INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **33**: 1121–1151

Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut cell method

G. Yang, D. M. Causon* and D. M. Ingram

Centre for Mathematical Modelling and Flow Analysis, The Manchester Metropolitan University, Manchester, U.K.

SUMMARY

A three-dimensional Cartesian cut cell method is described for modelling compressible flows around complex geometries, which may be either static or in relative motion. A background Cartesian mesh is generated and any solid bodies cut out of it. Accurate representation of the geometry is achieved by employing different types of cut cell. A modified finite volume solver is used to deal with boundaries that are moving with respect to the stationary background mesh. The current flow solver is an unsplit MUSCL–Hancock method of the Godunov type, which is implemented in conjunction with a cell-merging technique to maintain numerical stability in the presence of arbitrarily small cut cells and to retain strict conservation at moving boundaries. The method is applied to some steady and unsteady compressible flows involving both static and moving bodies in three dimensions. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: compressible flow; Cartesian cut cells; finite volume; Riemann-based schemes; static and moving bodies

1. INTRODUCTION

Although a variety of mesh generation techniques are now available (see, for example, References [1–7]), the generation of meshes around complicated, multi-component geometries is still a tedious and difficult task. Currently, the two most widely used methods for dealing with complex geometries involve the use of either structured or unstructured meshes. Structured mesh techniques are usually based on either multi-block [8], composite overlapping block structure approaches (Chimera) [9] or flexible mesh embedding techniques (e.g. FAME) [10,11]. Essentially, with an overlapping mesh technique, individual components are meshed separately using local structured meshes that overlay a background structured mesh. As any boundary/body moves its associated mesh moves with it, with the other meshes remaining

* Correspondence to: Centre for Mathematical Modelling and Flow Analysis, The Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, U.K.

Received March 1998
Revised August 1999

unchanged. Since information must be transferred between meshes, the identification of mesh intersection points and interpolation of data on all overlapping meshes is necessary. Furthermore, if an individual body geometry is very complex, the essential difficulties of mesh generation remain unresolved unless multiple embedded meshes are used on a single component, as in the FAME approach [10,11]. Unstructured mesh methods [2–7,12,13] utilize grids in which there is no concept of global or local co-ordinate directions. Triangular cells are used in two dimensions and tetrahedra in three dimensions; hence unstructured mesh techniques appear extremely powerful for arbitrarily complex geometries. For moving boundary/body problems, periodic local or global remeshing is required to account for the boundary/body movement. This is non-trivial and requires a significant degree of user intervention. In cases where body motion is large, such methods may encounter difficulties preserving boundaries and avoiding excessive mesh distortion and/or overlapping elements.

A third alternative regaining popularity is the use of a completely Cartesian mesh. Conceptually, this approach is quite simple. Solid bodies are cut out of a single static background mesh and their boundaries represented by different types of cut cell. The difficult and case-specific problem of generating a structured or unstructured mesh is replaced by the more general problem of finding the intersection points between a stationary or moving surface geometry and a background Cartesian mesh. In principle, the cut cell approach has the potential to greatly simplify and automate the difficult task of mesh generation. Also it can provide a method that can deal efficiently with steady or non-steady compressible flows around arbitrarily complicated, multi-element, geometries, which are either stationary or in relative motion, without the requirement for a moving mesh or local or global remeshing. Problems such as mesh distortion, body motion restriction, etc., which may occur when using other mesh approaches, are completely avoided. The computational overheads of the cut cell approach in terms of the required checks on critical cells, recalculation of cell volumes and side vectors as bodies move through the static mesh are typically in the order of 5%. Compared with a conventional body-fitted curvilinear structured grid, cell volumes and side vector quantities are trivial to evaluate and calculated quickly. On the other hand, a fully unstructured method would require local remeshing in the vicinity of a moving body together with interpolation of the solution data, which would be more costly in general than a cut cell approach, which only requires changes at cells cut by the body contour.

Cartesian mesh methods have been used with significant success for computing flows about complicated geometries (see References [14–25]). For example, Berger and Colella [22] and Quirk [23] have used adaptive mesh refinement (AMR) techniques to obtain highly resolved solutions of unsteady shock hydrodynamic problems. Upwind methods on adaptively refined Cartesian meshes have been described previously by Berger and Colella [22] for unsteady flows. Powell and colleagues [17,18,20] have also developed Cartesian mesh methods for steady and unsteady flows, employing a time step adaptation approach to improve computational efficiency. An assessment of the accuracy of Cartesian mesh approaches has been made by Coirier and Powell [20]. These results have demonstrated the value of cut cell methods for static bodies; however, it is for moving boundary problems that the method potentially offers most advantage when compared with more conventional mesh generation techniques. This is particularly so in cases where large amplitude body motion is involved. In this paper, we present an extension of the two-dimensional Cartesian cut cell method reported in References

[21,26] to three dimensions, including details of the algorithm for both static and moving body problems. In the next section, we first describe the generation of a Cartesian cut cell mesh in three dimensions and in Section 3 we present the details of a high resolution numerical integration scheme suitable for use on a cut cell mesh. Extension of the methodology to moving body problems is discussed in Section 4, while in Section 5 examples are provided to show the capability of the method for the simulation of compressible flows around complex static geometries and moving body problems. We include preliminary results for a case involving a body engaged in large amplitude motion through a static mesh. Finally, in Section 6, we make some concluding remarks and comments concerning future work.

2. PRINCIPLES OF THE CUT CELL METHOD IN THREE-DIMENSIONAL CASE

A Cartesian cut cell mesh can be generated simply by 'cutting' solid bodies out of a background Cartesian mesh. Three types of cells are formed in the computational domain: flow cells, cut cells and solid cells (see Figure 1). Each cut cell can be handled by replacing any number of intersecting planes within the cell by seven approximating planes, including six interfaces and one solid face.

Initially, all cells are flagged as either flow or solid cells. Once all the intersections between the boundaries of any solid bodies and the background mesh lines have been established, cells that intersect the surfaces of solid bodies are defined as cut cells. Sweeps across the background mesh are performed to identify which cells are surrounded by solid or cut cells. These cells are registered as solid cells.

2.1. Solid geometry description

Body surfaces can be described by triangulated surface facets obtained directly from a computer aided design (CAD) package or by a surface triangulation procedure. Unlike the surface triangulation used in unstructured mesh generation techniques, where the size of triangles are fixed by the requirements of the flow solver, the surface facets in our method are chosen to be just large enough to accurately define the surfaces of any bodies. For example, a planar surface of a solid body need only be divided into two triangular facets. The triangular facets are stored as a set of vertices with connectivity lists so that the surface normals are oriented properly in the flow domain. For example, the triangular facet M in Figure 2 has three vertices in a vertex set and its normal is

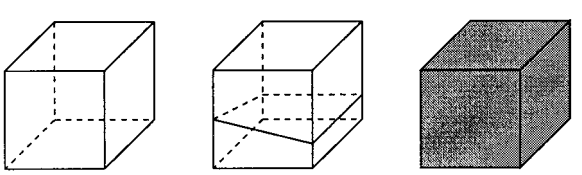

Figure 1. Flow cell, cut cell and solid cell.

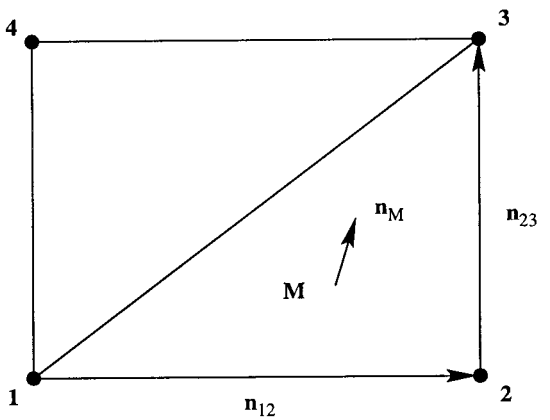

Figure 2. A triangular facet M.

$$\mathbf{n}_M = \mathbf{n}_{12} \times \mathbf{n}_{23} \tag{1}$$

$$\mathbf{n}_{12} = (X_2 - X_1,\ Y_2 - Y_1,\ Z_2 - Z_1) \tag{2}$$

$$\mathbf{n}_{23} = (X_3 - X_2,\ Y_3 - Y_2,\ Z_3 - Z_2) \tag{3}$$

If there is more than one solid body, additional sets of vertices are needed. Thus, there is no limitation to the number of bodies in a multi-component geometry. The triangular facets are used only to find the intersection points between the surfaces of solid bodies and the background Cartesian mesh. Therefore, the fewer triangular facets that are used the quicker the generation of the cut cell mesh will be.

2.2. Finding the intersection points

Once the background Cartesian mesh and the geometry of any solid bodies have been defined we must find the intersection points between them. Since the geometry of a solid body is represented by a surface fitting of triangular facets, the problem of mesh generation is reduced to finding the intersection points between an individual triangular facet and a Cartesian mesh line. The intersection of a straight line with a plane can be found in a straightforward manner. First, equations of both the plane and the straight line are described in parametric formulae. Unless these are parallel or coincident, solutions for the relevant parameters can be obtained exactly (see Figure 3). The equations of a triangular plane and a straight line are

$$\begin{cases} \mathbf{P}(u, v) = \mathbf{P}(0, 0) + \mathbf{r}_1 u + \mathbf{r}_2 v & (0 \le u + v \le 1) \\ \mathbf{P}(t) = \mathbf{P}(0) + \mathbf{r}t & (0 \le t \le 1) \end{cases} \tag{4}$$

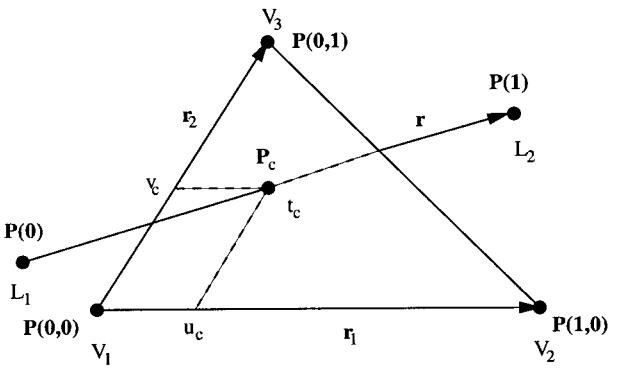

Figure 3. Intersection of line and triangular plane.

