JOURNAL OF COMPUTATIONAL PHYSICS 120, 278–304 (1995)

An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions*

RICHARD B. PEMBER AND JOHN B. BELL

Lawrence Livermore National Laboratory, P.O. Box 808, L-316, Livermore, California 94550

PHILLIP COLELLA

Department of Mechanical Engineering, University of California, Berkeley, Berkeley, California 94720

WILLIAM Y. CRUTCHFIELD AND MICHAEL L. WELCOME

Lawrence Livermore National Laboratory, P.O. Box 808, L-316, Livermore, California 94550

Received November 19, 1993; revised November 28, 1994

In this paper we describe an adaptive Cartesian grid method for modeling time-dependent inviscid compressible flow in irregular regions. In this approach a body is treated as an interface embedded in a regular Cartesian mesh. The single grid algorithm uses an unsplit second-order Godunov algorithm followed by a corrector applied to cells at the boundary. The discretization near the fluid-body interface is based on a volume-of-fluid approach with a redistribution procedure to maintain conservation while avoiding time step restrictions arising from small cells where the boundary intersects the mesh. The single grid Cartesian mesh integration scheme is coupled to a conservative adaptive mesh refinement algorithm that selectively refines regions of the computational grid to achieve a desired level of accuracy. Examples showing the results of the combined Cartesian grid integration/adaptive mesh refinement algorithm for both two- and three-dimensional flows are presented. © 1995 Academic Press, Inc.

1. INTRODUCTION

The modeling of inviscid compressible flows in irregular regions is often required in engineering applications. In this paper we present an adaptive Cartesian grid method for solving the time-dependent Euler equations in irregular domains for rectangular and cylindrical coordinates in two space dimensions and for rectangular coordinates in three space dimensions. In this method, the fluid-body interface is represented on a regular Cartesian grid in a volume-of-fluid manner. The basic integration scheme is an explicit two-step method. In the first step, a higher-order Godunov method is used in which the fluid-body boundary is "ignored." In the second step, a correction is computed in the irregular cells, i.e., where the fluid–body boundary intersects the mesh. A redistribution procedure is used in this step to maintain conservation while avoiding time step restrictions arising from the small irregular cells. The single grid integration scheme is coupled to an adaptive mesh refinement algorithm that selectively refines regions of the computational grid to achieve a desired level of accuracy while maintaining conservation. We present examples showing the results of the combined Cartesian grid integration/adaptive mesh refinement algorithm for both two- and three-dimensional and axisymmetric flows. The numerical results demonstrate that the algorithm is an effective means of calculating inviscid compressible flows in domains with complicated geometric features. Numerical results suggest that for smooth flows the method is globally second-order accurate and first-order accurate at the fluid-body boundary.

There are three general approaches for modeling fluid flows of any type in domains with irregular boundaries. The first approach is to use body-fitted structured or block structured grids [34, 33, 14, 9, 22, 6]. Thompson *et al.* [63] review the state of the art in body-fitted grid methods through 1980. A second approach is to use unstructured body-fitted grids of triangular or tetrahedral cells [36, 35, 43, 44]. Mavriplis [46] discusses some of the current issues regarding these methodologies.

A third approach is to maintain a uniform rectangular compu-

* This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. Support under Contract W-7405-Eng-48 was provided by the Applied Mathematical Sciences Program and the HPCC Grand Challenge Program of the Office of Scientific Computing at DOE and by the Defense Nuclear Agency under IACRO 93-817 and IACRO 94-831. Professor Colella was supported at U.C. Berkeley by DARPA and the National Science Foundation under Grant DMS-8919074 by a National Science Foundation Presidential Young Investigator award under Grant ACS-8958522, and by the Department of Energy High Performance Computing and Communications Program under Grant DE-FG03-92ER25140. The U.S. Government's right to retain a nonexclusive royalty-free license in and to the copyright covering this paper, for governmental purposes, is acknowledged.

0021-9991/95 $12.00

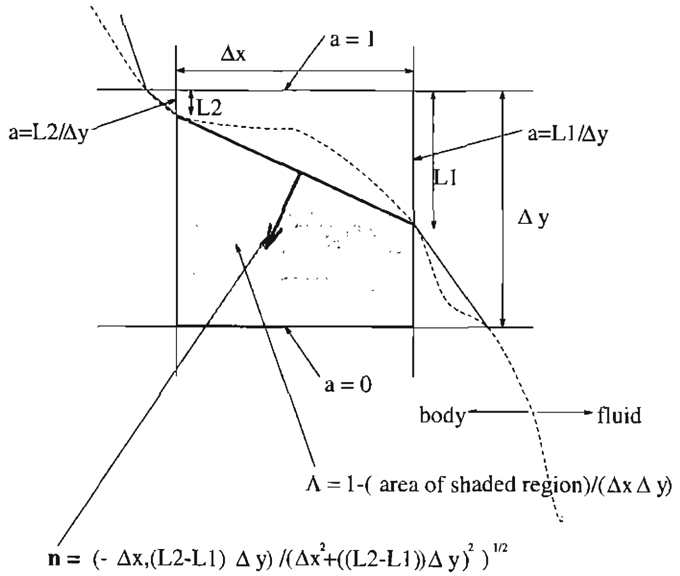

FIG. 1. Cartesian grid representation of geometry in a two-dimensional computational cell. The fluid–body interface is denoted by the dashed line. In the example shown, the boundary is approximated by a single line segment.

tational mesh and treat the geometric description as a specialized boundary embedded in the mesh (see Figs. 1 and 2). The methods using this approach are generally called Cartesian grid or Cartesian mesh methods. There are several reasons why these methods are a useful alternative to body-fitted grid methods. They are generally implemented as an extension of an existing scheme for regular domains on rectangular meshes.

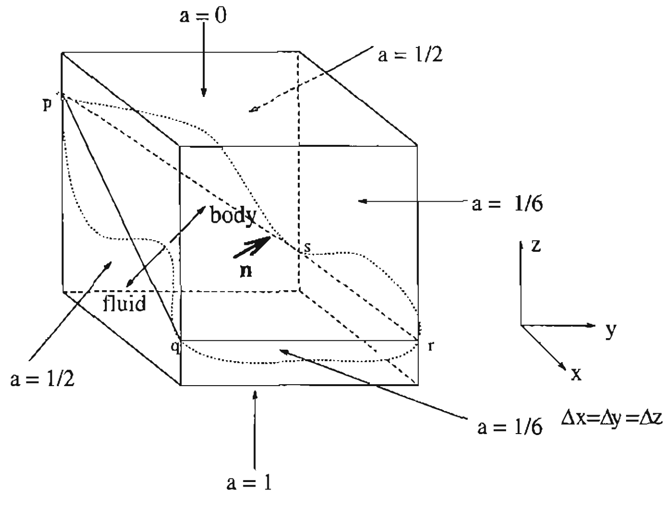

FIG. 2. Cartesian grid representation of geometry in a three-dimensional computational cell. The fluid–body interface is the surface bounded by the dotted line. In the example shown, the boundary is approximated by the non-planar quadrilateral *pqrs*. The volume fraction is Λ = (volume under *pqrs*)/($\Delta x\ \Delta y\ \Delta z$). In this particular example, the area fractions of the top, bottom, left, right, front, and back faces are 0, 1, $\frac{1}{2}$, $\frac{1}{6}$, $\frac{1}{6}$, and $\frac{1}{2}$, respectively. The effective surface normal is $n = (\frac{1}{2} - \frac{1}{6}, \frac{1}{2} - \frac{1}{6}, 0 - 1)/((\frac{1}{2} - \frac{1}{6})^2 + (\frac{1}{2} - \frac{1}{6})^2 + (0 - 1)^2)^{1/2}$.

Therefore, implementation is a simpler task because there are no metric terms and the data structures are simpler. They also avoid the difficulties that structured body-fitted grid schemes have with interpolation [49]. Finally, the specification of the geometric description needed by a Cartesian grid method is theoretically easier than the generation of a body-fitted grid [46, 69] because it involves only a set of cells of co-dimension one with respect to the problem domain, i.e., the cells at the fluid–body interface.

Cartesian grid methods were first used in the context of solving the equations of transonic potential flow by Purvis and Burkhalter [55]; see also [65, 37, 60, 68]. The use of these methods to calculate steady compressible flows was first investigated by Clarke *et al.* [19]; see also [29, 28, 27, 49]. The solvers described in these references are based on central-differencing techniques. Zeeuw and Powell [69] discuss an adaptive steady Euler solver which uses upwind differencing. Coirier and Powell [20] performed accuracy and efficiency assessments of the method in [69], including a comparison of two ways of handling the problem of arbitrarily small irregular cells: advancing the small cells at smaller time steps and cell merging of smaller cells with full cells. They found that both methods were globally second-order accurate and greater than first-order but less than second-order in the irregular cells and that the cell merging technique was more efficient without loss of accuracy.

In this paper we consider an explicit Cartesian grid method for the time-dependent Euler equations in irregular domains. The major issues in designing such a method are how to maintain accuracy, stability, and conservation in the irregular cells at the fluid–body interface while using a time step restricted by CFL considerations using the regular cell size alone, and, if applicable, how to couple the method to an adaptive mesh refinement scheme. Noh [50] did early work in this area in which he used both cell merging techniques and a redistribution methodology similar to that in [15] only in the most general terms. More recently, there has been additional work on developing Cartesian grid methods in two space dimensions. LeVeque [40, 41] and Berger and LeVeque [10] have developed explicit methods which use the large time step approach developed by LeVeque [39] to overcome the small cell problem. Berger and LeVeque [11, 12] have also studied approaches in which the small cell problem is avoided by the use of a rotated difference scheme in the cells cut by the fluid–body interface. Both the large time step and the rotated difference schemes are globally second-order and better than first-order but less than second-order at the fluid–body interface. Chiang *et al.* [16] use cell merging techniques to overcome the small cell problem. Bayyuk *et al.* [2] adapt cell-merging techniques to a method for moving and deforming bodies. Quirk [58, 56] also uses a cell merging methodology which he has coupled to his variation [57] of the adaptive mesh refinement algorithm of Berger and Collella [8]. Results for this method suggest it is globally second-order accurate and first-order at the boundary. Two other approaches for unsteady compressible flow are based on fluxvector splitting (Choi and Grossman [17]) and state-vector splitting (Öksüzoğlu [52], Gooch and Öksüzoğlu [30]). Neither of these approaches avoids the small cell problem; in other words, the time step used by these methods is restricted by the size of the small cells. The state-vector splitting approach, however, is promising because it extends to the Navier–Stokes equations.

The method presented here uses a different approach for handling irregular cells which is based on ideas previously developed for shock tracking by Chern and Colella [15] and by Bell, Colella, and Welcome [7]. In this approach the boundary is viewed as a "tracked front" in a regular Cartesian grid with the fluid dynamics at the boundary governed by the boundary conditions of a stationary reflecting wall. The geometric description is specified by volume and area fractions for each cell, i.e., the fraction of the volume of the cell that is inside the flow and the fraction of the area of each face of the cell that is inside the fluid. The basic integration scheme is an explicit two-step method. In the first step, a reference state is computed using fluxes generated by a higher-order Godunov method in which the fluid-body boundary is "ignored." In this second step, a correction is computed to the state in each irregular cell. A stable, but nonconservative, portion of this correction is applied to the irregular cell. Conservation is then maintained by a variation of the algebraic redistribution algorithm in [15] which distributes the remainder of the correction to those regular and irregular cells which are immediate neighbors of the irregular cell. The redistribution procedure allows the scheme to use time steps computed from CFL considerations on the uniform grid alone. The basic integration methodology for a single grid is coupled to the adaptive mesh refinement (AMR) algorithm of Bell *et al.* [3], which is an extension to three space dimensions of the AMR algorithm of Berger and Colella [8]. The AMR algorithm is modified to enforce a consistent description of the geometry across levels of refinement and to account for the additional transport of conserved quantities across coarse–fine grid boundaries arising from redistribution. The algorithm is formulated for rectangular coordinates in two and three dimensions and cylindrical coordinates in two dimensions.

We now outline the remainder of the paper. In the next section we discuss the basic Cartesian grid integration method for embedded fluid–body boundaries. In Section 3, we describe the coupling of the basic integration algorithm to the adaptive mesh refinement algorithm of Bell *et al.* [3]. Section 4 of the paper presents numerical examples for both two- and three-dimensional flows. Section 5 discusses conclusions, extensions of the methodology, and further work in this area.

In this paper we do not consider detailed general approaches for generating Cartesian grid descriptions of the fluid–body interface. We note that the generation of the Cartesian grid representation of a fluid–body interface has not been automated in three dimensions to the same extent as the generation of body-fitted grids, but progress is being made in this area (Melton *et al.* [47]). Procedures for the two-dimensional case are described in [2, 47, 58, 56, 69].

2. SINGLE GRID ALGORITHM

In this section we describe the basic Cartesian grid integration algorithm. Although the methodology is applicable to other hyperbolic systems of conservation laws, such as those arising in magnetohydrodynamics, we restrict the discussion in this paper to gas dynamics. The methodology has been developed for Cartesian grids in two and three dimensions and for cylindrical coordinates in two dimensions. For clarity of exposition we describe the method for the three-dimensional case. The restriction to two dimensions is straightforward, and the modification for cylindrical coordinates is sketched in Section 2.7. Hence, we want to solve

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} + \frac{\partial H(U)}{\partial z} = 0, \tag{2.1}$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ \rho u E + u p \end{pmatrix},$$

$$G(U) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ \rho v E + v p \end{pmatrix}, \quad H(U) = \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ \rho w E + w p \end{pmatrix},$$

$$E = e + \tfrac{1}{2}(u^2 + v^2 + w^2), \quad p = p(\rho, e).$$

In the remainder of this section, we first discuss the description of geometry used by the algorithm. An overview of the single grid algorithm is then given, followed by a review of the higher-order Godunov method [22, 61]. We then discuss each of the steps in the single grid Cartesian mesh algorithm. In discussing the algorithm in three dimensions, we assume a uniform computational grid with cell widths Δx, Δy, and Δz indexed by i, j, and k. The algorithm for cylindrical coordinates is presented in Section 2.7.

Without the use of special data structures in the implementation of the algorithm, there are certain limitations on the complexity of the geometry. These limitations are discussed in Sections 2.1, 2.4.1, and 2.6.1. The authors are working on an implementation which overcomes these limitations; see the discussion in Section 5.

2.1. Geometry: Terminology and Notation

The geometric description of the domain needed by the integration algorithm is provided by volume fractions and area fractions; see Figs. 1 and 2. The volume fraction $\Lambda_{i,j,k}$ is the fraction of the computational cell $\Delta_{i,j,k}$ that is inside the flow domain. We let $\Delta^f_{i,j,k}$ denote the intersection of a computational cell $\Delta_{i,j,k}$ with the fluid portion of the computational domain. Then

$$\Lambda_{i,j,k} = \int_{\Delta^f_{i,j,k}} dx\, dy\, dz/\Delta x\, \Delta y\, \Delta z. \tag{2.2}$$

Thus, $\Lambda_{i,j,k}$ equals 1 for cells completely inside the flow domain ("fluid" cells), 0 for cells completely outside the flow domain ("body" cells), and an appropriate fraction for cells which have one or more faces that intersect the fluid–body interface ("mixed" cells). We also designate a cell with $\Lambda = 1$ with a face aligning with the fluid–body interface as a mixed cell. The area fractions $a_{i+1/2,j,k}$, $a_{i,j+1/2,k}$, and $a_{i,j,k+1/2}$ specify the fraction of the area of each of the faces of each cell that lies on or inside the flow domain (where $i + \frac{1}{2}$, j, k refers to the face common to cells $\Delta_{i,j,k}$, $\Delta_{i+1,j}k$, etc). Thus,

$$a_{i+1/2,j,k} = \int_{(\partial\Delta^f_{i,j,k})\cap((x,y,z):x=x_{i+1/2})} dy\, dz/\Delta y\, \Delta z, \tag{2.3}$$

where $x_{i+1/2}$ is the value of x at edge $i + \frac{1}{2}$. $a_{i,j+1/2,k}$ and $a_{i,j,k+1/2}$ are defined similarly. By convention, the area fraction of a cell face separating cells with volume fractions of 1 and 0 is 0.

In a mixed cell, the geometric information required by the integration algorithm also includes an effective surface normal and an effective frontal area, i.e., the area of the surface formed by the intersection of the fluid and the body in the cell. Expressions for both of these quantities in terms of area fractions can be derived by integrating the gradient of a constant over the portion of the cell inside the fluid and applying the divergence theorem. For any constant ϕ we have

$$0 = \int_{\Delta^f_{i,j,k}} \nabla\phi\, dx\, dy\, dz = \int_{\partial(\Delta^f_{i,j,k})} \phi \boldsymbol{n}\, dS.$$

In particular, if $\phi = 1$, we obtain the following formula for the unit surface normal $\boldsymbol{n}^f_{i,j,k}$ and the frontal area $A^f_{i,j,k}$ in a mixed cell:

$$\begin{aligned} A^f_{i,j,k}\boldsymbol{n}^f_{i,j,k} &= \Delta y\, \Delta z(a_{i-1/2,j,k} - a_{i+1/2,j,k})\boldsymbol{i} \\ &\quad + \Delta x\, \Delta z(a_{i,j-1/2,k} - a_{i,j+1/2,k})\boldsymbol{j} \\ &\quad + \Delta x\, \Delta y(a_{i,j,k-1/2} - a_{i,j,k+1/2})\boldsymbol{k}. \end{aligned} \tag{2.4}$$

See Fig. 1 for a two-dimensional example. The above approach is unnecessarily complicated for two space dimensions. However, in three dimensions, it is difficult, and perhaps not desirable, to describe the fluid–body interface on a rectangular mesh as piecewise planar. The above expression for the surface normal reduces to the correct answer in the case of a planar interface and provides a consistent way of computing an effective normal when the interface is not planar; in particular, the use of (2.4) is essential for freestream preservation. See Fig. 2 for an example.

In practice, one does not directly use (2.2) and (2.3) to compute volume and area fractions, but rather some approximation. In this paper, general approaches for generating the volume fractions and area fractions are not considered. Some approaches we have employed under certain restrictive conditions are the following. In two dimensions, if the fluid–body interface crosses the cell boundary in exactly two places on distinct edges, we approximate the interface by the line segment defined by these two endpoints (Fig. 1). In three dimensions, we have used a similar approach. We assume that the intersection of the fluid–body interface with the computational cell is a connected set (Fig. 2). We also assume that the intersection of the boundary with any cell face is a connected planar region. Under these assumptions, we can use linear approximations of the intersections of the interface with each cell face, a nonplanar quadrilateral representation of the fluid–body interface throughout the cell, and integrate to get the area and volume fractions.

Our current implementation has the requirement that a meaningful surface normal be defined in each computational cell, and, consequently, it does not in general account for long thin sections of body (Fig. 3f).

2.2. Single Grid Algorithm: Summary

The single grid method consists of the following five steps:

(1) the computation of the cell edge fluxes $F^{n+1/2}_{i+1/2,j,k}$, $G^{n+1/2}_{i,j+1/2,k}$, and $H^{n+1/2}_{i,j,k+1/2}$, using a higher-order Godunov integration algorithm for rectangular grids [22, 61] which ignores the presence of irregular geometry. These are computed at all edges of fluid as well as mixed cells, even if the cell edges lie partially or wholly within the body.

