ICAS-96-1.1.1

A CARTESIAN CUT CELL METHOD FOR UNSTEADY FLOWS INVOLVING FREELY MOVING BODIES

G. Yang, D. M. Causon, D. M. Ingram, R. Saunders
Centre for Mathematical Modelling and Flow Analysis
the Manchester Metropolitan University
Manchester M1 5GD
United Kingdom

Abstract. A method for the calculation of 3-D compressible unsteady flows involving freely moving bodies is presented. Based on a Cartesian cut cell mesh and a high resolution, upwind, finite volume scheme, this approach deals with moving body problems as follows: firstly, a background Cartesian mesh is generated on the domain, where the boundaries of solid bodies are represented by different types of cut cells; secondly, solid bodies are allowed to move across the stationary mesh by utilising a cell merging technique. The flow solver used is a MUSCL-Hancock Godunov-type scheme. A HLLC approximate Riemann solver is used to estimate the fluxes at fluid interfaces but an exact Riemann solution for a moving piston is incorporated on static or moving solid boundaries. Several examples are provided to demonstrate the capability of the method for the simulation of unsteady flows involving either static or freely moving bodies.

Introduction

Numerical simulation of unsteady, compressible flows involving rigid bodies in relative motion has been a computational challenge for many years. The main problem in predicting such flows arises from the fact that body motion will not only produce a highly complex time-dependent flow, but will also necessitate a change in the computational domain. As a result, additional complexities in the governing equations and the mesh system will be introduced. The major capabilities required for the accurate simulation of unsteady flows involving moving bodies are:

1. *Efficient mesh systems.* A mesh system is required to efficiently incorporate the movement of computational boundaries as the bodies move;

2. *High resolution numerical schemes.* Such schemes are needed to simulate accurately time-dependent flows with shock waves and other flow discontinuities.

It is not surprising therefore that only a few attempts have been made to model problems with moving boundaries/bodies. These applications have utilised either structured or unstructured mesh techniques. Unstructured mesh methods commonly use a global unstructured mesh and incorporate periodic local or global remeshing to account for the moving boundaries or bodies. Either finite element [1, 2] or finite volume solvers[3] are used for the discretisation. For structured mesh methods, on the other hand, one promising approach is the *Chimera* [4] or FAME[5] philosophy of overlapping several meshes, each of which is specific to an appropriate section of the geometry. This approach requires continuous identification of mesh intersection points from all overlapping meshes to transfer information between the various mesh patches but does not eliminate the need for body-fitted grids.

In this paper, we present an alternative approach for the prediction of unsteady compressible flows involving freely moving bodies. Based on a Cartesian cut cell mesh and a high resolution upwind finite volume scheme, this approach differs from other approaches mainly in that the mesh used is stationary and moving bodies are allowed to cross the meshlines. Hence, it requires no changes to the governing equations, no need to enforce geometric conservation laws nor remedies for skewness of the moving mesh or cells near moving boundaries. The rest of the paper is organised as follows: firstly, the generation of a suitable Cartesian cut cell mesh is presented; secondly, details of the numerical flow solver are described; thirdly, the extension to moving boundary or body problems is given, followed by some typical test problems; finally, in the final section, some conclusions are drawn.

Cartesian Cut Cell Mesh

A Cartesian cut cell mesh can be generated simply by 'cutting' solid bodies out of a background Cartesian mesh. Hence three types of cells are formed in

the computational domain: flow cells, cut cells and solid cells (see Figure 1). Furthermore, each cut cell is handled by replacing any number of intersecting planes within this cell by 7 approximating planes including 6 interfaces and one solid face. Since we utilise a finite volume technique to discretise the governing equations, the generation of a Cartesian cut cell mesh involves specifying the type of each cell and providing geometric information relating to the cut cells.

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

Solid Geometry Description

Body surfaces are described by triangulated surface facets obtained directly from a CAD package or by a surface triangulation procedure. Unlike the surface triangulation used in many unstructured mesh generation techniques, where the size of the triangle is fixed by the requirements of the flow solver, the surface facets in our Cartesian cut cell approach are chosen to be just large enough to accurately define the surfaces of the bodies. For example, a planar surface of a solid body needs only to be divided into two triangular facets. These triangular facets are stored in a set of vertices with connectivity lists so that the surface normals are oriented properly in the flow domain. These triangular facets are used only to find the intersections between the boundaries of solid bodies and the background Cartesian mesh.

Finding Intersections

Once the geometry of a solid body has been defined, the next step is to find the intersection points between the background Cartesian mesh and the boundary of the solid body. Since the geometry of the solid body is represented by a series of triangular facets, the problem is reduced to finding the intersection points between an individual Cartesian mesh line and an individual triangular facet. The intersection of a straight line with a plane can be found in a straightforward manner. First, two equations of both the plane and the straight line are described by a parametric formula. Unless they are parallel or coincident, the solutions for the parameters can be obtained exactly. These parameters are then checked to determine whether the intersection point is on the plane or its extended plane. Details of intersection problems involving a straight line and a flat plane

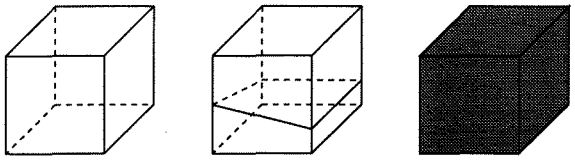

Figure 1: Flow cell, cut cell and solid cell

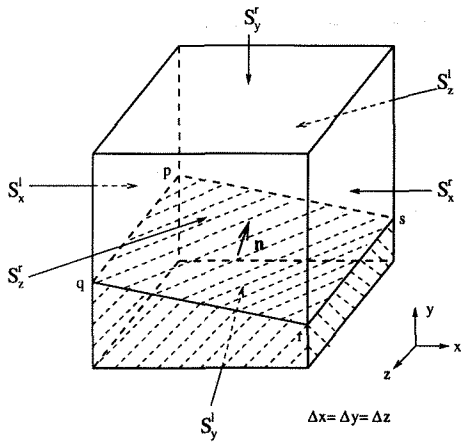

Figure 2: A cut cell in three dimensions

can be found in [6]. Once all the triangular facets have been checked, all the intersection points on an individual mesh line will have been found.