where

$$\begin{cases} \mathbf{r}_1 = (X_{V_2} - X_{V_1},\ Y_{V_2} - Y_{V_1},\ Z_{V_2} - Z_{V_1}) \\ \mathbf{r}_2 = (X_{V_3} - X_{V_1},\ Y_{V_3} - Y_{V_1},\ Z_{V_3} - Z_{V_1}) \\ \mathbf{r} = (X_{L_2} - X_{L_1},\ Y_{L_2} - Y_{L_1},\ Z_{L_2} - Z_{L_1}) \end{cases} \tag{5}$$

and

$$\begin{cases} \mathbf{P}(0) = (X_{L_1},\ Y_{L_1},\ Z_{L_1}) \\ \mathbf{P}(0,\ 0) = (X_{V_1},\ Y_{V_1},\ Z_{V_1}) \end{cases} \tag{6}$$

The system of equations that describe the point of intersection is

$$\mathbf{P}_c = \mathbf{P}(0,\ 0) + \mathbf{r}_1 u_c + \mathbf{r}_2 v_c = \mathbf{P}(0) + \mathbf{r} t_c \tag{7}$$

Here, the three unknowns, u_c, v_c and t_c, are uniquely determined by the three available equations in the respective co-ordinate directions. The solutions are then checked to determine whether the intersection point is on the triangular plane or its extended plane. If these solutions are out of range, the intersection lies on the extended plane (or line) of the triangular plane (or straight line). If the line is parallel to the plane or lies in the plane, there is no unique solution. Details of intersection problems involving a straight line and a flat plane can be found in Reference [27]. To find all the intersections on an individual Cartesian mesh line L, two end points L_1 and L_2 are usually chosen at two boundaries of the flow field. For example, if the mesh line L lies in the x-direction, the resulting points L_1 and L_2 will be

$$\begin{cases} \mathbf{P}(0) = (X_{\min},\ Y,\ Z) \\ \mathbf{P}(1) = (X_{\max},\ Y,\ Z) \end{cases} \tag{8}$$

Once all of the triangular facets have been checked, all intersection points along the mesh line L will have been found.

Usually, cut cells occupy only a small proportion of the total number cells in the flow domain. Hence, they are stored in a series of lists including connectivity data with respect to the background mesh. As each intersection point along an individual mesh line is obtained it is registered in the cut cell list. This process is repeated until all the mesh lines have been dealt with.

2.3. Determining the required cut cell data

Once the intersection points between the geometry of a solid body and the background mesh lines have been determined, all the required geometric data concerning the cut cells can be determined. For a finite volume method, this is the direction of the outward normal vector to each solid face, the area of each face and the volume of each cut cell. We calculate the area of six interfaces of each cut cell based on the intersection points at the cell edges. If a cut cell interface is located inside a solid, the area of the interface is set to zero. The solid boundary (or face) is approximated by the non-planar quadrilateral *pqrs* (see Figure 4). Although the normal vector and area of the solid face are not explicitly calculated in this case, they can be computed by taking the difference between the exposed areas of opposing interfaces, i.e.

$$|\mathbf{S}| = \sqrt{(S_x^{\mathrm{l}} - S_x^{\mathrm{r}})^2 + (S_y^{\mathrm{l}} - S_y^{\mathrm{r}})^2 + (S_z^{\mathrm{l}} - S_z^{\mathrm{r}})^2} \tag{9}$$

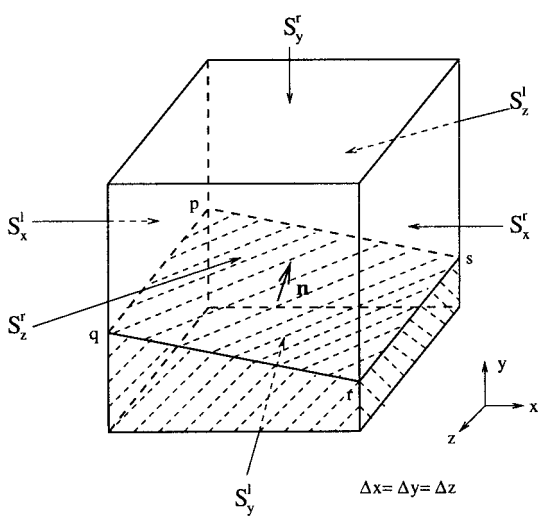

Figure 4. A cut cell in three dimensions.

$$\mathbf{n} = \frac{(S_x^{\mathrm{r}} - S_x^{\mathrm{l}},\, S_y^{\mathrm{r}} - S_y^{\mathrm{l}},\, S_z^{\mathrm{r}} - S_z^{\mathrm{l}})}{|\mathbf{S}|} \tag{10}$$

Finally, the volume is calculated using the Gauss divergence theorem, which transforms the required volume integrations into surface integrations over the exposed interfaces and solid faces, i.e.

$$V = \frac{1}{3} \sum_{m=1}^{M} \mathbf{P}_{\mathrm{l}} \cdot \mathbf{n}_{\mathrm{l}} S_{\mathrm{l}} \tag{11}$$

where $\mathbf{P}_{\mathrm{l}}$, $\mathbf{n}_{\mathrm{l}}$ and S_{l} are the position vector of the centroid, normal unit vector and area of an interface or solid face respectively, and M is the maximum number of cell interfaces.

3. THE FLOW CALCULATION SCHEME

3.1. Governing equations

The Euler equations for three-dimensional compressible flow in a general moving reference frame may be written in integral form as

$$\frac{\partial}{\partial t} \int_{V_t} \mathbf{U}\, \mathrm{d}V + \oint_{S_t} \mathbf{F} \cdot \mathbf{n}\, \mathrm{d}S = 0 \tag{12}$$

where $\mathbf{U}$ is the vector of conserved variables, $\mathbf{F}$ is the flux vector function and $\mathbf{n}$ is the outward unit vector normal to the boundary S_t, which encloses the time-dependent volume V_t. $\mathbf{U}$ and $\mathbf{F}$ are given by

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} \rho(\mathbf{v} - \mathbf{v}_{\mathrm{s}}) \\ \rho u(\mathbf{v} - \mathbf{v}_{\mathrm{s}}) + p\mathbf{i} \\ \rho v(\mathbf{v} - \mathbf{v}_{\mathrm{s}}) + p\mathbf{j} \\ \rho w(\mathbf{v} - \mathbf{v}_{\mathrm{s}}) + p\mathbf{k} \\ (e + p)(\mathbf{v} - \mathbf{v}_{\mathrm{s}}) + p\mathbf{v}_{\mathrm{s}} \end{bmatrix} \tag{13}$$

where ρ, u, v, w, p and e are density, the x-, y- and z-components of fluid velocity $\mathbf{v}$, pressure and total energy per unit volume; $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the Cartesian unit base vectors; and $\mathbf{v}_{\mathrm{s}}$ is the velocity of the boundary of the control volume V_t. We use a stationary background Cartesian mesh to deal with moving boundary problems so $\mathbf{v}_{\mathrm{s}} = 0$ on any flow interfaces of a cell, but on a moving solid face $\mathbf{v}_{\mathrm{s}}$ is the velocity of the moving boundary. Finally, the governing equations are closed by the ideal gas equation of state

$$p = (\gamma - 1)\left[e - \frac{\rho}{2}(u^2 + v^2 + w^2) \right] \tag{14}$$

3.2. Finite volume discretization

The flow solver used here is a MUSCL–Hancock [28] scheme of the Godunov type, with appropriate modifications for use on cut cell meshes and to incorporate body motion. This is a second-order, two-step upwind scheme. The predictor step uses a non-conservative approach, which defines an intermediate value over a half-time interval $\Delta t/2$

$$(V\mathbf{U})_{ijk}^{n+1/2} = (V\mathbf{U})_{ijk}^{n} - \frac{\Delta t}{2} \sum_{m=1}^{M} \mathbf{F}(\mathbf{U}_m)\cdot\mathbf{S}_m^n \tag{15}$$

where V is the cell volume, $\mathbf{S}_m^n$ is a cell face side vector and M is the maximum number of cell faces. For a flow (or uncut) cell, $M = 6$; for a cut cell, $M = 7$. The flux function $\mathbf{F}(\mathbf{U}_m)$ is evaluated at the midpoints of cell faces following a linear reconstruction of the flow solution within each cell, via

$$\mathbf{U}_m = \mathbf{U}_{ijk}^n + \mathbf{r}_m\cdot\nabla\mathbf{U}_{ijk}^n \tag{16}$$

where $\mathbf{r}_m$ is the normal distance vector from the cell centroid to cell face m and $\nabla\mathbf{U}_{ijk}^n$ is a limited gradient vector in space (see next section).

The corrector step of the scheme is fully conservative. The intermediate solution from the predictor step is used to define a set of left- and right-states for a series of Riemann problems at each cell interface. These are solved to provide a set of upwind interface fluxes used to update the flow solution over the time interval Δt

$$(V\mathbf{U})_{ijk}^{n+1} = (V\mathbf{U})_{ijk}^{n} - \Delta t \sum_{m=1}^{M} \mathbf{F}(\mathbf{U}_m^{\mathrm{L,R}})\cdot\mathbf{S}_m^{n+1/2} \tag{17}$$

where the upwind flux $\mathbf{F}(\mathbf{U}_m^{\mathrm{L,R}})$ is obtained by solving a local Riemann problem normal to the cell interface. The left- and right-states at interface m may be calculated by

$$\begin{cases} \mathbf{U}_m^{\mathrm{L}} = \mathbf{U}_{ijk}^{n+1/2} + \mathbf{r}_m^{\mathrm{L}}\cdot\nabla\mathbf{U}_{ijk}^{n} \\ \mathbf{U}_m^{\mathrm{R}} = \mathbf{U}_{n(ijk)}^{n+1/2} + \mathbf{r}_m^{\mathrm{R}}\cdot\nabla\mathbf{U}_{n(ijk)}^{n} \end{cases} \tag{18}$$

where $n(ijk)$ relates to the right neighbouring cell at interface m.

