INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **41**:1053–1084 (DOI: 10.1002/fld.479)

Anisotropic Cartesian grid method for steady inviscid shocked flow computation

Zi-Niu Wu* and Ke Li

Department of Engineering Mechanics, Tsinghua University, Beijing 100084, Peoples Republic of China

SUMMARY

The anisotropic Cartesian grid method, initially developed by Z.N. Wu (ICNMFD 15, 1996; CFD Review 1998, pp. 93–113) several years ago for efficiently capturing the anisotropic nature of a viscous boundary layer, is applied here to steady shocked flow computation. A finite-difference method is proposed for treating the slip wall conditions. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: anisotropic Cartesian grid; inviscid flow; shock wave; boundary condition

1. INTRODUCTION

There are three-well known methods for treating complex geometries: unstructured grid methods, structured multidomain methods, and adaptive Cartesian grid methods. Unstructured grid methods are very flexible in handling complex geometries. Structured multidomain methods are supposed to be as accurate as traditional structured grid method, and flexible for treating complex geometries, though they face a large theoretical problems such as stability, convergence, uniqueness, and conservation, see Reference [1] for an overview.

Cartesian grid method has been demonstrated to be a useful tool for computing flows with complex geometries. The Cartesian grid method has been successfully used to perform external flow computation with arbitrarily shaped solid walls, see for instance References [2–10] for inviscid flow computation and [11–13] for viscous flow computations. More recently, the Cartesian grid method has been used to compute three dimensional flows around complex geometries [14–17]. A Cartesian grid involves simple cells in interior regions and cutcells near solid walls. This method requires a mesh refinement strategy [2, 5, 10, 18, 19] to resolve the geometry and/or the flow and a special treatment of the solid boundary conditions [2, 4, 9, 10, 20].

* Correspondence to: Z.-N. Wu, Department of Engineering Mechanics, Tsinghua University, Beijing 100084, People's Republic of China.

Contract/grant sponsor: Chinese National Natural Science Foundation; contract/grant number: 10025210
Contract/grant sponsor: China NKBRSF project; contract/grant number: 2001CB409600

Received January 2002
Revised 16 October 2002

Traditional grid generation starts with a background Cartesian grid with squared cells. The final grid is generated by the recursive subdivision of a single cell (parent) into four squared cells (children) when it is necessary. The hierarchical relation between the children and parents is stored and used for cell neighbour searching. The resulting grid thus involves cells of various sizes. Since a cell is refined equally in both directions independently of the geometry and the flow gradient, the resulting Cartesian grid is isotropic. As pointed out by Powell in his review paper [22], the isotropic Cartesian grid approach unavoidably leads to impractical number of cells when the flow is of anisotropic nature. Most Cartesian grid method uses a finite-volume approach to define solid boundary condition by solving a conservative difference equation on each polygone resulting from cell cutting. Cell cutting is expensive in general. The finite-volume approach may lead to instability for small cutcells. One either merges small cutcells to form large ones [6, 10] or modifies the scheme near the wall [19] to maintain the stability. This unavoidably reduces the accuracy or complicates the interior scheme. The advantage of the finite-volume approach is that it ensures conservation near the wall. A conceptually rather different method was presented in Reference [21] where a hybrid Cartesian/curvilinear grid is applied. The grid is Cartesian in interior regions and is structured cuvilinear near the solid wall. This ensures a good precision both at interior regions and near the wall. Difference equations on the two grids can be matched in a stable way. Such a method is further developed in Reference [20] where the Cartesian grid in interior region is matched to a body-fitted-structured grid through a thin unstructured grid.

In order to make the Cartesian grid method more efficient, Wu [13, 23] proposed an anisotropic Cartesian grid method and a finite-difference solid wall treatment for viscous flow computation. A similar but slightly different method is proposed in Reference [12] for solid wall condition. The anisotropic method is expected to reduce the number of cells without lossing the accuracy in comparison with the isotropic method. Note that anisotropic mesh methods have been well developed for unstructured grid method, see for instance Reference [24], where general principles have been given for user-independent, mesh-independent, and solver-independent CFD purpose.

In this paper we extend the anisotropic Cartesian grid method and the finite difference wall treatment to inviscid flow computation, notably with *shock waves*.

The anisotropic method divides a cell into subcells independently in each direction. In References [13, 23], only geometry-oriented refinement is considered. Near the wall with small curvature, the cells are essentially refined in the normal direction. When there is a great curvature, the cells are refined in both directions. The anisotropic method proposed in References [13, 23] not only provides an efficient resolution of the geometry but also fits to the anisotropic nature of the viscous boundary layer near the body where the flow gradient is essentially in the direction normal to the wall. In Figures 1 and 2, we display an anisotropic Cartesian grid and an isotropic Cartesian grid. The anisotropic grid is obviously grid saving, while resolving the boundary layer. The problem considered in this paper involves shock waves. The anisotropic nature of shock waves is more pronounced than in viscous flow computations. Near an essentially vertical or horizontal shock, the use of an anisotropic grid will substantially reduce the number of grid points since a shock only needs to be refined in the normal direction.

The finite-difference solid wall condition is based on the slip condition on the wall and the known solution at several interior points. These solutions are then interpolated to the centre

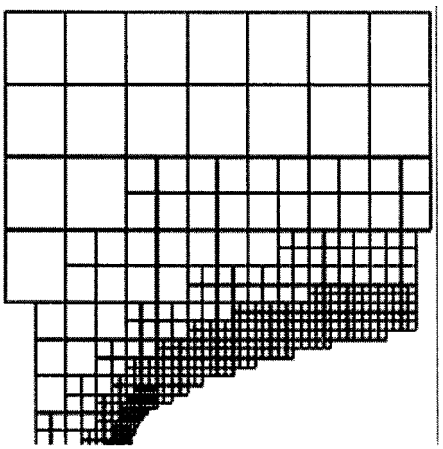

Figure 1. Isotropic Cartesian grid near a part of NACA0012 airfoil.

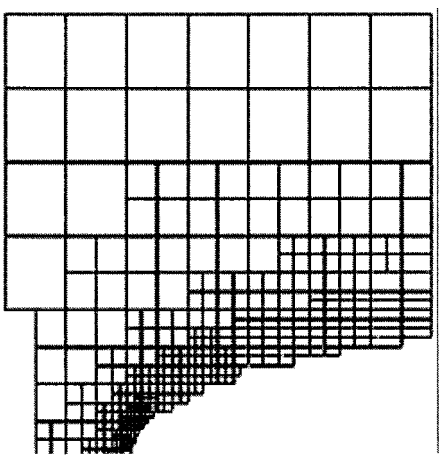

Figure 2. Anisotropic Cartesian grid near a part of NACA0012 airfoil.

of the boundary cell. The finite-difference approach does not require the boundary cells to be really cut.

This paper will be presented as follows. In Section 2, the geometry-oriented anisotropic Cartesian grid method developed in References [13, 23] is extended here to shocked flow problems.

Section 3 is devoted to boundary treatment near solid walls. We use bilinear interpolation to construct an easily implementable second-order accurate solid wall condition.

The anisotropic method along with the solid wall condition is finally validated by computing shocked flows in Section 4. The standard Roe scheme will be used.

Concluding remarks will be given in the end of this paper, where some discussions on stability and accuracy will be given.

2. ANISOTROPIC GRID GENERATION NEAR SHOCK WAVES

A key feature of the Cartesian grid is that it starts from a base grid and then a refinement algorithm is used to generate the final grid. The present paper is only concerned with steady flows. The refinement algorithm works for steady flows. But this algorithm could be straightforwardly extended to unsteady flows in the following way: at each time step, use the grid of the previous time step as a base grid, and repeat the subsequent steps as involved in the steady state approach.

2.1. Basic steps

A subgrid of level l with $l = 0, 1, \ldots, L$ refers to all the cells having the same mesh size h_l. The mesh size satisfies the relation $h_l = r^l h_0$ where $r = \frac{1}{2}$ is the the refinement ratio and h_0 is the mesh size on the coarse grid.

Algorithms for isotropic Cartesian grid generation can be found in References [6, 10]. The algorithm for anisotropic Cartesian grid generation was proposed by Wu [13, 23] with the primary concern of geometry-oriented refinement. Here we extend the anisotropic algorithm to inviscid flows with shock waves.

For convenience, cells are divided into three classes: fluid cells, boundary cells, and solid cells. Fluid cells are cells which are completely inside the flow field. Boundary cells are cells having an intersection with the boundary. Solid cells are cells which completely lie inside the solid.

The present grid generation/solution algorithm is divided into 6 steps:

(1) construction of the base grid, using the geometry-oriented anisotropic approach as developed in References [13, 23] and which is composed of five steps: (a) construction of the based grid (level $l = 0$), (b) determination of the refinement criterion, (c) refinement of each cell when necessary, (d) location of boundary cells (cutcells), (e) suppress of solid cells.
(2) compute the numerical solution using the Roe scheme as described in Section 4.
(3) determine the shock lines.
(4) refine the grids near the shock lines.
(5) recompute the flow using the refined grids.
(6) if obvious difference of solution is observed, go to step 3. Otherwise stop.

2.2. Refinement requirement near shock lines

The derivative of the solution along the normal of a shock wave is infinite. Hence it is impossible to construct a gradient-based refinement criterion. Otherwise the mesh size would be infinitely small. Since the refinement is dominated by the resolution of shock waves, we can simply fix the maximum refinement levels in the normal direction of the shock wave.

In the following, we will consider the refinement near shock waves, curved solid walls, and contact discontinuities in a uniform way.

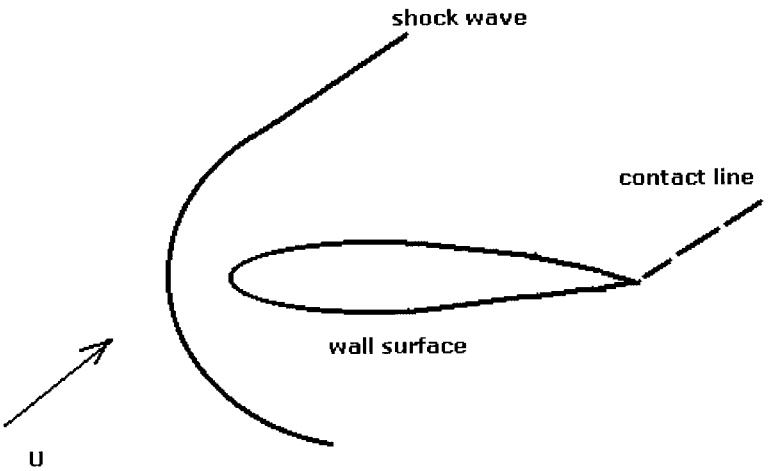

Figure 3. Refined lines: curved wall surfaces, shock lines, and contact discontinuity lines.

