Anisotropic Cartesian Grid Adaptation

Paulus R. Lahur, School of Engineering, Nagoya University, E-mail: lahur@aero4.nuae.nagoya-u.ac.jp
Yoshiaki Nakamura, Dept. of Aerospace Eng.. Nagoya University. E-mail: nakamura@nuae.nagoya-u.ac.jp
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Key Words: Anisotropic Cartesian Grid, Grid Adaptation

A grid adaptation method using anisotropic Cartesian grid in three dimensions has been developed to improve the efficiency of an existing Cartesian grid adaptation by reducing the total number of cells needed to resolve flow features. The method is capable of coarsening and refining a grid in such a way that the cell aspect ratio can take an arbitrary value. Flow computations with the present method for a supersonic flow around a cylinder and a transonic flow around an ONERA M6 wing are presented here, which show good agreement with experiment. For the case of cylinder, the present method produces about 67 times fewer cells than the isotropic approach. This dramatic saving is due to the ability of the three-dimensional code employed here to treat cells without any limitation on their aspect ratio. For the case of ONERA M6, the ratio of saving is about 2.5. Though the amount of memory per cell is higher than that of the isotropic approach, the overall memory requirement becomes lower, due to reduction in the number of cells. This means that the present method can make flow computation more efficient.

1. Introduction

Recently, Cartesian grids have become popular in computing the flow around a complex geometry due to their fast and automatic grid generation. Furthermore, they allow for simple local clustering of grid cells, which is particularly advantageous in performing solution adaptive for a realistic geometry. Grid refinement on a Cartesian grid is usually achieved by dividing a cell isotropically into eight sub-cells of equal size in 3D.

However, isotropic refinement results in an excessive number of cells because flow features are usually anisotropic in nature, as in the case of shock wave. To overcome this weakness, a grid that allows for anisotropic grid refinement should be employed.

Anisotropic Cartesian grid adaptation consists of two main processes: grid refinement and coarsening. At present, other researches in this particular area consider either grid refinement, as in Refs. [1], [2], and [3], or coarsening only, as in Ref. [4]. In the present study both processes are carried out, where no restrictions are imposed on the cell aspect ratio. The only restriction is concerning the minimum length of cell side. Calculated results of supersonic flow around a cylinder and transonic flow around a standard ONERA M6 wing will be described to evaluate the present algorithm.

2. Anisotropic Grid Adaptation

The main steps of the present method are shown in Fig. 1. The procedure starts with a rough grid distribution, which is generated isotropically around a body. First, on this grid the flow is solved up to a certain degree of convergence, and then the grid is improved based on the calculated result. Thus, one adaptation cycle is completed. The grid adaptation process consists of grid coarsening, refinement, and smoothing. The following sub-sections discuss the main aspects of the anisotropic grid adap-tation method proposed here.

2.1. Adaptation Parameter

The adaptation parameter employed here is a modified second difference of selected components of the flow solution vector, which is calculated for each cell in the x, y and z directions. The computation is actually carried out as in Eqs. 1 and 2b, where only x direction is shown for simplicity. Incidentally, Eq. 2a is a common formula to compute second derivative.

However, for our purpose, Eq. 2a is modified to Eq.2b. The first modification is the use of length scale L. Increasing its value will refine a large cell, but if its value is too high, it will create too many cells in smooth flow regions without sufficient cell clustering around important flow features. On the other hand, decreasing the value of L will refine a small cell, but its too low value will create a grid with abrupt spatial change in cell's relative size, which degrades solution accuracy. It is found that the value of L between around 2 and 3 gives satisfactory results for the test cases presented here.

The second modification is the use of a maximum of first solution derivatives at the cell face. It is necessary to increase the value of G for a cell with many neighbors, so that it is more likely to be refined.

$$\left(Q'_{ij}\right)^*_x = \left(Q_j - Q_i\right)/\left(x_j - x_i\right) \tag{1}$$

$$\left(G_i\right)_x = \left(\Delta x_i\right)^2 \left[\left(Q'_{ij}\right)^*_{x,right} - \left(Q'_{ij}\right)^*_{x,left}\right]/\Delta x_i \tag{2a}$$

$$\left(G_i\right)_x = \left(\Delta x_i\right)^L \left[\left(Q'_{ij}\right)^*_{x,max} - \left(Q'_{ij}\right)^*_{x,min}\right]/\Delta x_i \tag{2b}$$

where Q is the flow solution. x and Δx are the coordinate of centroid and the cell size in a given direction, respectively. Subscript i indicates the cell under consideration, and j its neighboring cell. The asterisk indicates cell face. L is a length scale that determines the balance of refinement between small and large cells.

