1. Introduction

Many would agree that the most fundamental and important questions facing users of modern computational methods for flow predictions are the following:

(I) How good are the answers?

(II) How can one obtain the best possible answers for a fixed computational effort (or a fixed computing budget, fixed manpower level, or a fixed and limited computing capability)?

The first question is exceedingly difficult since it includes both the issue of the validity of the physical and mathematical model of the flow phenomena itself as well as the issue of the quality of the numerical approximation of the equations characterizing the model. To simplify matters for purposes of the present discussion, we shall dispense with the first issue and take for granted that the classical Navier–Stokes or, in the present paper, the Euler equations are adequate models of nature for the applications in mind. Thus, the first question reduces to a word: accuracy—how accurate are the numerical solutions?

The second question is seldom asked, but it is intrinsically connected to the first. It is common practice in applications of computational fluid dynamics to complex flow domains to generate extremely fine finite difference meshes in hopes of capturing all important features of the flow, even though the location of these special points of interest changes in time. This leads some to employ fine meshes in all positions of the flow domain where some important aspect of the flow might possibly manifest itself. The quality of results is generally judged by the invariance of solutions to further refinement if one can afford the cost of another
calculation. The fact that coarse-mesh solutions may be adequate in much of the domain at most instants of time cannot be exploited in traditional fixed mesh schemes.

After some thought about these issues, rather broad answers to the fundamental questions present themselves:

(I) **Accuracy.** To determine the accuracy of a computed solution, one can attempt to develop reliable a-posteriori estimates of local error. In other words, one might hope to be able to develop procedures which use the evolving computed solution to determine sharp estimates of local errors in various norms over each mesh cell and at each time step.

(II) **"Optimal" meshes.** Use adaptive procedures to continually change the structure of the mesh—the size of mesh cells, the location of grid points—so as to keep the local errors within a preassigned limit.

Obviously, the second answer assumes that one has some means to measure the local quality of the numerical solution and, therefore, presumes the availability of some type of a posteriori error estimate.

We describe, in this paper, algorithms and results developed in an attempt to more sharply resolve these answers, particularly that to question II, for a class of problems in compressible flow. More specifically, we describe here a class of very effective adaptive schemes for time-dependent Euler equations in two dimensions which employ both mesh refinement (when the local error is large) and mesh "unrefinement" (when the local error is small) and which generate the appropriate mesh changes as the solution evolves in time. This requires that we estimate the local approximation errors at each time step. However, only an *indication* of the relative error between successive meshes is essential in our methods; the issue of very sharp a posteriori estimates of local error (our answer to question I) is one of great concern to us and is the subject of other papers [8, 18, 19].

In designing an adaptive scheme for Euler equations, we keep the following guidelines in mind:

(1) **Unstructured grids.** The method must be virtually grid-independent and global-coordinate-free. While an initial mesh can be defined to model the basic geometry of the flow domain and the initial data, thereafter it must be possible to automatically add or eliminate cells and grid points as needed to monitor local accuracy levels. This requirement considerably lessens the attractiveness of body-fitted coordinates, many elliptic/algebraic mesh generators, and various factorization algorithms which exploit such regular mesh topologies.

(2) **General geometries and boundary conditions.** The method must be applicable to arbitrary flow domains with virtually arbitrary geometry and general boundary conditions.

(3) **Solid mathematical basis.** Since, by its nature, any sound adaptive method must employ some type of local error estimator, it is important that the methods employed have a reasonably firm mathematical basis, e.g., that a priori or a posteriori error estimates exist and that the convergence characteristics of the method are acceptable.

(4) **High accuracy.** The method should be capable of delivering high-order accuracy.

(5) **Robustness.** The method must be numerically stable and not sensitive to singularities, distortions in the mesh, or irregularities in the data.

(6) **Supercomputing.** The method should lend itself to modern supercomputing methods for accelerating computational speed, such as easy vectorization or implementation on parallel processors, etc.
(7) **Computational efficiency.** The method and the supporting algorithms and data structures must be computationally efficient.

We feel that these criteria can be best met by finite element methods. In the present work, we use as the basis of our adaptive schemes a semi-explicit method used by several other authors (e.g., [4, 15, 16, 21]): a two-step Lax–Wendroff/Taylor–Galerkin scheme. It is far from optimal (and does not satisfy all of our criteria), but is perfectly adequate to use in conjunction with our adaptive scheme. Schemes which fulfill all of these criteria are under development and will be reported in subsequent papers.

We remark that there is a growing literature on adaptive methods in computational fluid mechanics. Adaptive procedures for incompressible viscous flow problems were developed by the authors in a series of recent papers (see, e.g., [8, 18, 19]). These methods employed a variety of different adaptive strategies, but did not come as close to satisfying the above criteria as the methods discussed in the present work. The general subject of adaptive finite element methods is dealt with in a volume of collected papers edited by Babuška et al. [2]. For a survey of adaptive finite difference schemes, see the works of Anderson [1]. Also, Berger and Oliger [5] and Berger and Jameson [4] have recently developed adaptive finite difference methods for hyperbolic conservation laws. Still other types of adaptive methods for hyperbolic problems have been recently proposed by Demkowicz and Oden [10, 11].

Following this introduction, we develop weak formulations of a class of problems in compressible gas dynamics. These space-time formulations are shown to be the basis of a class of Lax–Wendroff/Taylor–Galerkin schemes. Our derivation of this family of algorithms is nonstandard, in that we show that a two-step scheme follows easily from the use of a numerical quadrature scheme for evaluating appropriate flux integrals. In Section 3, finite element models of the space-time formulation are introduced, and in Section 4 we discuss the important issue of a posteriori error estimation. Section 5 is devoted to a detailed discussion of adaptive strategies. These include an $h$-method, wherein the mesh is refined or unrefined when local errors fall outside a preassigned upper and lower bound, and an $r$-method, in which the mesh is automatically distorted to equidistribute error. In Section 6 we present the results of several numerical experiments on two-dimensional problems. These results illustrate that the performances of the adaptive schemes are quite acceptable for a class of complex flow problems.

2. Preliminaries

We consider the motion of a perfect gas flowing through a domain $\Omega$ over a time interval $[0, T]$. We shall confine our attention to two-dimensional cases, $\Omega \subset \mathbb{R}^2$; we denote by $D$ the space-time domain, $D = \Omega \times (0, T)$, and by $\partial \Omega$ the boundary of $\Omega$. The motion of the gas is governed by the global balance laws of physics and the second law of thermodynamics. Thus, if $U = U(x, t)$, $(x, t) \in D$, is the 4-vector of conservation variables, $U = \{\rho, m, E\}$, with $\rho$ the mass density, $m$ the linear momentum, and $E$ the total energy, and if $d\Omega$ and $dS$ denote Lebesgue measures of area (volume) and length (area) of $\Omega$ and $\partial \Omega$, respectively, then we demand that $U$ satisfy the following system of conservation laws:

$$
\frac{d}{dt} \int_{\Omega} U \ d\Omega = - \int_{\partial \Omega} Q(U)n \ dS.
$$