To solve the Riemann problem, either an exact Riemann solver or various approximate Riemann solvers can be used. Here, a version of the Harten, Lax and van Leer approximate Riemann solver with modifications to improve the resolution of contact surfaces, hereafter denoted by HLLC, is used at fluid interfaces [29]. Other choices of approximate Riemann solvers are possible (e.g. Roe [30], Osher [31]), but we have found the HLLC solver to be sufficiently accurate and robust in practice. An exact Riemann solution for a moving piston is used on the solid boundaries (faces) of a cut cell, where the flux is evaluated as follows:

$$\mathbf{F}_n^* \cdot \mathbf{S} = \begin{bmatrix} 0 \\ p_n^*(S_x^{\mathrm{l}} - S_x^{\mathrm{r}}) \\ p_n^*(S_y^{\mathrm{l}} - S_y^{\mathrm{r}}) \\ p_n^*(S_z^{\mathrm{l}} - S_z^{\mathrm{r}}) \\ p_n^* \mathbf{v}_{sn} |\mathbf{S}| \end{bmatrix} \tag{19}$$

where p_n^* and $\mathbf{v}_{sn}$ are the pressure and normal velocity arising from the Riemann solution on the moving solid boundary. Full details of the Riemann solution can be found in Reference [26].

3.3. Gradient calculation

The gradient calculation on flow cells away from a solid boundary is straightforward and needs no detailed explanation. However, for cells near a solid boundary, the approach depends on whether the boundary is static or moving. We use reflection boundary conditions on a solid boundary, where the variables in fictional cell R (see Figure 5) are obtained as follows:

$$\begin{cases} \rho_R = \rho_{ijk} \\ \mathbf{v}_R = \mathbf{v}_{ijk} - 2(\mathbf{v}_{ijk} \cdot \mathbf{n})\mathbf{n} + 2(\mathbf{v}_s \cdot \mathbf{n})\mathbf{n} \\ p_R = p_{ijk} \\ e_R = \dfrac{p_R}{\gamma - 1} + \dfrac{1}{2}\rho_R |\mathbf{v}_R|^2 \end{cases} \tag{20}$$

The gradients on cut cell (i, j, k) may be of two types: fluid and solid. We calculate the fluid gradients and solid gradients separately, i.e.

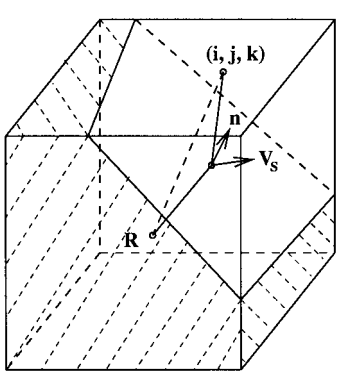

Figure 5. Reflection boundary conditions at a moving surface.

$$\mathbf{U}_x^{\mathrm{f}} = G\left(\frac{\mathbf{U}_{i+1,j,k} - \mathbf{U}_{i,j,k}}{\Delta x_{i+1/2,j,k}},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_{i-1,j,k}}{\Delta x_{i-1/2,j,k}}\right) \tag{21}$$

$$\mathbf{U}_y^{\mathrm{f}} = G\left(\frac{\mathbf{U}_{i,j+1,k} - \mathbf{U}_{i,j,k}}{\Delta y_{i,j+1/2,k}},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_{i,j-1,k}}{\Delta y_{i,j-1/2,k}}\right) \tag{22}$$

$$\mathbf{U}_z^{\mathrm{f}} = G\left(\frac{\mathbf{U}_{i,j,k+1} - \mathbf{U}_{i,j,k}}{\Delta z_{i,j,k+1/2}},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_{i,j,k-1}}{\Delta z_{i,j,k-1/2}}\right) \tag{23}$$

where $\Delta x_{i+1/2,j,k} = x_{i+1,j} - x_{i,j,k}$, $\Delta y_{i,j+1/2,k} = y_{i,j+1,k} - y_{i,j,k}$ and $\Delta z_{i,j,k+1/2} = z_{i,j,k+1} - z_{i,j,k}$;

$$\mathbf{U}_x^{\mathrm{s}} = G\left(\frac{\mathbf{U}_R - \mathbf{U}_{i,j,k}}{\Delta x_R},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_{i-1,j,k}}{\Delta x_{i-1/2,j,k}}\right) \tag{24}$$

$$\mathbf{U}_y^{\mathrm{s}} = G\left(\frac{\mathbf{U}_{i,j+1,k} - \mathbf{U}_{i,j,k}}{\Delta y_{i,j+1/2,k}},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_R}{\Delta y_R}\right) \tag{25}$$

$$\mathbf{U}_z^{\mathrm{s}} = G\left(\frac{\mathbf{U}_{i,j,k+1} - \mathbf{U}_{i,j,k}}{\Delta z_{i,j,k+1/2}},\ \frac{\mathbf{U}_{i,j,k} - \mathbf{U}_R}{\Delta z_R}\right) \tag{26}$$

where $\Delta x_R = x_R - x_{i,j,k}$, $\Delta y_R = y_{i,j,k} - y_R$, $\Delta z_R = z_{i,j,k} - z_R$ and G is a slope limiter function that is used to prevent over- or undershoots. The limiter function may take one of the following forms:

I. **The Superbee limiter**

$$G(a, b) = s \cdot \max[0,\ \min(2|b|, s \cdot a),\ \min(|b|, 2s \cdot a)], \quad s = \mathrm{sign}(b) \tag{27}$$

II. **The van Leer limiter**

$$G(a, b) = \frac{a|b| + |a|b}{|a| + |b|} \tag{28}$$

Once the two types of gradients have been calculated, an area average technique is used to obtain unique gradients within the cut cell; for example, in the x-direction

$$\mathbf{U}_x = \frac{S_x^{\mathrm{f}}\mathbf{U}_x^{\mathrm{f}} + S_x^{\mathrm{s}}\mathbf{U}_x^{\mathrm{s}}}{\max(S_x^{\mathrm{l}}, S_x^{\mathrm{r}})} \tag{29}$$

where

$$S_x^{\mathrm{s}} = |S_x^{\mathrm{l}} - S_x^{\mathrm{r}}| \tag{30}$$

and

$$S_x^{\mathrm{f}} = \max(S_x^{\mathrm{l}}, S_x^{\mathrm{r}}) - S_x^{\mathrm{s}} \tag{31}$$

An analogous treatment is used for the components of the gradient in the y- and z-directions. A gradient vector at cut cell (i, j, k) can then be obtained as

$$\nabla \mathbf{U}_{ijk} = \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \\ \mathbf{U}_z \end{bmatrix} \tag{32}$$

Given the gradient vector $\nabla \mathbf{U}_{ij}$, a reconstructed solution vector $\mathbf{U}(x, y, z)$ can be found anywhere within the cut cell from

$$\mathbf{U}(x, y, z) = \mathbf{U}_{ijk} + \mathbf{r} \cdot \nabla \mathbf{U}_{ijk} \tag{33}$$

where $\mathbf{r}$ is the normal distance vector from the cell centroid to any specific interface or solid boundary.

4. EXTENSION TO MOVING BODY PROBLEMS

4.1. Description of body motion

As in two-dimensional problems, three-dimensional rigid body motion involves both translation and rotation. Instead of prescribing the velocity of every data point on the body periphery, we first determine the translation velocity $\mathbf{v}_c$ and rotational velocity ω_c of the body's centre of mass and then calculate values at other points on the body from $\mathbf{v}_c$ and ω_c. Values for $\mathbf{v}_c$ and ω_c can be estimated from the pressure distribution about the surface of the moving body. Here, we briefly describe how to determine the new position of the moving body and the velocity of any point on it based on known values of $\mathbf{v}_c$ and ω_c.

In Figure 6, the position vector of point p on the surface of the solid body is given by

$$\mathbf{r}_p = \mathbf{r}_c + \mathbf{r}_o = \mathbf{r}_c + \mathbf{r}_\omega + \mathbf{r}_r \tag{34}$$

therefore, the velocity of point p is

$$\mathbf{v}_p = \mathbf{v}_c + \omega_c \times \mathbf{r}_o \tag{35}$$

Now, define unit vectors $\mathbf{e}_r$ and $\mathbf{e}_n$ as

$$\mathbf{e}_r = \frac{\mathbf{r}_r}{|\mathbf{r}_r|}, \qquad \mathbf{e}_n = \frac{\omega_c}{|\omega_c|} \times \mathbf{e}_r \tag{36}$$

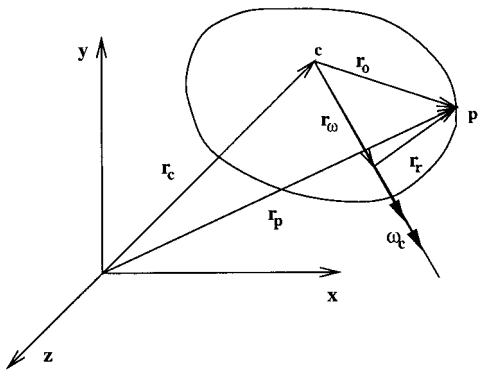

Figure 6. The position vector of a point p on a solid body.

Then, given the time step Δt, the new position and velocity of point p (see Figure 7) can be obtained from

$$\Delta\phi = \omega_c \Delta t \tag{37}$$

$$\mathbf{r}_o^{n+1} = \mathbf{r}_\omega^n + |\mathbf{r}_r^n|[\cos(\Delta\phi)\mathbf{e}_r + \sin(\Delta\phi)\mathbf{e}_n] \tag{38}$$

$$\mathbf{r}_c^{n+1} = \mathbf{r}_c^n + \mathbf{v}_c \Delta t \tag{39}$$

$$\mathbf{r}_p^{n+1} = \mathbf{r}_c^{n+1} + \mathbf{r}_o^{n+1} \tag{40}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_c + \omega_c \times \mathbf{r}_o^{n+1} \tag{41}$$

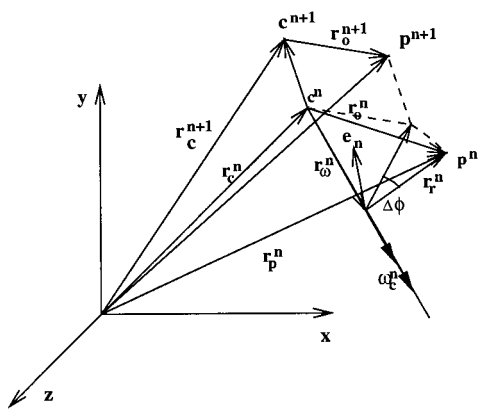

Figure 7. Finding the new position of point p.

