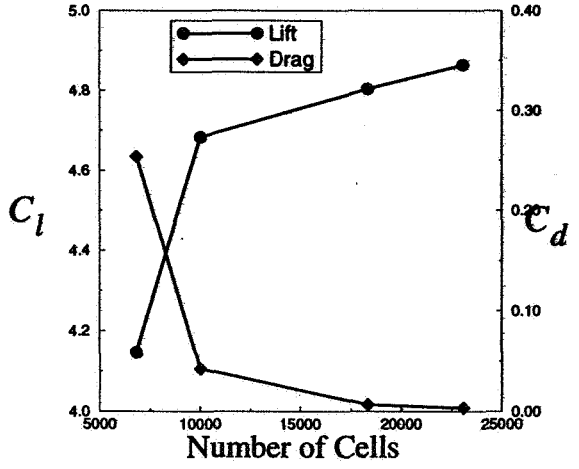


Figure 2 Computed lift and drag coefficients.

Laminar, Driven Cavity Flow: Re=100 and Re=400

The laminar flow inside a square driven cavity is computed and compared to the computed results of Ghia[8]. In [8], an incompressible formulation of the Navier-Stokes equations was solved using an implicit multi-grid method, where tabulated u- and v-velocity data were supplied

along the lines through the geometric center of the cavity. To compare with these incompressible results, the Mach number used here is taken to be $M_{lid} = 0.1$. For the Re=100 case, a uniform base grid of 1024 cells (32 by 32) is generated, and three levels of adaptive mesh refinement beyond the base grid are obtained. Adaptive mesh refinement improves the solution slightly, but the initial solution is quite good. Figure 3 shows the computed u- and v-velocity profiles along vertical and horizontal lines through the geometric center of the cavity for the Re=100 case. For the Re=400 case, the initial solution is poor, but the adaptive-mesh refinement improves the solution quality with each successive level of refinement, until an acceptably good solution is obtained at the final refinement level. Figure 4 shows the computed u- and v-velocity profiles through mesh refinement.

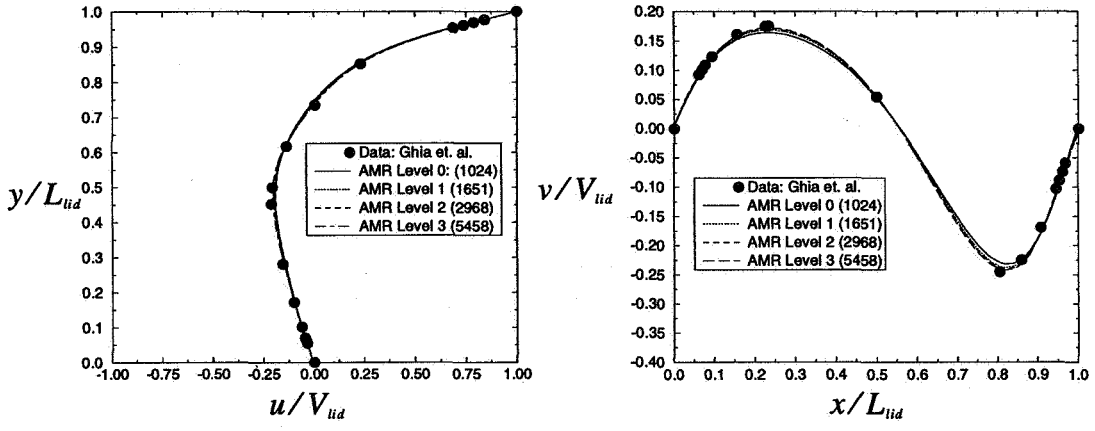


Figure 3 Computed u- and v-velocities through adaptive-mesh refinement for the Re=100 driven cavity problem.

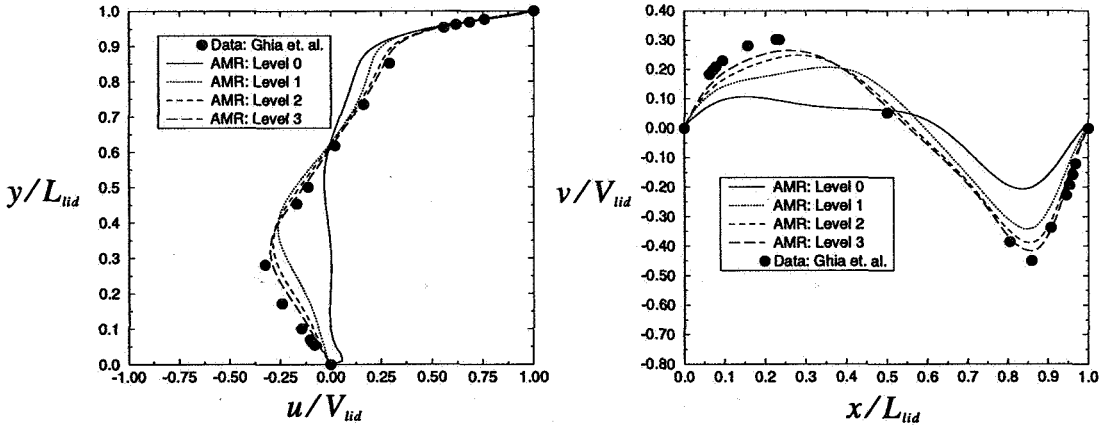


Figure 4 Computed u- and v-velocities through adaptive-mesh refinement for the Re=400 driven cavity problem.

Laminar Flow Over a Backward Facing Step

The laminar flow over a backwards facing step at two Reynolds numbers is used to validate the solver in [4]. The computed results are compared to the experimental data of [1] at the Reynolds numbers of Re=100 and Re=389. A parabolic velocity profile is specified at the inflow, and the exit pressure is specified. This ensures that the proper pressure gradient is imposed on the flow. A coarse base grid is generated, and adaptive mesh refinement is made for three subsequent levels of refinement for both Reynolds numbers. Figure 5 shows the effect of adaptive mesh refinement at a location corresponding to 2.55 step heights downstream of the step. Comparisons are made at other locations of the flow in [4]. The results compare well, and are not shown here. The agreement with the experimental data is good, and the adaptive mesh refinement improves the solution quality with each refinement.

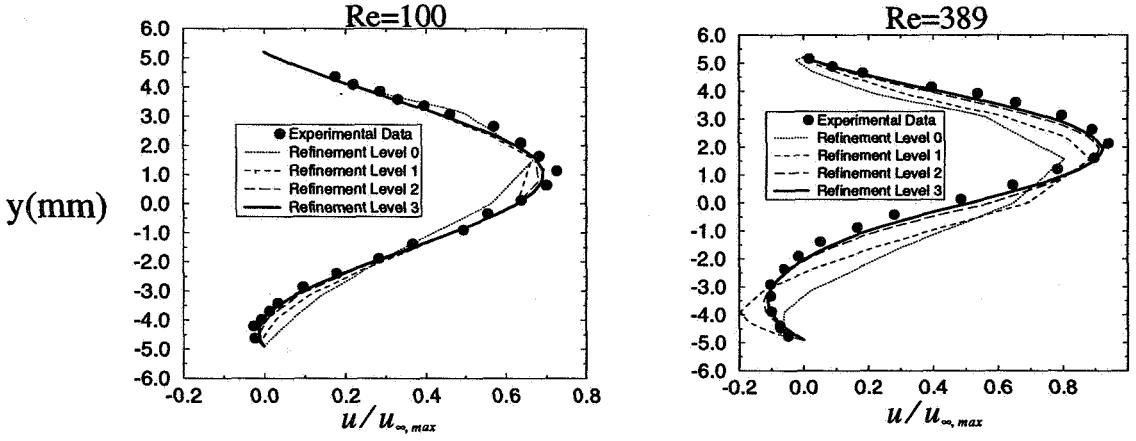


Figure 5 Computed u-velocities at 2.55 step heights beyond step for Reynolds numbers Re=100 and Re=389.

Laminar, Developing Boundary-Layer Flow

The laminar flow over a flat plate which is aligned with the free stream is computed with the Cartesian solver, and compared to theory. Uniform flow is imposed ahead of the plate leading edge, and the boundary-layer develops to a location where the Reynolds number based on distance from the leading edge is $Re_x = 10,000$. The effect of the introduction of cut cells, with their inherent non-smoothness, is illustrated by computing this flow on two series of grids. The first grid sequence is created by orienting the base axes of the Cartesian system coincident with the plate surface, yielding a base grid with no cell cutting, which is then adaptively-refined. The second grid sequence is created by rotating the plate surface 30° with respect to the x-axis, introducing many cut cells along the plate boundary, which also is adaptively refined.

For the axes-aligned cases, when sufficient resolution is supplied, the mean flow profiles com-

pare very well with theory, although the skin friction exhibits small scale oscillations whenever a refinement boundary is located near the wall. Figure 6 shows the computed u- and v-velocity profiles at a location corresponding to $Re_x = 8000$.

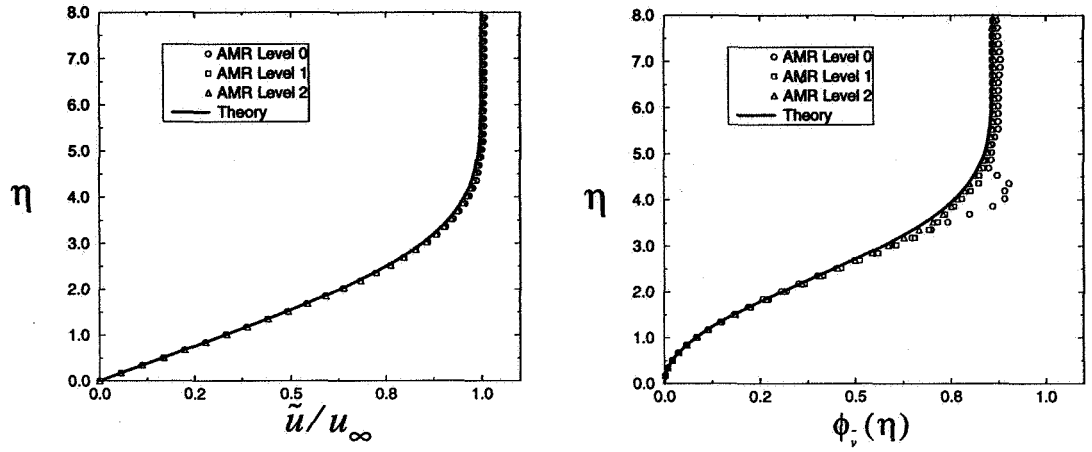


Figure 6 Predicted u- and v-velocity profiles for axes-aligned flat plate.

The grid non-smoothness induced on the non-axes aligned grid caused convergence problems, which was alleviated by using a local modification to the viscous gradient reconstruction procedure in cut cells and their neighbors. The computed u- and v-velocity profiles, shown in Figure 7, compared moderately well to theory, but the skin friction exhibit large scale oscillations.

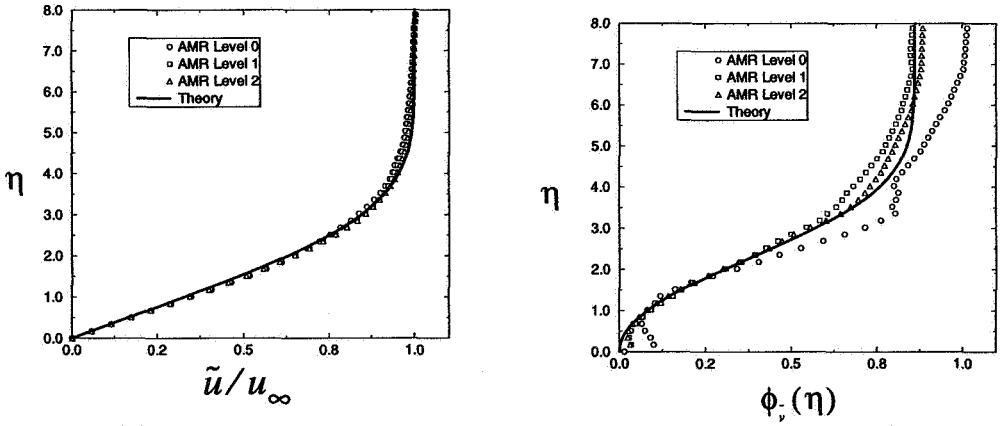


Figure 7 Predicted u- and v-velocity profiles for non-axes-aligned flat plate

The oscillations induced by the extreme grid non-smoothness caused by the cut cells is indicative of the sensitivity of current viscous flux functions to grid smoothness. This sensitivity is highlighted by the grids generated using the Cartesian approach, since extremely non-smooth grids are created near walls, where the shear is typically high. The result of this is typically oscillations in the skin friction and heat transfer rates, and due to the non-positivity of the viscous operators, the convergence can be adversely effected. Regardless of these negative findings, the approach can still prove useful in per-

forming automated grid generation and adaptive mesh refinement upon more geometrically and dynamically complicated flows, as is shown in the next example.

Simulated Branched Duct

To demonstrate the approach for complex geometries, the flow in a stylized duct is computed. This duct geometry corresponds to an experiment conducted at NASA LeRC designed to simulate, in a simplified manner, the flow in the cooling passages of a turbine blade [12]. The calculations shown here in no way try to simulate the experiment: The experimental conditions correspond to a turbulent flow, while the calculations shown here are laminar. A fully developed profile is introduced at the inflow, and the flow is diverted into the primary passage by the blockage introduced by the pin fins in the secondary passage. The Reynolds number based on maximum inflow velocity and pin fin diameter is Re=25. Only one level of adaptive-mesh refinement beyond the base grid level is obtained, due to positivity problems in the rear stagnation region of one of the pin fins. The final adapted grid and contours of total velocity are shown in Figure 8 and Figure 9.

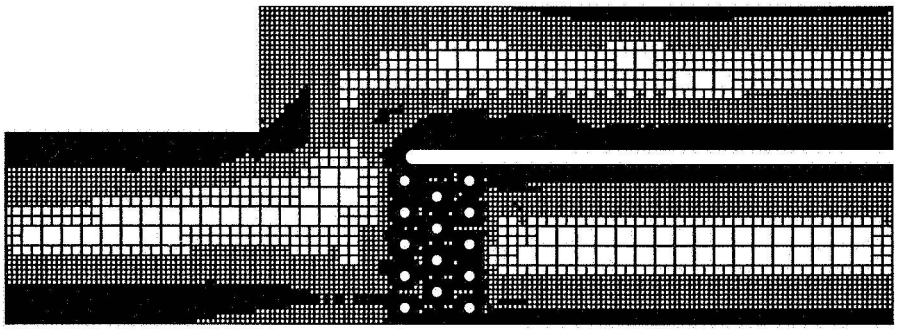


Figure 8 Adapted grid, branched-duct.

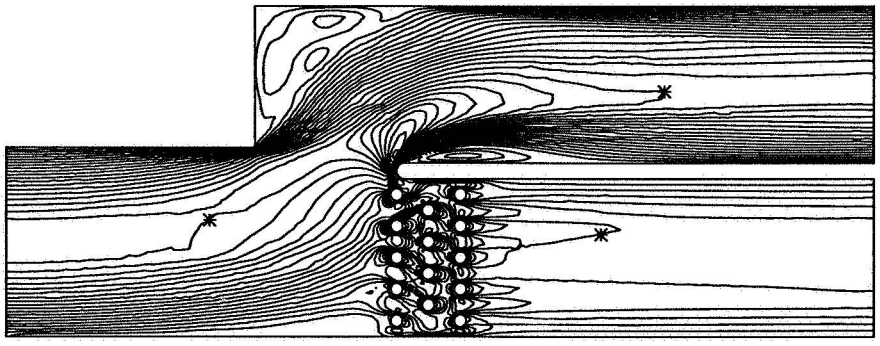


Figure 9 Computed total velocity contours.

The basic flow features predicted here correspond to those in the experiment, although some important features are grossly under-resolved, such as the individual pin-fin wakes. The primary passage separation and reattachment along the splitter plate and the separation anchored at the back step

portion are both properly predicted, as well as the upstream influence of the pin blockage upon the lower wall flow. Although many levels of refinement were not achieved, the larger scale flow features were adequately predicted and there resolution was improved by the mesh refinement procedure.

CONCLUSIONS

Adaptively-refined solutions of the Euler and Navier-Stokes equations using a Cartesian, cell-based approach have been made. Inviscid computations corresponding to test case 3 of the workshop compared favorably with theory. The emphasis here has been upon the extension of the Cartesian, cell-based method to computing viscous flows. Adaptively-refined solutions of the Navier-Stokes equations have been made, and the results compared well to accepted computational, experimental and theoretical data. An inherent weakness of the Cartesian approach is brought forth, that is directly tied to one of the properties that makes the approach useful: The Cartesian approach sacrifices grid smoothness for automation of the mesh generation. This grid non-smoothness is not handled well by the current generation viscous flux functions, which tend to produce non-positive and inaccurate stencils upon distorted grids. Regardless of this comparatively negative finding, the approach has proven to be useful, and can provide accurate, automatically meshed and adaptively-refined solutions of the Euler and Navier-Stokes equations.

REFERENCES

[1] B.F. Armaly, F. Durst, J.C.F. Pereira, and B. Schonung. Experimental and Theoretical Investigation of Backward-Facing Step Flow. *Journal of Fluid Mechanics*, 127:473–496, 1983.

[2] T. J. Barth and P. O. Frederickson. Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction. AIAA Paper 90-0013, 1990.

[3] T.J. Barth and D.C. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA Paper 89-0366, 1989.

[4] W.J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, The University of Michigan, Department Aerospace Engineering. Also published as NASA TM 106754, 1994.

[5] W.J. Coirier and K.G. Powell. A Cartesian, Cell-Based Approach for Adaptively-Refined Solutions of the Euler and Navier-Stokes Equations. AIAA paper AIAA-95-0566, 1995.

[6] D. DeZeeuw and K.G. Powell. Euler Calculations of Axisymmetric Under-Expanded Jets by an Adaptive-Refinement Method. AIAA Paper 92-0321, 1992.

[7] D.L. DeZeeuw. *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*. PhD thesis, The University of Michigan, Department of Aerospace Engineering, 1993.

[8] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[9] D.G. Holmes and S.D. Connell. Solution of the 2D Navier-Stokes Equations on Unstructured

Adaptive Grids. AIAA Paper 89-1932-CP, 1989.

[10] Kenneth G. Powell. The Adaptive Cartesian-Grid Approach, Warts and All. In *Proceedings of the ICASE/LaRC Workshop on Adaptive Grid Methods*, 1995.

[11] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *Journal of Computational Physics*, 43, 1981.

[12] L.M. Russell, D.R. Thurman, P.S. Simonyi, S.A. Hippensteele, and P.E. Poinsatte. Measurements and Computational Analysis of Heat Transfer and Flow in a Simulated Turbine Blade Internal Cooling Passage. AIAA Paper AIAA-93-1797, 1993.

[13] A. Suddhoo and I.M. Hall. Test Cases for the Plane Potential Flow past Multi-Element Aerofoils. *Aeronautical Journal*, pages 403–414, 1985.

PARALLEL ADAPTIVE MESH REFINEMENT WITHIN THE PUMAA3D PROJECT[1]

Lori Freitag
Argonne National Laboratory
Argonne, IL

Mark Jones
The University of Tennessee
Knoxville, TN

Paul Plassmann
Argonne National Laboratory
Argonne, IL

SUMMARY

To enable the solution of large-scale applications on distributed memory architectures, we are designing and implementing parallel algorithms for the fundamental tasks of unstructured mesh computation. In this paper, we discuss efficient algorithms developed for two of these tasks: parallel adaptive mesh refinement and mesh partitioning. The algorithms are discussed in the context of two-dimensional finite element solution on triangular meshes, but are suitable for use with a variety of element types and with $h-$ or $p-$refinement. Results demonstrating the scalability and efficiency of the refinement algorithm and the quality of the mesh partitioning are presented for several test problems on the Intel DELTA.

INTRODUCTION

Unstructured meshes have been used successfully in conjunction with finite element techniques to solve problems in a large number of application areas. Unfortunately, many of these applications are unable to take advantage of the power of parallel computing because of a lack of algorithms and portable software tools for distributed memory architectures. The PUMAA3D (Parallel Unstructured Mesh Algorithms and Applications) project will address this need by providing a publicly available, integrated software package for many important aspects of unstructured mesh computation. In particular, we are designing and implementing provably good, parallel algorithms in the following areas:

- **Mesh generation**: construction of meshes that satisfy user-specified properties over complex, irregular domains;
- **Mesh smoothing**: local adjustment of grid point position to improve the overall quality of the mesh;

[1]This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

- **Mesh refinement**: adaptive refinement and de-refinement of an initial mesh to accurately model rapidly changing solutions;
- **Domain partitioning**: decomposition of the mesh into equally sized, well-separated regions for distribution on multiple processor architectures; and
- **Linear system solution**: the assembly and solution of the linear systems generated by general, unstructured mesh problems.

We have made considerable progress in developing parallel, portable software in each of these areas and have already released the `BlockSolve` [5] software tool for solving large sparse linear systems. In this paper, we concentrate on the algorithms and software developed for the third and fourth components listed above: adaptive mesh refinement and domain partitioning.

Adaptive mesh refinement techniques are known to be successful in reducing the computational and storage requirements for solving a number of partial differential equations [7]. Much research has been done in this area, particularly in the development of sequential algorithms for refining simplicial meshes in two and three dimensions (see, for example, [1], [8], and [9]). Research on the corresponding parallel algorithms has just begun. We describe here an algorithm that uses independent sets to efficiently refine elements in parallel. This algorithm is suitable for use in two and three dimensions, with $h-$ or $p-$refinement, and with a variety of mesh element types.

Because adaptive mesh refinement is a dynamic process, it is often necessary to repartition the mesh after each modification to maintain load balance and good communication characteristics on parallel computers. We have developed a geometric partitioning algorithm that strives to minimize latency and transmission communication costs while evenly distributing the unknowns to the processors for load balance. Because the algorithm is geometric, the partitions are inexpensive to compute, and the algorithm requires only a small fraction of the total solution time. In addition, we have found that this algorithm is particularly effective for the smoothly varying meshes that typically arise in the solution of partial differential equations.

The remainder of the paper is organized as follows. We first describe the parallel refinement and partitioning algorithms. Then we present experimental results obtained on the Intel DELTA that demonstrate the effectiveness and efficiency of our algorithms for several test cases.

PARALLEL ADAPTIVE MESH REFINEMENT

One of the most attractive features of unstructured, simplicial meshes is the ease with which they may be adaptively refined to capture rapidly changing solutions in the numerical modeling of partial differential equations. One popular refinement technique is mesh enrichment, in which grid points are added or deleted from the mesh to increase or decrease accuracy in the numerical solution. Typically, one begins with an initial mesh and selectively adds grid points to regions in that mesh according to local error estimates. In this way, grid points are concentrated in areas where a high resolution is necessary to reduce error and placed more sparsely in other areas of the domain. In addition to appropriate placement of grid points, the mesh must meet several criteria if it is to be used in conjunction with the finite element discretization technique. Let T_0 be an initial

triangulation conforming to some geometric domain and T_k be the triangulation corresponding to the *k-th* refinement iteration. Then for $k = 0, 1, 2, ..., T_k$ must be conforming, T_k must be graded or smooth, and the angles in T_k must be bounded away from 0 and π.

Several techniques for adaptive refinement on simplicial meshes meet the requirements given above to produce valid finite element meshes (see [7] for an overview). The technique that we focus on in this paper is Rivara's two-dimensional bisection algorithm [9]. In this algorithm, a triangle marked for refinement is divided by connecting the midpoint of the longest side to the opposite vertex. This process creates a nonconforming point in a neighboring triangle. The refinement is then propagated until all nonconforming points are removed from the mesh (see Figure 1 for an illustration). Rivara has shown that this propagation will terminate in a finite number of iterations, L_P. In addition, the algorithm guarantees that the angles of T_k are bounded away from 0 and π if the angles in T_0 are. In particular, $\theta^k_{min} \geq \frac{1}{2}\theta^0_{min}$ [10].

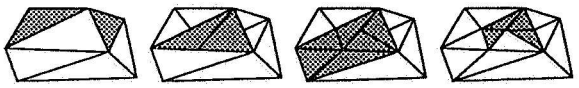


Figure 1: Propagation within the bisection algorithm

To implement Rivara's bisection algorithm on medium grain parallel architectures such as the Intel DELTA, we partition the vertices of the initial mesh and distribute them across the processors. For example, in Figure 2 partition boundaries are indicated by dashed lines. The processor assigned the center partition is responsible for the storage and computation relating to the vertices and triangles indicated by the black dots and shaded regions, respectively. In addition, this processor stores the nearest neighbor information in the finite element mesh, indicated by the clear dots and clear triangles in the figure.

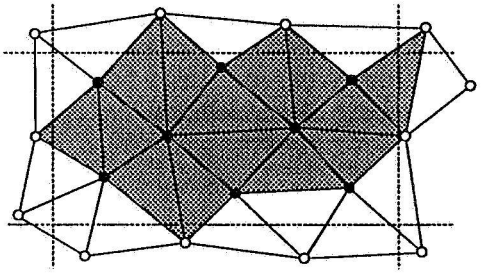


Figure 2: Partitioning of vertices and triangles across processors

One critical aspect of adaptive mesh refinement on distributed memory computers is the management of synchronization points so that global mesh information is correct. There are two ways in which this information can be incorrect: (1) two different processors can create vertices in the same location so that the global vertex list is not unique, and (2) outdated element neighbor information can be used if neighboring processors are not notified of refined elements on processor boundaries. To ensure that data is not corrupted in this manner, we select a near-maximal subset of triangles that can be refined simultaneously on different processors. These subsets are known as independent sets and are defined in the context of the dual graph of the mesh. The dual graph is defined to be $D = (T, F)$, where T is the set of triangles in the mesh and F is the set of edges that connect

two triangles if they share a common edge. We say that a triangle, t_i, is in the independent set, I, if for every neighboring triangle $t_j \in D$: t_j is not marked for refinement, t_j is owned by the same processor as t_i, and $\rho(t_j) < \rho(t_i)$, where $\rho(t)$ is a random number assigned to the triangle at its creation. We note that finding I requires no communication, since each processor stores the triangle neighbor information.

Using independent sets, we now describe an algorithm that avoids the synchronization problems mentioned above and has a provably good runtime (for a complete description of this algorithm see [4]).

$l = 0$
Based on local error estimates, let Q_0 be the
set of triangles initially marked for refinement
While $Q_l \neq \emptyset$ **do**
    Choose $I_l \in D$ from Q_l
    Simultaneously refine the triangles in I_l
    Distribute updated element information
    $l = l + 1$
    Q_l is the set of new nonconforming triangles
    $Q_l = Q_l \cup (Q_{l-1} - I_{l-1})$
Endwhile

The only communication required in this algorithm is the distribution of updated element information to the processors and the global reduction required to check whether Q_l is empty. Notice that the parallel refinement algorithm is not restricted to Rivara's bisection algorithm. Independent sets may be used successfully with a number of different refinement techniques including techniques for p–refinement, nonsimplicial meshes, and higher dimensions.

Jones and Plassmann [4] show theoretically that no two vertices will be created at the same position and that neighbor information in the dual graph is correctly updated. In addition, a P-RAM version of this algorithm is given whose expected runtime is $\mathcal{EO}(\frac{\log Q_{max}}{\log\log Q_{max}}) \times L_P$, where $Q_{max} = \max_l \mid Q_l \mid$ and L_P is the number of levels of propagation. This result implies that the running time is a *very* slowly growing function of the number of vertices, and thus the algorithm can be expected to scale well, as is shown in the Results section.

UNBALANCED RECURSIVE BISECTION

As grid points are dynamically added and deleted in an adaptive mesh, we must recalculate the partitioning of vertices across the processors of a distributed memory architecture. The quality of a partitioning is related to the equity of work assigned to the processors and the cost of communicating data among processors. In particular, for finite element meshes we use the following measures to determine the quality of a partition: the degree of imbalance between the sizes of the partitions, the maximum number of partition neighbors, and the maximum number of edges cut in the finite element mesh. The relative importance of these measures is dependent on the computer architecture and problems considered.

One partitioning algorithm that is effective for the meshes that typically arise in finite element calculations is orthogonal recursive bisection (ORB) [2]. The vertices of the mesh are partitioned according to their physical coordinates in the computational domain. An initial cut is made to divide the grid points in half. Orthogonal cuts are then made recursively in the new subdomains until the grid points are evenly distributed among the processors. This algorithm has the advantages of ease of implementation, inexpensive execution costs, and ease of parallelization. However, it also yields partitionings in which the maximum number of neighbors of any partition is $\mathcal{O}(\sqrt{p})$, where p is the total number of processors [2]. That is, the maximum number of messages that a processor may have to send is dependent on the total number of processors thereby implying a lack of scalability.

To address this problem, we have developed a modification of ORB which we call unbalanced recursive bisection (URB). Let the partition aspect ratio, a_p, be given by $\max(\frac{h}{w}, \frac{w}{h})$, where h is the height of the partition and w is the width. Instead of dividing the unknowns in half, we choose the cut that minimizes a_p and divides the unknowns into $\frac{nk}{p}$ and $\frac{n(p-k)}{p}$ sized groups, where n is the total number of unknowns and $k \in \{1, 2, ..., p-1\}$. Like the ORB algorithm, this algorithm is geometric in nature and hence is easy to implement, inexpensive to execute, and easy to parallelize. Unlike the ORB algorithm, this algorithm does not require that orthogonal cuts be made at each step. That is, we choose the cut that minimizes a_p regardless of the direction of the previous cut. This modification results in improved partitionings for which it can be shown that all of the above criteria can be bounded independently of p for smoothly varying finite element meshes. Hence, the URB algorithm yields scalable partitionings. For a complete description of this work, including proofs of the partition bounds, see [3]. In Figure 3, we show the resulting ORB and URB partitions for a smoothly varying mesh where the densest portion of the mesh is in the lower right i corner. Both algorithms generate partitions with an evenly distributed load. However, the ORB algorithm yields partitions with high aspect ratios which tend to have a large number of partition neighbors. In contrast, the partitions generated by the URB algorithm tend to be square and have fewer partition neighbors.

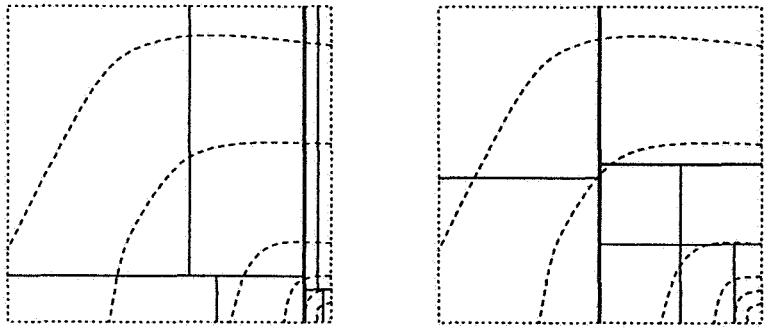


Figure 3: A comparison of ORB partitioning and URB partitioning.

EXPERIMENTAL RESULTS

To demonstrate the effectiveness and efficiency of the parallel refinement and partitioning algorithms, we consider a variety of large scale applications on the Intel DELTA. We use triangular meshes with linear finite elements to solve the following three partial differential equations.

Test Problem 1: (POISSON) Our first test problem arises from Poisson's equation,

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x,y) \text{ in domain } S \tag{1}$$

$$u = 0 \text{ on boundary} \tag{2}$$

on a square domain, where $f(x,y)$ is a Gaussian charge distribution which forces refinement around a point (S_x, S_y). We move the point (S_x, S_y) several times and find a new solution/mesh from the old solution/mesh. This movement requires mesh refinement around the new position and definement around the old position while the rest of the mesh remains nearly constant.

Test Problem 2: (ELASTIC) We solve the elasticity equations for the plane stress problem given here (without inclusion of the load):

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1+\nu}{2}\left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y}\right) \tag{3}$$

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = \frac{1+\nu}{2}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 u}{\partial x \partial y}\right). \tag{4}$$

These equations are solved on an annulus with a load placed on the edges of the domain. In each case above, we selectively refine the mesh until the local error estimate at each triangle is acceptable.

Test Problem 3: (SUPER) Our final test problem arises in the study of the internal structures and configurations of the vortices found in high-temperature superconductors. The model used in these calculations is the nondimensionalized Ginzburg-Landau free energy functional given by

$$F(\mathbf{u}) = F(\psi, \mathbf{A}) = F_{\text{cond}}(\psi) + F_{\text{kin}}(\psi, \mathbf{A}) + F_{\text{fld}}(\mathbf{A}), \tag{5}$$

where $\psi = a + ib$ is the complex-valued order parameter and $\mathbf{A} = [A_x, A_y]$ is the vector potential. These three terms are generally known as the condensation, kinetic, and field energy terms and are given by the formulae

$$F_{\text{cond}} = \int_\Omega -|\psi|^2 + \frac{1}{2}|\psi|^4 d\Omega, \tag{6}$$

$$F_{\text{kin}} = \int_\Omega |(\nabla + i\mathbf{A})\psi|^2 d\Omega, \tag{7}$$

$$F_{\text{fld}} = \int_\Omega \kappa^2 |\nabla \times \mathbf{A}|^2 d\Omega. \tag{8}$$

These equations are solved on a rectangular domain, which we assume is far from the boundaries of the physical sample so that magnetic periodic boundary conditions may be used. The mesh is

refined by proximity to the vortex core singularities.

The results of four typical runs for each of the test problems are shown in the table below. The number following each test case name gives the number of Intel DELTA processors used in the trial. We have constructed the problem sets so that the final solution mesh for each successive problem has roughly twice as many vertices/triangles as in the previous problem. In all three cases, we solve the linear systems arising from the finite element approximations using the parallel conjugate gradient method preconditioned by an incomplete factorization available in the `BlockSolve` software [5] [6].

To show the efficiency of the refinement and partitioning algorithms, the maximum time required to perform these operations is given as a percentage of the total solution time. As a comparison, we also include the percentage of total time required to solve the resulting linear systems. The refinement and partitioning operations required five percent or less of the total execution time in all cases and the solution of the linear systems dominates the cost of the calculation. The time not accounted for in these tables is problem initialization and setup, element evaluation, and linear system assembly.

Problem	Number of Elements	Percent Refine Time	Percent Partition Time	Percent Solution Time	Average Adjacent Partitions	Percent Cross Edges
POISSON16	40292	.795	.428	48.1	4.63	2.38
POISSON32	80116	.856	.516	60.8	5.06	2.60
POISSON64	159758	.647	.731	59.0	5.53	2.70
POISSON128	318796	.568	1.03	60.1	5.64	2.78
ELASTIC16	21043	.697	1.15	98.2	4.00	3.56
ELASTIC32	42049	.560	1.29	98.1	4.38	3.99
ELASTIC64	82997	.449	1.30	98.2	4.75	4.26
ELASTIC128	165468	.268	1.23	98.5	5.20	4.27
SUPER16	30484	.229	.193	69.5	5.31	6.72
SUPER32	48416	.091	.117	86.3	5.40	8.32
SUPER64	111660	.087	.167	88.8	5.64	10.0
SUPER128	196494	.181	.452	86.0	5.71	13.7

To show the quality of the partitions generated by the URB heuristic, we have also included statistics on the partitionings for each of the test cases. The average number of adjacent partitions gives the average number of messages that the processors are sending during each step of the solution process. In all cases, the average ranged from 4 to 6 neighbors; and although the results are not included in the table, the maximum number of partition neighbors ranged from 7 to 9. In the final column we show the maximum final percentage of edges of the finite element mesh cut by a partition boundary to the total number of edges in the partition. Recall that this indicates the message volume each processor is required to transmit in the solution of the partial differential equations. These percentages are quite low for the first two test problems and slightly higher for

the final test problem. This reflects the fact that the meshes for the first two problems are much more smoothly varying than in the final case.

CONCLUSIONS

We have described scalable, efficient parallel algorithms for the adaptive refinement and partitioning of finite element meshes. Work is currently under way to extend these algorithms to three-dimensional meshes and higher order elements. In addition, we are developing parallel algorithms for mesh generation and mesh smoothing. This software will be integrated with the software described in this paper and with `BlockSolve` to form a complete package for parallel solution of finite element problems on simplicial meshes.

REFERENCES

[1] R. E. Bank and A. H. Sherman. An adaptive multilevel method for elliptic boundary value problems. *Computing*, 26:91–105, 1981.

[2] M. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, C-36(5), 1987.

[3] Mark Jones and Paul Plassmann. Bounds on partition quality for orthogonal recursive bisection. Technical report, University of Tennessee, *in preparation*, 1994.

[4] Mark T. Jones and Paul E. Plassmann. Parallel algorithms for the adaptive refinement and partitioning of unstructured meshes. In *Scalable High Performance Computing Conference*, Knoxville, Tennessee, May 1994.

[5] Mark T. Jones and Paul E. Plassmann. BlockSolve v1.0: Scalable library software for the parallel solution of sparse linear systems. ANL Report ANL-92/46, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1992.

[6] Mark T. Jones and Paul E. Plassmann. Scalable iterative solution of sparse linear systems. *Parallel Computing*, 20:753–773, 1994.

[7] W. Mitchell. A comparison of adaptive refinement techniques for elliptic problems. *ACM Transactions of Mathematic Software*, 15(4):326–347, 1989.

[8] R. V. Nambiar, R. S. Valera, K. L. Lawrence, Robert B. Morgan, and David Amil. An algorithm for adaptive rerfinement of triangular element meshes. *International Journal for Numerical Methods in Engineering*, 36:499–509, 1993.

[9] M. Rivara. Mesh refinement processes based on the generalized bisection of simplices. *SIAM Journal on Numerical Analysis*, 21:604–613, 1984.

[10] I. Rosenberg and F. Stenger. A lower bound on the angles of triangles constructed by bisecting the longest side. *Mathematics of Computation*, 29:390–395, 1975.

UNSTRUCTURED VISCOUS FLOW SOLUTION USING ADAPTIVE HYBRID GRIDS

Martin Galle
DLR Institute of Design Aerodynamics
38108 Braunschweig, Germany

SUMMARY

A three dimensional finite volume scheme based on hybrid grids containing both tetrahedral and hexahedral cells is presented. The application to hybrid grids offers the possibility to combine the flexibility of tetrahedral meshes with the accuracy of hexahedral grids. An algorithm to compute a dual mesh for the entire computational domain was developed. The dual mesh technique guarantees conservation in the whole flow field even at interfaces between hexahedral and tetrahedral domains and enables the employment of an accurate upwind flow solver. The hybrid mesh can be adapted to the solution by dividing cells in areas of insufficient resolution. The method is tested on different viscous and inviscid cases for hypersonic, transsonic and subsonic flows.

INTRODUCTION

The development of adaptive methods for efficient and accurate flow calculations has led to two major strategies: One class of schemes is based on the application of triangles in two dimensions and tetrahedral cells in three dimensions. The employment of those grids for adaptive methods yields some advantages: For inviscid calculations the generation of suitable grids even for complex geometries and configurations can be done almost automatically by a powerful grid generator. Furthermore, grid refinement by introducing new points and retriangulating them can be done by some modules of existing grid generators. Such refined grids in general do not require any special treatment either in the metrical setup or in the flow solver part of existing codes. Since the grid generation for complex configurations has become more and more time consuming in comparison to flow calculation, the first point has to be considered to be the main advantage of this class of schemes.

Besides those typical unstructured methods, other adaptive schemes based on quadrilateral cells, or hexahedral cells in three dimensions, exist. Schemes applying to a semi-structured data treatment ([1] and [2]) yield only small advantages concerning flexibility when compared with structured methods. But even hexahedral schemes with an totally unstructured data treatment, that allow an arbitrary cell arrangement, can not compare with tetrahedral schemes in this respect. The main advantage of these methods, contrary to tetrahedral schemes, is the higher efficiency and accuracy especially for regions dominated by viscosity such as boundary layers in high Reynolds number flows. Cells of high aspect ratio can be generated without the need of introducing unwanted small angles. Employing the dual mesh technique, control volume faces are orthogonal to the respective edges.