(2.1)
Here, \( Q(U) \) is the flux and \( n \) is the unit outward normal to \( \partial \Omega \). If \( m_1, m_2 \) denote Cartesian components of \( m \), then

\[
U = \{ \rho, m_1, m_2, E \},
\]

\[
Q(U) = \begin{bmatrix}
\rho^{-1} m_1 + p(U) & \rho^{-1} m_2 \\
\rho^{-1} m_1 m_2 & \rho^{-1} m_2^2 + p(U) \\
\rho^{-1} m_1 (E + p(U)) & \rho^{-1} m_2 (E + p(U))
\end{bmatrix},
\tag{2.2}
\]

\[
n = \{ n_1, n_2 \}, \quad p(U) = (\gamma - 1)(E - \rho^{-1} m \cdot \frac{1}{2} m).
\]

In (2.2), \( p \) is the thermodynamic pressure and \( \gamma \) is the ratio of specific heats, assumed here to be constant. In addition to (2.1), \( U \) must satisfy the entropy production inequality

\[
\frac{d}{dt} \int_{\partial \Omega} \rho \eta(U) \, d\Omega + \int_{\partial \Omega} n \cdot \left( m \eta(U) + \theta^{-1} q \right) \, dS \geq 0,
\tag{2.3}
\]

with \( \eta(U) \) the entropy density of the gas, \( \theta \) the absolute temperature, and \( q \) the heat flux, as well as an initial condition,

\[
U(x, 0) = U_0(x), \quad x \in \Omega,
\tag{2.4}
\]

where \( U_0 \) is given.

It is of fundamental importance to note the smoothness requirements on \( U \) in order that (2.1) makes sense mathematically. Conservation laws (2.1) hold when the components of \( U \) are bounded measurable (with respect to Lebesgue measure in \( x \)) functions on \( D \). Thus, we may seek solutions in the function space

\[
V = \{ V = (V_1, V_2, V_3, V_4) \mid V_i = V_i(x, t) \in L^\infty(0, T; L^1(\Omega)), i = 1, \ldots, 4 \}.
\tag{2.5}
\]

In particular, (2.1) is not equivalent to the classical Euler equations,

\[
U_t + \text{div} \, Q(U) = 0,
\tag{2.6}
\]

(with \( U_t = \partial U / \partial t \) and \( \text{div} \, Q = \sum_i \partial Q_{a_i} / \partial x_i \)) since solutions of (2.1) may not possess derivatives across surfaces in \( D \). However, the conservation laws, initial and jump conditions are fully equivalent to the following weak boundary-initial value problem:

Find \( U \in V \) such that

\[
\int_{\partial \Omega} (U^t \phi + Q(U) \cdot \nabla \phi) \, d\Omega \, dt + \int_{\partial \Omega} U_t^t \phi(\cdot, 0) \, d\Omega = \int_0^T \int_{\partial \Omega} F^t \phi \, dS \, dt
\tag{2.7}
\]

for all \( \phi \in W \).
where \( F \) is the actual prescribed flux through \( \partial \Omega \) is \( W \) a suitable space of test functions, e.g.,

\[
W = \{ \varphi = (\varphi_1, \varphi_2, \varphi_3, \varphi_4) | \varphi_i = \varphi_i(x, t), \varphi_i \in C^1(\overline{D}), \varphi_i(x, T) = 0, i = 1, \ldots, 4 \}\quad (2.8)
\]

In (2.7), we use the notation

\[
U^t \varphi_i = \sum_{j=1}^4 U_a \frac{\partial \varphi_a}{\partial t}, \quad Q : \nabla \varphi = \sum_{i=1}^2 \sum_{j=1}^4 Q_{ai} \frac{\partial \varphi_a}{\partial x_j}. \quad (2.9)
\]

On the other hand, if \( U \) is known to have integrable derivatives in \( D \) everywhere except on a family of surfaces \( \{ \Gamma_k \}_{k=1}^K \), then we may consider the problem:

Find \( U \in \mathcal{V} \) such that

\[
\int_D (U^t \varphi + (\text{div } Q(U)) \varphi) \, d\Omega \, dt - \int_D U^t(\cdot, 0) \varphi(\cdot, 0) \, d\Omega \\
- \sum_{k=1}^K \int_{\Gamma_k} \varphi'(s_k \|U\| - \|Qn\|) \, dS \, dt + \int_D U_0 \varphi \, d\Omega + \int_0^T \int_{\delta \Omega} (Qn) \varphi \, dS \, dt \\
= \int_0^T \int_{\partial \Omega} F^t \varphi \, dS \, dt \quad \text{for all } \varphi \in W. \quad (2.10)
\]

Here, \( s_k \) are the speeds of propagation of discontinuities across \( \Gamma_k \) and \( V, W \) are appropriately redefined, e.g.,

\[
\mathcal{V} = \{ V = (V_1, V_2, V_3, V_4) | V_i \in L^\infty(D), V_i \in H^1(D_{ki}), D_{ki} = D - \Gamma_{ki} \}, \\
W = \{ \varphi | \varphi_i \in C^0(\overline{D}), \varphi_i(x, T) = 0 \},
\]

where \( \Gamma_{ki} \) are the surfaces on which \( V_i \) suffers a jump. If \( F \) is not a prescribed flux but is merely a notation for \( Qn \), then these flux terms cancel and do not appear in the formulation.

Consider an arbitrary time interval \( [\tau_1, \tau_2] \subset [0, T] \) and include in \( W \) functions \( \varphi(x, \tau_2) \neq 0 \). Let \( \omega \) be a subset of \( \Omega \) such that \( \omega \cap \bigcup_k \Gamma_k = \emptyset \), and let \( F = Qn \). Then another weak statement of the system conservation laws over \( \omega \times [\tau_1, \tau_2] \) is:

Find \( U \in V^{\omega, \tau} \) such that

\[
\int_{\tau_1}^{\tau_2} \int_{\omega} (-U^t \varphi + (\text{div } Q) \varphi) \, d\Omega \, dt + \int_{\omega} (U^t(\cdot, \tau_2) \varphi(\cdot, \tau_2) - U^t(\cdot, \tau_1) \varphi(\cdot, \tau_1)) \, d\Omega = 0
\]

for all \( \varphi \in W^{\omega, \tau} \),

\[
\quad (2.11)
\]

with \( V^{\omega, \tau} \) and \( W^{\omega, \tau} \) appropriate spaces of trial and test functions.
REMARK 2.1. It is well known that (2.7), (2.10), and (2.11) may all possess nonphysical solutions since none of these formulations involve the entropy inequality (2.3). Thus, in general, we seek solutions to (2.7), (2.10), or (2.11) in the subset $K \subset V; K = \{ v \in V \mid v$ satisfies (2.3) for appropriate $\theta, q(\nabla \theta) \}$. 