In the isotropic algorithm, the refinement criterion is independent of the direction. In the anisotropic one, the refinement depends on the direction so that the refinement criterion, expressed as the refinement ratio, should be a vector. Let $\mathbf{R}(\mathbf{x}) = (R^{(x)}(\mathbf{x}), R^{(y)}(\mathbf{x}))$ be the local refinement ratio at $\mathbf{x}$, then the mesh sizes at $\mathbf{x}$ are defined by

$$h^{(x)} = \frac{1}{l_x} h_0^{(x)}, \quad h^{(y)} = \frac{1}{l_y} h_0^{(y)}$$

where

$$l_x = \left\lceil \frac{1}{R^{(x)}} \right\rceil, \quad l_y = \left\lceil \frac{1}{R^{(y)}} \right\rceil$$

where $\lceil \bullet \rceil$ denotes the integer part of its argument.

The refinement of cells is determined by 'refined lines'. The refined lines include curved wall surfaces, shock lines, and contact discontinuity lines (Figure 3). Assume that there are I refined lines C_i, $i = 1, 2, \ldots, I$. Consider the line C_i which satisfies the equation

$$C_i(x, y, t) = 0$$

At each point $\mathbf{x}_{c_i}$ on C_i is defined a refinement criterion $\mathbf{R}(\mathbf{x}_{c_i}) = (R^{(x)}(\mathbf{x}_{c_i}), R^{(y)}(\mathbf{x}_{c_i}))$. Here $R^{(x)}$ is the refinement ratio in x and $R^{(y)}$ is the refinement ratio in y.

If C_i is a solid wall, then $\mathbf{R}(\mathbf{x}_{c_i})$ can be related to the local curvature $c(\mathbf{x}_{c_i})$. See Reference [23] for details of the computation of the local curvature. Let $l_{\min}$, $l_{\max}$ and l_{iso} be given three integers. The integer $l_{\min}$ is the minimal number of refinements on the wall and $l_{\max}$ is the maximal number of refinements on the wall. The integer l_{iso} is the number of anisotropic refinements counting from the finest subgrid. Let $c_{\min}$ and $c_{\max}$ be the minimum and maximum local curvatures on the wall. At any point $\mathbf{x}_{c_i}$ on the wall, define the local

relative curvature $\bar{c}(\mathbf{x}_{c_i})$ by

$$\bar{c}(\mathbf{x}_{c_i}) = \frac{c(\mathbf{x}_{c_i}) - c_{\min}}{c_{\max} - c_{\min}} \tag{1}$$

Then the number of refinements in the directions normal and tangent to the wall are computed as:

$$l_{\mathrm{n}} = (1 - \bar{c}) l_{\min} + \bar{c} l_{\max}, \quad l_{\mathrm{t}} = l_{\mathrm{n}} - l_{\mathrm{iso}}.$$

which means that the cell is 'refined' l_{n} times in the direction normal to the wall and 'refined' l_{t} times in the direction tangent to the wall.

If C_i is a shock wave or a slip line, then $\mathbf{R}(\mathbf{x}_{c_i})$ can be related to the local gradient. However, the local gradient on a shock line or slip line is infinite. Let l_{n} and l_{iso} be given two integers. This means the cell is 'refined' l_{n} times in the direction normal to the refined line and 'refined' $l_{\mathrm{t}} = l_{\mathrm{n}} - l_{\mathrm{iso}}$ times in the direction tangent to the refined line. The integer l_{iso} is the number of anisotropic refinements counting from the finest subgrid. In practice, we have to specify l_{n} according to the resolution requirement of shock waves.

Finally, l_x and l_y are computed by

$$l_x = \sqrt{(l_{\mathrm{n}} n_x)^2 + (l_{\mathrm{t}} n_y)^2}, \quad l_y = \sqrt{(l_{\mathrm{n}} n_y)^2 + (l_{\mathrm{t}} n_x)^2}$$

where (n_x, n_y) is the local unit normal on the wall. Finally, $\mathbf{R}(\mathbf{x}_{c_i})$ is computed as

$$\mathbf{R}(\mathbf{x}_{c_i}) = \left(\frac{1}{l_x}, \frac{1}{l_y} \right) \tag{2}$$

2.3. *Refinement criterion in interior points*

The grid refinement in interior regions originates from the refinement near the refined lines. The refinement criterion should be determined by requiring the cells to vary gradually from the refined lines to the other points. The simplest choice is to ensure approximatively a constant subgrid width in terms of grid points in the direction $\mathbf{n}(\mathbf{x}_{c_i})$.

Let $d_j = |\mathbf{x}_j - \mathbf{x}_{c_i}|$ and let $d_\infty = |\mathbf{x}_\infty - \mathbf{x}_{c_i}|$ where $\mathbf{x}_\infty$ is the point on the far flow field boundary closest to $\mathbf{x}_{c_i}$. If the local refinement ratio at $\mathbf{x}_j$ is a linear function of the distance d_j, i.e.

$$\mathbf{R}(\mathbf{x}_j) = \alpha + (1 - \alpha)\mathbf{R}(\mathbf{x}_{\mathrm{w}}), \quad \alpha = \frac{d_j}{d_\infty} \tag{3}$$

then the subgrid width in the direction $\mathbf{n}(\mathbf{x}_{c_i})$ is approximately constant.

When there are several refined lines, the refinement ratio at an interior point $\mathbf{x}_i$ computed from one refined line may differ from that from another refined line. Assume there are I separated refined lines on which the refinement ratios are $\mathbf{R}^{(c_i)}(\mathbf{x}_{c_i}) = (R_x^{(c_i)}, R_y^{(c_i)})$. Then $\mathbf{R}(\mathbf{x}_j)$ is computed as

$$\mathbf{R}(\mathbf{x}_j) = (R^{(x)}, R^{(y)}) \begin{cases} R^{(x)} = \min_{1 \leqslant i \leqslant I} [R_x^{(c_i)}] \\ R^{(y)} = \min_{1 \leqslant i \leqslant I} [R_y^{(c_i)}] \end{cases} \tag{4}$$

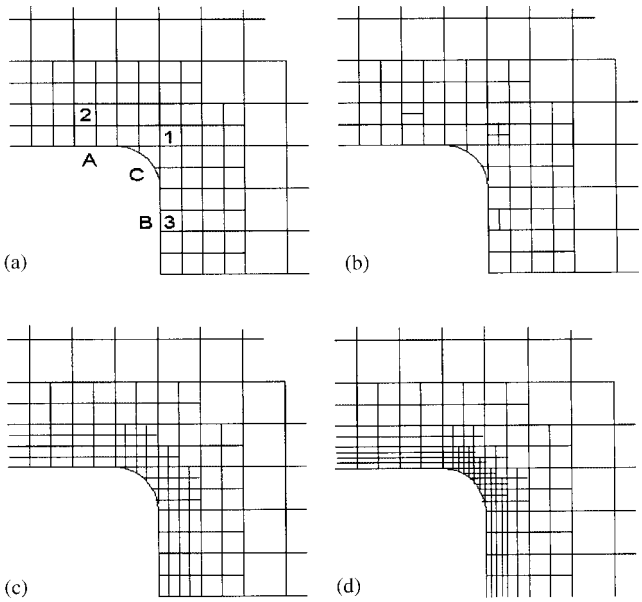

Figure 4. Anisotropic refinement near a refinement line. (a) grid after 2 isotropic refinements. (b) isotropic refinement for cell 1 and anisotropic refinement for cells 2 and 3. (c) grid after 2 isotropic refinements and 1 anisotropic refinement. (d) grid after 2 isotropic refinements and 2 anisotropic refinements.

2.4. *Anisotropic refinement*

Let L be the level of the finest grid (normally near shock waves). Then from $\min(R_x, R_y) = r^L$ we obtain $L = \ln \min(R_x, R_y)/\ln r$. Here $r = \frac{1}{2}$. The refinement starts from grid level $l = 0$. At each grid level l, we compute $\mathbf{R}(\mathbf{x}_j)$ for each cell where $\mathbf{x}_c$ is the cell centre.

Compute

$$l_x = \left\lceil \frac{\ln R_x(\mathbf{x}_j)}{\ln r} \right\rceil, \quad l_y = \left\lceil \frac{\ln R_y(\mathbf{x}_j)}{\ln r} \right\rceil \tag{5}$$

where $[\cdot]$ denotes the integer part of a real number.

If $l < l_x$ and $l < l_y$, then the current cell is divided into four squared cells. For example, this occurs for cell 1 in Figure 4(a), 4(b).

If $l \geqslant l_x$ and $l < l_y$, then the current cell is divided into two rectangle cells in the vertical direction. For example, this occurs for cell 2 in Figure 4(a), 4(b).

If $l < l_x$ and $l \geqslant l_y$, then the current cell is divided into two rectangle cells in the horizontal direction. For example, this occurs for cell 3 in Figure 4(a), 4(b).

If $l \geqslant l_x$ and $l \geqslant l_y$, then the current cell will not be divided.

Let us consider the refinement for the geometry in Figure 4. At level $l = l_{\text{iso}}$, the grid is still isotropic (Figure 4(a)). After two steps of anisotropic refinement, the anisotropic grids are respectively given by Figure 4(c) and Figure 4(d).

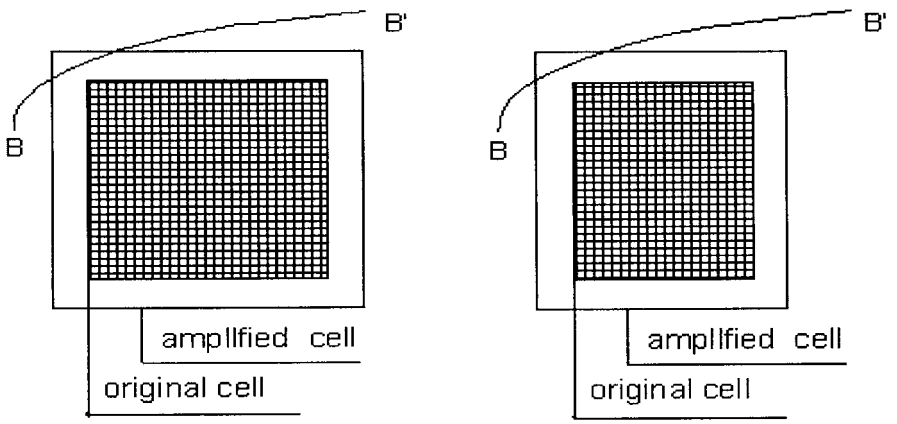

Figure 5. Definition of amplified cells.

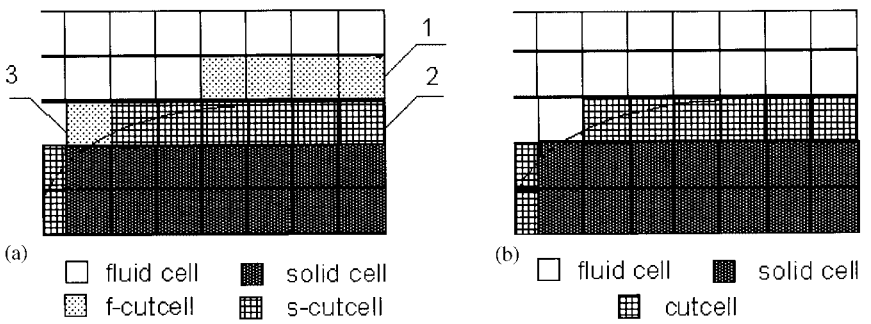

Figure 6. Generation and transformation of cutcells.