(2) the computation of the reference state $U^{n+1,\text{ref}}_{i,j,k}$ in all fluid and mixed cells by

$$\begin{aligned} U^{n+1,\text{ref}}_{i,j,k} = U^n_{i,j,k} &+ \frac{\Delta t}{\Delta x}(F^{n+1/2}_{i-1/2,j,k} - F^{n+1/2}_{i+1/2,j,k}) \\ &+ \frac{\Delta t}{\Delta y}(G^{n+1/2}_{i,j-1/2,k} - G^{n+1/2}_{i,j+1/2,k}) \\ &+ \frac{\Delta t}{\Delta z}(H^{n+1/2}_{i,j,k-1/2} - H^{n+1/2}_{i,j,k+1/2}). \end{aligned} \tag{2.5}$$

Except for mixed cells and fluid cells that are immediate neighbors of mixed cells, we set $U^{n+1}_{i,j,k} = U^{n+1,\text{ref}}_{i,j,k}$.

(1) the computation of an adjustment, $\delta M_{i,j,k}$, in the mixed cells so that

$$U^{n+1}_{i,j,k} = U^{n+1,\text{ref}}_{i,j,k} + \frac{\delta M_{i,j,k}}{\Lambda_{i,j,k}\Delta x\, \Delta y\, \Delta z} \tag{2.6}$$

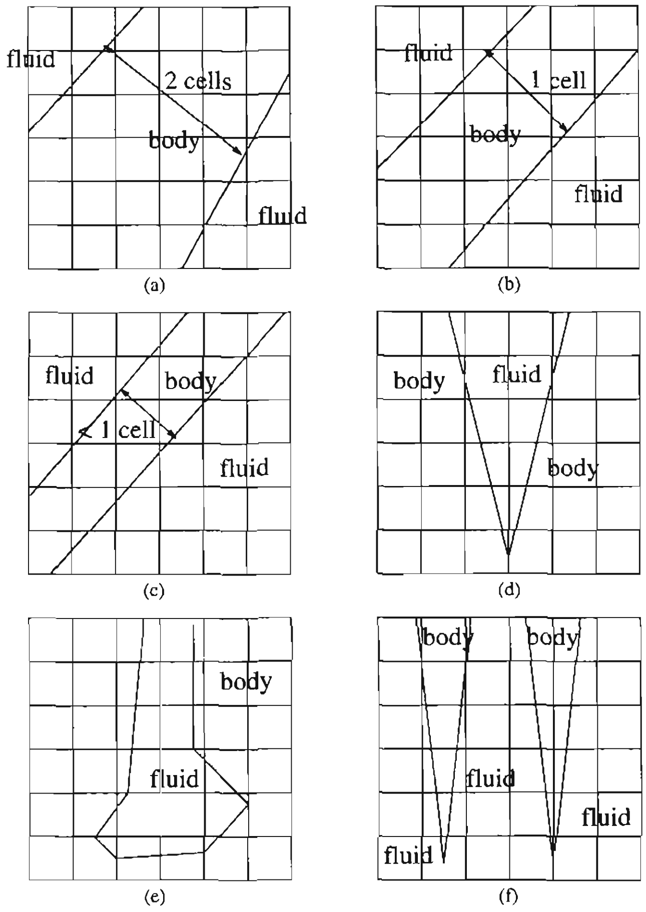

FIG. 3. Limits on geometric complexity. The geometry in (a) is not problematic. (b) is problematic with respect to the extended state formulation for computing the reference state (Section 2.4) but not the thin-wall algorithm (Section 1.4.1). (c) is disallowed because of the redistribution algorithm (Section 2.6). The redistribution algorithm may have difficulties with the geometries in (d) and (e) (Section 2.6.1). In (f), a unique surface normal can be defined in each cell only for the body on the right (Section 2.1).

is the correct and conservative but potentially unstable update (because $\Lambda_{i,j,k} < 1$) for $U^{n+1}_{i,j,k}$.

(4) the computation of the preliminary stable but nonconservative update in the mixed cells by

$$U^{n+1,p}_{i,j,k} = U^{n+1,\mathrm{ref}}_{i,j,k} + \frac{\delta M_{i,j,k}}{\Delta x\, \Delta y\, \Delta z}. \tag{2.7}$$

(5) the reestablishment of conservation by redistributing the amount by which conservation is violated, $(1 - \Lambda_{i,j,k})\delta M_{i,j,k}$, in a mass-weighted manner to fluid and mixed cells among the 26 neighbors of the mixed cell $\Delta_{i,j,k}$.

2.3. Review: Higher-Order Godunov Method

Before discussing the details of the single grid method, we briefly review the higher order Godunov method of Colella [22] on rectangular domains in the absence of irregular boundaries and its extension to three dimensions by Saltzman [61]. For the sake of brevity, we present the method for a scalar conservation law in two dimensions, discuss extending the method to three dimensions, and then we discuss the handling of systems of equations. We do not consider here the issues of fourth-order spatial slopes or additional numerical dissipation mechanisms; see [21, 25, 61].

We first consider the method for a scalar conservation law in two space variables,

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0, \tag{2.8}$$

where $u, a(u) = \partial f(u)/\partial u, b(u) = \partial g(u)/\partial u \in \boldsymbol{R}$. The quasilinear form of (2.8) is then

$$\frac{\partial u}{\partial t} + a(u)\frac{\partial u}{\partial x} + b(u)\frac{\partial u}{\partial y} = 0. \tag{2.9}$$

We assume a uniform computational grid with cell widths Δx and Δy indexed by i and j. To advance the solution at time t_n, we use a time step Δt which satisfies the Courant–Friedrichs–Lewy (CFL) stability condition,

$$\Delta t = \sigma \min_{i,j} \left(\min \left(\frac{\Delta x}{a(u_{i,j})} \right), \left(\frac{\Delta y}{b(u_{i,j})} \right) \right), \tag{2.10}$$

where σ is a constant satisfying $\sigma < 1$. The algorithm consists of four general steps:

(1) the calculation of limited central difference approximations, $\Delta_x u$ and $\Delta_y u$ in each cell i, j, where $\Delta_x u/\Delta x$ and $\Delta_y u/\Delta y$ are approximations of $\partial u/\partial x$ and $\partial u/\partial y$ at cell centers.

(2) the calculation of time-centered left and right states, $u^{n+1/2}_{i+1/2,j,L}$ and $u^{n+1/2}_{i+1/2,j,R}$, at all x-cell faces and bottom and top states, $u^{n+1/2}_{i,j+1/2,B}$ and $u^{n+1/2}_{i,j+1/2,T}$, at all y-cell faces. Since these values are calculated using (2.9), i.e., the linearized characteristic form of (2.8), we refer to this step as the *characteristic tracing* step.

(3) the evaluation of numerical fluxes $f^{n+1/2}_{i+1/2,j}$ and $g^{n+1/2}_{i,j+1/2}$ at x- and y-cell faces, respectively, using the solution of Riemann problems. At each x-cell face, we solve the Riemann problem defined by

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{2.11}$$

and the left and right states $u^{n+1/2}_{i+1/2,j,L}$ and $u^{n+1/2}_{i+1/2,j,R}$, evaluate that solution at $x/t = 0$ to obtain $u^{n+1/2}_{i+1/2,j}$, and compute $f^{n+1/2}_{i+1/2,j} = f(u^{n+1/2}_{i+1/2,j})$. At each y-cell edge, $g^{n+1/2}_{i,j+1/2}$ is found in a similar manner.

(4) the computation of $u^{n+1}_{i,j}$ by conservative differencing,

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\Delta t}{\Delta x}(f_{i-1/2,j}^{n+1/2} - f_{i+1/2,j}^{n+1/2}) + \frac{\Delta t}{\Delta y}(g_{i,j-1/2}^{n+1/2} - g_{i,j+1/2}^{n+1/2}).$$

Step (4) is self-explanatory. The solution of the Riemann problem in step (3) for a scalar conservation law is discussed many places in the literature; for example, see LeVeque [42].

Step (1) has the following details. Define $\Delta_{x,-}u$, $\Delta_{x,+}u$, and $\Delta_{x,c}u$ by

$$\Delta_{x,-}u = u_{i,j} - u_{i-1,j}$$
$$\Delta_{x,+}u = u_{i+1,j} - u_{i,j}$$
$$\Delta_{x,c}u = \tfrac{1}{2}(\Delta_{x,+}u + \Delta_{x,-}u).$$

$(\Delta_x u)_{i,j}$ is then given by

$$\begin{aligned}(\Delta_x u)_{i,j} &= \min(|(\Delta_{x,c}u)_{i,j}|, 2|(\Delta_{x,-}u)_{i,j}|, 2|(\Delta_{x,+}u)_{i,j}|) \\ &\quad \times \operatorname{sgn}((\Delta_{x,c}u)_{i,j}) \quad \text{if } (\Delta_{x,-}u)_{i,j}(\Delta_{x,+}u)_{i,j} > 0 \qquad (2.12) \\ &= 0, \quad \text{otherwise.}\end{aligned}$$

$\Delta_y u$ is computed in a similar manner.

A summary of step (2) completes the description of the method. Using Taylor's theorem, the differential equation, and the values of $\Delta_x u$ we have

$$u_{i+1/2,j,L}^{n+1/2} = u_{i,j}^{n} + P_L\left(\frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}a(u_{i,j}^{n})\right)(\Delta_x u)_{i,j}\right) - \frac{\Delta t}{2}\left(\frac{\partial g(u)}{\partial y}\right)_{i,j}^{n},$$

where $P_L(w) = w$ if $a(u_{i,j}^n) > 0$ and 0 otherwise. We refer to the term $(\partial g(u)/\partial y)_{i,j}^n$ as a "transverse derivative." We approximate $(\partial g(u)/\partial y)_{i,j}^n$ by

$$\left(\frac{\partial g(u)}{\partial y}\right)_{i,j}^{n} \approx (g(RP_y(u_{i,j}^n, u_{i,j+1}^n)) - g(RP_y(u_{i,j-1}^n, u_{i,j}^n)))/\Delta y,$$

where $RP_y(u_B, u_T)$ denotes the solution of the Riemann problem defined by

$$\frac{\partial u}{\partial t} + \frac{\partial g(u)}{\partial y} = 0 \qquad (2.13)$$

and bottom and top states u_B and u_T evaluated at $y/t = 0$. Similarly,

$$u_{i+1/2,j,R}^{n+1/2} = u_{i+1,j}^{n} - P_R\left(\frac{1}{2}\left(1 + \frac{\Delta t}{\Delta x}a(u_{i+1,j}^{n})\right)(\Delta_x u)_{i,j}\right) - \frac{\Delta t}{2}\left(\frac{\partial g(u)}{\partial y}\right)_{i+1,j}^{n},$$

where $P_R(w) = w$ if $a(u_{i,j}^n) < 0$ and 0 otherwise, and where $(\partial g(u)/\partial y)_{i+1,j}^n$ is approximated in a manner similar to that above. The expressions for $u_{i,j+1/2,B}^{n+1/2}$ and $u_{i,j+1/2,T}^{n+1/2}$ are similar to those of $u_{i+1/2,j,L}^{n+1/2}$ and $u_{i+1/2,j,R}^{n+1/2}$.

From the above we see that the flux at an edge is dependent on the values of $u_{i,j}^n$ in the six cells nearest the cell edge and the values in additional cells which enter the central difference approximations of the spatial derivatives. Setting $\Delta_x u$ or $\Delta_y u$ to zero adds dissipation to the method. Setting both to zero results in a scheme that is a first-order version of Godunov's method that has corner coupling, so that it is stable for CFL numbers up to 1.0.

2.3.1. *Review of Higher-Order Godunov Method: Extension to Three Spatial Dimensions*

We now consider the method for a scalar conservation law in three space variables,

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} + \frac{\partial h(u)}{\partial z} = 0. \qquad (2.14)$$

Saltzman [61] discusses the method in much greater detail.

The algorithm again consists of the same four general steps. However, the tracing step is more complicated. We compute $u_{i+1/2,j,k,L}^{n+1/2}$, for example, by

$$u_{i+1/2,j,k,L}^{n+1/2} = u_{i,j,k}^{n} + P_L\left(\frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}a(u_{i,j,k}^{n})\right)(\Delta_x u)_{i,j}\right) - \frac{\Delta t}{2}\left(\frac{\partial g(u)}{\partial y}\right)_{i,j,k}^{n} - \frac{\Delta t}{2}\left(\frac{\partial h(u)}{\partial z}\right)_{i,j,k}^{n}.$$

The transverse derivative terms are approximated as before, i.e., by solving Riemann problems at the appropriate opposite faces, evaluating g and h using those solutions, and differencing the results in the appropriate manner. However, the two states used in defining each Riemann problem must now be found by using the characteristic tracing step from the two-dimensional algorithm in order that the eight corner neighboring cells of cell i, j, k are correctly coupled to the update of i, j, k. Hence, for example, the value of $u_{i+1/2,j,k,L}^{n+1/2}$ depends on the values of $u_{i,j,k}^n$ in the nine neighbors of i, j, k in the i-plane and the values in additional cells which enter the approximations of the spatial derivatives. In three dimensions, then, the flux at an edge is dependent on the values of $u_{i,j,k}^n$ in the 18 cells nearest the cell edge, plus the values in additional cells which enter the central difference approximations of the spatial derivatives. Setting $\Delta_x u$, $\Delta_y u$, or $\Delta_z u$ to zero adds dissipation to the method. Setting all three to zero results in a scheme that is a first-order version of Godunov's method that has corner coupling so that it is stable for CFL numbers up to 1.0.

2.3.2. *Review of Higher-Order Godunov Method: Extension to Systems*

The issue of extending the higher-order Godunov method to systems of equations is mostly one of extending the method to a hyperbolic system of equations in one space variable. For the details of extending the algorithm to systems, see the discussion by Colella [22] and Saltzman [61], the detailed description of the one-dimensional algorithm by Bell, Colella, and Trangenstein [5], and the concise review of the one-dimensional algorithm by Pember [54].

For the most part, any Riemann solver for systems can be used in conjunction with the higher-order Godunov method and the Cartesian grid method. Approximate Riemann solvers are discussed many places in the literature, including [5, 24, 53, 59]. In our implementation for gas dynamics with a general equation-of-state $p = p(\rho, e)$, we use a solver described in [23] which is based on the techniques in [5].

2.4. Calculation of Numerical Fluxes and Reference State

In the first two steps of the Cartesian grid algorithm for a single grid, we use the unsplit second-order Godunov integration algorithm for rectangular grids [22, 61] to compute fluxes for U, and then we use those fluxes to calculate the reference state $U_{i,j,k}^{n+1,\mathrm{ref}}$ by

$$U_{i,j,k}^{n+1,\mathrm{ref}} = U_{i,j,k}^{n} + \frac{\Delta t}{\Delta x}(F_{i-1/2,j,k}^{n+1/2} - F_{i+1/2,j,k}^{n+1/2}) + \frac{\Delta t}{\Delta y}(G_{i,j-1/2,k}^{n+1/2} - G_{i,j+1/2,k}^{n+1/2}) + \frac{\Delta t}{\Delta z}(H_{i,j,k-1/2}^{n+1/2} - H_{i,j,k+1/2}^{n+1/2}). \tag{2.15}$$

The higher-order Godunov algorithm is reviewed above in Section 2.3. We modify the higher-order Godunov method in three main ways in order to use it to compute the reference state in fluid and mixed cells in the Cartesian grid integration scheme:

1. We must be able to calculate the numerical fluxes $F_{i+1/2,j,k}^{n+1/2}$, $G_{i,j+1/2,k}^{n+1/2}$, and $H_{i,j,k+1/2}^{n+1/2}$ at edges separating mixed cells from body cells in order to use (2.15) as an update in mixed cells. The flux calculations at these edges require that meaningful values be defined in all body cells that are among the immediate 26 neighbors of a mixed cell. We call these extrapolated values "extended states."

The extended states are found in the following way. In each cell requiring an extended state, i.e., any body cell that is one of the immediate 26 neighbors of some mixed cell, we define

$$U_{i,j,k}^{\mathrm{ext}} = \frac{\sum_{(l,m,n)\in nbh(i,j,k)} \Lambda_{l,m,n} U_{l,m,n}^{n}}{\sum_{(l,m,n)\in nbh(i,j,k)} \Lambda_{l,m,n}}, \tag{2.16}$$

where nbh(i, j, k) represents the set of the 26 neighboring cells of $\Delta_{i,j,k}$. By definition we are guaranteed that at least one cell in nbh(i, j, k) has nonzero $\Lambda_{l,m,n}$ so that (2.16) is defined.

2. We do not want the extended states to affect the estimates of the spatial derivatives in the fluid and mixed cells, and we do not want any additional body cells to require extrapolated values. Therefore, any of the central difference approximations $\Delta_x U/\Delta x$, $\Delta_y U/\Delta y$, and $\Delta_z U/\Delta z$ to the spatial derivatives $\partial U/\partial x$, $\partial U/\partial y$, and $\partial U/\partial z$ in a given cell are set to zero if the computation of that central difference uses values from a body cell.

3. The time step used to advance the solution is based on a full cell CFL condition over the fluid cells, the mixed cells, and the body cells which are immediate neighbors of mixed or fluid cells. In calculating the stable time step, $U_{i,j,k}^{n}$ is used in the mixed and fluid cells, and $U_{i,j,k}^{\mathrm{ext}}$ is used in the body cells.

2.4.1. *Geometry Limitations and "Thin-Wall" Algorithm*

The current implementation of the algorithm described so far requires that genuinely separate regions of the fluid (for instance, in the case where a flat plate is embedded in the flow) be separated in any direction by at least two computational cells that are entirely contained in the body (see Figs. 3a,b). This requirement is due to the construction of the extended states. Because the central difference approximations to the spatial derivatives are set to zero if any body cells are involved in their computation, the extended states ultimately enter the algorithm through the solution of Riemann problems. If necessary, then, we can reduce the minimum number of cells needed to separate truly disjoint regions of the fluid to one by the following approach.

REFERENCE STATE CALCULATION: THIN-WALL ALGORITHM.

(1) Do not compute or use extended states.

(2) Use the higher order Godunov method to compute numerical fluxes as before, with the following change:

Set the solution of any Riemann problem at an edge separating a body cell from a mixed or fluid cell to be the value of the edge state corresponding to the non-body cell.

(3) Use (2.15) to compute the reference state.

We have found that the numerical solutions generated using the extended state formulation of the scheme are generally superior to those found using the thin-wall algorithm; hence, this latter approach should be used only when the constraints on the former are not satisfied.

Even if the thin-wall algorithm is employed, the current implementation still requires that truly separate regions of the flow be separated in any direction by at least one full body cell (see Figs. 3b,c). The restriction is due to the redistribution algorithm described below. See the discussion below in Section 2.6.

2.4.2. *Discussion of Choice of Reference State*

There is no rigorous justification for our choice of reference state. Our definition of reference state is motivated by three criteria:

(a) The overall method is freestream preserving, i.e., constant flows parallel to a surface are maintained.

(b) In each mixed cell, $U^{n+1,\text{ref}}_{i,j,k} \to U^{n}_{i,j,k}$ as $\Delta t \to 0$. (2.17)

(c) The solution in the irregular cells is influenced by the wall and the solution in the neighboring fluid cells in a reasonable manner, regardless of cell size.

Condition (2.17b) disallows approaches such as defining the reference state to be the average of the solution, or some intermediate update, in nearby fluid cells. Further, the seemingly reasonable alternative, namely, $U^{n+1,\text{ref}}_{i,j,k} = U^{n}_{i,j,k}$, is also disallowed. This choice is the one used in the algorithm of Chern and Colella [15] and satisfies (2.17a), (2.17b). However, this reference state naturally results in the computation of large values of δM in the mixed cells; recall the definition of δM in Section 2.2. Since only a fraction of the value of δM in a mixed cell can be included in the update to that cell, the values of the solution in the mixed cell do not change as rapidly as they should. The computed solution then has as an artifact that the fluid–body boundary appears to be exerting drag on the flow, and (2.17c) is not satisfied.