An approach to combine the advantages of both strategies while circumventing their drawbacks is the employment of hybrid grids. The hybrid grids used in the work presented here consist of hexahedral

cells in regions near surfaces, where viscous dominated flow can be expected, and of tetrahedral cells at some distance from those regions to connect the hexahedral domains and the outer boundaries.

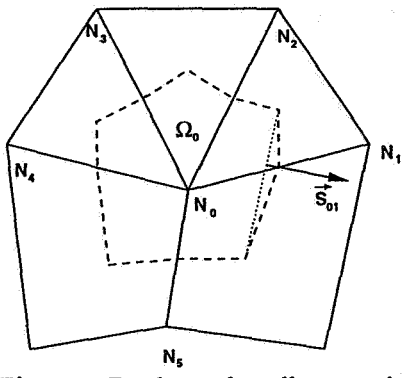


Fig. 1: Dual mesh cell around node N_0

This paper presents an efficient algorithm to solve the Euler- and Navier Stokes equations for hybrid grids consisting of hexahedral and tetrahedral cells. Because of the flexible data structure hexahedral cells do not have to be connected to their neighbors in any predefined arrangement and almost arbitrary combinations of both cell types are allowed. In order to obtain a natural coupling between hexahedral and tetrahedral domains the flow solver part of the presented code applies to a *dual mesh* with control volumes surrounding each node. Fluxes over the control volume boundaries are determined and related to the respective nodes. The dual mesh covers the entire computational domain and connects the two grid types. So, the flow solver does not distinguish between the different cell types, like some comparable methods ([3]). The use of the dual mesh technique yields automatically conservation in the flow field and enables the application of an efficient and accurate upwind flow solver to calculate the inviscid fluxes.

METRICAL SETUP

The input grid data contains information about the geometric positions of the grid nodes and their connections to neighboring nodes. The spatial discretization to determine the viscous and the inviscid fluxes for every node is based on the technique of auxiliary cells used as control volumes. The *metrical setup* encloses the determination of size and shape of the dual mesh cells from the given information.

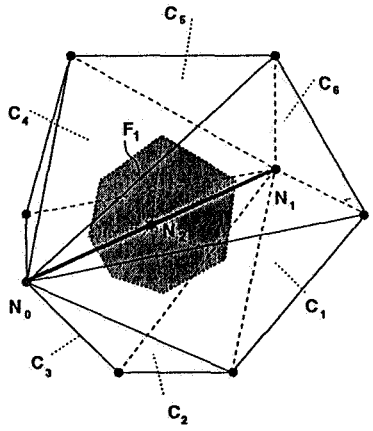


Fig. 2: Face of the dual mesh related to an edge connecting N_0 and N_1

As shown in figure 1, a control volume surrounding node N_0 inside the computational domain has one face for every neighboring node $N_{1..5}$. These faces cross the edges halfway between N_0 and its neighbors and are bounded by the geometric centers of the cells surrounding N_0. The faces of the control volumes are described by face vectors $\vec{S}$, representing the face orientation and the face size. Each edge of the computational grid has one related face in the dual mesh. The face is bounded by the midpoints of the cells surrounding the edge. Because of the need to deal with hanging nodes, control volume faces are not to be bounded by the midpoints of the faces of computational grid cells as e.g. in [4]. As the relations between the cells also have to be determined in order to connect the midpoints of two neighboring cells, the evaluation of the face vectors for three dimensional grids becomes a non trivial task.

Consider an edge connecting the nodes N_0 and N_1 inside the computational domain. This edge is shared by n cells $C_{1..n}$ as shown in figure 2. The Point N_2 is the midpoint of the edge connecting N_0 and N_1. The first task in the setup is the determination of the cells C_1 to C_n and the ascertainment of their order around the edge.

The face F is composed of triangles formed by midpoints of two neighboring cells and N_2. The respective face vector is the sum of all vectors related to the those triangles. The volume Ω_0 of the dual cell is composed of tetrahedra with the corner points N_0, N_2 and the midpoints of two neighboring cells. In a loop running over the cells C_1 to C_n the contributions to the face vector and cell volume are evaluated.

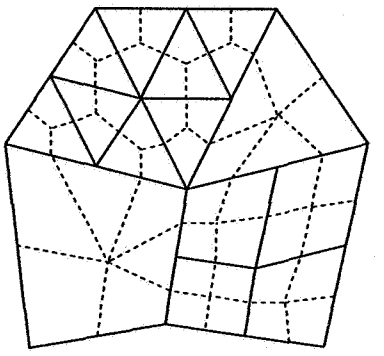


Fig. 3: Dual mesh at the interface between cell types and refinement levels

The metrical setup, including the determination of the volume of the dual mesh cells as well as the components of the face vectors, has to be executed before the flow calculation starts. Since the setup forms one dual mesh from the entire hybrid grid, conservation is guaranteed for the computational domain even at interfaces between regions of hexahedral and tetrahedral cells and at interfaces of refined and unrefined cells, as shown in figure 3.

SPATIAL DISCRETIZATION

Inviscid Flux Calculation

The Euler equations are solved employing a modified AUSM Upwind Scheme ([5]) as it is described in detail by Kroll and Radespiel ([6]). The special merits of AUSM compared to other upwind schemes are the low computational complexity and the low numerical diffusion.

To compute the inviscid flux over the face F is based on the flow conditions on both sides of the face. The values are taken directly from N_0 and N_1 for first order calculations. For second order accurate calculations the independent flow variables are linearly reconstructed on the control volumes around N_0 and N_1 and

$$u_L = u_0 + \nabla u_0 \cdot \frac{1}{2}\vec{V}_{01} \tag{1}$$

The gradient ∇u_0 of a variable u is obtained by employing a Green-Gauss formula:

$$\nabla u_0 = \frac{1}{\Omega_0} \cdot \sum_{i=1}^{n} \frac{1}{2} \cdot (u_0 + u_i) \cdot \vec{S}_{0i} \tag{2}$$

where Ω_0 is the volume of the dual cell around N_0 and $\vec{S}_{0i}$ is the normal vector of the dual mesh face F as shown in figure 1.

Near shocks the values on the edges have to be limited to avoid overshoots. The limiting is done by an minimum/maximum clipping like it is proposed by Barth in [7]. If a reconstructed value at any face of the control volume exceeds the minimum (or maximum) of the values given by node N_0 and the surrounding nodes $N_{1..n}$, the gradient ∇u is scaled by a factor Θ, so the reconstructed value becomes equal to the minimum (or maximum) of the nodes $N_{0..n}$.

Determination of Viscous Flux

In order to calculate the viscous flux over the face of an auxiliary cell the primitive variables and the derivatives of velocities and temperature on the face have to be computed. For Face F in figure 2 these values are evaluated by averaging the values of N_0 and N_1. The gradients of the primitive variables in three cartesian directions are the components of ∇u. For the calculation of the viscous flux the unlimited values have to be used. The determination of the gradients in the way described above provides the gradients in the x-, y- and z-direction. This enables the solution of the full Navier Stokes equations without relying on a Thin-Layer approximation.

GRID ADAPTION

The computational grids can be adapted to the calculated solution by dividing cells in regions of insufficient flow resolution. For tetrahedral cells only an isotropic division into eight children cells is allowed. Hexahedral cells can be divided in one, two or all three directions.

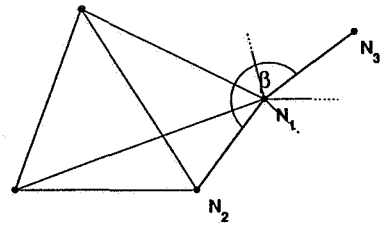


Fig. 4: Determination of helping node N_3

The procedure of tagging cells for division is split into two parts. During the first loop the second differences of pressure Δp with

$$\Delta p = \left| \frac{p_2 - 2p_1 + p_3}{p_2 + 2p_1 + p_3} \right|$$

are computed for each edge. In order to calculate the second differences the most appropriate node N_3 of the neighbors of N_1 has to be determined, as shown in figure 4: N_3 is the node the angle β becomes maximal for. If the value of Δp exceeds a certain threshold, the cell the respective edge belongs to is tagged for division. For hexahedral cells the relative position of the edge in each cell has to be considered in order to divide the cells in the right direction. For the second loop all cells that are already tagged are excluded. During this loop other flow quantities are computed. In the present code the following criteria are implemented:

- first differences of density
- second differences of density
- first differences of velocity

- first differences of velocity weighted by point distance

If the calculated quantity exceeds a second threshold, the respective cells are also tagged for division. The thresholds control the number of divided cells. They can either be fixed before the tagging starts or set iteratively in order to receive a predefined number of cells after the cell division is finished.

For both hexahedral and tetrahedral cells, only one hanging node on every edge is allowed. These hanging nodes are caused by neighboring divided cells as shown in figure 5. The hexahedral cell C_1 is divided once into the children C_{11} and C_{12} and introduces a hanging node on the edge of the tetrahedral cell C_2. If C_{11} was divided again, another hanging node on this edge would be introduced. So C_2 has also to be divided before the next division of C_{11}. An iterative process runs through the field focussing on the tagged cells. If a division would introduce a second hanging node on any edge, the respective cell has also to be tagged for division.

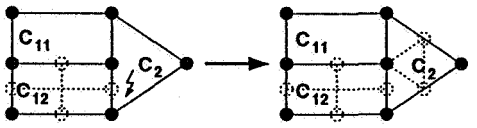


Fig. 5: Tagging of C_2 in order to prevent a second hanging node

NUMERICAL RESULTS

The first case is the supersonic inviscid flow ($Ma_\infty = 6$) about a blunt body. A coarse hexahedral mesh was modified in order to obtain a hybrid mesh with hexahedral cells near the body and tetrahedral cells in some distance from the surface. In this test case the metrical setup and the capability of cell division were subject to examinations. Figure 6 shows the position of the shock and the effect of a four times refinement that leads to a good shock resolution even for this coarse initial mesh.

The second case was chosen to test the Navier Stokes formulation of the scheme. For that purpose a pure hexahedral 60x40x3 grid over a flat plate was generated. Figure 7 shows the comparison at different points between the Blasius solution and the computed solution for a subsonic ($Ma_\infty = 0.5$) laminar flow with a Reynolds Number of 5000. The grid for this case is not adapted to the solution. The results are conform to the analytic solution. Inflow and outflow conditions are computed as proposed by Whitfield in [8].

The first 3D test case is the transsonic flow around an ONERA M6-Wing. Flow conditions are $Ma_\infty = 0.84$ with an incidence of 3.06°. The grid has been adapted twice. The initial grid contains about 78,000 nodes, 11,000 hexahedra and 370,000 tetrahedra. The once adapted grid contains about 113,000 nodes, 33,000 hexahedra and 403,000 tetrahedra, while the finest grid contains about 211,000 nodes, 95,000 hexahedra and 440,000 tetrahedra. The computed flow field is shown in figure 8. The characteristic λ-shock on the upper side of the wing is nicely resolved. Oscillations at interfaces between cells of different refinement levels occure. Those wiggles are subject to investigations in the future.

Also the AGARD 01 test case has been calculated. Figure 9 shows the five times adapted grid and the respective solution. The shock resolution is acceptable, the shock regions have been refined during each refinement step. The shock on the upper side is located at 0.63 chord length and the shock on the lower side at 0.37 chord length.

The adaptation for the AGARD 03 test case offers more difficulties. As shown in figure 10 the shock behind the airfoil is not resolved very well. The initial grid contains 6,000 nodes. The grid is a three times stacked two dimensional grid. The five times adapted grid contains about 55,000 nodes (also three stacked planes) and the shock distance is 2.75 chord length behind the trailing edge.

CONCLUSIONS

A finite volume scheme using hybrid grids was presented. The employed grids consist of hexahedral cells near body surfaces and tetrahedral cells connecting the hexahedral domains and the outer boundaries. The use of hexahedral cells offers the possibility to resolve viscous dominated flows such as boundary layers efficiently and accurately by applying high aspect ratio cells in those areas. Because of the tetrahedral parts, grids become quite flexible and the generation of grids, even for complex configurations, is relieved very much compared to structured approaches.

In a metrical setup a dual mesh is computed from the initial computational grid. This dual mesh covers the entire computational domain and connects the two grid types naturally. The feasibility of using hybrid grids even for three dimensional flows is shown, but since effective tools for the generation of hybrid grids are not available yet at DLR the presented test cases can not prove the expected advantages of the approach. The calculation of inviscid fluxes is efficient and accurate. Shocks are captured nicely by the employed upwind flow solver. Also the formulation to calculate the viscous fluxes has proved its accuracy. Difficulties still occure at interfaces between refined and unrefined cells.

The next step on the way to an automatic system to compute viscous flows around three dimensional configurations is the extension of the hybrid code to prismatic cells in the vicinity of surfaces. The prismatic cells can substitute for hexahedral cells, as they yield the same advantages as hexahedral cells and algorithms for generating prismatic grids are known from the literature ([9], [10]). The data structure of the new version is changed from a point based to an edge based structure. In order to enable a vectorization a edge colouring is employed. The performance on a NEC-SX 3 is about 1 GFlop, including the metrical setup that is not vectorizable. Also the adaption criteria have to be improved for the new version. This improvement requires intensive studies of different methods.

REFERENCES

[1] M.J. Aftosmis. Viscous flow simulation using an upwind method for hexahedral based adaptive meshes. AIAA 93-0772, 1993.

[2] J.E. Melton, S.D. Thomas, and G. Cappucccio. Unstructured Euler flow solution using hexahedral cell refinement. AIAA-91-0637, 1991.

[3] M. Soetrisno, S.T. Imlay, and D.W. Roberts. A zonal implicit procedure for hybrid structured-unstructured grids. AIAA-94-0645, 1994.

[4] P. Vijayan and Y. Kallinderis. A 3d finite volume scheme for the Euler equations on adaptive tetrahedral grids. *Journal of Computational Physics*, 113:249–267, 1994.

[5] M.S. Liou and C.J. Steffen. A new flux splitting scheme. *Computers and Fluids*, 107(1):23–39, 1993.

[6] N. Kroll and R. Radespiel. An improved flux vector split discretisation scheme for viscous flows. DLR-FB 93-53, 1993.

[7] T.J. Barth and D.C. Jesperson. The design and application of upwind schemes on unstructured meshes. AIAA-89-0366, 1989.

[8] D.L. Whitfield. Three-dimensional unsteady Euler equation solutions using flux vector splitting, 1983.

[9] K. Nakahashi. Adaptive prismatic grid method for external viscous flow computations. AIAA-93-3314-CP, 1993.

[10] M.J. Marchant and N.P. Weatherill. Unstructured grid generation for viscous flow simulations. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical grid generation in computational fluid dynamics and related fields*, pages 151–162, 1994.

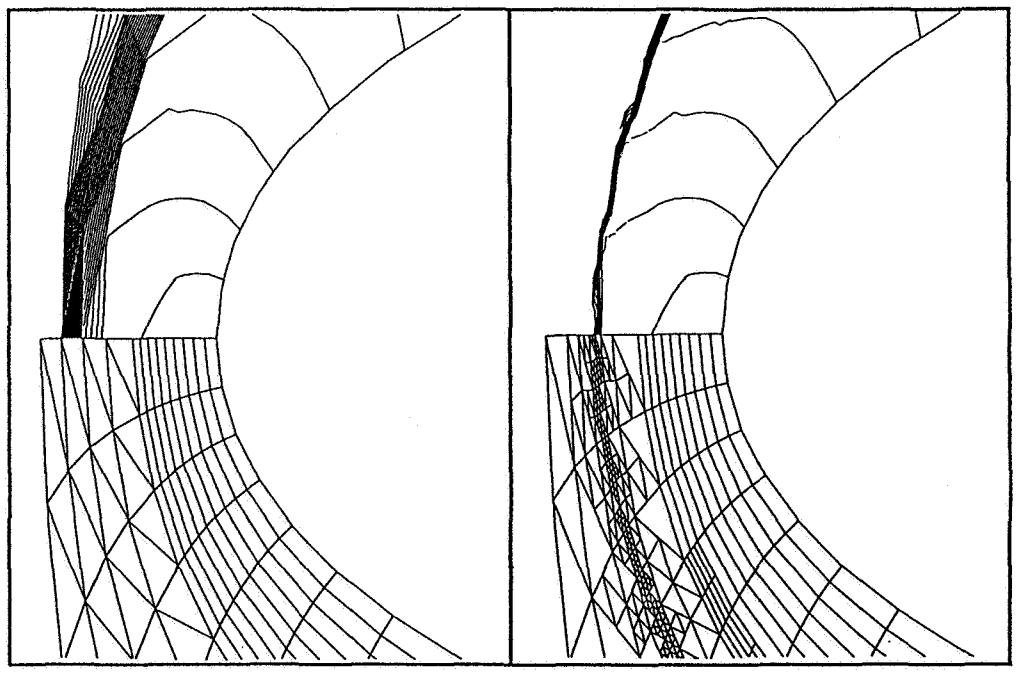


Fig. 6: Flow around a blunt body: Initial grid and four times adapted grid with respective solutions

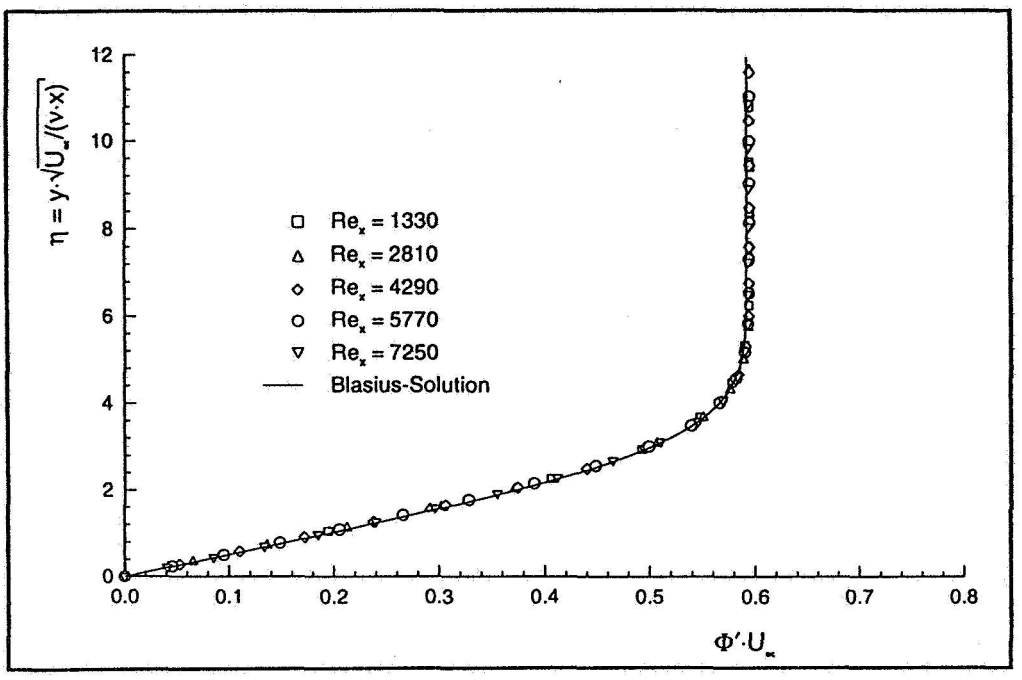


Fig. 7: Laminar flow over a flat plate: Computed Solution compared to analytic Solution

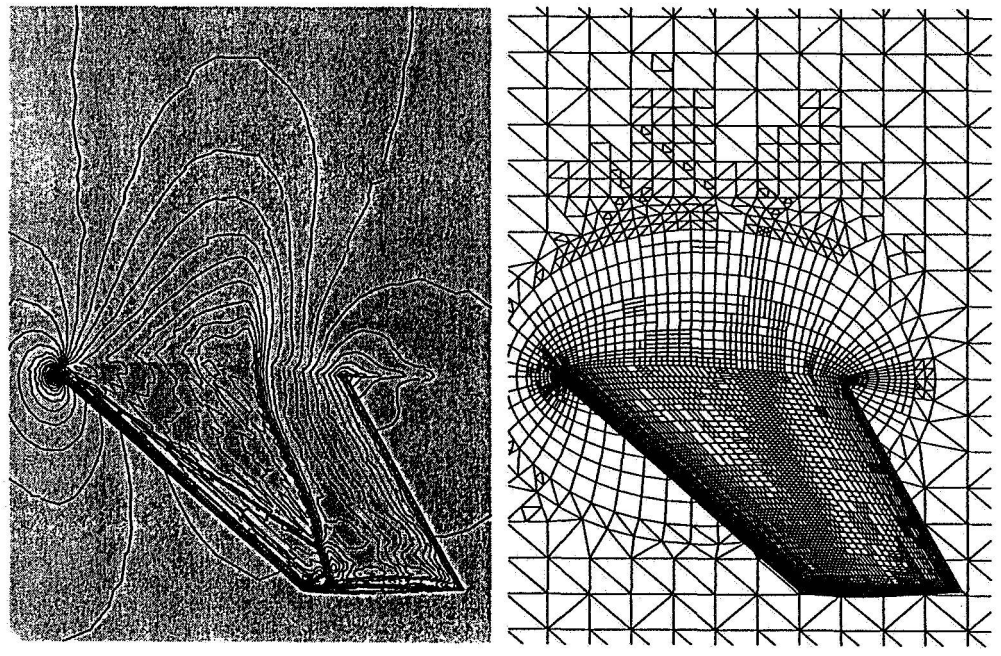


Fig. 8: Transsonic flow around ONERA M6-Wing: twice adapted grid and respective solution

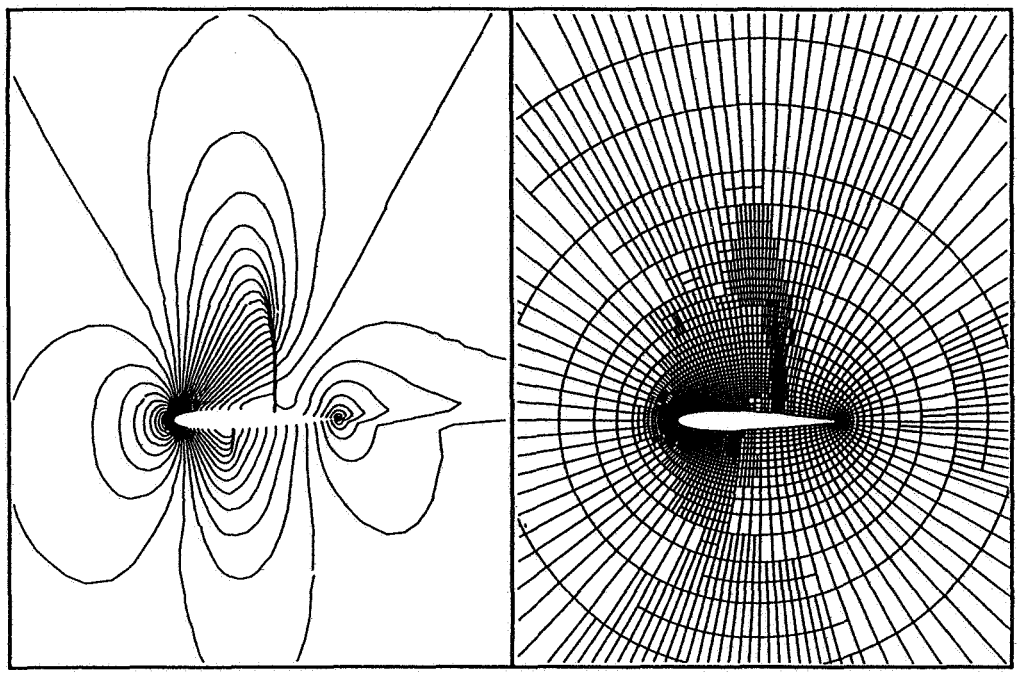


Fig. 9: AGARD01 test case: Five times adapted grid and respective solution

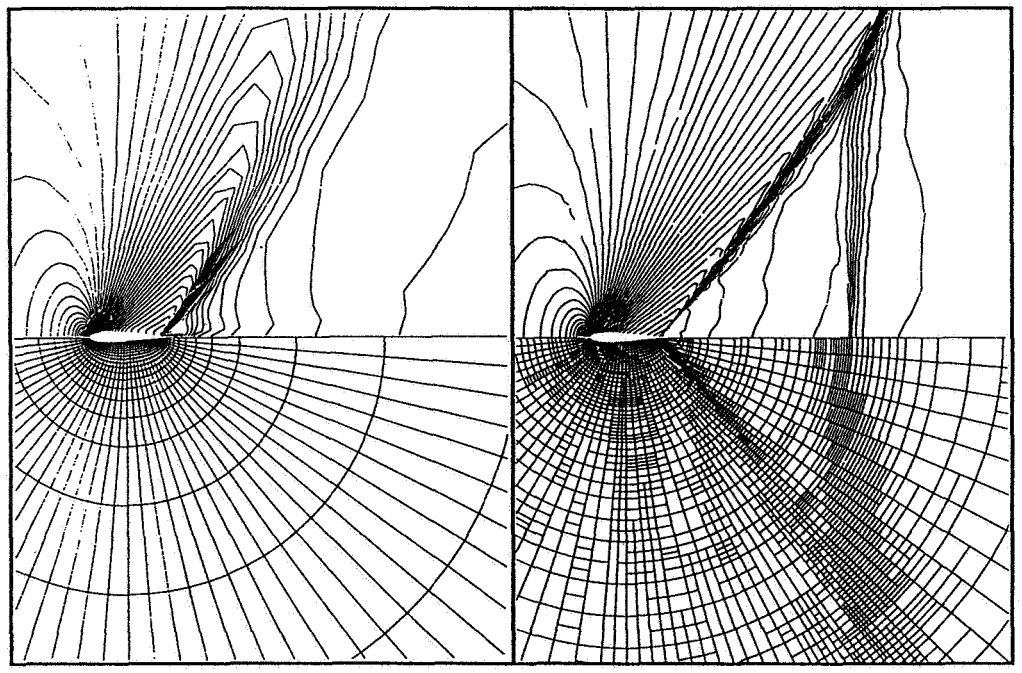


Fig. 10: AGARD03 test case: Initial grid and initial solution and five times adapted grid with respective solution

A STRUCTURED MULTI-BLOCK SOLUTION-ADAPTIVE MESH ALGORITHM WITH MESH QUALITY ASSESSMENT

Clint L. Ingram, Kelly R. Laflin, and D. Scott McRae
North Carolina State University
Raleigh, NC

SUMMARY

The dynamic solution adaptive grid algorithm, DSAGA3D, is extended to automatically adapt 2-D structured multi-block grids, including adaption of the block boundaries. The extension is general, requiring only input data concerning block structure, connectivity, and boundary conditions. Imbedded grid singular points are permitted, but must be prevented from moving in space. Solutions for workshop cases 1 and 2 are obtained on multi-block grids and illustrate both increased resolution of and alignment with the solution. A mesh quality assessment criteria is proposed to determine how well a given mesh resolves and aligns with the solution obtained upon it. The criteria is used to evaluate the grid quality for solutions of workshop case 6 obtained on both static and dynamically adapted grids. The results indicate that this criteria shows promise as a means of evaluating resolution.

INTRODUCTION

It is generally agreed that a geometry conformal structured mesh topology has demonstratable advantages for use in the numerical simulation of high Reynolds number viscous flows. However, the generation of geometry conformal structured meshes over complex shapes has proven to be a difficult task for two reasons: it is difficult to produce multiple block grids automatically that provide the necessary structured-mesh topology around the shape, and the grid must provide resolution of complex shear layers, shock waves, slip surfaces, etc. when the final location and extent of these features is not always known. Current research is underway to overcome the first difficulty, and is showing promise that multiblock structured meshes can be generated automatically.

It is the purpose of the present research to complement these advances by developing a means of adapting dynamically the initial structured mesh to the solution as it evolves, so that the features noted above plus any other chosen will be resolved automatically without laborious pre-clustering.

A brief review of the technique developed by Benson and McRae (refs. 1 to 4) for dynamic adaptation of 2-D and 3-D single block grids will be followed by a description of the extension of the dynamic adaptation algorithm to 2-D multi-block grids, including the adaptation of block boundaries (refs. 5 and 6) In all of the implementations of the solution adaptation algorithm presented herein, the terms which correct the final solution for cell volume variation with time have been included in the formulation. This ensures that temporal accuracy is preserved as the mesh translates to resolve moving features of the solution.

Finally, the question of cell shape (i.e., skewness, aspect ratio, etc.) effect on the solution has been a continuing issue for discussion among those who must defend the results obtained using structured meshes. An effort is presently underway to identify the errors which result from the interaction of the numerical algorithm with the mesh, including cell shape and mesh movement. Preliminary results from this research are included.

NUMERICAL PROCEDURE

Conservation Law

Conservation laws form the basis of our study of compressible fluid flow considered as a continuum. Since the volume over which conservation is being enforced can change independently, it is appropriate to examine briefly how these laws are obtained.

A conservation law results from the concept that a quantity or property may be physically conserved in both time and space. The mathematical definition of a conservation law results when the time rate of change of some quantity B summed over a given volume is shown to be equal to a quantity Γ, or

$$\frac{d}{dt}\int_V B dV = \Gamma. \tag{1}$$

In this application, B is either the density, linear momentum, or total energy. Applying the theorem of Leibnitz to the equation, two integrals result, one to account for the time rate of change of B summed over the volume and one for the motion of the boundary, *i.e.*

$$\frac{d}{dt}\int_V B dV = \int_V \frac{\partial B}{\partial t} dV + \oint_S B\vec{\omega}\cdot d\vec{s} = \Gamma. \tag{2}$$

It is common practice to assume the region or volume is not changing with time but only translating in space at the local fluid velocity. This is equivalent to assuming that the total mass is constant or the volume is "material." With this assumption, $\vec{\omega}$ becomes the local fluid velocity and thus,

$$\frac{d}{dt}\int_V B dV = \int_V \frac{\partial B}{\partial t} dV + \oint_S B(u\,\hat{i} + v\,\hat{j} + w\,\hat{k})\cdot d\vec{s} = \Gamma. \tag{3}$$

Equation 3 is easily reduced to the familiar form of the conservation laws, which are correct only if the volume of interest is fixed in magnitude or material. However, if the volume is allowed to expand or contract independently in time, then the volume boundaries no longer move at the local fluid velocity and a new relationship for $\vec{\omega}$ must be determined (ref. 7).

Consider a one-dimensional flow where the boundary of a region is moving with speed $\dot{x}$ and the fluid is moving at a velocity u. In this case, the fluid-surface interface velocity is $u - \dot{x}$, which is the velocity that should be used in the surface integral resulting from Leibnitz's Rule. Substituting the corresponding definition of $\vec{\omega}$ for a three-dimensional volume into Equation 3 results in

$$\int_V \frac{\partial B}{\partial t} dV - \oint_S B\left(\dot{x}\,\hat{i} + \dot{y}\,\hat{j} + \dot{z}\,\hat{k}\right)\cdot d\vec{s} + \oint_S B\left(u\,\hat{i} + v\,\hat{j} + w\,\hat{k}\right)\cdot d\vec{s} = \Gamma, \tag{4}$$

which is the correct statement of conservation for an arbitrary volume allowed to expand or contract in time.

If Γ, defined to ensure closure of the conservation laws, is substituted into Equation 4 and B is redefined to be the vector of properties that are conserved in fluid flow, $U = \left[\rho, \rho\vec{V}, E_t\right]^T$, a more familiar form of the conservation equations can be written as:

$$\int_V \frac{\partial U}{\partial t} dV - \oint_S U\vec{x}\cdot d\vec{s} + \oint_S \vec{A}\cdot d\vec{s} = 0. \tag{5}$$

$\vec{A}$ is a vector containing the flux components in the cartesian frame, $E\,\hat{i} + F\,\hat{j} + G\,\hat{k}$ and $\vec{\dot{x}}$ is the speed at which the surface is expanding or contracting, $\dot{x}\,\hat{i} + \dot{y}\,\hat{j} + \dot{z}\,\hat{k}$.

The second term accounts for the amount of U entrained or lost as a result of the change in volume over time, where $\vec{s}$ is the product of the surface unit normal vector and the surface area, and is the area of the projection of the given surface into the three spatial coordinate axes, *i.e.* $\vec{s} = s_x\,\hat{i} + s_y\,\hat{j} + s_z\,\hat{k}$.

Discretizing Equation 5 for a generic hexahedron changing in time from the n^{th} to the n^{th}+1 time level,

$$(UV)^{n+1} - (UV)^n \mp \left[(U\Delta V)_{i\pm\frac{1}{2},j,k} + (U\Delta V)_{i,j\pm\frac{1}{2},k} + (U\Delta V)_{i,j,k\pm\frac{1}{2}}\right]^n \pm\Delta t\left[\hat{E}_{i\pm\frac{1}{2},j,k} + \hat{F}_{i,j,k\pm\frac{1}{2}} + \hat{G}_{i,j,k\pm\frac{1}{2}}\right]^n = 0, \tag{6}$$

with $\hat{E}\Big|_{i+\frac{1}{2}} = \left(\vec{A}\cdot\vec{s}\right)\Big|_{i+\frac{1}{2}}$. $\hat{F}$ and $\hat{G}$ are similarly defined. ΔV represents the volume swept by the surface as it changes with time.

Modified Runge-Kutta Algorithm

The Navier-Stokes equations, plus the continuity and energy equations, are integrated in time using an explicit, multi-stage Runge-Kutta algorithm (ref. 8). The inviscid fluxes are described using the Advective Upwind Split Method of Liou and Steffen (ref. 9) and extended to higher-order spatial accuracy using MUSCL differencing and the van Albada and MINMOD limiters (refs. 10 and 11). For this work, a two-stage Runge-Kutta scheme will be utilized. The coefficients for each of the stages are $\alpha^1 = \frac{1}{2}$ and $\alpha^2 = 1$, which results in a second-order accurate scheme in time.

Although the changes in cell volume are independent of the time-advancement scheme, the manner in which this term is implemented is not independent and may decrease temporal accuracy. When the explicit Runge-Kutta algorithm is integrated as a single step, including the cell volume variation term, the mesh movement is at the $n-1$ level ($\Delta^{n-1}V$) and is based on mesh movement due to the prior time step. In an attempt to more closely couple mesh movement and solution, the Runge-Kutta algorithm is split into two steps. The first step integrates the terms in Equation 6 above which do not involve the ΔV term and can thereby be considered "steady" terms. The terms that involve ΔV, and thereby necessary to preserve temporal accuracy, are integrated in the second step of the procedure. Splitting the algorithm permits adaptation of the mesh based on weight functions determined from the steady portion of the integration. The second step of the integration algorithm is then applied to correct the solution for mesh movement to the $n+1$ time level. In the present work, the correction for mesh variation is first, thereby being equivalent to a first order interpolation of the steady solution to the new grid location. This algorithm is detailed below.

The modified, two-stage, time-advancement scheme that results from splitting the equations into a "steady" and an "unsteady" portion is as follows:

Stage i:

$$U^{(i)} = U^{(i-1)} - \alpha^{(i)}\frac{\Delta t}{V^n}\left\{\Delta_\xi \hat{E}^{(i-1)} + \Delta_\eta \hat{F}^{(i-1)} + \Delta_\zeta \hat{G}^{(i-1)}\right\} \tag{7}$$

Equation 7, the first step of the modified scheme, represents the advancement of the simulation in time for a static or non-moving mesh. In the definitions of the fluxes, $\vec{s}$ is evaluated at the n^{th} time level. The next step is to redistribute the nodes in the mesh using the method detailed below with

weight functions based on the solution at time level (2) obtained with the above equation. Finally, the time varying volume portion of the equations is used to advance the solution vector to the new mesh level by taking into account the amount of U that is entrained/lost by the expansion/contraction of the volume. This step is equivalent to correcting the convective velocity for a moving mesh. The equation used has the same form as the Runge-Kutta algorithm.

$$(UV)^{n+1} = V^n U^{(2)} + \left\{\Delta_\xi \left(U^{(2)}\Delta^n V\right) + \Delta_\eta \left(U^{(2)}\Delta^n V\right) + \Delta_\zeta \left(U^{(2)}\Delta^n V\right)\right\} \tag{8}$$

In Equation 8, the terms $\Delta^n V$ represent the change in volume between the n^{th} and $n^{th}+1$ time level.

ADAPTIVE ALGORITHM

The adaption process, to ensure resolution of solution features as they translate and evolve, takes place between the split steps of the integration scheme. Weight functions are based on the first stage of the integration scheme. A center of mass scheme is then used to relocate the points in a transformed parametric space. A straightforward truncation and interpolation scheme is used to determine the new location in physical space corresponding to the new parametric coordinates. An overview of these steps follows.

Weighting Function

The primary goals of an adaptive procedure must be to improve solution accuracy. A by-product of increased accuracy should be the ability to locate, identify and determine the extent of all features of the solution with increased certainty. A conventional examination of the truncation error of discrete integration techniques results in an infinite series of terms consisting of derivatives of the dependent variables multiplied by functions of the discrete spacing.

In general, these truncation error terms are large near discontinuities and rapidly changing regions in the flow. Therefore, it was decided to use the difference of the dependent variables as the basis for a suitable clustering function. Higher differences (usually second) may also be used if sufficient nodes are available to resolve them. The weight function is formed using a linear superposition of the differences where σ_k is a biasing coefficient, ϕ_k is the magnitude of the gradient, and k indicates a given flow variable. The maximum value of the difference for each flow variable varies, so a second coefficient will be introduced to scale the respective gradients thereby insuring the desired weighting (usually equal peak values) before biasing. The complete weighting function is then determined by:

$$\omega = \sum_k \sigma_k \alpha_k \left|\phi_k\right|. \tag{9}$$

In order to provide automated control of the degree of adaption, an average weight function is first determined along each coordinate line. This weight function is then limited according to predetermined functions of this average:

$$\omega_{min} \leq \omega \leq \omega_{max}. \tag{10}$$

After limiting, the weight function is then smoothed to control cell skewness. For the baseline weighting function, biasing coefficients, percentage values, and ratios are input at the beginning of the program. The remainder of the process occurs without user interaction.

Transformation To Parametric Plane

Using a similar idea to that used for obtaining a computational space for a single block grid, physical space can be mapped into a separate parametric space. The mapping is general and the standard requirements for one-to-one correspondence of grid-points and smooth mesh lines are maintained, thereby guaranteeing an inverse transformation. This initial trasnformation between physical and parametric space remains fixed during the computation and is used as a reference for remapping the changed grid locations into physical space after adaption. "Fixed" in this instance means that the original transformation does not change with time. The reason for performing the adaption in a parallelepiped in parametric space is twofold: intricate boundaries in the physical space will map into a plane in the parametric space, and a simple equation results for remapping the adapted parametric space into corresponding new locations in physical space. Adaption in parametric space will determine new values of ξ, η, and ζ for each node. The next step is to determine which metrics of the original transform must be used to map the new node locations (ξ, η, and ζ) to physical space such that the original transform is preserved. This maintains the ordering of the grid-points in both spaces.

Since the original mapping defines a parallelepiped in the parametric space and $\Delta\xi$=$\Delta\eta$=$\Delta\zeta$=1 by definition, the original nodes or grid-points are located at integer values which correspond directly to the i, j, k which are used to reference the arrays.

$$int(\xi^o, \eta^o, \zeta^o) = i, j, k \tag{11}$$

Adaption takes place in the parametric space, after which a mapping to determine the new x, y, and z locations from the new ξ, η, and ζ positions must be obtained. This process begins with a discrete approximation to the differential dx:

$$\Delta x = x_\xi \Delta\xi + x_\eta \Delta\eta + x_\zeta \Delta\zeta \tag{12}$$

Since the approximations to the differentials in the above equation are just differences, they are chosen to be the new mesh node location, referenced with i, j, k , minus a nearest original position, denoted with the superscript ^{o}. The metric derivatives are also identified with the superscript ^{o}, since the transform is only determined initially.

$$x_{i,j,k} - x^o = x^o_\xi(\xi_{i,j,k} - \xi^o) + x^o_\eta(\eta_{i,j,k} - \eta^o) + x^o_\zeta(\zeta_{i,j,k} - \zeta^o) \tag{13}$$

If the point at i, j, k is moved to a new position in the parametric space, the corresponding new position in the physical space must be determined. The vertex of the cube of the original parallelepiped that contains the new ξ, η, ζ position and is nearest the origin of the parametric mesh can be found by rounding these values down to integers.

$$\begin{aligned} l &= int(\xi_{i,j,k}) \\ m &= int(\eta_{i,j,k}) \\ n &= int(\zeta_{i,j,k}) \end{aligned} \tag{14}$$

The vertex of the cube of the original parallelepiped that is closest to the new ξ, η, ζ position is given by the nearest integer function:

$$ln = nint(\xi_{i,j,k}) = l + 1$$

$$mn = nint(\eta_{i,j,k}) = m + 1$$

$$nn = nint(\zeta_{i,j,k}) = n \tag{15}$$

Recall that ξ, η, and ζ were defined to be integers that corresponded directly to the reference coordinates i, j, k and therefore, the values defined in Equations 14 and 15 correspond directly to the array positions for x, y, and z of the original grid-point at those respective vertices.