2.5. *Search of boundary cells (cutcells) and solid cells*

This has been described in detail in Reference [23]. Here we repeat it for completeness.

Boundary cells. The boundary condition that will be studied uses a finite-difference approach so that we do not need to find the intersection between the wall and the cutcells. In order the search to be quite efficient, a cell (rectangle or square) will be considered to be a cutcell once the nearest point on the wall lies inside the $(1+\eta)$-amplified cell. The $(1+\eta)$-amplified cell has the same shape and the same centre as the original cell except that its width and length are amplified by $(1+\eta)$. Consider the cells displayed in Figure 5. The original cells have no intersection with the wall BB' but the $(1+\eta)$-amplified cells have an intersection with the wall. The parameter η is a small positive number. For example, we can take $\eta = 0.1$. Such a search leads to cutcells of two types: genuine cutcells and pseudo cutcells. A genuine cutcell has intersection with the wall while a pseudo cutcell is disjoint from the wall and its amplified cell intersects the wall. The cutcells can be further classified into f-cutcells and s-cutcells. The f-cutcells have fluid cells as neighbours but have no solid neighbours. The s-cutcells have solid cells as neighbours. See Figure 6(a).

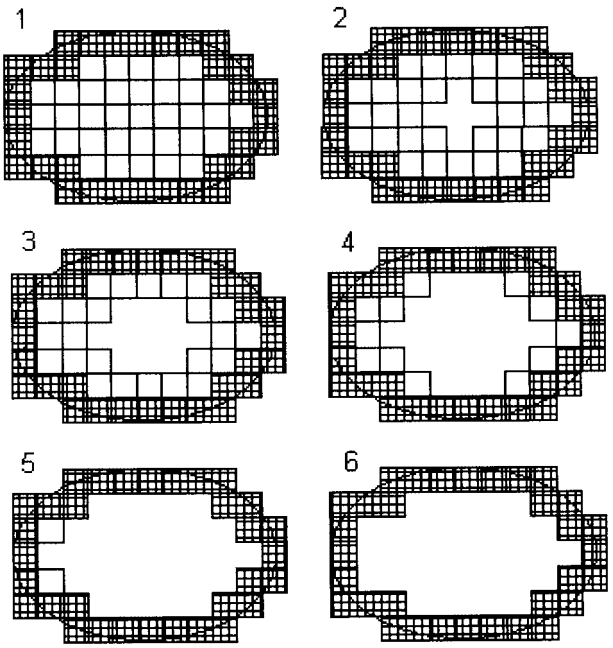

Figure 7. Search of solid cells.

After all the cutcells and solid cells are searched (see next section), only the s-cutcells are kept and the f-cutcells are transformed into fluid cells. See Figure 6(b).

Such a method, which broadens the search and keeps the s-cutcells, is extremely efficient. This treats without ambiguity the case of cell 1 and cell 2 displayed in Figure 6(a). The wall lies at the interface of cells 1 and 2. A different algorithm would face the difficulty of deciding which of cells 1 and 2 intersects with the wall.

The case of cell 3 in Figure 6 is also important. Cell 3 is a genuine cutcell but it is finally transformed into a fluid cell. Since cell 3 is essentially inside the fluid, it is better to treat it as a fluid cell instead of defining a boundary condition on it.

Search of solid cells. Once all the cutcells are searched, the search of solid cells is relatively simple. Consider for instance the body display in Figure 7. In order to suppress all the solid cells inside the oven, a reference point P is given. The closest cell, denoted as cell P, is searched and then suppressed, see Figure 7-1. Once cell P is suppressed, its four neighbours are suppressed, see Figure 7-2. The neighbours of all the suppressed cells are suppressed unless they are cutcells, see Figure 7-3,4,5. This process is repeated until all the solid cells are suppressed, see Figure 7-6.

2.6. Advantage of the anisotropic method

Consider a shock line defined by

$$y = x\tan(\alpha), \quad 0 \leqslant x \leqslant \cos\alpha \tag{6}$$

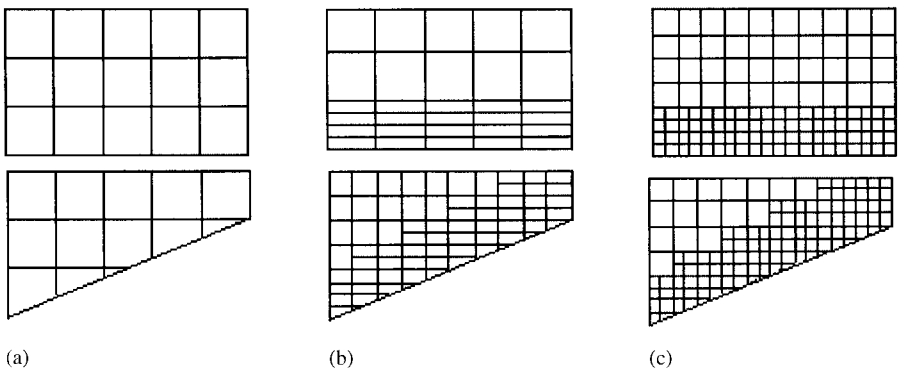

Figure 8. Isotropic and anisotropic refinements near a shock line. (a) initial grid, (b) anisotropic (c) isotropic.

where $0 \leqslant \alpha \leqslant \pi$. The grid is refined in the normal direction of the line starting from an isotropic initial grid as displayed in Figure 8(a). The anisotropic grid and isotropic grid after two levels of refinement are displayed in Figure 8(b) and 8(c) for two typical angles α. It is clear that the anisotropic grid has much fewer points than the isotropic one especially for $\alpha \approx 0$ and $\alpha \approx \pi/2$.

As proved in Reference [23], the statistical ratio between the number of grid points by anisotropic refinement and by isotropic refinement is

$$\bar{r} \approx 0.27 + \frac{0.67}{L}$$

Hence for L large enough, the anisotropic method is grid saving.

3. SOLID WALL TREATMENT

3.1. Various approaches

A Cartesian cell is a rectangle. A Cartesian cell having an intersection with the body surface is called an on-body Cartesian cell. An on-body Cartesian cell is divided by the body surface into two parts. One part lies inside the fluid and is called a cutcell. The other part lies in the solid. Boundary conditions are either defined at the centre of cutcells, or at the centre of on-body Cartesian cells. On-body Cartesian cells are also called uncut version of cutcells.

Cutcells are used to define boundary conditions in the framework of finite-volume approach. As pointed out by Powell [22], there are two major difficulties raised by cutcells: the accuracy and the stability. Cutcells have irregular shapes: they are polygones for two dimensional problems. In this case the geometric centre of the polygone is not at the centre of the uncutted cell so that it is not so easy to define an accurate boundary condition. Some cutcells can be very small, so that the maximum stable time step is reduced. In Figure 9 we show a typical cutcell (cell C). For convenience, we divide the cutcells into two classes: inner cutcell and outer cutcell. Inner cutcell is larger than the half of an uncut cell, and outer cutcell is smaller than the half of an uncut cell. For a finite-volume approach, we then compute the numerical

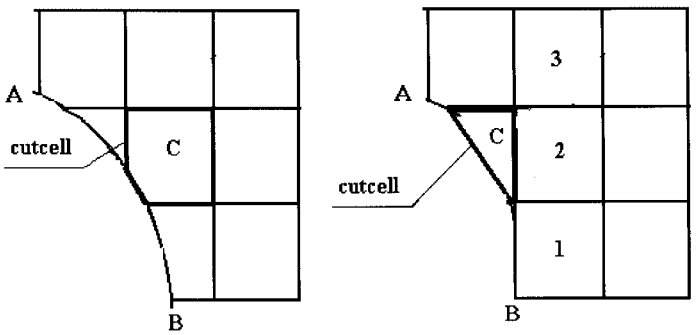

Figure 9. Cutcell definition for finite volume approach. Left: inner cutcell (more than half a part is inside the fluid). Right: outer cutcell (more than half a part is outside the fluid).

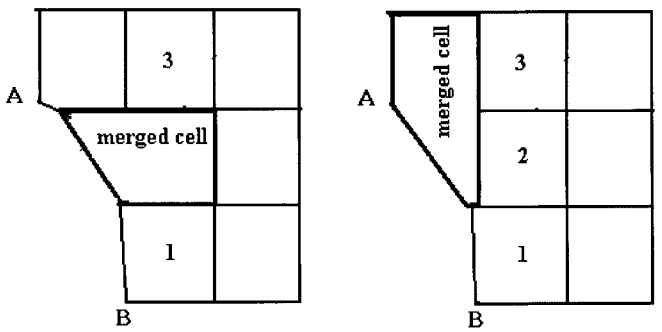

Figure 10. The small cutcell C of Figure 9 is merged with the right cell (Left) or upper cell (Right) to form a large merged cell.

fluxes across each side of the cutcell to update the solution at the centre of the cutcell. In order to avoid instability for small cutcells, two types of boundary treatments were proposed. One is to use a wave propagation approach in which the quantity is redistributed [2, 25]. Another is to use cell merging [6, 10], in which a small cutcell which is less than some user-defined fraction (say $\frac{1}{2}$) of the area of its uncut version is merged with a neighboring uncut cell to have a larger merged cell. Consider for instance the small cutcell C in the right of Figure 9. The cutcell C is merged with its right neighbour (which is an uncut cell) or with its upper neighbour (which is also a cutcell) to form a large merged cell (Figure 10). The numerical fluxes across each side of the merged cell are computed to update the solution at the centre of the cutcell.

In the finite difference approach, the boundary condition is directly defined at the cell centre of each on-body Cartesian grid. For the on-body Cartesian cell shown in Figure 11, the boundary condition is defined at its centre **C**. On-body Cartesian cells or boundary cells can be divided into two classes: inner boundary cell and outer boundary cell. An inner boundary cell has its cell centre inside the flow field. An outer boundary cell has its cell centre inside

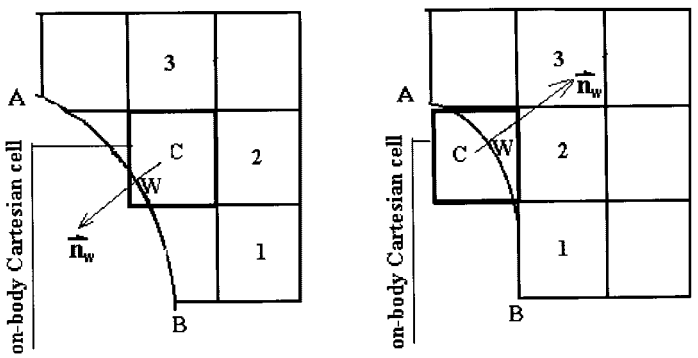

Figure 11. Boundary cell for finite difference treatment. Left: inner boundary cell (the cell centre C is inside the fluid). Right: outer boundary cell (the cutcell centre C is outside the fluid). **W** is on the wall such that **CW** is perpendicular to the wall.