Usually, cut cells only occupy a small portion 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 the intersection points along an individual mesh line are obtained, they are registered in the cut cell lists. This process is repeated until all the mesh lines have been dealt with.

Determining the Cut Cell Information

Once the calculation of all intersections between the mesh lines and the geometry of the solid body has been completed, all geometric information concerning the cut cells will have been determined. From the finite volume point of view, these include the direction of the outward normal vector for the solid face, the area of every face, and the volume of each cut cell. We only calculate the area of 6 interfaces of each cut cell according to the intersection points on the cell edges. If an interface of a cut cell 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 2). Although the normal vector and area of the solid face are not explicitly included, they can be computed by taking the difference in the exposed areas

of opposing interfaces, that is:

$$|S| = \sqrt{(S_x^l - S_x^r)^2 + (S_y^l - S_y^r)^2 + (S_z^l - S_z^r)^2} \quad (1)$$

$$\mathbf{n} = (S_x^r - S_x^l, S_y^r - S_y^l, S_z^r - S_z^l)/|S|. \quad (2)$$

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.

Finding Merging Cells

If time accurate solutions are required, the stability constraint will be severe on arbitrarily small cut cells. In our calculations, a cell merging technique is used to ensure stability and we first need to determine which cells should be merged. By providing a minimum volume criterion V_{min} for a cut cell, finding a neighbouring cell to merge with depends on the slope of the solid face of the cut cell.

The Numerical Scheme

Governing Equations

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

$$\frac{d}{dt}\int_{V_t} \mathbf{U} dV + \oint_{S_t} \mathbf{F}\cdot\mathbf{n} dS = 0 \quad (3)$$

where $\mathbf{U}$ is the vector of conserved variables and $\mathbf{F}$ is the flux vector function, $\mathbf{n}$ 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

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} \rho(\mathbf{v}-\mathbf{v}_s) \\ \rho u(\mathbf{v}-\mathbf{v}_s) + p\mathbf{i} \\ \rho v(\mathbf{v}-\mathbf{v}_s) + p\mathbf{j} \\ \rho w(\mathbf{v}-\mathbf{v}_s) + p\mathbf{k} \\ (e+p)(\mathbf{v}-\mathbf{v}_s) + p\mathbf{v}_s \end{bmatrix} \quad (4)$$

where ρ, u, v, w, p and e are density, 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 $\mathbf{v}_s$ is the velocity of the boundary of the control volume V_t. In this paper, we use a stationary background Cartesian mesh to deal with moving boundary problems, therefore, $\mathbf{v}_s = 0$ on the flow interfaces of a cell but on a moving solid face of the cell, $\mathbf{v}_s$ is the velocity of the moving boundary. Finally, the governing equations are closed by the ideal gas equation of state,

$$p = (\gamma - 1)\left[e - \frac{\rho}{2}(u^2 + v^2 + w^2)\right]. \quad (5)$$

Numerical Discretisation

The flow solver used here is a MUSCL-Hancock [7] Godunov type scheme with appropriate modifications. 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 $\Delta t/2$,

$$(V\mathbf{U})_{ijk}^{n+\frac{1}{2}} = (V\mathbf{U})_{ijk}^{n} - \frac{\Delta t}{2}\sum_{l=1}^{M_l} \mathbf{F}(\mathbf{U}_l)\cdot\mathbf{S}_l^n \quad (6)$$

where M_l is maximum number of cell faces. For a flow (or uncut) cell, $M_l = 6$; for a cut cell, $M_l = 7$. The flux function $\mathbf{F}(\mathbf{U}_l)$ is evaluated at the midpoints of cell faces following a linear reconstruction of the flow solution within each cell, via,

$$\mathbf{U}_l = \mathbf{U}_{ijk}^n + \frac{1}{2}\mathbf{n}_l \cdot \nabla\mathbf{U}_{ijk}^n \quad (7)$$

where $\mathbf{n}_l$ is the normal unit vector of face l 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-hand states for a series of Riemann problems. The solution of these Riemann problems provide a set of upwind interface fluxes which are used to update the flow solution over the time interval Δt,

$$(V\mathbf{U})_{ijk}^{n+1} = (V\mathbf{U})_{ijk}^{n} - \Delta t\sum_{l=1}^{M_l} \mathbf{F}\left(\mathbf{U}_l^{L,R}\right)\cdot\mathbf{S}_l^{n+\frac{1}{2}} \quad (8)$$

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

$$\begin{cases} \mathbf{U}_l^L = \mathbf{U}_{ijk}^{n+\frac{1}{2}} + \frac{1}{2}\mathbf{n}_l^L \cdot \nabla\mathbf{U}_{ijk}^n \\ \mathbf{U}_l^R = \mathbf{U}_{n(ijk)}^{n+\frac{1}{2}} + \frac{1}{2}\mathbf{n}_l^R \cdot \nabla\mathbf{U}_{n(ijk)}^n \end{cases} \quad (9)$$

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

To solve the Riemann problem, either an exact Riemann solver or various approximate Riemann solvers can be used. Here, an HLLC [8] approximate Riemann solver is used at fluid interfaces. However, 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

$$\mathbf{F}_n^* \cdot \mathbf{S} = \begin{bmatrix} 0 \\ p_n^*(S_x^l - S_x^r) \\ p_n^*(S_y^l - S_y^r) \\ p_n^*(S_z^l - S_z^r) \\ p_n^*\mathbf{v}_{sn}|S| \end{bmatrix} \quad (10)$$

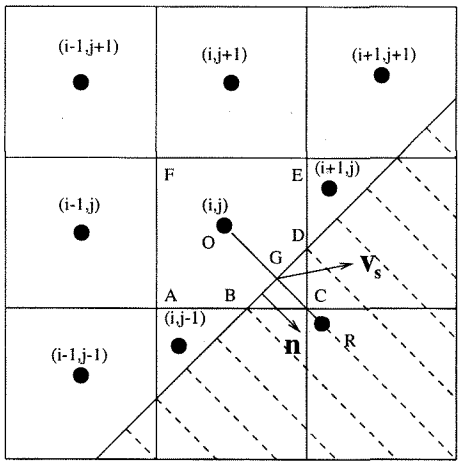

Figure 3: Calculation of gradients on a cut cell

where p_n^* and $\mathbf{v}_{sn}$ are the pressure and normal velocity arising from the Riemann solution on the moving solid boundary.

