ADAPTIVE CARTESIAN GRID METHODS FOR REPRESENTING GEOMETRY IN INVISCID COMPRESSIBLE FLOW*

Richard B. Pember‡
Lawrence Livermore National Laboratory
Livermore, CA 94550

John B. Bell†
Lawrence Livermore National Laboratory
Livermore, CA 94550

Phillip Colella§
University of California Berkeley
Berkeley, CA 94720

William Y. Crutchfield‡
Lawrence Livermore National Laboratory
Livermore, CA 94550

Michael L. Welcome‡
Lawrence Livermore National Laboratory
Livermore, CA 94550

Abstract

In this paper we describe a Cartesian grid algorithm for modeling time-dependent compressible flow in complex geometry. In this approach problem geometry is treated as an interface embedded in a regular Cartesian mesh. The discretization near the embedded boundary is based on a volume-of-fluid approach with a redistribution procedure to avoid time-step restrictions arising from small cells where the boundary intersects the mesh. The algorithm is coupled to an unsplit second-order Godunov algorithm and is fully conservative, maintaining conservation at the boundary. The Godunov / Cartesian grid integration scheme is coupled to a local 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 / local refinement algorithm for both two- and three-dimensional flows are presented.

Introduction

A broad range of engineering applications require the modeling of inviscid compressible flows in complex geometries. When viscous effects and heat transfer are important considerations a regular body-fitted grid is needed to accurately resolve the flow but when these effects are not critical a simplified procedure can be used. One approach is to use a fully unstructured grid in which triangles (or tetrahedra in three dimensions) are used to tessellate the flow domain. There are many references in the literature on this approach; the interested reader is referred to, for example, Jameson et al [1] and Lohner and Parikh [2]. An alternative approach is to maintain a uniform computational mesh and treat the problem geometry as a specialized boundary embedded in the mesh. This approach was first proposed by Wedan and South [3]. The interested reader is referred to recent work by Berger and LeVeque [4] and by Zeeuw and Powell [5] for a discussion of the more recent literature in this area.

In this paper we present a Cartesian grid / embedded boundary method by using ideas previously developed for shock tracking by Chern and Colella [6] and by Bell, Colella and Welcome [7]. In this approach we view the boundary as a "tracked front" in

* 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. Prof. Colella was supported at UC Berkeley by DARPA and the National Science Foundation under grant DMS-8919074; and 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-FGO3-92ER25140.

† Group Leader, Applied Mathematics Group

§ Associate Professor, Dept. of Mechanical Engineering

‡ Staff Scientist, Applied Mathematics Group

a regular Cartesian grid with the dynamics of the boundary given by specified boundary conditions such as, for example, a stationary reflecting wall. The discretization near the boundary uses a volume-of-fluid approach combined with an unsplit second-order Godunov difference method that is fully conservative. This basic integration methodology is coupled to a local adaptive mesh refinement algorithm given by Bell et al [8]. For this algorithm the geometry is specified by defining, for each cell in the computational mesh, the volume fraction of the cell inside the flow domain and the area fraction of each face of the cell that is in the computational domain.

The key technical difficulty in developing this approach for time-accurate flows arises from small cells that can occur when the geometry is overlaid on the mesh. There is essentially no control over that process and arbitrarily small cells can be formed. Our approach for dealing with this problem uses a variation of the algebraic redistribution algorithm of Chern and Colella [6] to conservatively distribute the update of small cells to their neighbors to maintain conservation. This allows the scheme to use time steps computed from CFL considerations on the uniform grid.

In the next section we provide an overview of the basic integration method for embedded boundaries. The details of the algorithm are discussed in section 3. In section 4, we describe the coupling of the basic integration algorithm to the local adaptive mesh refinement algorithm of Bell et al [8]. The last section of the paper presents numerical examples for both two- and three-dimensional flows.

Overview of the Integration Algorithm

In this section we describe the basic second-order Godunov algorithm with an embedded boundary in a summary form. Although the methodology is applicable to other systems of conservation laws such as, for example, magnetohydrodynamics, we will 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 will describe the method for the three-dimensional case. The restriction to two dimensions is straightforward, and the modification for cylindrical coordinate will sketched in section 3. Thus, we want to solve

$$\frac{\partial U}{\partial t} + \frac{\partial F^x}{\partial x} + \frac{\partial F^y}{\partial y} + \frac{\partial F^z}{\partial z} = 0 \qquad (2.1)$$

where, in the case of gas dynamics,

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \quad F^x(U) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uE + up \end{bmatrix}$$

$$F^y(U) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vE + vp \end{bmatrix} \quad F^z(U) = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wE + wp \end{bmatrix}.$$

The representation of the solution consists of two components, the state vector U and the representation of the geometry. Thus, for each cell Δ_{ijk} in the computational mesh we store the state vector U_{ijk} and volume fractions and "apertures" which provide the geometric description of the domain needed by the integration algorithm. The volume fraction Λ_{ijk} is the fraction of cell Δ_{ijk} that is inside the flow domain. Thus, Λ_{ijk} equals 1 for cells completely inside the flow domain, 0 for cells completely outside the flow domain and an appropriate fraction for mixed cells. The apertures $A_{i+\frac{1}{2},j,k}$, $A_{i,j+\frac{1}{2},k}$, and $A_{i,j,k+\frac{1}{2}}$ specify the fraction of the area of the faces of each cell that lie inside the flow domain (where $i+\frac{1}{2},j,k$ refers to the face common to cells $\Delta_{i,j,k}$ and $\Delta_{i+1,j,k}$.) We note that the geometry specification is typically sparse with only a small fraction of the computational cells being mixed. Sparse data structures are used that compress this information so that seven additional full arrays are not required.