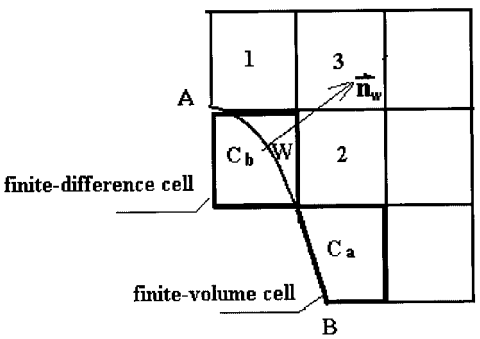

Figure 12. Cutcell definition for mixed finite-volume finite-difference approach. Cutcell C_a is a inner cutcell to be treated using the finite volume approach. Cutcell C_b is a outer cutcell to be treated using the finite difference approach.

the solid. The present boundary condition will be invariant whether the boundary cell is inner or outer.

It is also possible to build a mixed finite-volume finite-difference approach. If the cutcell is larger than a half of the uncut version, then one can use the finite-volume approach. If the cutcell is smaller than the half of the uncut version, then one can use the finite-difference approach. Figure 12 shows two boundary cells. The cutcell C_a is larger than a half of its uncut version and is treated using the finite-volume approach. The boundary cell C_b is treated by the finite difference approach.

In this paper we only consider the finite-difference approach.

3.2. The boundary conditions

The boundary condition is defined at the centre of each boundary cell, no matter this centre lies inside the fluid or the solid. The boundary conditions will be defined for the velocity components u and v, pressure p, and density ρ at the centre of the boundary cell.

The boundary condition is constructed by using the following bilinear interpolation easily extendible to three dimensions:

$$L(\phi, x, y) = d_0^{(\phi)} + d_1^{(\phi)} x + d_2^{(\phi)} y + d_3^{(\phi)} xy \tag{7}$$

where ϕ refers to the two velocity components u and v, the density ρ and the pressure p. The condition at the centre (x_c, y_c) of each boundary cell is then computed as

$$\phi_c = L(\phi, x_c, y_c) \tag{8}$$

The interpolation coefficients $d_i^{(\phi)}$ with $i = 1, 2, 3, 4$ are determined using known solutions at three adjacent fluid cells ϕ_1, ϕ_2, ϕ_3. This yields the following three relations:

$$d_0^{(\phi)} + d_1^{(\phi)} x_1 + d_2^{(\phi)} y_1 + d_3^{(\phi)} x_1 y_1 = \phi_1 \tag{9}$$

$$d_0^{(\phi)} + d_1^{(\phi)} x_2 + d_2^{(\phi)} y_2 + d_3^{(\phi)} x_2 y_2 = \phi_2 \tag{10}$$

$$d_0^{(\phi)} + d_1^{(\phi)} x_3 + d_2^{(\phi)} y_3 + d_3^{(\phi)} x_3 y_3 = \phi_3 \tag{11}$$

Since we have four interpolation coefficients, we need a fourth relation. The fourth relation can be of a Dirichlet boundary condition (such as the isothermal condition on the wall) or a Neumann condition (such as the adiabatic condition for temperature).

Let $\mathbf{x}_w$ be the co-ordinate of the point on the wall that is nearest to the boundary cell centre $\mathbf{x}_c$.

A Dirichlet condition generally has the following form

$$\phi_w = g \tag{12}$$

where $\phi_w = \phi(\mathbf{x}_w) = L(\phi, x_w, y_w)$ so that, by (7), we have the fourth relation for the interpolation coefficients

$$d_0^{(\phi)} + d_1^{(\phi)} x_w + d_2^{(\phi)} y_w + d_3^{(\phi)} x_w y_w = g \tag{13}$$

Let $\mathbf{n}_w$ be the unit normal at $\mathbf{x}_w$. A Neumann condition can be generally written as:

$$\frac{\partial \phi}{\partial n_w} = g \tag{14}$$

Remark that

$$\left.\frac{\partial \phi}{\partial n_w}\right|_w = \left(\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}\right)_w \cdot (n_w^{(x)}, n_w^{(y)}) \tag{15}$$

$$\left.\frac{\partial \phi}{\partial x}\right|_w = d_1^{(\phi)} + y_w d_3^{(\phi)} \tag{16}$$

$$\left.\frac{\partial \phi}{\partial y}\right|_w = d_2^{(\phi)} + x_w d_3^{(\phi)} \tag{17}$$

Combining the above relations with (14) leads to the following additional relation for the interpolation coefficients:

$$n_{\mathrm{w}}^{(x)} d_1^{(\phi)} + n_{\mathrm{w}}^{(y)} d_2^{(\phi)} + (n_{\mathrm{w}}^{(x)} y_{\mathrm{w}} + n_{\mathrm{w}}^{(y)} x_{\mathrm{w}}) d_3^{(\phi)} = g \tag{18}$$

Now consider the boundary condition for ϕ.

If the boundary condition on the wall is of Dirichlet type as given by (12), then relations (9)–(11) and (13) should be solved to determine the coefficients $d_i^{(\phi)}$ and then use (7)–(8) to determine the boundary condition ϕ_{c} at the boundary cell centre.

If the boundary condition on the wall is of Neumann type as given by (14), then the relations (9)–(11) and (18) should be solved to determine the coefficients d_i and then use (7)–(8) to determine the boundary condition ϕ_{c} at the boundary cell centre.

First recall how the nonslip condition as for a viscous fluid was used in the past. We have $u = v = 0$ on the wall. Thus for the velocity components we use the Dirichlet condition with $g = 0$. For the pressure, if the Reynolds number is high enough, we use the Neumann condition with $g = 0$. For the density, it depends on whether the wall is adiabatic or has a fixed temperature. If the wall is adiabatic, we still have a Neumann condition with $g = 0$. If the wall has a fixed temperature T_{w}, then we first need to compute the pressure on the wall by (7):

$$p_{\mathrm{w}} = d_0^{(p)} + d_1^{(p)} x_{\mathrm{w}} + d_2^{(p)} y_{\mathrm{w}} + d_3^{(p)} x_{\mathrm{w}} y_{\mathrm{w}} \tag{19}$$

where d_i are determined by using (9)–(11) and (18) for $\phi = p$. Then the density on the wall is computed as $\rho_{\mathrm{w}} = p_{\mathrm{w}}/(R_{\mathrm{g}} T_{\mathrm{w}})$ where R_g is the gas constant. Then the density at the boundary cell centre is computed by using the Dirichlet condition with $g = \rho_{\mathrm{w}}$.

Now consider the slip wall condition as used for the Euler equations in gas dynamics. The total enthalpy H_{c} and the entropy S_{c} at the boundary cell centres are computed using the Neumann condition with $g = 0$. Precisely, the normal derivatives of the total enthalpy and entropy are assumed to vanish on the wall. For steady flow problem the total entropy is indeed uniform so that it is reasonable to assume that the normal derivative of H_{c} on the wall vanishes. The normal derivative of entropy on the wall vanishes in most cases, except near the point of intersection of an oblique shock wave and the wall. Such boundary conditions are frequently used, see for instance Hirsh [26].

The normal velocity component $U_{\mathrm{n,c}}$ at the boundary cell centre is computed using the Dirichlet condition with $g = 0$. Now consider the pressure.

The momentum equation projected to the direction normal to the wall can be written as

$$\rho \frac{\partial U_{n_{\mathrm{w}}}}{\partial t} + \rho U_{n_{\mathrm{w}}} \frac{\partial U_{n_{\mathrm{w}}}}{\partial n_{\mathrm{w}}} + \rho U_{\tau_{\mathrm{w}}} \frac{\partial U_{n_{\mathrm{w}}}}{\partial \tau_{\mathrm{w}}} = -\frac{\partial p}{\partial n_{\mathrm{w}}}$$

where $\mathbf{n}_{\mathrm{w}}$ is the direction normal to the wall, and $\boldsymbol{\tau}_{\mathrm{w}}$ is the direction tangent to the wall. Since $U_{n_{\mathrm{w}}} = 0$ on the wall, the above equation yields the following condition on the wall

$$\frac{\partial p}{\partial n_{\mathrm{w}}} = -\rho U_{\tau_{\mathrm{w}}} \frac{\partial U_{n_{\mathrm{w}}}}{\partial \tau_{\mathrm{w}}} \tag{20}$$

Hence the pressure at the boundary cell centre can be computed by using the Neumann condition with $g = -\rho U_{\tau_{\mathrm{w}}} \partial U_{n_{\mathrm{w}}}/\partial \tau_{\mathrm{w}}$. To compute this g, we must know ρ, $U_{\tau_{\mathrm{w}}}$, and $\partial U_{n_{\mathrm{w}}}/\partial \tau_{\mathrm{w}}$ on the wall. The density ρ and the tagent velocity component $U_{\tau_{\mathrm{w}}}$ are computed using the

Neumann condition with $g=0$. Consider for instance the tagent velocity component U_{τ_w} on the wall. We use (9)–(11) and (18) for $g=0$ to compute the interpolation coefficients. Then we compute U_{τ_w} as

$$U_{\tau_w} = U_{\tau_w,w} = L(U_{\tau_w}, x_w, y_w) \tag{21}$$

The density on the wall is computed similarly. Now consider the gradient $\partial U_{n_w}/\partial\tau_w$ on the wall. Set $\phi = U_{n_w}$ and use (9)–(11) and (13) with $g=0$ to determine the interpolation coefficients. Then we use (16)–(17) to compute $\partial U_{n_w}/\partial x$ and $\partial U_{n_w}/\partial y$. The gradient $\partial U_{n_w}/\partial\tau_w$ on the wall is then computed as

$$\frac{\partial U_{n_w}}{\partial \tau_w} = \frac{\partial U_{n_w}}{\partial x}\,\tau_w^{(x)} + \frac{\partial U_{n_w}}{\partial y}\,\tau_w^{(y)} \tag{22}$$

Now we know $U_{n_w,c}$, p_c, H_c, and S_c at the boundary cell centre. For convenience, we need to know the density ρ_c, the Cartesian velocity components u_c and v_c. The density can be computed as

$$\rho_c = \left(\frac{p_c}{S_c}\right)^{1/\gamma}$$

The square of the total velocity at the boundary cell centre is given by

$$U_c^2 = 2\left(H_c - \frac{\gamma}{\gamma-1}\,\frac{p_c}{\rho_c}\right)$$

so that the tangent component is

$$U_{\tau_w,c} = \mathrm{sign}(U_{\tau_w,w})\sqrt{U_c^2 - U_{n_w,c}^2} = \mathrm{sign}(U_{\tau,w})\sqrt{2\left(H_c - \frac{\gamma}{\gamma-1}\frac{p_c}{\rho_c}\right) - U_{n_w,c}^2}$$

where $U_{\tau_w,w}$ is given by (21). Finally, u_c and v_c are given below:

$$u_c = \frac{n_w^{(y)}U_{\tau_w,c} - \tau_w^{(y)}U_{n_w,c}}{\tau_w^{(x)}n_w^{(y)} - n_w^{(x)}\tau_w^{(y)}}, \quad v_c = \frac{\tau_w^{(x)}U_{n_w,c} - n_w^{(x)}U_{\tau_w,c}}{\tau_w^{(x)}n_w^{(y)} - n_w^{(x)}\tau_w^{(y)}}$$

since $U_{n_w,c} = u_c n_w^{(x)} + v_c n_w^{(y)}$ and $U_{\tau_w,c} = u_c\tau_w^{(x)} + v_c\tau_w^{(y)}$.