Since the MUSCL-Hancock scheme is conditionally stable, stability problems will occur in the presence of arbitrarily small cut cells if a time step based on the flow cell size is used. Here, a cell merging[9, 10] technique is implemented since it is not only simple but can also be extended to moving boundary problems.

Gradient Calculation

The gradient calculation is straightforward on flow cells away from a solid boundary. For cells near a solid boundary, a different approach is used. Since a solid boundary can be either static or moving, appropriate boundary conditions must be implemented in the gradient calculation. Considering a 2-dimensional case, if reflection boundary conditions are used on a solid boundary, the variables in fictional cell R (see Figure 3) can be obtained by

$$\begin{cases} \rho_R = \rho_{ij} \\ \mathbf{v}_R = \mathbf{v}_{ij} - 2\left(\mathbf{v}_{ij}\cdot\mathbf{n}\right)\mathbf{n} + 2\left(\mathbf{v}_s\cdot\mathbf{n}\right)\mathbf{n} \\ p_R = p_{ij} \\ e_R = p_R/(\gamma-1) + \frac{1}{2}\rho_R|\mathbf{v}_R|^2 \end{cases} \tag{11}$$

The gradients on cut cell (i,j) may be of two types: flow and solid in each mesh direction. Consider the gradients in x-direction, the flow gradient can be calculated as for flow cells,

$$\mathbf{U}_x^f = G\left(\mathbf{U}_{i+1,j} - \mathbf{U}_{i,j}, \mathbf{U}_{i,j} - \mathbf{U}_{i-1,j}\right) \tag{12}$$

where G is a slope limiter function which is used to prevent over- or under-shoots. The limiter function may take the following forms:

the Superbee limiter

$$G(a,b) = s\cdot max\left\{0, min(2|b|, s\cdot a), min(|b|, 2s\cdot a)\right\} \quad s = sign(b) \tag{13}$$

or the van Leer limiter

$$G(a,b) = \frac{|b| + |a|b/a}{|a| + |b|}. \tag{14}$$

Then, the solid gradient is estimated similarly, that is

$$\mathbf{U}_x^s = G\left(\mathbf{U}_R - \mathbf{U}_{i,j}, \mathbf{U}_{i,j} - \mathbf{U}_{i-1,j}\right) \tag{15}$$

Once the two types of gradients have been calculated, an area average technique is used to obtain unique gradients in the x-direction:

$$\mathbf{U}_x = (S_x^f\mathbf{U}_x^f + S_x^s\mathbf{U}_x^s)/S_{max}^x \tag{16}$$

$$S_{max}^x = Max(S_x^l, S_x^r) \tag{17}$$

$$S_x^s = |S_x^l - S_x^r| \tag{18}$$

$$S_x^f = S_{max}^x - S_x^s \tag{19}$$

with a similar calculation for the gradients in y-direction. A gradient vector in cut cell (i,j) can then be obtained as

$$\nabla\mathbf{U}_{ij} = \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix}, \tag{20}$$

and a reconstructed solution vector $\mathbf{U}(x,y)$ can be found anywhere within the cut cell from

$$\mathbf{U}(x,y) = \mathbf{U}_{ij} + \frac{1}{2}\mathbf{n}\cdot\nabla\mathbf{U}_{ij} \tag{21}$$

where $\mathbf{n}$ is the normal unit vector for any specific interface or solid boundary.

For 3-dimensional problems, similar techniques can be used to provide gradients in the z-direction.

Extension to Moving Body Problems

Description of Body Motion

Body motion can be either one dimensional or multi-dimensional. For one dimensional motion, a velocity for each data point on the surface of the body is prescribed so that the total body motion follows a prescribed trajectory. If each data point has a different velocity, the problem involves body movement as well as body deformation. Otherwise, the problem can be considered as rigid body motion. Therefore, it is relatively simple to describe one dimensional rigid body motion by prescribing a single velocity. For multi-dimensional problems, however, rigid body motion often 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 the rotation angle velocity ω_c for one point (e.g. the mass centre) in the body, and calculate values at other points on the body from $\mathbf{v}_c$ and ω_c. In practice, $\mathbf{v}_c$ and ω_c can be estimated from the pressure distribution acting on the surface of the moving body. In the following, we briefly describe how to determine the new position of the moving body and the velocity of any point on the body based on known values of $\mathbf{v}_c$ and ω_c.