The basic algorithm is, essentially, a two-step process. In the first step we compute fluxes for U "ignoring" the presence of the embedded front. To perform this step, it is necessary to extrapolate states outside the domain within one cell of a mixed cell. Using these extrapolated values, whose exact specification is given below, we define states U^{ext}_{ijk} that extend the definitions of U to a slightly larger domain

In the first step of the algorithm, we use an unsplit second-order Godunov integration algorithm developed by Colella [9] to compute fluxes for U. This scheme has the form

$$U^{n+1}_{i,j} = U^n_{i,j} + \frac{\Delta t}{\Delta x}(F^x_{i-\frac{1}{2},j,k} - F^x_{i+\frac{1}{2},j,k}) +$$

$$\frac{\Delta t}{\Delta y}(F^y_{i,j-\frac{1}{2},k} - F^y_{i,j+\frac{1}{2},k}) + \frac{\Delta t}{\Delta z}(F^z_{i,j,k-\frac{1}{2}} - F^z_{i,j,k+\frac{1}{2}}) \qquad (2.2)$$

Here, $F^x_{i+\frac{1}{2},j,k}$, $F^y_{i,j+\frac{1}{2},k}$, $F^z_{i,j,k+\frac{1}{2}}$ approximate time averaged fluxes at the cell edges, and are assumed to be explicit functions of U^n of the form

$$F^x_{i+\frac{1}{2},j,k} = F^x(U^n_{i,j-1,k-1},\ldots,U^n_{i+1,j+1,k+1};\ (D^-_\alpha U)_{i-r,j-r,k-r},\ \ldots,(D^-_\alpha U)_{i+r,j+r,k+r})$$

$$F^y_{i,j+\frac{1}{2},k} = F^y(U^n_{i-1,j,k-1},\ldots,U^n_{i+1,j+1,k+1};\ (D^-_\alpha U)_{i-r,j-r,k-r},\ \ldots,(D^-_\alpha U)_{i+r,j+r,k+r})$$

$$F^z_{i,j,k+\frac{1}{2}} = F^z(U^n_{i-1,j-1,k},\ldots,U^n_{i+1,j+1,k+1};\ (D^-_\alpha U)_{i-r,j-r,k-r},\ \ldots,(D^-_\alpha U)_{i+r,j+r,k+r})$$

where D^-_α represents the three one-sided differences at

each point, namely,

$$(D_x^- U)_{i,j,k} = U^n_{i,j,k} - U^n_{i-1,j,k},$$
$$(D_y^- U)_{i,j,k} = U^n_{i,j,k} - U^n_{i,j-1,k},$$
$$(D_z^- U)_{i,j,k} = U^n_{i,j,k} - U^n_{i,j,k-1},$$

In other words, $F^x_{i+\frac{1}{2},j,k}$, $F^y_{i,j+\frac{1}{2},k}$ and $F^z_{i,j,k+\frac{1}{2}}$ depend on the values of U^n in the 18 cells nearest the cell edge where the flux is defined, plus a possible dependence on values of U^n farther away which appear only as one sided differences in U^n. In addition, the scheme has the property that setting any of the $D_x^- U$, $D_y^- U$ or $D_z^- U$ to zero adds dissipation to the scheme and when they are all set to zero the resulting scheme is a first-order version of Godunov's method that has corner coupling so that it is stable for CFL number's up to 1.0. The only modification made to the integration module is to zero $D_x^- U$, $D_y^- U$ and $D_z^- U$ when these differences involve cells that are entirely outside the fluid; i.e., difference contributions for computing fluxes that attempt to use values in cells where $\Lambda_{ijk} = 0$ are set to zero.

The second step of the algorithm uses the fluxes computed in the first step and the geometric specification to update cells near the embedded boundary. This procedure is based on ideas developed by Chern and Colella. The fluxes are combined with apertures that specify the area (in space-time along each edge) where that flux is applicable to compute a conservative update for U for each cell through with the front passes during a time step. Since cells intersected by the boundary at the new time have a reduced volume, fully updating the cell would lead to a violation of CFL conditions and would potentially lead to an instability. Thus, each cell receives a fraction of its specified conservative update that is consistent with stability. To preserve conservation the remainder of the update is redistributed to neighboring cells in a mass-weighted manner.

The algorithm as described so far requires that topologically separate regions of the fluid (for instance, in the case where a baffle plate in embedded in the flow) be separated by at least two computational cells that are entirely contained in the body. This requirement is due to the construction of the extended states. Because the one-sided differences $D_x^- U$, $D_y^- U$ and $D_z^- U$ are set to zero if any body cells are involved, the extended states enter the algorithm only in the solution of the Riemann problem. If necessary, then, we can overcome the above limitation by forcing 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. We call this approach the "thin-wall" approximation.

Even if the thin-wall approximation is employed, the algorithm still requires that truly separate regions of the flow be separated by at least one body cell. This requirement is due to the redistribution algorithm. Regardless of whether the thin-wall approximation is used or not, the cell-separation requirement needs to be met only at the finest level of calculation when the integration algorithm is coupled with AMR.

Integration Scheme with Embedded Boundaries

In this section we describe the integration scheme with tracking in more detail. We denote the grid cells by Δ_{ijk} where $i=i_{lo}, \ldots, i_{hi}$, $j=j_{lo}, \ldots, j_{hi}$. and $k=k_{lo}, \ldots, k_{hi}$. Before beginning the flux computation we must first define the extended states. (This step is not employed if the thin-wall approximation is used.) The extension is done in a two step process. First, we identify which cells require an extended state for the flux computation. The form of the difference scheme as specified by (2.2) indicates that cells where a value is required are those within the 18 nearest neighbors of edges that border mixed cells but which are outside the fluid domain, Ω. For later reference, we define a marker, P_{ijk} such that $P_{ijk}=1$ for cells that require an extended state and 0 otherwise. We then define an extended state

$$\overline{U}_{ijk} = \left[\frac{\sum_{nbh(i,j,k)} \Lambda^n U^{n,l}}{\sum_{nbh(i,j)} \Lambda^n} \right] \qquad (3.1)$$

for each cell where $P_{ijk}=1$, where $nbh(i,j,k)$ represents the 26 neighboring cells. We are guaranteed that at least one cell in $nbh(i,j,k)$ has nonzero Λ_{ijk} so that (3.1) is defined.