3. Finite element approximations

Finite element approximations of the gas dynamics problem are obtained by a direct approximation of (2.10) or (2.11) on finite-dimensional spaces approximating the spaces $V$ and $W$. The spatial domain $\Omega$ is partitioned into a collection $\mathcal{T}_h$ of finite elements $\Omega_e$ over which the components of trial functions $V$ are approximated by polynomials of degree $k$. In this way, we construct a family $\{V_h\}$ of finite-dimensional spaces of the type

$$V_h = \{ V^h, V^h_1, V^h_2, V^h_3, V^h_4 \} \in V \mid V^h \in P_k(\Omega_e), i = 1, \ldots, 4 \},$$

where $P_k(\Omega_e)$ is the space of polynomials of degree $k$ defined over $\Omega_e$. Alternatively, we can use $V^h_{1|\Omega_e} \in Q_k(\Omega_e)$, where $Q_k(\Omega_e)$ is the space of tensor products of polynomials of degree $k$ on $\Omega_e$ (e.g., $Q_1(\Omega_e)$ is spanned by bilinear functions, $Q_2(\Omega_e)$ by biquadratics, etc.). In addition, a family $\{W_h\}$ of finite-dimensional spaces of test functions is also constructed.

We then consider Galerkin approximations of (2.7), (2.10), or (2.11) by seeking solutions to these equalities in $V^h$, with $V$ and $W$ replaced by $V^h$ and $W^h$, respectively.

3.1. A two-step, Lax–Wendroff/Taylor–Galerkin scheme

We next derive a special semidiscrete, weak formulation from (2.11) which provides the basis for the construction of a popular family of finite element schemes. We proceed with the following steps:

**Step 1.** Partition the time interval $[0, T]$ according to $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$.

**Step 2.** Apply the weak balance law (2.11) to a typical time interval $[t_n, t_{n+1}]$ (with $\tau_1 = t_n$ and $\tau_2 = t_{n+1}$).

**Step 3.** Set $\varphi_i = 0$ in (2.11) suggesting the ultimate use of a time-invariant grid (we relax this assumption later).

**Step 4.** Replace the time integrations in (2.11) by the elementary midpoint quadrature rule

$$\int_{t_n}^{t_{n+1}} f(t) \, dt \approx \Delta t \, f^{n+1/2}, \quad \Delta t = t_{n+1} - t_n, \quad f^{n+1/2} = f(t_n + \frac{1}{2} \Delta t).$$

Thus, with $\omega = \Omega$, we obtain the semidiscrete approximation

$$\int_{\Omega} \varphi_i U^{n+1} \, d\Omega = \int_{\Omega} \varphi_i U^n \, d\Omega + \Delta t \int_{\Omega} Q^{n+1/2} : \nabla \varphi_h \, d\Omega$$

$$- \Delta t \oint_{\partial \Omega} \varphi_i (Q^{n+1/2} n) \, dS \quad \text{for all } \varphi_h,$$
where \( U_h^n = U_h(x, t_n) \), etc., \( U_h \) being the approximation of \( U \), and \( Q^{n+1/2} \) is the flux at the half step,

\[
Q^{n+1/2} = Q(U_h^{n+1/2});
\]  (3.3)

Step 5. To obtain an approximation \( U_h^{n+1/2} \), we use (2.11) again for time interval \([t_n, t_{n+1/2}]\), this time replacing the time integrals by a simple strip rule,

\[
\int_{t_n}^{t_{n+1/2}} \varphi^i_h U_h^n \, d\Omega = \int_{t_n}^{t_{n+1/2}} \varphi^i_h U_h^n \, d\Omega - \frac{1}{2} \Delta t \int_{R_c} \varphi^i_h (\text{div } Q^n) \, d\Omega \quad \text{for all } \varphi^i_h.
\]  (3.4)

We thus arrive at the following algorithm:

Step 1. With \((U_h^n, Q^n = Q(U_h^n))\) known at the \(n\)th time step, compute \(U_h^{n+1/2}\) using (3.4);
Step 2. Compute \(Q_h^{n+1/2}\) using (3.3);
Step 3. Compute \(U_h^{n+1}\) using (3.2);
Step 4. Go to Step 1.

This algorithm is the finite-element-based two-step Lax–Wendroff/Taylor–Galerkin scheme (see [7, 20]). It is one of a family of methods advanced by Donea [13], studied by Baker and Kim [3], and successfully refined and used by Löhner et al. [15, 16] and Bey et al. [6] in finite element applications in fluid dynamics. This semi-explicit method is of second order in time and can experience spurious oscillations near shocks and other types of irregularities in the solution. These deficiencies must be reckoned with in implementing the method.

3.2. Artificial viscosity

As noted earlier, artificial viscosity terms are usually added to schemes such as the one employed here so as to dampen out oscillations in the numerical solutions near shocks. The calculations described subsequently were performed adding a Lapidus viscosity term [14], which, at time step \(t_n\), is of the form

\[
-\nabla \cdot (c(u) \cdot \nabla U^{n+\alpha}),
\]  (3.5)

where

\[
c_i(u) = C |\partial u_i / \partial x_i| \quad \text{(no sum on } i),
\]

\(C\) is the Lapidus constant, \(u_i = m_i / \rho\) is the \(i\)th component of the flow velocity, and \(\alpha\) is a parameter which determines whether viscosity is to be included implicitly \((\alpha = 1)\) or explicitly \((\alpha = 0)\). The viscosity term, written out in component form, is

\[
- \sum_{\beta=1}^{2} \frac{\partial}{\partial x_k} \left( c_k(u) \frac{\partial}{\partial x_k} U^{n+\alpha}_\beta \right), \quad \beta = 1, \ldots, 4.
\]

Setting \(\alpha = 1\), we obtain for Step 2 of the procedure (instead of (3.2)),

...
for all admissible test functions $\varphi_h$.

3.3. Details of the finite element algorithm

The details of the implementation of the algorithm described above are crucial to successful computations. In this work, we use meshes of four-node quadrilateral ($Q_4$) elements over which the components $U_a$ ($a = 1, \ldots, 4$) of $U$ are piecewise bilinear functions. Similar approximations and algorithms are used by Bey et al. [6]. In addition, so-called group approximations of the flux $Q_{ai}$ ($a = 1, \ldots, 4$, $i = 1, 2$) are employed so that these components are also piecewise bilinear functions determined by their values at element nodes. In general, this finite element approximation will be of the form,

$$U_a = \sum_{j=1}^N U_a^j(t) \varphi_j(x), \quad Q_{ai} = \sum_{j=1}^N Q_{ai}^j(t) \varphi_j(x),$$

(3.7)

where $N$ denotes the total number of nodes in the discretization, and $U_a^j, Q_{ai}^j$ are values of $U^h, Q^h$ at node $j$, and $\varphi_j$ are the global piecewise bilinear basis functions.