See Figure 4, the position vector of point p on the surface of the solid body is given by

$$\mathbf{r}_p = \mathbf{r}_c + \mathbf{r}_o = \mathbf{r}_c + \mathbf{r}_\omega + \mathbf{r}_r \tag{22}$$

therefore, the velocity of point p is

$$\mathbf{v}_p = \mathbf{v}_c + \omega_c \times \mathbf{r}_o \tag{23}$$

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

$$\mathbf{e}_r = \frac{\mathbf{r}_r}{|\mathbf{r}_r|}, \quad \mathbf{e}_n = \frac{\omega_c}{|\omega_c|} \times \mathbf{e}_r \tag{24}$$

Given the time step Δt, the new position and velocity of point p (see Figure 5) can be obtained from

$$\Delta\phi = \omega_c \Delta t \tag{25}$$

$$\mathbf{r}_o^{n+1} = \mathbf{r}_\omega^n + |\mathbf{r}_r^n|[\cos(\Delta\phi)\mathbf{e}_r + \sin(\Delta\phi)\mathbf{e}_n] \tag{26}$$

$$\mathbf{r}_c^{n+1} = \mathbf{r}_c^n + \mathbf{v}_c \Delta t \tag{27}$$

$$\mathbf{r}_p^{n+1} = \mathbf{r}_c^{n+1} + \mathbf{r}_o^{n+1} \tag{28}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_c + \omega_c \times \mathbf{r}_o^{n+1} \tag{29}$$

Once all points on the body periphery have been updated, the new position of the moving body is known.

Cell Merging on Moving Boundaries

In the Cartesian cut cell approach, any solid bodies are simply cut out of a background Cartesian mesh, where the boundaries of the solid bodies are represented by different cut cells. Once the bodies move, the cut cell information undergoes changes. In essence, these changes can be divided into four categories:

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 within the finite volume scheme. However, where a cut cell becomes solid (category 1), the volume of the cell at the end of time step is zero; obviously this will lead to problems within the flow solver. For category 2, although the cell finally becomes a flow

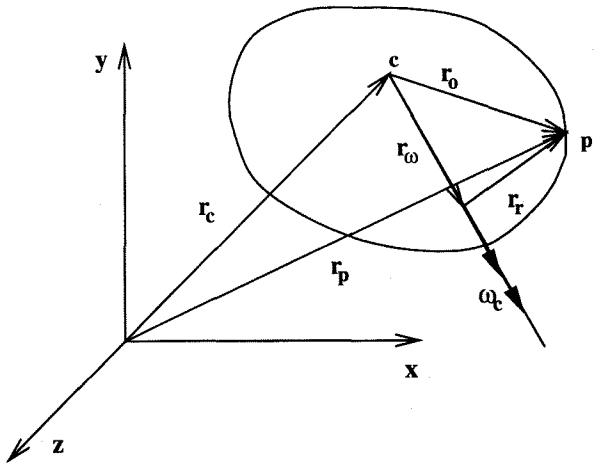

Figure 4: The position vector of a point p on the solid body

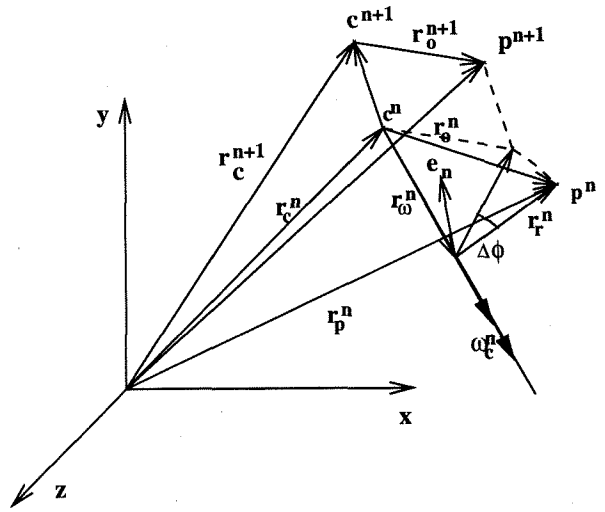

Figure 5: Finding the new position of point p

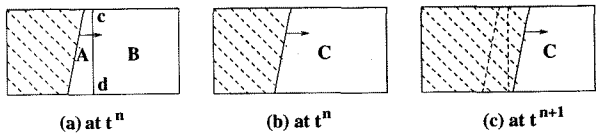

Figure 6: Cell merging on a moving boundary

cell, failure to consider the new-born cell will result in strict conservation being lost. In general, all these problems can be solved by using a cell merging technique [9, 10]. The basic idea is to merge a small cut cell with one or several neighbouring cells so that any interface between the merged cells is ignored and the waves are allowed to travel in the larger merged cell without reducing the global value of Δt.

Considering a two-dimensional example, a time step, Δt, based on flow cell B, will be too large for cut cell A which will become solid after Δt (see Figure 6). To merge the two cells, we first compute the updates at cells A and B as usual,

$$\Delta(V\mathbf{U})_{A,B} = -\Delta t \sum_{l=1}^{M_{A,B}} \mathbf{F}_l \cdot \mathbf{S}_l \qquad (30)$$

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

$$\Delta(V\mathbf{U})_C = \Delta(V\mathbf{U})_A + \Delta(V\mathbf{U})_B \qquad (31)$$

The fluxes on the interface $|cd|$ between cells A and B cancel out automatically since the flux calculation is conservative. The conserved variable $\mathbf{U}$ for cell C at time t^{n+1} is

$$\begin{aligned}(V\mathbf{U})_C^{n+1} &= (V\mathbf{U})_A^n + (V\mathbf{U})_B^n \\ &- \Delta t \left(\sum_{l=1}^{M_A} \mathbf{F}_l \cdot \mathbf{S}_l + \sum_{l=1}^{M_B} \mathbf{F}_l \cdot \mathbf{S}_l \right) \qquad (32)\end{aligned}$$

Although the cut cell A finally vanishes, its contribution to the mass, momentum and energy, will be transfered into neighbouring cells so that conservation is automatically maintained. The process of cell merging may reduce the integration accuracy at solid boundaries; however, it has been shown that this need not affect the global accuracy of the calculation[11] and has caused no noticeable detrimental effect in practice.

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

$$\Delta t_{x,y,z} = \frac{\sqrt[3]{V_{min}}}{max(|\mathbf{v}_s|_{x,y,z}, |\mathbf{v}|_{x,y,z}) + a}, \qquad (33)$$

$$\Delta t = C_{FL}\ min(\Delta t_x, \Delta t_y, \Delta t_z) \quad (C_{FL} < 1.0) \qquad (34)$$

where a is sound speed.