Having obtained adaptation parameter G for all cells, its mean value and standard deviation (sdev) are calculated to determine threshold values for grid coarsening and refinement, T_{low} and T_{high}, respectively, as shown in Eqs. 3 and 4. A cell with $G < T_{low}$ in a certain direction is a candidate for coarsening in that direction, while that with $G > T_{high}$ is a candidate for refinement.

$$T_{low} = \text{mean}(G) - R_{low} \cdot \text{sdev}(G) \quad (3)$$

$$T_{high} = \text{mean}(G) + R_{high} \cdot \text{sdev}(G) \quad (4)$$

where R_{low} and R_{high} are the parameters defined by users.

2.2. Grid Coarsening

Grid coarsening is performed by removing the interface between cell i and its neighbor j if the following conditions are satisfied.

$$(G_i)_n < T_{low} \text{ and } (G_j)_n < T_{low} \quad (5)$$

where n is the direction normal to the interface.
In general, a cell may also be a candidate for coarsening in several directions, in which case priority is given to the direction where the value of G is the lowest. It should be noted that the actual coarsening is carried out only when the two faces match exactly with each other. This is necessary to preserve the concept of Cartesian grid cell.

2.3. Grid Refinement

A cell is refined by dividing it into two parts in a direction where the value of G is higher than the upper threshold value, T_{high} in that direction. As a result, a cell can be refined in several directions, as shown in Fig. 2. For practical reasons, there are limitations on the minimum cell size and the maximum number of cells in this study.

2.4. Grid Smoothness

After the coarsening and refinement mentioned above, the cell size and orientation are not smoothly distributed. Of particular concern is the cell size relative to its neighbors, because large variation in size may degrade solution accuracy. Hence, it is necessary to smooth out the grid. The smoothing process, however, is rather time-consuming, due to the problem of neighbor searching in unstructured grid. Thus, it is preferred that the grid is already reasonably smooth before implementing the smoothing process, so that computational effort is kept to minimum. This is obtained by setting R_{low} and R_{high} in Eqs. 3 and 4 to high values, so that only cells which definitely have to be refined or coarsened are modified. However, setting the parameter to a too high value may cause insufficient grid adaptation, which requires more adaptation cycles. A reasonable balance is found when the values of R_{low} and R_{high} is set to 0.5.

The grid smoothness is defined in this study that the number of neighboring cells is limited to a maximum of two in each of two directions along the cell face. If this condition is violated, the cell is divided into two cells in the direction with too many neighbors. The division is made in such a way that the cell boundary matches that of its neighbors. Thus, this method can successfully generate a reasonably smooth grid.

2.5. Data Structure

In generating a grid, where its cell has an arbitrary value of the aspect ratio, an unstructured approach is adopted in this study. Here each cell stores the integer coordinates to locate its vertices. Unlike floating point value, the integer representation is accurate, so that a cell can find its neighbors easily.

A problem of this approach is that the cell searching process is linear, which takes excessive time for the case of a large number of cells. To overcome this problem, an ADT (Alternating Digital Tree) is employed at grid genera-tion and adaptation.[5] ADT is particularly suitable for storing and searching a finite-sized object in multi-dimensional space. A well-balanced ADT can reduce the searching task to a log N process, where N is the number of cells

3. Grid Generation and Flow Solver

The Cartesian grid generation and the flow solver employed here are basically the same as those reported previously by the authors (see Ref. [6]). The Euler equations are solved to calculate inviscid, compressible flows. The flow solver is based on a cell-centered finite volume scheme, where the numerical flux is computed using Hännel's flux-vector-splitting scheme.[7] The MUSCL method is used to compute the flux with second order accuracy. Time integration is carried out with a 3-stage Runge-Kutta method, where local time stepping is applied to accelerate the convergence rate. To make the time step size as large as possible, an extremely small cell at the solid boundary is merged with the largest of its neighbors.