In the first step of the algorithm, we use the extended states defined above or the thin-wall approximation to compute fluxes for the edges of cells using the second-order Godunov procedure described in the previous section. The use of extended states or of the thin-wall approximation and the modification to the integration algorithm allows fluxes $F^x_{i+\frac{1}{2},j,k}$, $F^y_{i,j+\frac{1}{2},k}$ and $F^z_{i,j,k+\frac{1}{2}}$ to be defined for all edges of cell where $\Lambda_{ijk}>0$ or where $\Lambda_{ijk} = 0$ and $P_{ijk} = 1$.

The next step in the integration algorithm uses the fluxes computed in the first step in a volume-of-fluid update for the solution. Although we will not make any distinction at this point in the discussion, for cells entirely within the flow domain the algorithm discussed below collapses to the update formula (2.2) using appropriate fluxes and for cells that are sufficiently far from the embedded boundary the scheme reduces to the full second-order Godunov algorithm. For mixed cells, we recall that Λ_{ijk} represents the fraction of the cell volume inside the flow domain and the apertures $A_{i+\frac{1}{2},j,k}$, $A_{i,j+\frac{1}{2},k}$ and $A_{i,j,k+\frac{1}{2}}$ represent the fraction of the area of their respective cell-faces contained within the problem domain. Using this notation, we apply the divergence theorem to each cell to obtain an initial approximation U^*_{ijk} to U at t^{n+1}

$$\Delta x\,\Delta y\,\Delta z\,\Lambda_{ijk}U^{*}_{ijk} = \Delta x\,\Delta y\,\Delta z\,\Lambda^{n}_{ij}U^{n,l}_{ij} -$$
$$\Delta t\,\Delta y\,\Delta z\,(A_{i+\frac{1}{2},j,k}F^{x}_{i+\frac{1}{2},j,k} - A^{l}_{i-\frac{1}{2},j,k}F^{x}_{i-\frac{1}{2},j,k})$$
$$+\Delta t\,\Delta x\,\Delta z\,(A_{i,j+\frac{1}{2},k}F^{y}_{i,j+\frac{1}{2},k} - A_{i,j-\frac{1}{2},k}F^{y}_{i,j-\frac{1}{2},k})+$$
$$+\Delta t\,\Delta x\,\Delta y\,(A_{i,j,k+\frac{1}{2}}F^{y}_{i,j,k+\frac{1}{2}} - A_{i,j,k-\frac{1}{2}}F^{y}_{i,j,k-\frac{1}{2}})-$$
$$\Delta t A^{f}_{ijk}F^{f}_{ijk}) \qquad (3.2)$$

where the additional aperture A^{f}_{ijk} is the surface area of the intersection of the embedded boundary with cell Δ_{ijk} and the additional flux F^{f}_{ijk} is the flux across that surface. The frontal flux is computed by solving a suitable Riemann problem at the boundary with data taken from U^{n}_{ijk}. The first step in this process is to determine an effective normal to the boundary. This is accomplished by integration of the gradient of a constant over the portion of the cell inside the domain and applying the divergence theorem. For any constant ϕ we have

$$0 = \iiint_{\Delta_{ijk}\cap\Omega} \nabla\phi\,dxdydz$$
$$= \int_{\partial(\Delta_{ijk}\cap\Omega)} \phi\,n dS \qquad (3.3)$$

If we take ϕ=1 in (3.3) we can specify A^{f}_{ijk} and n_f, the normal to the front, in terms of the other apertures using

$$A^{n}_{ijk}n_f = \Delta y\,\Delta z\,(A_{i-\frac{1}{2},j,k} - A_{i+\frac{1}{2},j,k})\mathbf{i}+$$
$$\Delta x\,\Delta z\,(A_{i,j-\frac{1}{2},k} - A_{i,j+\frac{1}{2},k})\mathbf{j}+$$
$$\Delta x\,\Delta y\,(A_{i,j,k-\frac{1}{2}} - A_{i,j,k+\frac{1}{2}}\mathbf{k}) \qquad (3.4)$$

For the typical case of a reflecting wall we can then reflect U^{n}_{ijk} using an odd reflection of the normal velocity (relative to the normal n_f) and an even reflection of the other quantities to obtain values of U on each side of the interface. Solution of the Riemann problem using these states then gives the value of F^{f}_{ijk}. Other types of boundary conditions can be treated in an analogous manner.

We want to now reexpress the update of U (3.2) in terms of the change in U caused by the embedded boundary. To accomplish this we first define an update that ignores the front

$$U^{ext,n+1}_{i,j,k} = U^{ext,n}_{i,j,k} + \frac{\Delta t}{\Delta x}(F^{x}_{i-\frac{1}{2},j,k} - F^{x}_{i+\frac{1}{2},j,k})$$
$$+ \frac{\Delta t}{\Delta y}(F^{y}_{i,j-\frac{1}{2},k} - F^{y}_{i,j+\frac{1}{2},k})$$
$$+ \frac{\Delta t}{\Delta z}(F^{z}_{i,j,k-\frac{1}{2}} - F^{z}_{i,j,k+\frac{1}{2}}) \qquad (3.5)$$

where the fluxes are computed from the extended states $U^{ext,n}$. Then we rewrite (3.5) as

$$\Lambda_{ijk}U^{*}_{ijk} = \Lambda_{ijk}U^{ext,n+1}_{ijk} + \delta M_{ijk} \qquad (3.6)$$