The metrics, x_ξ, x_η, and x_ζ, are appoximated such that they represent the distance between adjacent nodes in the ξ-, η-, and ζ-directions. The metrics are stored in arrays as forward differences and therefore, they are based at the point l, mn, nn for the ξ-direction, ln, m, nn for the η-direction, and ln, mn, n for the ζ-direction. By using the integer value of ln in place of ξ^o in Equation 13, this will subtract the distance $x^o_\xi(\xi - \xi^o)$ if $\xi_{i,j,k}$ is closer to the ξ-axis than the nearest original point and add the distance if $\xi_{i,j,k}$ is greater than the nearest original point. The result is similar for η^o and ζ^o. Therefore, a final expression for the new value of x in the physical space is:

$$x^n_{i,j,k} = x^o_{ln,mn,nn} + x^o_{\xi_{l,mn,nn}}(\xi_{i,j,k} - ln) + x^o_{\eta_{ln,m,nn}}(\eta_{i,j,k} - mn) + x^o_{\zeta_{ln,mn,n}}(\zeta_{i,j,k} - nn), \tag{16}$$

which is a Taylor series expansion in three dimensions utilizing the initial grid as a reference grid. Similar equations can be derived for y and z, by substituting for x.

The above can be shown to preserve the original boundary shape. Choosing the boundary where η=const=1., note that a term drops out of Equation 16 leaving

$$x^n_{i,j,k} = x^o_{ln,mn,nn} + x^o_{\xi_{l,mn,nn}}(\xi_{i,j,k} - ln) + x^o_{\zeta_{ln,mn,n}}(\zeta_{i,j,k} - nn). \tag{17}$$

This maps into the surface in the physical space that corresponds to the original boundary on which $\eta_{i,j,k}$=constant.

Adaption Procedure

With exceptions noted below for multi-block grids, performing the adaption in parametric spaces allows the restrictions on movement in the method of mean-value relaxation (ref. 12) and the method of minimal moments (refs. 13 and 14) to be removed.

Considering the grid-points to be point masses with the weighting function analagous to the mass, the location of the center of mass can be determined for the computational cell in three dimensions as:

$$\xi_{cm_{i,j,k}} = \frac{\sum_{k-1}^{k+1}\sum_{j-1}^{j+1}\sum_{i-1}^{i+1}\omega_{i,j,k}\xi_{i,j,k}}{\sum_{k-1}^{k+1}\sum_{j-1}^{j+1}\sum_{i-1}^{i+1}\omega_{i,j,k}} \tag{18}$$

This determines the movement of the point at i, j, k to a localized center of mass. This calculation is repeated for every point in the parametric domain with the boundaries being calculated using a reduced stencil. The grid points are locally redistributed until the entire grid of weights is globally at rest or has moved the desired amount.

For single-block grids, adaption in parametric space effectively eliminates crossover of mesh lines if the initial grid is generated by an elliptical grid generator. Crossover will result only when the center of mass of the stencil is outside of its original domain, which is most likely to occur at concave boundaries. These boundaries are not present in parametric space for single block grids.

Implementation Of The Algorithm

The adaptation algorithm is utilized in the following way: the transform and inverse transform for mapping the physical space into the parametric space is computed initially; the first part of the integration step on the governing equations is completed; the grid is adapted based on the solution from that step; the second step of the integration is performed; the transformation for the governing equations is recomputed; and the process starting with the integration step is repeated until the grid reaches a steady-state or until a total time in the solution evolution is reached. When the grid reaches a steady-state, this implies that only small changes are occurring in the flow variables that cause small changes in the weight function distribution. The grid motion is then ended and the simulation continues on the static adapted grid until the desired flow convergence is obtained. Although the algorithm was developed to be a dynamic solution-adaptive method, it can also be used as part of a grid refinement study. The first step is to run a simulation to convergence on an initial grid. Using the adaptive algorithm, the grid-points of the original grid are relocated based on the solution, which is interpolated to the new grid as part of the algorithm. The simulation is rerun using the clustered mesh as the initial grid. This can be repeated until the desired resolution is attained.

MULTI-BLOCK GRIDS

The multi-block solver/adaptor scheme is designed to accomodate general multi-block geometries and permit the application of the solution-adaptive mesh algorithm to structured, multi-block initial grids by specifying only the block connections (refs. 5 and 6). Therefore, the coupling of the multi-block solver with the multi-block solution-adaptive mesh algorithm allows dynamic adaption for flows over complex structures for which single-block meshes are inadequate.

In this case, effective adaption without *a priori* knowledge of the flow requires that the block boundaries be adapted in addition to the interiors. This implies that new mesh locations from one block can move on to the fixed reference locations of an adjacent block. Logic must, therefore, be installed to maintain overall connectivity and continuity of the adapted grids as this occurs.

The block connectivity is governed by a set of specified block splicings. The block splicings can be defined by any of ten possible combinations. That is, any block side can connect to any other block side. The only limitation is that both blocks of a splicing must contain the same number of mesh cells in the direction along the splicing. Each block may have its own orientation (ξ, η direction). Therefore, there may not be a global ξ, η direction. This feature not only allows the solver/adaptor to handle more complex grid structures in which a global ξ, η direction is not possible, but also allows imbedded singular points to reside in the grid domain. These splicings are defined in an input file along with the dimensions and boundary conditions of each block. The grid-blocks are read in as separate files, and can be generated using GRIDGEN or any other appropriate grid generator.

MULTI-BLOCK SOLVER

The extension of the solver to a multi-block grid is straightforward. Each block of a grid is treated as a separate domain and boundary conditions enforce propagation of data from adjacent

blocks or boundaries. Conditions along splice block boundaries are determined by continuity with adjacent blocks. Reference 6 describes the procedure and illustrates how the process provides a known boundary condition along the splices of the blocks in both directions. This process is dubbed a *splice boundary condition* and is performed at the same time that all other boundary conditions (viscous wall, free stream, etc.) are enforced.

MULTI-BLOCK ADAPTOR

The adaptive algorithm is extended to include adaption in complex multi-block domains with dynamically adapted block boundaries. The extension of the adaptor to multi-block grids is not as straightforward as the overlay approach above.

For multi-block configurations, the conditions for grid point movement lead to three primary types of outer block boundary and splice block boundary intersections. Reference 6 describes these intersections in detail. Points are constrained from movement around outer ξ, η corners in the parametric space. However, the splice boundary/outer boundary intersection of two adjacent blocks can move along the outer boundary, which allows adaptive movement of the splice boundary. One exception to movement of a splice boundary/outer boundary intersection occurs when more than one splice boundary intersects an outer boundary at the same point. This results in a concave corner in the parametric space and can cause crossover of grid lines if movement of such a point is allowed without special provision. Adaption of the block boundaries while preserving the original structure of the mesh leads to these criteria:

1. Points can move from one original transformed block to another. Therefore, the new mesh point locations must be determined, no matter which of the original transformed blocks they fall in.
2. Physical boundary conditions must be specified based on current mesh point location.
3. Crossover of grid lines near concave corners in the parametric space must be prevented.

Modifications to the adaptor are made to meet these criteria. Logic is added to determine the locations of points that move into other blocks and to insure that the boundary points acquire or retain the correct boundary condition representation. These processes are also detailed in Reference 6.

As noted previously, a center of mass scheme may result in crossover in the vicinity of concave boundaries. This does not occur in the parametric space for single block grids since no concave boundaries exist. When the method is extended to multi-block grids, concavities may be unavoidable in parametric space. The movement of the corner point, itself, can be easily limited by fixing its location at the concave corner. By locally restricting grid point movement near the concavity, crossover can be prevented. This is done by requiring mesh lines in the vicinity of the concavity to follow a von Neumann condition. The slopes of the lines at the concavity are set to be normal and change linearly to the slope defined by the adaption criteria at a set number of nodes away.

MULTI-BLOCK BOUNDARY CONDITIONS

Boundary conditions are coded generally as a set of options for defining any block boundary. For the transonic 0012 airfoils, four options are required. The block boundaries representing the far field inflow are assigned freestream density and velocity values and obtain the pressure values from the integration of the appropriate compatabity relation. The block boundaries representing the far field outflow are assigned freestream pressure values and obtain the density and velocity values from the integration of the appropriate compatibity relations. The block boundaries representing the surface are assigned entropy corrected inviscid wall conditions as described in Reference 15. The final boundary condition is the block splice boundary condition described above.

MESH QUALITY ASSESSMENT

The quality of a computational mesh is generally considered to be important to both the convergence and accuracy of a computational solution. Distorted mesh cells, e.g., those stretched and/or skewed, may cause significant errors in the developing solution. In addition, mesh cells whose sides are not aligned with local flow velocities or discontinuities can also cause inaccuracies. When adaptive mesh redistribution is used, the mesh is deformed automatically as the solution progresses. Therefore, distorted cells may occur near strong features in the course of the adaptive process. The following solution dependent mesh quality measure (SDMQM) has been developed to help assess the impact of mesh redistribution adaptation on computational solutions. It will also be useful for evaluating the results of standard mesh generation codes. The SDMQM couples the mesh cell geometry with the local solution field via the governing equations.

Consider the integral form of the conservation equations,

$$\frac{d}{dt}\int_V U dV + \oint_S \left(\vec{F}\cdot\vec{n}\right) ds = 0. \tag{19}$$

The second term of the above equation couples the solution field with the cell geometry. In the usual formulation of finite volume algorithms, this term is expressed as,

$$\oint_S \left(\vec{F}\cdot\vec{n}\right) ds = \sum_S \vec{\overline{F}}\cdot\vec{s}, \tag{20}$$

where $\vec{s}$ is the product of the surface unit normal vector and the surface area, and the overbar indicates average values over a cell surface, S. This equation is exact for planar sided cells and gives the average value of $(\nabla\cdot\vec{F})$ within the mesh cell. This is shown by using the divergence and mean-value theorems

$$\sum_S \vec{\overline{F}}\cdot\vec{s} = \int_V (\nabla\cdot\vec{F}) dV = V\left(\overline{\overline{\nabla\cdot\vec{F}}}\right)$$

where the double overbar indicates the average value over a cell volume, V.

Assume that the terms $\vec{\overline{F}}\cdot\vec{s}$ are evaluated at the center of their respective cell sides. Then, by using Taylor's series expansion we expand these terms about the cell center (cc) to obtain,

$$V\left(\overline{\overline{\nabla\cdot\vec{F}}}\right) = \sum_S \vec{\overline{F}}\cdot\vec{s} = V\left(\nabla\cdot\vec{F}\right)_{cc} + RE.$$

The resolution error, RE, is dependent on the cell geometry and the solution field and is a measure of the product of the cell volume and the difference between the average value of $(\nabla\cdot\vec{F})$ over the cell volume and the point value of $(\nabla\cdot\vec{F})$ at the cell center. The first few terms of the RE are expressed in Figure 19 for a 2-D quadrilateral cell.

We now have,

$$\mu = \frac{|RE|}{V} = \left|\left(\overline{\overline{\nabla\cdot\vec{F}}}\right) - \left(\nabla\cdot\vec{F}\right)\Big|_{cc}\right|,$$

which is a measure of the mean deviation of $(\nabla\cdot\vec{F})$ at the mesh cell center. Finally, μ is a vector quantity so we calculate $\|\mu\|_2$ which we dub the solution dependent mesh quality measure.

In the case of structured meshes, μ can be efficiently estimated by applying the transformation $(x, y, z) = (x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta))$ to $V(\nabla \cdot \vec{F})_{cc}$ which yields

$$\mu = \frac{1}{V} \left| \sum_S \left(\vec{\overline{F}} \cdot \vec{s} \right) (E_\xi + F_\eta + G_\zeta)|_{cc} \right|. \tag{21}$$

The first term is known from the finite volume computation and the derivatives of the second term are estimated by using high-order difference approximations in computational space. In the current computations fourth-order approximations are used. The μ is not an estimate of the local solution error, but, rather, it is a measure of the local flow resolution which directly effects the solution error. If $\|\mu\|_2$ is large, then the flow within the cell is poorly resolved. When this is the case, information about the flow is lost which in turn diminishes an algorithm's ability to accurately approximate new cell interface flux values for the next integration step. On the other hand, if $\|\mu\|_2$ is small, then it is assumed that it is small at all locations within the cell indicating good flow resolution. Well resolved flow within all mesh cells leads to more accurate cell interface flux approximations and higher accuracy.

There are combinations of the cell geometry and solution field for which $\|\mu\|_2$ is small at the center of the cell but would be large if it were evaluated at some other point within the cell. However, in these circumstances the cell geometry appears to be optimal for the corresponding solution field. For example, A local linear distribution of the solution field will cause $\|\mu\|_2$ to be predicted as zero, but only if the cell being examined and those involved in the approximation of the derivatives appearing in Equation 21 are orthogonal, of equal size, and aligned with the gradient of the solution field.

In theory $\|\mu\|_2$ is a measure of the local flow resolution within a cell, but the current centered stencil used in approximating the cell center value of $(\nabla \cdot \vec{F})$ is essentially a high-order finite difference evaluation. Therefore, $\|\mu\|_2$ may be thought of as a measure of the difference between a finite volume and a finite difference formulation connected with a particular cell. When $\|\mu\|_2$ becomes zero the finite volume scheme reduces to a finite difference scheme of the order of accuracy with which $(\nabla \cdot \vec{F})$ is approximated.

A third way to view $\|\mu\|_2$ is to think of the finite volume formulation as being equivalent to second-order accurate finite difference evaluations of $(\nabla \cdot \vec{F})$ in computational space. Then $\|\mu\|_2$ is a measure of the difference between second-order and fourth-order derivative approximations and predicts a second-order truncation error associated with calculating $(\nabla \cdot \vec{F})$ within a given cell. By interpreting $\|\mu\|_2$ in this manner, conditions which cause $\|\mu\|_2$ to be small can be easily assessed. Also, it is clear that flow discontinuities (shocks, contact surfaces, slip lines) will be predicted as causing a large $\|\mu\|_2$, despite a reduction in mesh spacing, since the higher derivative terms of the aforementioned truncation error will always persist. With this in mind, we can use $\|\mu\|_2$ to examine the effect of adaptive mesh redistribution on a computed solution.

MULTI-BLOCK BOUNDARY RESULTS

Case 1: AGARD 01

AGARD 01 is a transonic 0012 airfoil. The flow conditions for this test case are $M_\infty = 0.8$ and $\alpha = 1.25°$. The important aspects of the solution are the location of the upper and lower surface shocks as well as the shape of the sonic lines. The initial grid consists of four gridblocks of 69 by 69 points which make up a C-grid as shown in Figure 1. The far field boundaries are set at 30 chord

lengths away in the x-direction and 40 chord lengths away in the y-direction. Figure 2 shows a closer view of the initial grid. Notice that the cell center locations as well as the block boundaries are shown in these figures. Figure 3 shows the final mesh after dynamic grid adaption. Figures 4 and 5 show the numerical solutions of the pressure and mach number. Notice in Figure 3 that the grid is clustered at the shock and trailing gradient and is also aligned to the gradients in presure and mach number. The sonic line is emphasized on Figure 5. The interpolated shock locations on the upper and lower surfaces of the airfoil are at 48.1% and 35.0% chord, respectively. The sonic line extends .88 chords from the center line. The pressure coefficient along the surface of the airfoil is shown in Figure 6 and is compared to computational results obtained from Reference 16. Notice that the adaptor allows for finer resolution of the shock on the upper surface. The lift coefficient is .341.

Case 2: AGARD 03

AGARD 03 is also based on the 0012 airfoil. The flow conditions are $M_\infty = 0.95$ and $\alpha = 0.°$. This case has a "fish-tail" like shock structure in which the location of the normal shock depends upon the accuracy obtained. The initial grid distribution consists of four gridblocks of 69 by 69 points which make up a C-grid similar to the grid of Case 1. The outer boundaries have a radius of 150 chords with the upstream boundary located 200 chords upstream of the airfoil. Figures 7 and 9 show views of the initial mesh and the final mesh after dynamic grid adaption. Figures 8 and 10 show closer views of the initial and adapted mesh. Figures 11 and 12 show the numerical solutions of the pressure and mach number. Notice in Figure 10 not only the grid clustering at the shocks but also the overall grid alignment to the gradients in presure and mach number. The sonic line is emphasized in Figure 13. The sonic line extends out to nearly 21 chord lengths away from the airfoil.

The distance from the trailing edge to the first subsonic Mach number value downstream is plotted as a function of nondimensional time in Figure 14. Since interpolation is not used to gain a more accurate location of the normal shock, there are non-physical jumps in the shock location in Figure 14. Notice that the normal shock location begins to converge to a location near 3.5 chords from the trailing edge and then abruptly changes speed. This change in speed is attributed to the influence of a wave which results from the weak reflection from the outer boundary of the outgoing solution startup transient. In a nonadaptive scheme, these waves are observed infrequently due to the dissipation caused by the increased mesh spacing toward the outer boundary. However, in the adaptive solution, the startup waves are sufficiently resolved such that they are still discernable as they reach the outer boundary. Although a characteristic-like outer boundary condition is used, cancellation (*i.e.* pass-through) of the wave is not perfect and a small-amplitude expansion wave is reflected toward the interior. This wave results in a small increase in effective freestream Mach number as seen by the airfoil. This increased Mach number causes the normal shock to move further aft and the upper and lower sonic regions to increase. The solution does not converge to the correct value before there is any influence from the reflected waves. This phenomenon was most prevalent when local time stepping was used. A compromise maximum time step of five times the minimum allowable was found to reduce the effect. However, as the limit is reduced (toward having a global time step), the CFL limit due to the explicit time dependent scheme results in increased compution time.

Case 6: Shock Reflection from a Double Wedge (Time Dependent)

The time dependent workshop test case No. 6 consists of an inviscid planar normal shock wave which translates over a double angled ramp. The geometry is defined by a 20° section of the ramp beginning at the Cartesian coordinate $(2, 0)$ and a 55° section passing through the coordinate $(7, 4)$.

The translating ($M = 2.16$) shock is initially perpendicular to the y-axis and positioned at $x = 1$, which was taken as the reference length ($L = 1$) for non-dimensionalization. The non-dimensional quiescent pressure and density are both set to a value of 1. The initial mesh was the same for both the static and adaptive mesh computations. This mesh consisted of 468 cells in the streamwise direction and 162 cells in the normal direction and was constructed such that all of the cells where rhomboids. The mesh was redistributed every time step for the adaptive computations. The memory requirements were small enough for this problem that both the static and adaptive computations could be run interactively on the CRAY Y-MP at NASA LaRC.

Figures 15 and 16 compare static mesh and adaptive mesh solutions of density and pressure contours, respectively. The solutions are shown for $t/t_c = 3.5$. Here t is the physical time and $t_c = U/L$, where U is the velocity of the compression region behind the normal shock. The adaptive mesh solution shows improved resolution of shock waves. Also, flow within the double compression regions behind the reflected shocks are smoother and details are more resolved in the adapted mesh computations. In addition, a region of large amplitude high frequency oscillations behind the normal shock near the triple point which occurred in the static mesh computations has been greatly reduced in the adapted mesh solution.

Figure 17 shows a series of adapted meshes for subsequent times, for $2.0 \leq t/t_c \leq 4.5$ in increments of $t/t_c = .5$. The meshes have been coarsened for clarity of display by removing every other mesh line. Also, the upper portion of the mesh has been deleted to save space. This sequence illustrates how the adaptive algorithm continuously adapts the mesh to various flow features by clustering the mesh in regions of high flow gradients and increasing alignment of the mesh with discontinuities.

Figure 18 compares the relative errors of the computed normal shock position for the static and adaptive mesh calculations. The relative position error is defined as $|X - X_E|/|X_E|)$, where X_E is the theoretical normal shock position. The position of the shock, X, was chosen to be the interpolated x/L location in the pressure transition corresponding to the mid-point of the theoretical pre- and post-shock conditions. A linear behavior of the relative position error indicates that the computed shock is moving at a constant velocity, and the magnitude of the slope indicates how close the computed shock velocity agrees with the theoretical shock velocity. When the slope is zero the two velocities are equal.

The static computation predicts a more consistent relative error throughout the computational time history because of the uniformity of the static mesh. Abrupt changes in the wave form of the relative error for the static case are due to changes in grid spacing in the direction of the shock movement. Because the initial grid was constructed of rhomboids, the spacing between the vertical grid lines is proportional to the cosine of the wedge angle. As a result, a small but sudden change in the spacing of the vertical grid lines occurs at each angle change. The reductions in mesh spacing and the required reductions in time step by the CFL condition result in a more accurate shock velocity. This fact is shown by the successive reductions in the magnitude of the slope of the static case curve.

The relative position error of the shock computed by the adaptive mesh case is larger then the static mesh case. However, by comparing the slopes of the curves, it is seen that the trends in velocity are quite similar. Therefore, the difference between the two curves is primarily due to the difference in the magnitude of the initial jump in position error. This jump is believed to be caused by the emergence of a spike in the dependent flow variables on the high-pressure side of the shock just after start-up from initial conditions. This spike apparently alters the conditions behind the shock to the point of affecting its initial propagation velocity. This spike occurs in both the static and adaptive computations. The static mesh case has a smaller initial error in velocity, and thus position, because the spike is poorly resolved and damps quickly. However, the adaptive mesh, with its ability to resolve high gradient features, resolves the spike which more severely alters the initial propagation velocity. Eventually, the spike is damped, and the shock velocity of the adapted case becomes more

in agreement with the theoretical velocity. The jumps in position error occurring in the static mesh case, when the vertical mesh line spacing is suddenly reduced, can also be explained by the above hypothesis. The static and adaptive computations show similar trends in the shock velocity history, and both yield shock positions that are accurate to within one and a half percent of the theoretical value indicating that time accuracy is maintained on the adapted mesh.

Figures 20 and 21 show exponential contour plots of $\|\mu\|_2$ for solutions of the double Mach reflection workshop test case at $t/t_c = 3.5$ obtained on a static mesh and on an adapted mesh, respectively. The mesh was redistributed every time step during the adapted mesh computation. Light contour lines correspond to well resolved flow and dark lines to poorly resolved flow as indicated by $\|\mu\|_2$. By comparing Figures 20 and 21 we can see that the deformed mesh of the adaptive computations has not introduced any significant increase in $\|\mu\|_2$. In fact, deforming the mesh to the solution field has reduced $\|\mu\|_2$ in many regions of the flow, which of course is what we try to achieve by mesh redistribution adaptation. The adaptive mesh solution has reduced wave like structures which appear in the double compression regions of the static mesh computations. Because of this, the adaptive mesh case is able to more cleanly capture flow details in the double compression region over the 55° wedge section (refer also to Figures 15 and 16). Also, the regions of high $\|\mu\|_2$ about the shocks has been reduced in extent in the adaptive mesh case. In both cases the largest values of $\|\mu\|_2$ are associated with the fastest moving discontinuities. As the static mesh solution progressed to the present time, a region of large amplitude high frequency oscillations began to emerge behind the moving normal shock near the triple point. The presence of these oscillations is clearly visible by the large concentration of high values of $\|\mu\|_2$ in this region. The adaptive mesh computation did not produce this anomaly. The only noticeable increase in $\|\mu\|_2$ of the adaptive mesh case over the static mesh case is in the region behind the normal shock and above the reflected shock. However, this increase is on the order of 1E-6. The ability of $\|\mu\|_2$ to detect discontinuites as well as low amplitude waves has encouraged ongoing research to investigate its use as an adaptive criteria.

CONCLUSIONS

The dynamic solution adaptive mesh algorithm, DSAGA3D, is successfully extended to 2-D multi-block structured grids, including adaptation of block boundaries. The multi-block solver and grid adaptor has been developed to be general, requiring only description of the grid blocks, simple splicing and boundary condition information, and criteria for adaption. All of this can be done through straightforward input without code editing.

Solutions obtained for workshop cases 1 and 2 demonstrated that the multi-block adaptive algorithm resolves the important features of the flows very well, including alignment of the mesh such that shock definition is enhanced beyond that expected from mesh spacing alone. Solutions obtained for workshop case 6 indicated that the mesh adaption procedure maintains temporal accuracy.

A mesh quality assessment criteria is proposed that provides a measure of how well the mesh resolves and aligns with the solution. This criteria has been applied to even distributed and dynamically adapted grid solutions of workshop case 6. Analysis of the results reveals the criteria to show promise for determining the resolution quality of solution features.

REFERENCES

1. Benson, R.A.,"A Dynamic Solution-Adaptive Grid Algorithm in Two and Three Dimensions," Masters' Thesis, North Carloina State University, December 1989.

2. Benson, R.A. and M^{c}Rae, D.S., "A Three-Dimensional Dynamic Solution-Adaptive Mesh Algorithm," AIAA Paper 90-1566, Seattle WA, June 1990.

3. Benson, R.A. and McRae, D.S., "A Solution-Adaptive Mesh Algorithm for Dynamic/Static Refinement of Two and Three Dimensional Grids," *Proceedings of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Barcelona Spain, June 1991.

4. Benson, R.A. and McRae, D.S., "Time-Accurate Simulations of Unsteady Flows with a Dynamic Solution-Adaptive Algorithm ," *Proceedings of the Fourth International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Swansea UK, April 1994.

5. Ingram, C.L. and McRae, D.S., "Time-Accurate Simulation of a Self-Excited Oscillatory Supersonic External Flow with a Multi-Block Solution-Adaptive Mesh Algorithm ," *Proceedings of the 11th AIAA Computational Fluid Dynamics Conference* , Orlando Florida, July 1993.

6. Ingram, C.L.,"Extension Of A Dynamic Solution-Adaptive Mesh Algorithm And Solver To General Structured Multi-Block 2-D Configurations," Masters' Thesis, North Carloina State University, August 1995.

7. Tamura, Y. and Fujii, K., "Conservation Law for Moving and Transformed Grids," AIAA Conference Paper 93-3365-CP, Orlando Fla, July 1993.

8. Hirsch, C., Numerical Computation of Internal and External Flows, Volume 1, Wiley and Sons, 1990, pp. 445-449.

9. Liou, M.-S. and Steffen, Jr., C.J., "A New Flux Splitting Scheme," NASA TM-104404, May 1991.

10. van Albada, G.D., van Leer, B., and Roberts JR., W.W., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," Astronomy and Astrophysics, **108**, pp 76-84, 1982.

11. Scott, James N. and Niu, Yang-Yao (1993) "Comparison of Limiters in Flux-Split Algorithms for Euler Equations", AIAA paper 93-0068.

12. Eisman, P.R., Adaptive Grid Generation by Mean Value Relaxation, in: K.N. Ghia and U. Ghia, eds., Advances in Grid Generation, ASME-FED-5 1983 (ASME, New York, 1983) 29-34; also: ASME J. Fluids Engrg. 107 (1985) 477-483.

13. Connett, W.C., Agarwal, R.K., and Schwartz, A. L., "An Adaptive Grid-Generation Scheme for Flowfield Calculations," AIAA Paper 87-0199, 1987.

14. Connett, W.C., Agarwal, R.K., and Schwartz, A. L., "An Adaptive Grid-Algorithm for the Euler/Navier-Stokes Equations," AIAA Paper 88-0519, 1988.

15. Dadone, A. and Grossman, B., "Surface Boundary Conditions for the Numerical Solution of the Euler Equations," *Proceedings of the 11th AIAA Computational Fluid Dynamics Conference* , Orlando Florida, July 1993.

16. Warren, Gary Patrick, "Adaptive Grid Embedding for the Two- Dimensional Flux-Split Euler Equations," Masters' Thesis, Mississippi State University, May 1990.

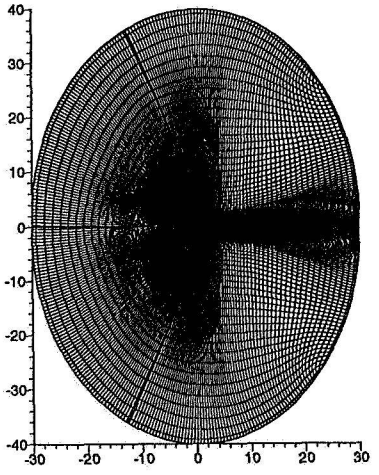


Figure 1: The Initial C-Grid: Entire Grid

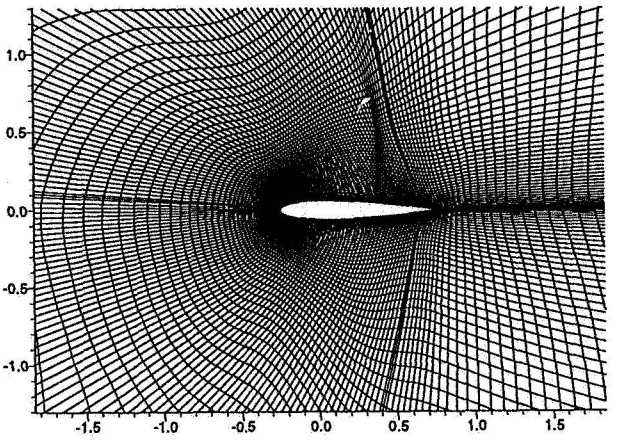


Figure 3: The Adapted C-Grid

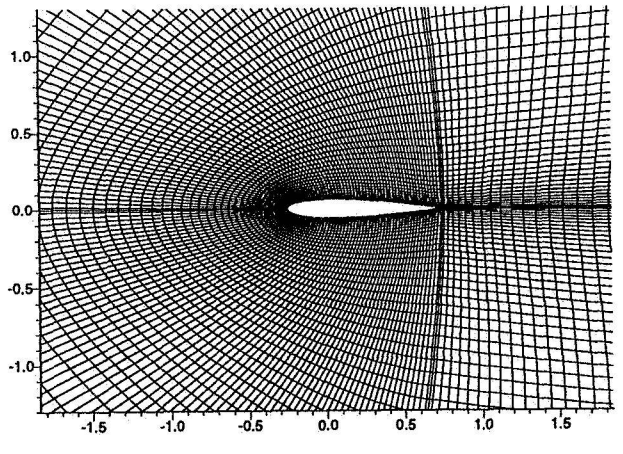


Figure 2: The Initial C-Grid: A Close View

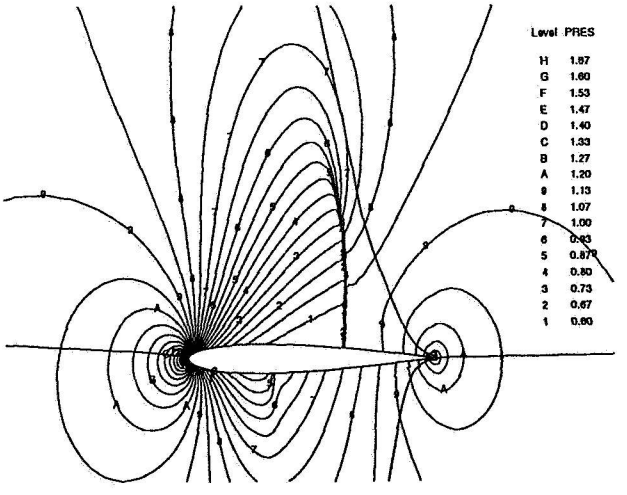


Figure 4: Pressure Contours

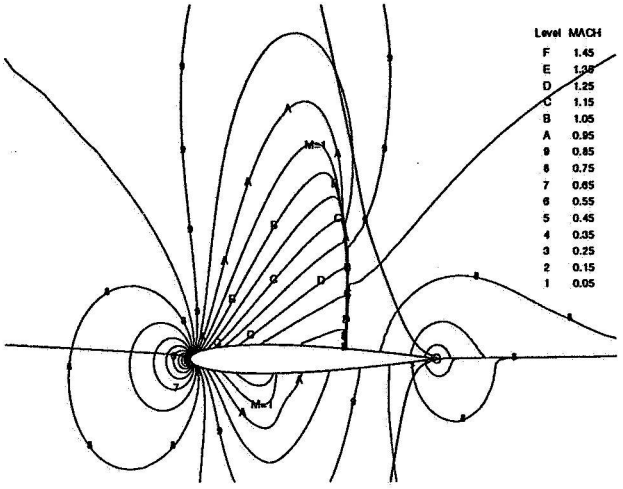


Figure 5: Mach Number Contours

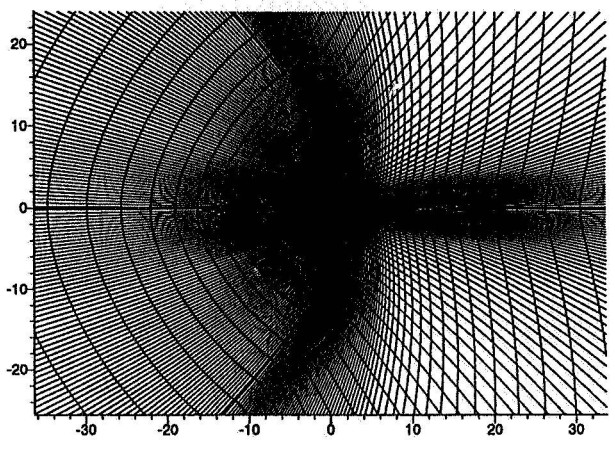


Figure 7: The Initial C-Grid

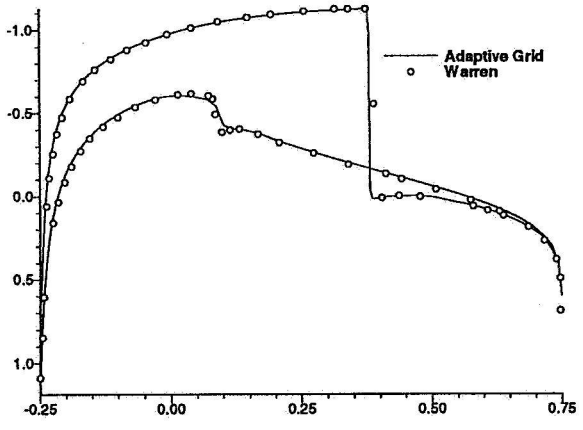


Figure 6: Pressure Coefficient Comparison

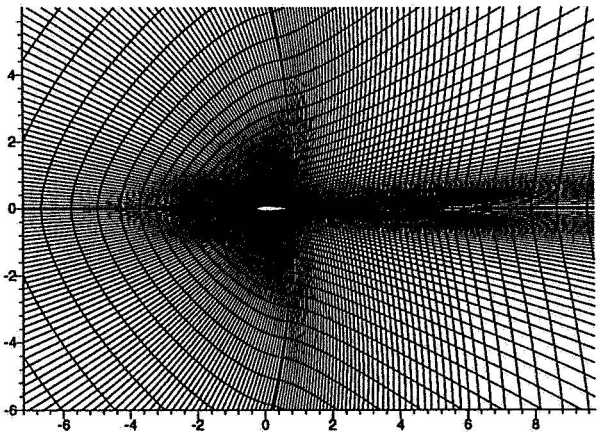


Figure 8: The Initial C-Grid: A Close View

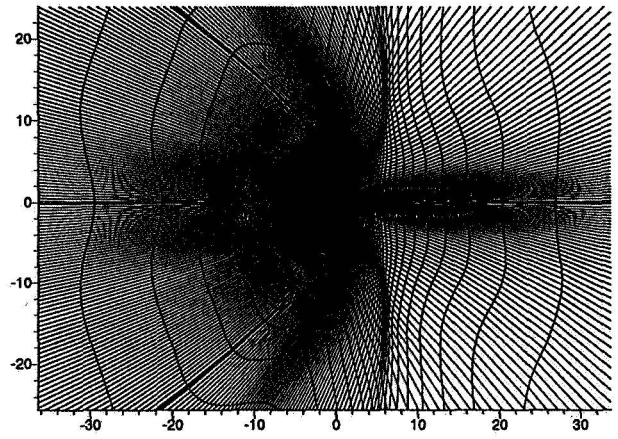


Figure 9: The Adapted C-Grid

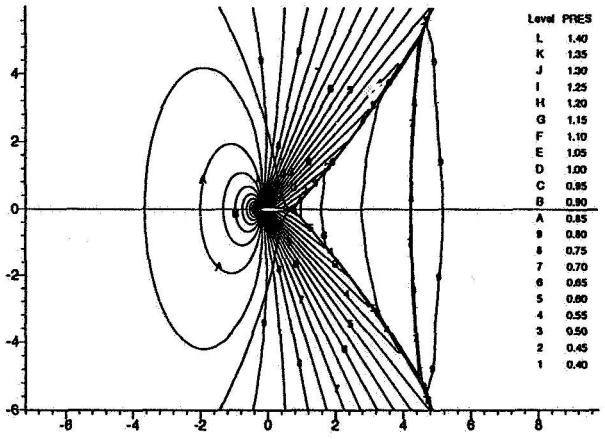


Figure 11: Pressure Contours

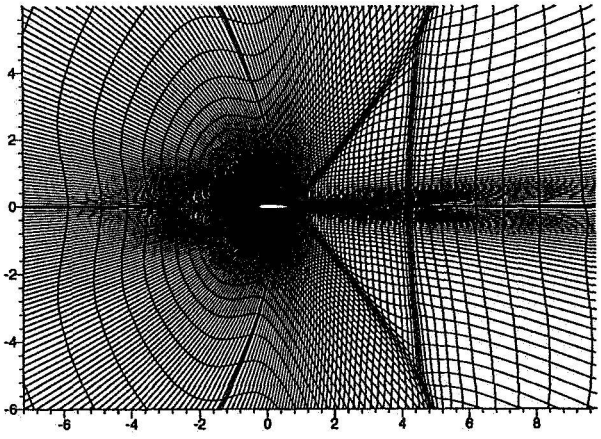


Figure 10: The Adapted C-Grid: A Close View

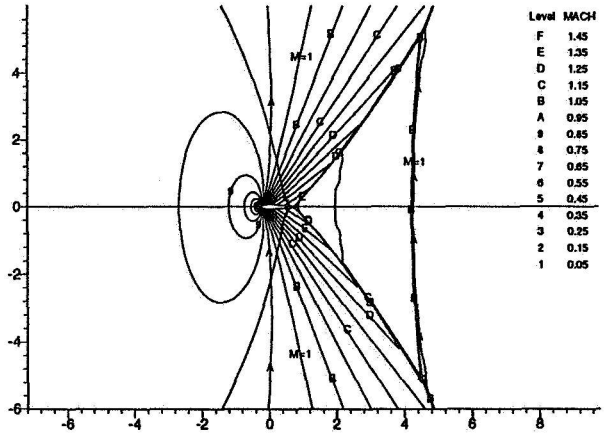


Figure 12: Mach Number Contours

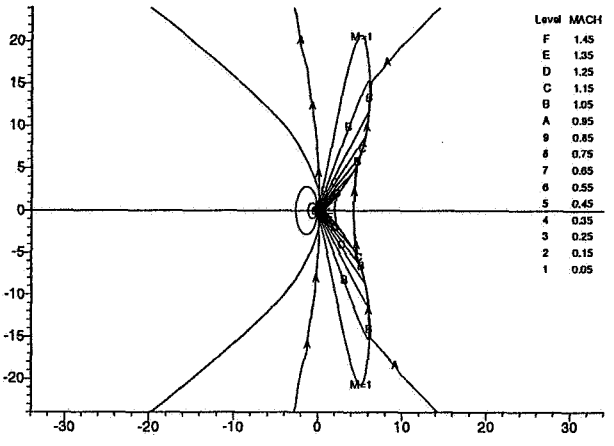


Figure 13: Mach Number Contours Featuring The Sonic Line

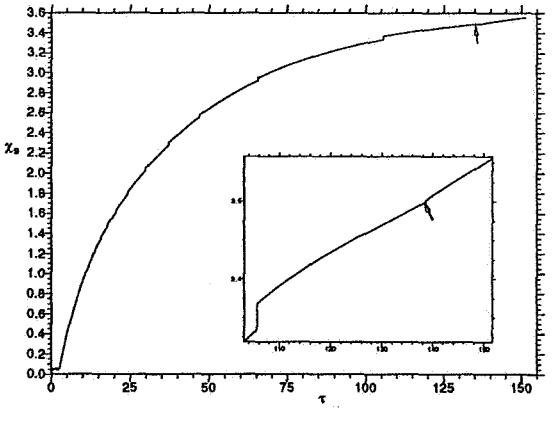


Figure 14: Normal Shock Distance From Trailing Edge

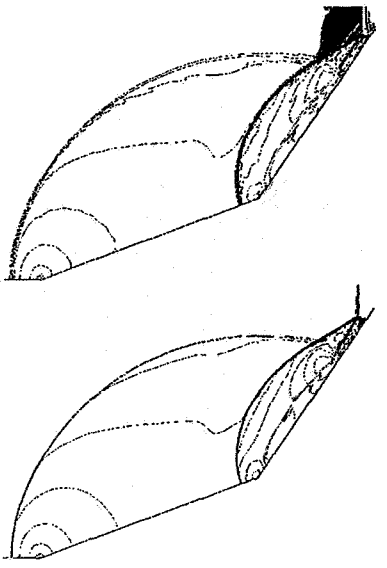


Figure 15: Density Contour Line ($\Delta\rho = 0.2$) For A Static Mesh Computation (Top) And An Adapted Mesh Computation (Bottom) For Case 6 At $t/t_c = 3.5$