4. Test Cases

4.1 Cylinder in Supersonic Flow

The first test case is a circular cylinder with a unit radius in a supersonic flow of M=3.0. The objective is two-fold. The first is to test applicability of the present 3D method to a 2D flow problem, since such 2D flow corresponds to a kind of anisotropic grid, where it is uniform in the spanwise direction. That is, all cells extend to the whole span region. The second objective is to examine whether the method can efficiently capture flow features of shocks and wake, as well as smooth flow regions.

First, an isotropic grid is generated around half of the cylinder, which contains 46,480 cells. This is quite a large number for this kind of 2D computations. The initial grid used in this test case is obtained by merging all cells of the isotropic grid in the spanwise direction, which results in 1,643 cells. The upstream, downstream and upper boundary conditions are set to free flow, whereas at the left, right and lower boundaries conditions are imposed.

The anisotropic grid adaptation is carried out using density and mach number as sensor parameters. The length scale L in Eq. 2b is set to 0.3. The initial, intermediate and final grids are shown in Fig. 3, and the corresponding flow solutions are shown in Fig. 4. The number of adaptation cycle is 9, beyond which the grid and solution change little.

It is observed that the grid adaptation quickly and sharply captures flow features such as shocks by refining the cells in these regions. At the 6th adaptation cycle, the finest cells at the main shock already stretch all the way to the downstream computational boundary, which is placed at a distance 10 times as large as the radius from the center of the cylinder. Cell refinement also takes place in the regions of small variation, such as behind the main shock, but at a slower pace, i.e. at the 9th adaptation cycle in this study. It is observed that no grid modifications are made in the spanwise direction throughout the whole adaptation process. This means that the grid and flow solution remains two-dimensional. Grid coarsening in the constant flow region upstream of the bow shock is carried out in one cycle. All grids show a reasonably smooth transition between large and small cells, which can be seen clearly in the regions around the shocks.

Figure 4c shows the final result, where the main shock, the shear flow behind the cylinder, and the shocks interacting with the wake are sharply captured. The main shock standoff distance is 0.7 for the unit radius, which is quite reasonable when compared with the empirical data of 0.65 for a wedge with a cylindrical head.[8] Moreover, the density jump across the shock is 3.878, which compares well with a theoretical value of 3.857.

Comparison of the number of cells between anisotropic and isotropic grids shows that even the final anisotropic grid contains considerably fewer cells than the initial isotropic grid. As the grid becomes more refined, the ratio of the number of cells in the isotropic grid to that in the anisotropic grid becomes even larger. In the final grid with 14,431 cells, the ratio reaches 67, which means that the equivalent isotropic grid contains about 970,000 cells.

The advantage of the anisotropic grid is evident in the requirement of the total memory, which in this study consists mainly of solution vector, flux vector, cell volume, cell face area, and face-based database. Even though the amount of memory per cell is 400 bytes for the anisotropic grid, the total memory is only 5.8MB for the final grid. On the other hand, the isotropic grid requires only 344 bytes per cell, but the total memory is 333MB, which is about 57 times larger. Thus, the 3D anisotropic grid adaptation method can compute a 2D flow far more efficiently than the isotropic approach.

4.2 ONERA M6 Wing in Transonic Flow

The second test case is transonic flow over ONERA M6 wing.[9] The objective is to apply the present method to a 3D transonic flow. The wing has ONERA D profile, with an aspect ratio of 3.8 and a taper ratio of 0.562. The leading and trailing edges sweep back at an angle of 30° and 15.8°, respectively. The flow condition is $M=0.84$ and $\alpha=3.06^\circ$, which is widely used for CFD validation.

Density is used to compute the adaptation parameter. The length scale L in Eq. 2b is set to 2.0, which enables us to capture two shocks on the upper surface of the wing quickly. Setting L to 3.0 as in the cylinder case will take more adaptation cycles to capture the shocks. As mentioned before, an isotropic grid is used as a starting grid. The adaptation is carried out twice, beyond which the solution on the wing changes little. The initial and adapted grids are shown in Figs. 6 and 7, where the grid is clustered in the regions of shocks on the upper surface and the leading and trailing edge regions.

The pressure distribution on the upper surface of the wing for each grid is shown in Fig. 9, which clearly depicts a lambda pattern of the shock. As the grid adaptation cycle is repeated, the pattern becomes more distinct. A more detailed view of the pressure distribution on the wing is shown in Fig. 8 at selected cross sections. It is evident here that the pressure distribution changes little after the second adaptation cycle. The final solution shows a good agreement with the experimental data, except for differences in shock locations and sharpness, due to the inviscid nature of the flow solver employed here.