where

$$\delta M_{ijk} = \Lambda^{n}_{ijk}U^{n}_{ijk} - \Lambda^{n}_{ijk}U^{ext,n+1}_{ijk} -$$
$$\frac{1}{\Delta x\,\Delta y\,\Delta z}(\Delta t\,\Delta y\,\Delta z\,[A_{i+\frac{1}{2},j,k}F^{x}_{i+\frac{1}{2},j,k} - A_{i-\frac{1}{2},j,k}F^{x}_{i-\frac{1}{2},j,k}]+$$
$$\Delta t\,\Delta x\,\Delta z\,[A_{i,j+\frac{1}{2},k}F^{y}_{i,j+\frac{1}{2},k} - A_{i,j-\frac{1}{2},k}F^{y}_{i,j-\frac{1}{2},k}]+$$
$$\Delta t\,\Delta x\,\Delta y\,[A_{i,j,k+\frac{1}{2}}F^{z}_{i,j,k+\frac{1}{2}} - A_{i,j,k-\frac{1}{2}}F^{z}_{i,j,k-\frac{1}{2}}]$$
$$-\Delta t A^{f}_{ijk}F^{f}_{ijk}) \qquad (3.7)$$

We note that with this definition of δM, sufficiently far away from the embedded boundary $\delta M = 0$ to that (3.7) is simply a re-expression of (2.2). If we divide (3.6) by Λ^{n+1}_{ijk} we would have a conservative update for U_{ijk}, namely,

$$U^{n+1}_{ijk} = U^{ext,n+1}_{ijk} + \frac{\delta M_{ijk}}{\Lambda_{ijk}} .$$

However, because the Λ's can be arbitrarily small, this update can require an excessive time step restriction to remain stable. (This update is, of course, stable for cells where Λ=1.) To avoid this type of time-step limitation we use the redistribution ideas of Chern and Colella [6]. Their basic idea is to define a preliminary update that adds a fraction of the update that will be stable at the time step determined by the CFL criteria on the regular grid. This gives a preliminary evolution in time

$$U^{*}_{i,j,k} = U^{n}_{i,j,k} + \delta M_{i,j,k} \qquad (3.8)$$

This update does not preserve discrete conservation form. In order to have conservation, we must distribute $(1-\Lambda_{ijk})\delta M_{ijk}$ onto the grid. In the general case we do this by decomposing these increments into characteristic variables and distributing them to nearby cells in a volume-weighted fashion. That more general procedure is discussed in the case of front tracking in Bell et al [7]. For the case of a reflecting wall, the redistribution procedure simplifies by not requiring an upwind, characteristic-based approach; simply redistributing all of $(1-\Lambda_{ijk})\delta M_{ijk}$ into the interior of the flow domain is sufficient. For this simpler case we define

$$\delta M^{red}_{ijk} = (1-\Lambda_{ijk})\delta M_{ijk}$$
$$m^{red}_{ijk} = \sum_{nbh(i,j,k)} \rho^{n+1}_{ijk}\Lambda_{ijk} \qquad (3.9)$$

Then we define the final values of U^{n+1}_{ijk} to be

$$U^{n+1}_{ijk} = U^{*}_{ij} + \sum_{nbh(i,j,k)} \frac{\delta M^{red}_{ijk}}{m^{red}_{ijk}}. \qquad (3.10)$$

We note the scaling the volumes by ρ^{n+1}_{ijk} in (3.9) causes the redistribution to be mass weighted in contrast to the volume weighting used by Bell et al for shock tracking. The use of mass weighting was found to be somewhat superior in the embedded boundary case; however, for a more general system of hyperbolic equations it may not be a meaningful choice. In fact, other possible choices for the weighting can be used such as total energy weighting or the original volume weighting.

The restriction of the embedded boundary integration technique described above to a Cartesian grid 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,

$$\frac{\partial U}{\partial t} + \frac{\partial A_r F}{\partial V_r} + \frac{\partial p}{\partial r} + \frac{\partial G}{\partial z} = 0 \qquad (3.11)$$

where $A_r = r$ 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 is free-stream preserving, i.e., no pressure gradients are generated in a uniform flow. The issues associated with generalized volume coordinates are discussed by Colella [10].

The first modification to the r-z Cartesian algorithm is the specification of the geometry. The volume fractions and apertures are determined using the actual volume measure $r\,dr\,\Delta z$;thus, for example,

$$\Lambda_{ijk} = \frac{\iint_{\Omega \cap \Delta_{ij}} r\,dr\,\Delta z}{\iint_{\Delta_{ij}} r\,dr\,\Delta z}$$

In addition, we also need the Cartesian volume fraction

$$\Lambda^C_{ij} = \frac{\iint_{\Omega \cap \Delta_{ij}} dr\,\Delta z}{\iint_{\Delta_{ij}} dr\,\Delta z}$$

The metric terms introduce two changes in the basic integration scheme. The determination of the normal to the boundary and the frontal aperture involve additional terms. As before, we derive the formula by integrating the gradient of a constant ϕ over the domain. This gives

$$\begin{aligned} 0 &= \iint_{\Delta_{ij} \cap \Omega} \nabla\phi r\,dr dz \\ &= \int_{\partial(\Delta_{ij} \int \Omega)} \phi\, n dS - \int_{\Delta_{ij} \int \Omega} \phi\, dr dz. \end{aligned} \qquad (3.12)$$

If we take ϕ=1 in (3.12) in this case we obtain

$$A^f_{ij} n_f = \Delta z\,(A_{i-\frac{1}{2},j} - A_{i+\frac{1}{2},j}) + \Delta r\,(A_{ij-\frac{1}{2}} - A_{ij+\frac{1}{2}}) + \Lambda^C_{ij}$$

where i,j are the indices in the r,z directions respectively. A final modification is needed for the computation of δM_{ij} Direct differencing of the pressure gradient in the r directions is not appropriate for mixed cells. For a mixed cell we rewrite

$$\frac{\partial p}{\partial r} = \frac{\partial A_r p}{\partial V_r} - p\,\frac{\partial A_r}{\partial V_r} = 0$$