Numerical Results

2-D Wedge Channel Flow

The first example concerns an inviscid channel flow at $M_\infty = 2$ past a fixed wedge. The channel is 1 unit high and 6.5 units long. The wedge is located from 4.25 to 6.25 units with 0.26795 unit height so that a 15° compression corner and a 15° expansion corner are formed by the wedge. Figure 7 shows the geometry of the wedge and channel, where a uniform Cartesian mesh of 260 × 40 cells was used. A free stream flow at Mach number $M_\infty = 2$ is assumed to pass through the channel from right to left, and an attached oblique shock is produced at the compression corner. The oblique shock is then reflected at the upper wall of the channel. Meanwhile, a strong expansion fan is created at the top rear corner of the base region. The expansion fan interacts with the lower wall of the channel so that an oblique shock appears in the base region. Figure 8 shows density contours at two different times $t = 1$ and 2.

We then specify an impulsively started wedge moving at Mach 2 into quiescent gas, expecting an identical flow field. In this case, the wedge is positioned initially from 0.25 to 2.25 units (see Figure 9). As in the fixed wedge case, an attached bow shock and an oblique shock in the base region are gradually produced and then reflected at the upper wall. At time $t = 2$, the wedge has moved to exactly the same position as in the fixed wedge case. Figure 10 shows density contours at two time stages $t = 1$ and 2 respectively. Comparing the two cases, we can see there is little difference between a wedge moving at $M_w = 2$ into quiescent gas and fixed wedge placed in a Mach 2 free stream. Identical flow features are created by the Cartesian cut cell method with fixed or moving boundaries.

3-D Store Release Prediction

Store release from aircraft has been a computational challenge since the beginning of flying. Consider the case corresponding to supersonic flight. During store separation, many complex physical flow features are produced. These features include shock/shock interactions, shock/boundary layer interactions, turbulent separated flow and relative body motion. Given the complexity of these physical flow features, our efforts are concentrated initially on the solution of the Euler equations.

The problem considered here 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 with $100 \times 62 \times 30$ cells was used on a domain of $200 \times 124 \times 60$ units. The translational and rotational velocities for the store centre of mass were $\mathbf{v}_c = [0.0, -0.5]^T$ and $\omega_c = 0.00023$,

respectively (non-dimensionalised with respect the free-stream speed of sound and the diameter of the store). Once the store has been ejected, the resulting flow-fields are shown at different times. The position of the store at each time is determined by the method described above. Figures 11 to 13 show the positions of the store, the Cartesian mesh and density contours at two different times.

Figure 12 shows the computed flowfields 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 13). 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.

Conclusions

A Cartesian cut cell method for the computation of 2 or 3-dimensional unsteady compressible flows involving both static and freely moving bodies has been presented. In this approach, a stationary background Cartesian mesh is used and moving bodies are allowed to arbitrarily cross the mesh-lines; consequently, only cells near solid boundaries need special treatment as the bodies move. The main advantage of this approach is that it can deal with moving body problems without a moving mesh, hence problems such as mesh distortion, body motion restriction etc. which occur when using other mesh approaches are avoided completely.

The Cartesian cut cell method has been validated against known unsteady flows and subsequently applied to a sample problem of a store release into a Mach 1.5 stream. The computational results indicate good potential for the Cartesian cut cell method for practical applications involving moving body problems.

Since the present approach is based on a stationary background Cartesian mesh for dealing with moving boundary/body problems, adaptive mesh refinement techniques can easily be implemented. With suitable mesh refinement it should be possible to begin the task of extending the inviscid method to viscous flows.

References

[1] R. Lohner. Adaptive Remeshing for Transient Problems. *Comp. Methods in Appl. Mech. and Eng.*, 75:195–214, 1989.

[2] E. J. Prodert, O. Hassan, K. Morgan, and J. Peraire. An Adaptive Finite Element Method for Transient Compressible Flows with Moving Boundaries. *Int. J. Numer. Meth. Eng.*, 32(4):751–765, 1991.

[3] J. Y. Trepanier, M. Reggio, M. Paraschivoiu, and R. Camarero. Unsteady Euler Solutions for Arbitrarily Moving Bodies and Boundaries. *AIAA J.*, 31(10):1869–1876, 1993.

[4] J. L. Steger, F. C. Dougherty, and J. A. Benek. A Chimera Grid Scheme. In *Advances in Grid Generation, ASME FED-5*, pages 59–69, 1983.

[5] T. A. Blaylock and S. H. Onslow. Application of the FAME Method to Store Release Prediction. In *Computational Fluid Dynamics '94, Stuttgart, Germany*, 1994.

[6] B. R. Dewey. *Computer Graphics for Engineers.* Harper & Row, 1988.

[7] B. van Leer. On the Relation Between the Upwind-Differencing Schemes of Godunov, Engquist-Osher and Roe. *SIAM Journal on Scientific and Statistical Computing*, 5, 1984.

[8] E. F. Toro, M. Spruce, and W. Speares. Restoration of the Contact Surface in the HLL Riemann Solver. *Shock Waves, Vol 4*, pages 25–34, 1994.

[9] Y. Chiang, B. Van Leer, and K. G. Powell. Simulation of Unsteady Inviscid Flow on an Adaptively Refined Cartesian Grid. In *30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, 1992.

[10] D. K. Clarke, M. D. Salas, and H. A. Hassan. Euler Calculations for Multielement Airfoils Using Cartesian Grids. *AIAA Journal Vol.24, No.3*, 1986.

[11] W. J. Coirier and K. G. Powell. An Accuracy Assessment of Cartesian Mesh Approaches for the Euler Equations. *AIAA Paper 93-3335-CP*, 1993.