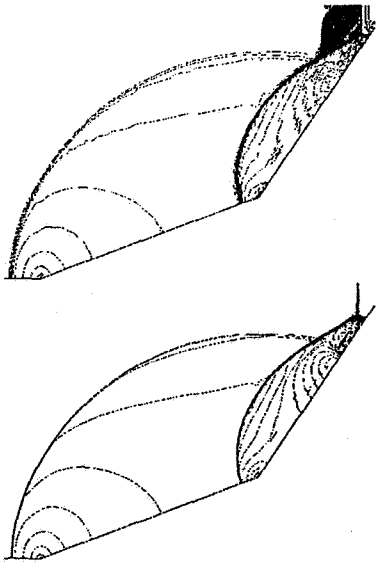


Figure 16: Pressure Contour Line ($\Delta P = 0.5$) For A Static Mesh Computation (Top) And An Adapted Mesh Computation (Bottom) For Case 6 At $t/t_c = 3.5$

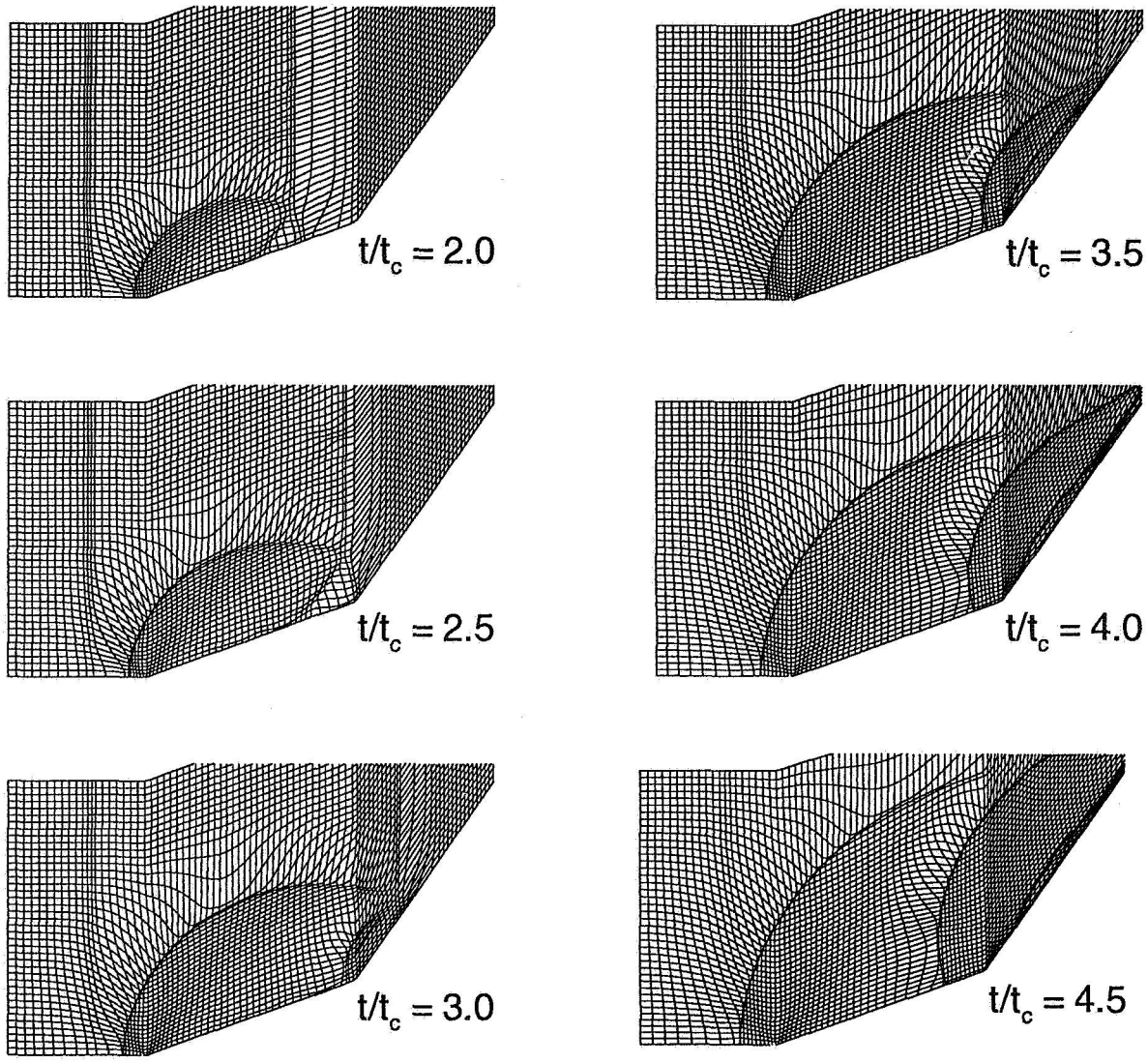


Figure 17: Adapted Mesh Series For Case 6

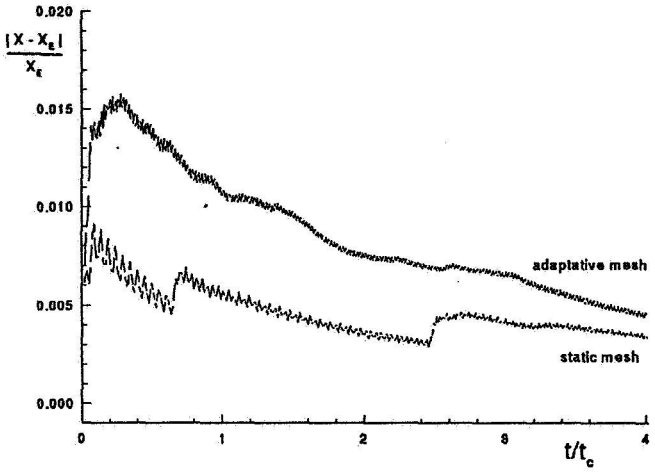


Figure 18: Shock Relative Position Error vs. Nondimensional Time For Case 6

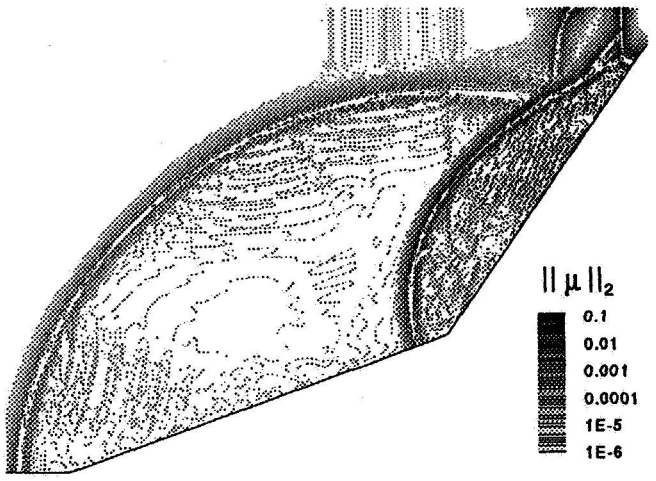
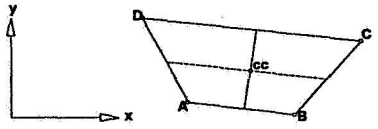


Figure 20: Solution Dependent Mesh Quality Measure (SDMQM) For The Static Mesh Computation of Case 6 at $t/t_c = 3.5$



$$\int_\Omega \nabla \cdot F dV = (f_x + g_y) V$$
$$+\frac{Y_1}{32}\left[\left(X_3^2 - X_2^2\right) f_{xx} \left(Y_3^2 - Y_2^2\right) f_{yy}\right]$$
$$+\frac{X_1}{32}\left[\left(X_3^2 - X_2^2\right) g_{xx} \left(Y_3^2 - Y_2^2\right) g_{yy}\right]$$
$$+\frac{1}{16}\left[\left(X_3 Y_3 - X_2 Y_2\right)\left(Y_1 f_{xy} - X_1 g_{xy}\right)\right]$$
$$+\frac{f_{xxx}}{384}\left(Y_2 X_3^3 - Y_3 X_2^3\right) + \frac{f_{yyy}}{384}\left(Y_2 Y_3^3 - Y_3 Y_2^3\right)$$
$$+\frac{g_{xxx}}{384}\left(X_3 X_2^3 - X_2 X_3^3\right) + \frac{g_{yyy}}{384}\left(X_3 Y_2^3 - X_2 Y_3^3\right)$$
$$+\frac{f_{xyy}}{128}\left(Y_2 X_3 Y_3^2 - Y_3 X_2 Y_2^2\right) + \frac{g_{xxy}}{128}\left(Y_2 X_3 X_2^2 - Y_3 X_2 X_3^2\right)$$
$$+\frac{f_{xxy}}{128}\left[Y_2 Y_3 \left(X_3^2 - Y_3^2\right)\right] + \frac{g_{xyy}}{128}\left[X_2 X_3 \left(Y_2^2 - X_2^2\right)\right] + HOT$$

$$X_1 = x_A - x_B + x_C - x_D \qquad Y_1 = y_A - y_B + y_C - y_D$$
$$X_2 = x_A + x_B - x_C - x_D \qquad Y_2 = y_A + y_B - y_C - y_D$$
$$X_3 = x_A - x_B - x_C + x_D \qquad Y_3 = y_A - y_B - y_C + y_D$$

Figure 19: Resolution Error For 2-D Quadrilateral Cell

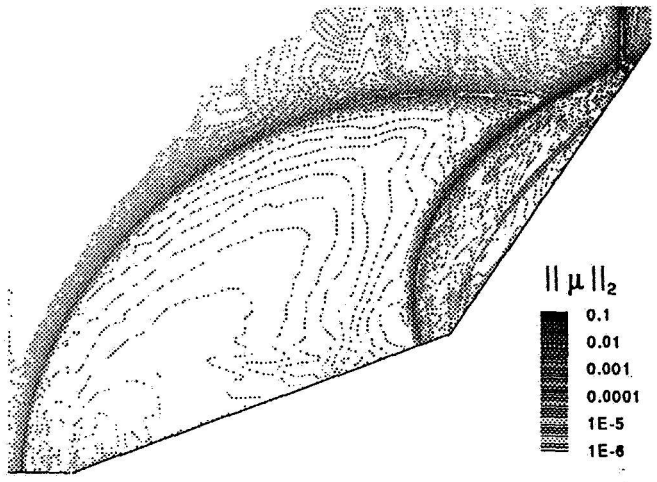


Figure 21: Solution Dependent Mesh Quality Measure (SDMQM) For The Adapted Mesh Computation of Case 6 at $t/t_c = 3.5$

Methods for Prismatic/Tetrahedral Grid Generation and Adaptation

Y. Kallinderis *

Dept. of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, TX 78712

Abstract

The present work involves generation of hybrid prismatic/tetrahedral grids for complex 3-D geometries including multi-body domains. The prisms cover the region close to each body's surface, while tetrahedra are created elsewhere [8]. Two developments are presented for hybrid grid generation around complex 3-D geometries. The first is a new octree/advancing front type of method for generation of the tetrahedra of the hybrid mesh. The main feature of the present advancing front tetrahedra generator that is different from previous such methods is that it does not require the creation of a background mesh by the user for the determination of the grid-spacing and stretching parameters. These are determined via an automatically generated octree. The second development is a method for treating the narrow gaps in between different bodies in a multiply-connected domain. This method is applied to a two-element wing case. A High Speed Civil Transport (HSCT) type of aircraft geometry is considered. The generated hybrid grid required only 170 K tetrahedra instead of an estimated two million had a tetrahedral mesh been used in the prisms region as well. A solution adaptive scheme for viscous computations on hybrid grids is also presented [2]. A hybrid grid adaptation scheme that employs both h-refinement and redistribution strategies is developed to provide optimum meshes for viscous flow computations. Grid refinement is a dual adaptation scheme that couples 3-D, isotropic division of tetrahedra and 2-D, directional division of prisms.

An Octree/Advancing Front Method for Tetrahedra Generation

A new method is presented for generating the tetrahedra of the hybrid grid. Advancing front type of methods require specification by the user of the distribution of three parameters over the entire domain to be gridded. These field functions are: (i) the node spacing, (ii) the grid stretching,

*Associate Professor

and (iii) the direction of the stretching. In the present work these parameters do not need to be specified. The distribution of grid size, stretching, and direction of stretching is automatically determined via an octree. There is no need for a special background mesh which has been the backbone of previous advancing front generators.

The tetrahedra that are generated should progressively become larger as the front advances away from the original surface. Their size, the rate of increase of their size, as well as the direction of the increase are given from an octree consisting of cubes which is generated automatically via a *Divide-and-Conquer* method. This process generates octants that are progressively larger with distance away from the body. Their size will be the characteristic size of the tetrahedra that will be generated in their vicinity.

Generation starts from the outermost surface of the layer of prisms surrounding the body. The triangles of this surface form the initial front. Then, a list of points is created that consists of a new node, as well as of "nearby" existing points of the front. One of these points is chosen to connect to the vertices of the face. Following choice of the point to connect to, a new tetrahedron is formed. The list of the faces, edges, and points of the front is updated by adding and/or removing elements. The algorithm followed in the present work is the one presented in [3, 4]. The method requires a data structure which allows for efficient addition/removal of faces, edges and points, as well as for fast identification of faces and edges that intersect a certain region. The alternating digital tree (ADT) algorithm is employed for these tasks [5].

Figure 1 illustrates the symmetry plane of the HSCT geometry. The quadrilaterals (dark lines) correspond to the faces of the octants on this plane, while the triangles (light lines) correspond to the faces of the tetrahedra. It is observed that the size of the tetrahedra, as well the stretching of the mesh and the direction of stretching is guided quite accurately by the octree.

Simplicity and no user intervention are main advantages of the octree. The usual trial-and-error procedures for constructing the field functions that give the local size of the tetrahedra, the stretching of the mesh, and the direction of the stretching (background mesh) for previous advancing front generators are avoided in the present method. The octree is generated once and remains the same throughout the generation process. Details of the octree are presented in [7, 8].

Hybrid Grid Generation for Multi-Body Domains

The developed hybrid grid generation method is flexible and general in order to treat domains that contain multiple bodies. A prismatic layer is created around each one of the bodies, while the regions in between these meshes are filled with tetrahedra. Any location and orientation of these bodies is allowed. This is accomplished via a special method for treatment of narrow gaps that frequently form in multiply-connected domains, such as multi-element wings. The key feature of the method is the fact that the prismatic grid around each of the bodies is generated independently of all the other bodies. As a result, such generation is as simple as the generation of prisms for a domain containing a single body. However, overlapping meshes are avoided here by employing a special technique that redistributes the prisms nodes along their corresponding marching lines after the initial generation. This redistribution occurs in the vicinity of the regions of overlapping prismatic meshes and results in formation of gaps in between the previously overlapping prisms layers. Then a tetrahedral grid is generated in order to fill in those gaps. It should be emphasized

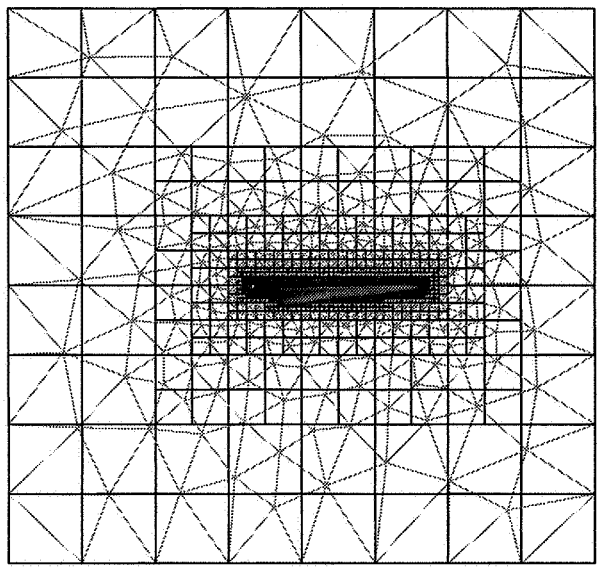


Figure 1: Effect of the octree on growth of the tetrahedra. View of the octants (quadrilateral faces), as well as of the tetrahedra (triangular faces) on the symmetry plane. Growth of the tetrahedra away from the outermost prisms surface follows growth of the octree quite faithfully.

that the structure of the prismatic grid is not destroyed. Further details of the method are given in [8].

In order to illustrate validity of the previous procedure, the case of a two-element wing with variable size of the gap between the main wing and its flap is considered. Stage one involves generation of the two separate prismatic meshes that cover each one of the two bodies. Generation is quite simple due to the fact that each layer of prisms is grown independently of the other layer. The two grids overlap locally as shown in Figure 2(a). In the second stage, the thickness of the prisms layers is reduced locally and the overlap no longer occurs as shown in Figure 2(b). Comparing the grids of Figure 2, it is observed that the receding occurs over a larger region which results in a smooth variation of the local thicknesses of both meshes. The final stage involves generation of the tetrahedral mesh that covers all areas in between the prisms. Figure 3 shows the final hybrid (prismatic/tetrahedral) grid on the plane of symmetry. The quadrilaterals are the signature of the prisms on that plane, while the triangles correspond to faces of the tetrahedral mesh.

Hybrid Grid for the HSCT

A High Speed Civil Transport (HSCT)-type of aircraft geometry was chosen as the test case for the developed grid generator. Figure 4 shows the triangulation of the initial surface. The mesh

consists of 4412 triangles and 2275 nodes. A symmetry plane is considered that divides the body. Thus, hybrid grid is generated for half of the space.

The time required to generate the prismatic grid around the HSCT was 90 seconds for 40 layers of prisms on an IBM 390 workstation. Generation of approximately 170,000 tetrahedra took about 67 minutes on the same station. It should be emphasized that employment of a hybrid grid for the HSCT geometry required only 170 K tetrahedra instead of an estimated two million had a tetrahedral mesh been used in the prisms region, as well.

Figure 5 illustrates the hybrid mesh on two different planes that are perpendicular to each other. The first plane is the symmetry and it is indicated by the darker lines, while the second is intersecting the fuselage at a location upstream of the wing and it is shown via light lines. It should be noted that the irregularity of the lines observed on the second plane are due to the fact that the grid it intersects is not planar as it is on the symmetry plane.

Combined Refinement/Redistribution for the Hybrid Grid

A dynamic grid adaptation algorithm has previously been developed for 3-D unstructured grids [6]. The algorithm is capable of simultaneously un-refining and refining appropriate regions of the flow domain. This method is extended to the present work and is coupled with prismatic grid adaptation to implement a hybrid adaptation method.

Directional Division of Prisms

The prisms are refined directionally in order to preserve the structure of the mesh along the normal-to-surface direction. The prismatic grid refinement proceeds by dividing only the *lateral* edges that lie on the wall surface and hence the wall faces. The faces are divided either into two or four subfaces. The resulting surface triangulation is replicated in each successive layer of the prismatic grid. This results in all the prisms that belong to the same stack (namely, the group of cells that originate from the same triangular face on the wall surface) getting divided alike. The prismatic grid refinement preserves the structure of the initial grid in the direction normal to the surface. The primary advantage of using such an adaptive algorithm for prisms is that the data structures needed for its implementation are essentially as simple as that for refining a 2-D triangular grid.

The directional division of the prisms does not increase resolution of flow features that are aligned in a direction that is normal to the wall surface. However, a grid redistribution algorithm can be employed in order to recluster nodes in the normal direction so as to better resolve the viscous stresses [1, 8].

The tetrahedral cells constitute the portion of grid where inviscid flow features are dominant. These features do not exhibit the directionality that is generally prevalent in viscous stresses. Hence, the tetrahedra are refined by division into eight, four, or two subcells.

Redistribution of Prisms

The redistribution algorithm increases local grid resolution by clustering existing grid points in regions of interest. A measure of the grid resolution required normal to the *no-slip* wall is the values

of $y^+ = \frac{u_\tau y}{\nu}$, with $u_\tau = \sqrt{\frac{|\tau_{wall}|}{\rho_{wall}}}$ being the wall friction velocity. A criterion based on the values of y^+ at the wall is employed to either *attract* nodes towards the wall or *repel* them away from the surface so that a specific value of y^+ is attained at all the wall nodes. This procedure in essence determines a new value for the spacing δ_{wall} of the first node off the wall at all locations on the wall surface. The nodes in the prismatic region are then reclustered along the marching lines emanating from the corresponding wall node, in accordance with the new value of δ_{wall}. Details are presented in [2].

Application of the Adaptation Method

The test case of flow past a sphere at a free stream Mach number of $M_\infty = 1.4$ and a Reynolds number of $Re = 1000$ (based on the radius of sphere) is considered. The flow is characterized by both inviscid and viscous flow features such as shock waves and boundary layer separation. Details are given in [2].

The hybrid grid adaptation algorithm developed in the present work is now implemented to obtain numerical solution for the same flow situation discussed above. A coarse hybrid grid comprising $\sim$ *1400* wall boundary nodes and $\sim$ *100K* tetrahedra is used as the initial grid. The prismatic region is constituted by *20* layers of prisms. The locally adapted grid obtained after h-refinement based on an initial solution is shown in Figure 6. The figure shows the tessellation on the wall surface, on the symmetry plane as well as on an equatorial plane cutting through the interior of the grid, normal to the symmetry plane. It is clearly seen that embedding in the tetrahedral region is focussed near the shock location just outside of the prismatic-tetrahedral interface. The prismatic region is also directionally refined near the upstream and downstream sections of the body. This is due to the flow upstream accelerating rapidly from the upstream stagnation point and the flow downstream separating that causes flow gradients in the lateral directions that are detected by the directional adaptive algorithm. The embedded hybrid grid comprises $\sim$ *2500* wall boundary nodes and $\sim$ *275K* tetrahedra. Mach number contour lines of the solution obtained on the adapted grid are shown in Figure 7.

The effect of grid redistribution in the viscous region is next shown by selecting an initial grid that has a relatively large wall spacing δ_w and further, the prism layers are equispaced as shown in Figure 8 (a). The grid has $\sim$ *1400* wall nodes and $\sim$ *100K* tetrahedra. Based on an initial solution obtained on this grid, the redistribution algorithm is used to recompute the values of δ_w at all the wall boundary nodes, using y^+ as the detection parameter. The hybrid grid with the redistributed prismatic region is shown in Figure 8 (b). Observing the grid in the fore section, it is seen that the redistribution algorithm reclusters the grid substantially by attracting the nodes very close to the wall in order to resolve the large gradients in the normal direction. In the aft region, the boundary layer thickens substantially and separates and the algorithm is seen to push the nodes away from the wall.

Acknowledgments

This work was supported by NASA Grant NAG1-1459, the NSF Grant ASC-9357677 (NYI Program), and the Texas Advanced Technology Program (ATP) Grant # 003658-413. We would

like to thank Ms. Karen Bibb of NASA Langley for giving us access to the FELISA mesh generation code. Supercomputing time was provided by the NAS Division of the NASA Ames Research Center.

References

[1] Y. Kallinderis and S. Ward, "Prismatic Grid Generation for 3-D Complex Geometries", *Journal of the American Institute of Aeronautics and Astronautics*, Vol. 31, No. 10, pp. 1850-1856, October 1993.

[2] V. Parthasarathy, Y. Kallinderis, and K. Nakajima, "A Hybrid Adaptation Method and Directional Viscous Multigrid with Prismatic/Tetrahedral Meshes," AIAA Paper 95-0670, Reno, NV, January 1995.

[3] J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O.C. Zienkiewicz, "Finite Element Euler Computations in Three Dimensions," *AIAA 26th Aerospace Sciences Meeting*, Reno, AIAA Paper 88-0032, January 1988

[4] J. Peiro, J. Peraire, and K. Morgan, "FELISA System Reference Manual", 1994.

[5] J. Bonet, and J. Peraire, "An Alternating Digital Tree (ADT) Algorithm for Geometric Searching and Intersection Problems", Int. J. Num. Meth. Eng., Vol. 31, No. 1, 1990.

[6] Y. Kallinderis and P. Vijayan, "An Adaptive Refinement Coarsening Scheme for 3-D Unstructured Meshes", AIAA Journal, Vol 31, No.8, pp 1440-1447, Aug. 1993.

[7] S. Ward and Y. Kallinderis, "Hybrid Prismatic/Tetrahedral Grid Generation for Complex 3-D Geometries", AIAA Paper 93-0669, Reno, NV, January 1993.

[8] Y. Kallinderis, A. Khawaja, and H. McMorris "Hybrid Prismatic / Tetrahedral Grid Generation for Complex Geometries", AIAA Paper 95-0211, Reno, NV, January 1995.

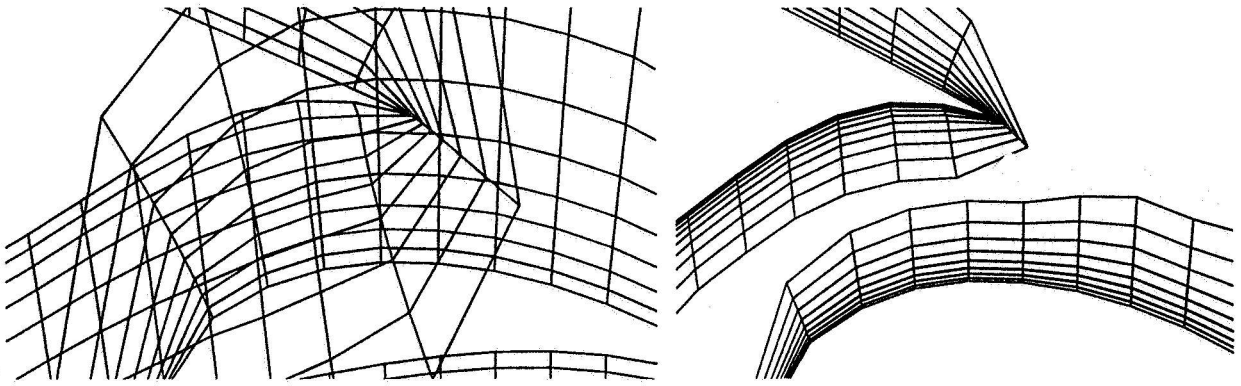


Figure 2: (a) Prismatic grids grow around each body independently of one another (b) Mutual receding of the two prismatic grids removes prior overlapping (view on the symmetry plane).

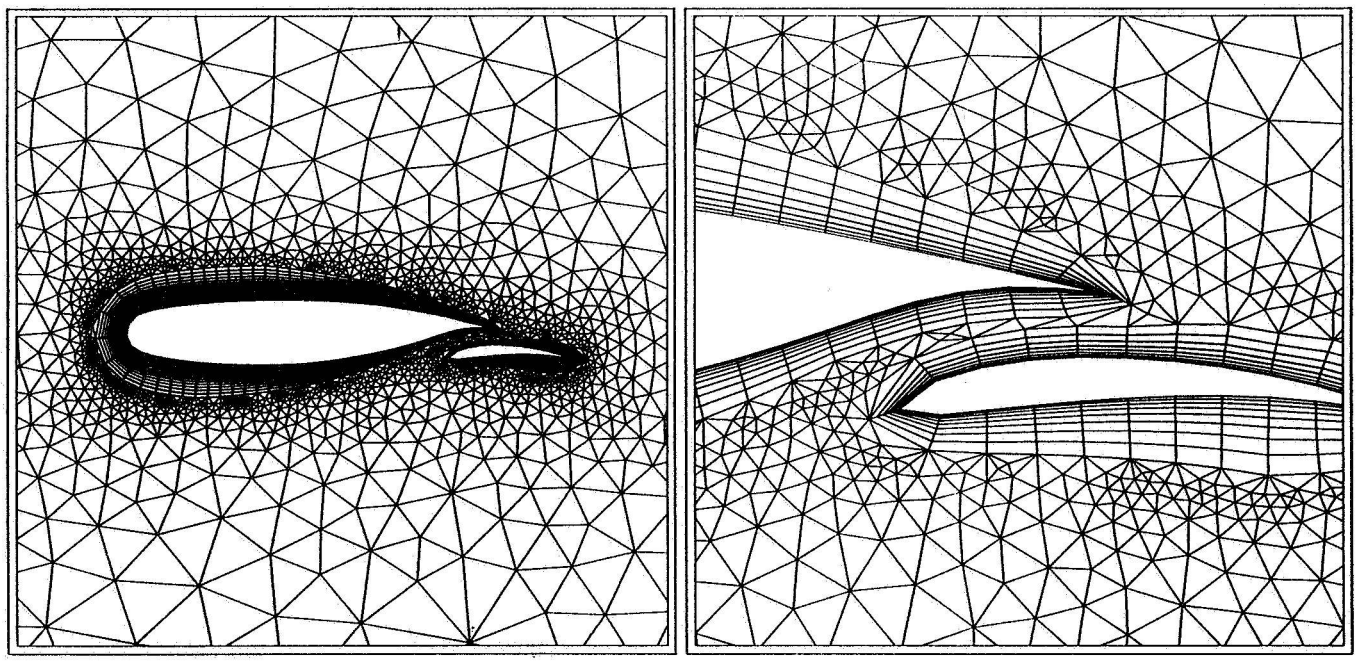


Figure 3: (a) Tetrahedral grid fills the areas in between the two prismatic meshes (b) Enlarged view of the gap region between the two bodies (view on the symmetry plane).

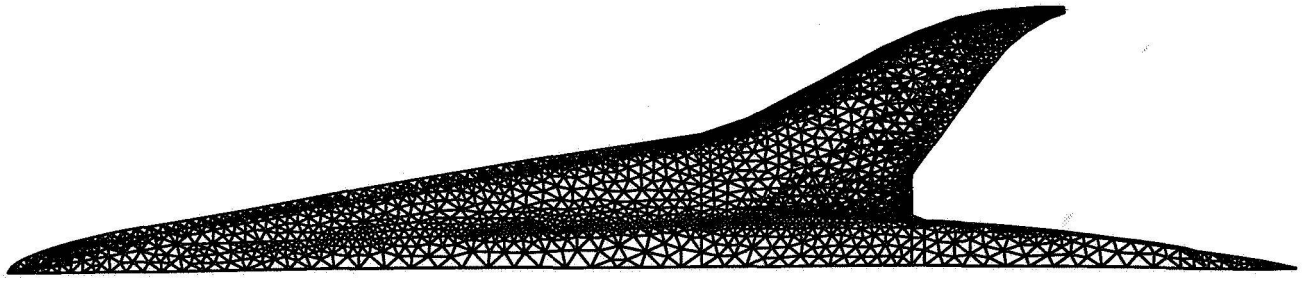


Figure 4: Triangulation of the HSCT surface (4412 triangles, 2275 nodes)

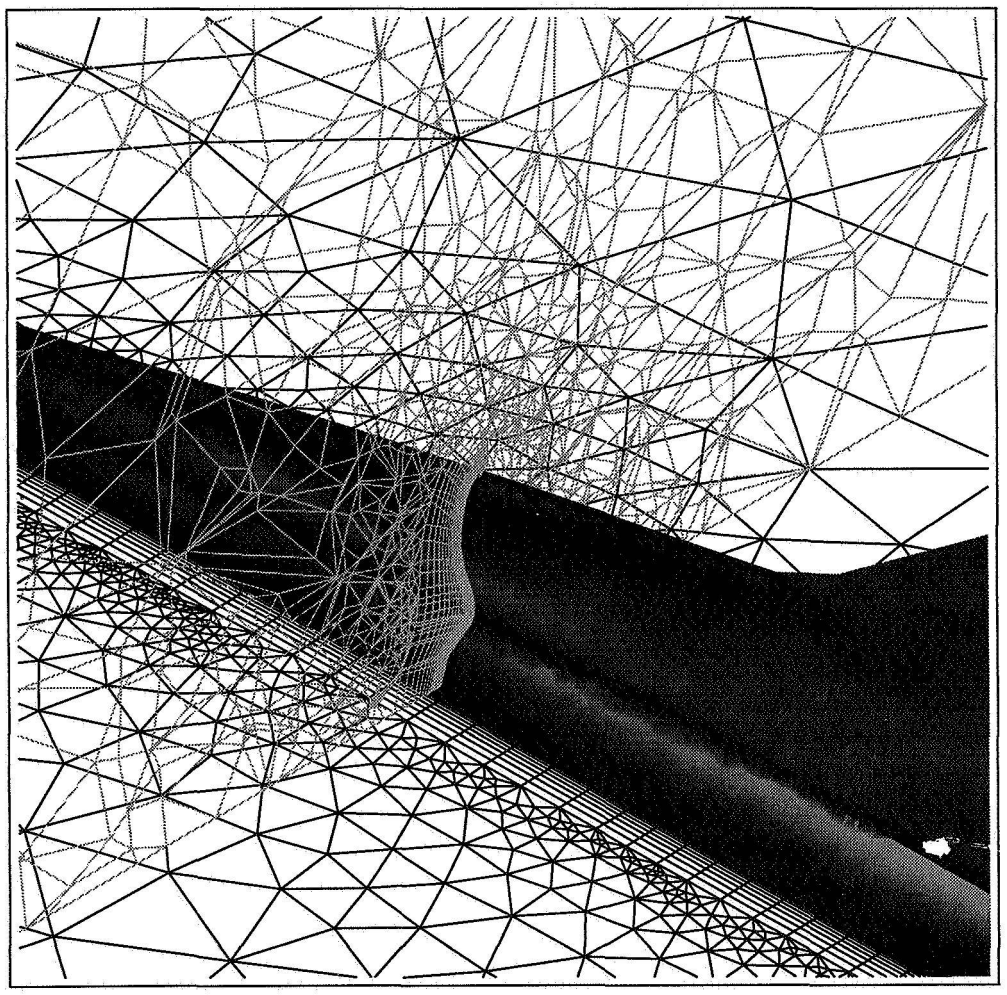


Figure 5: View of the hybrid mesh around the HSCT on two different planes that are perpendicular to each other. The first plane is the symmetry (dark lines), while the second is intersecting the fuselage at a location upstream of the wing (light lines).

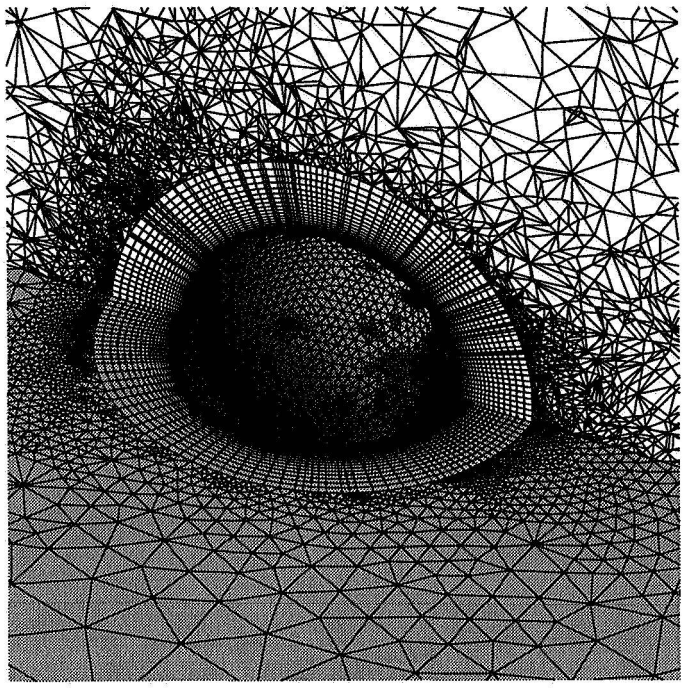


Figure 6: An isometric view of the tessellation on the wall surface, symmetry plane and an interior equatorial plane. Hybrid grid embedded isotropically in the tetrahedral region and directionally in the prismatic region.

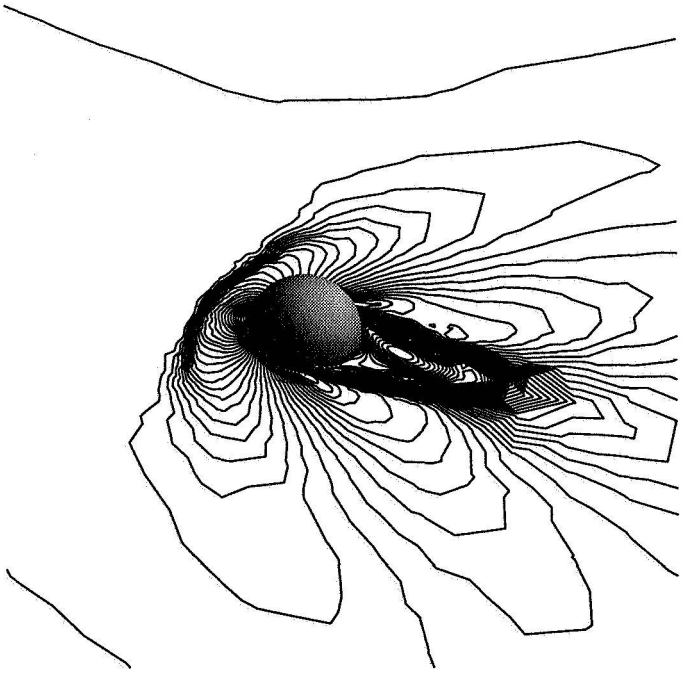


Figure 7: Mach number contour lines of the solution on the symmetry plane. Solution obtained using an embedded hybrid grid ($M_{min} = 0., M_{max} = 2., \Delta M = 0.05$).