The right hand side is now used for the finite volume discretization of $\partial p/\partial r$ in the computation of δM_{ij}. In particular,

$$\frac{\partial p}{\partial r}_{ij} = A^r_{i+\frac{1}{2},j} p_{i+\frac{1}{2},j} - A^r_{i+\frac{1}{2},j} p_{i+\frac{1}{2},j} - \Lambda^C_{ij} \frac{p_{i+\frac{1}{2},j} + p_{i-\frac{1}{2},j}}{2} + p^f_{ij} A^f_{ij} n^f_{ij,r}$$

where $p_{i\pm\frac{1}{2},j}$ are the edges fluxes computed by the Godunov module and p_f, A^f and n^f_r are the pressure, aperture, and r component of the normal at the boundary. This form maintains the free-stream preservation property of the basic integration scheme in the embedded boundary case.

Coupling to AMR

In this section we describe how the Cartesian grid methodology is coupled to a local adaptive mesh refinement algorithm described by Bell et al [8]. We will briefly review the basic adaptive refinement algorithm before describing the modifications needed to incorporate the embedded boundary. The algorithm is based on a hierarchical grid structure composed of grids of varying resolution. The grid hierarchy is constructed using an error estimation procedure to identify cells having unacceptable errors that are then clustered into logically-rectangular grids that are subdivided to form finer cells where more resolution is required. Integration of the differential equations on this hierarchical grid structure is a three-step procedure. First, the coarse grid is integrated to supply boundary data for finer grids. The fine grids are then integrated, subcycling in time, to catch up to the coarse grid. Finally, the coarse grids are corrected to reflect the improved resolution of the finer grids.

There are three modifications that are required to couple the Cartesian grid algorithm to adaptive refinement. First, in order to maintain consistency of the representation of the geometry on different levels of refinement, the volume fractions and the area fractions are calculated on the finest level of resolution and then averaged down to the coarser levels in a volume weighted or area weighted fashion. Second, interpolation from coarse grids to finer grids, which occurs in interpolating boundary conditions for the fine grids and in initializing fine grids from coarse grids when the error estimation criteria call for finer grids in a particular region, is modified in a fashion similar to the calculation of one-sided differences in the integration algorithm; that is, any one-sided slopes used in the conservation interpolation calculation are set to zero if any of the cells involved fall entirely out of the fluid domain. Similarly, in averaging fine grid cells to define values on an underlying coarse grid cell, the averaging must be weighted by the volume fractions in the fine cells to ensure conservation. The compatibility between levels is then guaranteed because the geometry of the coarse

grid is defined as the average of the fine grid geometry.

The third modification represents a more substantial change. Without the embedded boundary, the correction to the coarse grid to reflect the improved resolution of the fine grid is to replace underlying coarse grid values with the average of the covering fine grid values and to add a flux correction δF to coarse grid cells that border fine grids. More precisely, we set

$$U^c := U^c + \frac{\delta F}{V^c} \tag{4.1}$$

where

$$\delta F = \sum (A^f \Delta t^f F^f) - A^c \Delta t^c F^c$$

with the sum taken over the fine grid edges that cover the coarse edge and over the number of time steps the fine grid is subcycled for a coarse step. Here, V^c is the volume of the coarse cell, and $A^{c,f}$ are the areas of the coarse cell faces and the fine cell faces that cover it. Note that δF is an extensive quantity, e.g. mass not density. The update (4.1) 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.

When embedded boundaries are included, additional modifications are needed to account for the effects of the redistribution step of the algorithm. These corrections arise because redistribution provides an additional mechanism for communication across a coarse-fine boundary. There are four basic coarse-fine redistribution terms:

δR_f^I: These are the values redistributed into the fine grid from the grid boundary cells; hence, there are artificial and their effect must be removed.

δR_c^I: These are the values redistributed from the coarse grid into the coarse grid cells underlying the fine grid that are subsequently lost when the coarse values is redefined by averaging the fine values.

δR_f^E: These are the values redistributed from the fine grid into its boundary cell. These values are then lost.

δR_c^E: These denote the redistribution values from the coarse grid underlying the fine grid to the coarse grid cells on the boundary of the fine grid. Their effect should be removed.

We now define

$$\delta R^I = \delta R_c^I - \sum \delta R_f^I \tag{4.2}$$

and

$$\delta R^E = \sum \delta R_f^E - \delta R_c^E \tag{4.3}$$

These terms, which are accumulated in extensive form, represent the values that should be added to the coarse interior cells on the boundary of the fine grid (and the fine cells that cover them) and the correction to be added to the coarse grid exterior to the find grid. We associate δR^I with the coarse grid cell from which the values came and δR^E with the coarse grid cell that received them. We note that the δR's and the δF's must, in general, be accumulated on both the coarse and the fine grids and that the fluxes appearing in the definition of δF are aperture weighted. These terms are then combined to form

$$\delta M_{c-f} = \frac{1}{V^c}(\delta F + \delta R^E + \delta R^I) \tag{4.4}$$

which is a generalized reflux correction to coarse grid cells that border fine grids. As we did in the main integration step, we include a stable portion of the update in each cell and redistribute the remainder to its neighbors. Thus,

$$U^{n+1} := U^{n+1} + \delta M_{c-f}$$

and

$$\delta M_{c-f}^{red} = (1-\Lambda)\delta M_{c-f} \tag{4.5}$$

The δM_{c-f}^{red} are then redistributed to the neighboring coarse cells using the procedure defined in the previous section. Values redistributed during this procedure to coarse grids that are covered by fine grid cells are lifted to the fine grid, weighted by the fine grid Λ's. We note that for interfaces between coarse and fine grids that are not near a portion of the embedded boundary, the δR's in (4.4) vanish and the reflux correction reduces to (4.1).