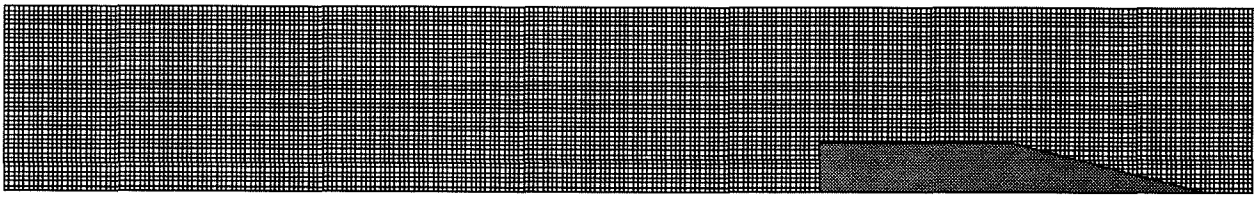

Figure 7: Cartesian mesh for fixed wedge

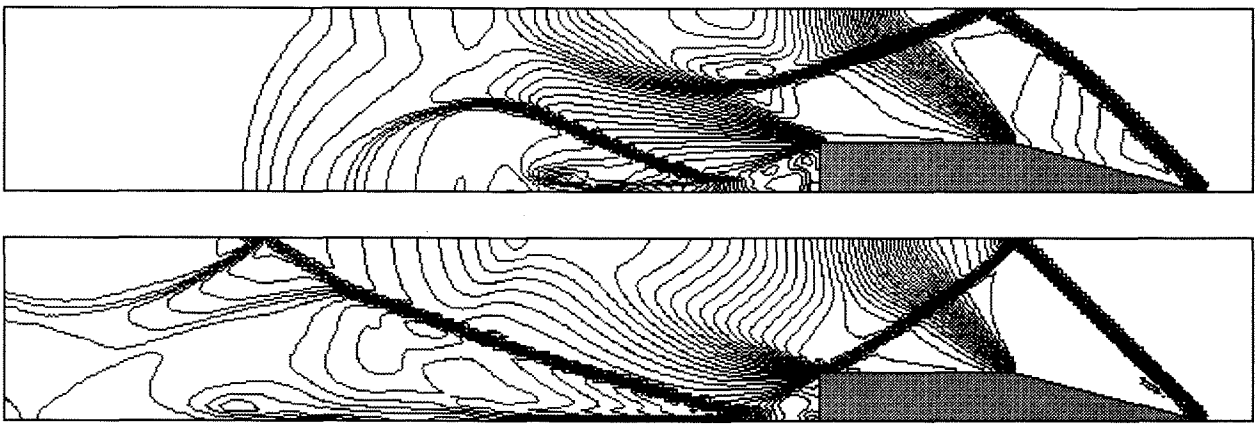

Figure 8: Density contours at $t = 1$ and 2 for fixed wedge in Mach 2 flow

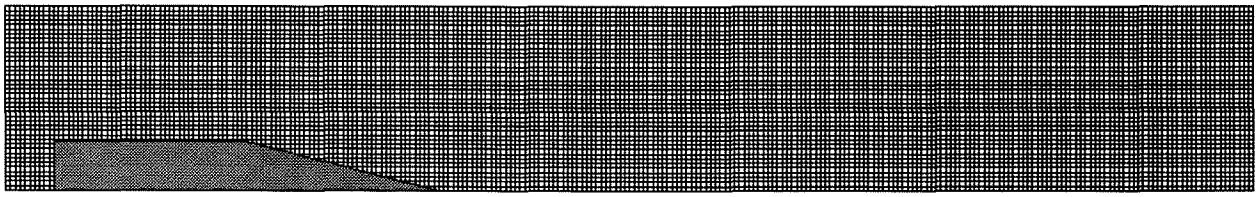