4.2. Cell merging on static and moving boundaries

When boundary fitting a static body and/or in cases involving body motion, a cut cell may turn out to be arbitrarily small and numerical stability considerations may force an unrealistically small time step. To overcome this problem, a cell merging technique [14,26,32] is used, which is simple to implement and extends easily to moving boundary problems. The basic idea is to combine several neighbouring cells together so that the interfaces between merged cells are ignored and waves can travel in a newly combined larger cell without reducing the global time step. To implement this technique, we need first to determine which cells should be merged to avoid problems in the flow integration process. By providing a minimum volume criterion $V_{\min}$ for a cut cell, the procedure for finding which cells to merge is

1. Check the cut cell volume against the $V_{\min}$ criterion. If its volume is larger than $V_{\min}$, no further processing is needed so go to step 4; if its volume is smaller than $V_{\min}$, a neighbouring cell will be needed for merging so go to step 2.
2. Check to see whether the cut cell has already appeared in the cell-merging list. If this is the case, no further processing is needed so go to step 4; if this is not the case, a suitable neighbouring cell must be found for merging so go to step 3.
3. The choice of which neighbouring cell to merge with depends on the slope of the solid side of the cut cell. For example, suppose two cut cells A and B in Figure 8 (shown in two-dimensional form) are smaller than $V_{\min}$, and both are of type one. By comparing the right and bottom flow interfaces and merging in the direction of the larger one we find that a suitable cell for cut cell A to merge with is cut cell C, but for cut cell B it is flow cell D. For the other types of cut cell, an analogous approach is used. Once a neighbouring cell has been identified it is saved to the cell-merging list.
4. Proceed to the next cut cell and then repeat steps 1–3 until all cut cells have been processed.

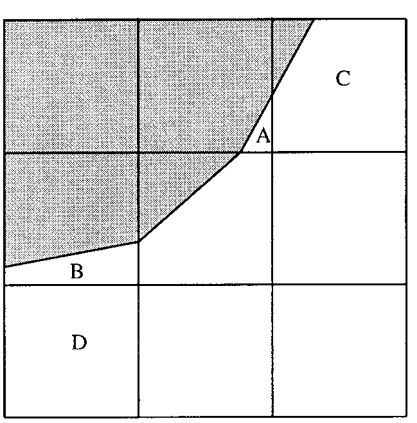

Figure 8. Finding a neighbouring cell to merge with.

The choice for the minimum volume criterion $V_{\min}$ is based in practice on a trade-off between the time step constraint and resolution accuracy. In our calculations, $V_{\min}$ was set to one half of the flow cell size.

In cases involving body motion, the cut cell data changes in one of the following four ways:

1. Cut cell becomes solid cell.
2. Cut cell becomes an uncut flow cell.
3. Cut cell remains unchanged.
4. Uncut flow cell becomes a cut cell.

Categories 3 and 4 do not cause any problems. However, where a cut cell becomes solid (category 1), the volume of the cell at the end of the time step is zero and this will obviously lead to problems within the flow solver. For category 2, although the cell finally becomes a flow cell, failure to consider the new born cell will result in strict conservation form being lost numerically.

For clarity of exposition, we take the two-dimensional case as an example of what to do. A time step Δt, based on flow cell B will be too large for cut cell A, which will also become solid at the end of the time step Δt (see Figure 9). To merge the two cells, we first compute updates at cells A and B as usual

$$\Delta(V\mathbf{U})_{A,B} = -\Delta t \sum_{m=1}^{M_{A,B}} \mathbf{F}_m \cdot \mathbf{S}_m \tag{42}$$

Then, we update the merged cell C simply by combining the volume updates of cells A and B

$$\Delta(V\mathbf{U})_C = \Delta(V\mathbf{U})_A + \Delta(V\mathbf{U})_B \tag{43}$$

The conserved variables $\mathbf{U}$ for cell C at time t^{n+1} are

$$(V\mathbf{U})_C^{n+1} = (V\mathbf{U})_A^n + (V\mathbf{U})_B^n - \Delta t\left(\sum_{m=1}^{M_A} \mathbf{F}_m \cdot \mathbf{S}_m + \sum_{m=1}^{M_B} \mathbf{F}_m \cdot \mathbf{S}_m\right) \tag{44}$$

We can see that if any of the merged cells finally vanish, their contribution to the mass, momentum and energy will be transferred into neighbouring cells so that conservation is automatically maintained.

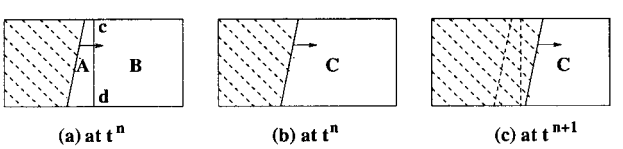

Figure 9. Cell merging technique for a moving boundary.

In order to prevent a single cut cell from becoming solid without first merging with neighbouring cells, an appropriate estimate of the time step is introduced as follows:

$$\Delta t_{x,y,z} = \frac{\sqrt[3]{V_{\min}}}{\max(|\mathbf{v}_s|_{x,y,z}, |\mathbf{v}|_{x,y,z}) + a} \tag{45}$$

$$\Delta t = v \min(\Delta t_x, \Delta t_y, \Delta t_z) \tag{46}$$

where a is the local speed of sound and v in our calculations was taken to be 0.9.

5. NUMERICAL RESULTS

5.1. ONERA M6 wing at transonic flow

The first computed example is a well-documented steady flow problem: transonic flow past an ONERA M6 wing. The M6 wing has a leading edge sweep angle of 30°, a trailing edge sweep angle of 15.8°, an aspect ratio of 3.8 and a taper ratio of 0.562. The wing has the ONERA 'D' aerofoil section, which is a 10 per cent maximum thickness-to-chord ratio conventional section. The standard test conditions are a Mach 0.84 free stream flow at 3.06° angle of attack. Since this is a transonic flow calculation, the computational domain was chosen as $14D \times 10D \times 12D$ (D is the maximum chord length of the root section of the wing). A Cartesian mesh with $140 \times 80 \times 95$ cells was used in the x-, y- and z-directions respectively. To accurately resolve the geometry of the wing, a fine mesh was used near the wing surface, which was then stretched to the outer boundaries of the flow domain (see Figure 10). Pressure contours on the symmetry plane and a cutting plane near the wing upper surface are shown in Figure 10. The figure clearly shows a sharply captured Lambda-type three-dimensional shock structure formed by the two inboard shock waves on the upper surface of the wing. These two shock waves merge together near a station at 80 per cent semi-span to form a single strong shock wave in the out-board region of the wing. In the symmetry plane, a normal shock can be seen as the vertical portion of the Lambda-type shock structure. Figure 11 shows a comparison between the computed pressure coefficient distributions and experimental data [33] at four semi-span sections. Generally, the computed results and experimental data agree to within five per cent around each section. However, in the calculated results, the shock strength is overpredicted by approximately 10 per cent and the shocks are located further aft than in the experimental data. These discrepancies are attributed to the fact that our calculations are inviscid and ignore viscous effects in the boundary layer.

5.2. Shock wave diffraction on a cone

Bryson and Gross [34] have studied shock wave diffraction by cones. In their experiments, a series of Schlieren images of shock diffraction on cones with semi-apex angles varying between 9.7° and 44.7° were made at a mean shock Mach number of 3.68 ± 0.16. Figure 12 shows a typical image from this series, where the semi-apex angle is 35.1° and shock Mach number $M_s = 3.55$. At these conditions, the main feature is a complex Mach reflection around the cone.

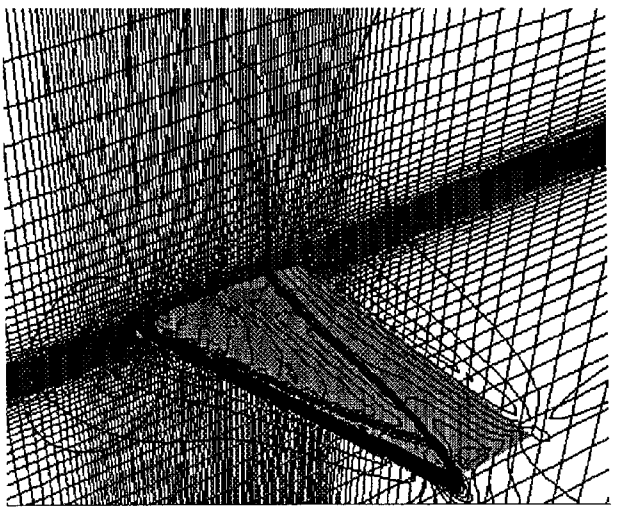

Figure 10. ONERA M6 wing: cut cell mesh and pressure contours on the symmetry plane and a cutting plane near upper surface.

Since the cone is axisymmetric, only one quadrant of it has been calculated with our three-dimensional method. A uniform Cartesian mesh with $90 \times 70 \times 70$ cells was used in the x-, y- and z-directions respectively on a computational domain of $9 \times 7 \times 7$ units with the origin of co-ordinates at the bottom left-hand corner of the mesh and the vertex of the cone located at $(1, 7, 0)$. Figure 13 shows the cut cell mesh and density contours, which compare favourably with the experimental results in Figure 12.