4. NUMERICAL EXPERIMENTS

4.1. Difference approximation

Consider the Euler equations in gas dynamics:

$$w_t + f_x + g_y = 0 \tag{23}$$

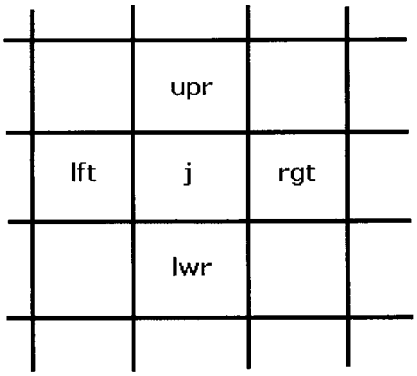

Figure 13. A normal cell for defining a conservative scheme. The time variation of the solution at the cell centre j is contributed by the numerical fluxes across the left face (lft), right face (rgt), lower face (lwr), and upper face (upr).

where

$$w=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho E\end{pmatrix},\quad f=\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uH\end{pmatrix},\quad g=\begin{pmatrix}\rho v\\ \rho vu\\ \rho v^2+p\\ \rho vH\end{pmatrix}$$

Here γ denotes the ratio of specific heats, ρ, p, E and H are respectively the density, the pressure, the total energy and the total enthalpy, $\mathbf{U}=u\mathbf{i}+v\mathbf{j}$ is the velocity vector with the components u and v. With the assumption of a perfect gas, the following relations hold:

$$H=E+\frac{p}{\rho},\quad p=(\gamma-1)\rho\left(E-\frac{1}{2}\mathbf{U}^2\right)$$

We use the Roe scheme [27] to solve the Euler equations in gas dynamics. First let us consider a normal cell without interface as displayed in Figure 13 and write the conservative scheme in the following form:

$$\Delta w_j=-\sigma_j(h^*_{\mathrm{rgt}}-h^*_{\mathrm{lft}}+h^*_{\mathrm{upr}}-h^*_{\mathrm{lwr}}) \tag{24}$$

Here $\sigma_j=\Delta t_j/S_j$, Δt_j being the time-step and S_j the area of the cell. Let Δx_j and Δy_j be the cell size in x and y, then the numerical fluxes h^*_{rgt} and h^*_{upr} are given by:

$$\frac{h^*_{\mathrm{rgt}}}{\Delta y_j}=\frac{1}{2}(f^n_j+f^n_{\mathrm{right}})+\frac{1}{2}|A^{(R)}_{\mathrm{rgt}}|(w^n_j-w^n_{\mathrm{right}}) \tag{25}$$

$$\frac{h^*_{\mathrm{upr}}}{\Delta x_j}=\frac{1}{2}(f^n_j+f^n_{\mathrm{upper}})+\frac{1}{2}|B^{(R)}_{\mathrm{upr}}|(w^n_j-w^n_{\mathrm{upper}}) \tag{26}$$

and the fluxes h^*_{lft}, h^*_{lwr} are similarly defined. Here $f^n_j = f(w^n_j)$, $f^n_{\text{right}} = f(w^n_{\text{right}})$, and $f^n_{\text{upper}} = f(w^n_{\text{upper}})$, w^n_{right} is the solution at the right neighbour of j, and w^n_{upper} is the solution at the upper neighbour of j. The matrices $A^{(R)}_{\text{rgt}} = A^{(R)}(w^n_j, w^n_{\text{right}})$ and $B^{(R)}_{\text{upr}} = B^{(R)}(w^n_j, w^n_{\text{upper}})$ are Roe matrix at the right face and the upper face, respectively. The symbols $|A^{(R)}|$ and $|B^{(R)}|$ denote the absolute values of $A^{(R)}$ and $B^{(R)}$. The explicit formulas for $A^{(R)}$ and $B^{(R)}$ are given by

$$A^{(R)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -u_R^2 + \frac{\gamma-1}{2}U_R^2 & (3-\gamma)u_R & (1-\gamma)v_R & \gamma-1 \\ -u_R v_R & v_R & u_R & 0 \\ -u_R((\gamma-1)H_R + \frac{3-2\gamma}{2}U_R^2) & (\gamma-1)(H_R - u_R^2) + \frac{2-\gamma}{2}U_R^2 & (1-\gamma)u_R v_R & \gamma u_R \end{pmatrix}$$

$$B^{(R)} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -v_R u_R & v_R & u_R & 0 \\ -u_R^2 + \frac{\gamma-1}{2}U_R^2 & (1-\gamma)v_R & (3-\gamma)u_R & \gamma-1 \\ -v_R((\gamma-1)H_R + \frac{3-2\gamma}{2}U_R^2) & (1-\gamma)v_R u_R & (\gamma-1)(H_R - v_R^2) + \frac{2-\gamma}{2}U_R^2 & \gamma v_R \end{pmatrix}$$

where $U_R^2 = u_R^2 + v_R^2$, and u_R, v_R and H_R are Roe averages defined by

$$u_R = \frac{\sqrt{\rho_j}u_j + \sqrt{\rho_{\text{nbr}}}u_{\text{nbr}}}{\sqrt{\rho_j} + \sqrt{\rho_{\text{nbr}}}}$$

$$v_R = \frac{\sqrt{\rho_j}v_j + \sqrt{\rho_{\text{nbr}}}v_{\text{nbr}}}{\sqrt{\rho_j} + \sqrt{\rho_{\text{nbr}}}}$$

$$H_R = \frac{\sqrt{\rho_j}H_j + \sqrt{\rho_{\text{nbr}}}H_{\text{nbr}}}{\sqrt{\rho_j} + \sqrt{\rho_{\text{nbr}}}}$$

Here the subscript nbr means the neighbour of j, such as the right neighbour and the upper neighbour.

Now consider a cell j with a coarse right neighbour j_R as displayed in Figure 14. To compute the numerical flux at the right face of j, we use a fictive right neighbour j_r, which is defined as if there were no interface. Precisely, the right state w^n_{right} used to compute $h^*_{\text{rgt}} = h^*_{\text{rgt}}(w^n_j, w^n_{\text{right}})$ for cell j is defined as

$$w^n_{\text{right}} = w_{j_r}$$

where w_{j_r} is computed through interpolation from the states around j and j_R.

When there is an interface such as the left face of cell j displayed in Figure 15, we follow Reference [18] to compute the numerical flux by $h^{(*)}_{\text{lft}} = h^{(*)}_{ab} + h^{(*)}_{bc}$ where $h^{(*)}_{ab}$ and $h^{(*)}_{bc}$ are numerical fluxes at the right face of the finer cells i and m. Such a treatment ensures conservation.

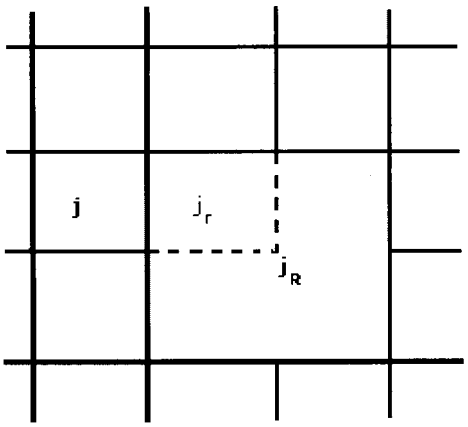

Figure 14. A Cartesian cell j with a coarse right neighbour j_R.

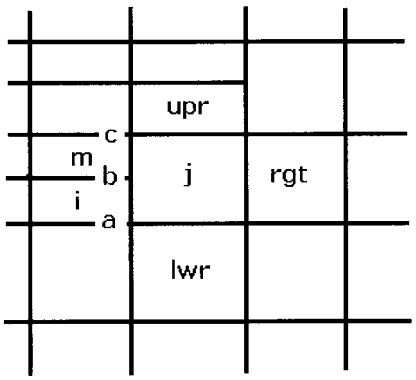

Figure 15. A cell with left interface for defining a conservative scheme. The left face for cell j has two neighbours (cell i and cell m). The left face ac is divided into two segments ab and bc.

4.2. Numerical results

In all the computations we have used local time stepping (matrix time step) to accelerate the convergence to a steady state.

4.2.1. Symmetric flow around a NACA0012 airfoil. Consider a symmetric flow a NACA0012 airfoil. The free stream Mach number is $M_\infty = 0.8$. The open boundary extends to a distance of 10 times the chord length. The corresponding Cartesian grids obtained by the isotropic algorithm and the anisotropic algorithm are displayed in Figures 16 and 17. Both grids have 12 levels. The isotropic refinement algorithm leads to 14128 cells while the anisotropic algorithm reduces the total number of cells to 6625. In comparison with the isotropic algorithm, the anisotropic algorithm has saved 53% cells. It remains to see whether the two grids lead to the same solution.

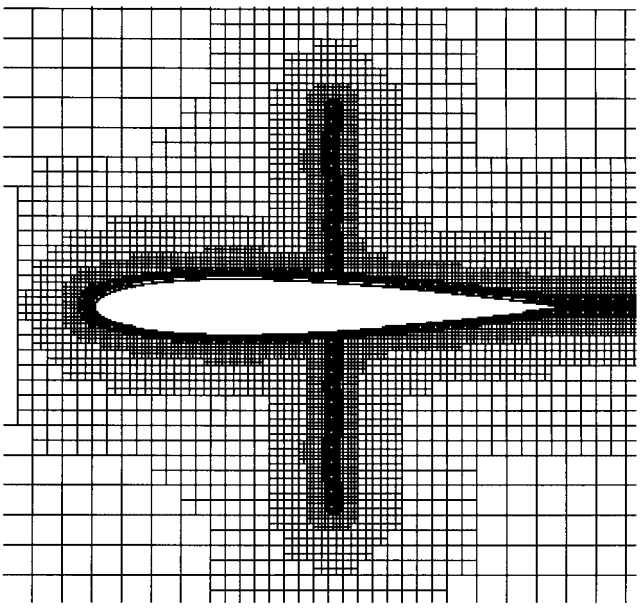

Figure 16. Isotropic grids (inner part) for NACA 0012 airfoil ($M_\infty = 0.8$).

The computed Mach contours for both grids are displayed in Figures 18 and 19. The difference between the solutions on both grids is not visible.

In Figure 20 we display the distribution of pressure coefficient on the chord. The result obtained from anisotropic Cartesian grid is not distinguishable from that obtained by isotropic Cartesian grid.

As a result, the anisotropic Cartesian grid method saves considerably the number of cells while maintaining the same accuracy as the corresponding isotropic Cartesian grid method.