Figure 9: Cartesian mesh for moving wedge at $t = 0$

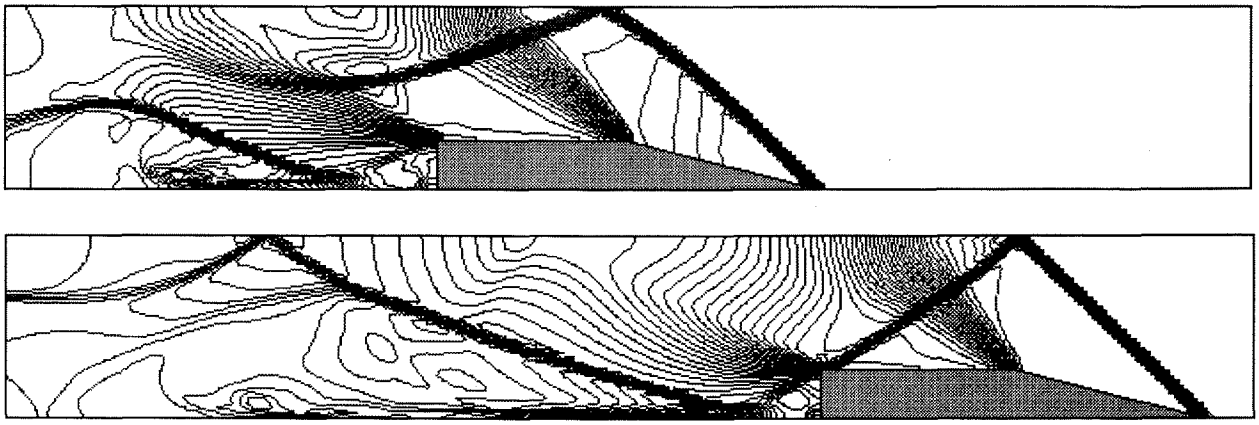

Figure 10: Density contours at $t = 1$ and 2 for wedge moving at Mach 2

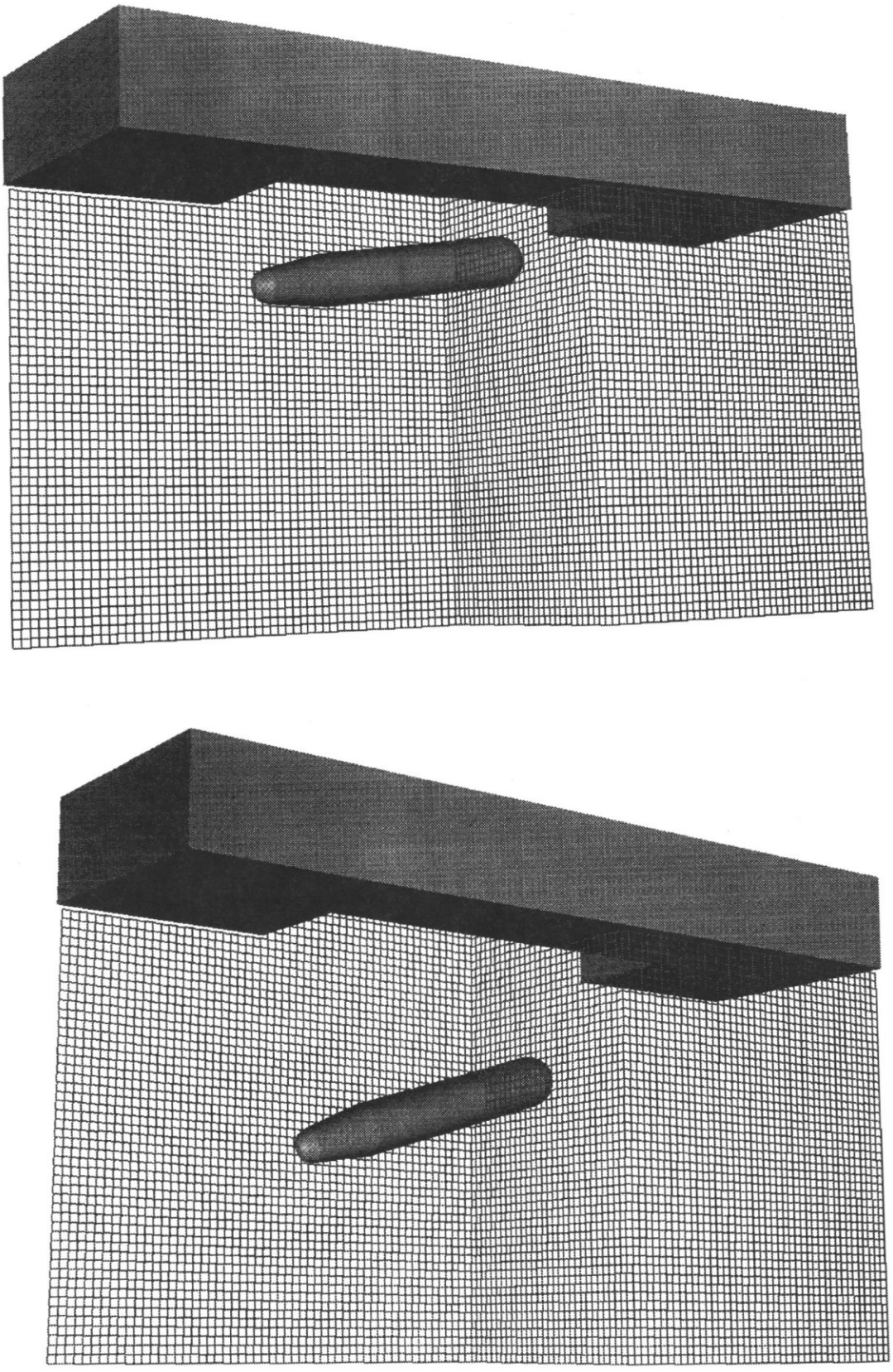