As an additional comment, it is observed that when an anisotropic Cartesian grid is coupled with a first order flow solver, as in our previous study, the shock near the leading edge fails to appear, even on a fine grid. Therefore, it seems that at least a second-order scheme is required to capture the pressure jump near the leading edge.

The change in the number of cells during the adaptation process is shown in Fig. 10. The final anisotropic grid contains 171,753 cells, which is equivalent to an isotropic grid with 430,260 cells. This is 2.5 times as large as that of the anisotropic grid.

In terms of the total memory required, although not as dramatic as the cylinder test case, the anisotropic grid is still significantly better than the isotropic grid. For the final grid, the total memory for the case of anisotropic grid is 68.7MB, whereas that of the isotropic grid is 148MB, which is 2.2 times as large as that of the anisotropic grid.

5. Concluding Remarks

An anisotropic Cartesian grid adaptation method has been developed in this study to improve the efficiency of an existing Cartesian grid adaptation. It has capabilities to both coarsen and refine a Cartesian grid in any direction without any limitation on cell aspect ratio, while keeping the grid smooth.

The present method was validated for 2D and 3D test cases: a supersonic flow around a cylinder and a transonic flow around ONERA M6, respectively. The method successfully captured the flow features and improved the solution with significantly less number of grid cells, as compared to the corresponding isotropic grid. The 2D flow was computed with very high relative efficiency, since the number of cells are as few as those needed by a 2D method. This is achieved by imposing no limitations on cell aspect ratio, so that a cell can span the whole computational domain in the spanwise direction. In the final grid, the number of cells in the isotropic grid is 67 times as large as that of the anisotropic grid, whereas the total memory is 57 times.

For the ONERA M6 test case, the number of isotropic cells is 2.5 times as large as that of the anisotropic grid, whereas the total memory is 2.2 times. From the above-mentioned, a flow containing more anisotropic features will benefit more from the anisotropic adaptation.

For future research, extension of the current method to more complicated situation such as moving boundary is being considered. For this purpose, the adaptation strategy as well as the data structure will have to be reorganized to improve their efficiencies.

References

(1) Aftosmis, M.J., "Solution Adaptive Cartesian Grid Mehods for Aerodynamic Flows with Complex Geometries," von Karman Institute for Fluid Dynamics, Lecture Series 1997-02 (1997).

(2) Welterlen, T.J., "Store Release Simulation on the F/A-18C using Splitflow," AIAA paper 99-0124 (1999).

(3) Wang, Z.J., Cphen, R.F., Hariharan, N., Przekwas, A.J., and Grove, D., "A 2^N Tree Based Automated Viscous Cartesian Grid Methodology for Feature Capturing," AIAA paper 99-3300 (1999).

(4) Deister, F., Rocher, D., Hirschel, E.H., and Monnoyer F., "Three-dimensional Adaptively Refined Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries," Proceeding of the 4th European CFD Conference, Vol. 1, Part 1 (1998), pp. 96-101.

(5) Bonet, J. and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," Int. J. for Num. Methods in Eng., Vol. 31 (1991), pp. 1-17.

(6) Lahur, P.R., and Nakamura, Y., "A New Method for Thin Body Problem in Cartesian Grid Generation," AIAA paper 99-0919 (1999).

(7) Hännel, D., Schwane, R., Seider, G., "On the Accuracy of Upwind Schemes for the Solution of the Navier-Stokes Equations," AIAA 87-1105, Proc. AIAA 8th Computational Fluid Dynamics Conference (1987), pp. 42-46.

(8) Ambrosio, A. and Wortman, A., "Stagnation Point Shock Detachment Distance for Flow around Spheres and Cylinders," ARS J. 32, 281 (1962).

(9) Schmitt, V. and Charpin, F., "Pressure Distributions on The ONERA-M6-Wing at Transonic Mach Numbers," Experiment Data Base for Computer Program Assessment, AGARD AR-138 (1979).