Calculations have been performed on three uniform Cartesian meshes with $45 \times 35 \times 35$, $70 \times 51 \times 51$ and $90 \times 70 \times 70$ cells respectively. Line contours of density on the two symmetry planes (x–y and x–z) are shown in Figures 14 and 15 respectively. In the calculations, density is non-dimensionalized with respect to the value in the quiescent gas. All the diffraction features of the complex Mach reflection such as the Mach stem, triple point and the kink in the reflected shock are resolved adequately on the two finest meshes. The Mach stem is not resolved on the coarsest mesh. These results have been used to derive a measure of grid convergence through the use of a grid convergence index (GCI), as proposed by Roache [35], which provides a mechanism for the uniform reporting of grid refinement studies in computational fluid dynamics [36,37]. The basic idea is to approximately relate the results from any grid refinement test to the expected results from a grid doubling using a second-order method. The GCI is based upon a grid refinement error estimator derived from the theory of generalized Richardson extrapolation. The simple formulae are independent of the equations being discretized and the dimensionality of the problem, and can be

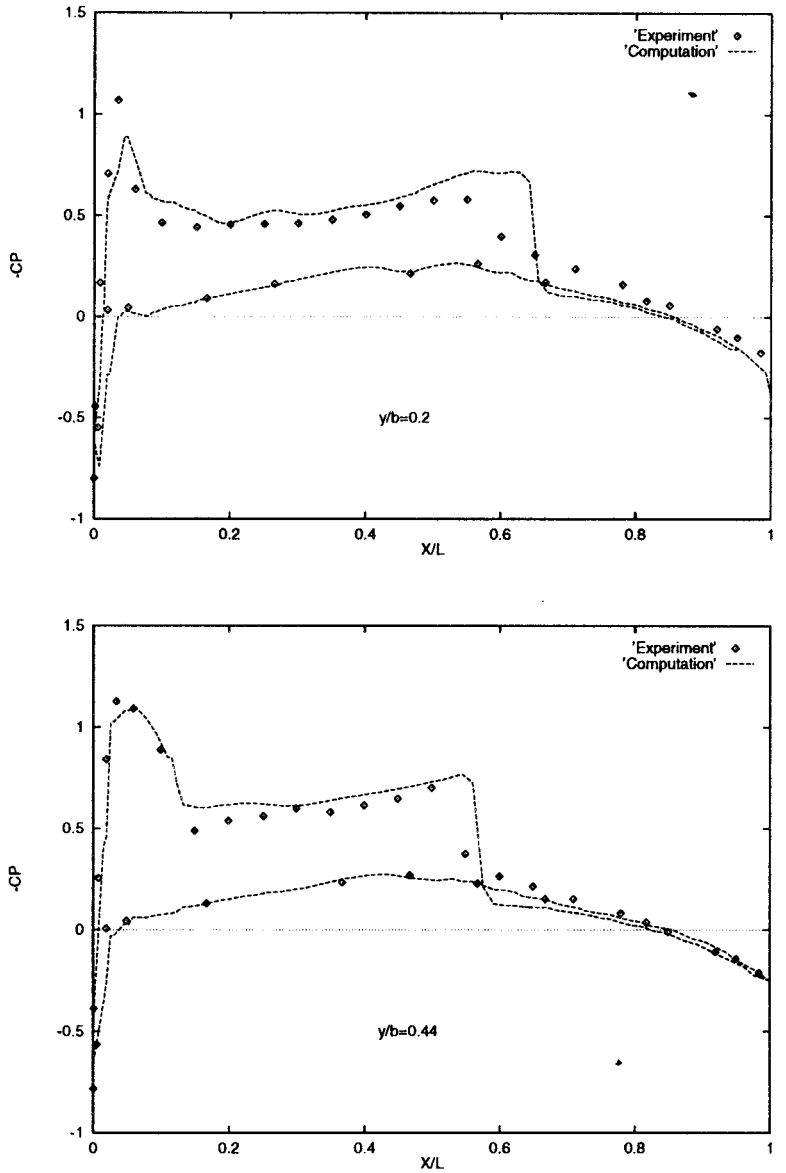

Figure 11. ONERA M6 wing: comparison between computed and experimental surface pressure coefficient distributions at four span-wise sections.

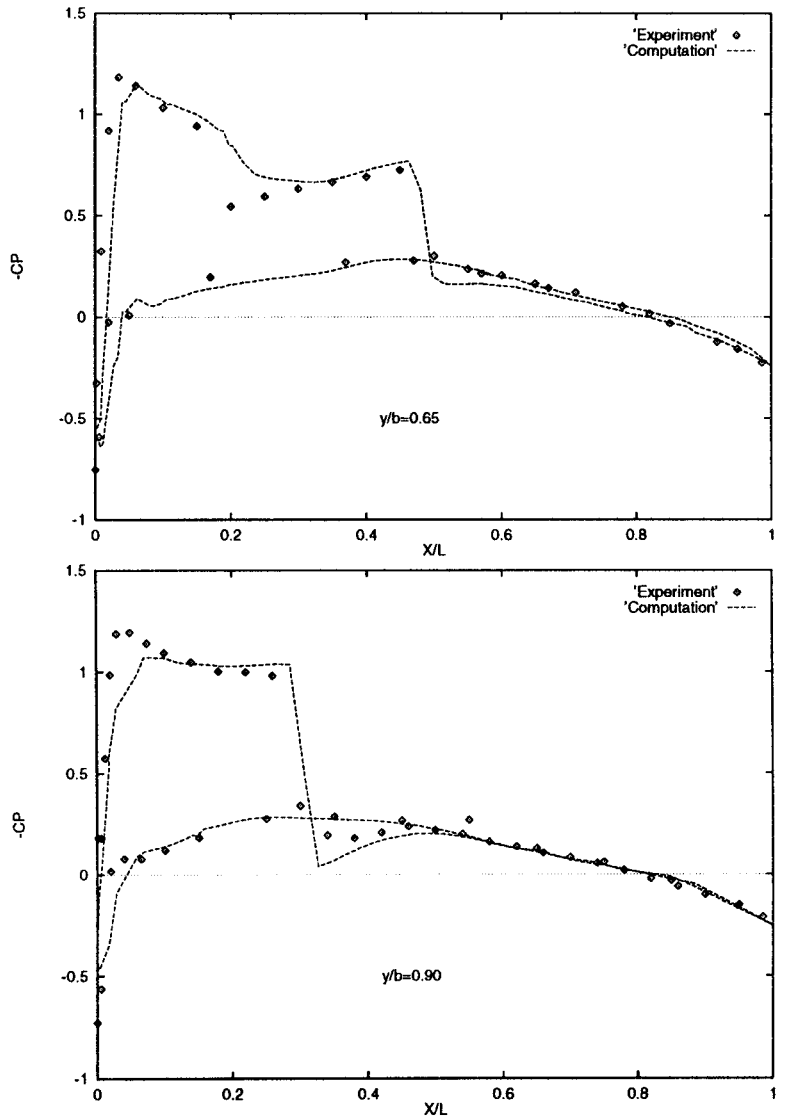

Figure 11 (*Continued*)

applied *a posteriori* to solutions on two grids with no reference to the codes, algorithms or governing equations that produced the solutions as long as the original solutions are second-order accurate.

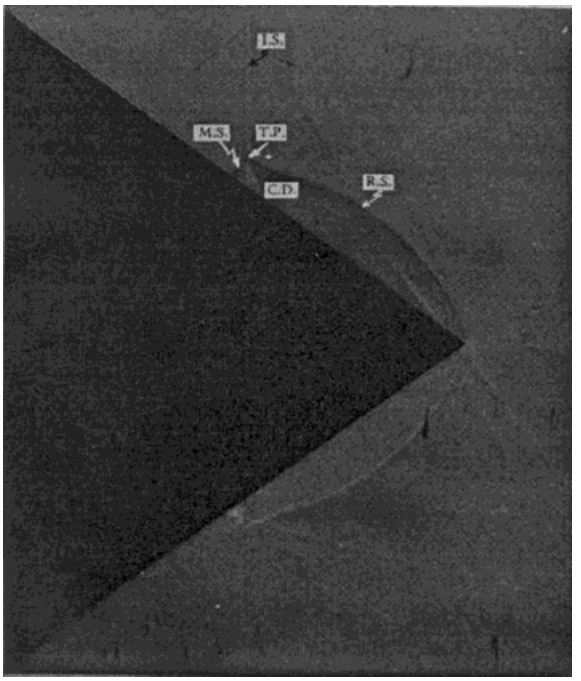

Figure 12. Shock diffraction on a cone: Schlieren image of shock diffraction on a cone of semi-apex angle 35.1° and $M_s = 3.55$. Notation: I.S., incident shock; M.S., Mach stem; R.S., reflected shock; C.D., contact discontinuity; T.P., triple point (after Bryson and Gross [34]).

Given solution data at the same location on a fine grid and a coarse grid denoted by ρ_1 and ρ_2 respectively, the GCI is given by

$$\mathrm{GCI}_{21} = 3\,\frac{|\epsilon_{21}|}{r^p - 1} \tag{47}$$

where p is the formal order of accuracy of the numerical method (2 in the present case), r is the mesh refinement factor

$$r = \frac{\Delta x_2}{\Delta x_1}$$

and ϵ_{21} is a measure of the percentage relative error between the two solutions at the given location, i.e.

$$\epsilon_{21} = 100\,\frac{\rho_1 - \rho_2}{\rho_1}$$

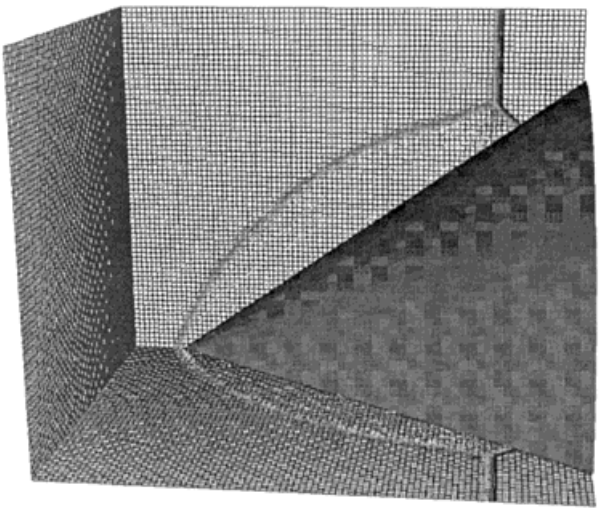

Figure 13. Shock diffraction on a cone: mesh and line contours of density.

If three or more grids are used in a mesh refinement study, the solution is in the asymptotic region of grid convergence if $GCI_{21} \leq r^p GCI_{32}$ [35].

Table I shows predicted densities and values of the GCI at various points in the domain on each grid. Since the present method is second-order ($p = 2$) we require $GCI_{21} \leq 3.5 GCI_{32}$ to demonstrate a converged solution at a particular point. Comparing GCI_{21} and GCI_{32} in this way shows the solution is convergent at the points listed and the GCI provides a measure of the percentage error in the solution at that point.

5.3. Shock interaction with a missile launcher

A numerical simulation of planar shock wave interaction with a missile launcher (due to Lohner [38]) obtained using the authors' two-dimensional Cartesian cut cell method was reported in Reference [26]. Here, we have used a three-dimensional version of the geometry for a calculation with the present method. The geometry for the three-dimensional problem is a launcher vehicle with an axisymmetric missile placed on it. Because of the vertical symmetry in the problem, only one half of the flow field was discretized. A uniform Cartesian mesh with $120 \times 90 \times 80$ cells was used in the x-, y- and z-directions respectively on a computational domain of $9.6 \times 7.2 \times 6.4$ units. Figure 16 shows the geometry of the missile launcher, the cut cell mesh in the symmetry plane and one cutting plane along the length of missile launcher. The initial conditions were set as follows: a planar incident shock at Mach 3.0 and $\gamma = 1.4$ was located a certain distance to the left of the missile launcher travelling from left to right. Once the incident shock impinges on the vehicle, part of it is reflected at its front face, while the transmitted part is diffracted around the vehicle. Figure 17 shows two snapshots of density contours at the time when the transmitted shock has nearly passed the whole missile launcher.