As noted earlier, we advance the solution in time in two steps. It is important to note that the first step is essentially local, computed over each element, while the second is global and contains the artificial viscosity terms:

Step 1. For each element $\Omega_e$, calculate a constant element vector $U_{a,e}^{n+1/2}$ from

$$U_{a,e}^{n+1/2} \int_{\Omega_e} d\Omega = \sum_{i=1}^4 \left( \int_{\Omega_e} \varphi_i d\Omega \right) U_a^{i,n} - \frac{1}{2} \Delta t \left( \int_{\partial \Omega_e} \frac{\partial \varphi_i}{\partial x_\beta} d\partial\Omega \right) Q_{a,i}^{n+1/2}. \quad \text{(3.8)}$$

Step 2. For each node $j$, calculate $U_a^{j,n+1}$ by solving the following system of equations:

$$\sum_{j=1}^N \left( \int_{\Omega_e} \left( \varphi_j \varphi_i + \tau_\beta \frac{\partial \varphi_i}{\partial x_\beta} \frac{\partial \varphi_j}{\partial x_\beta} \right) d\Omega \right) U_a^{j,n+1}$$

$$= \sum_{j=1}^N \left( \int_{\Omega_e} \varphi_i d\Omega \right) U_a^{i,n} + \Delta t \int_{\Omega_e} Q_{a,i}^{n+1/2} \frac{\partial \varphi_i}{\partial x_\beta} d\Omega$$

$$- \Delta t \int_{\partial \Omega_e} n_\beta (Q_{a,i}^{n+1/2} - \bar{Q}_{a,i}^n) \varphi_i ds - \Delta t \int_{\partial \Omega_e} n_\beta Q_{a,i}^n \varphi_i ds. \quad \text{(3.9)}$$

Here, $\bar{Q}^n$ denotes the elementwise averaged value of the flux. The coefficients $\tau_\beta$ are defined to be constant over each element,
where $c$ is a global constant ($c = 1$ in the examples), $A_e$ denotes the area of $\Omega_e$, and $u^h_\beta$ denote the components of the fluid velocity.

To speed up the calculation, we precalculate and store the following element integrals before the time stepping is started:

$$
\int_{n_i} \varphi_i \, d\Omega, \quad \int_{n_i} \frac{\partial \varphi_i}{\partial x_\beta} \, d\Omega, \quad \int_{n_i} \varphi_i \varphi_j \, d\Omega, \quad \int_{n_i} \frac{\partial \varphi_i}{\partial x_\beta} \frac{\partial \varphi_j}{\partial x_\beta} \, d\Omega,
$$

$i, j = 1, \ldots, 4, \beta = 1, \ldots, 4$.

An element-by-element Jacobi conjugate gradient method is used to obtain the solution of the matrix problem in Step 2. Due to the structure of the mass matrix, the iterative solver requires only a few iterations to converge fully.

### 3.4. Boundary conditions

In the finite element schemes developed here, we implement the following three types of boundary conditions:

(a) **Supersonic inflow.** On the part of the boundary with supersonic inflow, the values of all the conservation variables are imposed.

(b) **Supersonic outflow.** On the outflow part of the boundary, the values of the conservation variables and the normal flux are unknown. Boundary conditions of supersonic outflow are implemented by adding the contribution of the boundary integral of the normal flux to the right-hand side of the equations of Step 2.

(c) **Solid boundaries.** On a solid boundary, the normal component of the velocity $u_n = u_\beta n_\beta$ is zero. We note that, in general, the nodal directions are not uniquely defined. In such calculations, we compute the normal directions at the nodes which satisfy global mass conservation at the steady state, namely,

$$
n^I_\beta = \int_{n_i} \frac{\partial \varphi_i}{\partial x_\beta} \, d\Omega \left/ \left( \sum_{\beta=1}^2 \left( \int_{n_i} \frac{\partial \varphi_i}{\partial x_\beta} \, d\Omega \right)^2 \right)^{1/2} \right.
$$

### 3.5. Hourglass instabilities

We now show that the Taylor–Galerkin scheme presented above can propagate undetected spurious solutions. To demonstrate this, let us consider the scheme applied on the 2-D Burgers equations on a mesh of rectangular elements. Burgers equations may be obtained from the above formulation by redefining the flux as follows:

$$
\{Q_{a1}\} = \begin{pmatrix} 0 \\ U_2^2 \\ U_2 U_3 \\ 0 \end{pmatrix}, \quad \{Q_{a2}\} = \begin{pmatrix} 0 \\ U_3 U_2 \\ U_3^2 \\ 0 \end{pmatrix}.
$$

Consider a rectangular element with the following nodal solution at time $t_n$: 

Then the scheme gives
\[ \{U_a^n\}^{n+1/2} = [0, 1, 0, 0]^t, \]
and, by letting \( c = 0 \) (no artificial viscosity), we get:
\[ \{U_a^n\}^{n+1} = \{U_a^n\}. \]

This means that the scheme propagates "hourglass" solutions undetected. This fact explains why in the numerical examples the method produced oscillations near the outflow boundaries. This hourglassing phenomenon can be eliminated by considering each quadrilateral as a patch of two triangular elements joined along one diagonal of a quadrilateral.

4. Errors

The adaptive finite element methods described here involve two basic components:
(1) Error estimation—the determination of a posteriori estimates of the evolution of error in the numerical solution.
(2) Adaptation—the automatic restructurings of the approximation so as to reduce local element errors and the computational effort.

In this paper, adaptive procedures are based on estimates of error in a single principal dependent variable, such as the density, pressure, or the entropy. We shall choose the density \( \rho \) as the driving factor in adaptivity, although other choices could be used in the algorithms developed here. Two basic procedures are used to estimate local element errors.

4.1. Evolution equation for error

Consider the continuity equation for the evolution of mass density through a domain \( \Omega \) with known flow velocity \( u \). A weak form of the continuity equation is
\[ \int_\Omega \varphi \rho_t \, d\Omega = - \int_\Omega \nabla \cdot (u \varphi) \, d\Omega \quad \text{for all } \varphi \in W. \tag{4.1} \]

A semidiscrete Galerkin approximation of (4.1) consists of seeking an approximate density \( \rho^h \) such that, over some suitable finite-dimensional space of test functions \( W_h \),
\[ \int_\Omega \varphi_h \rho^h_t \, d\Omega = \int_\Omega \nabla \cdot (u \varphi^h) \varphi_h \, d\Omega \quad \text{for all } \varphi_h \in W_h. \tag{4.2} \]

If \( W_h \subset W \), we may choose \( \varphi = \varphi_h \) in (4.1), subtract (4.2), and obtain the following evolution equation for the error \( e^h(x, t) = \rho(x, t) - \rho^h(x, t) \):
The exact and approximate solutions are related according to

\[ \rho = \rho^h + e^h, \]  

(4.4)

where \( e^h \) is the approximation error. Thus, the error satisfies the evolution equation

\[ \int_{\Omega} (\varphi e^h + \nabla \cdot (u e^h) \varphi) \, d\Omega = \langle r_h, \varphi \rangle \quad \text{for all } \varphi \in W, \]  

(4.5)

where \( \langle r_h, \varphi \rangle \) is the residual functional,

\[ \langle r_h, \varphi \rangle = -\int_{\Omega} (\rho^h \varphi + \nabla \cdot (u \rho^h) \varphi) \, d\Omega. \]  

(4.6)

If we replace \( \varphi \) by \( \varphi_h, \langle r_h, \varphi_h \rangle = 0 \) by (4.3), and the evolution equation reduces to merely the orthogonality condition (4.3), which is automatically satisfied by error.