The method described in Section 2.4, on the other hand, computes a reference state that is a reasonable update to the solution in the mixed cells regardless of the volume fraction. There are undoubtedly other ways to compute the reference state so that (2.17) is satisfied. The method used here, however, is easy to implement and appears to be a simple, natural choice.

From the above discussion, we see that an alternative way to formulate (2.17c) is:

The components of $U^{n+1}_{i,j,k} - U^{n+1,\text{ref}}_{i,j,k}$ computed by the method should be small compared to the corresponding components of $U^{n+1,\text{ref}}_{i,j,k} - U^{n}_{i,j,k}$. (2.18)

Numerical tests (see Section 4.2.3) suggest that the method satisfies this condition.

2.5. Calculation of δM

In this step we compute an adjustment, $\delta M_{i,j,k}$, in the mixed cells so that

$$U^{n+1,*}_{i,j,k} = U^{n+1,\text{ref}}_{i,j,k} + \frac{\delta M_{i,j,k}}{\Lambda_{i,j,k}\,\Delta x\,\Delta y\,\Delta z}$$

is the correct and conservative but potentially unstable update for $U^{n+1}_{i,j,k}$.

To derive an expression for $\delta M_{i,j,k}$, in mixed cell i, j, k, we first apply the divergence theorem to each mixed cell to obtain the following conservative but potentially unstable update for the state in $\Delta_{i,j,k}$:

$$\begin{aligned} U^{n+1,*}_{i,j,k} = U^{n}_{i,j,k} + \frac{\Delta t}{\Delta x\,\Delta y\,\Delta z\,\Lambda_{i,j,k}} & \big(\Delta y\,\Delta z\,(a_{i-1/2,j,k}F^{n+1/2}_{i-1/2,j,k} - a_{i+1/2,j,k}F^{n+1/2}_{i+1/2,j,k}) \\ & + \Delta x\,\Delta z\,(a_{i,j-1/2,k}G^{n+1/2}_{i,j-1/2,k} - a_{i,j+1/2,k}G^{n+1/2}_{i,j+1/2,k}) \\ & + \Delta x\,\Delta y\,(a_{i,j,k-1/2}H^{n+1/2}_{i,j,k-1/2} - a_{i,j,k+1/2}H^{n+1/2}_{i,j,k+1/2}) \\ & - (F, G, H)_f A^f_{i,j,k}\boldsymbol{n}^f_{i,j,k}\big). \end{aligned} \tag{2.19}$$

$(F, G, H)_f$ represents a numerical approximation of the flux (F, G, H) at the fluid–body interface. $(F, G, H)_f$ is found by evaluating (F, G, H) with the solution of a Riemann problem at the fluid–body interface with left and right states given by $U^{n}_{i,j,k}$ and the state found by an odd reflection of the normal velocity with respect to $\boldsymbol{n}^f_{i,j,k}$ and an even reflection of the other state quantities. For the compressible Euler equations, the only nonzero elements of $(F, G, H)_f$ are the pressure terms appearing in the momentum flux elements of F, G, and H. We can then define $\delta M_{i,j,k}$ by

$$\begin{aligned} \delta M_{i,j,k} &= \Lambda_{i,j,k}\,\Delta x\,\Delta y\,\Delta z\,(U^{n+1,*}_{i,j,k} - U^{n+1,\text{ref}}_{i,j,k}) \\ &= \Lambda_{i,j,k}\,\Delta x\,\Delta y\,\Delta z\,(U^{n}_{i,j,k} - U^{n+1,\text{ref}}_{i,j,k}) \\ &\quad + \Delta t\,(\Delta y\,\Delta z\,(a_{i-1/2,j,k}F^{n+1/2}_{i-1/2,j,k} - a_{i+1/2,j,k}F^{n+1/2}_{i+1/2,j,k}) \\ &\quad + \Delta x\,\Delta z\,(a_{i,j-1/2,k}G^{n+1/2}_{i,j-1/2,k} - a_{i,j+1/2,k}G^{n+1/2}_{i,j+1/2,k}) \\ &\quad + \Delta x\,\Delta y\,(a_{i,j,k-1/2}H^{n+1/2}_{i,j,k-1/2} - a_{i,j,k+1/2}H^{n+1/2}_{i,j,k+1/2}) \\ &\quad - (F, G, H)_f A^f_{i,j,k}\boldsymbol{n}^f_{i,j,k}). \end{aligned} \tag{2.20}$$

To compute the term $(F, G, H)_f A^f_{i,j,k}\boldsymbol{n}^f_{i,j,k}$ appearing in (2.20) for the equations of gas dynamics, we proceed as follows. The equation to be solved in order to determine the pressure at the fluid body interface is

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial n} = 0, \tag{2.21}$$

where

$$U = \begin{pmatrix} \rho \\ \rho\hat{u} \\ \rho E \end{pmatrix}, \quad F(U) = \begin{pmatrix} \rho\hat{u} \\ \rho\hat{u}^2 + p \\ \rho\hat{u}E + \hat{u}p \end{pmatrix},$$

n denotes the direction normal to the fluid-body boundary, $\hat{u}$ is the velocity in that direction, $E = e + \frac{1}{2}\hat{u}^2$, and $p = p(\rho, e)$. To define the left and right states of the Riemann problem, we compute $\hat{u}^n_{i,j,k}$, the component of the velocity in cell i, j, k at time step n in the direction normal to the fluid-body interface:

$$\hat{u}^n_{i,j,k} = \mathbf{n}^f_{i,j,k} \cdot (u, v, w)^n_{k,j,k}. \tag{2.22}$$

We also compute

$$e^n_{i,j,k} = E^n_{i,j,k} - \tfrac{1}{2}((u^n_{i,j,k})^2 + (v^n_{i,j,k})^2 + (w^n_{i,j,k})^2). \tag{2.23}$$

The left and right states of the Riemann problem are then defined in terms of ρ, $\hat{u}$, and e by

$$\begin{aligned}(\rho, \hat{u}, e)_l &= (\rho^n_{i,j,k}, \hat{u}^n_{i,j,k}, e^n_{i,j,k})\\ (\rho, \hat{u}, e)_r &= (\rho^n_{i,j,k}, -\hat{u}^n_{i,j,k}, e^n_{i,j,k}).\end{aligned} \tag{2.24}$$

The pressure at the fluid–boundary interface, $p^f_{i,j,k}$, is found by solving the Riemann problem defined by Eqs. (2.21)–(2.24). $(F, G, H)_f A^f_{i,j,k}\mathbf{n}^f_{i,j,k}$ is then evaluated as

$$(F, G, H)_f A^f_{i,j,k}\mathbf{n}^f_{ijk} = \begin{pmatrix} 0 \\ \Delta y\, \Delta z\, (a_{i-1/2,j,k} - a_{i+1/2,j,k})p^f_{i,j,k} \\ \Delta x\, \Delta z\, (a_{i,j-1/2,k} - a_{i,j+1/2,k})p^f_{i,j,k} \\ \Delta x\, \Delta y\, (a_{i,j,k-1/2} - a_{i,j,k+1/2})p^f_{i,j,k} \\ 0 \end{pmatrix}. \tag{2.25}$$

2.6. Preliminary Update and Redistribution

In the fourth and fifth steps of the single grid algorithm, the values of δM are used to obtain a stable but nonconservative update in the mixed cells using (2.7); conservation is then reestablished via the redistribution algorithm.

The preliminary update for the solution in mixed cell i, j, k is given by (2.7), whereas the conservative update is (2.6). Hence, the total amount of "mass" that should be in i, j, k in order that conservation be maintained is

$$\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z U^{n+1,*}_{i,j,k},$$

whereas only

$$\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z U^{n+1,p}_{i,j,k}$$

is present after the preliminary update. (Throughout this discussion, the term "mass" refers to integrated values of the conserved quantities, i.e., mass, momentum, and energy rather than density, momentum density, and energy density.) Multiplying both (2.7) and (2.6) by $\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z$ and subtracting the former from the latter, then, results in the amount of "mass" lost by using (2.7):

$$\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z(U^{n+1,*}_{i,j,k} - U^{n+1,p}_{i,j,k}) = (1 - \Lambda_{i,j,k})\, \delta M_{i,j,k}.$$

In order to maintain conservation, then, we must distribute $(1 - \Lambda_{i,j,k})\, \delta M_{i,j,k}$ onto the grid. In the general case for a moving front [15, 7], we do this by decomposing these increments into characteristic variables and distributing them to nearby cells in a volume-weighted fashion. For the case of a stationary reflecting wall, the redistribution procedure simplifies by not requiring a characteristic-based approach; simply redistributing all of $(1 - \Lambda_{i,j,k})\, \delta M_{i,j,k}$ into the mixed and fluid cells among the immediate 26 neighbors of the mixed cell is sufficient. In addition, through numerical experiment, we have found that a mass-weighted approach to redistribution is superior to the volume-weighted approach of [15, 7] in the presence of strong shocks.

We first define $U^{n+1,**}_{i,j,k}$ in all mixed cells and immediate neighbors of mixed cells by

$$\begin{aligned} U^{n+1,**}_{i,j,k} &= U^{n+1,p}_{i,j,k} \quad \text{if } i, j, k \text{ is a mixed cell;} \\ &= U^{n+1,\text{ref}}_{i,j,k} \quad \text{if } i, j, k \text{ is a fluid cell.} \end{aligned} \tag{2.26}$$

In all mixed cells we then define

$$m^{\text{red}}_{i,j,k} = \sum_{(l,m,n)\in \text{nbh}(i,j,k)} \rho^{n+1,**}_{l,m,n} \Lambda_{l,m,n}\, \Delta x\, \Delta y\, \Delta z, \tag{2.27}$$

where $\rho^{n+1,**}_{l,m,n}$ is given by (2.26). The amount of "mass" to be redistributed from the mixed cell l, m, n to the neighboring mixed or fluid cell i, j, k is then

$$\begin{aligned}&\text{"mass" from } l, m, n \text{ to } i, j, k \\ &\quad = \rho^{n+1,**}_{i,j,k} \Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z \frac{(1 - \Lambda_{l,m,n})\, \delta M_{l,m,n}}{m^{\text{red}}_{l,m,n}}.\end{aligned} \tag{2.28}$$

The total amount of "mass" in cell l, m, n after redistribution, $\Lambda_{l,m,n}\, \Delta x\, \Delta y\, \Delta z U^{n+1}_{l,m,n}$, is then given by

$$\begin{aligned}\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z U^{n+1}_{i,j,k} &= \Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z U^{n+1,**}_{i,j,k} \\ &\quad + \Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z \rho^{n+1,**}_{i,j,k} \sum_{(l,m,n)\in \text{nbh}(i,j,k), \Lambda_{l,m,n}>0} \frac{(1 - \Lambda_{l,m,n})\, \delta M_{l,m,n}}{m^{\text{red}}_{l,m,n}}.\end{aligned}$$

Dividing both sides of this last expression by $\Lambda_{i,j,k}\, \Delta x\, \Delta y\, \Delta z$, we obtain the following expression for the final update to the cells affected by redistribution,

$$U^{n+1}_{i,j,k} = U^{n+1,**}_{i,j,k} + \rho^{n+1,**}_{i,j,k} \sum_{(l,m,n)\in \text{nbh}(i,j,k), \Lambda_{l,m,n}>0} \frac{(1 - \Lambda_{l,m,n})\, \delta M_{l,m,n}}{m^{\text{red}}_{l,m,n}}. \tag{2.29}$$

We note that the only cells affected by (2.29) are the mixed cells and those fluid cells that are among the immediate 26 neighbors of at least one mixed cell.

The scaling of the volumes by $\rho^{n+1,**}$ in (2.27) and (2.29) results in a redistribution scheme that is mass-weighted in contrast to the volume weighted scheme used by Bell *et al.* [7] for shock tracking; i.e., replacing $\rho^{n+1,**}$ with "1" in (2.27) and (2.29) results in a volume-weighted redistribution scheme. As noted above, the use of mass weighting in the Cartesian grid integration scheme has so far been found to give superior numerical results. In implementing mass-weighted redistribution, there are also a number of options as to which value of ρ to use, among these $\rho^n_{i,j,k}$ and $\rho^{n+1,\text{ref}}_{i,j,k}$. Through numerical experiments, we found that using $\rho^{n+1,**}_{i,j,k}$ resulted in a more robust scheme in the presence of strong shocks than the schemes resulting from using either $\rho^n_{i,j,k}$ and $\rho^{n+1,\text{ref}}_{i,j,k}$. We note that for a more general system of hyperbolic equations mass-weighted redistribution may not be a meaningful choice. In fact, even for gas dynamics, there are other possibilities such as total energy weighting.

The redistribution scheme discussed above is fully conservative because it is equivalent to modifying the cell edge fluxes calculated in the higher order Godunov predictor step in a manner that yields the same numerical results.

2.6.1. *Redistribution: Geometry Limitations and Stability*

The current implementation of the redistribution algorithm requires that genuinely separate regions of the fluid (again, for instance, in the case where a flat plate is embedded in the flow) be separated in any direction by at least one computational cell that is fully contained in the solid body (see Figs. 3b,c). The reason for this restriction is that in the current implementation a mixed cell redistributes to all its neighbors even when they are in truly distinct regions of the fluid. The algorithm also has the requirement that every mixed cell have an adequate number of fluid cells among its neighbors. Two possibly problematic configurations are shown in Figs. 3d,e.

Chern and Colella [15] do not prove that the redistribution algorithm is stable, and we are unable to provide a proof here. We have used this algorithm not only for the results presented in this paper in Section 4, but for other computations as well [31, 45]. In doing so, we have not yet observed any stability problems. A heuristic argument that the redistribution algorithm is stable is simply that the algorithm avoids dividing by arbitrarily small cell widths or volumes; throughout the entire single grid algorithm the only divisors are widths or volumes of full fluid cells. Further, although redistribution moves mass into mixed cells, the amount of that mass is roughly proportional to the volume of the cell; hence, dividing that extra amount by the cell volume in (2.29) should not be a problem.

2.7. Cylindrical Coordinates

The restriction of the Cartesian grid integration technique described above to rectangular coordinates in two dimensions is straightforward. The treatment of cylindrical coordinates in two dimensions introduces some additional issues. The discussion here will be restricted to the special case of gas dynamics. For cylindrical coordinates, the basic second-order Godunov integration scheme uses a volume coordinate form of the equations, namely,

$$\frac{\partial U}{\partial t} + \frac{\partial A_r F}{\partial V_r} + \frac{\partial}{\partial r}\begin{pmatrix} 0 \\ p \\ 0 \\ 0 \end{pmatrix} + \frac{\partial G}{\partial z} = 0, \tag{2.30}$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho u v \\ \rho u E + up \end{pmatrix},$$

$$G(U) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v E + vp \end{pmatrix},$$

$E = e + \frac{1}{2}(u^2 + v^2)$, $p = p(\rho, e)$, u and v are the radial and axial components of velocity, $A_r = r$, and $V_r = \frac{1}{2}r^2$. The rationale for using this form of the equations is that it retains the higher order discretization and that the discretization is free-stream preserving; i.e., no pressure gradients are generated in a uniform flow. The formulation of higher order Godunov methods for the equations of gas dynamics in generalized volume coordinate form is discussed in [21].

We now discuss the details of the single-grid algorithm specific to the implementation for cylindrical coordinates. We assume a uniform computational grid with cell widths Δr and Δz indexed by i and j.

2.7.1. *Cylindrical Coordinates: Geometry*

Volume and area fractions are defined as in the case of rectangular coordinates. Equations (2.2) and (2.3) of course must be modified to account for the cylindrical coordinate metric terms. In particular, the volume measure $r\ dr\ dz$ must be used in finding volume fractions and the area measures $r\ dz$ and $r\ dr$ must be used for computing r- and z-face area fractions, respectively.

The calculation of the surface normal in a mixed cell is different from that in the case of rectangular coordinates. The two-dimensional analogue of (2.4) cannot be used. We require a new quantity which we call the rectangular-coordinate volume fraction, $\Lambda^{xy}_{i,j}$, defined by:

$$\Lambda^{xy}_{i,j} = \int_{\Delta^f_{i,j}} dr\,dz/\Delta r\,\Delta z.$$

To derive an expression for the surface normal (n_r, n_z), we first recall that by the divergence theorem

$$\int_{\Delta^f_{i,j}} \nabla\phi\,dr\,dz = \int_{\partial(\Delta^f_{i,j})} \phi\mathbf{n}\,dS_{xy},$$

where dS_{xy} denotes the surface measure for rectangular coordinates. Letting $\phi = r$ and noting that $r\,dS_{xy} = dS_{rz}$, where dS_{rz} is the surface measure in cylindrical coordinates, we obtain

$$\int_{\Delta^f_{i,j}} dr\,dz = \int_{\partial(\Delta^f_{i,j})} n_r\,dS_{rz}$$

$$0 = \int_{\partial(\Delta^f_{i,j})} n_z\,dS_{rz}.$$

Hence,

$$A^f_{i,j}(n_r, n_z) = ((a_{i-1/2,j}r_{i-1/2} - a_{i+1/2,j}r_{i+1/2} + \Lambda^{xy}_{i,j}\Delta r)\,\Delta z, \quad (a_{i,j-1/2} - a_{i,j+1/2})\tfrac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})), \tag{2.31}$$

where A^f is the frontal area in the mixed cell and $r_{i+1/2}$ is the value of r at edge $i + \frac{1}{2}$.

The quantity Λ_{xy} is also used in computing δM; see below.

2.7.2. *Cylindrical Coordinates: Numerical Fluxes and Reference State*

The computation of the numerical fluxes and the reference state in cylindrical coordinates follows the same steps as in rectangular coordinates. First, a two-dimensional analogue of (2.16) is used to compute an extended state in each body cell that is one of the immediate eight neighbors of at least one mixed cell. Equation (2.16) is formulated under the assumption that all computational cells have the same volume. The form of (2.16) used for cylindrical coordinates must account for the cylindrical coordinate volume measure.

We then use the unsplit higher-order Godunov method to compute numerical fluxes $F^{n+1/2}_{i+1/2,j}$ and $p^{n+1/2}_{i+1/2,j}$ at all r-cell edges and $G^{n+1/2}_{i,j+1/2}$ at all z-cell edges. The reference state is then computed by conservative differencing of (2.30),

$$U^{n+1,\text{ref}}_{i,j} = U^n_{i,j} + \Delta t\,\frac{r_{i-1/2}F^{n+1/2}_{i-1/2,j} - r_{i+1/2}F^{n+1/2}_{i+1/2,j}}{(1/2)(r^2_{i+1/2} - r^2_{i-1/2})} + \frac{\Delta t}{\Delta z}(G^{n+1/2}_{i,j-1/2} - G^{n+1/2}_{i,j+1/2}) + \frac{\Delta t}{\Delta r}\begin{pmatrix} 0 \\ p^{n+1/2}_{i-1/2,j} - p^{n+1/2}_{i+1/2,j} \\ 0 \\ 0 \end{pmatrix}. \tag{2.32}$$

2.7.3. *Cylindrical Coordinates: Calculation of δM, Preliminary Update, and Redistribution*

In the computation of $\delta M_{i,j}$ for mixed cells in cylindrical coordinates, direct differencing of the pressure gradient in the r direction is not appropriate. We instead need to account for $\partial p/\partial r$ in a volume-of-fluid manner. To derive an appropriate expression accounting for $\partial p/\partial r$ in $\delta M_{i,j}$, we proceed as follows.