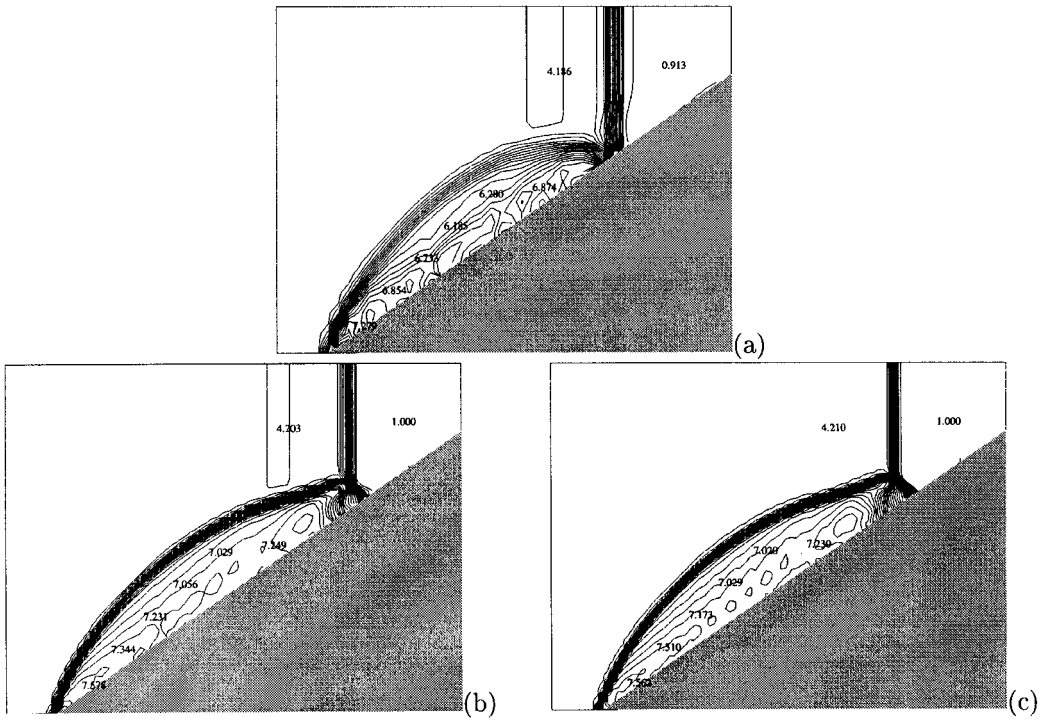

Figure 14. Shock diffraction on a cone: line contours of density for cone of semi-apex angle 35.1° and $M_s = 3.55$ (x–y plane); (a) $\Delta x = \Delta y = 0.2$, (b) $\Delta x = \Delta y = 0.13$ and (c) $\Delta x = \Delta y = 0.1$.

We can see that the structure of the shock diffraction in the symmetry plane is very similar to that found in the two-dimensional case [26,38]. Complex vortices are generated behind the head of the launcher, which appear to be entirely three-dimensional shock diffraction structures.

5.4. Muzzle brake flow fields

As an example of a moving-body calculation we have calculated an axisymmetric muzzle brake flow field. In Reference [39], Widhopf *et al.* presented experimental data and numerical results for a muzzle blast wave interaction with a moving projectile and multi-element brakes. The projectile was initially located inside a barrel and propelled by high-pressure propellant gas. As the projectile emerges from the barrel, the propellant gas escapes from the muzzle exit and impinges on the baffle brake. The expanding propellant gas rapidly reaches sonic conditions at the exit and then there follows a fast decay. In the numerical study, however, sonic conditions were set at the muzzle exit and consequently the time-dependent overpressure on the brake surface is characterized by a large amplitude wave relaxing to a steady state distribution. In our calculations, a uniform Cartesian mesh with 240×96 cells was used in the x- and

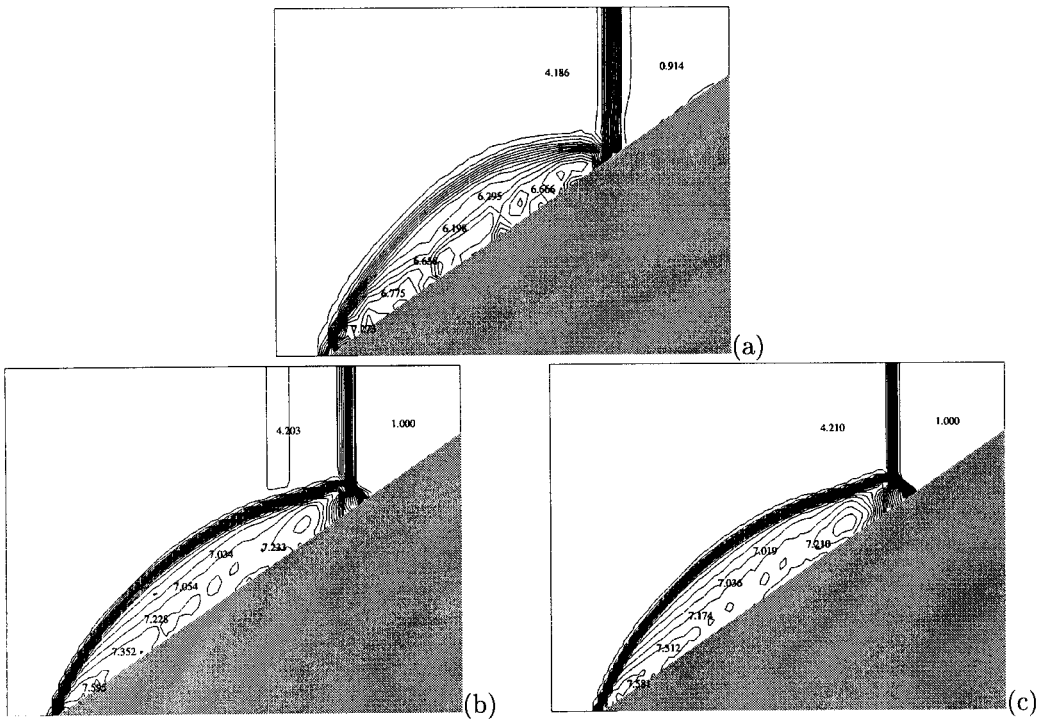

Figure 15. Shock diffraction on a cone: line contours of density for cone of semi-apex angle 35.1° and $M_s = 3.55$ (x–z plane); (a) $\Delta x = \Delta y = 0.2$, (b) $\Delta x = \Delta y = 0.13$ and (c) $\Delta x = \Delta y = 0.1$.

r-directions respectively on a $12D \times 4.8D$ domain ($D = 30$ mm). Figure 18 shows a partial view of the cut cell mesh, muzzle brakes and the moving projectile at time 0.18 ms. Figure 20 shows three sets of Mach number contours at times 0.18, 0.28 and 0.37 ms. The first set of contours show the emergence of the propellant gases behind the projectile, while the second and third sets show the gas emergence from the first and second brake respectively. The calculated peak and steady state overpressure values have been superimposed on the measured data in Figure 19. Except at locations close to the brake gap, where there are discrepancies attributed to differences in the initial conditions between the calculations and experiments, both peak and steady state overpressures agree with the measured data to within ten per cent.

5.5. A store separation problem

Our second example of a moving-body problem is a hypothetical store separation. The problem considered is as follows: initially, a store is placed in a cavity, external to which is a free stream flow at $M_\infty = 1.5$. Inside the cavity, the flow is stationary with the same pressure as the free stream flow. The store motion is prescribed. Because of the flow symmetry, only one half of the flow field has been calculated. In the present calculations, a Cartesian mesh

Table I. Shock wave diffraction on a cone: grid convergence indices comparing results from the coarse, medium and fine grids.

Point	ρ_3	ρ_2	ρ_1	GCI_{32}	GCI_{21}
x–y Plane					
(7.5, 5.5)	0.913	1.00	1.00	18.3	0.0
(5.3, 5.4)	4.19	4.20	4.21	0.8	0.9
(5.0, 3.1)	6.87	7.25	7.23	10.5	1.4
(4.0, 3.0)	6.28	7.03	7.02	22.9	0.7
(3.3, 2.4)	6.19	7.06	7.03	27.1	2.0
(2.7, 1.8)	6.73	7.23	7.17	14.2	4.3
(2.1, 1.2)	6.85	7.34	7.31	13.8	2.5
(1.5, 0.5)	7.28	7.58	7.56	7.9	1.1
Δx	0.20	0.13	0.10		
x–z Plane					
(7.5, 5.5)	0.914	1.00	1.00	18.1	0.0
(5.3, 5.4)	4.19	4.20	4.21	0.8	0.9
(5.0, 3.1)	6.67	7.23	7.21	16.4	1.7
(4.0, 3.0)	6.29	7.03	7.02	22.6	1.1
(3.3, 2.4)	6.20	7.05	7.04	26.6	1.4
(2.7, 1.8)	6.66	7.23	7.17	16.5	4.0
(2.1, 1.2)	6.77	7.35	7.31	16.4	2.9
(1.5, 0.5)	7.27	7.59	7.58	8.5	0.9
Δx	0.20	0.13	0.10		

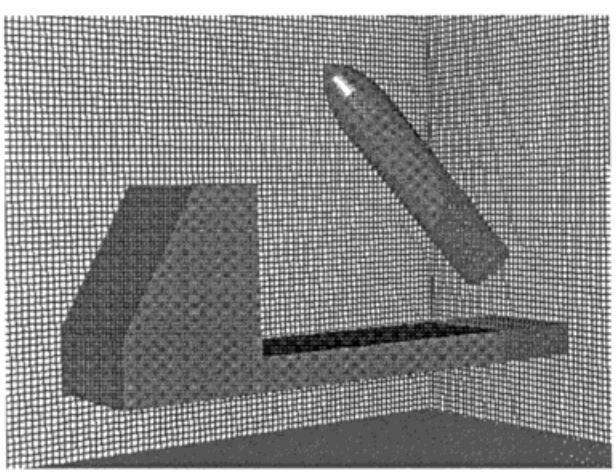

Figure 16. Missile launcher: cut cell mesh (shown on two cutting planes).