We obtain an approximate evolution equation for the error as follows. Let \( E^h \) denote a fine-grid approximation of \( e^h \); i.e.,

\[ e^h(x, t) = E^h(x, t) = \sum_N E^h(i) \psi_N(x), \]  

(4.7)

where \( \psi_N(x) \) denotes a polynomial basis function defined on a subgrid of finer mesh size than that used to calculate \( \rho^h \). Then, introduction of (4.7) into (4.5) and replacing \( \varphi \) by \( \psi_N \) gives

\[ \sum_M (m_{N M} \dot{E}^M + k(u)_{N M} E^M) - r_N(t) = 0, \quad N = 1, 2, \ldots, N, \]  

(4.8)

where

\[ m_{N M} = \int_{\Omega} \psi_N \psi_M \, d\Omega, \quad k(u)_{N M} = \int_{\Omega} \nabla \cdot (u \psi_N) \psi_M \, d\Omega, \quad r_N = \langle r_h, \psi_N \rangle. \]  

(4.9)

Many possible ways for implementing (4.9) present themselves. These equations, for example, need not be global in the sense that an element-by-element or patch of elements in a fine mesh obtained through a mesh refinement may produce sufficient accuracy to allow for an adequate indication of the evolution of error. The local velocities \( u \) and residual \( r_h \) can be interpolated using \( Q_1 \)-approximations on a fine-mesh level. Several of these alternatives are under study and are to be the subject of a forthcoming report.

4.2. Interpolation errors

Let \( u \) be a smooth function defined over a regular domain \( \Omega \). The \( W^{r,p}(\Omega) \) seminorm of \( u \) is defined by

\[ |u|_{W^{r,p}(\Omega)} = \left\{ \int_{\Omega} \sum_{i+j=r} \left| \frac{\partial^{i+j} u}{\partial x_i \partial x_j} \right|^p \, d\Omega \right\}^{1/p}, \]  

(4.10)
where \( 1 \leq p \leq \infty \) and \( r \) is a nonnegative integer. The Sobolev norm of \( u \) is

\[
\|u\|_{W^{r,p}(\Omega)} = \left\{ \sum_{k=0}^{r} \|u_k\|_{k+r,p(\Omega)}^p \right\}^{1/p}.
\]

(4.11)

Let \( G \) be an arbitrary convex subdomain (a finite element) of \( \Omega \) over which \( u \) is interpolated by a function \( \hat{u}_h \), which contains complete piecewise polynomials of degree \( k \). Then, it can be shown [17] that the local interpolation error in the \( W^{m,p}(G) \) seminorm is

\[
|u - \hat{u}_h|_{W^{m,p}(G)} \leq C h^{k+1} \cdot h^{n/p} \cdot h^{n/p - n/p'} |u|_{k+1,p(G)} ,
\]

(4.12)

where \( h \) is the diameter of the domain \( G \), \( \rho \) is the diameter of the largest sphere that can be inscribed inside \( G \), \( n \) is the dimension of the domain \( \Omega \), \( p' \) is \( p/(p - 1) \), and \( C \) is a constant independent of \( h \), \( \rho \), and \( u \). If \( \rho \) is proportional to \( h \) and if it remains proportional in refinements of \( G \) defined by parametrically reducing \( h \), we have

\[
|E^h|_{m,p,G} \approx Ch^{n/p - n/p + k + 1 - m} |u|_{k+1,p} ,
\]

(4.13)

with \( |\cdot|_{m,p,G} = |\cdot|_{W^{m,p}(G)} \), etc., and \( E^h = u - \hat{u}_h \).

Such estimates can be used to devise crude adaptive schemes. Suppose that \( u \) on the right side of (4.13) is replaced by a finite element approximation \( u_h \) and that \( |u_h|_{k+1,p} = |u|_{k+1,p} + O(h) \). Then (4.13) indicates that the local error in the \( W^{m,p}(G) \) seminorm is proportional to the error indicator, \( h^{n/p - n/p + k + 1 - m} |u|_{k+1,p} \). Some choices are:

(i) \( n = 2, m = 0, k = 1, p = p' = 2 \):

\[
|E^h|_{L^2(G)} \leq Ch^2 |\mathbf{u}|_{2,2,G} .
\]

In this case, one must approximate the \( W^{2,2} \) seminorm of \( u \) over \( G \); i.e., the \( L^2 \) norm of second partial derivatives of \( u \).

(ii) \( n = 2, p = \infty, p' = 1, k = 0, m = 0 \):

\[
|E^h|_{L^1(G)} = Ch^2 |E^h_{\text{average}}| \leq Ch^2 |u|_{1,\infty,G} = Ch^2 \max_{x \in G} |\nabla \cdot u(x)| .
\]

Such estimates can give only rough indications of local errors in sufficiently fine meshes. However, they are usually easy to implement and our experience is that they can provide a very effective basis for mesh refinement strategies.

5. Adaptive mesh strategies

Let us suppose that we can calculate an error indicator \( \theta_e \) for each finite element \( \Omega_e \) in a given mesh at a time \( t \). This indicator is, in general, a real number representing the local error in a suitable norm, and it is computed using one of the procedures described in Section 4. The decision to adapt the numerical procedure (to refine the mesh or to move nodal points) is
based on whether or not local error indicators exceed preassigned tolerances. We shall describe two adaptive procedures in this section.

5.1. An h-refinement/unrefinement method

Our h-procedure involves the following steps:

Step 1. For a given domain $\Omega$, such as the one shown in Fig. 1(a), a coarse finite element mesh is constructed which contains only a number of elements sufficient to model basic geometrical features of the flow domain.

Step 2. As our adaptive process will be designed to handle groups of four elements at a time, we generate a finer starting grid by a bisection process, indicated in Fig. 1(b), to obtain an initial set of element groups.

Step 3. We initiate the numerical solution procedures on this initial coarse grid, and compute error indicators $\theta_e$ over all $M$ elements in the grid. Let

$$\theta_{\text{max}} = \max_{1 \leq e \leq M} \theta_e.$$

Step 4. Next, we scan groups of a fixed number $P$ of elements and compute

$$\theta_{\text{group}}^m = \sum_{k=1}^{P} \theta_{e_k},$$

where $e_k$ is the element number for group $m$. We take $P = 4$ in our current codes.

Fig. 1. (a) A coarse initial mesh consisting of four-element groups and (b) the refinement and unrefinement of a group of elements.
Step 5. Error tolerances are defined by two real numbers, \( 0 < \alpha, \beta < 1 \). If \( \theta_e \geq \beta \theta_{\text{max}} \), we refine element \( \theta_e \). This is done by bisecting \( \theta_e \) into four new subelements. If \( \theta_{\text{group}} \leq \alpha \theta_{\text{max}} \), we unrefine group \( k \) by replacing this group with a single new element with nodes coincident with the corner nodes of the group. This is always possible because each group is itself the result of an initial bisectioning.

This general process can be followed for any choice of error indicator. Moreover, it can also be implemented at each time step in the numerical schemes discussed in Section 3.

5.2. Data structures

An important consideration in all adaptive schemes is the data structure and associated algorithms needed to handle the changing number of elements, their node locations and numbers, and the element labels.