Numerical Example

In this section we present several numerical examples showing the combined Cartesian grid / adaptive mesh refinement algorithm.

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 degrees. Figure 1 shows a contour plot of the density at late time for a uniform 160x80 grid; figure 2 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 three calculations on 80x40, 160x80, and 320x160 grids for 250, 500, and 1000 time steps, respectively. Two measures of the error in the solution at the final time step were used, an area weighted relative error for the entire problem domain and a length weighted relative error for the solution along the fluid/body interface. The two error measures are tabulated below for two quantities, log-entropy and stagnation enthalpy, both of which are constants for the flow under consideration:

grid	log-entropy		stag. enthalpy	
	error	wall error	error	wall error
80x40	.00154	.0233	2.22e-4	.00314
160x80	.00042	.0108	8.05e-5	.00193
320x160	.00010	.0049	2.70e-5	.00124

These results suggest that the algorithm is second order accurate away from the fluid/body interface and first order accurate at the interface.

The second example shows a Mach 10 flow past a 30 degree ramp. Figure 3 shows density contours of the solution obtained using the Cartesian grid algorithm. For comparison, the solution obtained using the second-order Godunov method described in Colella [9] with the adaptive mesh refinement scheme described in Bell et. al. [8]. is displayed in Figure 4. Each calculation uses the same size coarse level cells and the same refinement ratios in building two levels of successively finer cells. Hence, the effective resolution of the two calculations is the same at all levels of refinement. The grids at the different levels of refinement are shown as boxes. We observe that the results of the Cartesian grid calculation compare favorably with the results of the other calculation. (We note here that the jaggedness of the fluid/body interface in grids at the coarser levels of refinement in figure 3 and in later figures is a plotting artifact.)

The third example (Figure 4) shows a Mach 1.597 flow past a cone with a semi-apex angle of 9.5 degrees. This flow is calculated using the r-z formulation of the Cartesian grid algorithm. The results of the calculation for the most part compare favorably with experimental results [11]. However, the computed flow in the wake region is not as well developed as the wake flow observed experimentally since the former is calculated in a purely axially symmetric fashion.

The last example shows a Mach 2.33 flow past a cone/cylinder with a semi-apex angle of 8.58 degrees. Figure 6 shows the density at late time of four longitudinal slices of the flow. Figure 7 shows a rendering of the density at the same time in the entire flow region and in the wake region. The calculated results show good agreement with experimental results using the same configuration

References

1. A. Jameson, T. J. Baker, and N. P. Weatherhill, "Calculation of inviscid transonic flow over a complete aircraft," AIAA 86-0102, 1986.
2. R. Lohner, P. Parikh, and M. Merriam, "Parallel unstructure grid generation," in *Proceedings of the 10th AIAA Computational Fluid Dynamics Conference*, AIAA, Honolulu, 1991.
3. B. Wedan and J. South, "A method for solving the transonic full-potential equations for general configurations," in *Proceedings of the 6th AIAA Computational Fluid Dynamics Conference*, AIAA, Danvers, Mass., 1983.
4. M. J. Berger and R. J. LeVeque, "An adaptive Cartesian mesh algorithm for the Euler equations in aribtrary geometries," in *Proceedings of the 9th AIAA Computational Fluid Dynamics Conference*, AIAA, Buffalo, 1989.
5. D. D. Zeeuw and K. G. Powell, "An adaptively-refined Cartesian mesh solver for the Euler equations," in *Proceedings of the 10th AIAA Computational Fluid Dynamics Conference*, AIAA, Honolulu, 1991.
6. I.-L. Chern and P. Colella, "A Conservative Front Tracking Method for Hyperbolic Conservations Laws," UCRL-97200, LLNL, July 1987.
7. J. B. Bell, P. Colella, and M. Welcome, "Conservative Front-Tracking for Inviscid Compressible Flow," in *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference*, AIAA, Honolulu, 1991.
8. J. B. Bell, M. J. Berger, J. S. Saltzman, and M. Welcome, "Three Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws," UCRL-JC-108794, LLNL, December 1991. submitted to J. Comput. Phys.
9. P. Colella, "A Multidimensional Second Order Godunov Scheme for Conservation Laws," *J. Comp. Phys.*, vol. 87, pp. 171-200, 1990.
10. P. Colella, "A Direct Eulerian MUSCL Scheme for Gas Dynamics," *SIAM J. on Scientific and Statistical Computing*, vol. 6, pp. 104-117, January 1985.
11. Z. Kopal, *Tables of supersonic flow around cones*, MIT, Cambridge, Mass., 1947.
12. A. H. Shapiro, *The dynamics and thermodynamics of compressible fluid flow*, p. Ronald Press Co., New York, 1954.

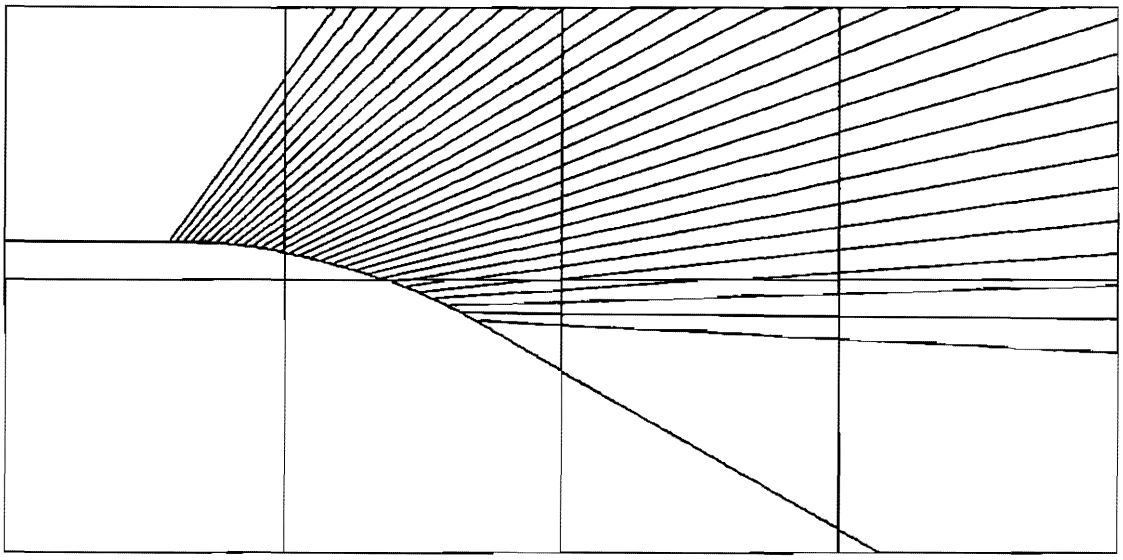

FIG. 1. Density contours from the computation of a Prandtl-Meyer expansion wave resulting from a Mach 1.2 flow turning through an angle of 30 degress.

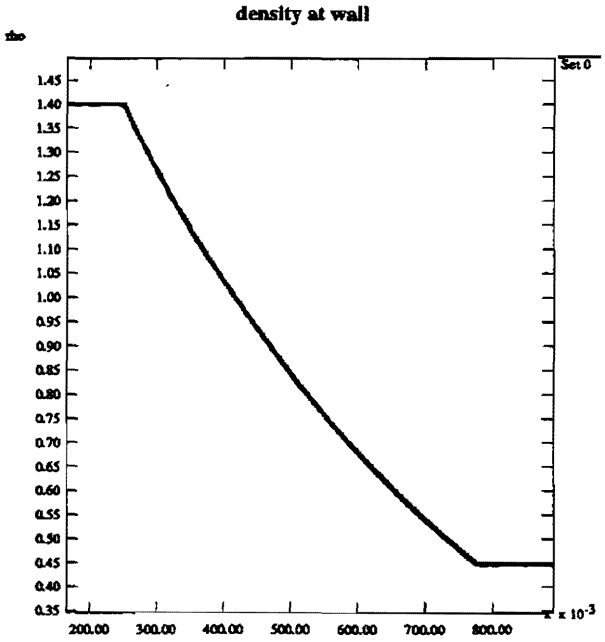

FIG. 2. Density profile along the fluid/body interface for the computed flow displayed in figure 1.

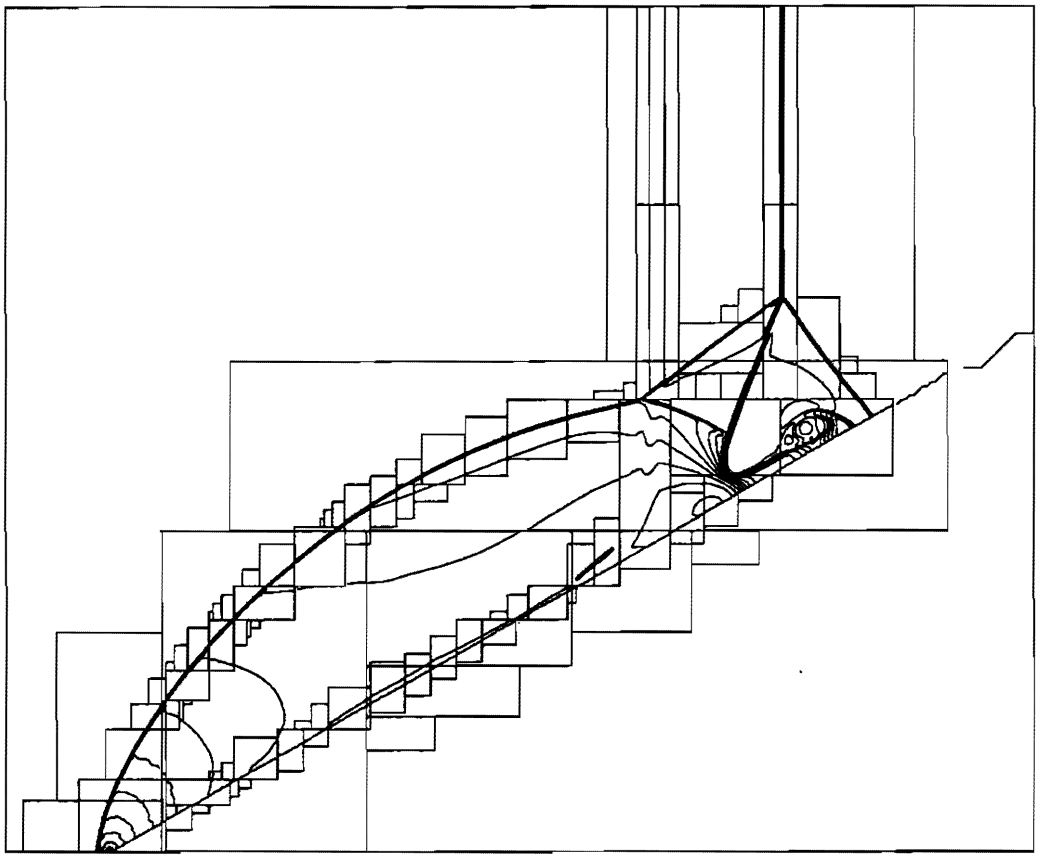

FIG. 3. Density contours from a computation of a Mach 10 flow past a 30 degree ramp.