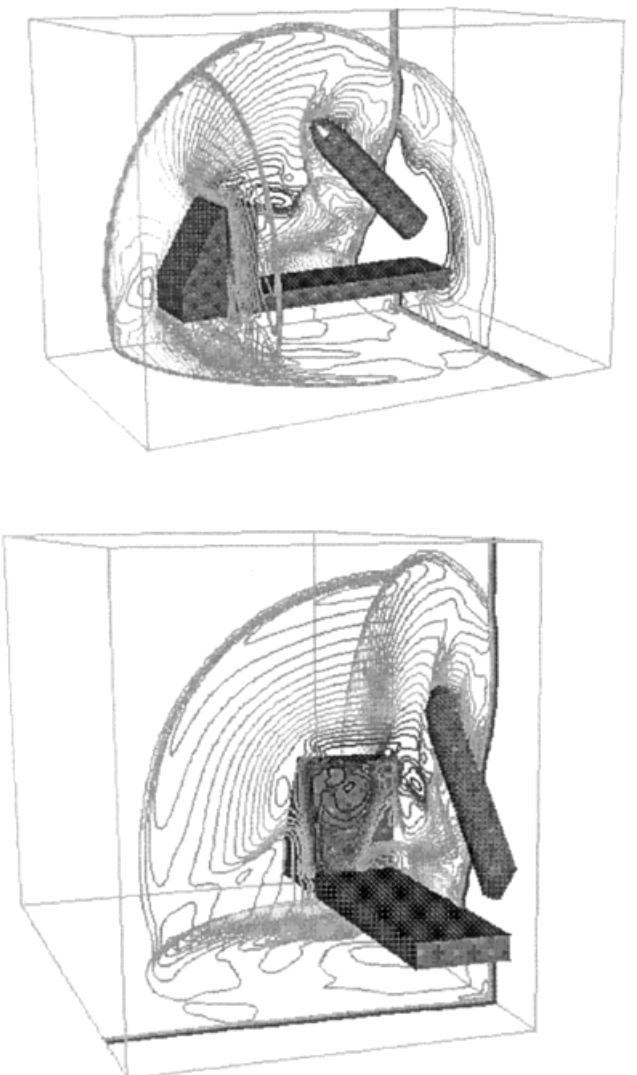

Figure 17. Missile launcher: line contours of density.

with $100 \times 62 \times 30$ cells was used in the x-, y- and z-directions respectively on a domain of $200 \times 124 \times 60$ units. The translational and rotational velocities for the store centre of mass were $v_c = [0.0, -0.5, 0.0]^T$ and $\omega_c = 0.00023$ respectively (non-dimensionalized with respect the free stream speed of sound and the diameter of the store). The resulting flow fields are shown at various times after the store has been ejected (at $t = 0$). The position of the store at each time is determined by the method described in Section 4.1. Figures 21 and 22 show the positions of the store, the cut cell mesh and density contours at two different times.

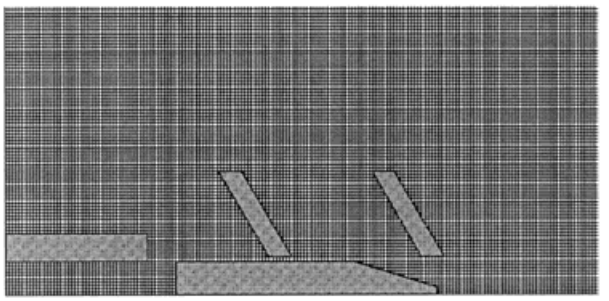

Figure 18. Muzzle brake problem: Cartesian mesh and muzzle brakes for a moving projectile at time 0.18 ms.

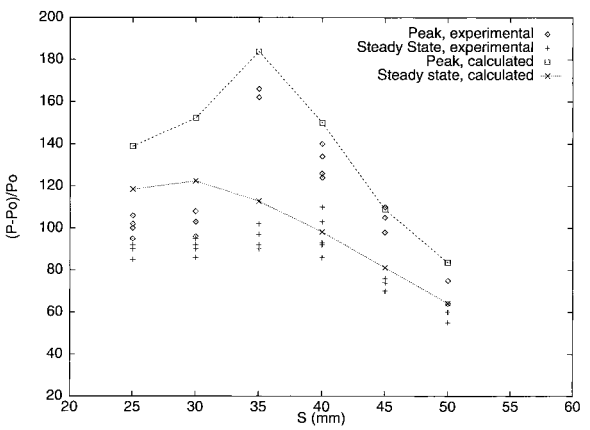

Figure 19. Muzzle brake problem: comparison of computed and measured peak and steady state overpressures on the first brake.

Figure 21 shows the computed flow fields at $t = 50$. At this time, the store has emerged from the cavity and a bow shock is produced around the nose of the store. Behind the bow shock, the flow expands to supersonic conditions and two normal shocks appear downstream on the body surface. Meanwhile, several vortices can be seen in the cavity. The store continues to fall and at $t = 100$, the bow shock moves with the store while the normal shocks have moved further aft. The flow features become very complex inside the cavity where shocks and three-dimensional vortices are visible (see Figure 22). From the computed results we can see that, although a relatively coarse mesh has been used, the shocks and other flow features are captured as sharp discontinuities, demonstrating the effectiveness and promise of the present Cartesian cut cell method for moving boundary/body problems in compressible flow, where large amplitude body motion occurs.

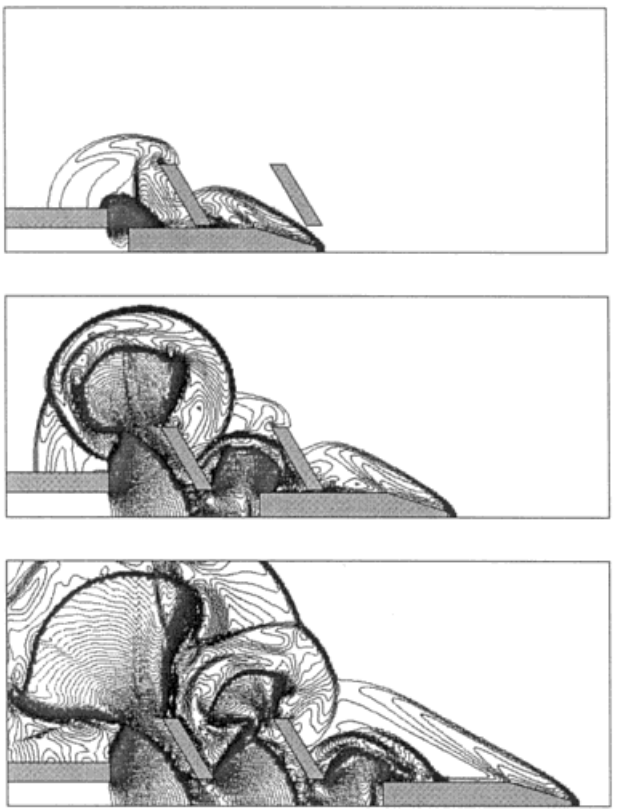

Figure 20. Muzzle brake problem: Mach number contours at times 0.18, 0.28 and $0.37 ms.

6. CONCLUSIONS

A method for the computation of compressible flows involving both static and moving bodies has been presented. The method, based on a Cartesian cut cell approach and multi-dimensional upwind finite volume scheme, can cope with compressible three-dimensional flows around arbitrarily complex configurations. For static body problems, the results indicate that the method is a viable alternative to unstructured meshes for dealing with arbitrarily complex, three-dimensional, geometries but the mesh generation procedures are much simpler and cheaper. For moving body problems, this approach allows bodies to move across a stationary background Cartesian mesh and hence problems such as mesh distortion and body motion restriction, which may occur when using other mesh approaches (e.g. involving moving meshes), are avoided completely. This may be of particular value in cases where large amplitude body motion is involved.

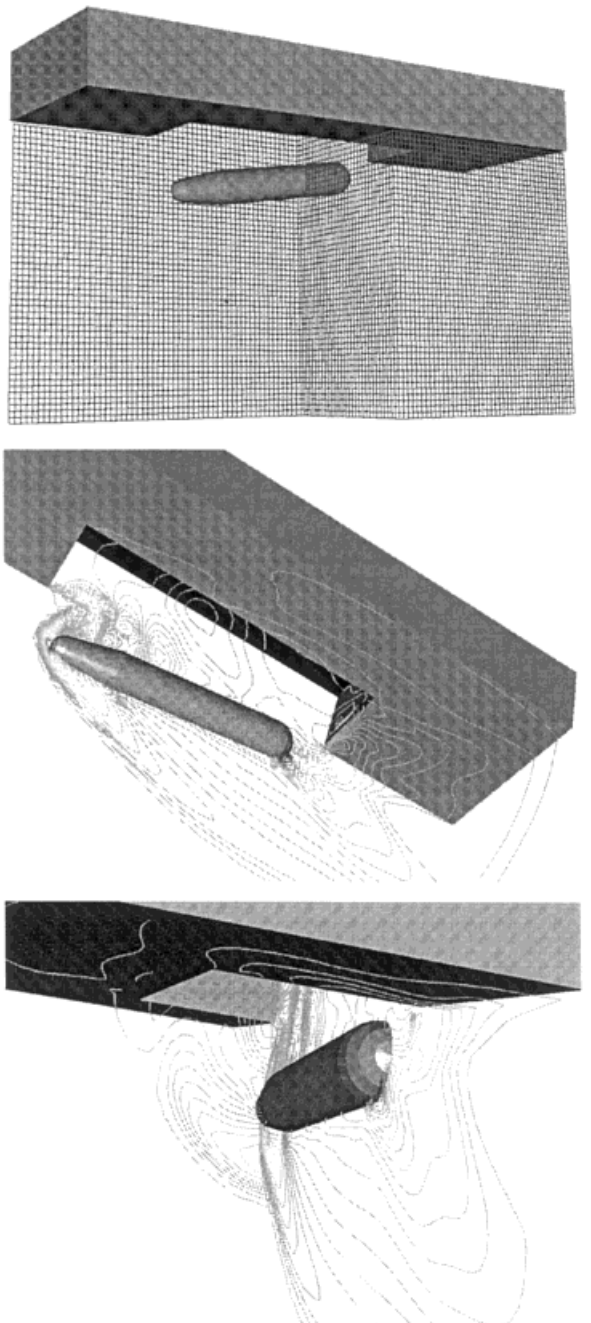

Figure 21. Store separation problem: cut cell mesh and density contours at $t = 50$.