4.2.2. Transonic flow around a NACA0012 airfoil with incidence. Consider a flow around a NACA0012 airfoil. The free stream Mach number is $M_\infty = 0.85$. The incidence of the flow is $\alpha = 1°$. The corresponding Cartesian grids obtained by the isotropic algorithm and the anisotropic algorithm are displayed in Figures 21 and 22, respectively. Both grids have 12 levels. The isotropic refinement algorithm leads to 17398 cells while the anisotropic algorithm reduces the total number of cells to 6643. In comparison with the isotropic algorithm, the anisotropic algorithm has saved 62% of the cells. It remains to see whether the two grids lead to the same solution.

The computed Mach contours for both grids are displayed in Figures 23 and 24. The difference between the solutions on both grids is not visible.

In Figure 25 we display the distribution of pressure coefficient on the chord. The curve obtained from anisotropic Cartesian grid is not distinguishable from that obtained by isotropic Cartesian grid.

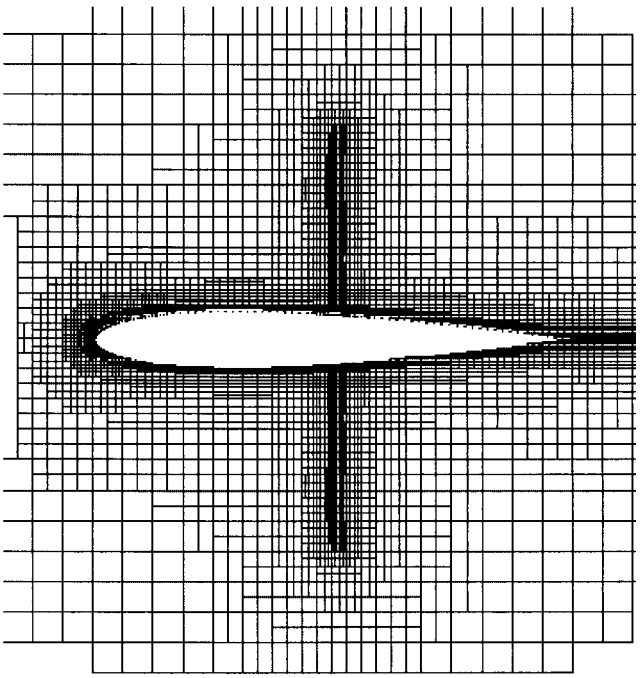

Figure 17. Anisotropic grids (inner part) for NACA 0012 airfoil ($M_\infty = 0.8$).

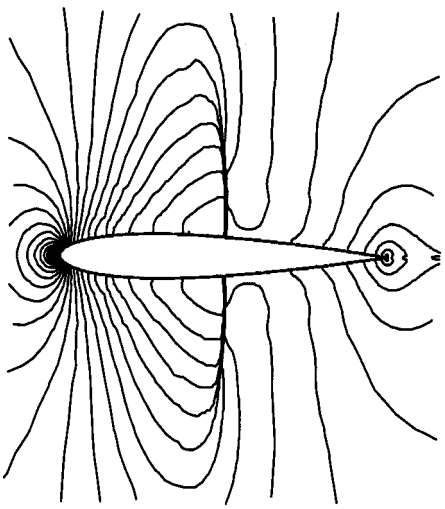

Figure 18. Mach contours for NACA 0012 airfoil with isotropic grids ($M_\infty = 0.8$).

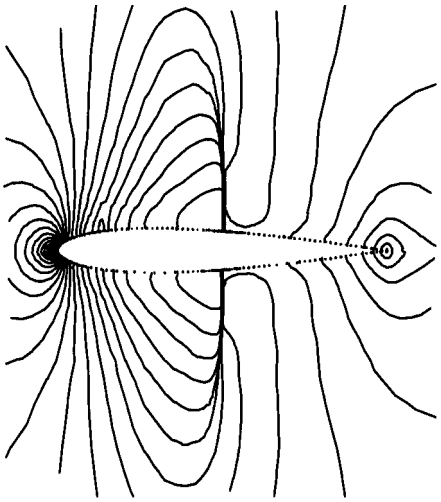

Figure 19. Mach contours for NACA 0012 airfoil with anisotropic grids ($M_\infty = 0.8$).

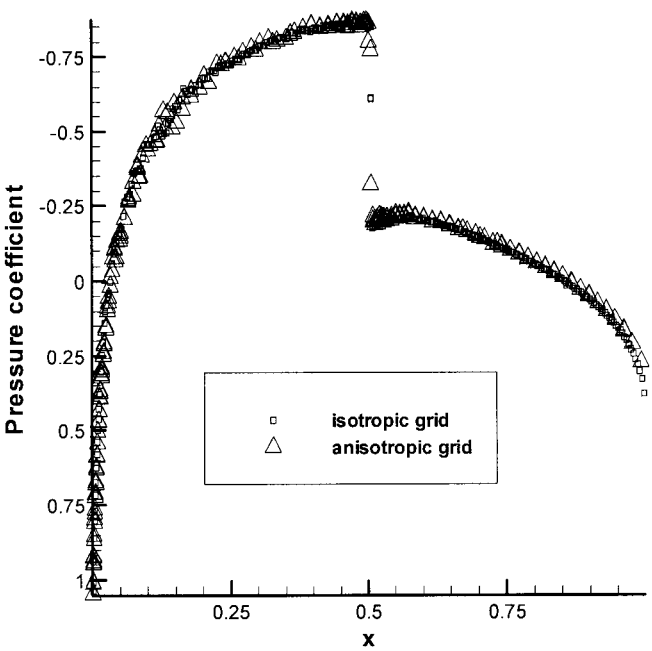

Figure 20. Pressure distribution around NACA 0012 ($M_\infty = 0.8$).

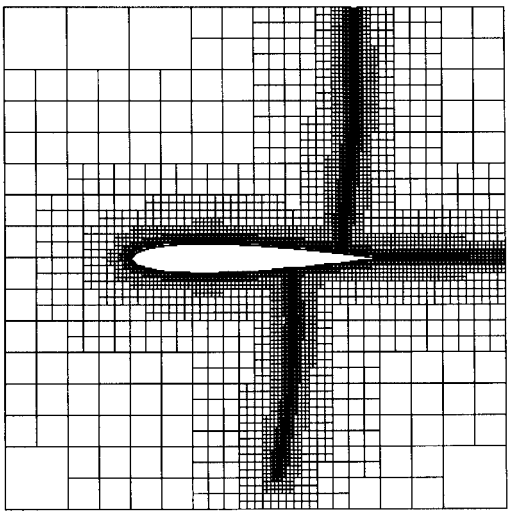

Figure 21. Isotropic grids ($M_\infty = 0.85$. The incidence of the flow is equal to 1°).

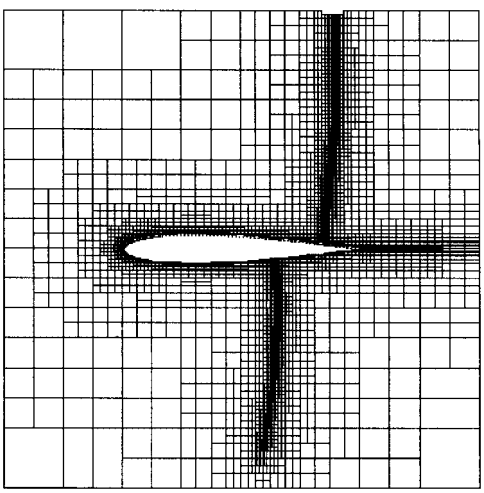

Figure 22. Anisotropic grids ($M_\infty = 0.85$. The incidence of the flow is equal to 1°).

4.2.3. Supersonic flow around a NACA0012 airfoil. Consider a supersonic flow around a NACA0012 airfoil. The free stream Mach number is $M_\infty = 1.2$. The incidence of the flow is $\alpha = 7^\circ$. The corresponding Cartesian grids obtained by the isotropic algorithm and the anisotropic algorithm are displayed in Figures 26 and 27. Both grids have 12 levels. The isotropic refinement algorithm leads to 15631 cells while the anisotropic algorithm reduces the total number of cells to 9325. In comparison with the isotropic algorithm, the anisotropic

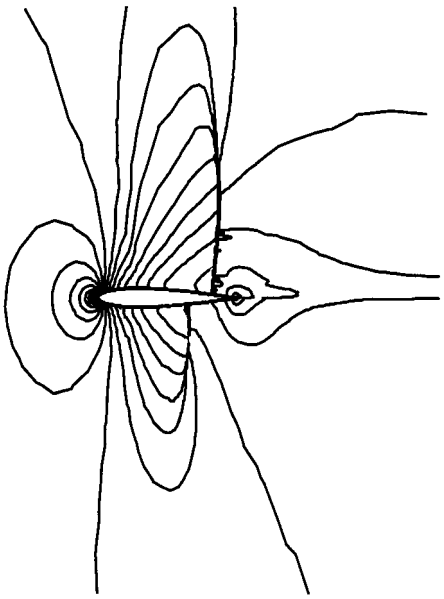

Figure 23. Mach contours with isotropic grids ($M_\infty = 0.85$. The incidence of the flow is equal to 1°).

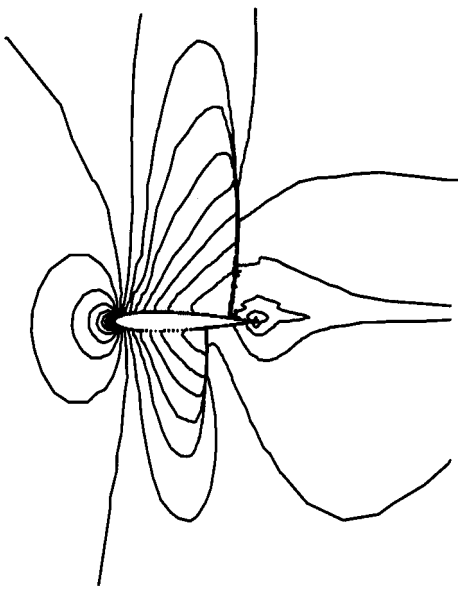

Figure 24. Mach contours with anisotropic grids ($M_\infty = 0.85$. The incidence of the flow is equal to 1°).

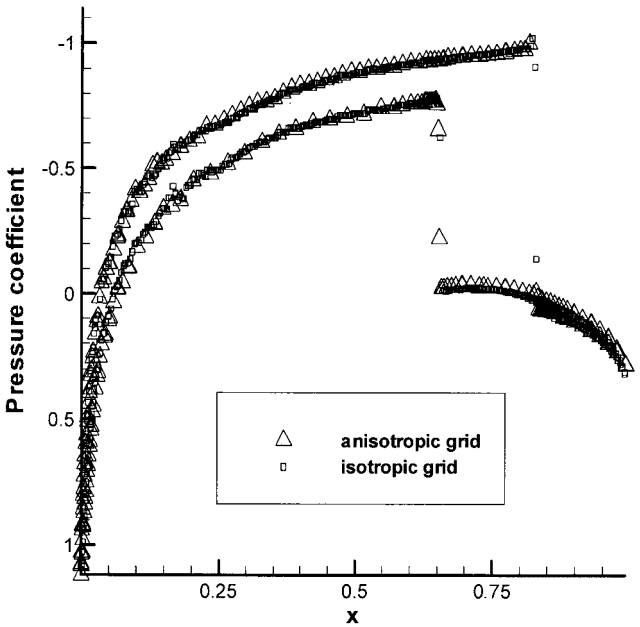

Figure 25. Pressure distribution ($M_\infty = 0.85$. The incidence of the flow is equal to 1°).