Figure 11: Cartesian mesh and the store positions at $t = 50$ and 100

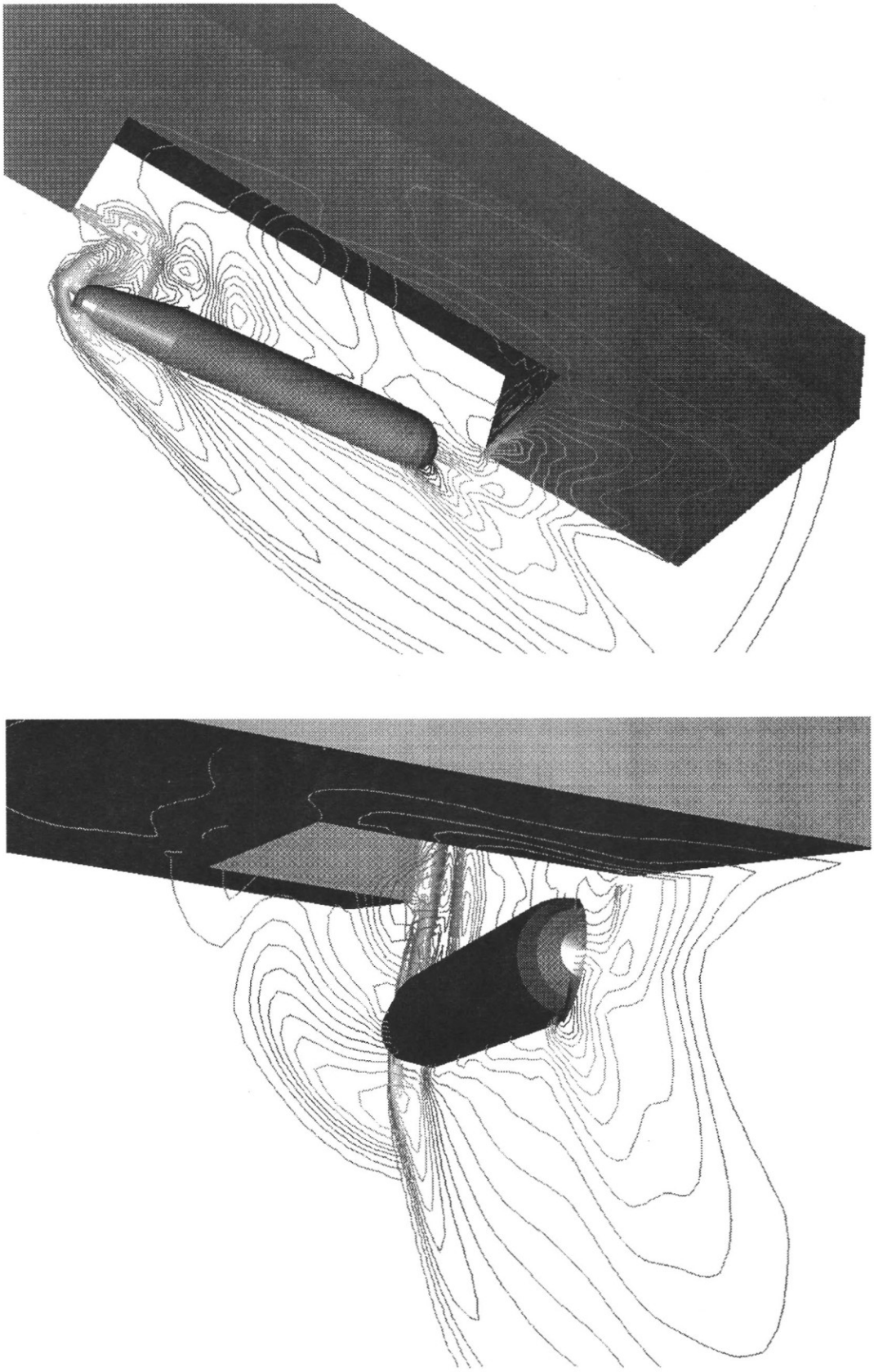

Figure 12: Density contours at $t = 50$

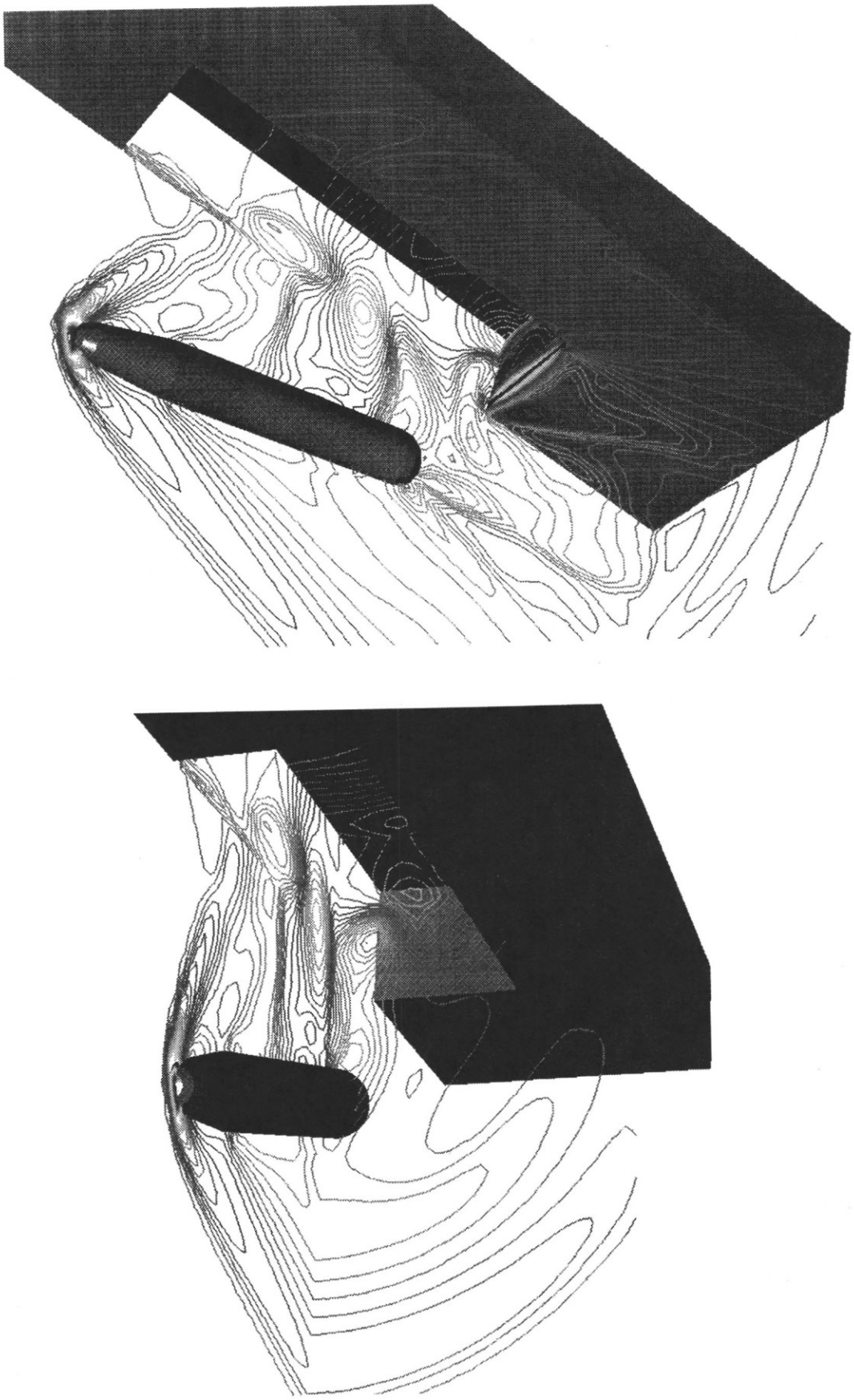

Figure 13: Density contours at $t = 100$