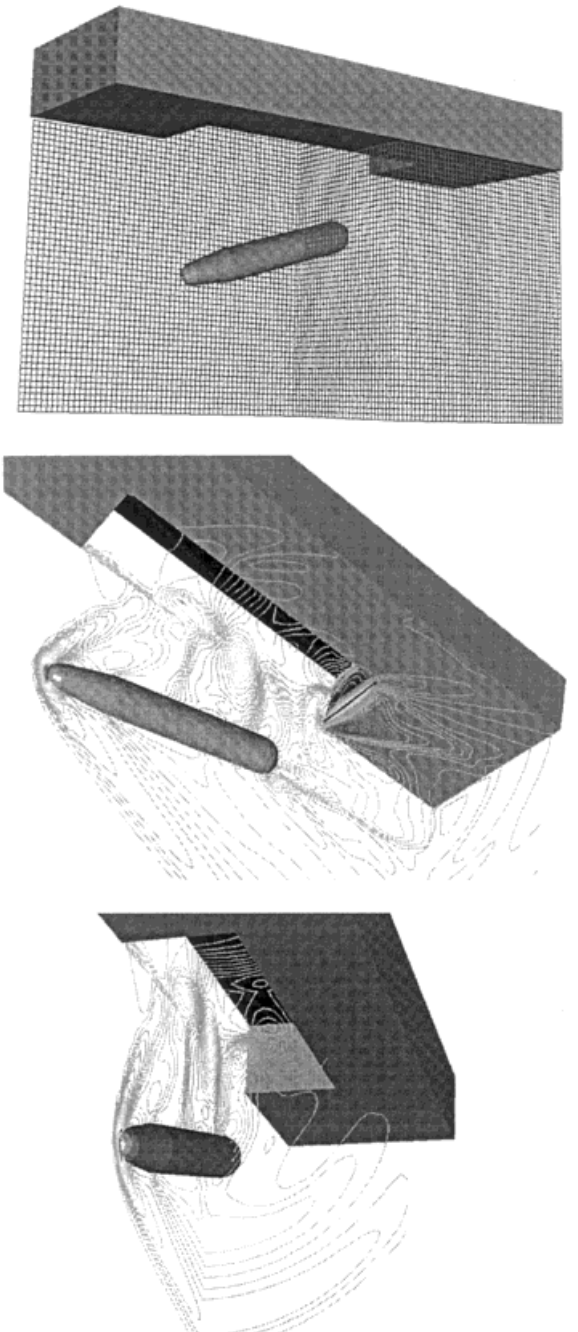

Figure 22. Store separation problem: cut cell mesh and density contours at $t = 100$.

At this time, the method is limited to the simulation of three-dimensional inviscid compressible flows. However, a three-dimensional adaptive Cartesian cut cell gridding method is currently under development in which both geometry based mesh refinement and solution based mesh refinement techniques are employed. These developments will enable the inviscid method to be extended to viscous flows in the near future, as well as improving the resolution of features of the inviscid flow in some of the examples presented here.

APPENDIX A. NOMENCLATURE

a	sound speed
e	total energy per unit volume
$\mathbf{F}$	flux vector function
M	Mach number
$\mathbf{n}$	normal unit vector
p	static pressure
$\mathbf{r}$	normal distance from a cell centroid or point position vector
S	surface of a domain or cell face area
$\mathbf{S}$	cell face area vector
t	time
u, v, w	Cartesian components of velocity
$\mathbf{U}$	vector of conserved variables
$\mathbf{v}$	velocity vector
V	cell volume
$\mathbf{V}$	contravariant velocity
x, y, z	Cartesian co-ordinates

Greek letters

γ	ratio of specific heats
ν	Courant–Friedrichs–Lewy (CFL) number
ρ	density
ω_c	rotational velocity
Ω	integration domain

Superscripts

f	fluid value
s	solid value
n	iteration step
*	star region in a Riemann fan
l, r	left-, right-hand side

Subscripts

f	fluid value
s	solid value or shock wave
l	face of a cell
L, R	left-, right-hand states of a Riemann problem
∞	infinity at free stream

REFERENCES

1. Brenner P. Three dimensional aerodynamics with moving bodies applied to solid propellant. Technical Report 91-2304, AIAA Paper, 1991.
2. Batina JT. Unsteady Euler algorithm with unstructured dynamic mesh for complex aircraft aerodynamic analysis. *AIAA Journal* 1991; **29**(3): 327–333.
3. Lohner R, Baum D. Adaptive h-refinement on 3D unstructured grids for transient problems. *International Journal of Numerical Methods in Fluids* 1992; **14**: 1407–1419.
4. Peraire J, Peiro J, Morgan K. Adaptive remeshing for three dimensional compressible flow computations. *Journal of Computational Physics* 1992; **103**: 269–285.
5. Weatherill NP, Hassan O, Marchant MJ, Marcum DL. Adaptive inviscid flow solutions for aerospace geometries using efficently generated unstructured tetrahedral meshes. Technical Report 93-3390, AIAA Paper, 1993.
6. Trepanier JY, Reggio M, Paraschivoiu M, Camarero R. Unsteady Euler solutions for arbitrarily moving bodies and boundaries. *AIAA Journal* 1993; **31**(10): 1869–1876.
7. Voinovitch P. A locally adaptive unstructured Euler solver. Technical Report 96/40, IMFL, 1996.
8. Shaw JA, Forsey CR, Weatherill NP, Rose KE. A block structured mesh generation technique for aerodynamic geometries. In *Numerical Grid Generation in CFD*, Hauser JA, Taylor C (eds). Pineridge Press: Swansea, 1986.
9. Steger JL, Dougherty FC, Benek JA. A chimera grid scheme. In *Advances in Grid Generation*, *ASME FED-5*, Ghia KN, Ghia U (eds), 1983; 59–69.
10. Albone CM. Embedded meshes of controllable quality synthesised from elementary geometric features. AIAA Paper No. 92-0662, 1992.
11. Blaylock TA, Onslow SH. Application of the fame method to store release prediction. In *Computational Fluid Dynamics '94*, Stuttgart, Germany, Periaux J, Wagner SJ, Pira R, Hirschel EH (eds). Wiley: Chichester, 1994; 1–20.
12. Jameson A, Baker TJ, Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper No. 86-0103, 1986.
13. Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics* 1987; **72**: 449–466.
14. Clarke J, Kassoy D, Riley N. On the direct initiation of a plane detonation wave. *Proceedings of the Royal Society of London* 1986; **A408**: 129–148.
15. Epstein B, Luntz AL, Nachson A. Cartesian Euler method for arbitrary aircraft configurations. *AIAA Journal* 1992; **30**(3): 679–687.
16. LeVeque RJ. High resolution finite volume methods on arbritary grids via wave propagation. *Journal of Computational Physics* 1988; **78**: 36–63.
17. De Zeeuw D, Powell KG. An adaptively refine Cartesian mesh solver for the Euler equations. *Journal of Computational Physics* 1993; **104**(1): 56–68.
18. Chaing Y, van Leer B, Powell KG. Simulation of unsteady inviscid flow on an adaptively refined Cartesian grid. AIAA Paper No. 93-3335-CP, 1993.
19. Berger MJ, Aftomis MJ, Melton JE. Adaption and surface modelling for Cartesian mesh methods. AIAA-95-1725-CP, 1995; 881–891.
20. Coirier WJ, Powell KG. An accuracy assessment of Cartesian mesh approaches for the Euler equations. *Journal of Computational Physics* 1993; **117**: 121–131.
21. Yang G, Causon DM, Ingram DM, Saunders R, Batten P. A Cartesian cut cell method for compressible flows—part A: static body problems. *Aeronautical Journal* 1997; **101**(1001): 47–56.
22. Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**: 64–84.
23. Quirk JJ. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Computers and Fluids* 1994; **23**(1): 125–142.

24. Slack DC, Walters RW. An interactive remeshing algorithm for the two dimensional Euler equations. AIAA Paper No. 90-0331, 1990.
25. Falcovitz J, Alfandary G, Hanoch G. A two dimensional conservation laws scheme for compressible flows with moving boundaries. *Journal of Computational Physics* 1997; **138**: 83–102.
26. Yang G, Causon DM, Ingram DM, Saunders R, Batten P. A Cartesian cut cell method for compressible flows—part B: moving body problems. *Aeronautical Journal* 1997; **101**(1001): 57–65.
27. Dewey BR. *Computer Graphics for Engineers*. Harper and Row: New York, 1988.
28. van Leer B. On the relation between the upwind-differencing schemes of Godunov, Engquist-Osher and Roe. *SIAM Journal on Scientific and Statistical Computing* 1984; **5**(1): 1–20.
29. Toro EF, Spruce M, Speares W. Restoration of the contact surface in the hll Riemann solver. *Shock Waves* 1994; **4**: 25–34.
30. Roe PL. Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics* 1981; **43**: 357–372.
31. Osher S, Solomon F. Upwind difference schemes for hyperbolic conservation laws. *Mathematics of Computing* 1982; **38**(158): 339–374.
32. Chiang Y, van Leer B, Powell KG. Simulation of unsteady inviscid flow on an adaptively refined Cartesian grid. AIAA Paper 92-0443-CP, 1992.
33. Schmitt V, Charpin F. Pressure distributions on the ONERA M6-wing at transonic mach numbers. AGARD AR-138, 1979.
34. Bryson AE, Gross RWF. Diffraction of strong shocks by cones, cylinders, and spheres. *Journal of Fluid Mechanics* 1961; **10**(1): 1–16.
35. Roache PJ. Perspective: a method for uniform reporting of grid refinement studies. *Journal of Fluids in Engineering* 1994; **116**: 405–412.
36. AIAA. Editorial policy statement on numerical accuracy and experimental uncertainty. *AIAA Journal* 1994; **32**(1): 3.
37. Metha UB. Some aspects of uncertainty in computational fluid dynamics results. *Transactions of the ASME* 1991; **113**: 538–543.
38. Lohner R. The efficient simulation of strongly unsteady flows by the finite element method. AIAA Paper No. 87-0555, 1987.
39. Widhopf GF, Buell JC, Schmidt EM. Time-dependent near field muzzle brake flow simulations. AIAA Paper No. 82-0973, 1982.