As noted in the preceding paragraphs, the algorithm is designed to process (refine or unrefine) in groups of four elements at each local refinement/unrefinement step. Consider, for example, the case of an initial mesh of 20 square elements shown in Fig. 2. We assign to each element in this mesh an element number, \( NEL = 1, 2, \ldots, \text{NELEM} \), and to each global node a label \( \text{NODE} \). The array \( \text{NODES}(J, NEL) \) relates the local node number \( J(J = 1, \ldots, 4) \) of element \( NEL \) to the global node number \( \text{NODES} \). In addition, the coordinates \( X_J, Y_J \) of each node are also provided relative to a fixed global coordinate system. We file these numbers in two arrays:

- \( \text{NODES}(J, NEL) \) is the array of global node numbers assigned to node \( J \) of element \( NEL \),
- \( \text{XCO}(JCO, \text{NODE}) \) is the array of \( JCO \)-coordinates of global node \( \text{NODE} \) (\( JCO = 1 \) or 2).

Suppose that an error indicator is computed that signals that an element should be refined, say element 11 in the example. We must have some system for assigning appropriate labels to the new elements and nodes. Toward this end, we can establish a convention that defines the connectivity of the specified element with its neighbors in the mesh. This information is provided by a third connectivity array:

- \( \text{NELCON}(NC, NEL) \) is the \( NC \)th connection of element \( NEL \), \( NC = 1, 2, \ldots, 8 \).

As seen in Fig. 2, each side of an element may be connected to two other elements so that Dimension \( \text{NELCON} = (8, \text{MAXEL}) \), with \( \text{MAXEL} \) an appropriately large number.

The entire refinement (or its inverse—the unrefinement process) just described is accomplished by specifying a series of element levels. For example, the initial coarse mesh could be assigned level 0. When an element is refined, its subelements belong to a higher level, level 1, and when these subelements are refined, elements of level 2 result, and so on. In this way, if the maximum level any element in the mesh can achieve is limited, then the maximum number of elements the mesh can contain is also limited. In general, no such limit need be set.

Thus, the bookkeeping of element and node numbers evolving in a refinement process is monitored by the arrays \( \text{NODES}(:, :) \), \( \text{XCO}(:, :) \), \( \text{NELCON}(:, :) \), and an array \( \text{LEVEL}(NEL) \) which assigns a level number to element \( NEL \). Initially, the same level can be assigned to all elements, and this level is an arbitrary parameter, prescribed in advance by the user. Thus, provisions are now in hand for an arbitrary, dynamic renumbering of elements and nodes. If, for example for the mesh in Fig. 2, element 11 is to be refined, we proceed through the following steps:
Step 1. Loop over the neighbors of element 11 (which is made possible with the NELCON array) and check the level of the neighboring elements relative to the level of element 11.

Step 2. If any neighboring element has a level lower than element 11, then the element cannot be refined at this stage.

Step 3. If element 11 can be refined (as is the case in Fig. 2), we generate new element numbers (thus changing NELEM and new node numbers for unconstrained nodes).

Step 4. Compute the connectivity matrix NELCON for the new elements.

Step 5. Adapt the connectivity matrices for the neighboring elements (since the refinement of element 11 has now changed this connectivity).

Step 6. Interpolate the solution between the unconstrained nodes.

It is clear that some strategy is needed to test if a designated element is appropriately connected for a refinement to take place.

Consider, for example, the uniform grid of four elements shown in Fig. 3(a) and suppose that the error estimators dictate that element A is to be refined. Thus, A is divided into four
elements, I, II, III, IV, as shown, and the solution values at the junction nodes, shown circled in the figure, are constrained to coincide with the averaged values between those marked $\times$. Note that the connectivities change in this process, e.g., the connectivities 4 and 8 of element B are different.

Next, assume that an additional refinement is required, and that we must next refine element III. We impose the restriction that each element side can have no more than two elements connected to it. Thus, before III can be refined, element B must first be refined, as indicated in Fig. 3(b). The constrained node B in Fig. 3(a) now becomes active, while node C1 remains a constrained node. With B bisected, we proceed to refine III into subelements $\alpha, \beta, \gamma, \delta$, and new constrained nodes, again circled in Fig. 3(c), are produced. In this case, only element B had to be refined first in order to refine III, but, in general, the number of elements that must be refined in order to refine a particular element cannot be specified. The
following subroutine determines the necessary refinements prerequisite to refining an element NEL1:

```
SUBROUTINE DIVIDE(NEL1, NEL2)
    NEL1 = the input element that needs to be refined
    if NEL1 has been divided
    NEL2 = output element =
    NELD = element that needs to be divided prior to NEL1
```

Then, symbolically, we have the algorithm (for the example in Fig. 3),

```
Repeat
    NEL1 = III
    CALL DIVIDE (NEL1, NEL2)
    WHILE (NEL2.NE.NEL1)
        NEL1 = NEL2
        CALL DIVIDE (NEL1, NEL2)
    END WHILE
UNTIL (NEL2.EQ.III)
```

5.3. Moving mesh (node redistribution) methods

Another family of adaptive schemes we have considered is a node redistribution scheme which progressively moves a fixed number of nodes as to reduce local error. One basis for such schemes is to equidistribute error at each time step.

For example, let \( \theta_x \) be an error indicator for element \( \Omega_e \) in a mesh containing a fixed number \( M \) of elements in a two-dimensional mesh. Let \( h = h(x_1, x_2) \) be a mesh function such that

\[
h(x_1, x_2) = h_e = \text{dia}(\Omega_e) \quad \text{for} \quad (x_1, x_2) \in \Omega_e,
\]

and note that, approximately,

\[
M = \int_0^1 \frac{d\Omega}{h^2}, \quad (5.1)
\]

with \( d\Omega = dx_1 \, dx_2 \) (this being exact for domains which are unions of square elements). Let \( \theta = \theta(x_1, x_2) \) be the mesh function which gives the local error indicator when evaluated at a point \( (\theta = \theta_e \text{ for } x \in \Omega_e) \). We wish to minimize the total error indicator functional,

\[
J(\theta) = \sum_{e=1}^M \int_{\Omega_e} \theta_e^2 \, d\Omega, \quad (5.2)
\]

subject to the constraint (5.1). Using Lagrange multipliers, this leads to the optimality condition,
Suppose that \( \text{meas}(\Omega_e) = \sigma_0 h_e^2 \) and that \( \theta_e \) is of the form \( \theta_e = h_e^\gamma f(u) \). Then, integrating this last result over a typical element gives

\[
\int_{\Omega_e} \sigma h_e^\gamma \theta_e h_e^{\gamma-1} f(u) \, d\Omega = \lambda \sigma_0 h_e^2.
\]

Hence, the optimal mesh size distribution results when

\[
\int_{\Omega_e} \theta_e^2 \, d\Omega = \lambda \sigma_0 / \sigma = \text{CONST}.
\]

In other words, to obtain the optimal mesh, we must equidistribute the indicators \( \int \theta_e^2 \).

To use this result to redistribute nodes, we proceed as follows (cf. [12]):

Step 1. Generate an initial (generally regular) mesh with a fixed number \( M \) of elements and compute a trial solution on this mesh at one time step.

Step 2. Compute the corresponding error indicators \( \theta_e \).

Step 3. For a group \( k \) of \( P \) elements (with \( P \) always 4 in this work), let \( A_e \) denote the area of element \( i \) in the group. The area-weighted indicators for group \( k \) are the \( P \) numbers \( \theta_i / A_{i} \).

---

**Fig. 4.** Calculation of area center-of-error \( x^N \) to equidistribute element error indicators in a cluster of four elements.
Step 4. Let \( y_e \) denote a vector from the origin of a global coordinate system to the centroid of element \( e \) of group \( k \). Then the center of error of group \( k \) is defined as the vector

\[
x^k = \sum_{i=1}^{4} y_e \left( \frac{\theta_e / A_e}{\sum_{i=1}^{4} \theta_e / A_e} \right)
\]

(5.4)

Step 5. Relocate the node at the center of group \( k \) to lie at the vertex of \( x^k \), as shown in Fig. 4.

Step 6. Continue this sequence of operations over each group \( k \) of four elements until the new location of each node does not change more than a preassigned tolerance.

This process should approximately equidistribute the element error indicators.

6. Numerical examples

In this section, we present the results of several numerical experiments on representative test problems. Six examples are presented, the first five involving steady-state examples, one of the following two strategies is used.

**STRATEGY A**

Step A.1. The numerical solution is computed on a fixed mesh and is advanced in time until a steady state is reached.

Step A.2. After convergence to a steady state, error indicators \( \theta_e \) are computed over each element. In the calculations discussed below, we employ the interpolation estimates and use

\[
\theta_e = A_e |p^h|^2_{2, \alpha_e} = A_e \int_{\partial \Omega_e} \| \partial p^h \| \partial n \| \, dS,
\]

(6.1)

where \( A_e \) is the area of the element.

Step A.3. The mesh is refined/unrefined using the criteria and algorithms discussed in Section 5.

**STRATEGY B**

Step B.1. Same as Step A.1.

Step B.2. After convergence to a steady state, error indicators \( \theta_e \) are computed according to

\[
\theta_e = A_e \int_{\Omega_e} \nabla p^h \cdot \nabla p^h \, d\Omega.
\]

(6.2)

Step B.3. In applying the node redistribution (moving mesh) algorithm, a modified error indicator \( \tilde{\theta}_e \) is employed, which is designed to be always greater than unity even when \( \theta_e = 0 \). In particular, we use

\[
\tilde{\theta}_e = 1 + \alpha \theta_e / (\beta + \gamma \theta_e).
\]

In our examples \( \alpha = 81 \), \( \beta = 1 \), and \( \gamma = 8 \).
Step B.4. Nodes are redistributed a total of $K$ times using the procedure described in Section 5.3. In the examples, we take only two iterations ($K = 2$).

We proceed to the examples.

6.1. Shock reflection problem

We begin with a problem for which an exact solution is known and which has been used as a benchmark problem by others.

The problem involves the steady flow of a perfect gas in a rectangular duct in which density, velocity, and energy are prescribed in each of four triangular wedges in such a way that the appropriate jump conditions (the Rankine-Hugoniot conditions) are exactly satisfied. Thus, a problem of shock reflection for which an exact solution is known is obtained. Dimensions and data are given in Fig. 5. In this and all the other problems, the solution is considered to have converged to steady state when the magnitude of the $L^2$ norm of the density is reduced by three orders of magnitude.

The time step is monitored by the formula

$$\Delta t = \min(0.50\sqrt{A_e/(|u| + C)}) .$$

Here, $C^2 = \gamma P/\rho$ and $|u|^2 = u_1^2 + u_2^2$, $\gamma = 1.40$. The constants multiplying the artificial viscous terms were selected locally as:

$$\tau_x = A_e|\partial u/\partial x|_e, \quad \tau_y = A_e|\partial u/\partial y|_e,$$

where the bar denotes average element values. A Lapidus constant of 1.0 was used in all calculations.

The results of a uniform coarse initial mesh approximation are shown in Fig. 6. The computed density contours are also shown in this figure. Note that only a rough indication of the location of the shock is possible with this mesh.

A much better resolution is given in Fig. 7, where the adaptively refined mesh shown is computed with refinement parameters $\alpha = 0.10$, $\beta = 0.50$ (recall Section 5). Note that no

![Fig. 5. A shock reflection problem. Inflow values of the conservation variables are prescribed as indicated in regions I and II, and outflow values are computed in region III to satisfy the conservation laws.](image-url)
Fig. 6. Reflecting shock problem. (a) Initial mesh and (b) density contours.

Fig. 7. Reflecting shock problem. (a) Mesh and (b) density contours obtained with one level of refinement ($\alpha = 0.10$, $\beta = 0.50$).
"unrefinement" appears to have taken place with these parameter choices, but that the simple error estimation scheme is capable of detecting the general area of the shock line. The much improved density profiles are indicated in the figure.

Still better results are obtained with the same $\alpha$ and $\beta$ but with two levels of refinement, as indicated in Fig. 8. Note that in this case large elements appear in the mesh, indicating unrefinement as well as refinement of the original mesh. The corresponding density surface is given in Fig. 9, where quite sharp shock fronts are observed. Note some spurious oscillation is encountered near the outflow boundary, as should be expected from the deficiencies of the algorithm noted in Section 4.

The same problem was also analyzed using the node redistribution algorithm discussed in Section 5.3 with ten node redistribution iterations. Results are shown in Fig. 10. There, the original coarse initial mesh of Fig. 6 is progressively distorted to conform to the reflected shock locations. Corresponding density contours are also given in the figure.

Fig. 8. Reflecting shock problem. (a) Mesh and (b) density contours obtained with two levels of refinement ($\alpha = 0.10$, $\beta = 0.50$).
6.2. NACA 0012 airfoil in supersonic wind tunnel

In this example, the supersonic flow through a narrow wind tunnel containing a NACA 0012 airfoil is studied. The inflow Mach number was set at $M_\infty = 2$, with $\gamma = 1.40$, and symmetry is exploited to reduce the computational effort.

The initial coarse mesh and density computed contours are given in Fig. 11. Note that the critical features of the solution—the reflected shock and contact discontinuity—are lost with this coarse mesh. A refined/unrefined mesh obtained with parameters $\alpha = 0.10$, $\beta = 0.10$ is shown in Fig. 12 together with a greatly improved density approximation. In these and subsequent calculations, a CFL number of 0.5 and a Lapidus constant of 1.0 were employed. Results of a node redistribution scheme for the coarse mesh are shown in Fig. 13. In these results, ten iterations of the node redistribution algorithm were used.

6.3. Supersonic flow in a wind tunnel with a step

The steady-state solution of the problem of a wind tunnel with a step introduced into the flow is next considered. The inflow Mach number was selected $M_\infty = 3.0$ and $\gamma = 1.40$. The initial coarse mesh is shown in Fig. 14 with the corresponding density profiles, and results of the adaptive refinement/unrefinement scheme with $\alpha = 0.15$ and $\beta = 0.20$ are shown in Fig. 15. The mesh refinement algorithm was also used, with the mesh and density profiles obtained after 10 iterations shown in Fig. 16. We see that the adaptive scheme captures well the features of the flow including the contact discontinuity at the top near the point of reflection of the bow shock. However, some oscillations are present downstream, and they are believed to be due to the nonmonotonicity of the solution algorithm. The results presented for the refinement/unrefinement procedure have been constrained by a maximum number of 2000
Fig. 10. Reflecting shock problem. (a) Mesh and (b) density contours obtained after 10 applications of the mesh redistribution algorithm.

Fig. 11. NACA 0012 airfoil in supersonic wind tunnel. (a) Initial mesh and (b) density contours.
Fig. 12. NACA 0012 airfoil in supersonic wind tunnel. (a) Mesh and (b) density contours obtained with one level of refinement ($\alpha = 0.10, \beta = 0.10$).

Fig. 13. NACA 0012 airfoil in supersonic wind tunnel. (a) Mesh and (b) density contours obtained after 10 applications of the mesh redistribution algorithm.
Fig. 14. Supersonic flow in a wind tunnel with a step. (a) Initial mesh and (b) density contours.

Fig. 15. Supersonic flow in a wind tunnel with a step. (a) Mesh and (b) density contours obtained with one level of refinement ($\alpha = 0.15, \beta = 0.20$).
Fig. 16. Supersonic flow in a wind tunnel with a step. (a) Mesh and (b) density contours obtained after 10 applications of the mesh redistribution algorithm.

Fig. 17. Supersonic flow over a 20° ramp. (a) Initial mesh and (b) density contours.
nodes or 2000 elements that can be allowed. In the refined mesh shown, this constraint has been achieved.

6.4. Supersonic flow over a 20° ramp

We next consider the steady supersonic flow through a conduit with a 20° ramp. The gas (with $\gamma = 1.4$) enters as a uniform $M = 3.0$ flow through the left side of the ramp and a shock develops at the ramp root. A coarse initial mesh and the computed density contours are illustrated in Fig. 17. For this problem, a reasonably good indication of the orientation of the shock is obtained.

Adaptive mesh results are shown in Figs. 18 and 19 for choices of the parameters of $\alpha = 0.20$ and $\beta = 0.50$ with one and two levels of refinement, respectively. Notice that spurious oscillations at the outflow boundary above the ramp root, due to the hourglass oscillations described in Section 3, cause unnecessary refinements in this region. Similarly, in regions between the shock and the ramp, some unnecessary refinement results from oscillations in the numerical solution. Nevertheless, striking improvement in the quality of the solution is seen to result from the refinement procedure.

In this particular problem, the node redistribution algorithm works remarkably well. A computed distorted coarse mesh, obtained after ten applications of the node redistribution algorithms, is shown in Fig. 20 with the resulting density contours.

6.5. Blunt leading edge of 8' HTT panel holder in hypersonic flow

The problem of the blunt leading edge of the 8' HTT panel holder in a supersonic flow field with free-stream Mach number $M_{\infty} = 6.57$, $\gamma = 1.38$, and 0° angle of attack was solved to obtain the steady-state solution. This problem has also been studied by Bey et al. [6].

A coarse mesh solution is indicated in Fig. 21 and an adaptively refined/unrefined mesh and solution, obtained for $\alpha = 0.05$ and $\beta = 0.15$, are shown in Fig. 22. A distorted mesh and corresponding density map are indicated in Fig. 23. In this particular problem, neither the $h$-method nor the $r$-method gave particularly good results, as a poor approximation of the solution between the shock and blunt body results from spurious oscillations in the basic time-marching algorithm. In the case of mesh adaptation using redistribution, the solution actually diverges after four passes through the adaptive scheme due to the badly graded (hourglassed) mesh produced from the oscillations of the adaptive scheme downstream of the shock.

6.6. Transient adaptive solution for supersonic flow over a 20° ramp

In all the examples presented above, a time-accurate time stepping scheme is used, but the adaptive scheme was not used stepwise for the transient solution since our primary interest was to increase accuracy in the steady-state solution. The adaptive method used to track transient fronts is described as follows:

**Step 1.** Choose a structured mesh with the finest mesh size to be allowed in the calculation to be the initial mesh. This is done to avoid large variations of the time step during the time stepping.
Fig. 18. Supersonic flow over a 20° ramp. (a) Mesh and (b) density contours obtained with one level of refinement ($\alpha = 0.20$, $\beta = 0.50$).

Fig. 19. Supersonic flow over a 20° ramp. (a) Mesh and (b) density contours obtained with two levels of refinement ($\alpha = 0.20$, $\beta = 0.50$).
Fig. 20. Supersonic flow over a 20° ramp. (a) Mesh and (b) density contours obtained after 10 applications of the mesh redistribution algorithm.

Fig. 21. Blunt leading edge in hypersonic flow. (a) Initial mesh and (b) density contours.
Fig. 22. Blunt leading edge in hypersonic flow field. (a) Mesh and (b) density contours obtained with one level of refinement ($\alpha = 0.05$, $\beta = 0.15$).

Fig. 23. Blunt leading edge in hypersonic flow. (a) Mesh and (b) density contours obtained after 4 applications of the mesh redistribution.
Step 2. Every $N$ time steps ($N = 50$ in the present problem) go through the refinement/unrefinement process (only unrefinement after the first $N$ time steps).

The above adaptive strategy was employed to solve the problem of a $20^\circ$ ramp which is suddenly introduced in a supersonic flow field with $M_* = 3.0$, $\gamma = 1.40$. The solution was integrated to a steady state, and it is demonstrated that the mesh adapts to the shock front as the shock front moves from its initial to its steady-state position.

The initial coarse mesh is shown in Fig. 24 and the evolution of a refined/unrefined mesh for various time intervals is illustrated in Figs. 25–29. The refinement parameters used were $\alpha = 0.5$ and $\beta = 0.25$, and a total of 250 time steps were used to track the solution from its initial to the final steady state. The final steady result is similar to that obtained earlier and shown in Figs. 18 and 19.

Acknowledgment

The bulk of the work reported here is a component of a program for the development of finite element methodologies for aerothermal heating analysis and related computational fluid/structural dynamics problems supported by the NASA Langley Research Center. The support of Mr. A. Weiting and Dr. G. Ohlsen of NASA is gratefully acknowledged. Much of the preliminary work on this effort was done in collaboration with Lawrence W. Spradley of the Lockheed Missile and Space Co., Huntsville Research and Engineering Center. The adaptive data structures discussed in Section 5 were developed in a study of adaptive finite element methods in solid mechanics sponsored by the Office of Naval Research under Contract N00014-84-K-0409. The support of some of the effort of P. Devloo and T. Strouboulis through Contract NAS8-36647 with the NASA Marshall Space Flight Center is also acknowledged.
Fig. 25. Transient transonic flow over a 20° ramp. (a) Mesh and (b) density contours after 50 time steps ($\alpha = 0.05$, $\beta = 0.25$).

Fig. 26. Transient supersonic flow over a 20° ramp. (a) Mesh and (b) density contours after 100 time steps ($\alpha = 0.05$, $\beta = 0.25$).
Fig. 27. Transient supersonic flow over a 20° ramp. (a) Mesh and (b) density contours after 150 time steps ($\alpha = 0.05, \beta = 0.25$).

Fig. 28. Transient supersonic flow over a 20° ramp. (a) Mesh and (b) density contours after 200 time steps ($\alpha = 0.05, \beta = 0.25$).
Fig. 29. Transient supersonic flow over a 20° ramp. (a) Mesh and (b) density contours after convergence to steady state ($\alpha = 0.05$, $\beta = 0.25$).

References


J.T. Oden et al., *Adaptive FEM for inviscid compressible flow*