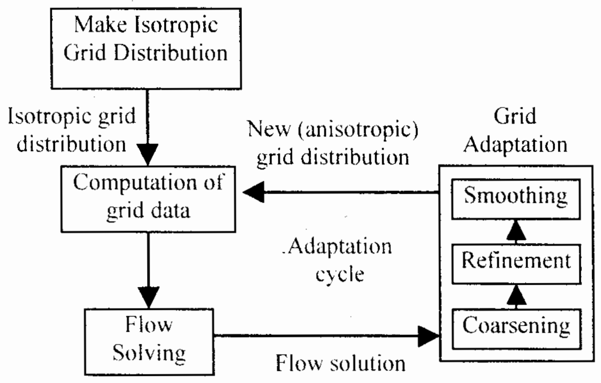

Fig. 1 Grid adaptation cycle.

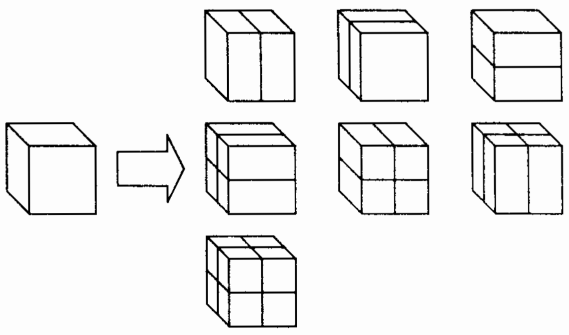

Fig. 2 Possibilities of grid refinement.

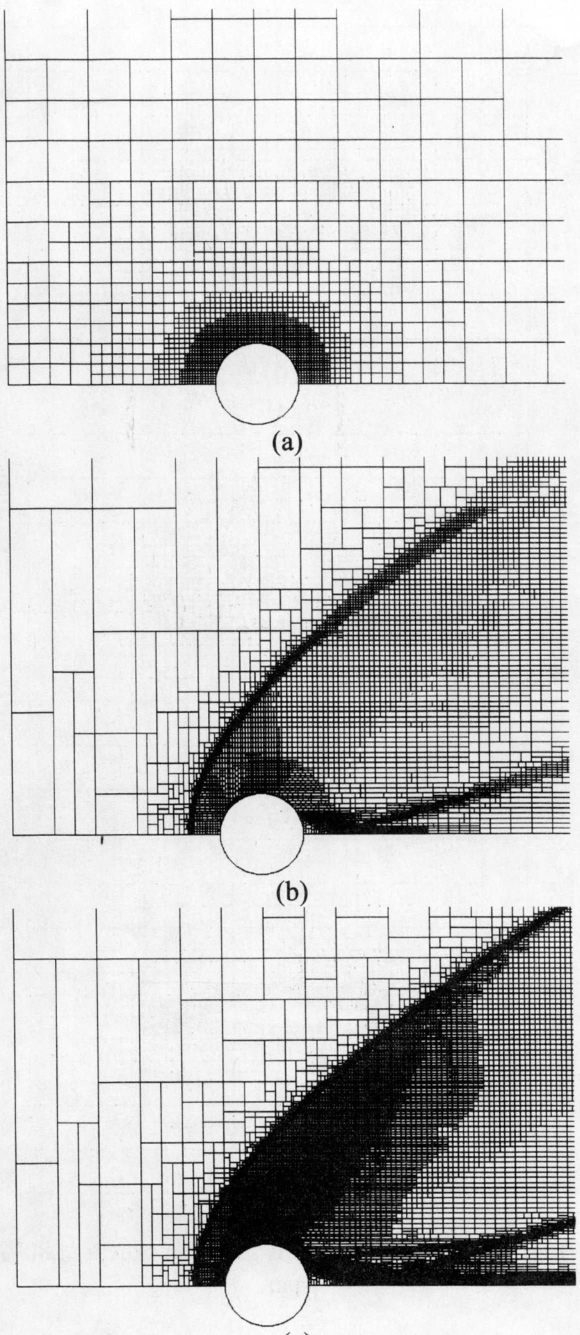

Fig. 3 Grid around cylinder: (a) at initial, (b) 4th adaptation cycle, (c) 9th.