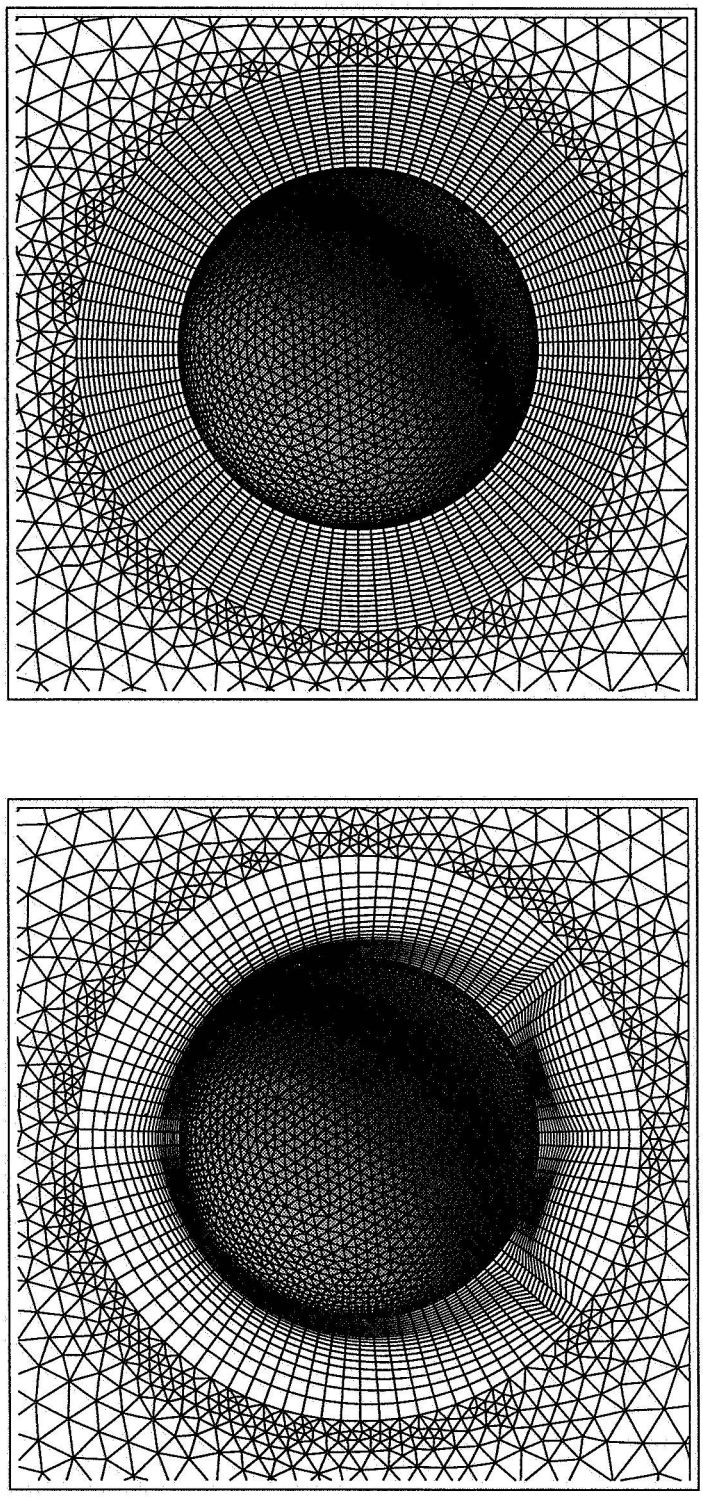


Figure 8: Tessellation on the symmetry plane showing the clustering of grid points in the prismatic region for (a) an initial grid with equispaced prismatic layers and (b) the grid obtained after redistributing the former.

AN ADAPTIVE REMESHING SCHEME FOR VORTEX DOMINATED FLOWS USING THREE-DIMENSIONAL UNSTRUCTURED GRIDS

Paresh Parikh
ViGYAN, Inc.
Hampton, Virginia

SUMMARY

An adaptive remeshing procedure for vortex dominated flows is described, which uses three-dimensional unstructured grids. Surface grid adaptation is achieved using the static pressure as an adaptation parameter, while entropy is used in the field to accurately identify high vorticity regions. An emphasis has been placed in making the scheme as automatic as possible so that a minimum user interaction is required between remeshing cycles. Adapted flow solutions are obtained on two sharp-edged configurations at low speed, high angle-of-attack flow conditions. The results thus obtained are compared with fine grid CFD solutions and experimental data, and conclusions are drawn as to the efficiency of the adaptive procedure.

INTRODUCTION

Mesh adaptation has been recognized as an efficient way to obtain accurate Computational Fluid Dynamic (CFD) solutions to complex flow problems. Adapting a mesh by placing more points in the region of dominating flow features (such as shocks, vortices etc.), increases the solution accuracy. The ability to automatically adapt a grid helps a user by not requiring him to have a knowledge of the location of the important flow regions in the field. As an example, for vortex dominated flows considered here, the locations of the origin of the vortices are usually known, but their trajectories in the field are not. For such a case, other than using a very fine grid everywhere, it would be extremely difficult to maintain a high grid resolution around the vortex core throughout the flow field.

Currently available mesh adaptation techniques can be classified into two broad categories: mesh movement and mesh enrichment. Mesh movement involves obtaining flow solution on an initial mesh, and at prescribed times in the solution process, moving mesh points towards the regions of important flow features. A major disadvantage of this procedure is that the accuracy of the final computation is limited by the number of points in the initial mesh. In mesh enrichment methods, starting from a relatively coarse mesh, a finer mesh is obtained either by introducing new points in the region of dominant flow features (refinement) or by generating a new mesh which is based on solution on the preceding mesh (remeshing). In the past all of these methods have been applied, to a varying degree of success, using both the structured and the unstructured meshes. A comprehensive review on this subject has appeared in a recent paper by Thompson et. al [1].

Unstructured grids are ideally suited for mesh refinement strategies, as the addition of points do not alter the data structure. However, this method has a shortcoming that the number of points increase rapidly with each refinement [2]. In the remeshing method, on the other hand, several new meshes are generated during advancement of the solution towards a steady state. A new mesh is based on the information from the previous flow solution. Since at each grid remeshing, finer meshes are produced in the regions of interest while they are coarsened in other regions, a good control is maintained on the total number of points while at the same time a good solution accuracy can be obtained.

*Work done under NASA Contract No. NAS1-19672.

In the present paper, a solution adaptive remeshing technique for vortex dominated flows is demonstrated that couples a three-dimensional, unstructured mesh generator with an inviscid flow solver. An emphasis has been placed on developing an automated procedure that requires very little user interaction between remeshing cycles. In the section that follows, the grid generator, the flow solver and the adaptation strategy are discussed. This is followed by presentation of results on two sharp-edged configurations at flow conditions characterized by vortex domination. Finally, efficiency of the adaptation procedure is evaluated.

PROCEDURE

The basic idea underlying any mesh adaptation scheme is to combine efficiently the mesh generation and the flow solver functions into a single procedure. The challenge is to integrate these functions in such a manner that the information produced by one is successively used by the other to ultimately generate a mesh which accurately resolves important flow features without using a large number of grid points. In the remeshing procedure described here, an advancing-front mesh generator (program VGRID) is integrated with an Euler equation solver (program USM3D) with an automatic procedure that produces input parameters for the mesh generator based on the solution on a previous grid. A similar study involving the same two programs was conducted for shock dominated, transonic flows, and was reported in Reference 3.

Grid Generator

The advancing-front technique used in VGRID is a powerful tool for constructing tetrahedral meshes around complex configurations [4]. In this technique, the configuration of interest is represented by several surface patches. The grid element size is controlled by grid spacing parameters specified at the nodes of a secondary grid called the 'background grid'. The background grid is made up of an uniform Cartesian grid. Associated with the Cartesian grid is a number of 'point' and 'line' sources, at which the desired grid spacing parameters are specified. The spatial variation of these parameters in the field is determined by a process similar to that of computing the diffusion of heat from a number of discrete heat sources in a conducting medium. In addition to the symmetrical propagation of grid spacings, the program has the capability to specify directional propagation, thus giving the user uniform as well as directional control of grid spacing [5]. Thus, in VGRID the location, strength and directionality of the background grid sources need to be specified by the user. In the present adaptation strategy, these grid spacing parameters are automatically determined based on an earlier solution.

Flow Solver

The flow solver used is an efficient Euler equation solver for unstructured tetrahedral cells. In USM3D, the spatial discretization is achieved by a cell centered finite-volume formulation using Roe's flux-difference splitting. A novel cell reconstruction process, which is based on an analytical formulation for computing solution gradients within tetrahedral cells, is used for higher order differencing. Solutions are advanced in time by a 3-stage Runge-Kutta time-stepping scheme with convergence accelerated to steady state by local time stepping and implicit residual smoothing. Recently the flow solver has been made more efficient by the use of a grid coloring scheme that has resulted in reduced CPU and memory requirements [6].

Adaptation Procedure

In the adaptation method, the mesh generation and the flow solver functions are integrated into an unified procedure. A typical adaptive remeshing cycle is as follows. A coarse mesh is first generated on the configuration of interest. This first mesh is obtained by the user assigned 'first guess' for the spacing parameters, and is, usually, just fine enough to capture

approximate locations for vortices in the flow. Based on a partially converged flow solution on this mesh, spacing parameters are next calculated. The spacing parameters include location and strength of 'line' sources to be used in the background grid for the generation of the next mesh. This completes one adaptation cycle. As mesh is refined in the vicinity of vortices, and simultaneously coarsened in other areas of the flow, on successive meshes a more accurate representation of the flow is obtained. In practice several adaptation cycles are required to obtain a desired accuracy. The flow solution is allowed to converge fully on the final adapted grid.

Adaptation Parameters

An important step in the present grid adaptation strategy is the calculation of the grid spacing parameters from a solution on the current mesh. These parameters are, in turn, used to generate a subsequent mesh. For the vortex dominated flows considered here, a finer mesh is needed in the vicinity of the vortices, and on the configuration surface along the vortex-induced suction peak. The task, then is to accurately find the location of the vortices in a three dimensional field. Once this is done, a finer mesh can be generated in and around these regions by placing 'line' sources in the background grid.

For vortex dominated flows, the core of the vortex can be successfully identified using entropy as the adaptation parameter [7]. Since an approximate location for the vortex origination is known (usually at the apex or kinks in the leading edge), one edge of the 'line' source is located there. The other end is found, by calculating entropy at all points in a user defined plane (search plane) downstream and tagging all the points at which the value of the adaptation parameter exceeds a specified threshold. The average of the coordinates for all such tagged points is then taken as the other end of the line source. The strength of the line source is taken as proportional to the value of entropy averaged over all the tagged points. To account for the variation in the strength and the location of the core, the line sources are placed in segments, instead of placing one long line source from apex to the end of the configuration and beyond. A similar procedure is followed for placing line sources on the surface along the vortex induced suction peak. The adaptation parameter used for the surface is the variation in 'static' pressure. The strength of the line source is inversely proportional to the value of static pressure.

The procedure described above has been automated so that the user only needs to specify approximate locations for the vortex origins and locations for the 'search planes' during the first cycle and threshold values for the adaptation parameters between remeshing cycles. All other required parameters are automatically calculated including creation of the input parameters for the next mesh. Once the location and the strength of the line sources are determined, a new mesh is generated, and a partially converged flow solution is obtained. This completes one remeshing cycle.

RESULTS

The adaptation procedure described above has been applied to two vortex dominated cases, one with a single vortex and the other with multiple vortices. The adapted solutions thus obtained are compared with 'fine' grid CFD and experimental data for assessing the efficiency and accuracy, respectively of the adapted solutions. This section presents some of the results.

Case 1: Hummel Delta Wing

The first case considered is that of a sharp-edged delta wing with an aspect ratio 1, known in the literature as 'Hummel' delta wing. The wing has a flat upper surface and a narrow triangular cross-section with a maximum thickness of 2.1 % of the chord located at 90 % root chord. The flow conditions selected are a M_∞ = 0.2 and an angle-of-attack equal to 20.5°, in order to compare with the experimental data reported in Reference 8. At these conditions, there is a single pair of vortices emanating from the leading edge near the apex of the wing.

The adaptive remeshing scheme was tested beginning with a coarse grid (GRID 1) with 61,908 tetrahedra and 12,414 points. Of these 3,713 points represented the wing surface and the rest were field points. Based on a partially converged flow solution on this grid, an adapted grid was generated using the procedure described above. The adapted grid (GRID 2) had 287,652 tetrahedra, 53,380 total points and 8,344 surface points. The coarse grid solution was interpolated on the adapted grid and converged until the residuals were reduced by a 2.5 order of magnitude.

Figure 1 shows the effect of adaptation on the surface by comparing surface triangulation on the upper surface of both the unadapted and the adapted meshes. An increased mesh resolution on the surface under the vortex induced suction peak is clearly seen for the adapted case. Figure 2 shows the effect of adaptation on the field grid density by showing a slice through the grid at a plane normal to the wing, located at 70% of root chord. The higher mesh density in the region of vortex core is evident.

The efficiency of the adapted solution is established by comparing the adapted solution with an unadapted mesh which has fine resolution everywhere in the flow field. The 'fine' mesh was generated pretending limited *a priori* knowledge of the location of the vortex. This mesh has 412,567 tetrahedra and 76,865 points. Of these as many as 12,804 points represented the surface, thus giving a fine resolution on and near the surface.

Figure 3 shows a comparison of C_p at two locations on the wing, at 50% and 70% of root chord, respectively. Comparison is made between the adapted grid, the 'fine' grid and the experimental data. The adapted grid results agree closely with the 'fine' grid results. The computed inviscid results differ from the experimental (viscous) data in an expected manner [9]. The total CPU time for the adaptive cycle was 5,963 seconds on a Cray-YMP compared to 8,893 seconds for the 'fine' grid, which demonstrates the efficiency of the adaptive scheme.

Case 2: A MTVI Configuration

The next test case is a Modular Transonic Vortex Interactions (MTVI) wind tunnel model. It employs a 60° sharp-edged cropped delta wing with a segmented leading edge flap, and a chine shaped fuselage. Experimental data is available from a wind tunnel test conducted to investigate the interactions between the chine-forebody vortices and different vertical tail arrangements. The adaptive grid study was conducted for $M_\infty = 0.4$ and $\alpha = 10.54°$. This flow condition is characterized by the presence of multiple vortices.

For this case, computations were begun on a relatively coarse grid with 188,304 tetrahedral cells and 35,388 points (GRID 1). Two successively finer adapted grids were generated using the procedure described above. The final adapted grid (GRID 3) had 371,360 cells and 68,158 points.

Figure 4 shows a comparison of surface triangulation for the unadapted (GRID 1) and the adapted (GRID 3) cases. For the adapted case, clustering of the surface grid near leading edge as well as near the wing tip is evident representing the static pressure peak due to proximity of vortices to the configuration surface. Figure 5 shows a comparison of the field grid projected on to a cross sectional plane located at 93 % of root chord. Grid clustering around the vortex cores between the fuselage and the vertical tail and near the wing tip can be seen clearly. Finally, in figure 6 flow field results are compared between the unadapted grid, the adapted grid and experimental data. The C_p comparison is shown at a streamwise station located at 93 % of the root chord. In this figure, results are also compared from a 'fine' grid with 825,469 tetrahedra and 148,285 points [10]. During generation of the 'fine' grid, no special effort was made to cluster grid points around vortex core locations. A higher suction peak resulting from grid adaptation is evident. The fine grid did not capture the suction peak at the station shown due to lack of grid resolution there, while the adapted grid automatically provided the grid resolution needed. This case clearly establishes the ability of the adaptation procedure to automatically provide adequate grid resolution where needed. The whole adaptation cycle required about 2.5

hours of CPU time on a Cray-YMP computer, compared to about 4.0 hours for the 'fine' grid. The adapted results show a better accuracy with about 2.2 times less number of cells and about 38 % CPU saving compared to the 'fine' grid case.

CONCLUSIONS

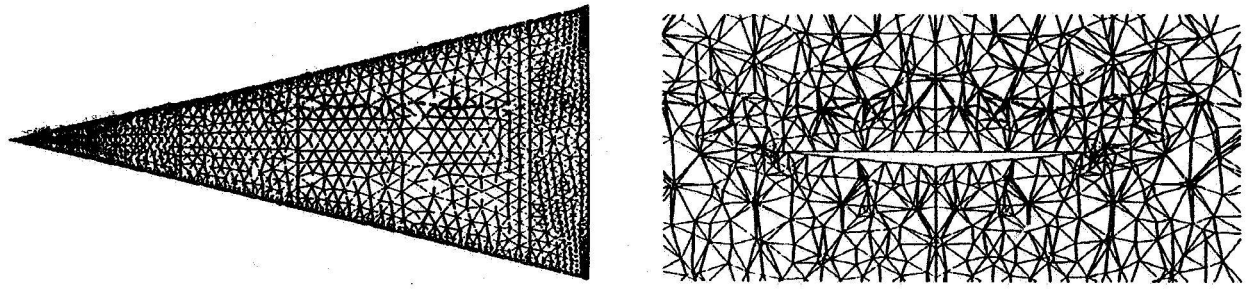
An automated adaptive remeshing scheme for vortex dominated flows is described. The procedure is shown to be an efficient technique for obtaining solutions especially when the locations of important flow features within the three dimensional flow field are not known. It is also shown that an adapted grid requires a smaller total number of grid elements compared to an unadapted grid, thus making the solution more amenable to a workstation environment.

ACKNOWLEDGEMENTS

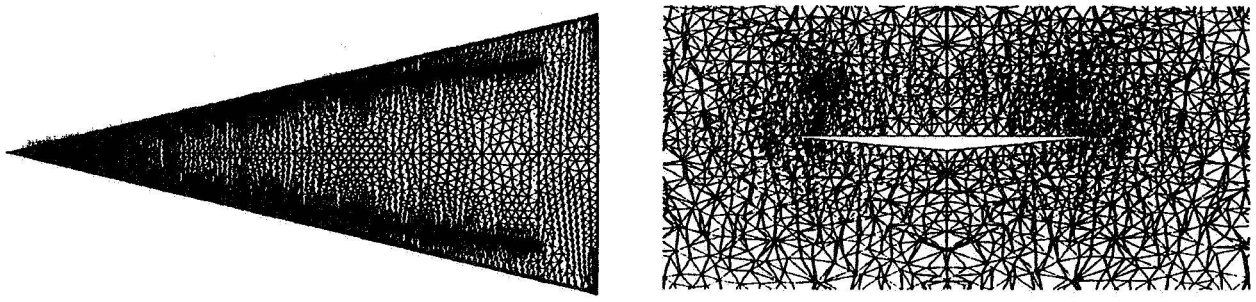
The work was supported under NASA Langley Research Center contract NAS1-19672 to ViGYAN, Inc. The author would like to thank Dr. Neal T. Frink, Transonic Supersonic Aerodynamics Branch and Dr. Shahyar Pirzadeh of ViGYAN, Inc. for many fruitful technical discussions. Thanks are also due to Mr. Brent L. Bates, ViGYAN,Inc. for MTVI fine grid solution.

REFERENCES

1. Thompson,J.F.; and Weatherhill,N.P., Aspects of Numerical Grid generation: Current Science and Art. AIAA-93-3539-CP, August 1993.
2. Dannenhoffer III,J,F., A comparison of Adaptive Grid Redistribution and Embedding for Steady transonic Flows,*International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 653-663.
3. Parikh, P.; and Frink, N.T., An Adaptive Remeshing Procedure for Three Dimensional Unstructured Grids, 4^{th} International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
4. Frink,N.T.; Parikh,P.; and Pirzadeh,S., Aerodynamic Analysis of Complex Configurations using Unstructured Grids, AIAA Paper 91-3292, 1991.
5. Pirzadeh,S., Structured Background Grids for Generation of Unstructured Grids by Advancing Front Method, *AIAA Journal*, Vol.31, No.2, February 1993, pp.257-265.
6. Frink, N.T., Improvements to a Three Dimensional Unstructured Grid Flow Solver, AIAA Paper 94-0061, 1994.
7. Borsi, M., et al, Vortical Flow Simulation by using Structured and Unstructured Grids', In *Vortex Flow Aerodynamics*, AGARD-CP-494, Scheveningen, The Netherlands, October 1990.
8. Hummel, D., On the Vortex Formation over a Slender Wing at Large Angles of Incidence, AGARD-CP-247, Paper 15, 1978.
9. Ghaffari, F., On the Vortical-Flow Prediction Capability of an Unstructured-Grid Euler Solver, AIAA-94-0163, January 1994.
10. Bates, B. L., Private Communication.



UNADAPTED (GRID 1)



ADAPTED (GRID 2)

Figure 1: Surface Adaptation

Figure 2: Field Adaptation

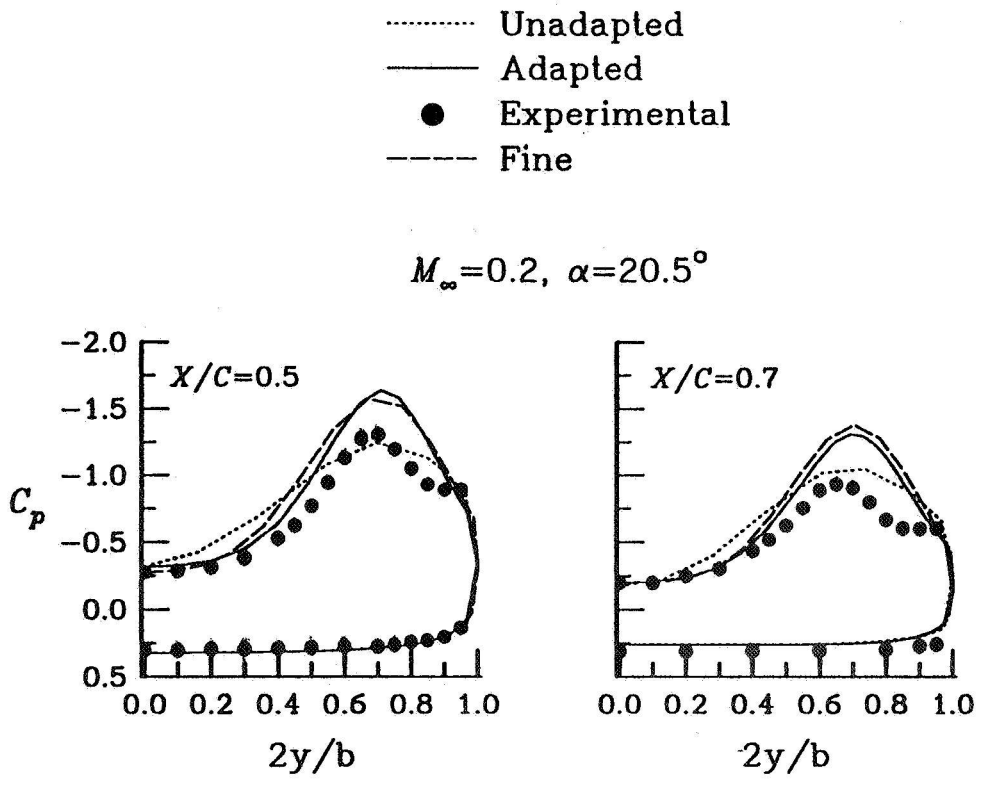


Figure 3: Comparison of Pressure Coefficient

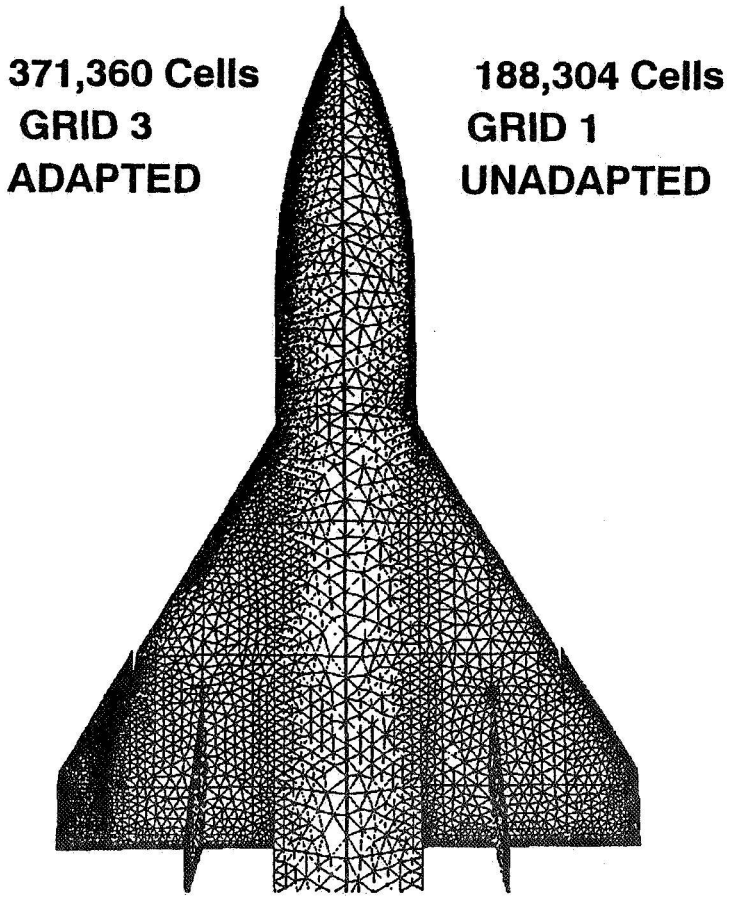


Figure 4: Surface Adaptation

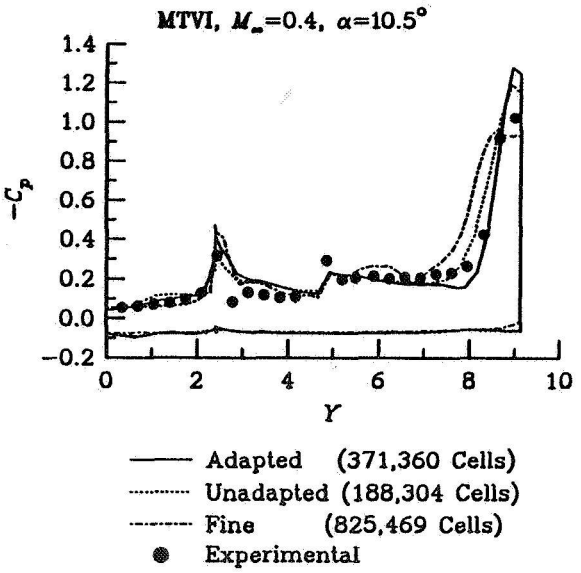


Figure 6: Comparison of Pressure Coefficient at X/C = 93 % Root Chord Location

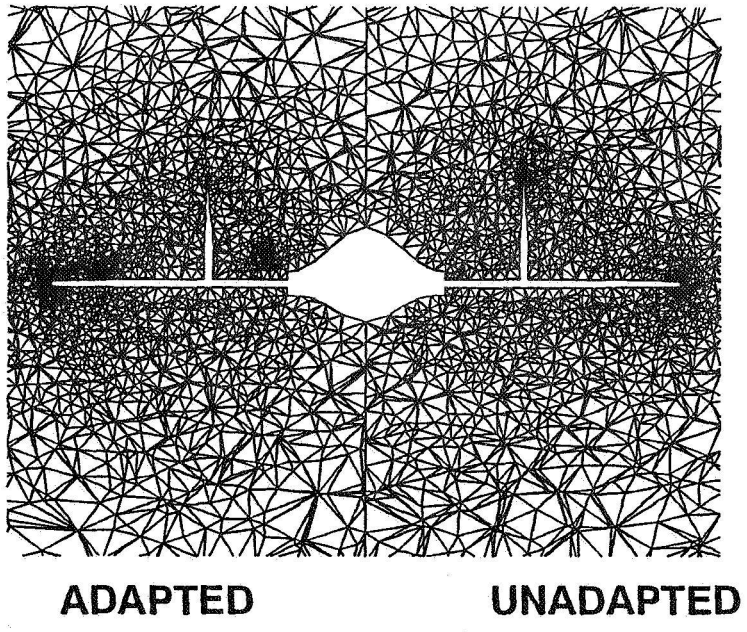


Figure 5: Field Adaptation

AUTO-ADAPTIVE FINITE ELEMENT MESHES

Roland RICHTER * and Pénélope LEYLAND **
* *CRAY Research (Switzerland) S.A., Scientific Parc (PSE)*
** *Institut de Machines Hydrauliques et de Mécanique des Fluides*
Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

Abstract

Accurate capturing of discontinuities within compressible flow computations is achieved by coupling a suitable solver with an automatic adaptive mesh algorithm for unstructured triangular meshes. The mesh adaptation procedures developed rely on non-hierarchical dynamical local refinement/derefinement techniques, which hence enable structural optimisation as well as geometrical optimisation. The methods described are applied for a number of the ICASE test cases are particularly interesting for unsteady flow simulations.

INTRODUCTION

One of the most important advantages of finite element type unstructured grids is the possibility to refine/derefine locally the mesh during the computation. Successive mesh concentration in "critical" zones may be performed, without knowing them *a priori* during the initial mesh creation, as well as coarsening in regions where the nodes seem superfluous. The overall reduction of the number of nodes gives an optimal relation between precision and calculation cost (CPU time and memory constraint), by tracking the strong physical gradients within the flow field by higher grid point concentrations. This is especially important for unsteady flows.

In this paper, inviscid compressible flow calculations from the ICASE Workshop on adapted grids are performed using dynamically auto-adaptive finite element triangular meshes. The mesh optimisation algorithms as well as the adaptation procedures are completely non-hierarchical, which allows more freedom for imposing optimisation strategies for obtaining admissible, regular grids, which can be geometrical, or structural and physical. The algorithm of the dynamical refinement/derefinement procedure is based on a certain number of basic algorithmic principles taking into consideration the particularities of local mesh refinement for finite element type generated meshes. A new anti-data structure has been adopted, where the successive subdivisions are performed independently of the former operations.

LOCAL MESH REFINEMENT

As the goal of mesh adaptation is to increase the accuracy of the solution process by locally enforcing the h-adaptivity by usage of smaller discretisation cells, the first step in a mesh adaptation algorithm is to locally refine the mesh by adding new nodes according to some criteria, thus diminishing local size of the concerned elements. The criteria used should be as close as possible to the error estimations of the underlying discretisation scheme. The principle of adaptive meshing is to uniformly equidistribute the error enhancing the overall convergence.

Error estimates coming from the theory of finite-element simplex type meshes are based upon either *a priori* error estimates coming the governing equations (see e.g. [3]); those based on an *a*

posteriori error estimate where the computed residual of the solution $R(u_h^n)$ is used to define the error [4, 6]; and those which evaluate a combination of the derivatives of the computed variables (e.g. [7]). There is in fact a close relationship between such methods, based on physical gradients, and those of the second kind, [6], at least for hyperbolic transport equations. For the mesh adaptation procedures developed here, the latter strategy has proven to be robust and precise for tracking discontinuities for problems presenting strong and weak shocks, contact discontinuities and so on. For steady state flow resolution, local error criteria are based upon combinations of the L^2 average of $\nabla_h Mach$, $\nabla_h Density$ and $\nabla_h Entropy$, with respect to a certain tolerance; whereas for unsteady flows, these are augmented by combinations of $\Delta_h Mach$, $\Delta_h Density$ and $\Delta_h Entropy$.

We describe here the techniques of mesh adaptation employed in the algorithm, further details can be found in [8, 9].

Symmetrical subdivision of triangles is obtained by adding a node in the middle of each side, rather than in the centre of each triangle, as only this method increases the precision of the numerical method, Figure 1. Indeed, from finite element theory, the interpolation is optimal when the triangle is as equilateral as possible within a certain metric.

Figure 1: Symmetrical triangle subdivision.

The local mesh refinement algorithm consists of testing the segments of an existing mesh whether they are to be split or not, which means that either 1, 2 or 3 new nodes will be created per corresponding marked triangle. For geometrical considerations of maintaining non-acute angles, the creation of 2 new nodes is replaced by a symmetrical subdivision (Figure 1-right). The procedure performs refinement of triangles by subdividing them into 4 or 2 new elements. Division by two leads to zones of elements which may have a non-optimal distribution of the number of neighbours, leading to irregularities within the mesh. The close link between admissibility, regularity and optimality of an unstructured mesh to inherent properties of node neighbour numbers, has been fully exploited in the algorithms developed here, producing new constraints which are often more rigorous and less ad-hoc than existing ones where the adaptation criterion and its tolerance are the only reference points. For a 2D triangular mesh the optimal number of neighbours is 6. Typically, the two following refinement configurations should be avoided - Figure 2, and next section.

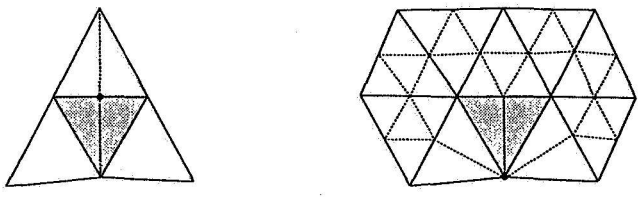


Figure 2: Creation of nodes, with a non optimal node neighbour number.

Their elimination is obtained by requiring that the the number of nodes that can be added within the 3 neighbouring elements should be greater than 2 and smaller than 5. This needs to be completed

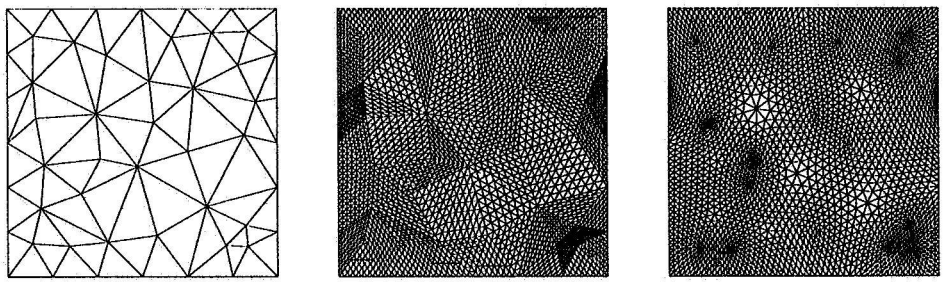
by a geometrical criterion on the length relation between the smallest edge and the others in order to avoid creating highly stretched cells, (see next section).

GEOMETRICAL CRITERIA OPTIMISATION

After a refinement sweep, the new grid still *"remembers"* the old grid structure. To minimise such dependencies it is necessary to smooth globally the mesh as shown in Figure 3, using for instance the standard spring analogy :

$$\widetilde{x}_i = \frac{\sum_{j \in k(i)} \alpha_j \, x_j}{\sum_{j \in k(i)} \alpha_j}$$

$\alpha_j = 1$ and $k(i)$ denotes the nodes neighbouring node i. The resulting mesh shows two specific behaviours : nodes having more than 6 neighbours tend to repulse their neighbours and those having less than 6 tend to attract them.



Initial mesh Refined mesh Smoothed mesh

Figure 3: Geometrical smoothing procedure.

In order to reduce the above magnetic effect, a weight function related to the number of neighbours is introduced :

$$\alpha_j = \max[\, 1,\ 6 \ + \ \beta \, (\, \mathcal{N}_j \ - \ 6 \,) \,]$$

with

$$0 \ < \ \beta \ < \ 4$$

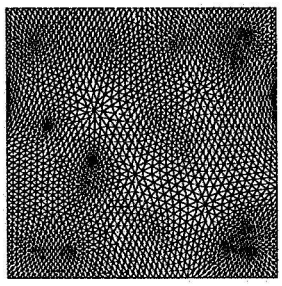


Figure 4: Smoothing with a weight function.

where $\mathcal{N}_j$ denotes the number of neighbours of node j. The result is illustrated in Figure 4.

This optimisation could be more effective if nodes having more than 7 or less than 5 neighbours could be eliminated from the mesh. Such a requirement can be performed, by using the procedure of diagonal swapping with a neighbour node number minimising criterion. Let us denote by $\mathcal{N}_1$, $\mathcal{N}_2$ the number of neighbours for the vertices of a segment and $\mathcal{N}_3$, $\mathcal{N}_4$ these numbers after swapping the segment as shown in Figure 5. Diagonal swapping is accepted if either :

$$\mathcal{N}_3 + \mathcal{N}_4 < \mathcal{N}_1 + \mathcal{N}_2$$

or

$$\begin{cases} \mathcal{N}_3 + \mathcal{N}_4 = \mathcal{N}_1 + \mathcal{N}_2 \\ \max(\mathcal{N}_3, \mathcal{N}_4) < \max(\mathcal{N}_1, \mathcal{N}_2) \end{cases}$$

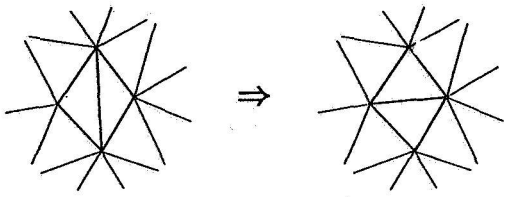


Figure 5: Diagonal swapping.

Then the procedure allows to limit, in most cases, the maximum number of neighbours to 7 and thus increase the number of nodes having an optimal number of neighbours.

To set the minimum number of neighbouring nodes, a technique of suppressing undesirable triangles can be used, which acts as shown in Figure 6.

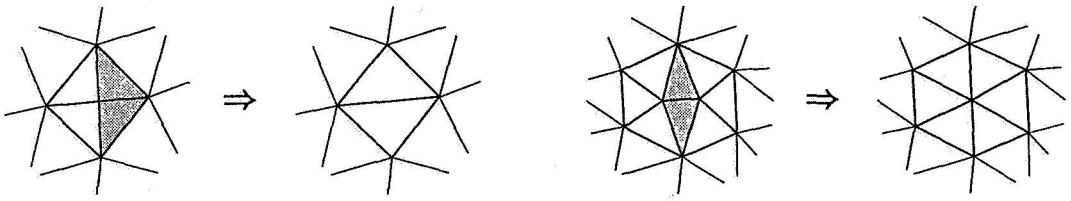
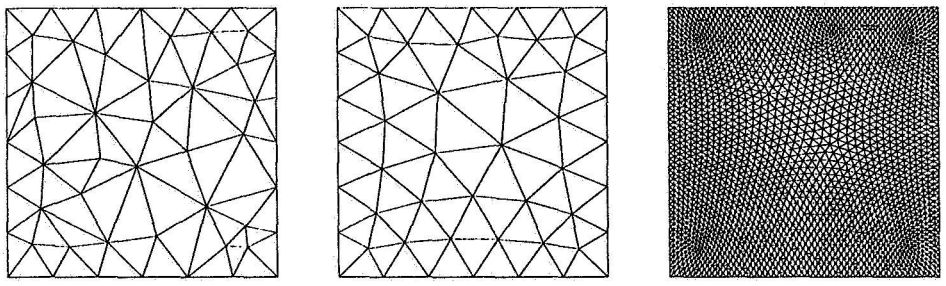


Figure 6: Cancellation of degenerate configurations.

The algorithm used consists in marking segments having a vertex with less than 5 neighbours or with exactly 5 on each side. In both cases, the specific segment and their associate triangles, the shaded elements of Figure 6, can be suppressed.

For the above square mesh, these techniques allow to regulate the optimal node neighbour number, and, combined with the weighted smoothing function, a considerable improvement in mesh regularity is achieved.

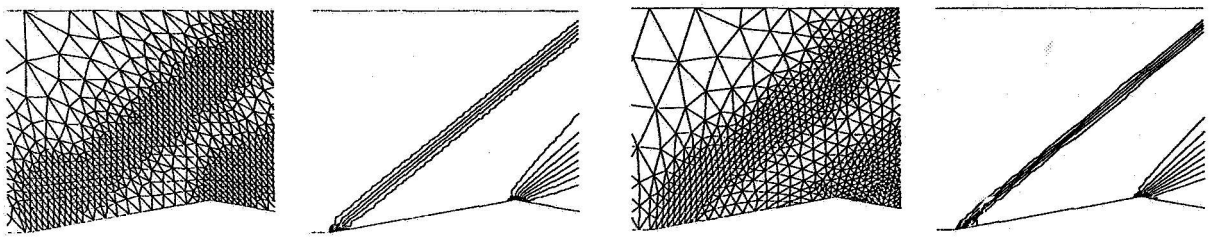


Initial mesh Optimised mesh Optimised and refined

Figure 7: Geometrical optimisation procedure.

PHYSICAL CRITERIA OPTIMISATION

The techniques of structural changes via moving nodes and diagonal swapping can also be applied depending on physical quantities. An improvement of the accuracy of the capture of a discontinuity can be obtained by aligning the edges. In Figure 8 the orientation of the edges were originally normal to a shock, which produces, on the left hand side, a relative thick shock. On the right hand side, a procedure based on diagonal swapping has been used, minimising the angles between the discontinuity and edge.

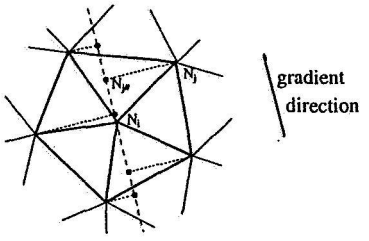


Standard refinement Refinement with edge alignment

Figure 8: Improvement by aligning edges in the direction of the discontinuity.