For a mixed cell we rewrite $\partial p/\partial r$ as

$$\frac{\partial p}{\partial r} = \frac{\partial A_r p}{\partial V_r} - p\frac{\partial A_r}{\partial V_r}. \tag{2.33}$$

Integrating the first term on the right of (2.33) over the fluid portion of the computational cell, we find

$$\int_{\Delta^f_{i,j}} \frac{\partial A_r p}{\partial V_r} r\,dr\,dz \approx a_{i+1/2,j}r_{i+1/2}\,\Delta z p^{n+1/2}_{i+1/2,j} - a_{i-1/2,j}r_{i-1/2}\,\Delta z p^{n+1/2}_{i-1/2,j} + A^f_{i,j}n_r p^f_{i,j},$$

where $p^{n+1/2}_{i-1/2,j}$ and $p^{n+1/2}_{i+1/2,j}$ are the pressures at $t^{n+1/2}$ at r cell edges found by the Godunov integrator, $p^f_{i,j}$ is the pressure at the fluid–body interface found by the solution of the reflecting wall Riemann problem, and n_r is the r-component of the surface normal in (2.31). Integrating the second term on the right of (2.33) over the fluid portion of the computational cell, we find

$$\int_{\Delta^f_{i,j}} p\frac{\partial A_r}{\partial V_r} r\,dr\,dz \approx \tilde{p}_{i,j}\Lambda^{xy}_{i,j}\,\Delta r\,\Delta z,$$

where $\tilde{p}$ is an as yet to be specified estimate of the average pressure in the cell over the time step. Consequently, $\partial p/\partial r$ in the r-momentum term of δM is approximated as

$$\begin{aligned} &\Lambda_{i,j}\tfrac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})\,\Delta z\left(\frac{\partial p}{\partial r}\right)_{i,j} \\ &\quad\approx a_{i+1/2,j}r_{i+1/2}\,\Delta z\,p^{n+1/2}_{i+1/2,j} - a_{i-1/2,j}r_{i-1/2}\,\Delta z\,p^{n+1/2}_{i-1/2,j} \\ &\quad - \Lambda^{xy}_{i,j}\,\Delta r\,\Delta z\,\tilde{p}_{i,j} + A^f_{i,j}n_r p^f_{i,j}. \end{aligned} \tag{2.34}$$

We require that this expression results in the usual differencing in a fluid cell, i.e., when $\Lambda^{xy}_{i,j} = a_{i-1/2,j} = a_{i+1/2,j} = 1$ and $A^f_{i,j} = 0$, and that $\partial p/\partial r = 0$ when p is constant. We thus define $\tilde{p}_{i,j}$ by

$$\tilde{p}_{i,j} = \frac{p^{n+1/2}_{i-1/2,j} + p^{n+1/2}_{i+1/2,j}}{2}. \tag{2.35}$$

From (2.31), (2.34), and (2.35), we see that indeed $\partial p/\partial r = 0$ when p is constant. The expression for $\partial p/\partial r$ thus maintains the free-stream preservation property of the basic Cartesian grid integration scheme.

The expression for $\delta M_{i,j}$ in cylindrical coordinates is defined, then, by

$$\begin{aligned}\delta M_{i,j} = {} & \Lambda_{i,j}\tfrac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})\,\Delta z\,(U^n_{i,j} - U^{n+1,\mathrm{ref}}_{i,j}) \\ & + \Delta t\,(\Delta z\,(r_{i-1/2}a_{i-1/2,j}F^{n+1/2}_{i-1/2,j} - r_{i+1/2}a_{i+1/2,j}F^{n+1/2}_{i+1/2,j}) \\ & + \tfrac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})(a_{i,j-1/2}G^{n+1/2}_{i,j-1/2} - a_{i,j+1/2}G^{n+1/2}_{i,j+1/2}) \\ & - \Lambda_{i,j}\tfrac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})\,\Delta z \begin{pmatrix} 0 \\ (\partial p/\partial r)_{i,j} \\ 0 \\ 0 \end{pmatrix} \\ & - (F, G)_f A^f_{i,j,k}\mathbf{n}^f_{i,j}),\end{aligned} \tag{2.36}$$

where $(\partial p/\partial r)_{i,j}$ is given by (2.34) and (2.35). The expression $(F, G)_f A^f_{i,j,k}\mathbf{n}^f_{i,j}$ can be expanded as

$$(F, G)_f A^f_{i,j,k}\mathbf{n}^f_{i,j} = \begin{pmatrix} 0 \\ 0 \\ (a_{i,j-1/2} - a_{i,j+1/2})\frac{1}{2}(r^2_{i+1/2} - r^2_{i-1/2})p^f_{i,j} \\ 0 \end{pmatrix}.$$

The preliminary update and redistribution steps are two-dimensional analogues of the steps in Section 2.6 in which cylindrical coordinate volume measures are used instead of rectangular measures.

3. COUPLING TO ADAPTIVE MESH REFINEMENT

In this section we describe how the Cartesian grid integration scheme is coupled to the adaptive mesh refinement algorithm described by Bell *et al.* [3], which is an extension to three space dimensions of the AMR algorithm of Berger and Colella [8]. We briefly review the adaptive mesh refinement algorithm and, in particular, those operations which must be modified in order to incorporate the single grid Cartesian mesh algorithm of Section 2. We then discuss the changes or additions needed in order to incorporate the single mesh Cartesian grid algorithm. AMR has been developed for both two dimensions, including cylindrical coordinates, and three dimensions. For clarity of exposition we will describe the method for the three-dimensional case. The restriction to two dimensions is straightforward. The extension of the basic AMR scheme to cylindrical coordinates is discussed in Section 3.1.4. The extension of AMR for cylindrical coordinates to a Cartesian grid setting simply entails using cylindrical coordinate volume and area measures in place of rectangular coordinate measures wherever applicable.

It is possible to implement the AMR algorithm with the Cartesian grid modifications in Fortran. However, there are many operations in AMR even without the Cartesian grid enhancements for which Fortran is ill-suited. In our implementation, we use an object-oriented, hybrid C++/Fortran approach in which most single grid operations are coded in Fortran while the driver, overall data and memory management, sparse data structures, inter-grid communications, etc., are coded in C++. Further, we make extensive use of a C++ library for many basic grid manipulations. Crutchfield and Welcome [26] discuss the implementation issues in more detail.

3.1. Adaptive Mesh Refinement: Review

The AMR algorithm uses a hierarchical grid structure which changes dynamically and which is composed of grids of varying resolution. The grid of coarsest resolution is referred to as the level 0 grid. This grid by definition covers the entire problem domain. There are also one or more additional levels of grid, each finer than the rest. These finer levels of grid do not cover the whole domain but only those regions where more resolution is determined to be needed by a "grid generation" procedure described below. The grid generation procedure is not just performed at initialization but throughout the time stepping process so that the grids can change dynamically in response to the solution. The cell widths of all the grids at level $L - 1$ are proportional by an even integer factor r_{L-1} (called a "refinement ratio") to the cell widths at level L, i.e., $\Delta x_L = \Delta x_{L-1}/r_{L-1}$, etc. Further, each cell boundary of the level $L - 1$ cells coincides with cell boundaries of level L cells. In other words, $(r_{L-1})^3$ cells at level L coincide spatially with exactly one cell at level $L - 1$; we refer to the coarse cell as an "underlying coarse cell" and the fine cells as "overlying fine cells." Further, the levels of grid are properly nested in that each cell in a buffer of r_{L-1} cells enclosing the grids at level L overlays a coarse cell at level $L - 1$. The figures in [8, 3] illustrate these concepts.

The generation of grids at level L, $L \geq 1$ proceeds as follows. At level $L - 1$, a cell tagging procedure identifies level $L - 1$ cells which need to be refined. Two different cell tagging criteria are used. One is to tag cells for which an error estimate found by a Richardson extrapolation procedure exceeds some user-defined tolerance. The other is to use additional, optional user-defined error or cell tagging criteria; for example, it might be desirable to refine in regions containing large density gradients. The tagged cells are then clustered into logically rectangular grids that are subdivided to form level L grids. For the sake of grid efficiency, a certain percentage of untagged cells are clustered with the tagged cells during this process. Once the new level L grids are defined, the solution is defined in any new level L cells that were added through the grid generation process by conservatively interpolating the solution from level $L - 1$. The procedure is then repeated for all finer levels of resolution. Additional steps are taken to ensure proper nesting during the grid generation process.

We now outline the procedure to advance the solution on level L one level L time step of size Δt from time t to time $t + \Delta t$. This procedure is recursive with respect to refinement level. Further details are given in [8, 3].

ALGORITHM TO ADVANCE SOLUTION ON LEVEL L FROM TIME t TO $t + \Delta t$.

1. If it is time to regrid, generate new grids at level L.
2. Advance solution on each level L grid from time t to $t + \Delta t$.
   (a) Set boundary conditions needed to integrate grid in one of three ways:
      i. Use physical boundary conditions, if applicable.
      ii. Otherwise, use data from level L grids, if available.
      iii. Otherwise,
         A. Interpolate data in time on level $L - 1$ cells underlying the level L boundary cells.
         B. Do conservative interpolation: Interpolate the coarse cell data spatially to get the values in the level L boundary cells in a manner that maintains conservation.
   (b) Advance the solution on the grid from time t to $t + \Delta t$ using the higher order Godunov method.
3. For $J = 1, r_L$
   (a) Advance the solution on level $L + 1$ from $t + (J - 1)\,\Delta t/r_L$ to $t + J\,\Delta t/r_L$
4. Fix up conservation between level L and level $L + 1$ at $t + \Delta t$ in two ways:
   (a) Average the fine grid solution onto coarse grid cells: In level L cells that underlay level $L + 1$ cells, replace the solution in the level L cell by the average of the solution in the overlying level $L + 1$ cells
   (b) Reflux: In level L cells that share a cell edge with a fine grid interface but do not actually underlay level $L + 1$ cells, replace the effect of the coarse grid flux at the shared cell edge with the effect of the fine grid fluxes.

Regridding (step 1) is typically done every other time step, i.e., when J is even.

The modifications to the AMR methodology in order to extend it to the Cartesian grid scheme involve the following operations: averaging of fine grid data onto coarse grid cells, conservative interpolation, and refluxing. We now quickly review these for both three dimensions and cylindrical coordinates; the restriction to two dimensions is straightforward.

In our discussion, we use the following conventions: The map between the index space of level L and level $L + 1$ is

Level L cell i, j, k is covered by the $(r_L)^3$ level $L + 1$ cells with indices l, m, n satisfying the conditions

$$r_L i \le l \le r_L(i + 1) - 1, \quad r_L j \le m \le r_L(j + 1) - 1,$$
$$r_L k \le n \le r_L(k + 1) - 1. \tag{3.1}$$

Further, the center of level L cell i, j, k is the physical location (x, y, z):

$$(x, y, z) = ((i + 0.5)\,\Delta x_L, (j + 0.5)\,\Delta y_L, (k + 0.5)\,\Delta z_L).$$

3.1.1. *Averaging Down*

After level L and level $L + 1$ grids have been advanced to the same time, the solution in a level L cell that underlays level $L + 1$ cells is replaced by the average of the overlying fine grid solution as follows. We call this operation "averaging down," for short. Let i, j, k denote the index of a coarse cell. Let U^L and U^{L+1} denote the level L and $L + 1$ solutions, respectively. The averaging procedure is then given by

$$U^L_{i,j,k} = \sum_{l=r_L i}^{r_L(i+1)-1} \sum_{m=r_L j}^{r_L(j+1)-1} \sum_{n=r_L k}^{r_L(k+1)-1} \frac{\Delta x_{L+1}\,\Delta y_{L+1}\,\Delta z_{L+1}\,U^{L+1}_{l,m,n}}{\Delta x_L\,\Delta y_L\,\Delta z_L}. \tag{3.2}$$

3.1.2. *Conservative Interpolation*

When setting boundary conditions for fine grids or when defining the solution on newly formed fine grids, the adaptive mesh refinement scheme uses conservative interpolation to define the solution in a fine grid cell when coarse grid data alone is available. Suppose that l, m, n is a fine grid cell at level L with cell center (x_l, y_m, z_n). Let i, j, k be a coarse cell underlaying l, m, n at level $L - 1$ with cell center x_i, y_j, z_k. Let U^{L-1} and U^L denote the level $L - 1$ and L solutions, respectively.

(1) Define limited central difference approximations

$$(\Delta_x U^{L-1})_{i,j,k}/\Delta x_{L-1}, \quad (\Delta_y U^{L-1})_{i,j,k}/\Delta y_{L-1}, \quad (\Delta_z U^{L-1})_{i,j,k}/\Delta z_{L-1},$$

to the spatial derivatives

$$\partial U^{L-1}/\partial x, \quad \partial U^{L-1}/\partial y, \quad \partial U^{L-1}/\partial z,$$

at x_i, y_j, z_k of the coarse grid data using a procedure analogous to (2.12).

(2) Define the fine grid data by

$$\begin{aligned} U^L_{l,m,n} = U^{L-1}_{i,j,k} &+ (x_l - x_i)(\Delta_x U^{L-1})_{i,j,k}/\Delta x_{L-1} \\ &+ (y_m - y_j)(\Delta_y U^{L-1})_{i,j,k}/\Delta y_{L-1} \\ &+ (z_n - z_k)(\Delta_z U^{L-1})_{i,j,k}/\Delta z_{L-1}. \end{aligned} \tag{3.3}$$

If there is no cell i, j, k at level M for which l, m, n satisfies (3.1), the procedure is repeated recursively from the finest level possible.

3.1.3. *Refluxing*

After level L and level $L+1$ grids have been advanced to the same time, the solution in a level L cell that shares a cell edge with a fine grid boundary, but does not actually underlay any level $L+1$ cells, must be modified in order to maintain conservation. The modified solution is found by replacing the effect of the coarse grid flux at the shared cell edge with the effect of fine grid fluxes. The general procedure for refluxing can be found in [8, 3]. We explain one specific case and briefly discuss how to generalize it.

Suppose coarse grid cell i, j, k at level L shares its right x-cell face with a level $L+1$ grid boundary. Let $F^L_{i+1/2,j,k}$ denote the flux at the right cell face of i, j, k used in the level L integration step to update the solution in i, j, k from time t to $t+\Delta t$. Let $U^L_{i,j,k}$ denote the solution in i, j, k after the coarse grid integration step. Let $F^{L+1,J}_{l-1/2,m,n}$, where l, m, n satisfy $l=(r_L+1)i$, $r_L j \le m \le r_L(j+1)-1$, and $r_L k \le n \le r_L(k+1)-1$, denote the flux used in updating the level $L+1$ solution in cell l, m, n from time $t+(J-1)\Delta t/r_L$ to $t+J\Delta t/r_L$. We can then define the flux correction $\delta F^L_{i,j,k}$ by

$$\delta F^L_{i,j,k} = \Delta t\, \Delta y_L\, \Delta z_L F^L_{i+1/2,j,k} - \sum_{J=1,r_L} \sum_{m=r_L j}^{r_L(j+1)-1} \sum_{n=r_L k}^{r_L(k+1)-1} \left(\frac{\Delta t}{r_L} \Delta y_{L+1}\, \Delta z_{L+1}\, F^{L+1,J}_{l-1/2,m,n} \right). \tag{3.4}$$

The solution in cell i, j, k is then updated by

$$U^L_{i,j,k} = U^L_{i,j,k} + \frac{\delta F^L_{i,j,k}}{\Delta x_L\, \Delta y_L\, \Delta z_L}. \tag{3.5}$$

The update (3.5) is equivalent to repeating the integration of the coarse cell using the sum of the fine grid fluxes to update the cell instead of the coarse grid flux.

The refluxing strategy at upper y- and z-faces is similar; at lower faces, the order of subtraction in (3.4) must be reversed. Sums of expressions of the form (3.4) are used to define δF if the coarse cell has more than one cell edge in common with the fine grid boundaries.

3.1.4. *Cylindrical Coordinates*

The only modifications needed for averaging down and refluxing are that cylindrical coordinate volume and area measures need to be used in (3.2), (3.4), and (3.5).

The modification for conservative interpolation is slightly more involved. The equations to compute the central difference approximations $(\Delta_r U^{L-1})_{i,j}/\Delta r$ and $(\Delta_z U^{L-1})_{i,j}/\Delta z$ to the spatial derivatives $\partial U/\partial r$ and $\partial U/\partial z$ of the level $L-1$ solution are simply the two-dimensional analogues of (2.12). Equation (3.3), however, must be replaced by the following procedure to ensure that the interpolation is conservative:

(1) Let r_i, z_j and r_l, z_m denote the centers of the coarse and fine grid cells i, j and l, m, respectively.

(2) Define v^L_l and v^{L-1}_i by

$$v^L_l = \tfrac{1}{2}((r_l + \tfrac{1}{2}\Delta r_L)^2 + (r_l - \tfrac{1}{2}\Delta r_L)^2)$$
$$v^{L-1}_i = \tfrac{1}{2}((r_i + \tfrac{1}{2}\Delta r_{L-1})^2 + (r_i - \tfrac{1}{2}\Delta r_{L-1})^2).$$

(3) Define Δv^{L-1}_i by

$$\Delta v^{L-1}_i = (r_i + \tfrac{1}{2}\Delta r_{L-1})^2 - (r_i - \tfrac{1}{2}\Delta r_{L-1})^2.$$

(4) Define the fine grid solution in cell l, m by

$$U^L_{l,m} = U^{L-1}_{i,j} + (v^L_l - v^{L-1}_i)(\Delta_r U^{L-1})_{i,j,k}/\Delta v^{L-1}_i + (z_m - z_j)(\Delta_z U^{L-1})_{i,j,k}/\Delta z_{L-1}. \tag{3.6}$$

3.2. Cartesian Grid Adaptive Mesh Refinement: Summary

There are four modifications that are required to couple the Cartesian grid integration method to the adaptive mesh refinement algorithm. (Throughout the discussion on these modifications, we assume the convention of mapping between the index space of level L and level $L+1$ described in (3.1) and the accompanying text.)

(1) Volume and area fractions must be consistent across levels of refinement. Specifically, averages of overlying fine grid volume and area fractions must equal the corresponding underlying coarse grid fractions.

(2) The algorithm to average fine grid data onto coarse grid cells must correctly account for mixed cells. Specifically, the following expression replaces (3.2):

$$U^L_{i,j,k} = \sum_{l=r_L i}^{r_L(i+1)-1} \sum_{m=r_L j}^{r_L(j+1)-1} \sum_{n=r_L k}^{r_L(k+1)-1} \frac{\Lambda^{L+1}_{l,m,n}\, \Delta x_{L+1}\, \Delta y_{L+1}\, \Delta z_{L+1}\, U^{L+1}_{l,m,n}}{\Lambda^L_{i,j,k}\, \Delta x_L\, \Delta y_L\, \Delta z_L}, \tag{3.7}$$

where Λ^L and Λ^{L+1} denote the level L and $L+1$ volume fractions, respectively.

(3) Any central difference approximation used in conservative interpolation (3.3) is set to zero if the computation of that central difference uses values from a body cell.

(4) The refluxing algorithm must be modified to account for area fractions and augmented by a re-redistribution algorithm to account for the effects of redistribution across coarse–fine grid boundaries.

Modifications (1), (2), and (3) are self-explanatory. Modification (4) is explained below. Note that the purpose of modifications (2) and (4) is to maintain conservation, and that modification (1) is necessary to ensure that (2) and (4) do so.

The restriction of these modifications to two dimensions is straightforward. The extension of them to cylindrical coordinates simply involves using cylindrical coordinate volume and area measures in place of the rectangular coordinate measures wherever applicable.

3.3. Refluxing and Re-redistribution

Away from the embedded fluid–body boundary, the correction to the coarse grid solution due to coarse grid/fine grid interactions consists of adding a flux correction δF to the coarse grid cells that border fine grids. This refluxing process is summarized above in Section 3.1.3.