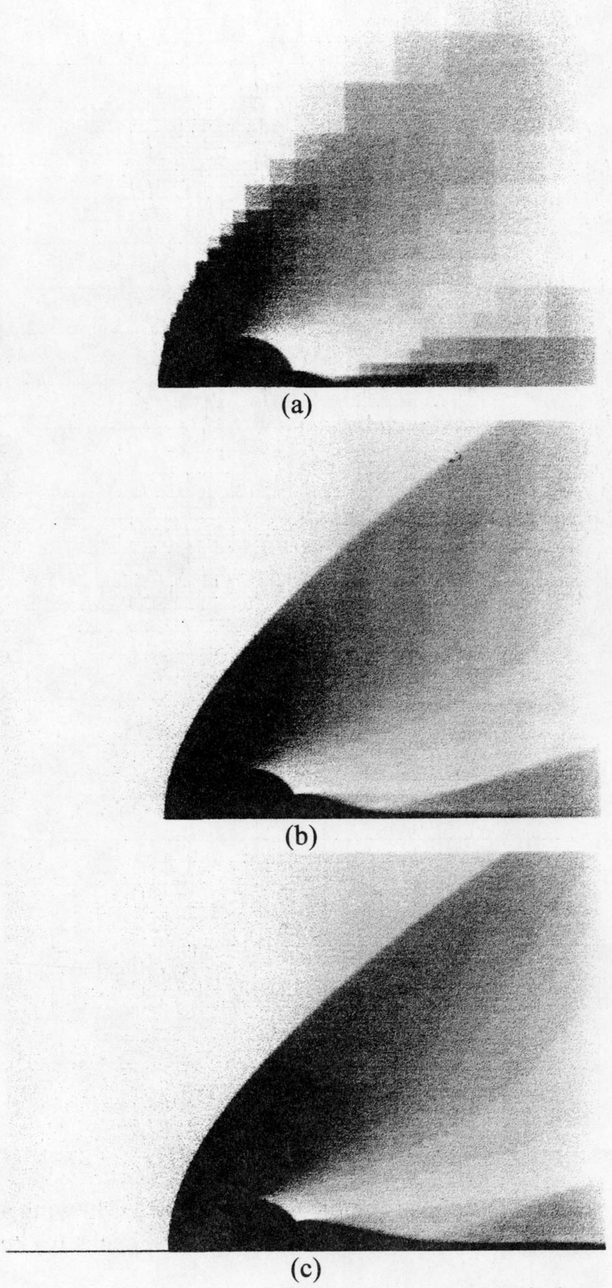

Fig. 4 Mach number distribution around cylinder: (a) at initial, (b) 4th adaptation cycle, (c) 9th.

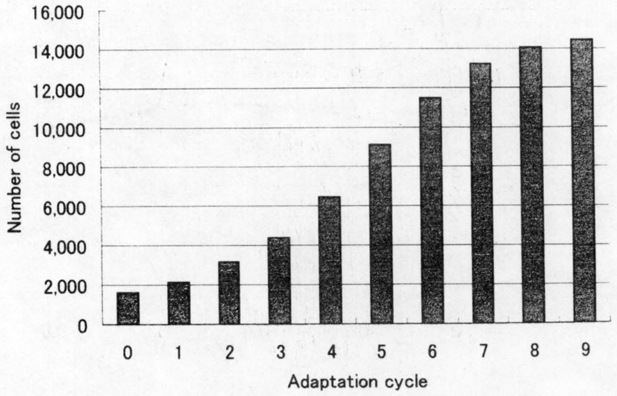

Fig. 5 Change in the number of cells with adaptation cycle for flow around cylinder.