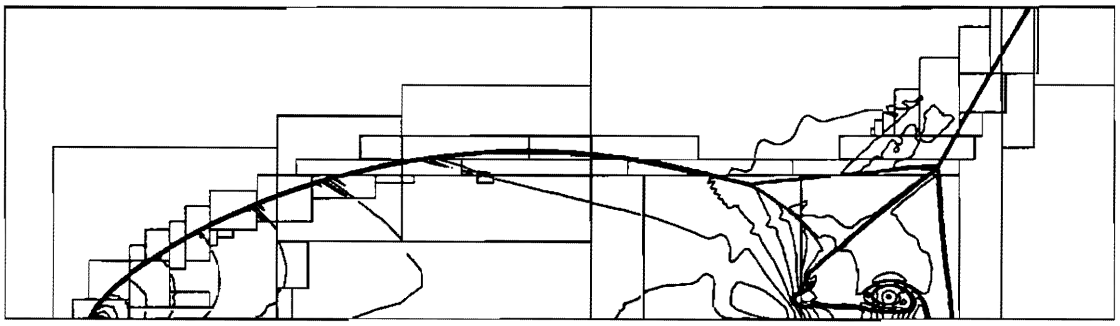

FIG. 4. Density contours from a higher order Godunov/AMR computation for the flow displayed in figure 3.

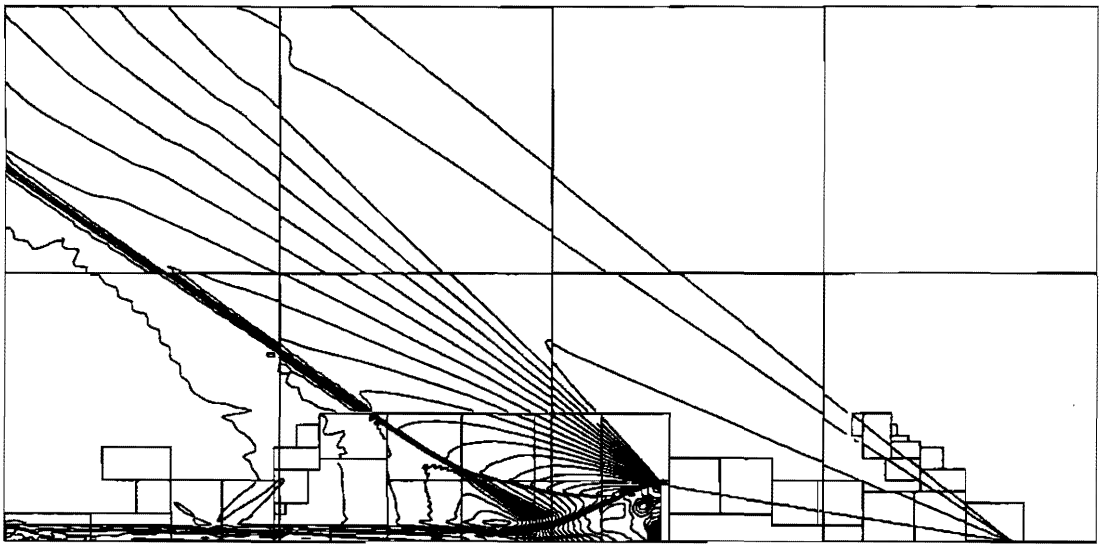

FIG. 5. Density contours from an r-z compuation of a Mach 1.597 flow past a cone with a semi-apex angle of 9.5 degrees.

Figure 6. Longitudinal slices of density field for Mach 2.33 flow past a cone-cylinder, with a semi-apex angle of 8.58 degrees.

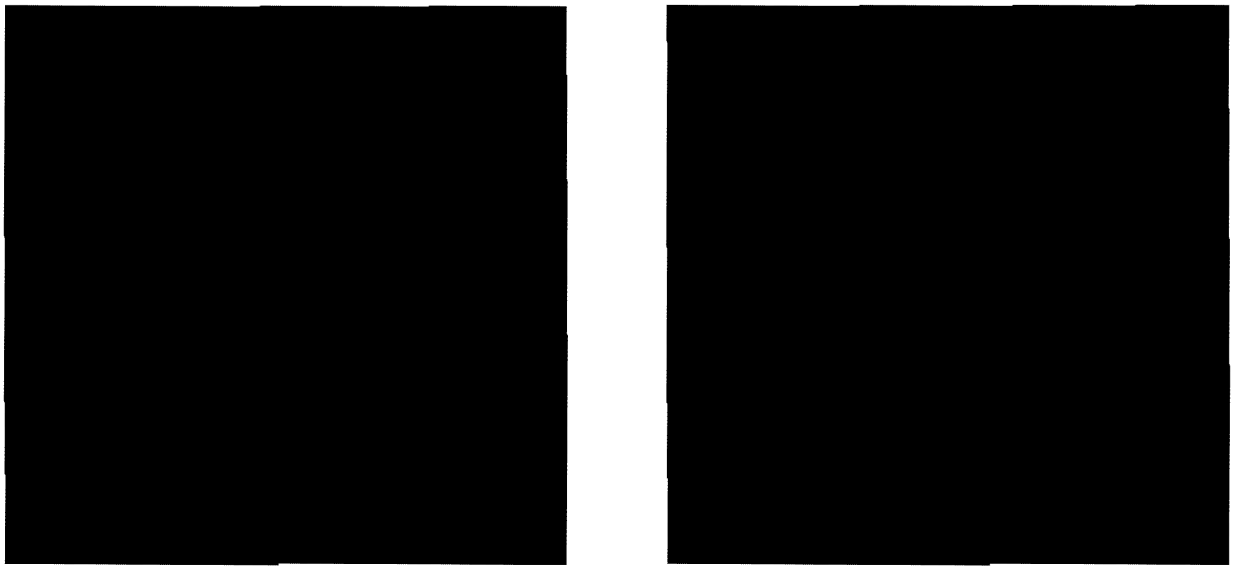

Figure 7. Rendering of density field for Mach 2.33 flow past a cone-cylinder, with a semi-apex angle of 8.58 degrees. Full and wake views.