In the vicinity of the intersection of the fluid–body interface with a coarse grid/fine grid boundary, the refluxing algorithm described in Section 3.1.3 must be replaced with a composite refluxing/redistribution algorithm that makes corrections to the solution in selected cells in order to maintain conservation. The new algorithm is needed for the following reasons:

(1) to account for the effect of geometry on the flux increment δF;

(2) to account for the additional transport of "mass" due to redistribution across coarse-fine grid boundaries;

(3) to make corrections in a manner that is both stable and conservative.

(Throughout this discussion, the term "mass" refers to integrated values of the conserved quantities, i.e., mass, momentum, and energy rather than density, momentum density, and energy density.)

The composite refluxing/redistribution algorithm is applied only to coarse cells which would normally be affected by refluxing and which also satisfy the condition:

(1) The coarse cell is a mixed cell, or

(2) the coarse cell is a neighbor of a mixed coarse cell underlaying the adjacent fine grid (3.8)

The refluxing algorithm described in Section 3.1.3 is used for all other coarse cells at coarse–fine grid boundaries.

The composite algorithm consists of the following steps in an applicable coarse cell i, j, k on level L:

1. Compute a flux increment $\delta F^L_{i,j,k}$ to account for the transport of "mass" due to the difference between the level L and the level $L+1$ fluxes.

2. Compute a redistribution increment $\delta R^L_{i,j,k}$ to account for the transport of "mass" due to redistribution across the level L/level $L+1$ grid boundaries.

3. Define the coarse-fine grid correction, $\delta M^{L,c-f}_{i,j,k}$ by

$$\delta M^{L,c-f}_{i,j,k} = \delta F^L_{i,j,k} + \delta R^L_{i,j,k}. \tag{3.9}$$

4. Add the stable portion of these increments to the level L solution in cell i, j, k, $U^L_{i,j,k}$, by

$$U^L_{i,j,k} = U^L_{i,j,k} + \frac{\delta M^{L,c-f}_{i,j,k}}{\Delta x\, \Delta y\, \Delta z}. \tag{3.10}$$

5. If $\Lambda_{i,j,k} < 1$,

(a) Redistribute $(1 - \Lambda_{i,j,k})\ \delta M^{L,c-f}_{i,j,k}$ to the coarse grid neighbors of i, j, k using volume weighted versions of (2.27) and (2.29).

(b) If a coarse grid neighbor l, m, n of i, j, k underlays a level M cell, $M \geq L+1$, distribute the "mass" of the redistribution increment $(1 - \Lambda_{i,j,k})\ \delta M^{L,c-f}_{i,j,k}$ onto the level M cell in a volume-weighted manner.

Step (1) is identical to the operation described in (3.4) and the accompanying text with the modification that the cell areas are weighted by the area fractions. For example, Eq. (3.4), which computes the flux increment in the case that coarse grid cell i, j, k at level L shares its right x-cell face with a level $L+1$ grid boundary, is modified to be

$$\begin{aligned} \delta F^L_{i,j,k} = {} & \Delta t\, a^L_{i+1/2,j,k}\, \Delta y_L\, \Delta z_L\, F^L_{i+1/2,j,k} \\ & - \sum_{J=1,r_L} \sum_{m=r_Lj}^{r_L(j+1)-1} \sum_{n=r_Lk}^{r_L(k+1)-1} \\ & \left(\frac{\Delta t}{r_L} a^{L+1}_{I-1/2,m,n}\, \Delta y_{L+1}\, \Delta z_{L+1}\, F^{L+1,J}_{I-1/2,m,n} \right), \end{aligned} \tag{3.11}$$

where a^L and a^{L+1} denote area fractions at level L and $L+1$, respectively. Steps (3) and (4) are self-explanatory. Steps (2) and (5) are described below.

3.3.1. *Computation of δR*

Additional corrections to maintain conservation when the fluid–body boundary crosses a coarse–fine boundary are needed because redistribution provides an additional mechanism for numerical transport across a coarse–fine boundary.

To describe this mechanism, we divide the fine cells at a coarse–fine grid interface into two groups: the fine interior cells, i.e., the cells at the grid boundary in the interior of the grid, and the fine exterior cells, i.e., the cells in the boundary region of the grid at the grid boundary. We also divide the coarse cells into two groups: the coarse interior cells, i.e., the coarse cells underlaying the fine interior cells, and the coarse exterior cells, i.e., the cells underlaying the fine exterior cells. We label these four groups of cells (b), (c), (a), and (d), respectively, for ease of reference; see Fig. 4.

As the level L grids advance, mixed cells in the coarse exterior region (region (a)) redistribute mass into the coarse interior region (region (d)), and mixed cells in the interior region redistribute into the exterior region. Similarly, as the level $L+1$ grids advance, mixed cells in the fine exterior region (region (c)) redistribute mass into the fine interior region (region (b)), and mixed cells in the interior region rerdistribute

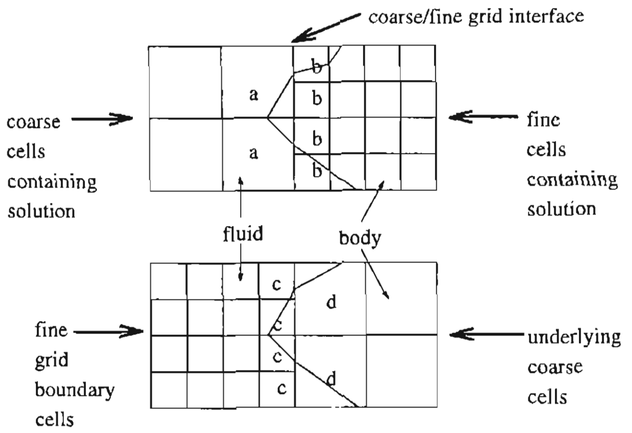

FIG. 4. Categorization of coarse and fine cells near the intersection of a coarse–fine grid boundary with the fluid–body boundary. The coarse and fine cells at the top of the figure contain the best representation of the numerical solution. The cells in regions (a), (b), (c), and (d) are labeled.

into the exterior region. After the coarse (level L) and the fine (level $L + 1$) grids have been advanced to the same time, the only effects of redistribution that should be present in the solution (in order that conservation be maintained) in the cells at the coarse–fine interface are those that resulted from redistribution from the fine interior cells into the fine exterior cells (i.e., from region (b) into region (c)) and from redistribution from the coarse exterior cells into the coarse interior cells (i.e., from region (a) into region (d)). However, these effects have been lost and must be recovered from the following reasons. The effect due to redistribution from fine interior cells into the fine exterior cells is lost because the fine exterior cells are only in the fine grid boundary region; the coarse grid cells (region (a)) contain the actual solution. The coarse interactions have been lost because the values on the coarse interior cells (region (d)) have been redefined by averaging the values of the overlying fine interior cells (region (b)). In addition, the effects of redistribution from the fine exterior cells (region (c)) into the fine interior cells (region (b)) and from the coarse interior cells (region (d)) into the coarse exterior cells (region (a)) are present in the solution and must be removed. These effects are present simply because the redistribution algorithm (2.27) and (2.29) applies to cells in the boundary region of the grid as well as in the interior of the grid.

In summary, there are four basic coarse–fine re-redistribution quantities that must be computed:

δR^I_{L+1}. This represents the sum of the values redistributed into the fine interior cells from the fine exterior cells over a single level L time step. Their effect is present, but unwanted in the solution, and must be removed.

δR^I_L. This represents the values redistributed from the coarse exterior cells into the coarse interior cells over a single level L time step. Their effect is lost when the coarse values are redefined by averaging the overlying fine values and must be recovered.

δR^E_{L+1}. This represents the sum of the values redistributed from the fine interior cells into the fine exterior cells over a single level L time step. Their effect must be recovered because the solution in the boundary region is represented by the solution on the underlying coarse exterior cells.

δR^E_L. This represents the redistribution values from the coarse interior cells to the coarse exterior cells over a single level L time step. Their effect is present, but unwanted in the solution, and must be removed.

We actually associate all four of the above corrections with the coarse exterior cells. Specifically, we associate δR^I_L and δR^I_{L+1} with the coarse exterior cell from which the redistribution values come and the coarse exterior cell that underlays the fine exterior cells from which the values come, respectively. Further, we associate δR^E_L and δR^E_{L+1} with the coarse exterior cell that receives values and the coarse exterior cell that underlays the fine cells that receive them, respectively. In other words, all four corrections are associated with the appropriate cells i, j, k in region (a), satisfying (3.8). We define $\delta R^L_{i,j,k}$ by

$$\delta R^L_{i,j,k} = (\delta R^I_L)_{i,j,k} - (\delta R^I_{L+1})_{i,j,k} + (\delta R^E_{L+1})_{i,j,k} - (\delta R^E_L)_{i,j,k}. \tag{3.12}$$

Equations (3.13)–(3.16) below define the four components of $\delta R^L_{i,j,k}$. The following terms and notation, along with Fig. 4, facilitate these definitions:

$\delta M^L_{\mathcal{I},\mathcal{J},\mathcal{K}:\mathcal{L},\mathcal{M},\mathcal{N}}$. This is the amount of "mass" redistributed from level L mixed cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ to level L cell $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ while advancing the level L solution one level L time step from t to $t + \Delta t$ as computed by (2.28).

$\delta M^{L+1,J}_{\mathcal{I},\mathcal{J},\mathcal{K}:\mathcal{L},\mathcal{M},\mathcal{N}}$. This is the amount of "mass" redistributed from level $L + 1$ mixed cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ to level $L + 1$ cell $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ while advancing the level $L + 1$ solution one level $L + 1$ time step from $t + (J - 1)\,\Delta t/r_L$ to $t + J\,\Delta t/r_L$ as computed by (2.28).

$(\text{a–d})_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level L cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (d) to which "mass" is redistributed from level L mixed cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a):

$$(\text{a–d})_{\mathcal{I},\mathcal{J},\mathcal{K}} = \text{nbh}(\mathcal{I},\mathcal{J},\mathcal{K}) \cap \{(\mathcal{L},\mathcal{M},\mathcal{N}) : (\mathcal{L},\mathcal{M},\mathcal{N}) \in (\text{d}) \text{ and } \Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} > 0\}.$$

$(\text{d–a})_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level L mixed cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (d) which redistribute "mass" to level L cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a):

$$(\text{d–a})_{\mathcal{I},\mathcal{J},\mathcal{K}} = \text{nbh}(\mathcal{I},\mathcal{J},\mathcal{K}) \cap \{(\mathcal{L},\mathcal{M},\mathcal{N}) : (\mathcal{L},\mathcal{M},\mathcal{N}) \in (\text{d}) \text{ and } 0 < \Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} < 1\}.$$

$(\text{b}-\text{c})^{src}_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level $L+1$ mixed cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (b) which redistribute "mass" to level $L+1$ cells in region (c) overlaying level L cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a):

$$
\begin{aligned}
(\text{b}-\text{c})^{src}_{\mathcal{I},\mathcal{J},\mathcal{K}} = \{(\mathcal{L}, \mathcal{M}, \mathcal{N}) : ((\mathcal{L}, \mathcal{M}, \mathcal{N}) &\in (\text{b}) \\
\text{and } 0 < \Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} &< 1 \text{ and} \\
(\mathcal{L}, \mathcal{M}, \mathcal{N}) \in \text{nbh}(\mathcal{P}, \mathcal{Q}, \mathcal{R}) &: ((\mathcal{P}, \mathcal{Q}, \mathcal{R}) \\
\text{overlays } &(\mathcal{I}, \mathcal{J}, \mathcal{K}) \\
\text{and } \Lambda_{\mathcal{P},\mathcal{Q},\mathcal{R}} &> 0))\}.
\end{aligned}
$$

$(\text{b}-\text{c})^{dst}_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level $L+1$ cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (c) overlaying level L cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a) to which mass is redistributed from level $L+1$ mixed cells in region (b):

$$
\begin{aligned}
(\text{b}-\text{c})^{dst}_{\mathcal{I},\mathcal{J},\mathcal{K}} = \{(\mathcal{L}, \mathcal{M}, \mathcal{N}) : ((\mathcal{L}, \mathcal{M}, \mathcal{N}) &\text{ overlays } (\mathcal{I}, \mathcal{J}, \mathcal{K}) \text{ and} \\
\Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} &> 0 \text{ and} \\
(\mathcal{L}, \mathcal{M}, \mathcal{N}) \in \text{nbh}(\mathcal{P}, \mathcal{Q}, \mathcal{R}) &: ((\mathcal{P}, \mathcal{Q}, \mathcal{R}) \in (\text{b}) \\
\text{and } 0 < \Lambda_{\mathcal{P},\mathcal{Q},\mathcal{R}} &< 1))\}.
\end{aligned}
$$

$(\text{c}-\text{b})^{src}_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level $L+1$ mixed cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (c) which overlay $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a) and which redistribute "mass" to level $L+1$ cells in region (b):

$$
\begin{aligned}
(\text{c}-\text{b})^{src}_{\mathcal{I},\mathcal{J},\mathcal{K}} = \{(\mathcal{L}, \mathcal{M}, \mathcal{N}) : ((\mathcal{L}, \mathcal{M}, \mathcal{N}) &\text{ overlays } (\mathcal{I}, \mathcal{J}, \mathcal{K}) \\
\text{and } 0 < \Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} &< 1)\}.
\end{aligned}
$$

$(\text{c}-\text{b})^{dst}_{\mathcal{I},\mathcal{J},\mathcal{K}}$. This is the set of level $L+1$ cells $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ in region (b) to which mass is redistributed by mixed cells in region (c) which overlay $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ in region (a):

$$
\begin{aligned}
(\text{c}-\text{b})^{dst}_{\mathcal{I},\mathcal{J},\mathcal{K}} = \{(\mathcal{L}, \mathcal{M}, \mathcal{N}) : ((\mathcal{L}, \mathcal{M}, \mathcal{N}) &\in (\text{b}) \\
\text{and } \Lambda_{\mathcal{L},\mathcal{M},\mathcal{N}} &> 0 \\
(\mathcal{L}, \mathcal{M}, \mathcal{N}) \in \text{nbh}(\mathcal{P}, \mathcal{Q}, \mathcal{R}) &: ((\mathcal{P}, \mathcal{Q}, \mathcal{R}) \\
\text{overlays } &(\mathcal{I}, \mathcal{J}, \mathcal{K}) \\
\text{and } 0 < \Lambda_{\mathcal{P},\mathcal{Q},\mathcal{R}} &< 1))\}.
\end{aligned}
$$

Using the forgoing terminology, we can now define the four components of $\delta R^L_{i,j,k}$ in (3.12) as

$$(\delta R^I_L)_{i,j,k} = \sum_{(l,m,n)\in(\text{a}-\text{d})_{i,j,k}} \delta M^L_{i,j,k;l,m,n} \tag{3.13}$$

$$(\delta R^E_L)_{i,j,k} = \sum_{(l,m,n)\in(\text{d}-\text{a})_{i,j,k}} \delta M^L_{l,m,n;i,j,k} \tag{3.14}$$

$$(\delta R^I_{L+1})_{i,j,k} = \sum_{J=1}^{r_L} \sum_{(l,m,n)\in(\text{c}-\text{b})^{src}_{i,j,k}} \left(\sum_{(I,J,K)\in\text{nbh}(l,m,n)\cap(\text{c}-\text{b})^{dst}_{i,j,k}} \delta M^{L+1,J}_{l,m,n;I,J,K} \right) \tag{3.15}$$

$$(\delta R^E_{L+1})_{i,j,k} = \sum_{J=1}^{r_L} \sum_{(l,m,n)\in(\text{b}-\text{c})^{dst}_{i,j,k}} \left(\sum_{(I,J,K)\in\text{nbh}(l,m,n)\cap(\text{b}-\text{c})^{src}_{i,j,k}} \delta M^{L+1,J}_{I,J,K;l,m,n} \right). \tag{3.16}$$

3.3.2. *Redistribution of $\delta M^{L,c-f}_{i,j,k}$*

If $\Lambda_{i,j,k} < 1$, we must redistribute $(1 - \Lambda_{i,j,k})\, \delta M^{L,c-f}_{i,j,k}$ to the level L neighbors of i, j, k using volume weighted versions of (2.27) and (2.29) as follows. We define

$$m^{\text{red}}_{i,j,k} = \sum_{\text{level } L \text{ cells}(l,m,n)\in\text{nbh}(i,j,k)} \Lambda_{l,m,n}\, \Delta x\, \Delta y\, \Delta z. \tag{3.17}$$

Then, for all coarse cells $(l, m, n) \in \text{nbh}(i, j, k)$, the value of $U^{n+1}_{l,m,n}$ is modified by

$$U^{n+1}_{l,m,n} = U^{n+1}_{l,m,n} + \frac{(1 - \Lambda_{i,j,k})\, \delta M^{L,c-f}_{i,j,k}}{m^{\text{red}}_{i,j,k}}. \tag{3.18}$$

Further, if any of the level L cells l, m, n underlays a level M cell I, J, K, $M > L$, we modify the level M solution in I, J, K by

$$U^{n+1}_{I,J,K} = U^{n+1}_{I,J,K} + \frac{(1 - \Lambda_{i,j,k})\, \delta M^{L,c-f}_{i,j,k}}{m^{\text{red}}_{i,j,k}}, \tag{3.19}$$

in other words, using the same equation as (3.18). The same expressions are used because the total amount of mass redistributed to a cell $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ on level $\mathcal{L}$, $\mathcal{L} = L$ or M, with volume fraction $\Lambda_{\mathcal{I},\mathcal{J},\mathcal{K}}$ is

$$\Lambda_{\mathcal{I},\mathcal{J},\mathcal{K}}\, \Delta x_{\mathcal{L}}\, \Delta y_{\mathcal{L}}\, \Delta z_{\mathcal{L}} \frac{(1 - \Lambda_{i,j,k})\, \Delta M^{L,c-f}_{i,j,k}}{m^{\text{red}}_{i,j,k}};$$

the update to the solution U is found by dividing this expression by $\Lambda_{\mathcal{I},\mathcal{J},\mathcal{K}}\, \Delta x_{\mathcal{L}}\, \Delta y_{\mathcal{L}}\, \Delta z_{\mathcal{L}}$.

4. NUMERICAL RESULTS

In this section we present several numerical examples showing the combined Cartesian grid integration/adaptive mesh refinement algorithm. Except as noted, the results were all obtained using the extended state formulation of the integrator and mass-weighted redistribution. In all the examples, we assume a polytropic gas with $\gamma = 1.4$, the computational cells are square, and the CFL number is 0.9.

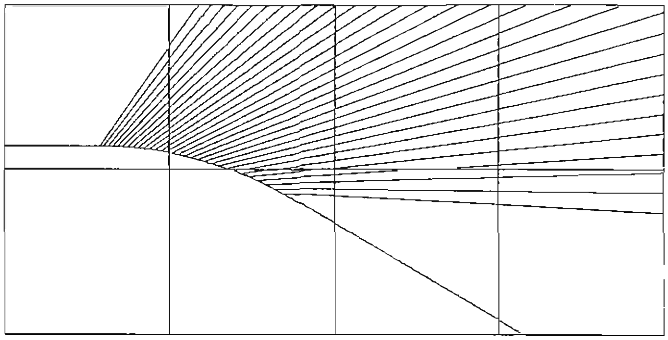

FIG. 5. Density contours from the computation of a Prandtl–Meyer expansion wave resulting from a Mach 1.2 flow turning through an angle of 30°.