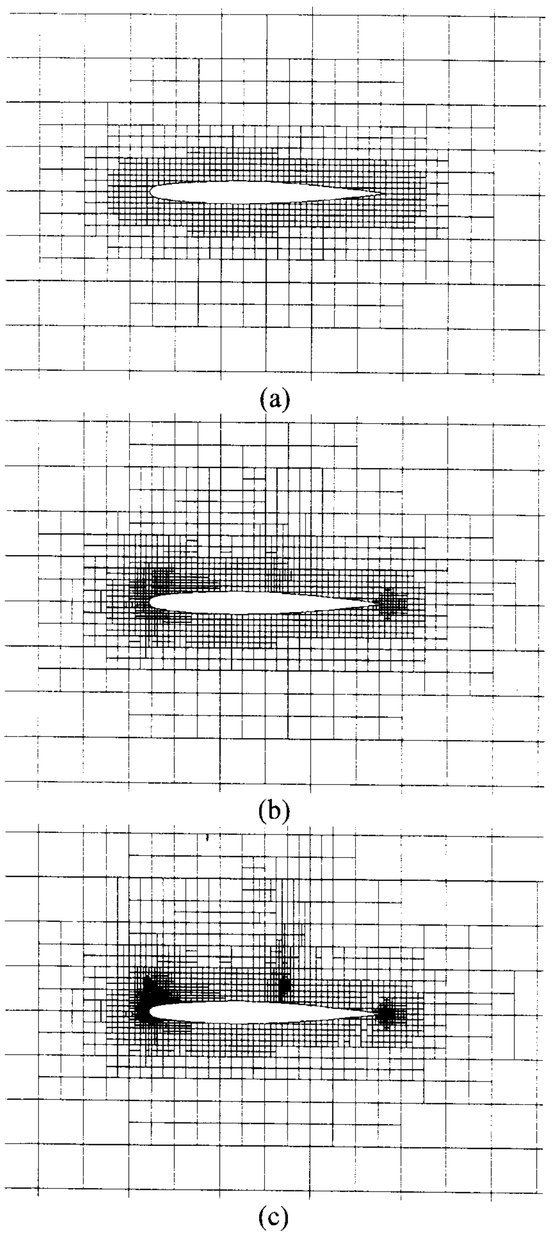

Fig. 6 Grid distribution around ONERA M6 wing at 44% span: (a) at initial, (b) 1st adaptation cycle, (c) 2nd.

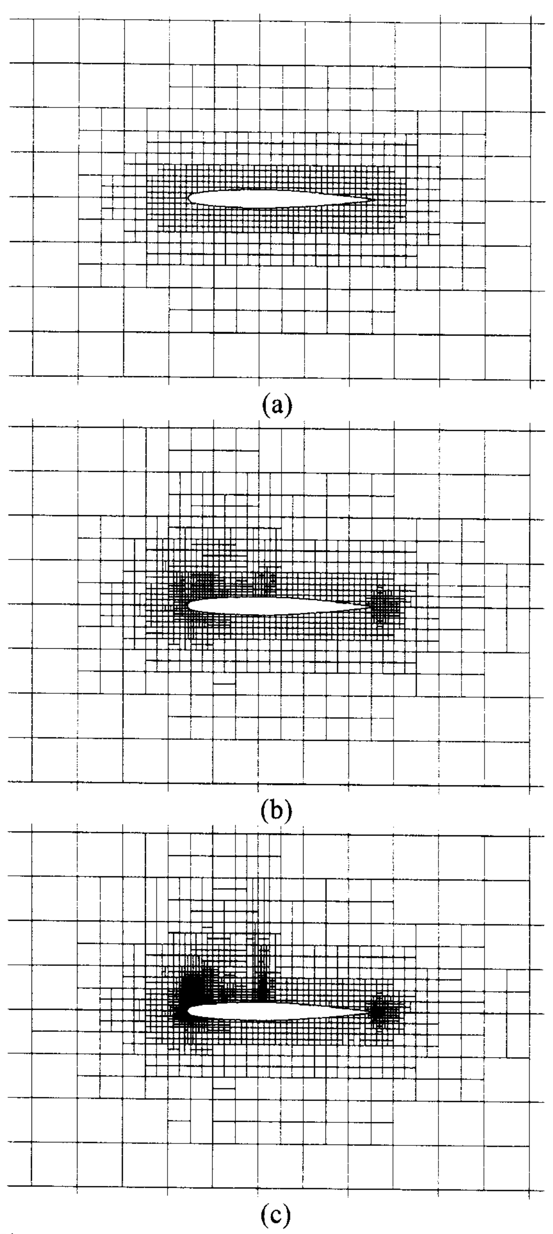

Fig. 7 The same captions as Fig. 6 except at 80% span.