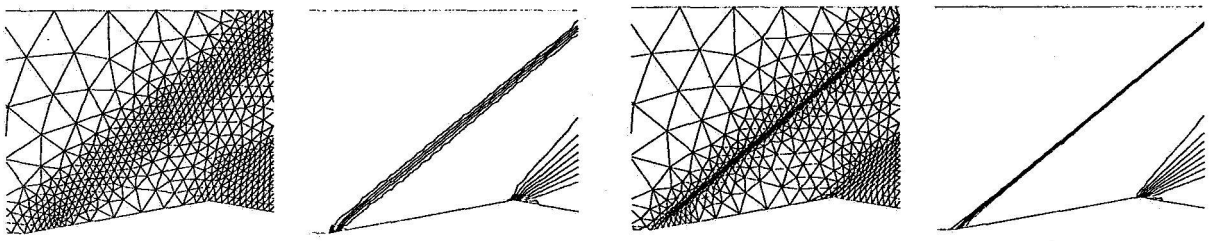
A modified spring analogy using a weight function related to a physical quantity can also be applied to obtain a squeezing property in regions of strong gradients. The weighted spring analogy is increased by the projection of neighbouring nodes onto the direction of the local gradient, and by weighting this projection by the local physical difference. This part of the algorithm corresponds to:

$$\tilde{x}_i = \frac{\sum_{j \in k(i)} \alpha_j x_j + \beta_{ij} \left(x_i + \frac{\overline{x}_i x_j \cdot \nabla \rho_i}{\|\nabla \rho_i\|} \right)}{\sum_{j \in k(i)} \alpha_j + \beta_{ij}}$$



Projection of node neighbours onto a gradient's direction.

where α_j denotes the node number weight function, β_{ij} the local physical difference and $\nabla \rho_i / \|\nabla \rho_i\|$ the gradient direction of the physical quantity. Here the density has been chosen as the specific physical quantity.



Standard refinement Refinement with squeezing

Figure 9: Improvement by local squeezing around discontinuities.

MESH DEREFINEMENT OR COARSENING

In order to optimise the ratio precision and accuracy versus minimal number of discretisation nodes, dynamically auto-adaptive remeshing via local refinement *and* local derefinement (or coarsening) is performed. Most local mesh derefinement techniques are based on a hierarchical data structure, which allows forward and backward scanning through the mesh, and maintains a history of previously added nodes. We have developed a new anti-hierarchical data structure; this algorithm enables nodes of an initial mesh to be removed, and also allows the use of the above structural optimisation techniques, which are incompatible with conventional hierarchical data structures.

This new algorithm is constructed as follows. The graph of points to be derefined is not straightforward, so the inverse problem is solved. A list of nodes, called fixed points, which are never to be derefined is identified. These are either the singular nodes defining, for instance, the corners of the domain, symmetry points..etc., or the nodes belonging to segments to be refined, or nodes having more than 6 neighbours. Then there are a number of possible fixed nodes allowing a maximal coarsening which come from either boundary or internal configurations. To find these, first the border of the domain is scanned so that each second node becomes a fixed point, and finally the configuration of Figure 10 is searched through the mesh. Thus all triangles containing no fixed point have at least three fixed neighbouring nodes, as shown in Figure 10.

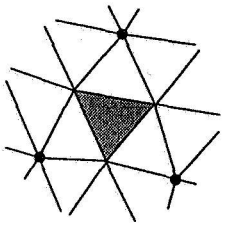
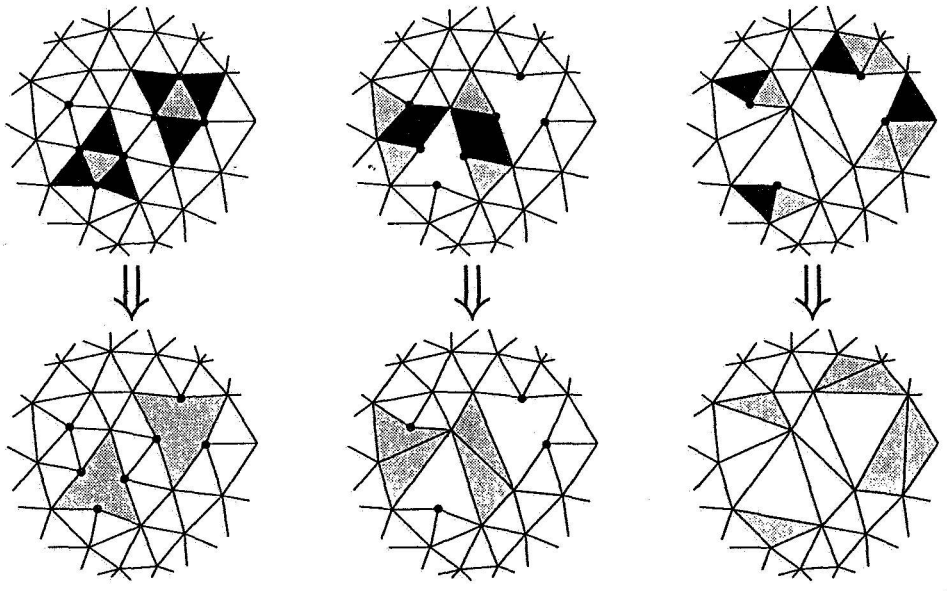


Figure 10: An element defined with 3 removable nodes should be surrounded by 3 fixed nodes.

Thus the grid becomes coloured according to the nodes to be removed and those to be kept in the mesh. The derefinement can then be done by performing successively the operation presented in Figure 11. It is necessary to treat firstly the element with 3 nodes to be invalidated, then the cases with 2 nodes to be removed, and finally the case having only one node which disappear. The dark shaded elements of Figure 11 vanish during the procedure and the light shaded triangles change their shape.



Remove triangles with 3 marked nodes. | Remove triangles with 2 marked nodes. | Remove triangles with 1 marked node.

Figure 11: Derefinement stages.

ALGORITHM

The whole algorithm becomes thus : first, a list of nodes to be invalidated is defined, followed by their destruction, even if they belong to the initial macro-mesh. Then the phase of local enrichment (adding new nodes) is performed by adding them in the middle of segments. This is followed by both structural optimisation procedures, (first the physical diagonal swapping and then the geometrical ones), and finally the mesh is smoothed with the weighted spring analogy to obtain a regular, conformal and admissible mesh. For steady state simulations the squeezing procedure is performed after a specified convergence on the last two meshes. The total number of different meshes created depends upon the physical solution, for steady state calculations, between 4 to 8 different stages of remeshing are performed. The whole procedure is directly integrated into the flow solver giving a fully dynamic auto-adaptive mesh algorithm. For unsteady calculations, the remeshing depends greatly upon the speed of the transient states, and for a calculation cycle, there can be as many as 400 or more new meshes created. The increase of CPU time introduced by these procedures is the order of 2 to 5 % for steady state calculations, and is inferior to 30 % for unsteady ones. The internal accounting of memory requirements are managed by a dynamical memory manager and a archival data base structure; an updated mesh replaces the former one which requires only a minimal extra memory allocation.

APPLICATIONS FOR EXTERNAL FLOWS OVER PROFILES

The flow solver used for all these applications is based upon an equivalent Galerkin finite volume approximation on the dual control volumes of the P1 triangular simplex. A Jameson type centred scheme is employed with artificial dissipation, [9, 8].

Transonic flow over a NACA0012 at $M_\infty = 0.80$, $\alpha = 1.25°$

This is the AGARD 01 test case [1], concerning transonic flow over a NACA profile. Its particularities are the presence of a weak shock on the windward side which is a good challenge for testing shock capturing criteria for local refinement and squeezing in this region, where the gradients are not as strong as those that can be observed, for instance, with another very standard test case, $M_\infty = 0.85$ $\alpha = 1.°$, (AGARD 02). The shock on the leeward side is a much sharper discontinuity, and thus is easier to localise. Since the test case concerns a non-zero angle of attack flow over a profile in a bounded domain, a non-zero circulation is induced on the profile, it is necessary to correct the outer boundary condition in order to simulate as well as possible an infinite domain [10]. The sensibility of the values of the lift coefficient to this induced circulation effect is high. The outer boundary should be chosen as far as possible, and be of a suitable dimension, as well as a "vorticity" correction which is applied to the infinite boundary conditions. For transonic flows, the infinite boundary values correspond to three entering characteristics and one outgoing one. The outgoing component can allow the generated circulation to influence the infinite boundary values of the solution.

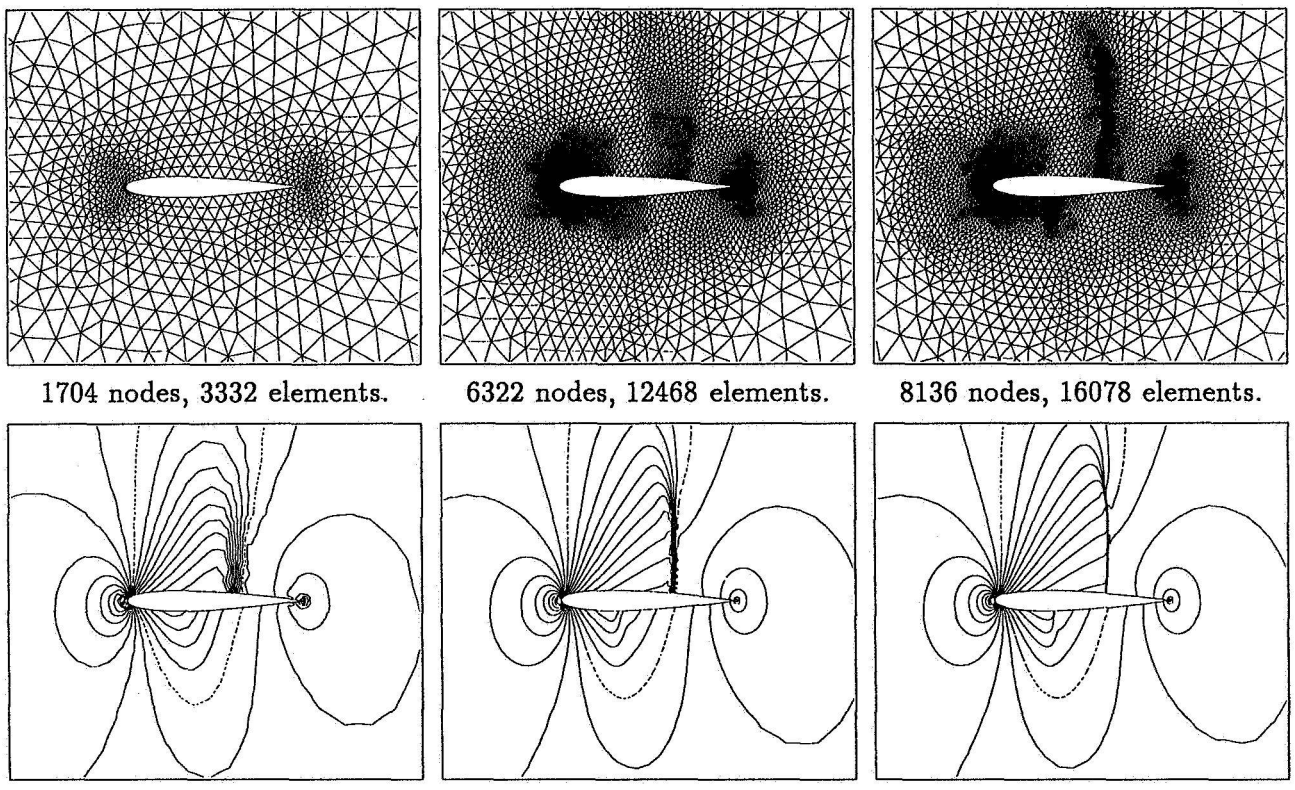
Several different geometrical shapes for the domain boundaries were tested, as well as varying profile to infinite boundary lengths, as a function of the chord length. For a fixed outer boundary distance, the best results were obtained by a circular outer boundary. The aerodynamical coefficients are tabulated below, where a significant variation is found even by taking the precaution of applying the vorticity correction. The values stabilize after as distance of 1000 chord lengths, (up to 10000 chord lengths were tested). They correspond well to the references values established by AGARD.

The solution adaptive mesh procedures presented in this paper were applied, and despite the

Table 1: Flight coefficients for NACA0012 at $M_\infty = 0.80$, $\alpha = 1.25°$

Outer boundary	C_l	C_d	C_m	X_{upper}	X_{lower}
10 chords	.3375	.0224	-.0351	.6274	.3441
30 chords	.3495	.0232	-.0371	.6327	.3422
100 chords	**.3549**	**.0235**	**-.0382**	**.6353**	**.3382**
1000 chords	.3569	.0237	-.0387	.6363	.3357

difficulties of capturing the weak windward side shock, a highly satisfactory adaptive mesh was obtained. For this test case and the next one, where shocks and expansion fans are present within the flow field, a combination of gradients and differences of physical variables were chosen to provide the adaptation criteria as explained in the previous sections. All six distinct meshes were generated during the process. The initial mesh was of 1704 nodes and 3332 elements, the final one of 8136 nodes and 16078 elements, Figures 12.



1704 nodes, 3332 elements. 6322 nodes, 12468 elements. 8136 nodes, 16078 elements.

Figure 12: Evolution of the meshes and Iso C_p lines for a NACA0012 at $M_\infty = 0.80$, and $\alpha = 1.25°$.

A partial view is given below, Figure 13, showing clearly the various types of adaptation performed, local refinement, regularisation, alignment, squeezing, and the corresponding pressure coefficient body profiles are presented to demonstrate the precision of solver on this adapted mesh.

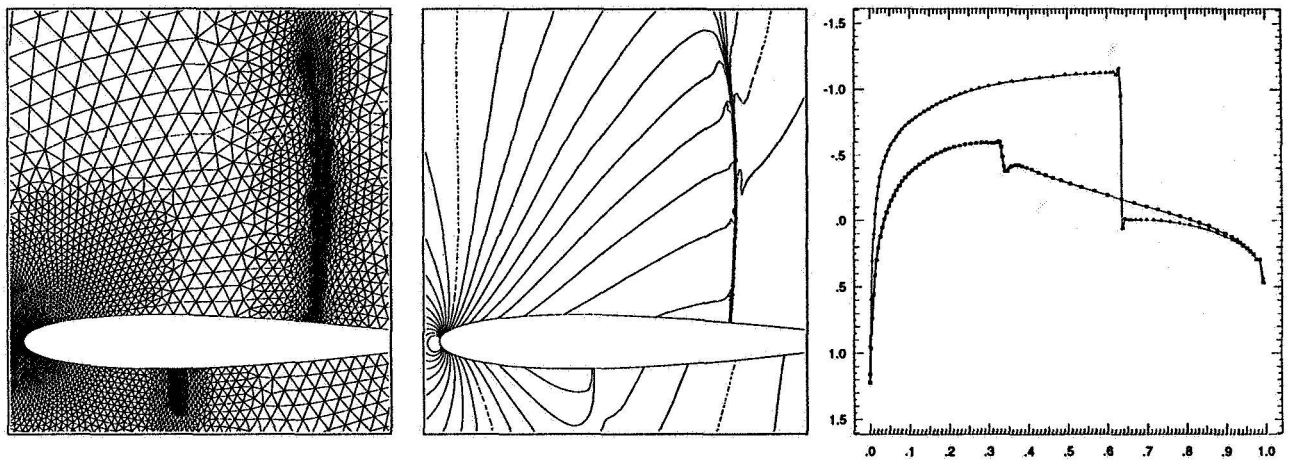
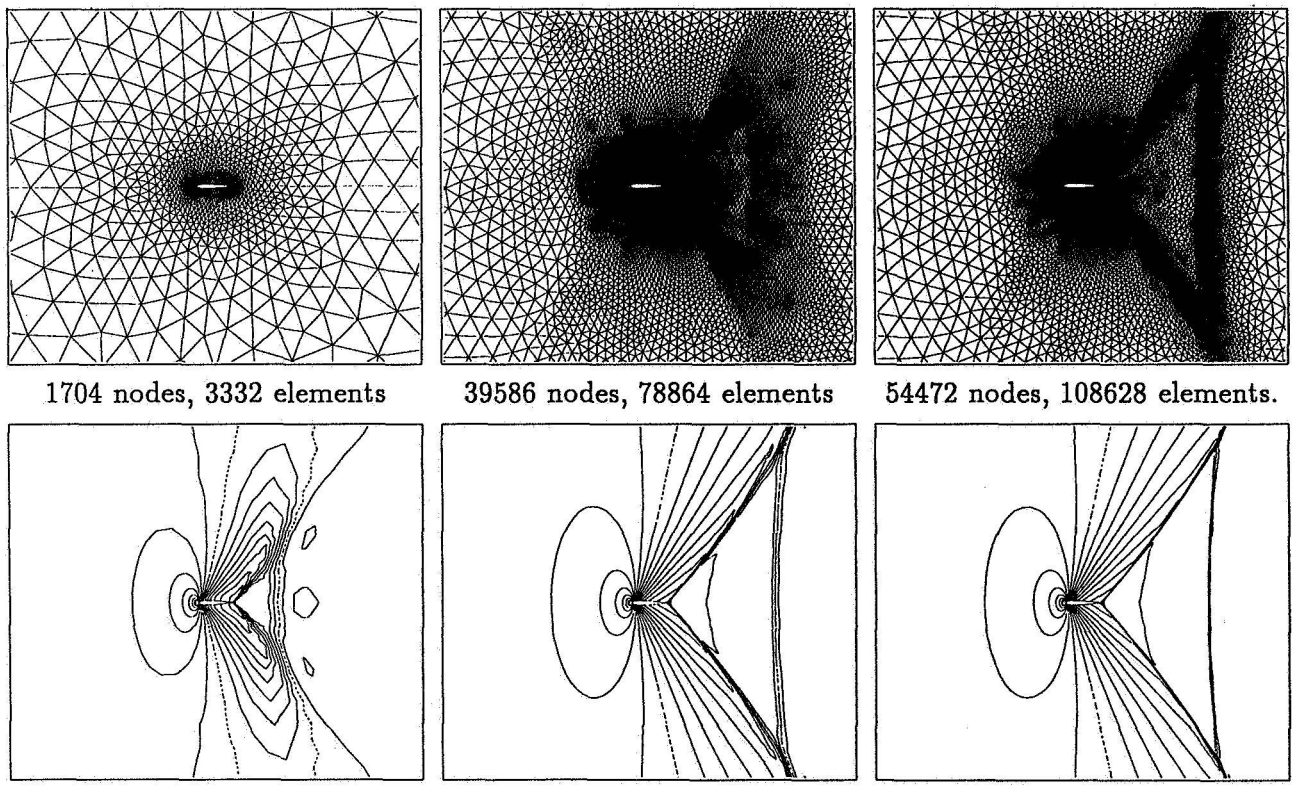


Figure 13: Mesh Details, Iso C_p lines ($\Delta C_p = 0.1$) and C_p body coefficients.

Transonic flow over a NACA0012 at $M_\infty = 0.95$, $\alpha = 0°$

The second test case over the NACA 0012 presented here corresponds to the AGARD 03 test case. Despite the zero angle of attack for this high transonic flow, there is again a great sensibility of the solution to the position of the outer boundary. Indeed, the test case gives rise to a fish-tail shock structure, with an oblique shock attached to the trailing edge and a normal shock emanating from the intersection of the sonic line with the trailing edge oblique shock (see Figures 14). The distance



1704 nodes, 3332 elements | 39586 nodes, 78864 elements | 54472 nodes, 108628 elements.

Figure 14: Evolution of meshes and Iso C_p lines, for a NACA0012 at $M_\infty = 0.95, \alpha = 0.0°$

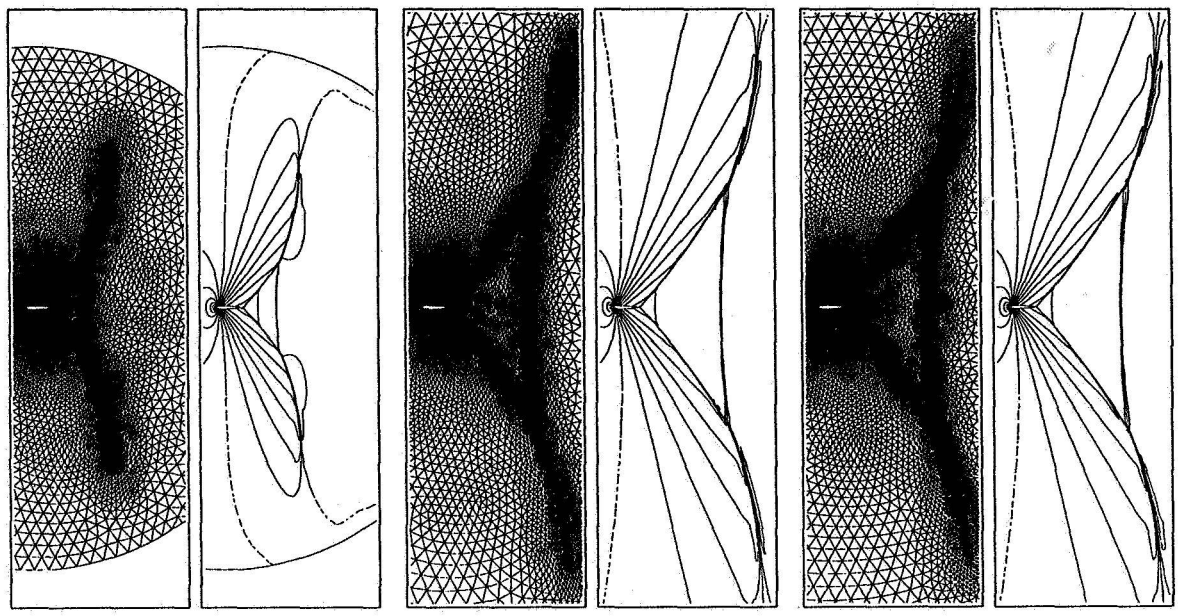


Figure 15: Meshes and C_p solutions for different outer boundary distances (10, 100, 10000 chords).

of this normal shock to the trailing edge is extremely sensitive not only to the position of the outer boundary, but also to the degree of adaptation around the body itself due to the strong expansion waves which are created along the profile.

The outer boundary was placed successfully at 10, 30, 100, 1000 and 10000 chord lengths from the body. Whereas the flight coefficients converged after 100 chords, the normal shock position continues to change throughout the series, increasing with increasing outer boundary distance. The degree of adaptation for these comparisons (Figures 15) was kept constant at a level equivalent to those of the final mesh shown in the Figures 14. The evolution of the refinements for the case with 100 chords are given in the Figures 14.

The convergence of the flight coefficients and the distance X_S of the normal shock to the trailing edge (normalised to the chord length) are tabulated below. As mentioned above, the normal shock distance evolves with the outer boundary distance, as has also been observed in the literature, [1]. The orders of magnitude of the drag coefficient are in agreement with the references, however the

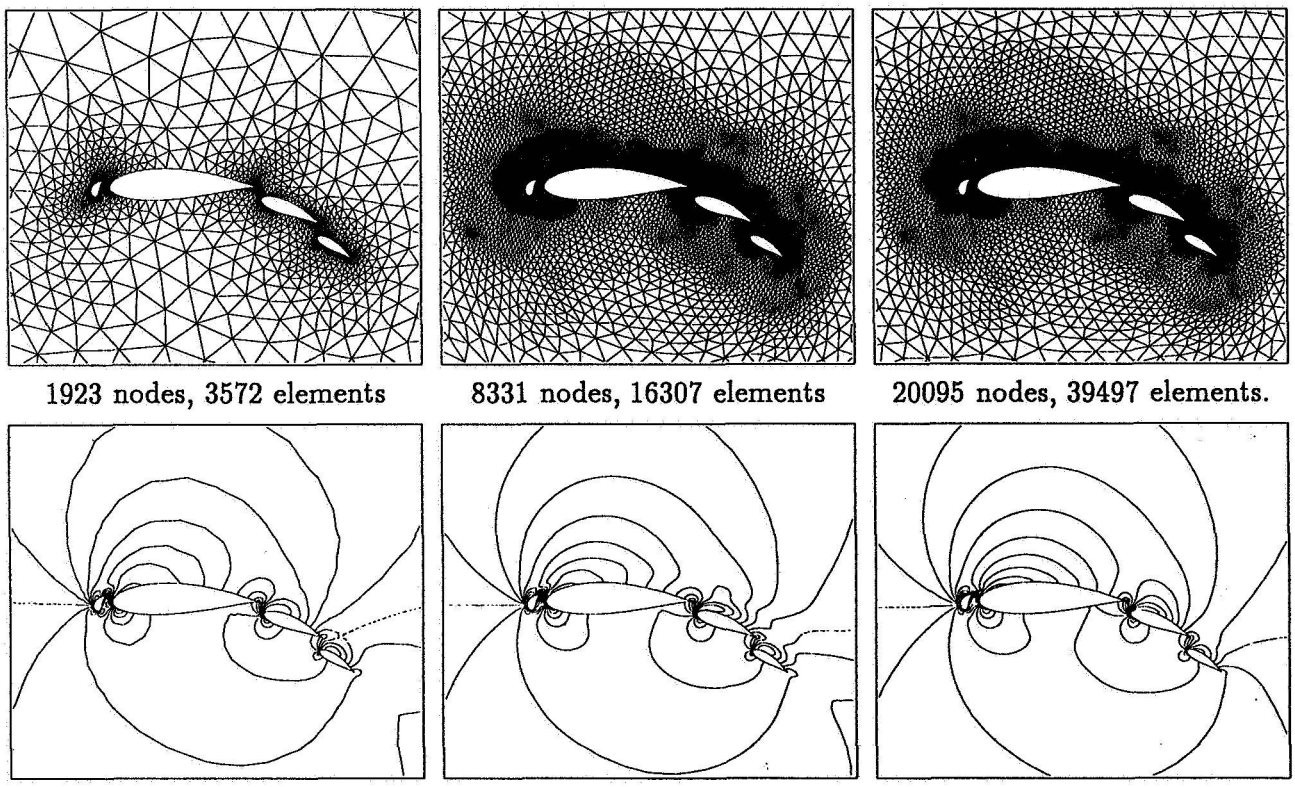
Table 2: Flight coefficients for NACA0012 at $M_\infty = 0.95$, $\alpha = 0.0°$

Outer boundary	C_l	C_d	C_m	X_S
10 chords	.0007	.1090	-.0002	1.280
30 chords	-.0003	.1091	.0002	2.515
100 chords	**.0004**	**.1092**	**-.0001**	**3.125**
300 chords	-.0009	.1091	.0001	3.179
1000 chords	.0003	.1091	.0000	3.195
10000 chords	-.0004	.1091	.0000	3.231

normal shock stand-off distance is found to be closer than the references for a low number of chord distances for the outer boundary. The calculations presented within this monograph however were often on non-adaptive grids, and for a relatively low number of chord lengths for the outer boundary distance.

Subsonic flow over a multi element airfoil at $M_\infty = 0.2$, $\alpha = 0.0°$

This test case, corresponds to a four-element airfoil of Suddhoo and Hall, obtained by applying the Karman-Trefftz mapping function, [2]. The flow conditions are subsonic with zero angle of attack. The adaptation criteria here were taken to be based upon simply the gradient of the local Mach number, as there are no shocks present within the flow field. Again, this test case is a steady state calculation. Despite the absence of discontinuities the mesh adaptation algorithms provided very regular final meshes, starting from an initial coarse and non-optimal mesh, Figures 16.



1923 nodes, 3572 elements | 8331 nodes, 16307 elements | 20095 nodes, 39497 elements.

Figure 16: Evolution of the meshes and Iso C_p lines, ($\Delta C_p = 0.3.$) for the Multi-Element airfoil test case, for a subsonic flow of $M_\infty = 0.2, \alpha = 0.0°$

Airfoils presenting Non uniqueness of the solutions for the Euler equations

In an AIAA paper concerning airfoils that can occur during an optimum design procedure, where the surface splines are perturbed, A.Jameson found the possibility of obtaining two distinct solution branches of the Euler equations, whereas the meshes used were extremely fine, and convergence was pushed to a maximum, in order to eliminate possibilities of anomalies due to a non-entropy preserving solution [5]. In fact, the Euler equations only admit weak solutions, and the only admissible ones must preserve the mathematical entropy condition. The airfoils studied here show a hysteresis effect in the lift/incidence polar when varying the angle of attack back and forth. The solutions generated

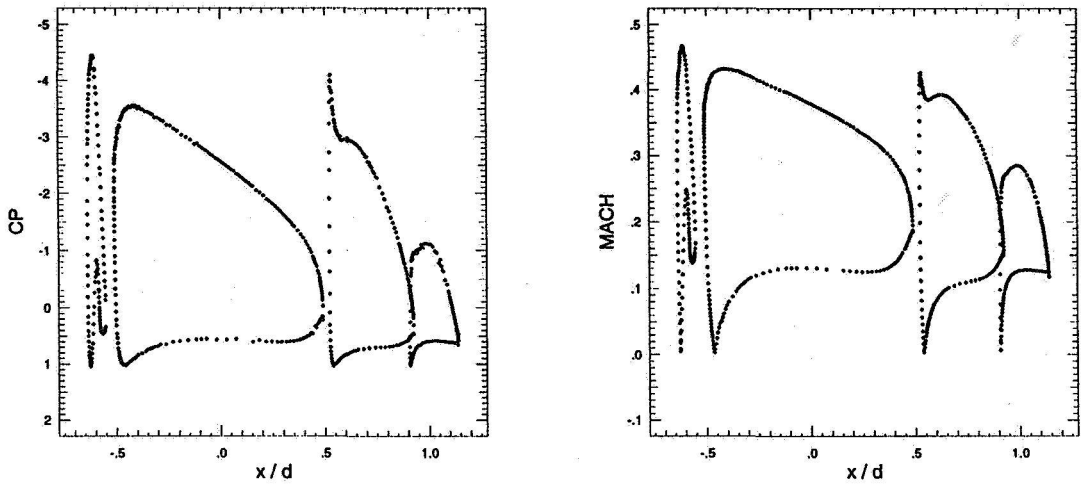


Figure 17: C_p and Mach number body plots, for a multi element airfoil at $M_\infty = 0.2$, $\alpha = 0.0°$.

have two separate supersonic zones and shock waves for a particular angle of attack, and then by increasing or decreasing the lift angle around some critical value, a single supersonic zone is generated with a shift in the lift coefficient. It is extremely important to start this varying α process with an extremely fine and regular grid around the airfoil. The freestream Mach number is fixed at .78, and an initial series of computations is performed upto complete convergence for increasing angle of attack from zero to $-.70$, (up to 30000 time step iterations or more). The different lift coefficients are noted. Then for the solutions around a C_L of .6, angle of attack around $-.43$, a second series are initialised by taking the solution of an angle of attack slightly inferior. As the mesh adaptation is dynamically linked to the solver, the essential characteristics of the changing solution are taken into account, and each calculation is thus made on its own specific adapted mesh. Several different stages of resolution are illustrated in the Figures 19. The hysteresis effect is plotted in Figure 18. The values do not correspond exactly to those of Jameson, as the initial definition of the airfoil was not sufficiently detailed to obtain an equivalent initial shape. The transient stages are well captured by the dynamical mesh method, and the non-uniqueness can be established within a certain margin

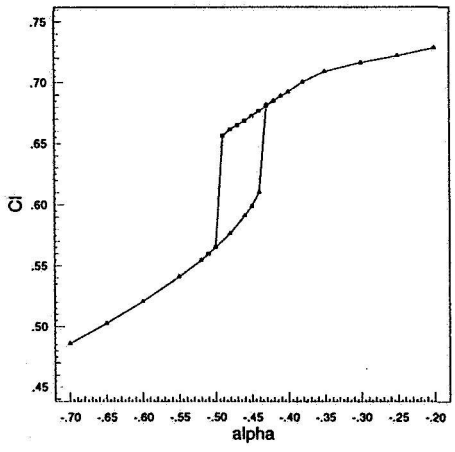


Figure 18: Lift versus angle of attack polar for $Mach_\infty = .78$

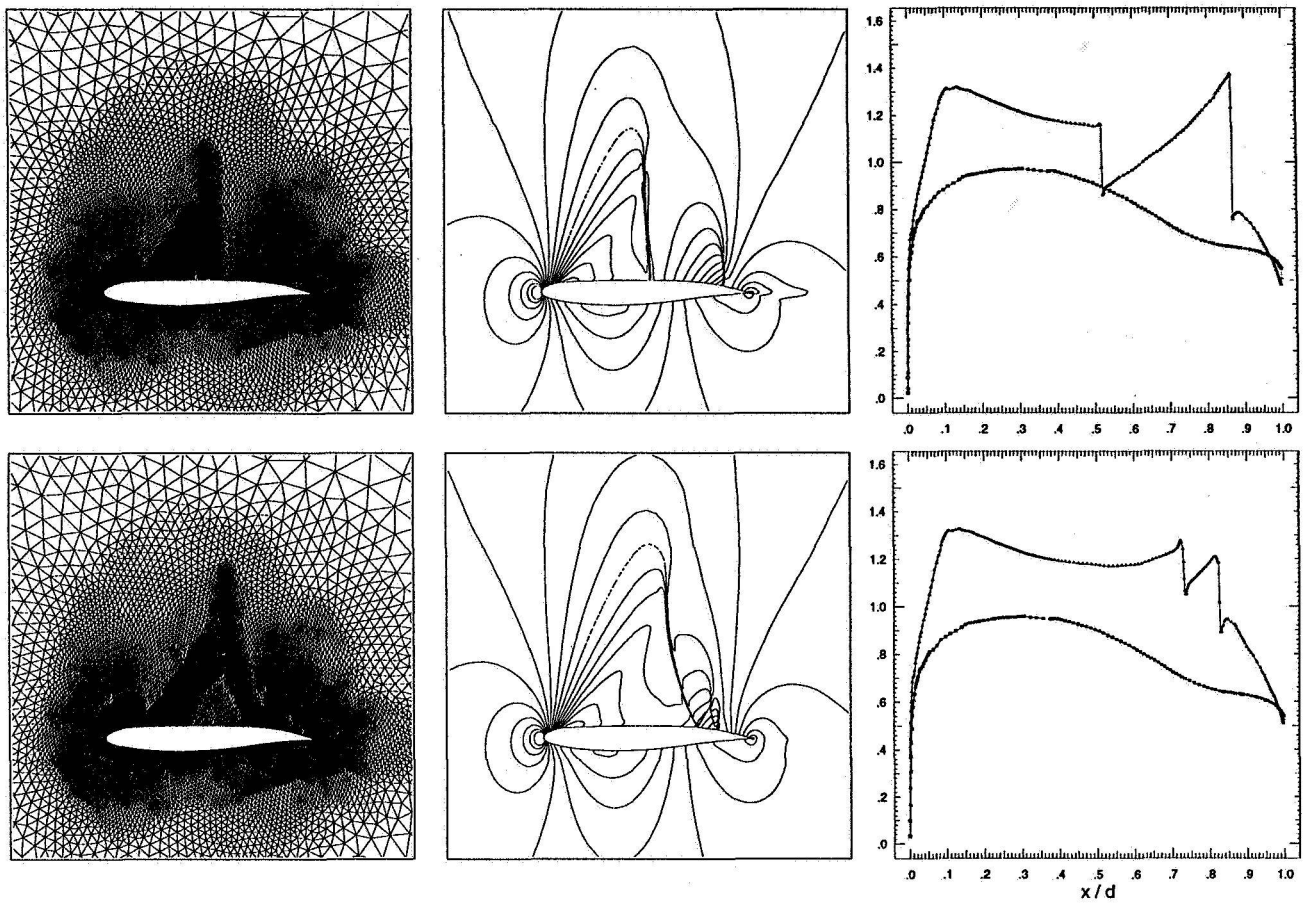


Figure 19: Different Meshes and Mach isolines/body plots obtained for $M_\infty = .78, \alpha = -.47°$, corresponding to hysterisis effect of the lift polar.

of Mach, angle of attack and lift.

CONCLUSIONS

A new "*anti*"-data structure unstructured mesh adaptation procedure has been presented enabling the use of structural optimisation, together with freedom for derefinement of any level of mesh transformation. The improvement, for steady Euler flows solutions, in number of nodes and CPU time requirement can be up to a factor of 30, and the algorithms developed have also proved to produce very accurate results for unsteady cases. The methods have been applied to a number of test cases, and have proved to be a valuable design tool for aerodynamic investigation. The solver and the dynamical mesh algorithm are intimately coupled, their implementation on a distributed computational platform was undertaken in a Master-Slave environment between a CRAY YMP and a CRAY-T3D. The efficiency of such a technique is extremely promising, and in vue of the minimisation of the communication phases by performing the adaptation and the partitioning on the Master, proves to be an interesting alternative to a complete MPP algorithm of adaptation and solution process, which necessitates sorting graphs, redistribution searches and high communication patterns.

References

[1] Test Cases For Inviscid Flow Field Methods, AGARD - AR - 211, WG 07.

[2] ICASE/LaRC Workshop on Adaptive Grid Methods, Hampton Virginia, 7-9 November, 1994.

[3] Babuška, I. and Rheinboldt, W. C. : *Error estimates for adaptive finite element computations.* SIAM J. Numer. Anal., vol. 15, no.4, 1978.

[4] Eriksson, K.; Johnson, C. and Lennblad, J. : *Error estimates and automatic time and space step control for linear parabolic problems.* SIAM J. Num. Anal. 1990.

[5] Jameson, A. : *Airfoils Admitting Non-unique Solutions of the Euler equations* AIAA 91-1625.

[6] Leyland, P.; Benkhaldoun, F.; Maman, N. and Larrouturou, B. : *Dynamical mesh adaptation criteria for accurate capturing of stiff phenomena in combustion.* To appear in Int.Journ. Num. Meth. in Heat and Mass Transfer 1993.

[7] Löhner, R.; Morgan, K.; Vahdati, M.; Boris, J. P. and Book, D.L. : *FEM-FCT: Combining unstructured grids with high resolution.* Comm. Appl. Num. Meth., vol. 4, 1988, pp. 717–730.

[8] Richter, R. : *Schémas de capture de discontinuités en maillage non-structuré avec adaptation dynamique.* PhD Thesis, EPFL, 1993.

[9] Richter, R. ; Leyland, P. *Auto-Adaptive Finite Element Meshes and Distributed CFD* Article submitted for publication, January 1995.

[10] Thomas, J.L. and Salas, M.D. : *Far-Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations* AIAA Journal, no.24, pp. 1074-1080, 1986.

ADAPTIVE MESH REFINEMENT IN CURVILINEAR BODY-FITTED GRID SYSTEMS

Erlendur Steinthorsson and David Modiano
Institute for Computational Mechanics in Propulsion
NASA Lewis Research Center, Cleveland, Ohio

Phillip Colella
Department of Mechanical Engineering
University of California, Berkeley, California

SUMMARY

To be truly compatible with structured grids, an AMR algorithm should employ a block structure for the refined grids to allow flow solvers to take advantage of the strengths of structured grid systems, such as efficient solution algorithms for implicit discretizations and multigrid schemes. One such algorithm, the AMR algorithm of Berger and Colella, has been applied to and adapted for use with body-fitted structured grid systems. Results are presented for a transonic flow over a NACA0012 airfoil (AGARD-03 test case) and a reflection of a shock over a double wedge.

INTRODUCTION

Solution adaptive mesh refinement (AMR) can be used to enhance accuracy and efficiency of many practical flow solvers. By now it is used, almost routinely, in flow solvers employing unstructured grids. Yet, in flow solvers designed for body-fitted curvilinear grid systems (structured grids), similar techniques are rarely used. A possible reason is that the limitations of FORTRAN77, the programming language typically used for structured flow solvers, make it difficult to implement AMR algorithms that are truly compatible with the use of structured grids.