4.1. Prandtl–Meyer Expansion

The first example is the calculation of the Prandtl–Meyer expansion wave resulting from a Mach 1.2 flow turning through an angle of 30°. The initial conditions of the calculation are the exact solution. The calculation is terminated when an approximate numerical steady state is reached. Figure 5 shows a contour plot of the density at late time for a uniform 160 × 80 grid; Fig. 6 displays the density in the mixed cells, i.e., the density profile along the fluid–body interface. We performed a convergence study of the algorithm using this problem by doing calculations on 80 × 40, 160 × 80, and 320 × 160 grids that were run until an approximate numerical steady state was reached, i.e., for 250, 500, and 1000 time steps, respectively. Two L_1 measures of the error in the solution at the final time

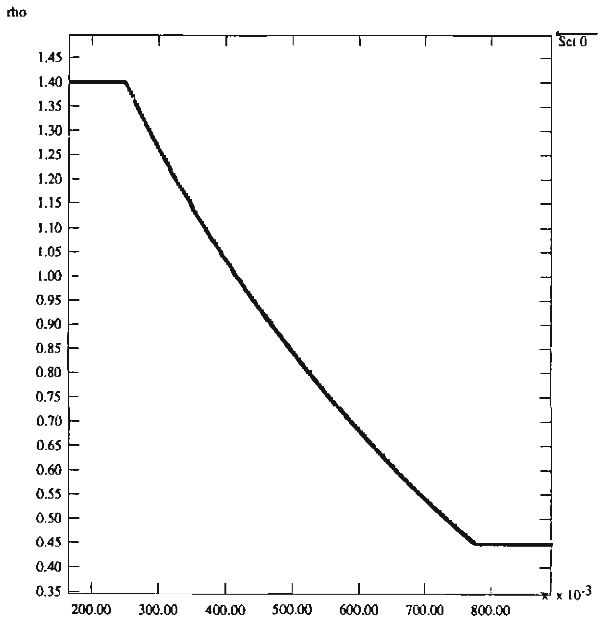

FIG. 6. Density profile along the fluid/body interface for the computed flow displayed in Fig. 5.

step were used, one over the entire domain and the other over the cells at the fluid–body interface:

$$\text{error} = \sum_{\Lambda_{i,j}>0} |u^e_{i,j} - u^c_{i,j}|\Lambda_{i,j}\,\Delta x\,\Delta y$$

$$\text{wall error} = \sum_{\Lambda_{i,j}>0,\Delta_{i,j}\in\text{bndry}} |u^e_{i,j} - u^c_{i,j}|\Lambda_{i,j}\,\Delta x,$$

where u^e and u^c are the exact and the computed solutions, respectively. The error measures are tabulated in Table I for two quantities, $\log_{10} p/\rho^\gamma$, a quantity which is proportional to entropy and stagnation enthalpy, both of which are uniform constants for the flow under consideration. The errors shown in Table I have been divided by the exact values of these two quantities. Under the assumption that the errors can be expressed as $A\,\Delta x^q$, the exponent q in h^q is computed by comparing the errors for successive grid resolutions and tabulated. The results for entropy suggest that the algorithm is second-order accurate away from the fluid–body interface and first-order accurate at the interface, although the results for stagnation enthalpy are less favorable.

4.2. Unsteady Shock Reflection

The second example is the calculation of an unsteady shock reflection resulting from a Mach 10 planar shock impinging on a ramp inclined at a 30° angle to the direction in which the shock propagates. This problem has been studied extensively; see, for example, [66]. For the sake of discussion the pre-shock values of density and pressure are $\rho_0 = 1.4$ and $p_0 = 1$.

This problem offers an opportunity to compare the results of the Cartesian grid algorithm with the results of a "body-fitted" grid method for which the body-fitted grid is quite simple, namely, a rectangular grid in which the ramp is aligned with the lower boundary of the problem domain. The higher order Godunov method [22] coupled with AMR [3] is used as the body-fitted grid method. In this approach, second-order reflecting wall boundary conditions are used to model the ramp.

We note that startup errors are present in either approach. These arise because the numerical traveling wave representation of the incident shock takes several time steps to develop. The occurrence of startup error in numerical solutions of this problem using shock-capturing methods is well documented [66, 22]. The only way to eliminate the startup error is to reinitialize the postshock state after the traveling wave has developed. We have chosen not to do this in the interest of providing an unbiased comparison.

We examine the results of three sets of calculations. We use the first set to qualitatively compare the Cartesian grid method with the body-fitted grid method (Section 4.2.1). In the Cartesian grid calculation, the incident shock is parallel to the vertical axis while the ramp is modeled as an inclined wall. In the body-fitted grid calculation, the shock is inclined at 30° to the lower boundary. The second set of calculations is used for

TABLE I

Results of Convergence Study for Prandtl–Meyer Expansion Test Problem

Grid	$\log_{10} p/\rho^{\gamma}$ Error	q	Wall error	q	Stagnation enthalpy Error	q	Wall error	q
80 × 40	0.00154	—	0.0233	—	2.22e-4	—	0.00314	—
160 × 80	0.00042	1.87	0.0108	1.11	8.05e-5	1.46	0.00193	0.70
320 × 160	0.00010	2.07	0.0049	1.14	2.70e-5	1.57	0.00124	0.63

a more detailed comparison of the two methods (Section 4.2.2). In these calculations, the ramp is parallel to the bottom boundary (Fig. 10), and modeled either with reflecting wall boundary conditions or as a Cartesian grid boundary. One final calculation (Section 4.2.2) is used for examining the size of δM in support of the heuristic arguments in Sections 2.4.2 and 2.6.1.

4.2.1. *Qualitative Comparison*

We now compare results for the unsteady shock reflection problem generated by the Cartesian grid method and by the method of [22, 3]. Figure 7 shows density contours of the solution obtained using the adaptive Cartesian grid algorithm. The domain is 3.0 units × 2.5 units, the initial shock position is 0.3 units from the left boundary, and the time displayed is $t = 0.21$. The body-fitted grid results at the same time are displayed in Fig. 8. In the second calculation the ramp is represented by a reflecting wall boundary at the bottom of the domain and the planar shock initially intersects that boundary at a 30° angle; outflow boundary conditions are used at the lower boundary upstream of the initial shock position. Each calculation uses the same size coarse level cells and the same refinement ratios in building two levels of successively finer cells, so that the effective resolution of the two calculations is the same at each level of refinement. The two calculations can serve, then, as a qualitative comparison of the Cartesian grid method with an body-fitted grid method; a quantitative comparison is inappropriate, however, because of the different representation of the lower boundary upstream of the initial shock position. The base grid for the first calculation is 60 × 50 with an effective resolution of 960 × 800 at the finest level; for the second calculation, the grid is 70 × 20 at the coarsest level and effectively 1120 × 320 at the finest. The grid locations at all three levels are shown as boxes. In each figure, 30 density contours are shown. The values of density in Fig. 7 range from ρ_0 to $15.8\rho_0$ just inside the reflected shock at the beginning of the ramp; in Fig. 8, they range from to ρ_0 to $16.0\rho_0$. The results of the Cartesian grid calculation compare favorably with the results of the other calculation. The solution shown in Fig. 8, however, does show better resolution of the flow near the ramp in the vicinity of the Mach stem and the slip line.

Figure 9 shows results at a slightly earlier time for the same problem obtained using the thin-wall algorithm formulation (Section 2.4.1) of the integrator. The calculation uses the same gridding strategy as the Cartesian grid calculation described in the previous paragraph. The results are nearly identical to those displayed in Fig. 7 except in the vicinity of the leading edge of the wall jet. In this region, the results shown in Fig. 7 are a better match to the results in Fig. 8 than those in Fig. 9. In particular, the results obtained with the thin-wall algorithm show a numerical artifact of the fluid–body boundary exerting drag on the wall jet.

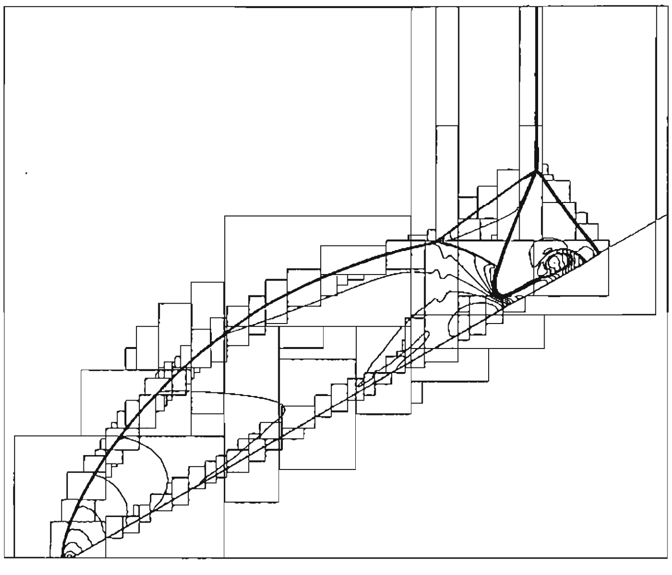

FIG. 7. Density contours from a computation of a Mach 10 shock impinging on a 30° ramp.

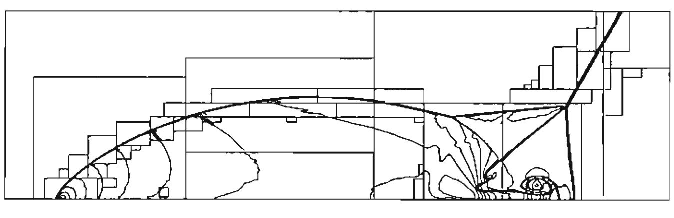

FIG. 8. Density contours from a higher order Godunov/AMR computation for the flow displayed in Fig. 7.

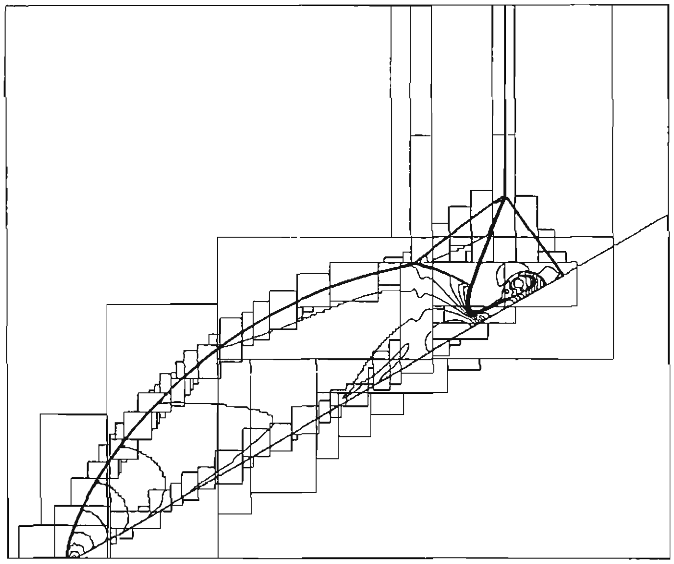

FIG. 9. Density contours from a computation of a Mach 10 shock impinging on a 30° ramp using the thin-wall algorithm.

4.2.2. *Quantitative Comparison*

A detailed comparison of the two simulations above is inappropriate because it was necessary to use different boundary conditions on the lower boundary to the left of the initial shock position in the two calculations. We therefore present results from two additional sets of calculations of the same shock reflection problem. In both sets of calculations, the ramp is parallel to the bottom boundary as in the example shown in Fig. 8. An additional fluid–body boundary that is parallel to the direction of shock propagation intersects the ramp at the same location where the planar shock initially intersects the ramp; see Fig. 10. (This same arrangement was used for the shock ramp calculation discussed in [22].) This additional interface is also represented in a Cartesian grid formulation. We note that its presence does not significantly affect our error analysis because this boundary is parallel to the direction of shock propagation and upstream from the initial position of the incident shock; hence, its only effect is to guide the flow into the bottom boundary.

In the first set of calculations, the ramp itself is aligned with the lower boundary of the domain and is represented with second-order reflecting wall boundary conditions. In the second set, the ramp is a fixed distance above the bottom boundary and hence is represented as volume and area fractions in the Cartesian grid algorithm formulation. The distance is chosen so that on a 384 × 96 grid, there is exactly one row of cells with zero volume fraction along the lower boundary; i.e., the volume fraction of the cells adjacent to the ramp is 1. Both sets of calculations use 96 × 24, 192 × 48, and 384 × 96 grids. In the Cartesian grid calculations, the cells along the ramp have volume fractions of 0.75 and 0.5, respectively, on the 96 × 24 and 192 × 48 grids. These two sets of calculations therefore allow us to do a comparative convergence study because the cells in the grids from the reflecting wall set of runs can be mapped with the cells from the Cartesian grid set of runs in a 1–1 fashion. We are interested in particular in the difference between representing the bottom boundary as a reflecting wall and as mixed or body cells. For the sake of discussion the domain is 3.2 by 0.8 units, the initial shock location is at 0.2, and the preshock values of density and pressure are 1.4 and 1. The numerical solutions discussed in the next paragraph are at $t = 0.21$.

We use results from a higher resolution calculation on a 768 × 192 grid with a reflecting wall boundary condition at the lower boundary to compute errors for the two sets of calculations described in the previous paragraph. By considering the average of the results from the 768 × 192 grid onto a coarser grid as the "exact" solution, we can calculate L_1 errors for the two sets of calculations on the entire domain and on the lower boundary. We define $e_{i,j} = |u^e_{i,j} - u^c_{i,j}|$, where u^e and u^c are the exact and computed solutions. We then define the measure of error for the entire domain by

$$L_1 = \frac{\sum_{\Lambda_{i,j}>0} e_{i,j}\Lambda_{i,j}\,\Delta x\,\Delta y}{\sum_{\Lambda_{i,j}>0} \Lambda_{i,j}\,\Delta x\,\Delta y},$$

and for the subset of the domain consisting of the cells along the lower boundary of the fluid domain by

$$L_1^b = \frac{\sum_{\Lambda_{i,j}>0,\Delta_{i,j}\in \mathrm{bndry}} e_{i,j}\Lambda_{i,j}\,\Delta x}{\sum_{\Lambda_{i,j}>0,\Delta_{i,j}\in \mathrm{bndry}} \Lambda_{i,j}\,\Delta x},$$

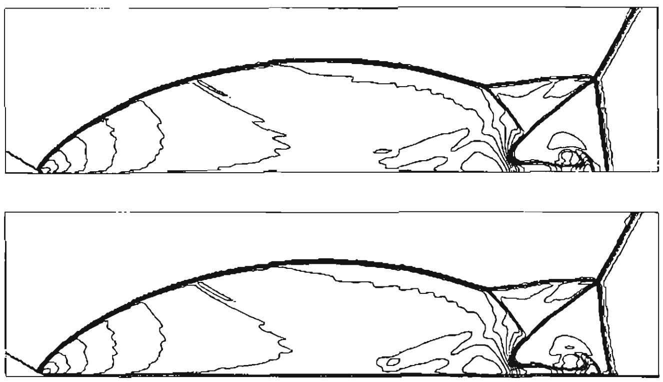

FIG. 10. Density contours from two computations of a Mach 10 shock impinging on a 30° ramp. The ramp is at the bottom boundary. The ramp is treated as a reflecting wall in the calculation that produced the upper plot and as a Cartesian grid fluid/body interface in the calculation that produced the bottom plot.

TABLE II

Results of Convergence Study for Unsteady Shock Reflection Problem: Total Error

Grid	Body-fitted grid		Cartesian grid	
	L_1	q	L_1	q
96 × 24	0.2859	—	0.4950	—
192 × 48	0.1408	1.02	0.1543	1.68
384 × 96	0.0657	1.09	0.0760	1.02

where bndry is the set of the bottommost row of cells containing fluid and/or mixed cells. Tables II and III display the values of these two errors for the quantity ρ for the two sets of calculations. Assuming the errors can be expressed as $A\Delta x^q$, the exponent q in h^q is computed by comparing the errors for successive grid resolutions and tabulated.

Because of the shocks present in the solution, we see only approximately first-order convergence in the L_1 norm for both the reflecting wall and the Cartesian grid representations of the ramp. The Cartesian grid calculations do produce larger errors than the reflecting wall calculations. The difference in the errors between the two approaches is not that large. However, we note that the bulk of the error incurred by either approach is in the vicinity of the incident and reflected shocks, the Mach stem, and the slip line, and not at the ramp.

Density contours for both methods on a 384 × 96 grid are shown in Fig. 10. The most notable difference in the numerical results is right in the vicinity of the intersection of the Mach stem with the ramp. We see in the Cartesian grid calculation that the Mach stem is skewed forward with respect to the Mach stem in the reflecting wall case. We also see that the leading edge of the wall jet exhibits the same skewing. Plots of the density and the pressure along the ramp for both 192 × 48 grid computations are displayed in Fig. 11. The results for the 768 × 192 case are also shown for comparison. We note that the magnitudes of the local extrema of density and pressure at $x \approx 0$, 2.15, 2.6, and 2.7 are poorly approximated in the Cartesian grid results. Curiously, the positions of the peaks at 2.8 and 2.9 are more poorly approximated in the body-fitted grid results. We also see the forward skewing of the Mach stem in the Cartesian grid results.

TABLE III

Results of Convergence Study for Unsteady Shock Reflection Problem: Error at Lower Boundary

Grid	Body-fitted grid		Cartesian grid	
	L_1^b	q	L_1^b	q
96 × 24	0.7824	—	0.8576	—
192 × 48	0.4522	0.79	0.5482	0.64
384 × 96	0.2319	0.96	0.2590	1.08

The representation of the geometry we used in this section truly is a simple rotation of the Cartesian grid representation used in the previous section, and some additional comparison of results is possible. In Fig. 12, we plot the density and the pressure along the ramp for the 768 × 192 case and the Cartesian grid calculation in Section 4.2.1, which has an effective resolution of 960 × 800 on a 3-unit by 2.5-unit domain. The Cartesian grid results compare favorably with the body-fitted grid results. There is some oscillation in the Cartesian grid results at the local extrema at $x \approx 2.15$ and 2.7. Further, even with the increased resolution the local extrema at $x \approx 2.15$, 2.6, and 2.7 are still underresolved, and the positions of the extrema at 2.6 and 2.7 lag those in the reflecting wall case. The dip in the 768 × 192 results at $x \approx 1.7$ is due to startup error.

4.2.3. *Analysis of the Magnitude of δM*

In this section we present numerical results which suggest that the two-dimensional analogue of criterion (2.18), i.e., that the components of $U_{i,j}^{n+1} - U_{i,j}^{n+1,\text{ref}}$ computed by the method should be small compared to the corresponding components of $U_{i,j}^{n+1,\text{ref}} - U_{i,j}^{n}$, is satisfied. We present two different measures for determining if (2.18) is satisfied:

$$m_1 = \frac{\sum_{(i,j)\in\mathfrak{R}} |U_{i,j}^{n+1,r} - U_{i,j}^{n+1,\text{ref},r}|}{\sum_{(i,j)\in\mathfrak{R}} |U_{i,j}^{n+1,r} - U_{i,j}^{n,r}|} \tag{4.20}$$

and

$$m_\infty = \frac{|U_{I,J}^{n+1,r} - U_{I,J}^{n+1,\text{ref},r}|}{|U_{I,J}^{n+1,r} - U_{I,J}^{n,r}|} \tag{4.21}$$

such that $(I, J) \in \mathfrak{R}$ and

$$|U_{I,J}^{n+1,r} - U_{I,J}^{n+1,\text{ref},r}| \geq |U_{i,j}^{n+1,r} - U_{i,j}^{n+1,\text{ref},r}| \quad \forall (i,j) \in \mathfrak{R},$$

where $\mathfrak{R}$ denotes the set of all mixed cells and fluid cell neighbors of mixed cells, and U^r denotes the rth component of U. The two measures can be considered L_1 and L_∞ norms. (We chose the measure (4.21) rather than a simple maximum proportion because that maximum could be arbitrarily large for arbitrarily small values of the two terms in the proportion.)

Figure 13 displays these measures as functions of time for a computation on a uniform 240 × 200 grid of the problem discussed in Section 4.2.1 and displayed in Fig. 7. The values of m_1 for ρ, ρu, and ρv are small and show linear growth in time while the values for ρE are somewhat larger, but still small, and decay linearly in time. The linear growth is probably due to the self-similar nature of the solution, i.e., the number of cells for which $U^{n+1} \neq U^n$ grows linearly in time. We lack

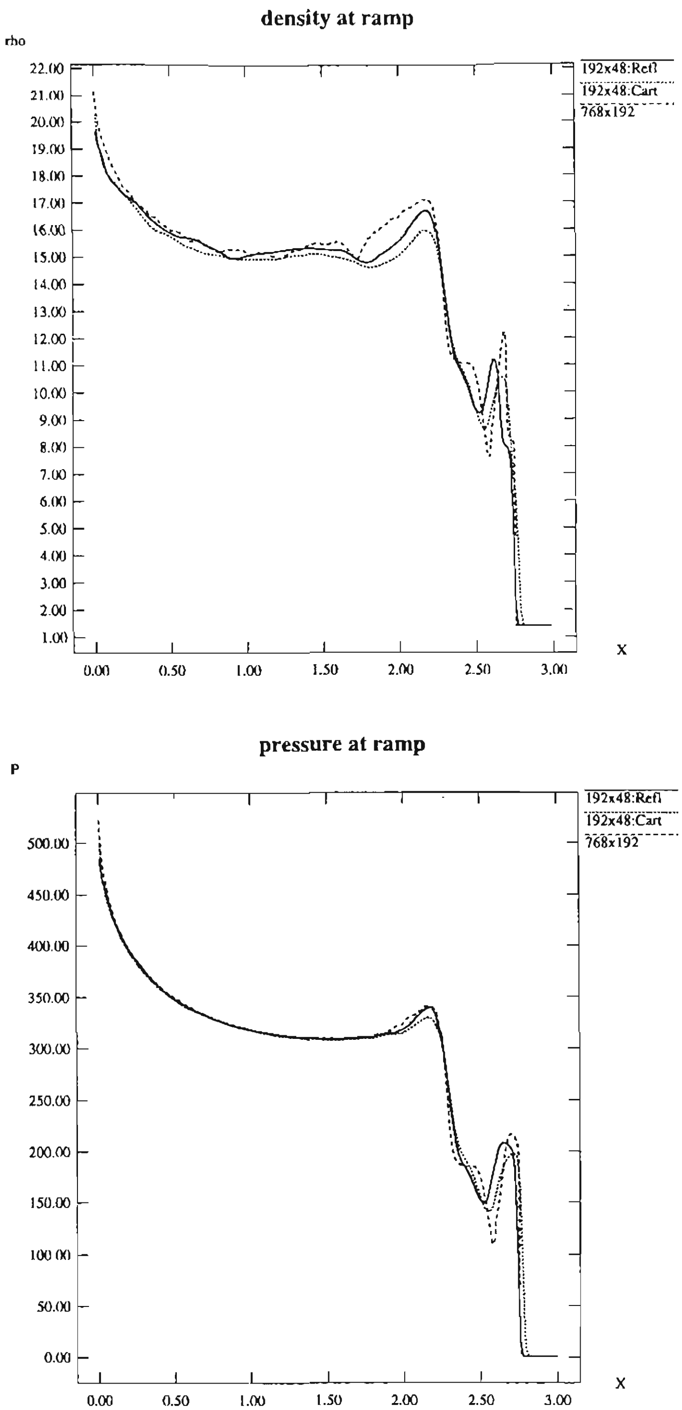

FIG. 11. Density and pressures profiles along the fluid/body interface from a computation of a Mach 10 flow past a 30° ramp. The ramp is represented either with reflecting wall boundary conditions ("192 × 48.Refl." "768 × 192") or as a horizontal Cartesian grid fluid/body interface. ("192 × 48.Cart").

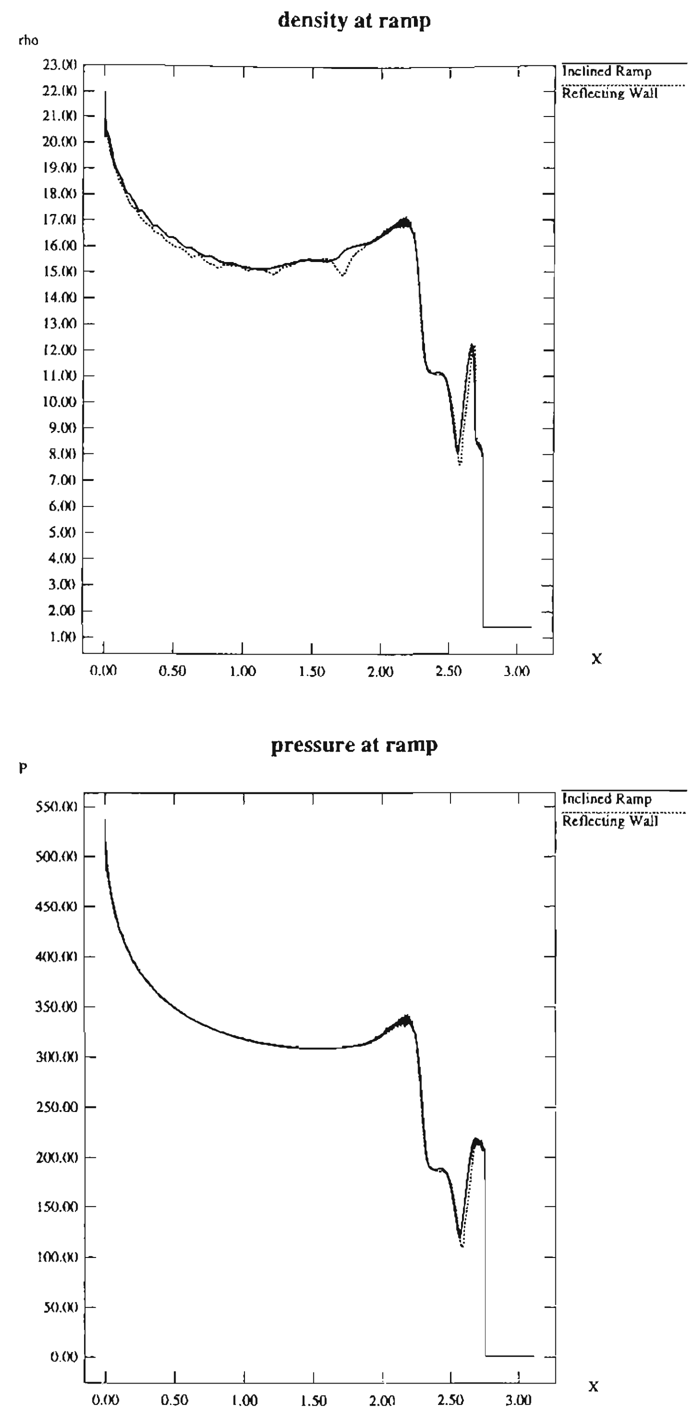

FIG. 12. Density and pressures profiles along the fluid/body interface from a computation of a Mach 10 flow past a 30° ramp. The ramp is represented either as a reflecting wall ("Reflecting Wall") or as an inclined Cartesian fluid/body interface ("Inclined Ramp").

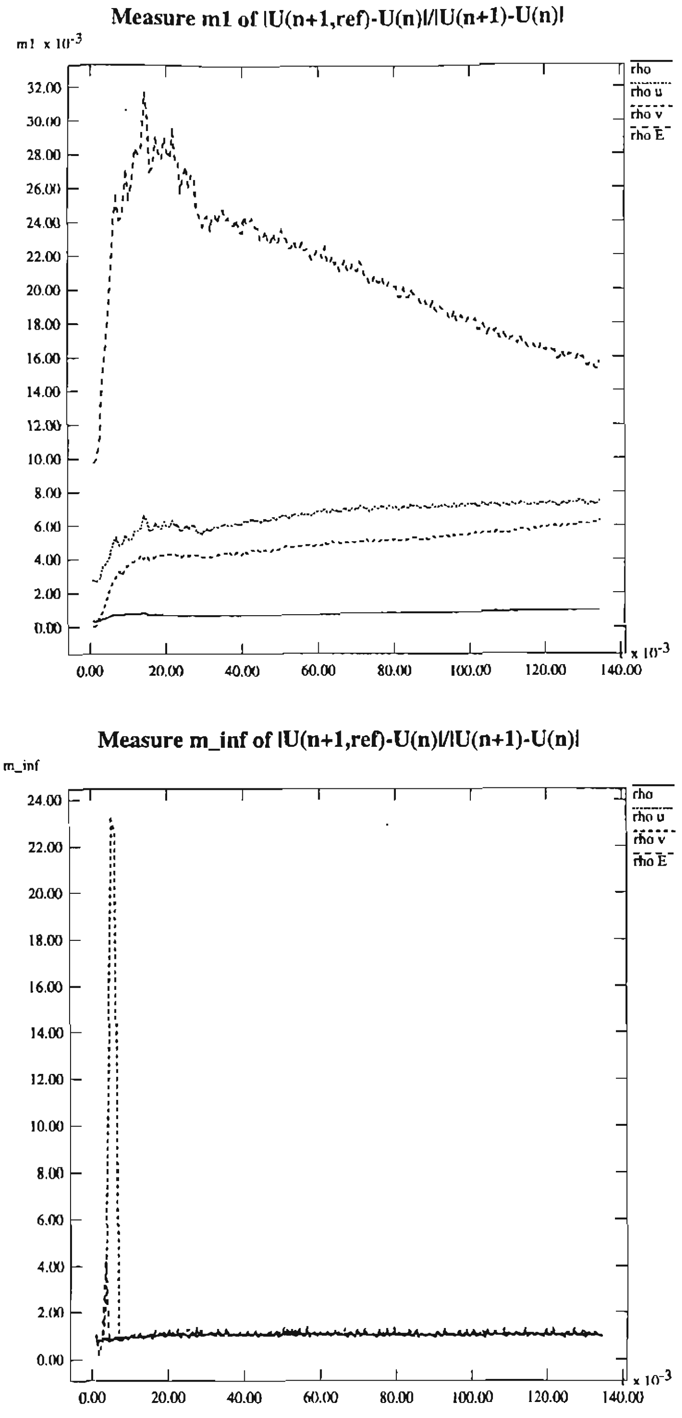

FIG. 13. Values of m_1 (top) and m_∞ (bottom) from a computation of a Mach 10 flow past a 30° ramp on a 240 × 200 uniform grid in which the ramp is represented as an inclined Cartesian fluid/body interface as in Fig. 7.

an explanation why the value for ρE should decay linearly. The values of m_∞ are all approximately $O(1)$ at large time; all these values are attained in the vicinity of the Mach stem. We conclude for this calculation that (2.18) is satisfied in an L_1 sense but not in an L_∞ sense.

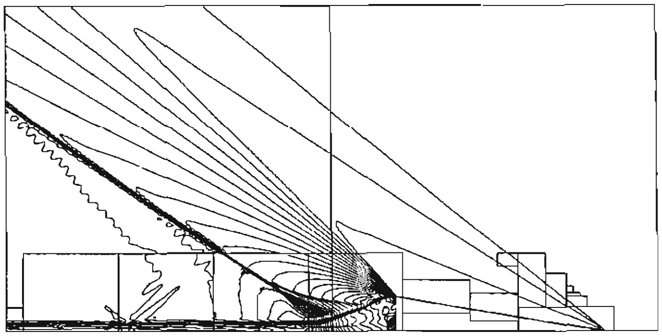

FIG. 14. Density contours from an $r - z$ computation of a Mach 1.597 flow past a cone with a semi-apex angle of 9.5°.

4.3. Axisymmetric Flow Past a Cone

The third example is the computation of an axisymmetric flow, $M_\infty = 1.597$, past a cone with a semi-apex angle of 9.5°. The initial conditions are those of an impulsive start, i.e., $M = 1.597$ throughout the domain. The flow is calculated using the cylindrical coordinates formulation of the Cartesian grid algorithm. The computation used a base grid of 192 × 96 cells and adaptive mesh refinement with a single level of finer grids so that the effective resolution is 768 × 384 cells. For the sake of discussion the domain is 128 by 64 units, the base of the cone has a radius of 6.86 units, and the upstream values of pressure and density, p_∞ and ρ_∞, are 1.0 and 1.4, respectively. The apex angle is such that the leading part of the cone is represented entirely by mixed cells; the first body cell is located 1.083 units downstream of the cone vertex.

Figure 14 depicts the density field at $t = 206.1$. The coarse and fine grids are shown again as boxes. Thirty density contours are shown. The density ranges from a minimum of $0.13\rho_\infty$ in the vortex adjacent to the base to a maximum of $1.27\rho_\infty$ near the z-axis just downstream from the recompression shock. The results of the calculation appear to compare favorably with the schlieren photograph of experimental results for the same configuration shown in Kopal [38]. However, the computed flow in the wake region is not as complex as the wake flow observed experimentally because the computation is axially symmetric and, hence, three-dimensional effects are not represented.

In Fig. 15, we examine the density more closely at several axial locations along the cone at $t \approx 206.1$. z is the distance downstream from the cone vertex; r is the distance from the cone centerline. At each axial location, values are plotted up to and including the mixed cell adjacent to the body. We recall that a conical shock weakens as one moves away from the center of the disturbance. Hence we expect to see monotonically decreasing profiles in density. We do at the first axial location. At the second location, which corresponds to the computational cell just upstream from the first full body cell in the cone, we see a large undershoot near the centerline. The undershoot spans

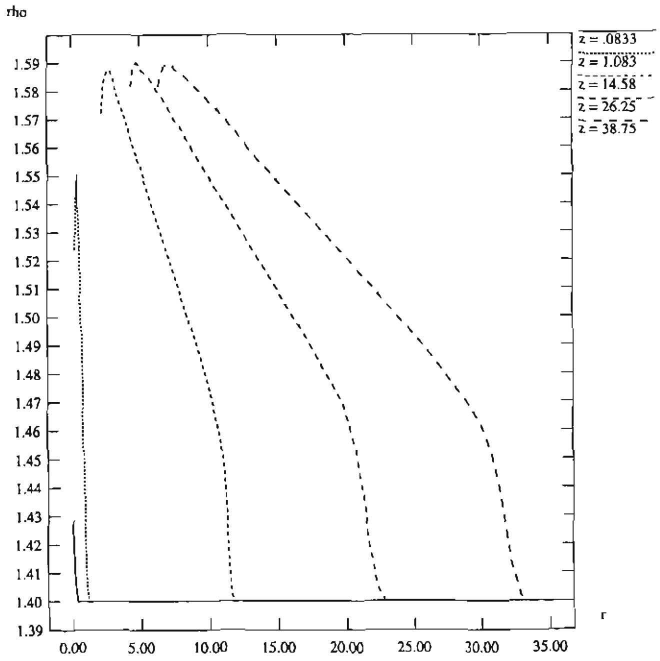

FIG. 15. Density at late time at several axial locations for the Mach 1.597 axisymmetric flow past a cone with a semi-apex angle of 9.5°. z is the distance downstream from the cone vertex; r is the distance from the cone centerline.

two computational cells (note that the cell width is 0.16667 units). This second location is the location of maximum undershoot along the cone surface. As we move further downstream, the undershoot continues to persist over the two computational cells nearest the fluid–body boundary although its magnitude lessens. The last position shown is just upstream from the base of the cone. The magnitude of the density undershoot appears to be consistent with an explanation that the undershoot is a wall heating error [51, 48] due to the underlying shock-capturing method.

4.4. Three-Dimensional Flow Past a Cone-Cylinder

The last example is a three-dimensional flow, $M_\infty = 2.33$, past a cone-cylinder with a semi-apex angle of 8.58°. The initial conditions are those of an impulsive start. The cone is given a pitch of 0.06° and a yaw of 0.03° in order to eliminate both planes of symmetry. The computation used a base grid of $36 \times 12 \times 12$ and adaptive mesh refinement with two refinements of 2 and 4 so that the effective resolution is $288 \times 96 \times 96$. For the sake of discussion the domain is $24 \times 7 \times 7$ units, the base of the cylinder is 1.8 units in diameter, and the upstream value of ρ is $\rho_\infty = 1.4$. The vertex of the cone is aligned with a grid line so that the surface normals are unambiguously defined. The apex angle is such that the leading part of the cone is represented entirely by mixed cells; the first body cell is located 0.792 units downstream of the cone vertex.

The density field of the flow at late time ($t \approx 37.5$) is shown in Fig. 16. Four raster plots of longitudinal slices of the density field, one axial and three off-axis, are shown. The centers of the cells in the four slices are, respectively, 0.7917, 0.5417, 0.2917, and 0.04167 units off-axis (note that the width of a cell at the finest level is 0.08333 units). The jaggedness of the boundary of the cone is an artifact of the graphics which represents mixed cells as full cells. The three sizes of raster cells correspond to the sizes of the computational cells used in the different levels of refinement. The color map used in the figure is a linear grey scale. The density ranges from a minimum of $0.20\rho_\infty$ (shown as black) in the wake region to $1.19\rho_\infty$ (white) on the surface of the cone near the base. The computed density at the nose tip is approximately $0.8\rho_\infty$.

The development of the flow in time is shown in Fig. 17 through a sequence of raster plots of the density in an axial longitudinal slice of the flow. The individual plots correspond to times $t \approx 7.5$, 22.5, 37.5, and 54. The color map is a grey scale modified to show more detail at the lower end of the range. (For comparison, the bottom left plot corresponds to the bottom right plot in Fig. 16.) The density ranges from $0.17\rho_\infty$ to $1.20\rho_\infty$. The late time calculated results in the axial longitudinal slice show qualitative visual agreement with the schlieren photograph of the experimental results using the same configuration reported in Shapiro [62].

In Fig. 18, we examine the density more closely in several vertical slices of an axial longitudinal slice at $t \approx 45$. x is the distance downstream from the cone vertex; y is the distance from the cone-cylinder centerline. At each axial location, values are plotted up to and including the mixed cell adjacent to the body. As in the axisymmetric calculation discussed in Section 4.3, we expect to see monotonically decreasing profiles in density. We do at the first two axial locations. At the third location, which corresponds to the computational cell just upstream from the first full body cell in the cone, we see a dramatic undershoot near the centerline. The undershoot spans two computational cells. This third location is the location of maximum undershoot along the cone surface. As we move further downstream, the undershoot continues to persist over the two computational cells nearest the fluid–body boundary although its magnitude lessens. As in the case of the axisymmetric calculation in Section 4.3, the size of the undershoot is consistent with an explanation that it is a wall heating error [51, 48]. The last position shown is just upstream from the base of the cone. Downstream of that location the density should be monotonically increasing in the vicinity of the boundary due to the presence of a rarefaction. Hence, separating numerical artifacts from the solution is more difficult along the surface of the cylinder and we do not attempt to do so.

5. CONCLUSIONS AND DISCUSSION

We have presented an explicit, adaptive Cartesian grid method for calculating time-dependent inviscid compressible flows in irregular domains. The methodology is applicable to rectangular coordinates in two and three dimensions, as well

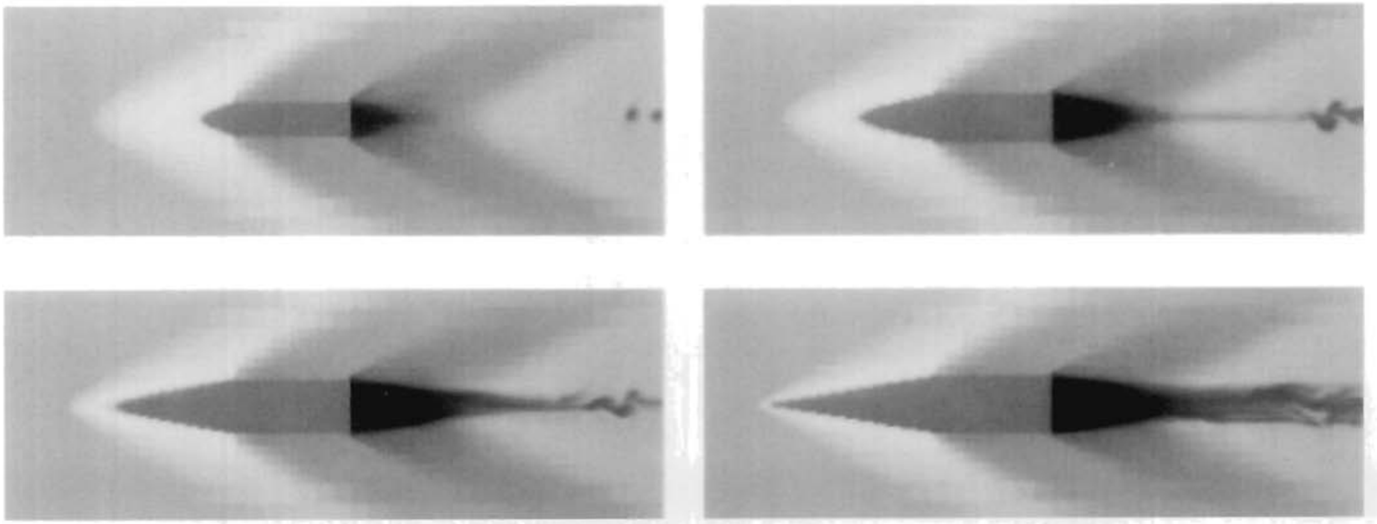

FIG. 16. Longitudinal slices of density field for a Mach 2.33 flow past a cone-cylinder with a semi-apex angle of 8.58°.

as cylindrical coordinates in two dimensions. We believe a strength of our approach is that the implementation of both the single grid scheme and the modifications to the AMR algorithm in three space dimensions or cylindrical coordinates presents little additional complexity over the implementation in two-gimensional rectangular coordinates. Some of the numerical results (Section 4.1) suggest that the method may be globally second-order accurate and first-order accurate at the boundary in smooth flows.

We describe both a single grid algorithm and a coupling of that algorithm to an adaptive mesh refinement scheme. The single grid algorithm (Section 2.2) uses the higher order Godunov method [22, 61] (Section 2.3) to which it is equivalent away from the fluid–body interface. A variation (Section 2.6) of the redistribution algorithm in [15] enables the algorithm to remain stable using the full cell time step even in the presence of arbitrarily small cells. The single grid algorithm is coupled to an extension of the adaptive mesh refinement algorithm of Bell *et al.* [3] modified so that conservation is maintained in the presence of geometrically irregular boundaries (Section 3).

We have used the algorithm to calculate two- and three-dimensional flows (Section 4). The results show that the algorithm is an effective means of calculating inviscid compressible flows in irregular domains as long as the lower resolution of effects at the fluid–body interface is acceptable. The numerical results for a Prandtl–Meyer expansion problem (Section 4.1) suggest that for smooth flows the method is globally second-order accurate and first-order accurate at the fluid–body bound-

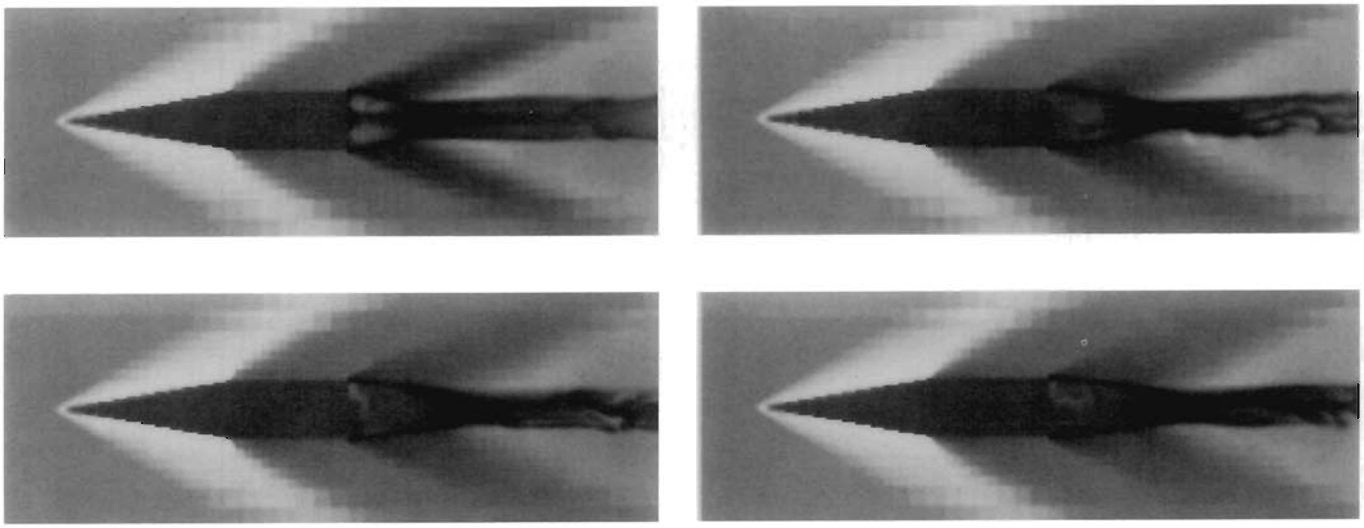

FIG. 17. Time sequence of density field slice for a Mach 2.33 flow past a cone-cylinder with a semi-apex angle of 8.58°. The times shown are $t \approx 7.5$, 22.5, 37.5, and 54 corresponding to top left and right, and bottom left and right.

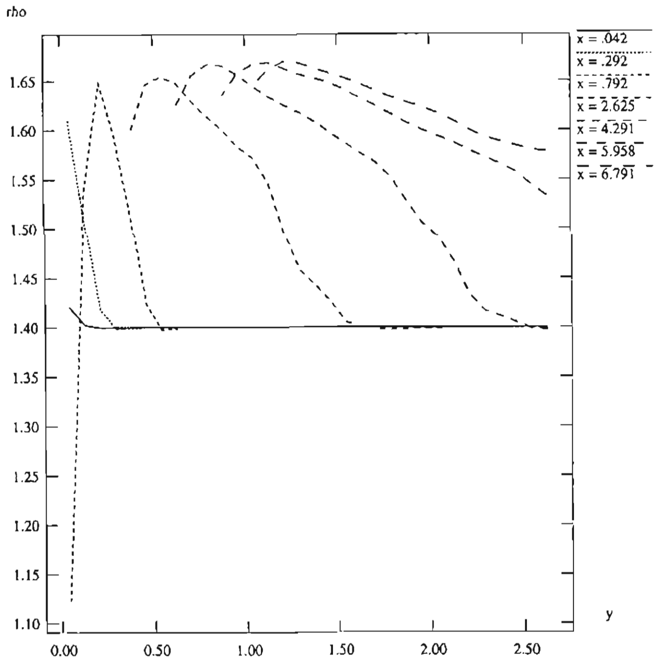

FIG. 18. Density at late time at several axial locations in axial longitudinal slice for the Mach 2.33 flow past a cone-cylinder with a semi-apex angle of 8.58°. x is the distance downstream from the cone vertex; y is the distance from the cone-cylinder centerline.

ary. The method of computing the reference state (Section 2.4) is an important factor in attaining reasonable numerical results, since poor choices for the reference state can clearly result in gross inaccuracies (Section 2.4.2). In two other examples (Sections 4.3, 4.4) the method computes a numerical boundary layer in that the density is underestimated along portions of the fluid–body interface. The amount by which the density is underestimated in these two examples is consistent with an explanation that the undershoot is a wall heating error [51, 48] arising because the underlying higher order Godunov method is a shock-capturing scheme with inherent numerical viscosity.

Our methodology can be extended to different basic integration schemes. Although we present the single grid algorithm as being built upon the higher order Godunov method in [22], the method can in fact be used in conjunction with any reasonable shock-capturing difference scheme on regular grids for hyperbolic conservation laws as long a the alternate scheme has some mechanism for generating a flux at a fluid–body interface, i.e., in the case of gas dynamics, a pressure. A representative, but far from exhaustive list of these methodologies includes FCT [13], ENO [32], the Roe scheme [59], and MUSCL [64]. Yee [67] gives a very thorough discussion of modern shock-capturing schemes. The Cartesian grid extensions to adaptive mesh refinement techniques, on the other hand, are very specific to the approach in [8], although not specific to the use of any particular underlying shock capturing scheme.

The authors are currently pursuing several areas of research in Cartesian grid methods. We are developing an interface between the code implementing the algorithms described in this paper and a commercial CAD/CAM package so that the volume and area fraction data needed by the code can be produced from a CAD/CAM description of the problem geometry. The algorithm in this paper is also the basis for a code that simulates a pulse combustor [45]. An implementation of a compressible Navier–Stokes version of the algorithm is in progress as well. There are also investigations [1] in developing Cartesian grid algorithms for incompressible flow using higher order projection methods [18, 4]. We are also extending the methodology to multifluid flows [31] using a formulation in which the base integration scheme is an operator-split Godunov scheme rather than the unsplit scheme of [22, 61]. This formulation also uses a simple extension for non-homogeneous hyperbolic systems. Finally, we are working on an implementation of the algorithm that overcomes the current limitations on geometric complexity (Sections 2.1, 2.4.1, 2.6.1), in particular, the inability to handle long thin sections of body. The new implementation uses domain-cut techniques similar to those employed in the TRANAIR code [68]. In this approach, each computational cell intersected by the fluid–body boundary is divided into one or more regions, one for each truly separate region of the fluid intersected by the cell. Each region has its own fluid state, geometry, and a list of the cells in the domains of influence and dependence of the region.

ACKNOWLEDGMENTS

We thank Marsha Berger for a number of helpful comments and discussions during the course of this research. We also thank Ann Almgren for her thorough reading of the manuscript.

REFERENCES

1. A. Almgren, J. B. Bell, P. Colella, and T. Marthaler, *SIAM J. Sci. Comput.*, submitted; UCRL JC-118091, Lawrence Livermore National Laboratory, June 1994.
2. S. A. Bayyuk, K. G. Powell, and B. van Leer, in *Proceedings, AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL, July 1993*, p. 423, AIAA Paper 93-3391-CP (unpublished).
3. J. B. Bell, M. J. Berger, J. S. Saltzman, and M. Welcome, *SIAM J. Sci. Comput.* **15**, 127 (1994).
4. J. B. Bell, P. Colella, and H. M. Glaz, *J. Comput. Phys.* **85**, 257 (1989).
5. J. B. Bell, P. Colella, and J. Trangestein, *J. Comput. Phys.* **82,** 362 (1989).
6. J. B. Bell, P. Colella, J. A. Trangenstein, and M. Welcome, in *Proceedings, AIAA 9th Computational Fluid Dynamics Conference, Buffalo, New York, June 14–16, 1989*, p. 471.
7. J. B. Bell, P. Colella, and M. Welcome, in *Proceedings, AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24–27, 1991*, p. 814.
8. M. J. Berger and P. Colella, *J. Comput. Phys.* **82,** 64 (1989).
9. M. J. Berger and A. Jameson, *AIAA J.* **23**(2), 561 (1984).
10. M. J. Berger and R. J. LeVeque, in *Proceedings, AIAA 9th Computational Fluid Dynamics Conference, Buffalo, New York, June 14–16, 1989*, p. 1.

11. M. J. Berger and R. J. LeVeque, in *Proceedings, AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24–27, 1991*, p. 1; AIAA Paper 91-1602.
12. M. J. Berger and R. J. LeVeque, *Comput. Systems Eng.* **1,** 305 (1990).
13. J. Boris and D. Book, *J. Comput. Phys.* **11,** 38 (1973).
14. D. A. Caughey and A. Jameson, *AIAA J.* **18**(11), 1281 (1980).
15. I. Chen and P. Colella, UCRL JC-97200, Lawrence Livermore National Laboratory, July 1987 (unpublished).
16. Y-L. Chiang, B. van Leer, and K. G. Powell, Technical Report AIAA Paper 92-0443-CP, 1992 (unpublished).
17. S. Choi and B. Grossman, in *AIAA 26th Aerospaces Meeting, Reno, NV, Jan. 1988*, AIAA Paper 88-0227.
18. A. Chorin, *Math. Comput.* **22,** 745 (1969).
19. D. K. Clarke, M. D. Salas, and H. A. Hassan, *AIAA J.* **24**(3), 353 (1986).
20. W. J. Coirier and K. G. Powell, in *Proceedings, AIAA 11th Computational Fluid Dynamics Conference. Orlando, FL, July 1993*, p. 423.
21. P. Colella, *SIAM J. Sci. Stat. Comput.* **6,** 104 (1985).
22. P. Colella, *J. Comput. Phys.* **87,** 171 (1990).
23. P. Colella, R. E. Ferguson, and H. M. Glaz, in preparation.
24. P. Colella and H. M. Glaz, *J. Comput. Phys.* **59,** 264 (1985).
25. P. Colella and P. R. Woodward, *J. Comput. Phys.* **54,** 174 (1984).
26. W. Y. Crutchfield and M. Welcome, *Sci. Prog.* **2,** 145 (1993).
27. B. Epstein, A. L. Luntz, and A. Nachson, *AIAA J.* **30**(3), 679 (1992).
28. S. Falle and J. Giddings, "An Adaptive Multigrid Applied to Supersonic Blunt Body Flow," in *Numerical Methods in Fluid Dynamics, III*, edited by K. W. Morton and M. J. Baines (Clarendon Press, Oxford, 1988).
29. R. L. Gaffney, H. A. Hassan, and M. D. Salas, in *AIAA 25th Aerospaces Meeting, Reno, NV, Jan. 1987*, AIAA Paper 87-0356.
30. C. F. Gooch and H. Öksüzoğlu, in *Proceedings, AIAA 11nd Computational Fluid Dynamics Conference. Orlando, FL, July 1993*, p. 832.
31. J. A. Greenough, V. Beckner, R. B. Pember, W. Y. Crutchfield, J. B. Bell, and P. Colella, in *Proceedings, AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA, June 1995*, p. 817.
32. A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *J. Comput. Phys.* **71,** 231 (1987).
33. T. Holst and W. Balhaus, *AIAA J.* **17**(2), 145 (1979).
34. A. Jameson, in *Proceedings, AIAA 2nd Computational Fluid Dynamics Conference. June 1975*, p. 148.
35. A. Jameson and T. J. Baker, in *AIAA 25th Aerospaces Meeting, Reno, NV, Jan. 1987*, AIAA Paper 87-0452.
36. A. Jameson, T. J. Baker, and N. P. Weatherill, in *AIAA 24th Aerospaces Meeting, Reno, NV, Jan. 1986*, AIAA Paper 86-0103.
37. F. T. Johnson, R. M. James, J. E. Bussoletti, A. C. Woo, and D. P. Young, AIAA Paper 82-0953, 1982 (unpublished).
38. Z. Kopal, *Tables of Supersonic Flow Around Cones* (MIT Press, Cambridge, MA, 1947).
39. R. J. LeVeque, *SIAM J. Numer. Anal.* **22,** 1051 (1985).
40. R. J. LeVeque, "Cartesian Grid Methods for Flow in Irregular Regions," in *Numerical Methods in Fluid Dynamics, III*, edited by K. W. Morton and M. J. Baines, (Clarendon Press, Oxford, 1988), p. 375.
41. R. J. LeVeque, *J. Comput. Phys.* **78,** 36 (1988).
42. R. J. LeVeque, *Numerical Methods for Conservation Laws* (Birkhauser, Boston, 1990).
43. R. Löhner, K. Morgan, and O. C. Zienkiewicz, "Adaptive Grid Refinement for the Compressible Euler Equations," in *Accuracy Estimates and Adaptivity for Finite Elements*, edited by I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. de A. Oliveira (Wiley, New York, 1986), p. 281.
44. R. Löhner and P. Parikh, AIAA Paper 88-0515, 1988 (unpublished).
45. D. L. Marcus, R. B. Pember, V. Beckner, J. B. Bell, D. Simkins, and M. Welcome, AIAA Paper 94-2315, 1944 (unpublished).
46. D. J. Mavriplis, *AIAA J.* **28**(2), 213 (1990).
47. J. E. Melton, F. Y. Enomoto, and M. J. Berger, in *Proceedings, AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL, July 6–9, 1993*, p. 959; AIAA Paper 93-3385-CP.
48. R. Menikoff, *SIAM J. Sci. Comput.* **15**(5), 1227 (1993).
49. K. Morinishi, *Comput. Fluids* **21**(3), 331 (1992).
50. W. F. Noh, "Cel: A Time-Dependent, Two-Space-Dimensional, Coupled Eulerian–Lagrangian code," in *Fundamental Methods of Hydrodynamics, Methods of Computational Physics, Vol. 3*, (Academic Press, New York/London, 1964), p. 117.
51. W. F. Noh, *J. Comput. Phys.* **72,** 78 (1987).
52. H. Öksüzoğlu, AIAA Paper, 92-0326, 1992 (unpublished).
53. S. Osher and F. Solomon, *Math. Comput.* **38,** 339 (1982).
54. R. Pember, *SIAM J. Sci. Comput.* **14**(4), 824 (1993).
55. J. W. Purvis and J. E. Burkhalter, *AIAA J.* **17**(11), 1170 (1979).
56. J. J. Quirk, Technical Report ICASE Report 94-51 (unpublished).
57. J. J. Quirk, Ph.D. thesis, Cranfield Institute of Technology, UK, 1991 (unpublished).
58. J. J. Quirk, *J. Comput. Phys.* **23**(1), 125 (1994).
59. P. L. Roe, *J. Comput. Phys.* **43,** 357 (1981).
60. P. E. Rubbert, J. E. Bussoletti, F. T. Johnson, K. W. Sidwell, W. S. Rowe, S. S. Samant, G. SenGupta, W. H. Weatherill, R. H. Burkhart, B. L. Everson, D. P. Young, and A. C. Woo, "A New Approach to the Solution of Boundary Value Problems Involving Complex Configurations," in A. K. Noor, editor, *Computational Mechanics—Advances and Trends*, AMD, Vol. 75, edited by A. K. Noor (Clarendon Press, Oxford, 1986), p. 49.
61. J. S. Saltzman, *J. Comput. Phys.* **115,** 153 (1994).
62. A. H. Shapiro, *The Dynamics and Thermodynamics of Compressible Fluid Flow* (Ronald Press, New York, 1954).
63. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *J. Comput. Phys.* **47,** (1982).
64. B. van Leer, *J. Comput. Phys.* **32,** 101 (1979).
65. B. Wedan and J. South, in *Proceedings, AIAA 6th Computational Fluid Dynamics Conference. Danvers, MA, July 1993*; AIAA Paper 83-1889.
66. P. R. Woodward and P. Colella, *J. Comput. Phys.* **54,** 115 (1984).
67. H. C. Yee, Technical Report NASA Technical Memorandum 101088, 1989; also as NASA-TM-89464, May 1987.
68. D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti, *J. Comput. Phys.* **62,** (1991).
69. D. D. Zeeuw and K. G. Powell, *J. Comput. Phys.* **104,** (1993).