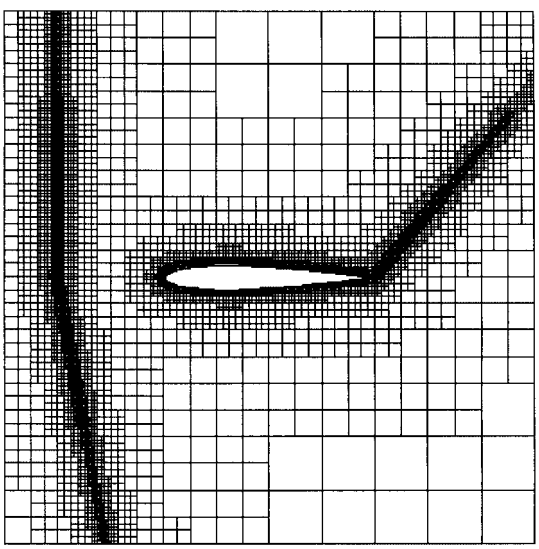

Figure 26. Isotropic grids ($M_\infty = 1.2$. The incidence of the flow is equal to 7°).

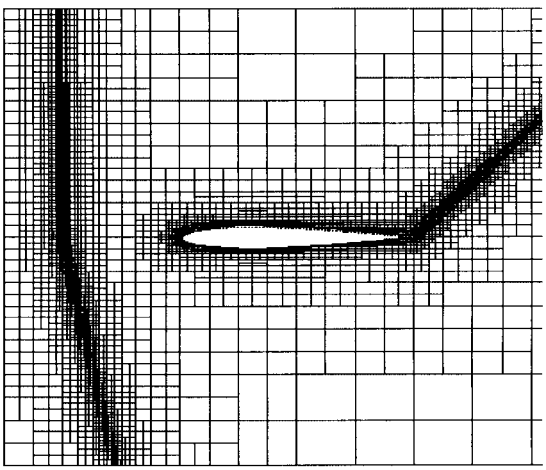

Figure 27. Anisotropic grids ($M_\infty = 1.2$. The incidence of the flow is equal to 7°).

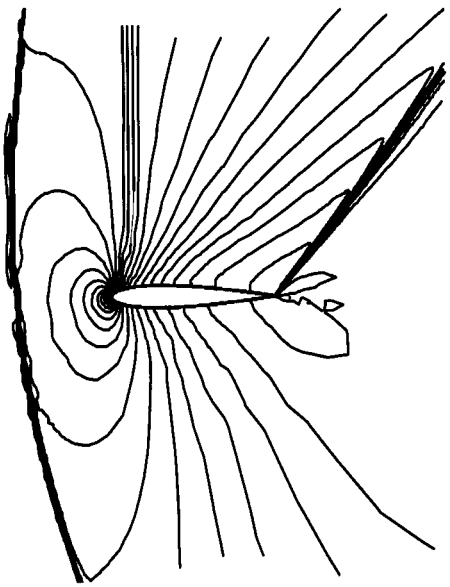

Figure 28. Mach contours with isotropic grids ($M_\infty = 1.2$. The incidence of the flow is equal to 7°).

algorithm has saved 41% of the cells. It remains to see whether the two grids lead to the same solution.

The computed Mach contours for both grids are displayed in Figures 28 and 29. The difference between the solutions on both grids is not visible.

In Figure 30 we display the distribution of pressure coefficient on the chord. The curve obtained from anisotropic Cartesian grid is not distinguishable from that obtained by isotropic Cartesian grid.

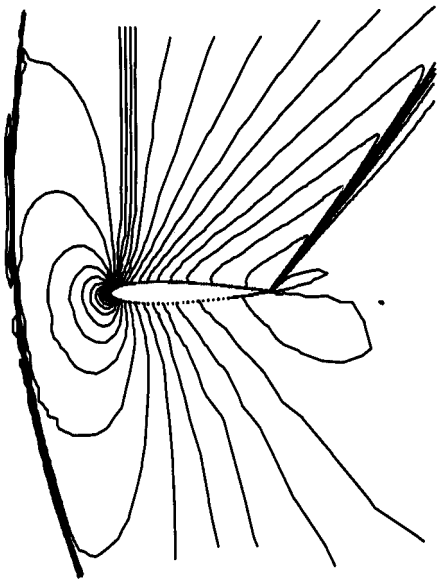

Figure 29. Mach contours with anisotropic grids ($M_\infty = 1.2$. The incidence of the flow is equal to 7°).

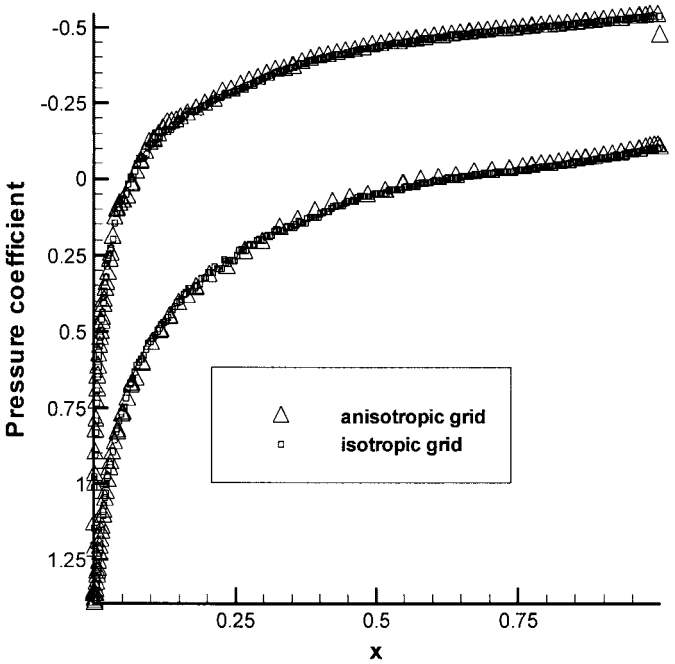

Figure 30. Pressure coefficient distribution ($M_\infty = 1.2$. The incidence of the flow is equal to 7°).

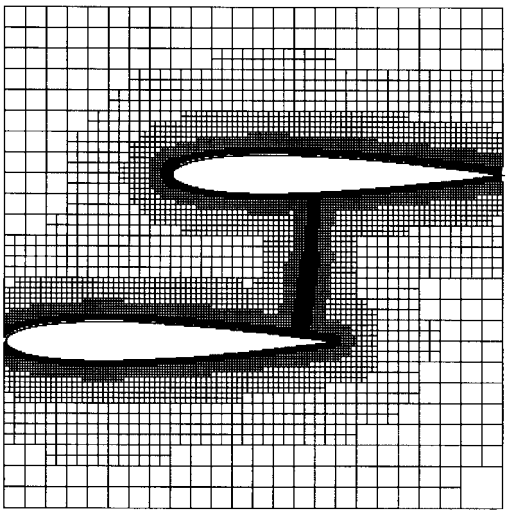

Figure 31. Isotropic grid (inner part) for Bi-NACA airfoil.

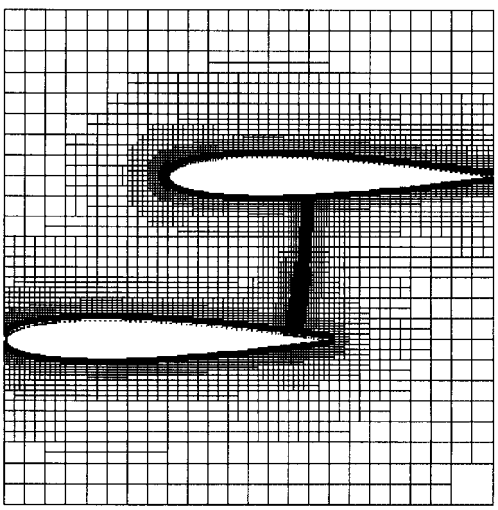

Figure 32. Anisotropic grid (inner part) for Bi-NACA airfoil.

4.2.4. Flow around a Bi-NACA airfoil. Consider a flow around a Bi-NACA airfoil. The Bi-NACA airfoil is formed by two parallel NACA0012 airfoils. They are shifted by a distance equal to half a chord in both the parallel and perpendicular directions. The free stream Mach number is $M_\infty = 0.7$. The flow has no incidence. The open boundary extends to a distance of 10 times the chord length. The corresponding Cartesian grids obtained by the isotropic algorithm and the anisotropic algorithm are displayed in Figures 31 and 32. The isotropic and anisotropic grids have16472 and 9362 cells, respectively.

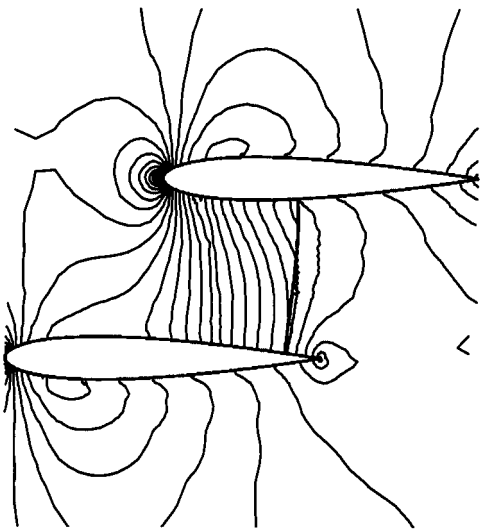

Figure 33. Mach contours for BI-NACA airfoil with isotropic grids ($M_\infty = 0.7$).

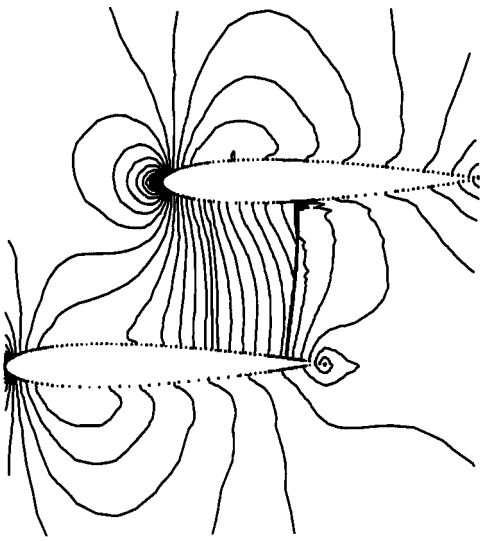

Figure 34. Mach contours for BI-NACA airfoil with anisotropic grids ($M_\infty = 0.7$).

The computed Mach contours for both grids are displayed in Figures 33 and 34. We observe no difference between the solutions on the two grids. The agreement between the two solutions is more convincingly shown in Figure 35 where the distributions of pressure coefficients are displayed.

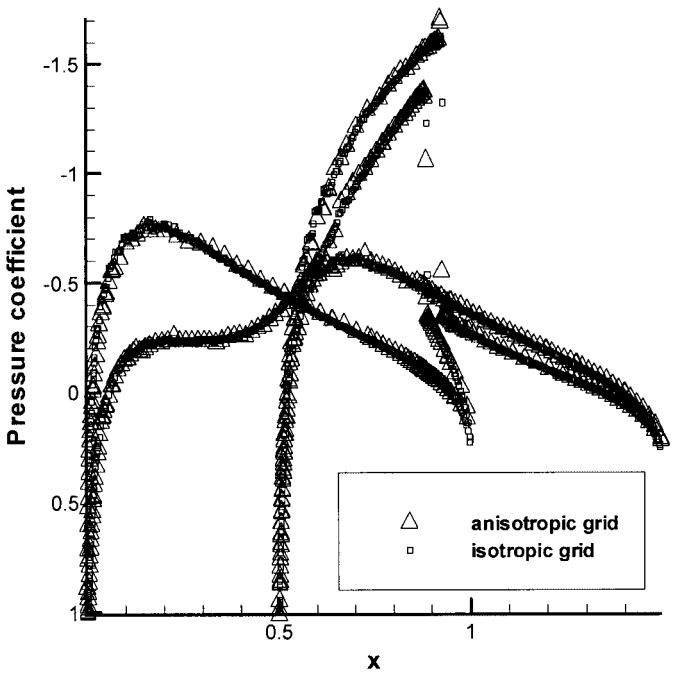

Figure 35. Pressure distribution for BI-NACA airfoil ($M_\infty = 0.7$).

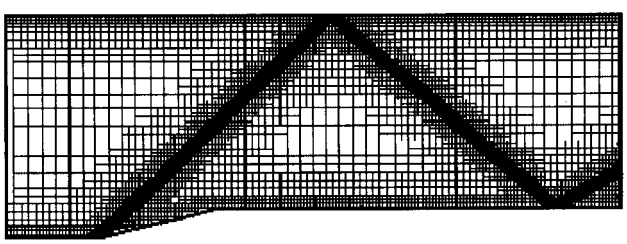

Figure 36. Isotropic grids.

In consequence, the anisotropic Cartesian grid method saves considerably the number of cells while maintaining the same accuracy as the corresponding isotropic Cartesian grid method even for a multi-element body.

4.2.5. Forward step flow. Consider the forward step problem with $M_\infty = 2$.
The corresponding Cartesian grids obtained by the isotropic algorithm and the anisotropic algorithm are displayed in Figures 36 and 37. Both grids have 8 levels. The computed Mach contours for both grids are displayed in Figures 38 and 39. The solutions for both grids are very close. The isotropic refinement algorithm leads to 6916 cells while the anisotropic algorithm reduces the total number of cells to 5974. In comparison with the isotropic algorithm,

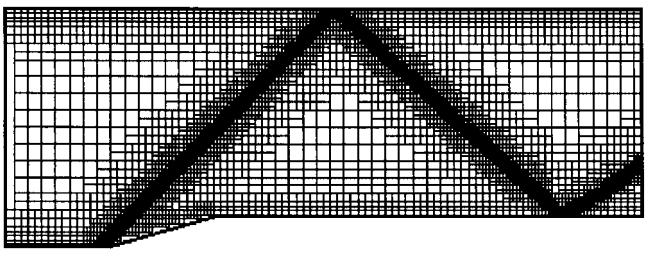

Figure 37. Anisotropic grids.

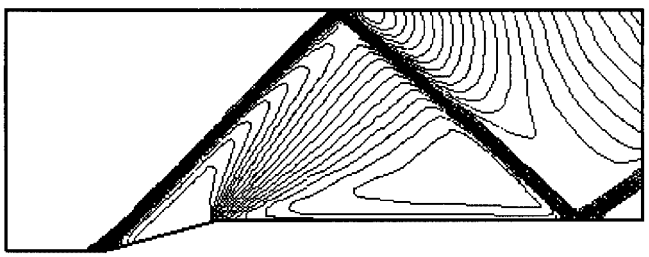

Figure 38. Mach contours with isotropic grids ($M_\infty = 2$).

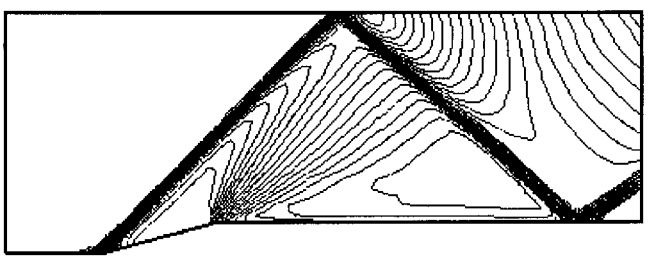

Figure 39. Mach contours with anisotropic grids ($M_\infty = 2$).

the anisotropic algorithm has saved only 13% of the cells. This means if the shock lines is in the 45° direction, the advantage of the anisotropic approach is unimportant.

5. CONCLUDING REMARKS AND FURTHER REMARKS

The anisotropic Cartesian grid method initially developed in References [13, 23] for viscous flow computation has been successfully extended to inviscid flow computations with shock waves. The refinement near shock waves does not involve new difficulties. Besides, a very simple and efficient finite-difference boundary condition has been constructed for solid wall treatment. This method ensures stability and does not involve special treatment such as cell

cutting. As we have stated in Section 3.1, only a finite difference boundary condition is considered and this method does not require cell cutting as would occur in finite volume treatment of boundary conditions. Besides, it does not create trouble as regard to conservation.

There are several important issues with regard to stability and accuracy. Notably, we have found that Cartesian grids involving multiple refinement interfaces create positive or negative dissipation which would have some consequence on stability. This issue has been discussed in a separate paper [28] and the main conclusion, which holds for both uniform timestepping and matrix time stepping, can be summarized as follows.

Theorem 5.1
For a general semi-discrete three-point difference approximation with multiple refinement interfaces, if the wave moves in the fine-to-coarse direction then the dissipation is positive (stabilizing), and if the wave moves in the coarse-to-fine direction then the dissipation is negative (de-stabilizing). Moreover, the amount of dissipation is insensitive to the subgrid width if the total refinement degree is fixed.

It is also interesting to know the accuracy on an adaptive Cartesian grid with multiple refinement interfaces. This issue will also be discussed in a separate paper and the main conclusion, which holds for steady state solution, can be summarized as follows.

Theorem 5.2
For a general three-point difference approximation with multiple refinement interfaces, the steady state solution has a second order accuracy with respect to the coarsest mesh size and that it has the same accuracy as the conventional smooth refinement method if the ratio of the finest mesh size to the coarsest mesh size is the same.

ACKNOWLEDGEMENTS

This work was supported by Chinese National Natural Science Foundations (Contract No. 10025210) and by the China NKBRSF project (Contract No. 2001CB409600).

REFERENCES

1. Wu ZN. Transmission of slowly moving shock waves across a nonconservative interface. *Journal of Computational Physics* 2001; **171**:579–615.
2. Berger MJ, LeVeque R. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. *AIAA Paper-89-1930*, 1989.
3. Bayyuk SA, Powell KG, van Leer B. An algorithm for the simulation of 2-D unsteady inviscid flows around moving and deforming bodies of aribitrary geometry. *AIAA Paper-93-3391*, 1993.
4. Clarke DK, Salas MD, Hassan HA. Euler calculations for multi-element airfoils using Cartesian grids. *AIAA Journal* 1986; **24**:353–358.
5. Coirier WJ, Powell KG. An accuracy assessment of Cartesian-mesh approaches for the Euler equations. *Journal of Computational Physics* 1995; **117**:121–131.
6. De Zeeuw D, Powell KG. An adaptively refined Cartesian mesh solver for the Euler equations. *Journal of Computational Physics* 1993; **104**:56–68.
7. Gropp WD. Local uniform mesh refinement with moving grids. *SIAM Journal on Scientific and Statistic Computing* 1987; **8**:292–304.
8. Mitcheltree RA, Salas MD, Hassan HA. Grid embedding technique using Cartesian grids for Euler solutions. *AIAA Journal* 1998; **26**:754–756.
9. Morinishi K. A finite difference solution of the Euler equations on non-body-fitted Cartesian grids. *Computers and Fluids* 1992; **21**:331–344.

10. Quirk J. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Computers and Fluids* 1994; **23**:125–142.
11. Coirier WJ, Powell KG. Solution adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA Journal* 1996; **34**:938–945.
12. Nakano A, Satofuka N, Shimomura N. A Cartesian grid approach to compressible viscous flow computations. In *Computational Fluid Dynamics*'96. John Wiley & Sons Ltd: New York.
13. Wu ZN. A genuinely second-order accurate method for viscous flow computations around complex geometries. In *The 15th Conference on Numerical Methods in Fluid Dynamics*, Monterey, USA, June 24–28, 1996.
14. Deister F, Hirschel EH. Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometry. *AIAA Paper* 99-0782, 1999.
15. Deister F, Rocher D, Hirschel EH, Monnoyer F. Three-dimensional adaptive refined Cartesian grid generation and Euler flow solutions for arbitrary geometry. *Proceedings of the 4th European CFD Conference*, 1, Part 1, 1999; 96–101.
16. Domel ND, Karman Jr SL. Splitflow: progress in 3D CFD with Cartesian Omni-tree grids for complex geometries adaptive. *AIAA Paper* 2000-1006, 2000.
17. Welterlen TJ. Store release simulation on the F/A-18C using splitflow. *AIAA Paper 99-0124*, 1999.
18. Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**:64–84.
19. Berger MJ, Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics* 1984; **53**:484–512.
20. Liou MS, Kao KH. Progress in grid generation: from chimeria to dragon grids. *Frontiers of Computational Fluid Dynamics*, Caughey A, Hafez MM (eds). John Wiley & Sons: New York, 1994; 385–411.
21. Starius G. On composite mesh difference methods for hyperbolic differential equations. *Numerical Mathematics* 1980; **35**:241–255.
22. Powell KG. Solution of the Euler equations on solution-adaptive Cartesian grids. *CFD Review*. World Scientific: Singapore, 1998; 65–92.
23. Wu ZN. Anisotropic Cartesian grid approach for viscous flow computations. *CFD Review*. World Scientific: Singapore, 1988; 93–113.
24. Habishi WG, Dompierre J, Bourgault Y, Ait-Ali-Yahia D, Fortin M, Vallet M-G. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD, Part I: general principles. *Journal of Computational Physics* 2000; **32**:725–744.
25. LeVeque RJ. Cartesian grid methods for flow in irregular regions. In *Numerical Methods in Fluid Dynamics*, vol. III, Morton KW, Baines MJ (eds). Clarendon Press: 1988; 375–382.
26. Hirsch C. *Numerical Computation of Internal and External Flows*, vol. I. John Wiley & Sons: Chichester, 1990.
27. Roe PL. Approximate Riemann Solvers: parameter vector and difference schemes. *Journal of Computational Physics* 1981; **43**:357–372.
28. Wu ZN. Numerical dissipation for three-point difference approximations to hyperbolic equations with uneven meshes. *Journal of Computational Mathematics*, accepted.