To be compatible with the use of structured grids, an AMR algorithm should allow the flow solver to take full advantage of the strengths of structured grid systems, such as allowing effective use of various efficient solution algorithms for implicit discretizations, various schemes based on dimensional splitting, and multigrid schemes. In essence, the AMR algorithm should use a block structure for the refined grid. To date, only a few methodologies of this nature have been proposed. The method of Berger and Oliger[1] is one of the earliest. In their method, the refined grids are allowed to overlay the underlying coarse grids in an arbitrary manner. The blocks of the refined grids are constructed in physical space based on the shape and size of the region to be refined. The resulting grids tend to align with discontinuities and other features that determine the shape and size of the region. Although this approach has not been widely adopted, the work systematically addressed many important issues related to adaptation of structured grids via local refinement, such as proper nesting of the grid levels and a suitable time-stepping scheme for multi-level grids.

Building on the work of Berger and Oliger, Berger and Colella[2] devised a methodology in which the refined grids conform with the coarse grids, i.e., the boundaries of the fine-grid blocks are made

to coincide with grid lines of the coarse grid. The block structure for the refined grids is created using a special algorithm that operates purely in the index space of the coarse grid. The algorithm fits topologically rectangular patches over the regions of the coarse grids where the error in the solution is estimated to be above a specified threshold value. The blocks on the refined grids are then created by subdividing the coarse grid cells within each of the patches. Advantages of the new approach, compared to that of Berger and Oliger, include greatly simplified prolongation and restriction operators for transferring data between a coarse grid and a refined grid, and rigorously enforced conservation at interfaces between coarse and fine grids. This approach has been used extensively for Cartesian grids (see *e.g.* Ref. 2–5) but only to a limited extent for body-fitted structured grids (see Ref. 6 to 8). Algorithms of similar nature to those by Berger and co-workers have been suggested by other authors but have not been as fully developed (see Ref. 1 for a review). Also, Quirk[9] developed an AMR algorithm, one very similar to that of Berger and Colella, to locally refine Cartesian grid systems.

All the approaches reviewed above have in common that the block-structure of the refined grid is determined automatically at run time when the solution is computed. A different approach was proposed by Davis and Dannenhoffer[10]. In this approach, the entire structured grid system is divided up into sub-blocks of uniform size and dimensions. During adaptation, each sub-block is refined either in its entirety or not at all. In a recursive manner, a sub-block that has been refined is itself divided into sub-blocks, each of which can be refined. In the algorithm of Davis and Dannenhoffer, directional refinement is used, i.e., the grid can be refined in only one, two or three directions as desired.

The methodology used in the present work is based on the AMR algorithm of Berger and Colella. It inevitably resembles the methodology used in Ref. 7, although important differences exist. The present approach is described in some details in Ref. 8. Here, only an outline is given. Results are presented for two inviscid flows, a transonic flow over a NACA0012 airfoil (AGARD-03 test case) and a reflection of a shock over a double wedge.

AMR ALGORITHM—SPECIAL ISSUES FOR STRUCTURED GRIDS

In the AMR algorithm of Berger and Colella, the solution exists on several *levels* of finer and finer meshes which form an hierarchical structure. Each *mesh level*, excluding the coarsest which covers the entire domain, is embedded within the next coarser level and is created where high resolution is required by refining cells on that coarser level. The AMR algorithm performs two major tasks, i.e., to create and maintain hierarchy of mesh levels and to advance in time the solution on the hierarchy. Both are described in detail in Ref. 2. The present adaptation of the algorithm to body-fitted grids is described in Ref. 8. Here, only an outline of the AMR algorithm is given and some special issues related to curvilinear grid systems are pointed out.

Assuming a time-accurate discretization, the AMR algorithm refines simultaneously in space and time by an integer refinement ratio r. Refinement is done where an error estimation procedure based on Richardson extrapolation determines that error is above a specified tolerance. The refined cells are organized into a small number or rectangular blocks, the union of which make up the mesh level. The solution on the hierarchy of mesh levels is advanced in time recursively, level by level, such that every time the solution on a coarse mesh level is advanced by a time step Δt, the solution on the next finer level is advanced r times by a time step $\Delta t/r$. After the r fine grid time steps,

the solution on the fine mesh level is restricted to the underlying coarse grid. At interfaces between coarse and fine grids, boundary conditions for the fine grid are obtained by interpolating the solution on the coarse grid. Conservation on the interface is ensured by agglomorating, through the r time steps on the fine grid, the boundary fluxes on the fine grid that correspond to each coarse grid cell adjacent to the interface, and in a special "refluxing" step, correcting the solution in the adjacent coarse grid cell by the difference between the agglomorated flux and the original flux computed on the coarse grid. This treatment of the interface ensures that under mesh refinement the numerical solution converges to a weak solution of the governing equations.[11]

The AMR algorithm of Berger and Colella operates purely in the index space of the grid system rather than the physical space. Thus, it applies equally to structured body-fitted grid systems and Cartesian grids. Nonetheless, when the algorithm is applied to body-fitted grid systems, special issues arise as refined curvilinear grid systems must be generated from an existing coarser curvilinear grid. This grid refinement must be done carefully to ensure sufficiently smooth grids on all levels of refinement. In this study, the grid refinement is done by combining parametric cubic spline interpolation and Hermite interpolation. The cubic splines, here natural cubic splines, are used to "reconstruct" the grid lines from the discrete grid points. The interpolation is done grid line by grid line and produces cubic polynomials that bridge between any two neighboring grid points on a grid line. Then, the Hermite interpolation is used to bridge between the four cubic polynomials that define the edges of the coarse grid cell and to define the grid points of the refined grids. Since the polynomials describing the shape of the edges of the cell were constructed using cubic splines, the overall refined grid system, obtained by refining the coarse grid cell by cell, will be smooth and at least C^1 continuous. In the flow solver, the proper shape of the cells (*i.e.*, cubic polynomials) is taken into account when cell areas are computed. Thus, the total area of fine grid cells created by refining a single coarse grid cell will always equal that of the coarse grid cell. This greatly simplifies communication between grids on different levels of refinement over the case when straight-side cells are assumed.

For greater details on the AMR algorithm for body-fitted structured grids, see Ref. 8 as well as Ref. 2.

OBJECT ORIENTED IMPLEMENTATION

The methodology described in this paper has been implemented using mixed language programming. A driver module for the AMR algorithm was written in the C++ programming language while all routines performing floating point intensive parts of the algorithm (e.g., the approximate Riemann solver) were written in FORTRAN. This implementation is possible due to the modularity of the AMR algorithm and allows one to take advantage of the strengths of the different programming languages. Here, the strengths of C++, including object-oriented capability, flexible data structures and dynamic memory allocation, make that language very effective for the implementation of a driver for the AMR algorithm while the extensive optimization by FORTRAN compilers of floating point operations on array data structures makes FORTRAN the language of choice for most of the compute-intensive parts. The implementation makes extensive use of the AMR *library* developed by Crutchfield and Welcome.[12] The AMR library is written in C++ and FORTRAN, and is a collection of space-dimension independent *classes* specially designed to aid in implementation of schemes employing the AMR algorithm.

In the present implementation for structured grids, the main objects manipulated by the AMR driver module are instances of *classes* called *LevelBlock* and *MeshLevel*. A *LevelBlock* consists of all data for a single block from a block structured grid and a set of routines that essentially form a self-contained generalized single block flow solver. A *MeshLevel* is a collection of *LevelBlocks* which form a multiblock grid system on a single level of refinement, along with routines that control communication between blocks on the level, communication between coarse and fine levels, and time stepping scheme for the levels. A doubly-linked list of *MeshLevels* forms the hierarchical, adaptively refined grid structure on which computations are performed.

RESULTS

Results are presented for two of the test cases proposed for this workshop, namely, the AGARD-03 test case of a transonic flow over a NACA0012 airfoil, and a reflection of a plane shock over a double wedge. Both cases are inviscid. The computations presented here were done using a discretization of the Euler equations based on the multi-dimensional upwind scheme of Colella.[13]

The first results to be presented are for steady flow over a NACA0012 airfoil at free-stream Mach number of 0.95 and zero angle of attack. This case is a surprisingly tough test case for AMR algorithms as the location of the normal shock that forms behind the airfoil is very sensitive to the proper capturing of the smooth expansion that takes place in the flow at the leading edge of the airfoil.[15] The starting grid system used in the computation was a C–grid with 32 cells on half the airfoil surface, 64 cells from the trailing edge to the outflow boundary, and 32 cells from the airfoil surface to the free-stream boundary. The outer limits of the grid were 100 cords away. The grid in the neighborhood of the airfoil is shown in Fig. 1. In the present computations, the flow conditions at the free-stream boundary were fixed while all variables are extrapolated at the outflow boundary. This treatment of the boundary conditions has the potential to affect the accuracy of the computed solutions. However, the large distance from the airfoil to the free-stream surface does to some extent compensate for the treatment at the boundary. The sensitivity of the solution to the distance to the free-stream boundary has not been studied.

The flow over the airfoil was computed using zero to three levels of refinement with refinement ratio $r = 2$. Figures 2-3 show the computed solution obtained using three levels of refinement. Apparent in Fig. 3 is a low amplitude oscillation behind the oblique shock. This oscillation is due to the low level of artificial dissipation in the present discretization. According to the computations, the location of the normal shock trailing the airfoil, based on where the Mach number on the symmetry line is unity, is at 3.456 cords, 3.346 chords, 3.279 chords, and 3.274, for zero to three levels of refinement, respectively. In comparison, Ref. 14 reports values between 3.32 and 3.35 chords as the correct location. Thus, the error in the solution on the finest grid is within 2%. In the computation shown here, less than 20% of the cells on the coarsest grid were refined to the finest level.

The second test case to be presented is the reflection of a plane shock over a concave double wedge. The wedge angles are 20° and 50°. The shock Mach number is 2.16. An ideal gas with specific heats ratio of 1.4 is assumed. Under these conditions, a Mach reflection is formed as the plane shock hits the first wedge. A second Mach reflection is formed as the Mach stem of the first reflection hits the second wedge. The triple point of the second reflection travels faster in the direction parallel to the plane shock than the triple point of the first reflection and gradually overtakes it. Complicated interactions take place as discontinuities of the two reflections intersect.

Computations were done using two levels of refinement with refinement ratio $r = 4$. The starting grid system used in the computations is shown in Fig. 4. Figures 5 and 6 show the computed solution at two instances in time. Figure 5 shows the solution shortly after the Mach stem of the first reflection has reflected off the second wedge. At the specific moment shown, the reflected shock from the second reflection is about to overtake the slip line emanating from the triple point of the first reflection. In Fig. 6, the triple point of the second reflection has overtaken the triple point of the first reflection.

The computed solution for the double Mach reflection case has not been compared in specific details to experimental data. However, at least qualitatively, the results compare very well to experimental results by Itoh, *et al.*[15] At any time in the simulations, only a small percentage of the coarse grid cells were refined to the finest level. Consequently, the overall cost of the simulation was only a small fraction of the cost of a simulation done using a single-level grid with the same resolution as the finest grid. This demonstrates how a simulation, intractable if traditional unadapted structured grids are used, can become feasible when adaptive refinement is used.

CONCLUSIONS

An algorithm for adaptive refinement of body-fitted structured grids has been tested on two of the benchmark cases of this work shop. The algorithm used is based on the AMR algorithm of Berger and Colella.[2] It uses a block structure for the data on the refined grids, which makes the algorithm very well suited for adaptive refinement of structured body fitted grid systems. The AMR algorithm has been found to work very well for the two test cases attempted and holds great promise for use in general purpose flow solvers employing structured grids.

REFERENCES

1 Berger, M. J. and Oliger, J., "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. of Comp. Physics,* Vol. 53, pp. 484-512, 1984.

2 Berger, M.J. and Colella, P., "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *J. of Comp. Physics,* Vol. 82, pp. 64-84, 1989.

3 Colella, P. and Henderson, L.F., "The von Neumann paradox for the diffraction of weak shock waves," *J. Fluid Mech.*, vol. 213, pp. 71-94, 1990.

4 Bell, J.B., Berger, M., Saltzman, J. and Welcome, M.L., "Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws," *SIAM J. of Sci. and Stat. Computing*, vol. 15, no. 1, pp. 127-138, Jan., 1994.

5 Pember, R.B., Bell, J.B., Colella, P., Crutchfield, W.Y., and Welcome, M.L., "Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow," AIAA-93-3385-CP, Proceedings, 11th AIAA Computational Fluid Dynamics Conference, July, 1993.

6 Berger, M. J. and Jameson, A., "Automatic Adaptive Grid Refinement for the Euler Equations," *AIAA Journal,* Vol. 23, No. 4, pp. 561-568, April 1985.

7 Bell, J., Colella, P., Trangenstein, J., Welcome, M., "Adaptive Mesh Refinement on Moving Quadrilateral Grids," AIAA-89-1979-CP, Proceedings, AIAA 9th CFD Conference, 1989.

8 Steinthorsson, E., Modiano, D. and Colella, P., "Computations of Unsteady Viscous Flows Using Solution Adaptive Mesh Refinement in Curvilinear Body-Fitted Grid Systems" AIAA-94-2330, June, 1994. Also, ICOMP-94-17, NASA TM 106704.

9 Quirk, J.J., Dissertation, Cranfield Institute of Technology.
10 Davis, R. L. and Dannenhoffer, J. F., "Three-Dimensional Adaptive Grid Embedding Euler Technique," AIAA 93-0330, January 1993.
11 Berger, M.J., "On Conservation at Grid Interfaces," *SIAM J. Numer. Anal.*, Vol. 24, No. 5, pp. 967-984, October, 1987.
12 Crutchfield, W.Y. and Welcome, M.L., "Object Oriented Implementation of Adaptive Mesh Refinement Algorithm,", *Scientific Programming,* Vol. 2, number 4, winter 1993.
13 Colella, P., "Multidimensional Upwind Methods for Hyperbolic Conservation Laws," *J. of Comp. Physics,* Vol. 87, pp. 171-200, 1990.
14 Warren, G.P., Anderson, W.K., Thomas, J.L., and Krist, S.L., "Grid Convergence for Adaptive Methods," AIAA-92-1592; AIAA 10th Computational Fluid Dynamics Conference, June 24-26, 1991.
15 Itoh, K., Takayama, and Ben-Dor, G., "Numerical Simulation of a planar shock wave over a double wedge," *Int. J. for Num. Meth. in Fluids*, Vol. 13, pp. 1153-1170, 1991.

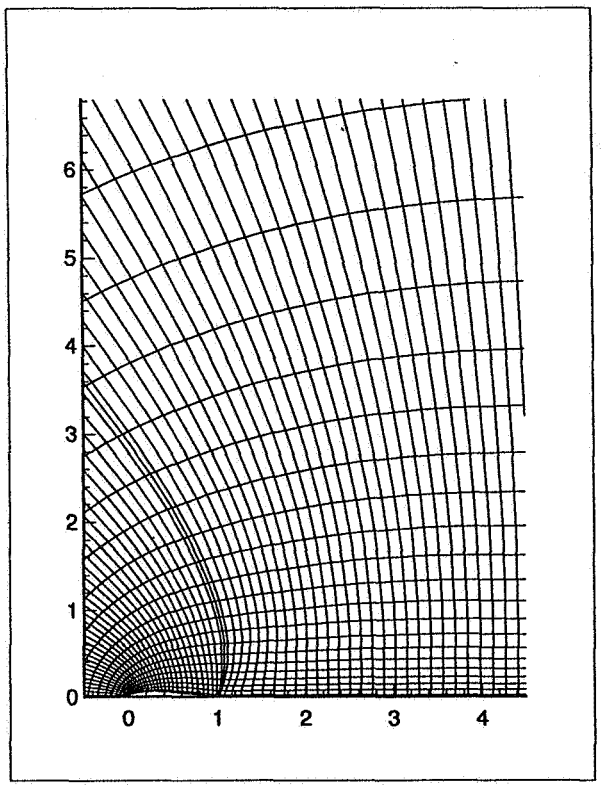


Figure 1. Grid system in two blocks for NACA0012 airfoil—grid lines drawn through cell centers and boundary points.

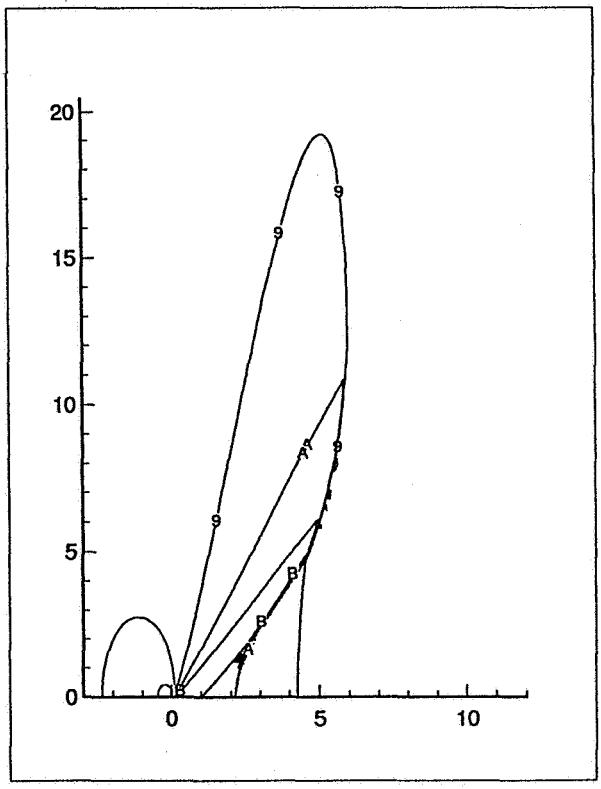


Figure 2. Transonic flow over a NACA0012 airfoil at $M = 0.95$—contours of Mach number: $M_1 = 0.2$, $\Delta M = 0.1$.

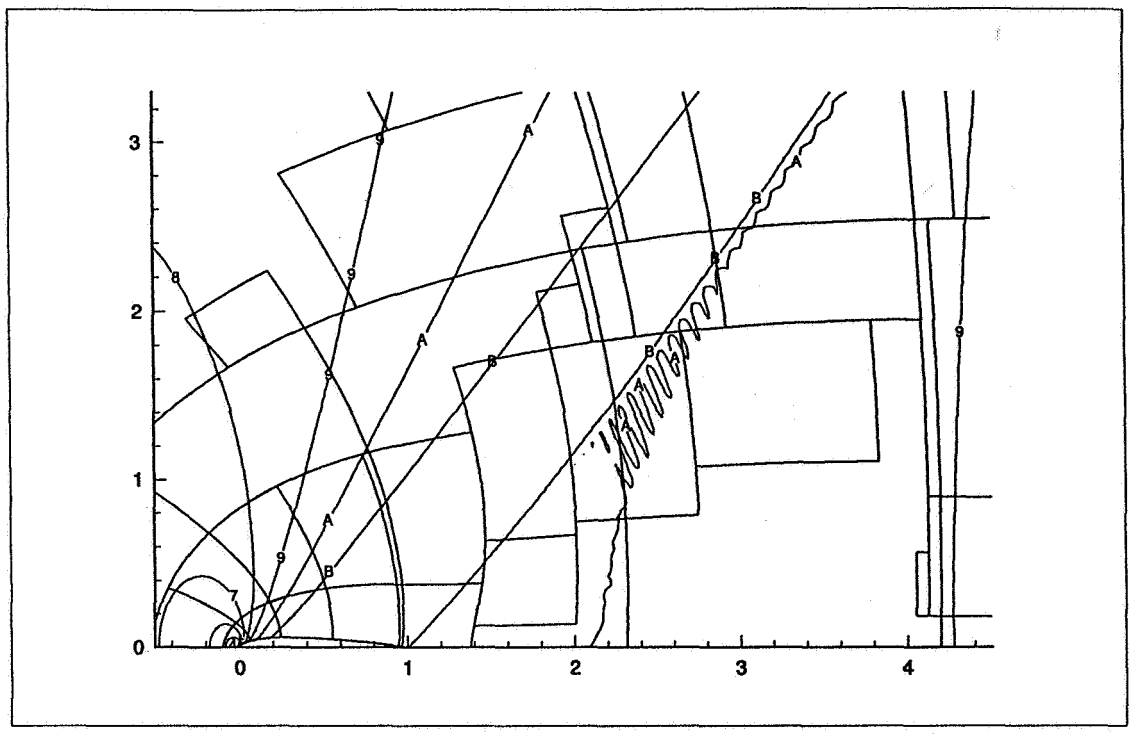


Figure 3. Transonic flow over a NACA0012 airfoil at $M = 0.95$—contours of Mach number: $M_1 = 0.2$, $\Delta M = 0.1$ (boxes indicate boundaries of blocks on the two finest mesh levels).

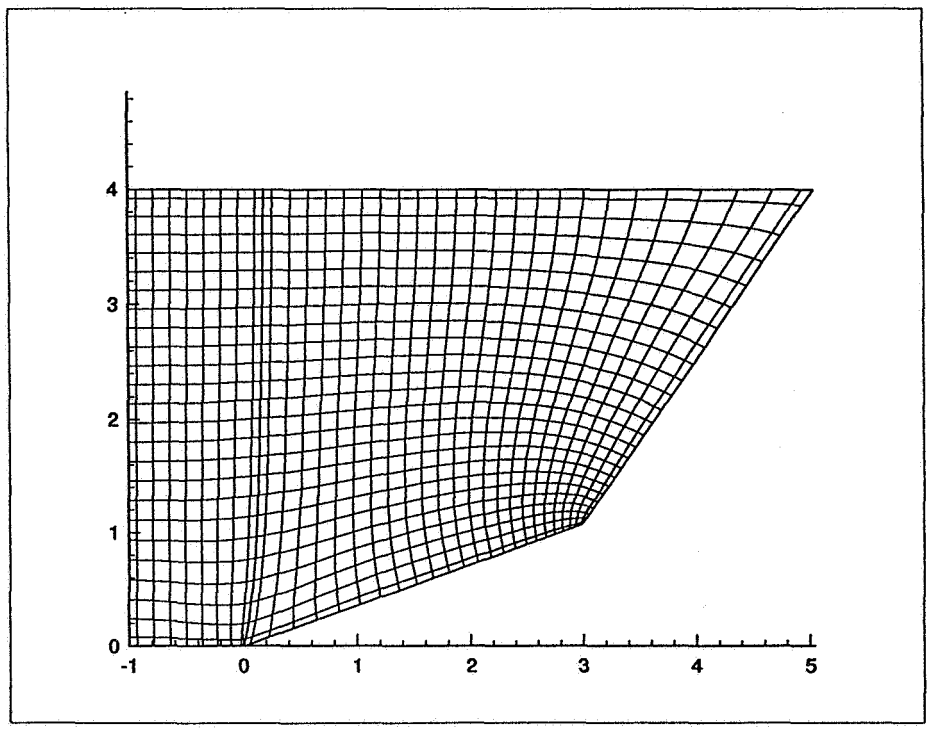


Figure 4. Grid system in two blocks for concave double wedge—grid lines drawn through cell centers and boundary points.

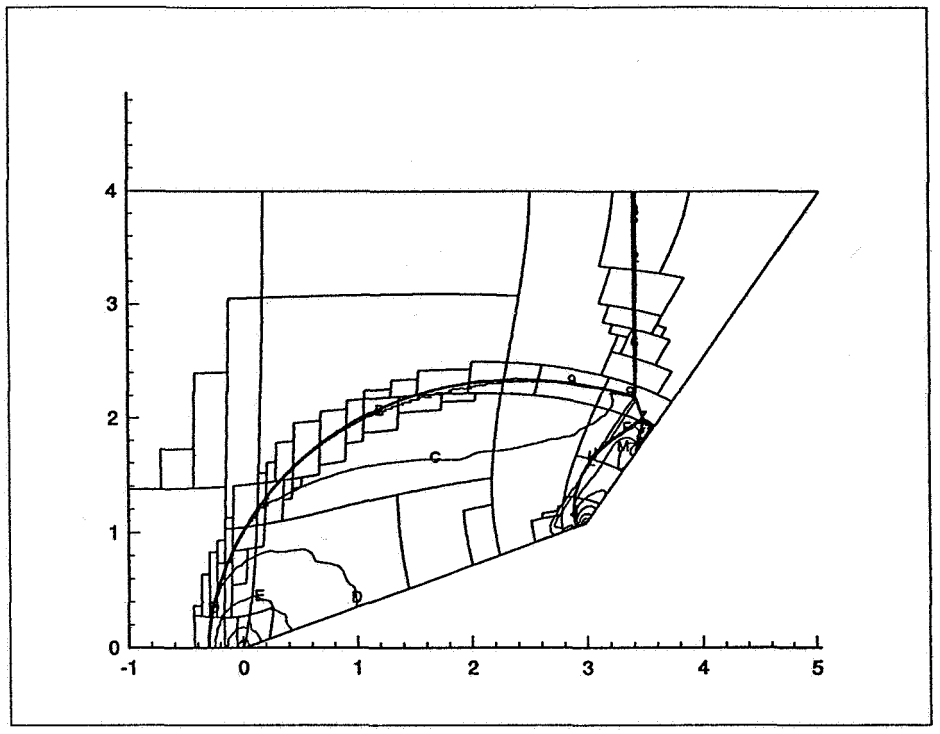


Figure 5. Shock reflection over a double wedge—Mach stem of first reflection has hit the second wedge. Contours of density; $\rho_1 = 1.35$, $\Delta\rho = 0.2$ (boxes indicate boundaries of blocks on the refined mesh levels).

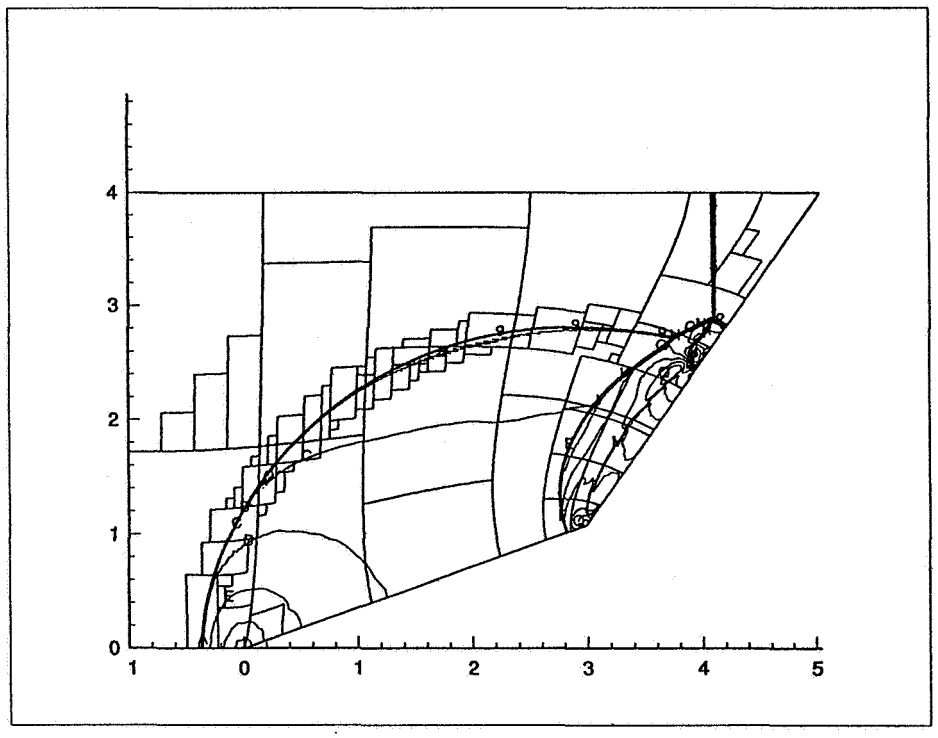


Figure 6. Shock reflection over a double wedge—triple point of second reflection has overtaken that of the first. Contours of density; $\rho_1 = 1.35$, $\Delta\rho = 0.2$ (boxes indicate boundaries of blocks on the refined mesh levels).

OMIT TO END

PANEL DISCUSSION

Panelists: James L. Thomas, NASA Langley Research Center (moderator)
Steven Allmaras, Boeing Commercial Airplane Company
Stuart Connell, General Electric
Philip Roe, University of Michigan
Venkat Venkatakrishnan, ICASE, NASA Langley Research Center

Jim Thomas: Welcome to the panel discussion on adaptive grids. Each of the panelists will comment on what they have heard over the past several days and make some general observations on the future of adaptive grid technology. We haven't rehearsed or gotten consensus from the panel, so you may well see some controversial or disparate viewpoints. We do encourage the audience to participate, and we hope there's some good interaction.

In summary, there were 66 attendees; we had a 2-1/2 day format with 12 invited papers, 10 contributed papers, and a panel discussion. We asked people to focus on a set of benchmark cases (cases 1-6), and I have a summary of the contributions made for those test cases.

Case 1 was the NACA 0012, Mach 0.85 inviscid calculation. There were six people that computed that, and they generally agreed that it was an easy case to compute. Case 2 was the NACA 0012 at Mach 0.95. There were nine people that computed that, and it was a very discriminating test case. You could get the lift and drag correctly, but it's very difficult to get the downstream position of the terminating normal shock; it's a very discriminating test case for the use of adaptive-grid methods. For the four-element inviscid airfoil, there were six calculations made. For the three-element viscous airfoil, there were only two calculations, and one of them was a nonadaptive calculation. The Jameson airfoil, which is the airfoil that exhibits a nonunique solution with the Euler equations, was a case where we really wanted to see the power of adaptive grid methods, i.e., show beyond a shadow of a doubt that you were getting solutions to the partial differential equations (PDE's) for this non-uniqueness. We only had two contributions in that area, and we really couldn't say from the results that were presented at the workshop if the non-uniqueness is there or if, in fact, it might go away as you refine the grid. It remains a very challenging test case for the community. There were six calculations made for the time-dependent double wedge. When we initially set these test cases, I thought almost no one would do the time-dependent double-wedge calculations, but, in fact, the adaptive mesh refinement (AMR) community stepped up to that. There was a very good consensus from the solutions presented for this last case.

In terms of the methods used, I speculated beforehand that it was going to be dominated by unstructured-grid methods, but that didn't happen. There were five Cartesian-grid methods, five block-structured methods, seven unstructured-grid methods, and three hybrid-grid schemes. These latter hybrid schemes are an emerging class of methods developed to attack some of the high Reynolds number problems.

In terms of the payoff areas for adaptive grids, in most problems for which CFD can play a role, adaptive grid methods can prove useful. These include 3-D simulations, solutions over general geometries, time-dependent simulations, high-lift systems development, and drag reduction. We're now computing drag to within 1/2 count from CFD methods at cruise conditions at supersonic speeds. We can't do the same thing at subsonic speeds, but we're putting together optimization methods to actually reduce drag with an optimization scheme; this requires that you compute drag accurately. We can do that near cruise conditions but we can't do that at all design conditions. Sonic boom prediction and shaping is a very natural area for adaptive grid methods. Underhood cooling and external aerodynamics are two automotive applications. Some of the automotive underhood cooling applications are low Reynolds number simulations, and some of the methods we've seen, in terms of being turn-key operations, apply. Other areas are maneuvering aircraft, off-design assessment, and wake vortex hazard prediction and minimization. There was one paper presented at the workshop that enriched the resolution of the vortical wakes as they come off the leading and trailing edges, and a substantial improvement was made. Still more areas are turbomachinery, helicopter analysis, and combustion design. There's a lot of payoff areas, and it's somewhat paradoxical to me that's there are so many areas but we really have not seen adaptive grid methods being used in the industrial environment.

Ken Powell: The payoff is going to be a long way off in some of those areas. For example, the automotive underhood cooling area requires so much effort resolving the geometry that there's nothing left for any kind of adaptation theory.

Jim Thomas: That's a good comment. When Clinton was trying to get elected, there was a poster in the campaign offices that read, "It's the economy, stupid." Here, "It's grid generation, stupid." Adaption is something put on top of grid generation, and grid generation is a big issue right now for complex geometries.

One example for which we will make great strides in the near future is the multi-element airfoil, with a main element, a slat, and a flap. We had a high-lift workshop last summer where a number of contributors focused on one case and compared with experimental data. Velocity profiles for two stations on the leading edge of the flap and the trailing edge of the main element were

compared with experimental data. Calculations which Kyle Anderson has done with a coarse, a medium, and a fine mesh indicate you can't get sufficient grid refinement for this very difficult case with practical resource expenditures. We would like to assess turbulence models for these kinds of flows because we know resolution of the slat wake and the main element wake has first-order influence on the performance. But, in fact, with traditional grid refinement methods, we're unable to get grid convergence; we don't know whether the prediction error is due to turbulence modeling or to the numerics. Until we resolve the numerics, we won't be able to improve the turbulence model in order to predict drag and lift of this complex configuration. This is a challenging test case for the adaptive-grid community to step up to and actually show grid refinement; experimental data exist which are of high quality and were taken in the Langley Low Turbulence Pressure Tunnel at very high Reynolds numbers.

In terms of adaptive grid methods, we'd like to be able to use adaptive grid methods as a black box for the practical engineer -- to input the problem statement and either an error bound or a dollar figure for the computer account/workforce. We'd like to output, after you go through this black box, the physical solution. If we input the error bounds, we'd like to know the cost of the solution; or if we input how much money we have to spend, we'd like to know the error. We're a long way from that in the adaptive grid work that we've seen so far. Where we need to go in terms of making this happen is to look at the details of this black box. This was what the workshop was all about here.

Jim Thomas: I agree with that. The black box elements include grid generation, CFD solution, error assessment, and remeshing. All the methods we've seen fall into this general framework. There are two major deficiencies that I see: one is grid generation and the second is error assessment.

Now I'll give some general impressions from the workshop. If you think back to the time 10 years ago when Marsha Berger showed her adaptive solution for the drag of a NACA 0012 airfoil, there has been a lot of development. There have been advances in grid generation, flow solvers, and computational infrastructures, i.e., the mechanics of getting the job done. We've seen advances in block-structured and Cartesian grids, unstructured grids, and also hybrid grids; no one technique seems to be falling by the wayside and there seems to be a niche for all the techniques. One of the things I look for in a workshop is what can I really stop doing in order to concentrate on things I know should be done. But, from this workshop, I can't really say there's one technique that doesn't look promising. For instance, Ken Powell showed this morning that the Cartesian adaptive mesh refinement or unstructured grids are the methods of choice for Euler calculations. The real issue is high Reynolds number viscous flow

calculations. But, you can take these Cartesian or unstructured-grid approaches and apply them with wall functions so that you need to resolve only the outer part of the boundary layer. The stretching requirements are much less, and you thereby extend them to viscous effects. Ken also mentioned coupling them with integral boundary layers.

The conservative mesh alignment schemes that Trepanier and Van Rosendale showed are very useful, in that they don't cluster so many points near the shock. The real application for those are in sonic boom minimization and that application is 3-D; that's yet to be demonstrated. After listening to all the 2-1/2 days, a general, and also a realizable, method, i.e., realizable with the finite computer resources that we have for high Reynolds number large-scale computations is really not apparent. What we're going to see in the future are hybrid schemes, i.e., advancing layer/advancing front type schemes, which alleviate the anisotropic problem near the wall. You can resolve the boundary layer near the wall because you know which direction to cluster. However, if you have a separated shear layer or a wake that cuts across the flow field, (for example, a slat wake), you don't know where that slat wake is located. To resolve this shear layer at high Reynolds numbers is really going to require a feature recognition approach. You'll have to recognize that something's there and adapt to it. For instance, on the multi-element airfoil, there have been some calculations where we adapted to the wake and put an anisotropic grid in the wake region and wake. There was a tremendous difference in the results for the isotropic grid and the anisotropic grid for that case.

There was some talk of getting around the high Reynolds problem by using nonconservative schemes. I just have one reservation in that respect in that errors due to conservation lie in wait. A number of years ago we computed one of the test cases of this conference–Case 1–with patched grids. Because of certain technical difficulties associated with resolution of the boundary layers at high Reynolds numbers, we elected to go with a nonconservative scheme. We ran about 100 cases where we moved the patched-grid interface location near the shock; 99 of those cases worked fine. However, for one case, the shock popped out into an incorrect location. That's why I said errors in conservation lie in wait. We've also run a lot of blunt body calculations for which the nonconservative patching scheme works fine. We're still using nonconservative patching, but we're always on the lookout for possible errors.

In terms of error estimation, there's been little development in the estimation methods since the work of Marsha Berger in 1984 where she basically advocated Richardson extrapolation. Two major contributions have been made since that time. One is where Gary Warren and Kyle Anderson looked at the adaptation problem and showed that the "emperor has no clothes," in that you could actually get to the wrong result if you adapt to

undivided differences. If you adapt to an undivided difference, you're going to have a nicely defined shock, but the error in the smooth region doesn't go away. The computer graphics from the solution will look very nice, but if you didn't start out with the shock in the right place, it won't end up in the right place. For the Mach 0.95 case, some of the calculations at the workshop are in error because they're not resolving the supersonic expansions over the airfoil, although there are a lot of points in the shock. There's no reason to really resolve these shocks to that degree of detail, so there's actually an inefficiency in terms of global mesh refinement. We know from Lax's theorem that we can actually resolve discontinuities correctly without putting an excessive number of points in there, because we get them from enforcing conservation. The other major contribution in error estimation is the element residual method of Kim Bey and Tinsley Oden. The theory is not complete, but some of the ideas for the estimation of the error a priori and a posteriori need to be extended into the nonlinear and nonelliptic cases.

The efficiency of the adaption process compared to uniform grid refinement is unclear. You can actually lose out in terms of the calculation because you're putting points near shocks where you don't necessarily have to have them. The time-dependent AMR calculations are a counterexample because they're enabling technologies; you really can't do them except with adaptive grids.

In summary, we've made great strides, but the acceptance of the adaptive grid schemes in an industrial environment is virtually nonexistent. There are a few exceptions, but we haven't gotten to the point where we can turn this technology over to a practicing engineer and say that, given a dollar amount or an error tolerance, we'll tell you the solution and either the associated error or the dollar cost of the calculation.

Steve Allmaras: When Jim asked me to be on the panel, he wanted me to say a few words on lessons learned from TRANAIR. So, I am going to split up things; I'm going to first talk about TRANAIR and then make comments on the workshop. I think I'm uniquely qualified to talk about TRANAIR. I've been sitting next to the development group for 5-1/2 years, but I've managed to avoid ever working on or running the TRANAIR code. Before I came out, I got together with Forrestor Johnson and talked about what they're doing. I'm going to give five major lessons learned from TRANAIR. For those of you not familiar with TRANAIR, it's a full potential solver using Cartesian grids, mainly used for complex geometries, i.e., modern airline configurations, nacelle integration problems, jet fighters, and wind tunnel interference prediction.

The number one point is that adaptation reduces the grid generation burden on the user. Now, this is an obvious point, but this is the real utility or usefulness of adaptation in an engineering environment. This is something we should keep in mind as we develop adaptation methods for Navier-

Stokes. Currently, the real bottleneck in Navier-Stokes analyses of complex geometries is the grid generation issue. There's a lot of fruitful territory to be explored with adaptation methods.

The second is you must give the user control over the adaptation. You need to do this in a relatively straightforward, easy manner. To illustrate this, an early TRANAIR experiment in adaptation involved the simulation of a half model in a wind tunnel. They ran it and looked at the wing pressure; added some more grid, looked at the wing pressure. The solution didn't change much, so they added more grid and reran. The wing pressures still didn't change much, because their adaptation was resolving a wake singularity, or wake cut singularity, all the way down the length of the tunnel. This illustrates a problem, because for the most part you really don't care about this singularity, since you really want to find out what's happening on the wing. There are certain cases for Euler and Navier-Stokes where you actually want to resolve a wing tip or flap vortex and see where it impinges on components downstream (e.g., the tail). So now you have to be able to tell the method, "Yes, I want to resolve this feature; no, I don't want to resolve that feature." The way that TRANAIR does it is with something they call zones of interest, or disinterest. You input a hexahedron in space and at each of the vertices you tell it the minimum and maximum grid spacing you're going to allow. You can also put in certain quantities to emphasize refinement in a particular region. For instance, you may want to really look at what your nacelle installation is doing, but you don't really care about the body. You can tell it to resolve here, but don't resolve there.