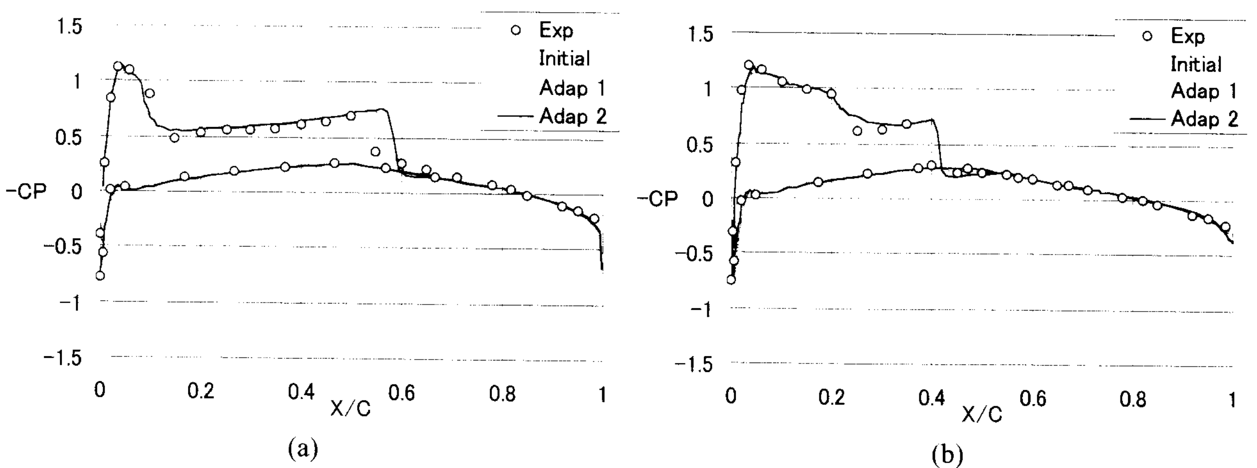

Fig. 8 Pressure distribution around ONERA M6 wing: (a) 44% span, (b) 80%.

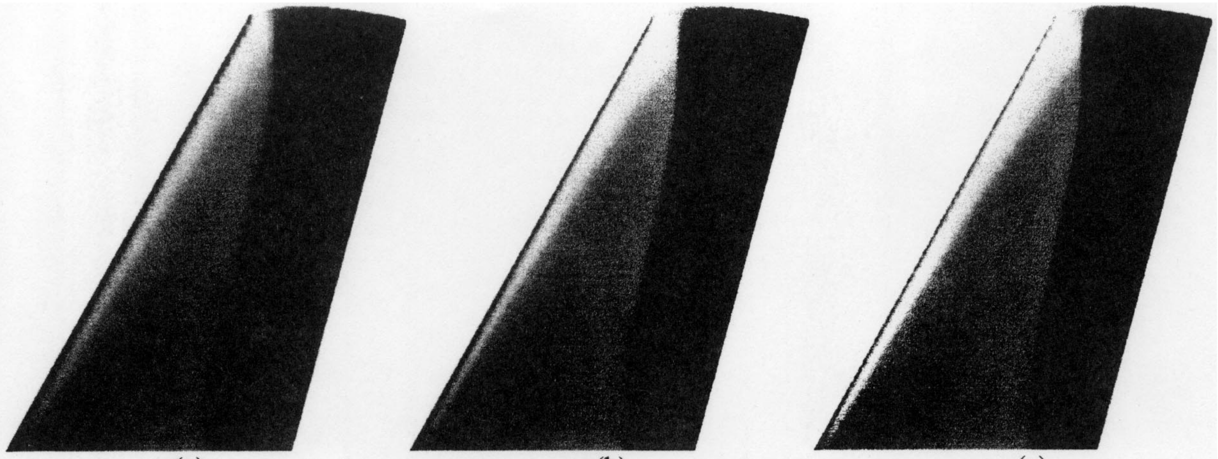

Fig. 9 Pressure distribution on the upper surface of ONERA M6 wing: (a) at initial, (b) 1st adaptation cycle, (c) 2nd.

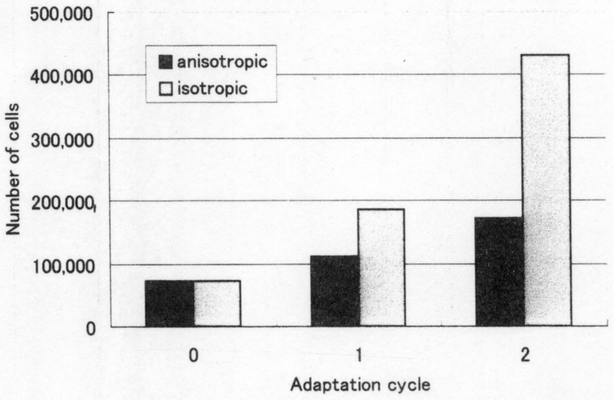

Fig. 10 Change in the number of cells with adaptation cycle for flow around ONERA M6.