The third point regards error estimators. They use an undivided maximum jump in velocities between cells. They did try other quantities, but it turns out that individual velocity differences actually work better for them. It tends to better resolve surface features and also emphasizes large and small scales equally in the flow field. Much of the wing design, for example, is concerned with sweating the details. You need adaptation that is going to resolve those types of things, but you also need some sort of control because, otherwise, the grid will all be drawn into the leading edges. For wing body nacelle installation, the error estimator without any type of limiting results in all the adaptation being drawn into the leading edge regions. With zones of interest, or disinterest, the smallest grid cell in the domain can be reduced to a certain level. The grid is resolved down to that level in these regions then put elsewhere in the flow field. By doing this, they're able to capture a very weak oblique shock in the outer part of the wing. Basically, it's a limit on how fine you want to resolve the grid.

The fourth point concerns latent features, i.e., features that only show up when you have a sufficiently fine grid. A good example of this is the oblique shock on the outer part of the wing. TRANAIR uses two good ways of capturing latent features. The first is to improve the numerics of the flow

solver, and the second is to limit the amount of grid that is attracted to sharp features, like normal shocks, on coarse initial grids. As an example of the improved numerics, a 2-D case of a supposedly shock-free Korn airfoil used first-order upwinding in supersonic zones. The first-order scheme picked up one shock. Second-order upwinding in supersonic zones picked up two shocks. The true solution, if you have infinite resolution, is a double shock. Definitely, improved numerics helped dramatically. The other way to resolve these latent features is limiting. This is part of an overall conservative adaptation strategy. TRANAIR was forced to use adaptation because they have a Cartesian grid solver. There is relatively little rigorous theory out there as to how you should proceed with adaptation. So, in an engineering production environment, it has to be robust. If it's not robust, it's not going to be used. They chose this strategy to get the code into use by the designers and to gain experience for developing a better strategy. Dr. Johnson commented that the last 5 years has been a continuous learning experience for them in this area. Typically they only refine about 20 percent of the cells going from one grid level to the next; at the same time, they derefine about 40 percent of the cells. This results in approximately twice the number of cells each time you adapt. They also do grids where they don't actually change the total number of cells, they just redistribute things. In a typical solution, you might have eight or ten adaptation levels and maybe two of those will be equal-distribution steps. The total time spent is typically on the order of two or three times the CPU time it takes to solve the fine grid. If you went to a much more aggressive adaptation strategy, you could probably cut that down to perhaps 25 percent overhead.

Are there any questions or comments before I head on to my workshop comments?

Jerry South: Does TRANAIR use the conservative version of the potential equation, or nonconservative?

Steve Allmaras: Conservative.

Jerry South: Does it also include an interacting strip boundary layer?

Steve Allmaras: In the last couple of years, they have begun including an integral boundary-layer formulation.

Jerry South: What do you do about wakes? You showed the wind-tunnel turning vanes. For the 2-D potential, every time you have a lifting surface, you have a jump in the potential, but in the 3-D case, it's not real clear how well established the theory is.

Steve Allmaras: They need to fix the wakes a priori. They don't move with the solution, but doublets in the wake approximate where the actual wake

location should be. There is a jump and also other singularities that they add into the wake cut.

Jerry South: I have a comment of theoretical significance. You mentioned the shock-free Korn airfoil and that when you do a lower order method on a given grid you get a single shock; then you do a higher order method and you get a double shock. It would be interesting for you to go back to the lower order method and see whether the sonic line doesn't have a reflex in it somewhere in the neighborhood of where the second shock appears. There's an old theorem by Nikolsky and Taganov that as you traverse the sonic line in a 2-D flow, the velocity vector has a monotonically-turning tangent. You can conclude that you can't have a reflex in the sonic line. So, if there is one, maybe you could say that you need to do more refinement because there's obviously a shock in the flow down underneath.

Steve Allmaras: One point I want to make on the shock-free Korn airfoil is that if TRANAIR continues to adapt on the first-order method they will eventually get the double shock. It doesn't miss it, it just takes many more grid levels to actually resolve that.

Onto the workshop. There were a lot of pretty pictures of moving shocks and well resolved flows and grids, but very little hard data and head-to-head comparison, particularly on the test cases. I didn't keep an accurate count, but it seemed that at least half the papers showed one or fewer of the test cases. It's really hard to do the head-to-head comparisons when that occurs. As far as participation, there was a good variety of cases attempted.

A general comment is that there are a wide variety of strategies taken, but no real clear winners, partly because there's no real comparison between them. I saw an overemphasis on shock adaptation. There are other features in flow fields that are equally important. The area of viscous adaptation is a hard area; lots of fruitful research will come out of that. As an employee of a commercial airplane group, we have limited need for methods that do really well at resolving strong shocks. If you have a strong shock in your design, you're in trouble and you need to start over again. You don't need to accurately know that it's a strong shock. I saw a lot of excessive grid refinement. The real purpose of adaptation is not to show that you can get the densest grid. There is a tradeoff that should be shown; if I add more grid, am I getting a better answer? That type of data I didn't see here, and I would really like to see it, because it's an economic issue. Also, how well can your method do at finding features on coarse grids? That really has an engineering applicability, because we don't want to expend a lot of resources finding our solutions.

In regards to positive things, it's nice to see that there are packages out there that are available and useful. One or two papers basically picked technology

off the shelf and quickly got it to work; that's nice to see. The final observation is that, when I calculate a flow simulation, I want to be able, with a little extra effort, to get some bound on the numbers I'm calculating. I've got drag to four digits of accuracy, for example, plus or minus. I'd like to see more work done in estimating solution errors, and in getting these bounds cheaply. It's a deficiency in CFD as a whole right now. Its applicability is not just to adaptation, but flow simulation as well, in general. Any comments or questions?

Richard Barnwell: In regards to the extensive grid refinement, do you have an algorithm or a method you can suggest for judging grid economy?

Steve Allmaras: No, but I think Gary Warren alluded to it in his first talk Monday morning. You look at the error on one scale versus the amount of grid or resource expended to get that on the second scale. That's the type of measure one needs. Plus, it shows that as you're adding more grid, you're getting a better answer.

Richard Barnwell: The second question is that I realize you don't want to have to calculate strong shocks, but would industry see more usefulness in a shock alignment scheme, as opposed to shock capturing, if they were available?

Steve Allmaras: I like the work that I saw on shock fitting. It's nice to see good answers on coarse grids. I think that has a lot of potential in capturing features like viscous shear layers where you can align the grid and then do more or less directional adaptation. I have a lot of enthusiasm for that type of work.

Richard Barnwell: It could probably be used for wakes as well.

Steve Allmaras: Yes, definitely.

Kyle Anderson: My comment regards the solution accuracy. Normally you'd proceed with grid convergence and plot some answer. Right now the only way you're going to assess the error is with normal global grid convergence. It's pretty clear that you can't trust the adaption by itself because it will level off to some answer that is not necessarily correct.

Steve Allmaras: Correct. The way to do the error assessment is to do the grid refinement and grid redistribution studies. We just don't have the time or resources in industry. I know on a routine basis in research you can't do it either. It's an issue.

Stuart Connell: Well, I arrived here on Sunday. Jim Thomas had invited me to make a presentation, and I had no viewgraphs, but with judicious use of e-

mail, fax, and express mail, I managed to put something together. What I want to talk about is this whole adaptive unstructured meshing area -- the view from industry and where we see it going. First, I will say where I fit into the picture, what we're doing at General Electric, what needs we see, and the strategy we're using to meet these needs. Obviously, this is going to be largely the work we're doing. We believe this is the best way to meet our needs.

The work I'm involved with is primarily steady flows. I'm in the business of developing codes for use by designers. This is primarily in the aircraft engine group and the power generation group. They provide funding to us, so we develop a very close relationship with them. If we don't do what they want, they don't fund us, and we don't eat. From that point of view, we do what they need. The applications we look at are typically turbines, compressors, exhaust, and really any geometry in the engine which gas flows through. An example of some of the codes we've been developing in 2-D, is an unstructured mesh viscous code that has a 1-D refinement in the quadrilateral region around the blade. This code is in regular production use. Designers use this all the time. Solutions are generated routinely for a variety of exit Mach numbers, which represent the decrease in the exit static pressure. Quite different shock patterns are predicted, and the whole thing works quite nicely.

On to the needs of designers. What they need is the shortest possible lapsed time for a solution. Design times are getting shorter and shorter. We've got to make better and better designs, so we need to make more and more design iterations. I also see the need for more complex 3-D geometries, viscous solutions, and high quality robust codes. If it doesn't work the first time, they're apt to throw it away and to not use it again. What are we doing in this area? What codes do we have? A variety really. In the 2-D arena, we've got structured Euler/Navier-Stokes codes and the unstructured adaptive Euler/Navier Stokes code I just mentioned. As I said, these are all used routinely in the design process. In 3-D, we have structured Euler/Navier-Stokes codes. Some of these are multiblock codes that use CFL3D, and are very heavily used.

Designs using these codes are becoming close to optimal. There's not much more to gain. What are we doing to improve things, to make better engines, better designs? One big area is meshing. If you want to mesh a complex configuration with current technology, spending man months is not unusual. You can take 6-9 months to mesh something, and by the time you've meshed it, you've forgotten what the problem was. They also want more and more geometric fidelity. They want to see the effects of smaller items in the geometry--should they be concerned about this, should they be designing to allow for this, that kind of thing. I really believe that mesh generation is a pacing item. The solvers are there; the post processors are there. We need to accelerate this area. The 2-D adaptive unstructured code is

very successful and widely used. That's really what the designers want. It can mesh arbitrary 2-D geometries. The meshing is automatic; you enter two or three parameters and hit the return key and it comes back with a mesh. The solver is adaptive and user independent. Hopefully, if you give the same geometries to two different designers, you'll get the same answer. With the previous code, perhaps you wouldn't be too sure of that. What we're trying to do is extend the success of 2-D to 3-D. To do that, you need to mesh directly from a CAD file. You can't ask a designer to convert his CAD geometry into the geometry you need for your mesh generator. You need to go straight from the CAD file. Again, it's going to be automatic and mustn't have too much user intervention. As little user intervention as possible. You need viscous meshing, high-aspect-ratio cells in boundary layers, and obviously adaptive cells.

I'll talk a little bit now about the 3-D meshing I've been involved with to make high-aspect-ratio cells in boundary layers. Out of the available technologies, we believe surface triangulation, followed by inflation, to provide a prismatic mesh, provides the best optimization. We use a technique very similar to that of Kallendaris, but developed independently. We've taken the Morgan/Peraire advancing front mesh generator and developed an in-house inflation code to link the whole thing to a CAD system. To give a schematic of how we generate these meshes, we start off with a surface triangulation, then we select the surfaces we want to inflate. We then inflate that, sliding the nodes on the surface up, to provide a nice prismatic mesh. We then triangulate the remaining surfaces and fill the interior with tetrahedra using the advancing front procedure. We have applied it to a large variety of test cases including a wing pylon nacelle, and an HSCT mixing nozzle. The other reason for doing this prismatic mesh is that it opens the door for 1-D refinement and 1-D multigrid. If you have a boundary layer, you can refine in one direction. This makes very, very efficient use of points.

In summary, in general, adaptive unstructured meshing holds the most promise. You can look at arbitrary geometries and mesh them automatically. Also, the real world is 3-D, is viscous, and it contains complex geometries. Any solver that is used must address all three issues. Also, the end user, our designers, needs quality solutions as quickly as possible. That really means that these four items here, the meshing, solving, refinement, and post processing, are all done efficiently and fast. There's no point having the world's fastest solver if it takes you 6 months to make a mesh. You've got to accelerate them all. With that, I'm done, and I'd be happy to answer any questions.

Jerry South: Stuart, are you using multigrid with your prismatic approach?

Stuart Connell: Not yet, but we're planning to do that.

Jerry South: What kind of method are you using to achieve convergence now?

Stuart Connell: It's Runge-Kutta time stepping. We got the multigrid working for the inviscid code on a tetrahedral mesh and we want to extend that to the mixed mesh.

Dimitri Mavriplis: Do users use the adaption mode frequently?

Stuart Connell: Yes, by default. We've had the default parameter define a certain number of adaption steps or iterations. The code, in its default mode of running, runs to convergence on the initial mesh, then fires up with the adapter turned on. So, they always use adaption when they run it.

Jerry South: What criteria are you using to refine the mesh?

Stuart Connell: It's fairly crude–undivided gradients. We've done lots of evaluations versus experimental data. For the geometries we're looking at, that seems more than adequate. I recognize there are problems where that is probably not the best thing to do.

Phil Roe: I haven't actually prepared any viewgraphs, so maybe that will help speed things up a little bit. For me, this is a bit of an anniversary because it was pretty much 10 years ago that I made the move into academic life and started teaching people about CFD, as well as actually practicing the art. In trying to be an honest teacher, I always try to tell people where the demarcation line is between problems that CFD could solve and problems that CFD could not solve. At that time, I always used to stress the fact that CFD techniques had big problems with anything where there was a large range of length or time scales involved. At that time, you'd say these are problems for which CFD will really not be able to provide an answer. This is the reason we still need wind tunnels. Ten years later, Ken Powell has shown solutions which resolve something like 5-1/2 orders of magnitude in scale. There is a sense where the first statement is obviously no longer true. I have had to modify the way I tell people about the subject. Of course, the only reason he can successfully resolve those scales is that, in that particular flow, the solution looks simple at any scale at which you look. So, you could not conclude that we can immediately do direct simulation of high Reynolds number turbulence. That's an entirely different problem. So, I will still have to tell people that there are problems that CFD cannot attack, but the boundaries are moving all the time. I'm really impressed by how far the boundaries have moved over the past decade.

To give any kind of general survey about what's possible and what's not possible, you have to ask the question, "Why are we doing this simulation?"

The reason why you're doing the simulation varies very much from one example to another. The non-unique airfoil solutions are an instance where you're doing the simulation in order to establish some kind of ultimate truth. At the basis of this is academic curiosity about whether the Euler equations do or do not have unique solutions. In order to answer that, you're going to have to satisfy the PDE's as well as you can. In the present state of the art, you may still not be sure what the answer is; you're going to need all the technical help you can get to solve those problems as accurately as possible.

This is not the case in every application. You want, very often, in an engineering context, simply to find out some information about the flows.

The fish-tail shock is a good example. That's been described as a very discriminating case because it's very tough to get that rear standoff distance unless you resolve all the details. But why did you want to know that? If you merely wanted to know the surface pressure distribution, you could have gotten a perfectly adequate answer without resolving that detail. If, on the other hand, that tail shock is going to interfere in some way with some other component of the vehicle, and you really want to know what it's doing, the answer is that this flow is already a mess. This is not a reliable component of any engineering design, because it is so sensitive to the details of what is going on. So, all you want to know is some rough indication of that, so you can eliminate that from amongst the designs you're considering.

What you want to use adaptivity for, in that case, is to be able to get some useful information on a coarse grid, and to be able to get that fast enough, to not waste any more time. This raises interesting questions about whether you can give some kind of general-purpose adaptivity criteria. What norm are you going to try to optimize? For any particular design objective, you can perhaps formulate a norm that would reflect your interest. But that might not be the best way of doing it, especially given the great difficulty of taking even the simplest norms and converting them to some kind of automatic strategy. For most of the applications, the man is going to remain in the loop for quite a long time, and I find myself really disagreeing with Stuart on this note. He wants procedures that are as automatic as possible. I can understand why he says that in certain contexts, but I don't really think it's a universal conclusion. I was talking with Jerry South yesterday, and he made the point then, as he's repeated now, that the ultimate objective of adaptive algorithms, and indeed computational fluid dynamics as a whole, is to turn itself into a black box. In other words, the job of all algorithm developers is to make themselves redundant.

In thinking over that, I began to see some possibilities for a rather nice Raymond Chandleresque novel on this theme. It begins with an ICASE researcher who has been observed to be behaving rather secretively and not communicating with his colleagues. I don't know if this brings anybody in

particular to mind, but one or two of his colleagues become aware that this guy was actually very close to developing the ultimate CFD code that will cope with all Mach numbers, all Reynolds numbers, and all geometries. The week after that, his body, wearing concrete boots, is dredged out from the marina behind his hotel. I thought I really had the beginnings of a very promising scenario here, so last night, I put through a phone call to the west coast and discussed with MGM the possibility of the movie rights for this. It was a very interesting conversation actually, because the studio executive that I spoke with had the same list that Jim had earlier. The same list of nine items of potential benefit for adaptive grid generation. She put to me, "Could I really imagine that there's going to be one single CFD code that would be the method of choice both for under-the-hood cooling flows and for sonic booms". She convinced me that I really had no future in the entertainment business because I was unable to think of a convincing script.

The role of adaptivity, really the role of CFD in general, for a long time before it eventually collapses into a black box that can be put away and never be opened again, is to push envelopes. All of those items on that list were items where we want to push the envelope of the capability. I don't think that pushing envelopes is, or ever can be really, a blackbox style activity. There's going to be a lot of roles for the individually tailored codes, individually tailored adaptation schemes and individual systems, if the man is in the loop. One of the determining factors about what methods get used will be user friendliness. Is this system actually easy to understand? If you're going to sit there and drive it, as computers get faster and faster, we will be able to take these complicated problems, and in real time as the solution evolves, we can interfere with what we want to know; ask for different quantities, and make suggestions about how the code may give us those quantities more efficiently. That's really how I see the next ten years going.

As regards actual methods, it seems that there's a range of methods available now that are sufficiently robust to be able to support this kind of thing. It isn't very crucial what you do. The good point has been made that unless you can see the interesting features on the coarse grid, then your adaptation really has nothing to get a grip on. You're going to miss those for all time, so your underlying algorithm has got to be a good one. You can't solely rely on the adaptation to get you out of trouble.

The big issue, which I really have very little constructive ideas about at the moment, is the issue of viscous adaptation. You may know enough about the flow to be able to say in advance where the boundary layers and wakes are going to be. For separated flows, and for vortices that are shed into the flow, things move around and have to be kept track of all the time, and you've no real idea where they are to begin with. Nevertheless, there are localized regions that need refinement and other regions that don't need refinement. We do need more ideas on how to deal with that kind of thing efficiently and

that is the method developer's challenge for the next ten years, from my perspective. Thank you.

John Van Rosendale: It seems to me that making things blackbox is not equivalent with putting researchers out of business.

Phil Roe: I'm definitely not against blackboxes. I am very grateful that I can do CFD without having to know how to design a circuit board or write a compiler. You certainly don't want to have to tweak things like artificial viscosities. In that sense, CFD should definitely turn itself into a blackbox. It shouldn't turn itself into a blackbox in the sense that you detach yourself too much from the things you want to be concerned with. I don't know whether I'm expressing that very well, but blackboxes are very good in their place. However, if essentially you're pushing the envelope of something which is difficult because it is very difficult physics, or difficult because it is really at the limit of what you can squeeze onto your machine just because of geometric complexity, then you're always going to have to be involved with the detailed knowledge of your problem.

John Van Rosendale: What about the feature in TRANAIR where you specify in different regions the mesh size maximum and minimum?

Phil Roe: What I would see as optimum, is the opportunity to interact with the solution as it evolves. Change your mind about your initial parameters long before you see a final converged solution. You say, "This is not working out; I asked the wrong questions; I want to change my mind." No, you're going to need things which are to some extent blackboxes so you can use them in the same way you use a compiler. But, you're not just going to pose a question, and then sit back, and two hours later see the answer. That's not how you want to operate. That's not how you get the maximum use out of it. The user should not be distracted from the things that he's interested in. The way the solution is turning out is precisely what he's interested in. What he needs is a good way to control that, so that he doesn't have to understand all the fancy details. He doesn't have to get down to the coding level, but he needs to understand in broad terms how the thing works, and have some things that he can control that allows him to track something without getting too far away from it. I think that's where adaptivity allows you to get reasonable information on the coarse grid.

Jim Thomas: That's a good point. It has to do with the available computer resources that you have. They are not infinite. So you might have a scheme that could do the problem, but you really can't afford it. It's like they had an architect design a house for you that was a 5,000 sq. ft. house. He gives you the price and you say, well, I can't afford it. But, I do understand that I'd like to have two bedrooms, and you might leave the back half of the house

vacant; then what would that cost? I don't know how to build a house, but I can talk to the architect about what I would like it to have.

Jerry South: I think in light of Phil's comments about his movie plot, we're going to increase security around Dimitri's office.

Venkat Venkatakrishnan: I just have a few observations about the workshop. I'm not a practitioner of adaptive grids, but to an extent almost anybody who does CFD is a practitioner when you start computing viscous flows. Because you do have to worry about scales in the boundary layer and you do use the appropriate spacings near the wall to resolve whatever phenomenon interests you. I see in the workshop itself, two classes of adaptive grids. One class attempts to solve problems better, and this was well addressed by Gary Warren. That's why the benchmark problems were carefully chosen with "exact solutions." I agree completely with Warren that you should use uniform refinement once the errors in the coarse regions are comparable to the errors in the fine regions. You can't keep refining just near the shocks or other discontinuities, because the errors that you have introduced elsewhere will kill you.

The second class I would call enabling technologies, whose purpose is to solve problems that just cannot be addressed otherwise. Because, only by using adaptive grids can you even capture the scales of interest, e.g., the unsteady flow phenomena shown at this workshop with complicated interactions of shocks and detonation waves. When you go to three dimensions, you'll do well to employ adaptation, just to be able to resolve things to the level you need.

Onto the content of the workshop. Regarding AMR, it's definitely work of a very high degree of sophistication that is being routinely used to solve difficult time-dependent flows. As Steve Allmaras has mentioned, it looks like it's very well packaged, supported, and disseminated. There's no better way to get followers of your method of thinking. As Ken Powell mentioned, there are some problems obviously when you start dealing with viscous flows and there just seems to be no easy way around them for computing high Reynolds number flows of interest. For the inviscid methodology, I think the Cartesian approach is extremely attractive.

Professor McRae showed some of the best results I've seen in just warping structured grids. The method does yield improvement in solutions, and it's a lot simpler. In the case of unstructured grids, it looks like many of the people showed good results using standard criteria and showed improvements compared to unadapted cases. Bookkeeping issues are somewhat comparable to the AMR, but are not daunting; most people are willing to expend the effort and time required to master these challenges. Mesh warping showed some excellent results, but I would have liked to have seen more careful

studies of the test problems using this particular idea. I think it's a very popular idea, and I was very impressed when I first saw it.

A very strong case has been made for using adaptive grids, especially by the AMR community. A combination of error estimates and heuristic criteria appears to be sufficient. Engineers and applied mathematicians have shown a real willingness to learn programming and other computer science issues.

The challenges are immense and you have to invest a lot of time and effort in the infrastructure. There are many more computer science books in my bookshelf than engineering books, and I wonder what area I got my degree in. This is before I heard Stuart's talk but apart from the AMR school of thought, adaptive grids are not used routinely, and I think there are two reasons. I think the first reason is the overpowering one. It's just a lack of trust. Just adapt and assume that you get a good solution. This lack of trust obviously needs to be overcome, and that's why I think we should pay a lot of attention to what Gary Warren described very well in his talk about the optimal relocation of points and doing test problems and nailing these issues. The bookkeeping issue definitely arises in three dimensions. This is not such a big problem because people have already invested the time, but it will probably make people think twice about getting into the area. Since there are so many people who work in the area and have software that's available and so forth, I think it's not much of an issue.

The accuracy issue with adaptive grids persists and there's definitely an inadequate framework here. Error analysis for weak solutions of hyperbolic conservation laws is lacking both a priori and a posteriori estimates. A well-known mathematician asked me a question about a posteriori error estimates. I've taken courses in hyperbolic PDE's, and we never talked about errors when we do hyperbolic PDE's. When you go to elliptic PDE's, everything is neat, you know a lot about the regularity of the solution and the global errors in terms of the higher order derivatives, solutions, and so forth. There is an inadequate framework for hyperbolic problems and the engineers cannot hope to solve this problem by themselves. We need very good mathematicians and applied mathematicians to help us in this regard. Most of them are spending their time addressing programming and computer science issues; we are missing out in the analysis part, because some of the best people who could actually contribute here are doing other things. You have to spend the time to get things working to get your framework right.

I've read some of Kim Bey's papers and she's probably one of the few people who can bridge the gap between the finite elements and the finite volume community. What she's doing is very interesting, and it will be nice to be able to extend it to the nonlinear hyperbolic conservation class.

Jim Thomas: I'd like to open up the general discussion now, and in particular I'd like to address several questions. Do you think the workshop was worthwhile? And if you think it could be improved, how? Should we keep benchmark test cases and follow-ons? We specifically did not have a 3-D benchmark test case since a lot of people wouldn't have the mechanisms in place to do a 3-D adaptation. Should we add a 3-D benchmark case, if we had a follow-on workshop? Any comments/questions?

Ken Powell: I guess I would echo some of the things Steve Allmaras said. I think the workshop was worthwhile, but I was a little disappointed. The idea behind the workshop sounded really great. The idea of picking the test cases and doing a careful study of how we're doing with adaptations seemed a great idea at the time, but was not realized well here: partly from the talks people put together, and partly because of the organization of the workshop. I'd like to see a follow-on. I'm perfectly willing to rerun some of the cases if we have some input format, and that would be a worthwhile follow-up.

Gary Warren: Ken Powell and I talked earlier about this; there really weren't that many benchmark cases run. Ken and I were talking about the possibility of an addendum in the proceedings. (Editor's note: no such addendum appeared.)

Jim Thomas: We need to work harder on the organizational side to make sure that we define what we need from the contributors beforehand. We made it pretty open-ended for this workshop, and we need to a little bit better job there.

Scott McRae: One suggestion that I would have is that referring to the real world, most of the cases are missing something, and that is viscosity. There is too much attention to adaption to the shock wave.

Jim Thomas: I agree, and there is one test case that is viscous . And that's a practical test case; people that are doing engineering calculations would like to see a grid refined solution on that.

Scott McRae: But again there should be both laminar and turbulent test cases. We need to be able to remove turbulence modeling errors from numerical errors.

Jerry South: I might add that we initially didn't have a viscous test case, but we decided that it would be a challenge to the adaptive community. Actually very few of the initial abstracts we got even addressed solving the inviscid test cases. We were a little disappointed, so we actually delayed the workshop hoping that more people would agree to do a new case. Nobody did a real adaptation on a viscous case, so I think that ought to stay in as a test case. A

three-element airfoil is a great case that can be done in reasonable computation time.

Steve Allmaras: You have to be careful about comparisons with experiments since, for viscous 2-D airfoils, wind-tunnel data and CFD are comparable in their accuracy right now.

Kyle Anderson: Even for stall?

Steve Allmaras: Especially for stalls. Because, near stall, 2-D doesn't exist. Especially in a wind-tunnel. I know of experiments where to eliminate the 3-D sidewall separation they use blowing. By turning the knob on the blowing magnitude, they get 2-D results, but can get whatever level they want. It remains 2-D across the wing section but you can basically dial in your own answer.

Gary Warren: So the adaptive grid stuff does fit in.

Steve Allmaras: Thus, you have to be careful about comparing against test data.

Tom Roberts: The other thing is that there is an exact inviscid case. Part of the purpose of that case was because there weren't many exact solutions. In terms of the issue of adaptation to the PDE's, that's a very good test case.

Gary Warren: I remember Jim Thomas making a comment several years ago, and it had to do with an experiment. He said, "Given some experimental data, a graduate student, and an adaptive grid code, you can always match the experiment."

Jerry South: I think that he didn't say it quite that way. I think he said that you can always find a coarse enough a grid to agree with a 2-D experiment.

Jim Thomas: Well, for the hypersonic ramp case, both statements apply.

Venkat Venkatakrishnan: What did you exactly hope to get out of the test cases? Did you expect people who use different adaptation criteria to distinguished one from another? It looks like everybody was doing reasonably well. Therefore, I say the answers are all about okay. You then have to look at the effort put in, the amount of work required, grid points needed, and the level of sophistication of the user.

Jim Thomas: We had talked about that when we originally set up the test problems. We wanted to assess the efficiency of the scheme. Not only accuracy, but whether you could run that scheme on a parallel computer, supercomputer, or on a desktop workstation.

Phil Roe: We also felt that the adaptive area, being the very amorphous animal that it is, would have so many parameters to try and control that it was maybe not all that appropriate on a first occasion to try and tie things down. So it really finished up being very much a free-style event. And maybe one of the things that we should discuss is how much more focused do we now feel, having had this first occasion, and how much more stringent should things be a second time around.

Gary Warren: Ken Powell's talk this morning was great. I really enjoyed the talks that showed the problems, and I think we get a lot more out of it when people show the problems; we all know that they exist. It would be a great idea for a workshop if for every good result you have, you have to show a bad result or problem. I think we'd get more out of the workshop. For instance, John vanRosendale's talk with the aligning of the grid points. Even though he didn't get the right answer on the Mach 0.95 case, that was outstanding work on the methodology behind it, and I got a lot out of it. So I don't think we should view things like winners because it's really an instructional thing.

John Van Rosendale: I agree with Venkatakrishnan's comment that as a community we're spending too much time programming. It takes six months or a year to get a scheme developed. It's only after that, that you start to think about error estimates. So, if we have a follow-on, I would suggest that part of it be devoted to tools. Things that could be given out. The AMR community has gone further with C++ packaging of their work than the rest of us.

Jim Thomas: I think that's a good suggestion. I think we were all impressed with AMR in that respect. They have adopted C++ and then put some libraries together so they could assemble codes more quickly.

David Keyes: As an outsider relatively unskilled with adaptive grids, I would strongly affirm this workshop has been very insightful and helpful. I was only here part of the time, but I've heard some good issues, especially in error estimation issues, computer science, and adaptivity. If a follow-on workshop is proposed and tools are considered it would be good to consider partitioning because adaptivity confounds every generalized attempt to do parallel composition of the grid. Doing a partitioning in the abstract without considering upstream/downstream flow dependencies is inappropriate.

Jerry South: I would suggest the addition of a 3-D test, if we do it again. I would also suggest that a good 3-D case to do is one where we can afford to do some grid refinement. The Euler equations do quite well for slender supersonic airplanes, in the cruise condition. If you have separation, you didn't design it right anyway. There are many old experiments on supersonic transport designs that are available. We could choose one of those cases.

One of the things that the adaptive grids should do quite well is the problems of the sonic boom. What people are doing now in the industry is taking the Euler equations and solving the flow of the near field around a supersonic airplane; then from the near-field pressure distribution, they're extracting a Whitham F function to extrapolate that pattern of pressure in various directions down to the ground to get a signature. It makes good engineering sense when you do that. The problem is setting up the appropriate F function. Now I wonder how many of you have looked at the pressure distribution in the near field of a supersonic airplane. When you see it, you think that there was something wrong with the wind tunnel data. You see a relatively smooth aircraft and about a half a body radius away, you measure the pressure with a probe and it's the bumpiest thing you ever saw. The bumps are from little waves produced from smooth, initial data which coalesce and eventually produce in the far-field an N-wave which is the sonic boom. I would suggest that a case like that would be a real good one for adaptive grids. Because if you don't correctly adapt to the smooth parts of the flow, then you're not going to get to the right answer. And it's a nice example for the Euler equations, and that would be a reasonable one to include if people would like to tackle that kind of a problem.

Steve Allmaras: I'd like to thank the people that set this up and put all the effort into organizing it as well as the participants who's papers were presented. I'd also like to thank John vanRosendale for hosting the reception last night.

Jim Thomas: I agree. And I'd like to thank the panel for today's panel discussion. I appreciate it very much. And I thank all the participants.

PARTICIPANTS

Steve Allmaras
Boeing Commercial Airplane Company
Mail Stop 7H-96
P. O. Box 3707
Seattle, WA 98124-2207
(206) 865-6257
sra3772@cfdd27.ca.boeing.com

Kyle Anderson
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2164
w.k.anderson@larc.nasa.gov

Gregory Ashford
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118
(313) 936-0107
ashford@engin.umich.edu

Harold Atkins
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2308
h.l.atkins@larc.nasa.gov

Richard Barnwell
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-4129
r.w.barnwell@larc.nasa.gov

Sami Bayyuk
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118
(313) 936-0107
sam@engin.umich.edu

R. Jeffrey Benko
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118
(313) 764-7478
jbenko@engin.umich.edu

Marsha Berger
Courant Institute
of Mathematical Sciences
251 Mercer Street
New York, NY 10012
(212) 998-3305
berger@cims.nyu.edu

Kim Bey
Mail Stop 395
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-1351
k.s.bey@larc.nasa.gov

Karen Bibb
Mail Stop 395
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-8005
k.l.bibb@larc.nasa.gov

Ricardo Camarero
Ecole Polytechnique de Montreal
C.P. 6079 Suce "Centre-ville"
Montreal, Quebec H3C 3A7
CANADA
(514) 340-4896
ricardo@cerca.umontreal.ca

James G. Carpenter
Department of Mechanical
and Aerospace Engineering
North Carolina State University
P. O. Box 7910
Raleigh, NC 27695-7910
(919) 515-5250
jgcarpent@eos.ncsu.edu

Mark Carpenter
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2318
carpentr@tab00.larc.nasa.gov

Eric Charlton
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118
(313) 936-0107
charlton@umich.edu

William Coirier
Mail Stop 5-11
NASA Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135
(216) 433-5764
coirier@marmite.lerc.nasa.gov

Phillip Colella
Department of Mechanical Engineering
University of California, Berkeley
6189 Etcheverry
Berkeley, CA 94720
(510) 423-6393
colella@watt.berkeley.edu

Stuart Connell
General Electric CRD
Room K1
Schenectady, NY 12301
(518) 387-5848
connell@crd.ge.com

Darren DeZeeuw
Department of Atmospheric,
Oceanic and Space Sciences
The University of Michigan
Ann Arbor, MI 48109-2143
(313) 763-6224
darrens@engin.umich.edu

Comer Duncan
Department of Physics and Astronomy
Bowling Green State University
Bowling Green, OH 43403
(419) 372-8108
gcd@riemann.bgsu.edu

Lori Freitag
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439-4844
(708) 252-7246
freitag@mcs.anl.gov

Neil Frink
Mail Stop 361
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2864
frink@aero00.larc.nasa.gov

Richard Gaffney
Analytical Services and Materials, Inc.
Mail Stop 168
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-7872
r.l.gaffney@larc.nasa.gov

Martin Galle
DLR Institute of Design Aerodynamics
Lilienthalplatz 7
Braunschweig 38108
GERMANY
011 49 531295 2470
ea2n@beacdc1.ea.bs.dlv.de

Philip Hughes
Department of Astronomy
University of Michigan
Ann Arbor, MI 48109-1090
(313) 764-3430
hughes@astro.lsa.umich.edu

Dongming Hwang
MCNC
North Carolina Supercomputing Center
P. O. Box 12889
Research Triangle Park, NC 27709-2889
(919) 248-9250
dongming@mcnc.org

Clint Ingram
Department of Mechanical
and Aerospace Engineering
North Carolina State University
Box 7910
Raleigh, NC 27695-7910
(919) 515-5250
ingram@predator.mae.ncsu.edu

Henry Jones
Mail Stop 461
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2158
h.e.jones@larc.nasa.gov

Yannis Kallinderis
Department of Aerospace Engineering
The University of Texas at Austin
Austin, TX 78712
(512) 471-4190
kallind@cfdlab.ae.utexas.edu

David Keyes
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-6873
keyes@icase.edu

Michael Marchant
Department of Civil Engineering
University of Wales
Singleton Park
W. Glamorgan Swansea SA2 8PP
UNITED KINGDOM
011 44 1792 295521
m.j.marchant@swansea.ac.uk

C. Wayne Mastin
Mail Stop 125
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-5781
cwm@geolab.larc.nasa.gov

Rohit Mathur
MCNC
North Carolina Supercomputing Center
P. O. Box 12889
Research Triangle Park, NC 27709-2889
(919) 248-9246
mathur@ncsc.org

Dimitri Mavriplis
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2213
dimitri@icase.edu

Scott McRae
Department of Mechanical
and Aerospace Engineering
North Carolina State University
P. O. Box 7910
Raleigh, NC 27695-7910
(919) 515-5244
mcrae@eos.ncsu.edu

William Mitchell
National Institute
for Standards and Technology
Bldg. 101, Room 238
Gaithersburg, MD 20899
(301) 975-3808
mitchell@cam.nist.gov

David Modiano
Institute for Computational Mechanics
in Propulsion
NASA Lewis Research Center
Brook Park, OH 44142
(216) 962-3151
fsdave@icomp01.lerc.nasa.gov

Jens-Dominik Muller
Department of Aerospace Engineering
The University of Michigan
Ann Arbor, MI 48109-2118
(313) 936-0107
muller@engin.umich.edu

Michael Neaves
Department of Mechanical
and Aerospace Engineering
North Carolina State University
Box 7910
Raleigh, NC 27695-7910
(919) 515-5250
neaves@predator.mae.ncsu.edu

Paresh Parikh
VIGYAN, Inc.
30 Research Drive
Hampton, VA 23666
(804) 864-2244
parikh@aero00.larc.nasa.gov

Mary-Anne Posenau
Mail Stop 125
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-6717
m.a.k.posenau@larc.nasa.gov

Ken Powell
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118
(313) 764-3331
powell@umich.edu

Russ Rausch
National Research Council
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2261
r.d.rausch@larc.nasa.gov

Roland Richter
Cray Research
Center PATP
Scientific Parc (PSE)
Lausanne, CH - 1015
SWITZERLAND
011 41 21 693 7099
richter@dgm.epfl.ch

Thomas Roberts
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-6804
t.w.roberts@larc.nasa.gov

Philip Roe
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2140
(313) 764-3394
philroe@caen.engin.umich.edu

Jeffrey Saltzman
Los Alamos National Laboratory
Mail Stop B265
Los Alamos, NM 87545
(505) 667-4285
jss@c3serve.c3.lanl.gov

Mark Shephard
Scientific Computation Research Center
Rensselaer Polytechnic Institute
110 8th Street, Building CII - Room 7011
Troy, NY 12180-3590
(518) 276-6795
shephard@scorec.rpi.edu

David Sidilkover
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-7312
sidilkov@icase.edu

Robert Smith
Mail Stop 125
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-5774
robert.e.smith@larc.nasa.gov

Jerry South
Mail Stop 285
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-3520
j.c.south@larc.nasa.gov

Erlendur Steinthorsson
Institute for Computational Mechanics
in Propulsion
NASA Lewis Research Center
22800 Cedar Point Road
Brook Park, OH 44142
(216) 962-3162
fsstein@icomp01.lerc.nasa.gov

Roy C. Swanson
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2235
swanson@tab00.larc.nasa.gov

James Thomas
Mail Stop 128
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2163
j.l.thomas@larc.nasa.gov

Jean-Yves Trepanier
Ecole Polytechnique de Montreal
C.P. 6079 Suce "Centre-ville"
Montreal, Quebec H3C 3A7
CANADA
(514) 340-4896
jyves@kepler.meca.polymtl.ca

John Van Rosendale
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2189
jvr@icase.edu

V. Venkatakrishnan
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2183
venkat@icase.edu

Gary Warren
Mail Stop 139
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-2162
g.p.warren@larc.nasa.gov

Jeff White
Analytical Services & Materials, Inc.
Mail Stop 168
NASA Langley Research Center
Hampton, VA 23681-0001
(804) 864-7967
jawhite@tab00.larc.nasa.gov

Shawn Woodson
Cessna Aircraft Company
P. O. Box 7704
Department 178P
Wichita, KS 67277
(316) 652-2898
woodson@aero4.nasa.larc.gov

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1995	Conference Publication

4. TITLE AND SUBTITLE
ICASE/LARC Workshop on Adaptive Grid Methods

5. FUNDING NUMBERS
505-90-52-01

6. AUTHOR(S)
Jerry C. South, Jr., James L. Thomas, and John Van Rosendale, Editors

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001 and
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17539

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CP-3316

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category ~~71~~ 34
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Solution-adaptive grid techniques are essential to the attainment of practical, user friendly, computational fluid dynamics (CFD) applications. In this three-day workshop, experts gathered together to describe state-of-the-art methods in solution-adaptive grid refinement, analysis, and implementation; to assess the current practice; and to discuss future needs and directions for research. This was accomplished through a series of invited and contributed papers. The workshop focused on a set of two-dimensional test cases designed by the organizers to aid in assessing the current state of development of adaptive grid technology. In addition, a panel of experts from universities, industry, and government research laboratories discussed their views of needs and future directions in this field.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Adaptive grids; Computational fluid dynamics; Mesh refinement; Solution adaptive methods	275
	16. PRICE CODE A12

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified