Journal of
Mechanical
Science and
Technology

Journal of Mechanical Science and Technology 21 (2007) 1306~1319

High-Density Mesh Flow Computations by Building-Cube Method

L. S. Kim*, K. Nakahashi, H. K. Jeong, M. Y. Ha

School of Mechanical Engineering, Pusan National University, San 30, Jangjeon-Dong, Kumjung-Ku, Busan 609-735, Korea

(Manuscript Received March 6, 2007; Revised April 26, 2007; Accepted April 8, 2007)

Abstract

With the conventional computational fluid dynamics (CFD) approaches like unstructured high-density mesh method, the problem about the solution dependencies on the grid and on the physical models isn't completely resolved. In this study, thus, a new approach named Building-Cube Method based on a Cartesian mesh is proposed using a high-density mesh in order to solve the problem of the current CFD approaches. In the present method, a flow field is divided into a number of cubes (squares in 2D) of various sizes. Each cube is a computational sub-domain with Cartesian mesh of equal spacing and equal number of nodes. The geometrical size of each cube is determined by adapting to geometry and flow features so as to cope with broadband characteristic lengths of the flow. Equal spacing and equal number of Cartesian mesh in each cube make it easy to parallelize the flow solver and to handle huge data output. The method is applied to several airfoils including NACA0012 and two-element airfoils at relatively low Reynolds number, RAE2822 airfoil with transition trip at high Reynolds number and a four-element airfoil at high Reynolds number. These results validate the capability of the present approach.

Keywords: Building-cube method; High-destiny method; Cartesian mesh; Adaptive mesh; Parallel computation

1. Introduction

CFD has become an indispensable tool for designing an airplane. However, wind tunnel experiments are still required to confirm the computed results. For flows at off-design conditions, the experiment is the central player and the CFD plays a subordinate part. This situation is mainly because of the lack of the reliability in the current CFD due to the dependencies of computed results on the grid and on the turbulence models. To remove the grid dependency, the most straightforward way is to use a highly dense and regular grid. To remove the dependency on physical models, we have to move from the Reynolds-averaged Navier-Stokes equations to LES and DNS. This also means to use a high-density mesh and a spatially higher-order numerical scheme (Mellen et al., 2003).

Meanwhile, the advancement of the computer performance is expected to keep the current pace for a while. The allowable mesh size will become an order of one giga ($1x10^9$) points soon, and will be one tera points in near future. To meet the near-future expectation of the advanced computer with a large number of high performance processors and huge memories, however, a simple extension of the conventional numerical scheme will be stuck soon. CFD using structured meshes has a difficulty in the mesh generation for 3D complex geometries. CFD using unstructured meshes can now treat really complex geometry (Nakahashi et al., 2003), but has a difficulty in constructing a higher-order numerical scheme in space. Both approaches will also have a

*Corresponding author. Tel.: +82 51 510 3090, Fax.: +82 51 515 3101
E-mail address: lskim12@pusan.ac.kr

difficulty in the post processing of huge data produced by the large-scale computations on near-future computers. For the next-generation CFD applicable for large-scale LES/DNS computations, the simplicity of the numerical algorithm in all aspect of the computations, from the mesh generation to the post processing, is most important.

The requirements for the next generation CFD method are; (1) capability to treat complex geometries, (2) capability to use locally dense grid fitted to the local characteristic flow length, (3) easy adaptive refinement, (4) easy construction of higher-order scheme, (5) efficient use of parallel computers, (6) easy post processing for extremely large-scale computations.

The final goal of the present study is to construct a next-generation CFD that does not need to worry about the grid and physical model dependencies. To achieve this goal, a new approach named 'Building-Cube Method' (Nakahashi, 2003), is developed in this study.

2. Building-Cube Method

The Building-Cube Method (BCM) basically employs a uniform-spacing Cartesian mesh because of the simplicity in the mesh generation, the solution algorithm, and the post processing. The simplicity will become more important for large-scale computations in future.

The most critical problem of the conventional Cartesian mesh, however, is how to fit to the local characteristic flow length without introducing the algorithm complexity. Although the Cartesian mesh approach with the adaptive mesh refinement and the cut-cell near the boundaries shows the high capability for complex geometries, the introduction of irregular subdivisions into the Cartesian grid complicates the algorithm and increases the memory requirement. Therefore this approach is basically same with the unstructured mesh approach and the advantages of the Cartesian mesh over the unstructured grid, such as simplicity and less memory requirement, may disappear.

Here, a flow field is divided into a number of sub-domains, named 'Cube' as shown in Fig. 1. The geometrical size of each cube is determined by adapting to the geometry and the flow features. In each cube, a uniform-spacing Cartesian mesh is used. All cubes have the same number of Cartesian mesh so

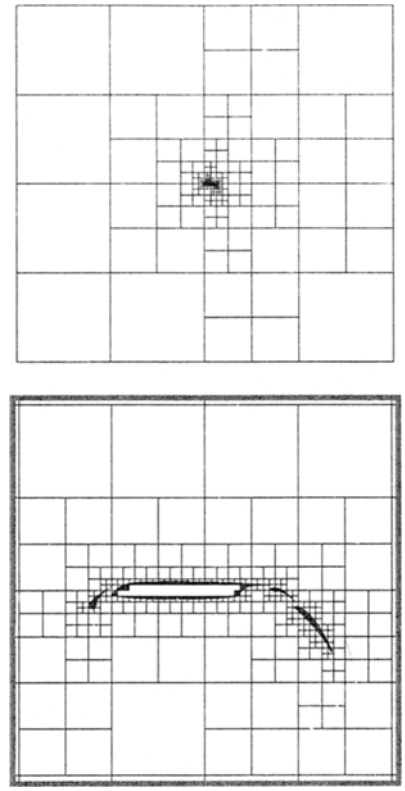


Fig. 1. Cube boundaries of the BCM mesh around a four-element airfoil; the entire view (upper) and the enlarged view. (lower)

that the local computational resolution is determined by the cube size. The same mesh density among all cubes also simplifies the parallelization.

The wall boundary is defined by a staircase representation in order to keep the simplicity of the algorithm and to minimize the memory requirement per node. To keep the geometrical accuracy by the staircase representation of wall boundaries, very fine mesh that resolves viscous sublayers in the boundary layers is used. With this mesh size, the staircase representation is accurate enough to approximate the wall boundary as shown later. To use extremely fine mesh is basically imperative for the DNS which is the final goal of the present study.

With this approach, it is capable (1) to treat complex geometries, (2) to use locally dense grid fitted to the local characteristic flow length, (3) to implement a higher-order scheme, (4) to use parallel computers, (5) to treat extremely huge data in the post-processing. These considerations are summarized in Table 1.

Table 1. Requirements for the next-generation CFD method and the compatibility of the current approaches.

Approaches / Requirements		structured mesh	unstructured mesh	Cartesian mesh		
				uniform spacing	with adaptive mesh refinement	Building-Cube Method
(1)	Capability to treat complex geometries	Poor	good	good	excellent	excellent
(2)	Capability to use locally dense mesh adapted to the local characteristic flow length	Good	excellent	poor	excellent	excellent
(3)	Easy construction of spatially higher-order scheme	Good (3rd ~)	fair (at most 2nd order)	excellent	fair	excellent
(4)	Efficient use of parallel computers	Good	good	good	fair	excellent
(5)	Easy post processing for huge computational data	Good	poor	good	poor	excellent

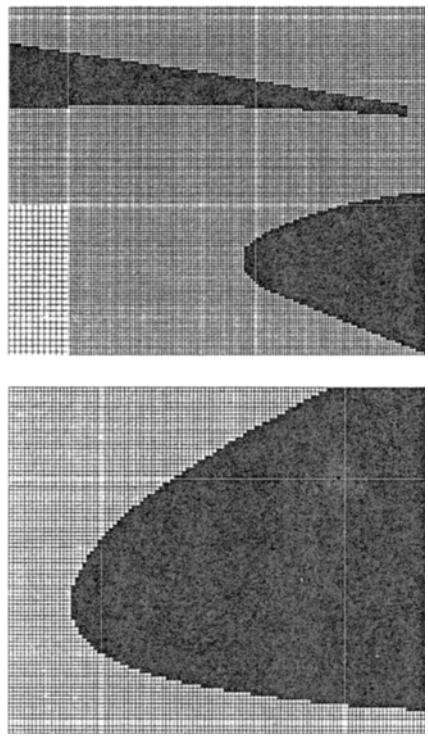


Fig. 2. Enlarged views of the BCM mesh around a four-element airfoil, showing 64x64 Cartesian mesh in each cube and the staircase representation of the wall boundaries.

3. Generation of 2D BCM mesh

The BCM-mesh generation of the present method has two steps. The first step is a cube generation that is to fill in a flow field with various sizes of cubes as shown in Fig. 1. The second step is to generate Cartesian mesh in cubes that locate near the body (Fig. 2).

3.1 Division of a Computational domain

The cube-frame is generated by the geometry-adaptive refinement method (Nakahashi and Egami, 1991), which is similar to the quadtree/octtree method commonly used for generating Cartesian and unstructured meshes. At first, a computational domain is divided into a coarse Cartesian grid with equal mesh spacing in all directions. Then, cubes (squares for 2D) that include or cross the body boundary are divided into eight cubes (four squares for 2D). This refinement procedure is repeated until the minimum cube size becomes smaller than a specified value. After this refinement procedure, cubes that locate completely inside of the body are removed.

The next procedure is a smoothing of the size differences among cubes. At present, the size difference between two adjacent cubes is limited to two in order to minimize the data transfer error between cubes. For this, a cube whose adjacent cube is smaller than a quarter of the own size is detected and divided.

Figure 1 is the entire view of the cube boundaries around a multi-element airfoil and the enlarged view with Cartesian mesh in cubes is shown in Fig. 2.

3.2 Generation of Cartesian mesh in each cube

Cartesian-mesh generation is not required for most of the cubes because the Cartesian mesh inside of each cube is simply defined by the number of division along the cube edge. However, cubes that include the wall boundary in them must have information about the wall boundary location.

In the wall cube, Cartesian mesh is generated and then each cell of the Cartesian mesh is checked

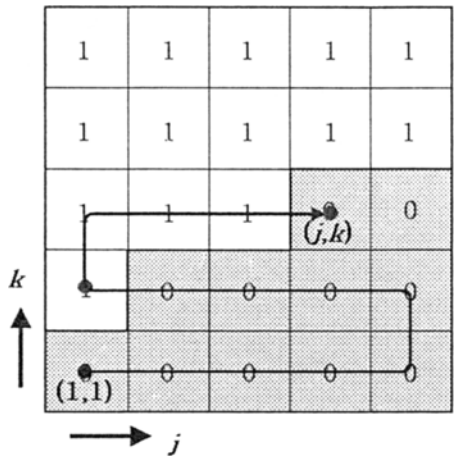


Fig. 3. Cell flags and the runlength data line in j-k plane.

whether it is located inside or outside of the body. Those cells inside of the body are marked 0, and those outside of body are marked 1. The body boundary is defined by the cell boundaries between adjacent cells having different marks as shown in Fig. 3.

3.3 BCM mesh data

For flow computations using a high-density mesh, the size of the mesh file may become very large. This is especially true for the unstructured mesh because the unstructured mesh has to have the connection data as well as the coordinate data of node points. For large scale computations, the huge size of the mesh file sometimes imposes the time-consuming pre- and post-processings. One of the important advantages of the Cartesian mesh is that the mesh data is simple and compact. This advantage must be taken over by the present method.

The BCM mesh has the following data;

(1) number of cubes,

(2) size and x, y, z coordinates of the front-lower-left corner for each cube,

(3) cube-neighboring information for each cube,

(4) number of division of a cube,

(5) in-out information on cells in each wall cube.

Data from (1) to (3) are for the cube mesh, and data (4) and (5) are for the Cartesian mesh in the wall cubes. The number of cubes is about several hundreds to at most thousands for two dimensional computations. Therefore, the data size from (1) to (3) is very small. Even for three-dimensional computations around an airplane expected in near future, the data size of the cube mesh is small. Moreover, the data (3) is not indispensable because it can be easily constructed by the data (2) in the flow solver. However, the data (5) may become very large because it contains the information of all cells in the wall cubes.

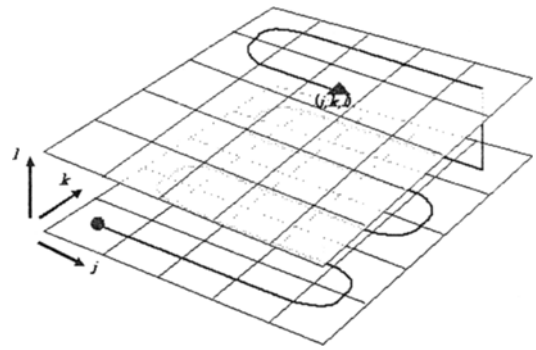


Fig. 4. The runlength data line in j-k-l space.

In order to compress the data (5), the runlength data structure, which is used for the image data compression, is utilized. The original runlength is used mainly to compress a monochrome bitmap data. It describes the number of continues bits in the scanning line of the image. Here the runlength concept is extended to 2D and 3D fields by the following equations.

In the present data, all cells have flags of 0 or 1 as shown in Fig. 3. Therefore, the minimum information is the location where the cell changes the flag value. The location in the 2D field defined by (j, k) as shown in Fig. 3 can be described by the one-dimensional array of i by the following equations.

$$i = n_j(k-1) + pj + (1-p)(n_j + 1 - j), \quad (1)$$

$$p = \mathrm{mod}(k,2), \quad (2)$$

where the n_j is the number of cells in j-direction.

With these equations, the one-dimensional array, say JK_runlength(i), can be as follows.

JK_runlength(1) = number of locations,

JK_runlength(2) = flag value at (1,1),

JK_runlength(i) for i=3, JK_runlength(1)+2 : locations of the flag change.

Decoding of the one-dimensional array is given by,

$$k = INTEGER[(i-1)/n_j] + 1 \quad (3)$$

$$j = [i - n_j(k-1) - (1-p)(n_j+1)]/(2p-1) \quad (4)$$

For 3D field shown in Fig. 4,

$$i = qi_1 + (1-q)i_2 + n_j n_k (k-1) \quad (5)$$

$$i_1 = n_j(k-1) + pj + (1-p)(n_j + 1 - j) \quad (6)$$

$$i_2 = n_j(n_k - k) + p(n_j + 1 - j) + (1-p)j \quad (7)$$

where $p = \mathrm{mod}(k,2)$, $q = \mathrm{mod}(l,2)$.

With this compression, the mesh data size reduces to three-order of magnitude smaller than the conventional data structure. For example, the file size of a mesh about $3\text{x}10^7$ cells (500 cubes with 256x256 cells in each cube) is only 87 Kbytes in ASCII text format

4. Numerical method

4.1 Governing equations

A non-dimensional form of the compressible Navier-Stokes equations can be written in the Cartesian coordinates x_j, (j=1, 2, 3) as

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_j}{\partial x_j} - \frac{1}{Re}\frac{\partial \mathbf{G}_j}{\partial x_j} = 0 \quad (8)$$

where $\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, e]^T$ is the vector of conservative variables, ρ is the density, u, v, w are the velocity components in the x, y, z directions and e the total energy. The vector F(Q) and G(Q) represent the inviscid and viscous flux vector respectively and Re is the Reynolds number.

4.2 Basic numerical scheme

The Navier-Stokes equations are solved by a cell-centered scheme on an equally-spacing Cartesian mesh. For a cube-k, the Cartesian mesh spacing is simply given by $\Delta x_k = \Delta s_k / \mathrm{n_{cell}}$ where Δs_k is the size of the cube-k and $\mathrm{n_{cell}}$ is the number of cells along each coordinate in a cube. This value is same for x, y and z directions in 3D computations. However, for 2D computation shown in this paper, $\mathrm{n_{cell}}$ in y-coordinate is set to be 2 in order to compute the two-dimensional field by the 3D flow solver.

Equation (3.8) can be written in an algebraic form as follows,

$$\frac{\partial \mathbf{Q}_i}{\partial t} = -\frac{1}{\Delta x_k}\left[\sum_{j(i)} \mathbf{h}\left(\mathbf{Q}_{ij}^+, \mathbf{Q}_{ij}^-, \mathbf{n}_{ij}\right) - \frac{1}{Re}\sum_{j(i)} \mathbf{G}\left(\mathbf{Q}, \mathbf{n}_{ij}\right)\right] \quad (9)$$

where the summation $j(i)$ means all six faces around the regular hexahedron cell-i. The term $\mathbf{h}(\mathbf{n}_{ij})$ is an inviscid numerical flux vector normal to the control volume boundary, and $\mathbf{Q}_{ij}^+$, $\mathbf{Q}_{ij}^-$ are values on both sides of the control volume boundary. The numerical flux $\mathbf{h}$ is computed using an approximate Riemann solver of Harten-Lax-van Leer-Einfeldt-Wada (HLL-EW) (Obayashi and Guruswamy, 1995). The primitive variables on the cell interface are evaluated by the third-order MUSCL scheme with the differential limiter or the fourth-order compact MUSCL scheme (Yamamoto and Daiguji, 1993).

The fourth-order compact MUSCL scheme is as follows.

The numerical flux of the MUSCL approach is written in the splitting form

$$\mathbf{h}_{l+1/2} = \mathbf{h}^+(\mathbf{q}_{l+1/2}^L) + \mathbf{h}^-(\mathbf{q}_{l+1/2}^R) \quad (10)$$

The values $\mathbf{q}^L$ and $\mathbf{q}^R$ are calculated as

$$\left.\begin{aligned} \mathbf{q}_{l+1/2}^L &= \mathbf{q}_l + \tfrac{1}{6}(\Delta^* \overline{\mathbf{q}}_{l-1/2} + 2\Delta^* \overline{\mathbf{q}}_{l+1/2}), \\ \mathbf{q}_{l+1/2}^R &= \mathbf{q}_{l+1} - \tfrac{1}{6}(2\Delta^* \overline{\mathbf{q}}_{l+1/2} + \Delta^* \overline{\mathbf{q}}_{l+3/2}), \end{aligned}\right\} \quad (11)$$

where

$$\left.\begin{aligned} \Delta^* \overline{\mathbf{q}}_{l-1/2} &= \min\mathrm{mod}\left[\Delta^* \mathbf{q}_{l-1/2}, b\Delta^* \mathbf{q}_{l+1/2}\right], \\ \Delta^* \overline{\mathbf{q}}_{l+1/2} &= \min\mathrm{mod}\left[\Delta^* \mathbf{q}_{l+1/2}, b\Delta^* \mathbf{q}_{l-1/2}\right], \\ \Delta^* \overline{\mathbf{q}}_{l+1/2} &= \min\mathrm{mod}\left[\Delta^* \mathbf{q}_{l+1/2}, b\Delta^* \mathbf{q}_{l+3/2}\right], \\ \Delta^* \overline{\mathbf{q}}_{l+3/2} &= \min\mathrm{mod}\left[\Delta^* \mathbf{q}_{l+3/2}, b\Delta^* \mathbf{q}_{l+1/2}\right], \end{aligned}\right\} \quad (12)$$

and

$$\Delta^* \mathbf{q}_{j+1/2} = \Delta \mathbf{q}_{j+1/2} - \tfrac{1}{6}\Delta^3 \overline{\mathbf{q}}_{j+1/2}, \quad (13)$$

$$\Delta^3 \overline{\mathbf{q}}_{j+1/2} = \Delta \overline{\mathbf{q}}_{j-1/2} - 2\Delta \overline{\mathbf{q}}_{j+1/2} + \Delta \overline{\mathbf{q}}_{j+3/2}, \quad (14)$$

$$\left.\begin{aligned} \Delta \overline{\mathbf{q}}_{j-1/2} &= \min\mathrm{mod}\left[\Delta \mathbf{q}_{j-1/2}, b_1\Delta \mathbf{q}_{j+1/2}, b_1\Delta \mathbf{q}_{j+3/2}\right], \\ \Delta \overline{\mathbf{q}}_{j+1/2} &= \min\mathrm{mod}\left[\Delta \mathbf{q}_{j+1/2}, b_1\Delta \mathbf{q}_{j+3/2}, b_1\Delta \mathbf{q}_{j-1/2}\right], \\ \Delta \overline{\mathbf{q}}_{j+3/2} &= \min\mathrm{mod}\left[\Delta \mathbf{q}_{j+3/2}, b_1\Delta \mathbf{q}_{j-1/2}, b_1\Delta \mathbf{q}_{l+1/2}\right], \end{aligned}\right\} \quad (15)$$

and

$$1 < b \le 4. \quad (16)$$

The lower/upper symmetric Gauss-Seidel (LU-SGS) implicit method is used for the time integration. In order to keep the time accuracy, the following sub-

iteration scheme (Matsuno, 1993) developed for the higher-order time accuracy is employed. It can be briefly described by the following simplified equation.

$$\frac{\partial \mathbf{Q}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}=0 \tag{17}$$

A typical implicit scheme can be written as,

$$\left[\frac{1}{\Delta t}+\theta\frac{\partial}{\partial x}\mathbf{A}^{n}\right]\Delta\mathbf{Q}^{n}=-\left(\frac{\partial \mathbf{F}}{\partial x}\right)^{n} \tag{18}$$

where $\Delta\mathbf{Q}^{n}=\mathbf{Q}^{n+1}-\mathbf{Q}^{n}$, $\mathbf{A}$ is the Jacobian of $\mathbf{F}$, and θ is 1 for fully implicit scheme of the first order, and 0.5 for the second order time accuracy.

Let superscript (ν) be the sub-iteration index, we can rewrite the $\Delta\mathbf{Q}^{n}$ as,

$$\Delta\mathbf{Q}^{n}=\mathbf{Q}^{n+1}-\mathbf{Q}^{n}=\mathbf{Q}^{(\nu)}+\delta\mathbf{Q}^{(\nu)}-\mathbf{Q}^{n} \tag{19}$$

where $\delta\mathbf{Q}^{(\nu)}=\mathbf{Q}^{(\nu+1)}-\mathbf{Q}^{(\nu)}$ is the correction value for the time accuracy. The final implicit scheme, then, can be written as,

$$\begin{aligned}\left[\frac{1}{\Delta t}+\theta\frac{\partial}{\partial x}\mathbf{A}^{(\nu)}\right]\delta\mathbf{Q}^{(\nu)}=&-\frac{1}{\Delta t}\left[\mathbf{Q}^{(\nu)}-\mathbf{Q}^{n}\right]\\&-\theta\left(\frac{\partial \mathbf{F}}{\partial x}\right)^{(\nu)}-(1-\theta)\left(\frac{\partial \mathbf{F}}{\partial x}\right)^{n}\end{aligned} \tag{20}$$

In the present computation, fully implicit scheme ($\theta=1$) of the first-order time accurate was used to save the memory. The sub-iteration for the time accuracy is also utilized to update the values on the cube boundaries.

The memory requirement of the present flow solver is small. The required arrays for the entire flow field are only for the five flow variables in two time planes. This makes the total memory requirement to be about 10 to 12 words per node. For example, if each cube has 256x256 cells and the total number of cubes is 500, the total number of computational cells is 32,768,000. This is 1.3Gb memory (4 byte per words). If the higher-order time accuracy is employed, solutions on one or more time planes are required. However the total memory requirement is still relatively small.

4.3 Boundary conditions

The wall boundary is defined by a staircase representation as shown in Fig. 3 in order to keep the simplicity of the algorithm and to minimize the memory requirement per node. To keep the geometrical accuracy by the staircase representation of wall boundaries, very fine mesh that resolves viscous sublayers in the boundary layers is used. Although this needs an excessive mesh density, it is a straightforward approach to realize the DNS around a complex geometry in future.

The body boundary is defined by the cell boundaries between adjacent cells having different marks as shown in Fig. 3. The density and pressure at the ghost cell, which locates inside of the body and next to the flow-cells, are given by the following manner.

$$\rho_{ghost}=\sum_{i\,(\text{adjacent cells})}(\rho_i * flag_i)/\sum_{i\,(\text{adjacent cells})}flag_i \tag{21}$$

Here the value of flag is 0 for a cell in the body and 1 in the flow field. Meanwhile, the velocity components at these ghost cells for the solid wall boundary are set to be zero.

At cube boundaries, the information exchange is required between adjacent cubes. The simplicity of the algorithm is also kept by using ghost cells at the cube boundaries. At each side of cube, three ghost cells are added beyond the cube boundary. Therefore, if the sizes of two adjacent cubes are same, the six exact overlapping of cells at boundary ensure the accurate transfer of the information. The information transfer from smaller cube to larger one is performed by assigning the average values of smaller cube near the boundary. The transfer from the larger cube to smaller one is by just taking the value on in the larger cells. Because this procedure is slightly less accurate, therefore, the same size of cubes should be used near the body boundary.

Updates of the solution at the cube boundaries are performed combined with the sub-iteration for the time accuracy as described in the previous section.

4.4 Efficiency improvements

With the present approach using the cubes, various techniques for improving the computational efficiency are possible.

(a) Parallel computation: The parallel computation of the present method is straightforward because of the same mesh density at all cubes. The simplest parallelization is to distribute the cubes to the CPUs

in a sequential manner. However, the computational density of the cube is not large enough for 2D computations. Therefore, a grouping of cubes for parallelization will be required to obtain high parallelization efficiency for 2D case.

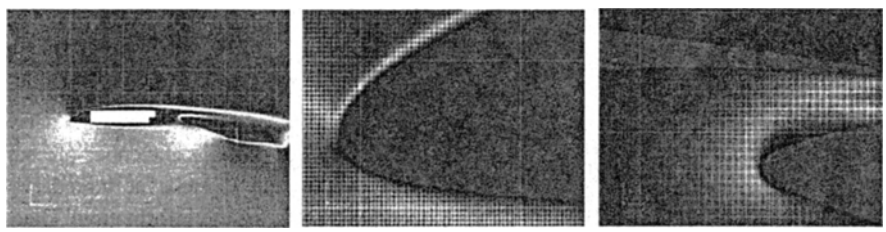
(b) Mesh sequence: In order to reduce the computational time especially for steady flows, a mesh sequence is employed utilizing the advantage of the Cartesian mesh. This is to use the pixel skipped mesh at each cube at the early stage of the computation. Figure 5 shows an example of multi-element airfoil computations with the mesh sequence. The final BCM mesh has 64x64 cells in each cube. The computation is started by the same cube mesh with 16x16 cells in each cube for the first 500 steps [Fig. 5 (a)]. Then, the next 500 steps are computed by 32x32 cells in each cube [Fig. 5 (b)]. Then the finest mesh is used for the final result [Fig. 5 (c)]. This mesh sequence is very effective not only for steady flows but also for setting up the flow field for unsteady flows.

The pixel skipping is also very useful for the post-processing of large-scale computations.

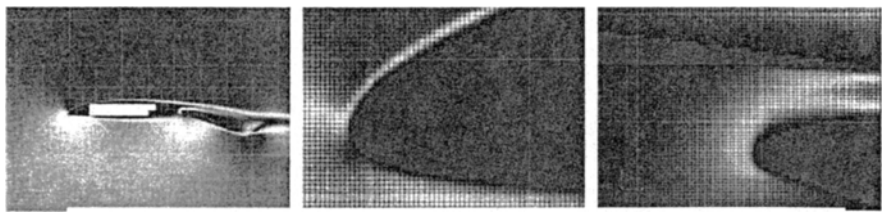
(c) Active-cube selection: A method of an active cube selection is also developed where the residual at each cube is monitored at the end of each time integration, and the small residual cubes, named non-active cubes, are skipped in the next time integration. This is especially effective for supersonic flows. Also, for supersonic flows, the active-domain marching proposed for the unstructured mesh (Nakahashi and Saitoh, 1997) is also utilized in a straightforward manner.

4.5 Parallelization

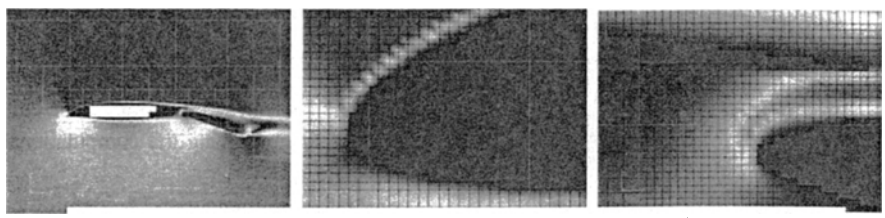
The present method is well suited to parallel computers. In this study, the OpenMP (www.openmp.org) is used to parallelize the flow solver and run on the NEC SX-7 parallel computer at Supercomputing System Information Synergy Center in Tohoku University. This parallel computations were performed by 64x64 Cartesian mesh in 481 cubes for a four-element airfoil at AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6. The performance of the parallelization was very close to the ideal



(c) Computed Mach contours with 64×64 mesh in each cube and adaptive cube refinment.



(b) Computed Mach contours with 32×32 mesh in each cube.



(a) Computed Mach contours with 16×16 mesh in each cube.

Fig. 5. Computations by mesh sequence from 16x16 to 64x64 Cartesian meshs in cubes for a two-element airfoil at M_∞=0.2, α=0°, Re=5000.

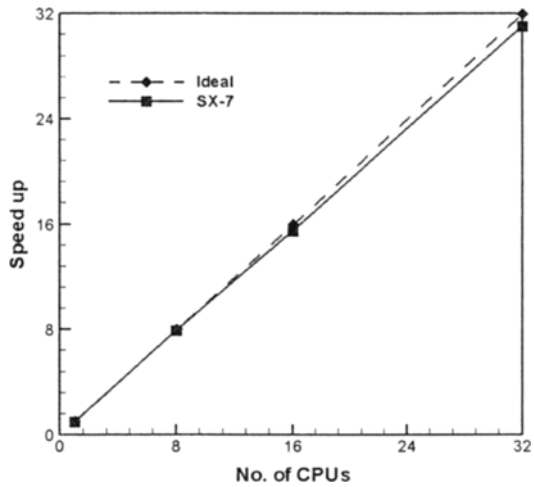


Fig. 6. Speedup in SX-7; Number of cubes is 481.

one when 32 CPUs were used for the computations as shown in Fig. 6.

5. Adaptive cube refinement

For computations using unstructured meshes, the adaptive mesh refinement (AMR) is very effective to improve the local resolution. However, the use of refinement/un-refinement for unstructured mesh may accompany some problems, such as the increase of memory overhead and the difficulty of the dynamic load balancing for parallel computations. The AMR is also used for the Cartesian mesh to locally increase the mesh resolution. However, it introduces the complexity into the Cartesian mesh approach.

In contrast, the implementation of the AMR is straightforward in the present approach if the adaptive refinement is applied not to the mesh but to cubes. Since the number of cubes is relatively small, the cube refinement is quite simple and the refinement does not cause the difficulty of the dynamic load balancing in parallel computations.

For the selection of cubes that are divided, Laplacian filter that is used in the image processing to capture the contours is employed. The Laplacian filter is given in the following equations.

$$g_{j,k} = f_{j,k-1} + f_{j-1,k} + f_{j+1,k} + f_{j,k+1} - 4f_{j,k} \tag{22}$$

The Laplacian filter is effective to detect the flow discontinuity.

The summations of the absolute values of Laplacian of all flow variables in a cube are computed for all cubes at first. Then the cubes having larger values are selected for the refinement by specifying the allowable increment of the total number of cubes. At each refinement, the cube size difference between adjacent cubes has to be always checked.

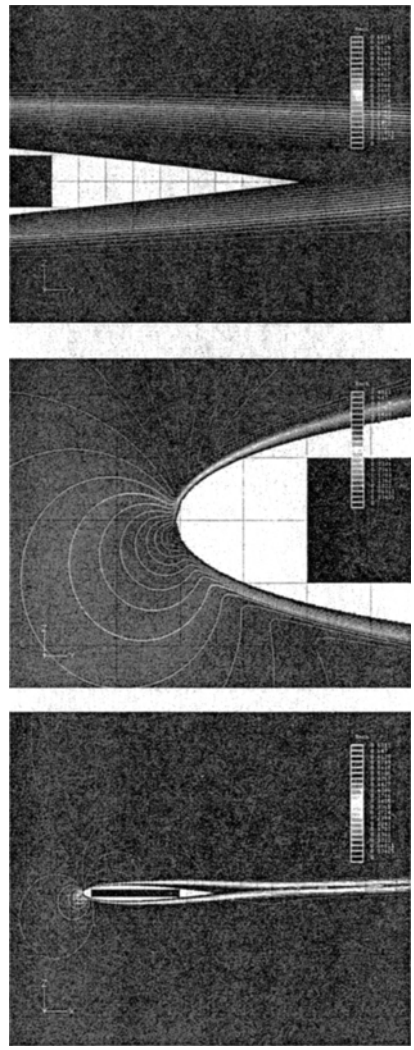


Fig. 7. Computed Mach contours around NACA0012 airfoil at $M_\infty = 0.5$, $\alpha = 3^\circ$, $Re = 5000$. Total number of cubes is 561 with 64x64 cells in each cube. The minimum spacing is 4.58×10^{-4}.

6. Computational results

6.1 NACA0012 airfoil

Figure 7 is a test case for the NACA0012 airfoil computations at the freestream Mach number of 0.5, angle of attack of 3 degrees and the Reynolds number

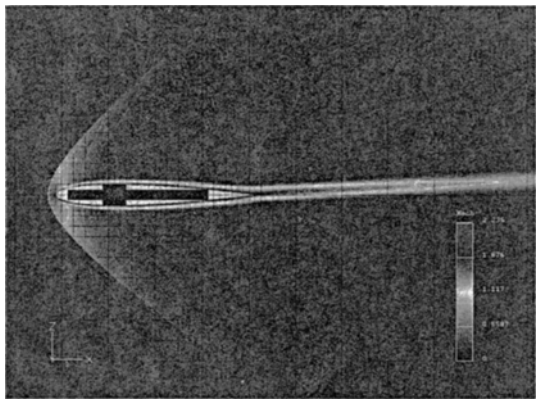


Fig. 8. Computed Mach contours around NACA0012 airfoil at $M_\infty = 2$, $\alpha = 3°$, $Re = 5000$.

of 5,000. The total number of cubes is 561 after the adaptive cube refinement with 64x64 cells in each cube. The minimum spacing near the airfoil surface is 4.58×10^{-4}. The computation was performed with the fourth-order spatial accuracy and the first-order time accuracy, though the computed flow field was steady state.

As shown in the enlarged figures around the leading and trailing edges, the minimum spacing of 4.58×10^{-4} is much smaller than the boundary layer thickness and the staircase representation of the wall boundary does not cause any disturbances on the Mach contours near the surface.

Figure 8 shows the computed Mach contours at Mach 2, angle of attack of 3 degrees and the Reynolds number of 5,000. Owing to the fine mesh, the shock waves are crisply captured. The wake is also sharply shown and continues to the downstream boundary.

6.2 Two-element airfoil

Figure 9 is a test case for a two-element airfoil. The total number of cubes is 336 after the refinement, and the minimum spacing near the wall boundaries is 7.63×10^{-4}. The freestream Mach number is 0.2, the angle of attack is 0 and the Reynolds number is 5,000.

The computations were performed with the fourth order spatial accuracy and the first order in time. Figure 9 is the instantaneous density and Mach contours showing the periodical shedding of vortices from the flow separation on the upper surface of the flap. With the higher order accuracy and the very fine mesh, the contours of vortices are sharply depicted.

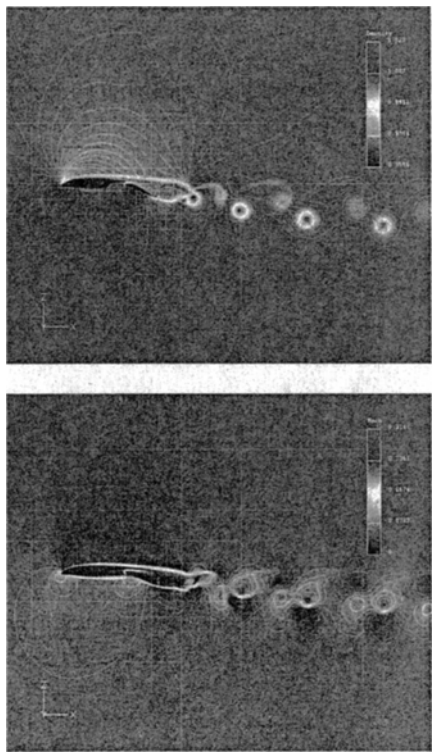


Fig. 9. Computed density (upper) and Mach (lower) con-tours around a two-element airfoil at $M_\infty = 0.2$, $\alpha = 0°$, $Re = 5000$.

6.3 RAE2822 airfoil

RAE2822 airfoil is commonly used as a test case of the newly developed flow solver as well as the turbulence models. In this experiment (Cook et al., 1979), the boundary layer transition trip of diameter 0.762 mm was attached at 3% chord length on the upper surface of the airfoil. The airfoil chord length was 0.61 m. There is no description in the report about the type of the transition trip whether it is a wire or spheres. Here we assume that the transition trip is a wire and compute as the two-dimensional flow. Although the turbulence at high-Reynolds number always has a three-dimensionality, it will be worth to apply the present method for testing the capability.

Total number of cubes is 486 with 256x256 cells in each cube. Therefore the total number of computational cells is about 32 millions. Even with this large number of cells, the size of the mesh file is small owing to the runlength compression. The required memory of the flow solver with this mesh size is about 1.4 GBytes which can be handled by personal computers.

The minimum spacing near the airfoil surface is

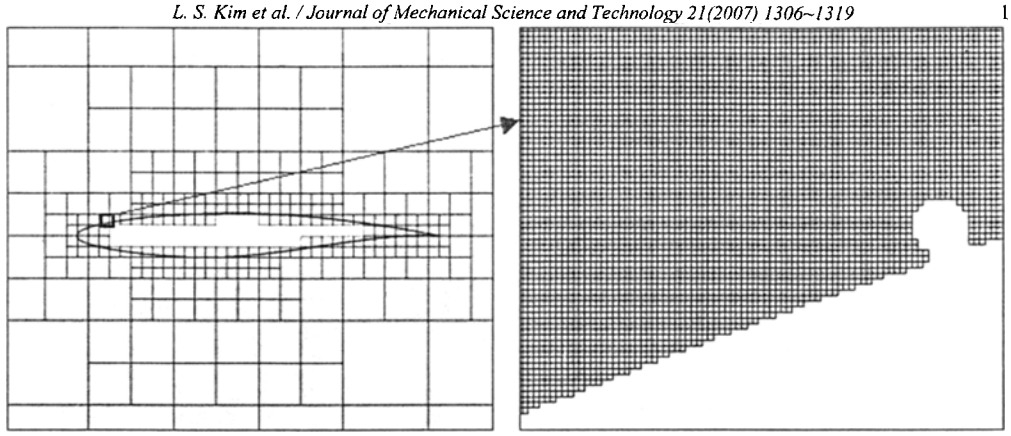


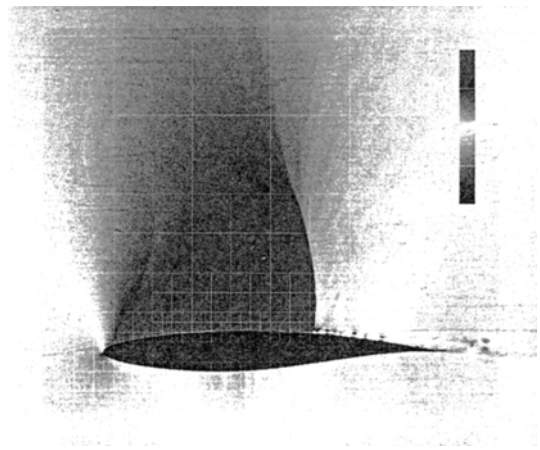
Fig. 10. Cube boundaries and an enlarged view of the Cartesian mesh in a cube that includes the transition trip.

$1.14x10^{-4}$. This value is very close to the thickness of the viscous sublayer for flow of Reynolds number of million. The Reynolds number of the present case is $6x10^{6}$, so that some more small spacing may be required.

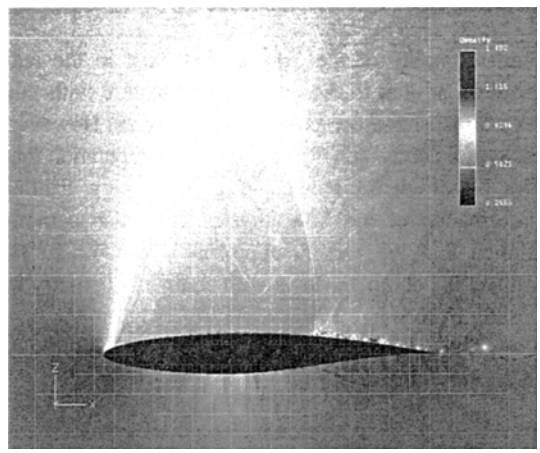
The diameter of the transition trip is $1.25x10^{-3}$ in the non-dimensional value with the chord length. Therefore, the trip wire can be described by the present mesh about 10 cells in x and y coordinates directions as shown in Fig. 10. Even with the transition trip, the BCM mesh generation was quite simple. The geometry was defined by a one-dimensional array of points including the trip geometry, and the remaining procedure of the mesh generation is fully automatic.

The flow was computed by the third order spatial accuracy and the first order time accuracy. Three levels of the mesh sequences, namely 64x64, 128x128, and 256x256, were used to settle the initial flow field. In the post- processing, the pixel skipped mesh was also utilized for drawing the flow field. Figure 11 was painted by 64x64 mesh using the 256x256 results because of the memory limitation of the post-processing software. Figure 12 is the original resolutions of 256x256 in a cube because of the smaller size of drawing region.

The computed result with this fine mesh shows the detailed flow features as shown in Figs. 11 and 12. Especially, in Fig. 12, the thin boundary layer in the leading edge region is thickened in the upstream of the trip wire. In the downstream of the trip wire, very small vortices are shed periodically. The flow field at the root of the shock wave becomes more complex due to the interaction between the on-coming vortices and the shock wave. In this region, however, two-

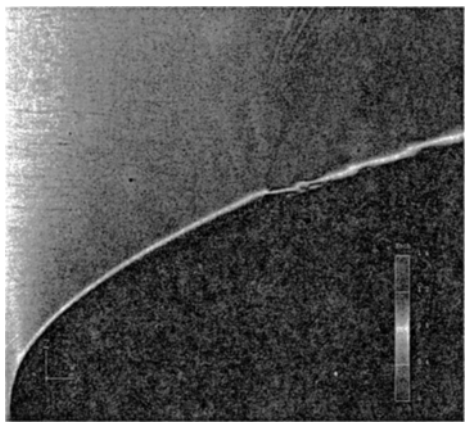


(a) Mach contours

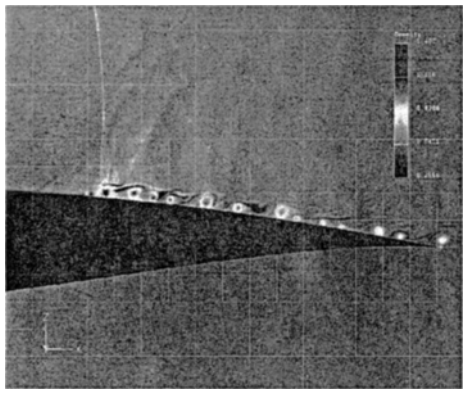


(b) Density contours

Fig. 11. Computed contours around RAE2822 airfoil at M_∞ = 0.73, α =2.79°, Re = $6.56x10^{6}$.



(a) Computed Mach contours near transition trip



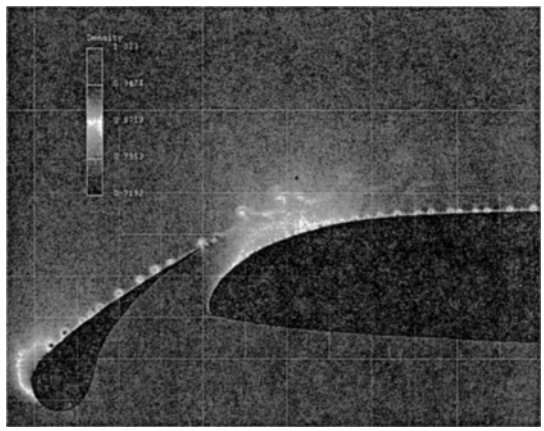
(b) Computed density contours near trailing edge

Fig. 12. Computed contours in the leading edge and trailing region of RAE2822 airfoil at M_∞ = 0.73, α=2.79°, Re= 6.56x10^6.

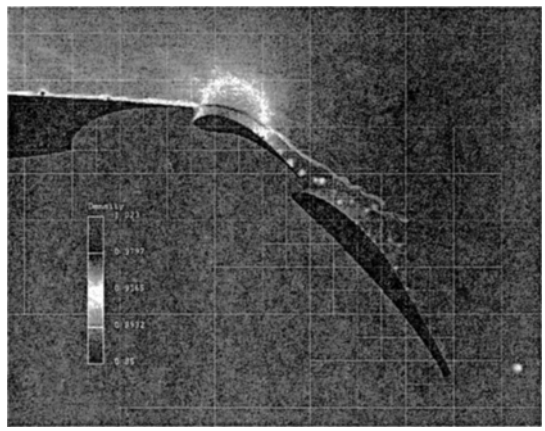
dimensionality of the flow cannot exist in the real physics, so that to discuss the flow feature with this two-dimensional computation is dangerous. However, at least it can be concluded with this computation that the highly dense mesh can give us very fruitful information of the flow physics. It is also demonstrated that the present approach is capable to simulate the complex flows with complex geometry in a simple procedure.

6.4 Four-element airfoil

Figures 13 and 14 are the computed instantaneous density and Mach contours of a four-element airfoil, respectively. The angle of attack is 8.16 degrees, the freestream Mach number is 0.201, and the Reynolds number is 2.83x10^6. The total number of the cubes is 481, and the mesh in each cube is 256x256. The



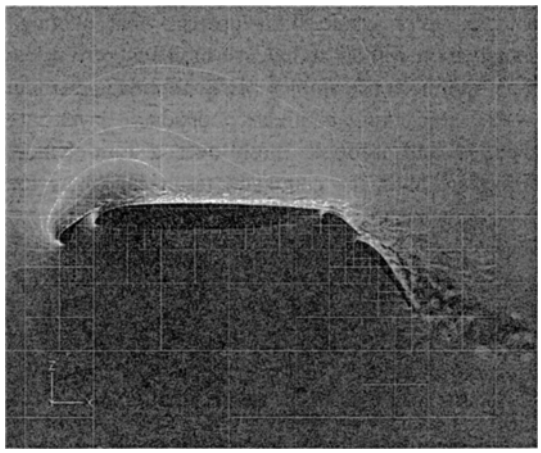
(a) Enlarged view near slat



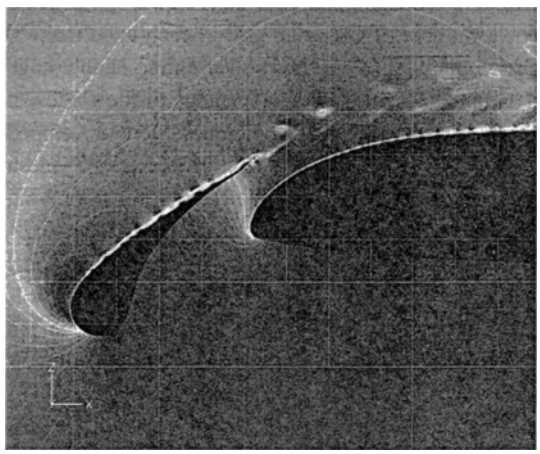
(b) Enlarged view near flaps

Fig. 13. Computed density contours over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6. Number of cubes=481, Mesh in each cube=256x256, Minimum spacing=1.4x10^{-4}.

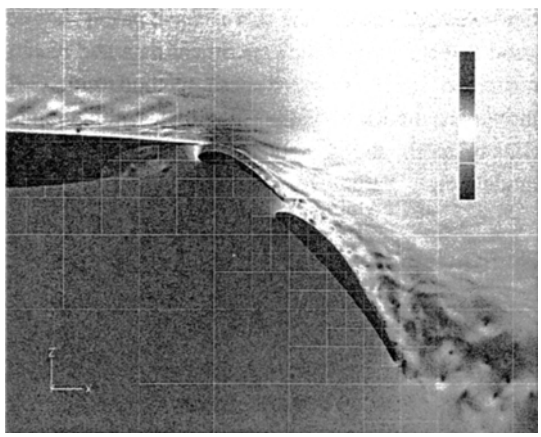
minimum spacing for this mesh is 1.4x10^{-4}. The wakes of the slat, main airfoil, bane and flap can be seen clearly. It is also observed that many small vortices flow downward in the wall boundary layers. Figure 15 shows the instantaneous Cp distribution over a four-element airfoil. Because of the small vortices along the airfoil surface, many sharp suction peaks appeared in this figure. However, the overall Cp distribution shows quite good agreement with the experiment. This result is very interesting because the computation was conducted without any turbulence models. A comparison of the pressure coefficients of the time-averaged result with the experiment (Omar et al., 1979) is shown in Fig. 16. The computation did not use any turbulence models but the result showed



(a) Entire view



(b) Enlarged view near slat



(c) Enlarged view near flaps

Fig. 14. Computed Mach contours over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6. Number of cubes=481, Mesh in each cube=256x256, Minimum spacing=1.4x10^{-4}.

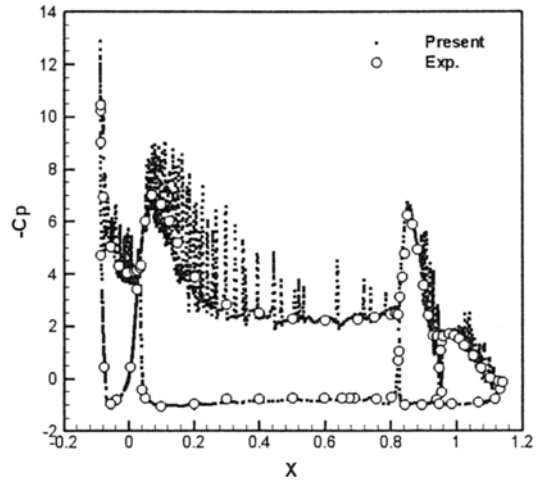


Fig. 15. Instantaneous pressure distribution over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6.

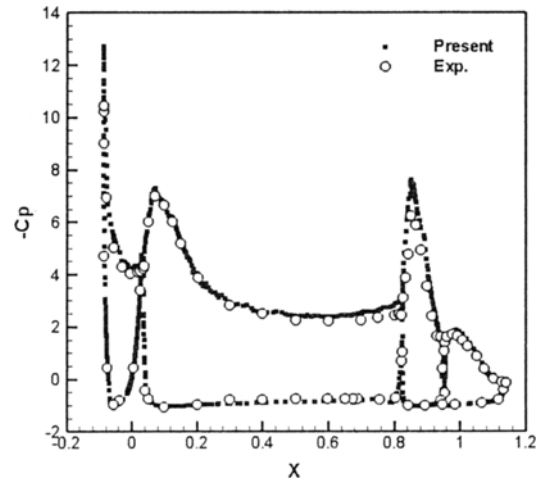


Fig. 16. Time-averaged pressure distribution over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6.

good agreement with the experiment. This indicates that, with the enough resolution, the computation without physical models can accurately simulate the flows.

Figures 17 and 18 show the comparisons of the Cp distributions and pressure contours over a four-element airfoil among the experiment, the unstructured-mesh computation (Kim and Nakahashi, 2003) with Goldberg-Ramakrishnan (G-R) turbulence model (Goldberg and Ramarkrishnan, 1993) and the present BCM com-putation without turbulence model, respectively. It is observed that good agreement was

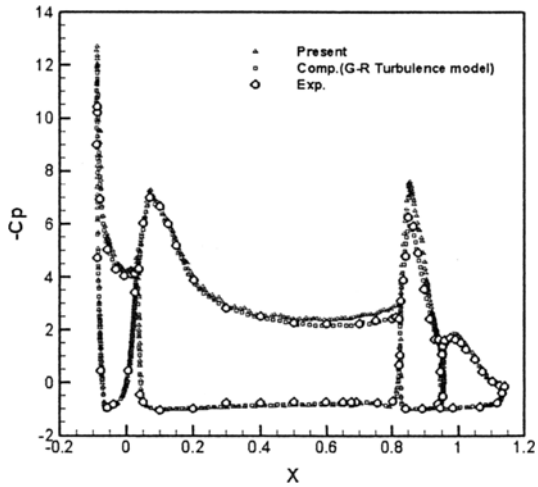
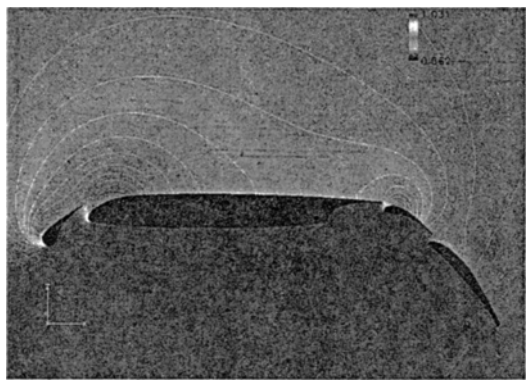
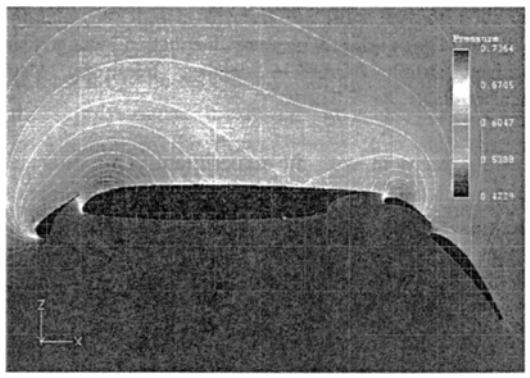


Fig. 17. Comparison of Time-averaged pressure distribution over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6.



(a) Pressure contours computed by unstructured-mesh computation with G-R turbulence model.



(b) Pressure contours computed by present BCM computation without turbulence model.

Fig. 18. Comparison of computed pressure contours over a four-element airfoil; AOA=8.16 degrees, freestream Mach number=0.201, Re=2.83x10^6.

obtained as a whole. This presents that although computation without turbulence model is performed, if as fine mesh as complex physical phenomena like wake interaction, confluent boundary layer are captured and isotropic mesh over a four-element airfoil are used, the computed result can show the good agreement with experiment data.

7. Conclusions

In this study, an approach named Building-Cube Method, aimed for large-scale computation on near-future advanced parallel computers, was developed. The method is based on the Cartesian mesh, and the local grid density is adapted to the flow characteristic length by changing the cube size. Equal spacing and equal number of Cartesian grid in each cube make it easy to parallelize the flow solver and to handle huge data output. By use of the advantage of the Cartesian mesh in cubes, several techniques were developed such as, the mesh data compression, the mesh sequence and the active-cube selection for improving the computational efficiency and the post processing.

The method was applied to the two-dimensional viscous flows; NACA 0012 and two-element airfoils at relatively low Reynolds number, RAE2822 airfoil with transition trip at high Reynolds number and four-element airfoil at high Reynolds number. The computed results showed detailed flow features near the airfoil surfaces owing to the high-density mesh. Especially, the pressure coefficient distributions of the time-averaged result of the four-element airfoil showed good agreement with the experiment data and results computed by unstructured-mesh method with G-R turbulence model. It is interesting that, with the high-density mesh, the computations without turbulence models show reasonable results.

Nomenclature

i	: One-dimensional array
j,k	: Mesh index
$X(x,y)$	: Cartesian coordinates
u,v	: Velocity components
Q	: Conservative variable
e	: Total energy
t	: Time
$\mathbf{F}$	: Inviscid flux vector
$\mathbf{G}$	: Viscous flux vector
Δs	: The size of the cube

n_{cell} : The number of cells
A : Jacobian of $\mathbf{F}$
h : Numerical flux
n : Normal vector

Greek symbol

ρ : Density

Superscript

v : Sub-iteration index

References

Cook, P. H., McDonald, M. A. and Firmin, M. C. P., 1979, "Airfoil RAE 2822 – Pressure Distributions, and Boundary Layer and Wake Measurements," Experimental Data Base for Computer Program Assessment, AGARD-AR-138, A6.

Goldberg, U. C. and Ramakrishnan, S. V., 1993, "A Pointwise Version of Baldwin-Barth Turbulence Model," *Comp. Fluid Dyn.*, Vol. 1, pp. 321~338.

http://www.openmp.org

Kim, L. S. and Nakahashi, K., 2003, "Navier-Stokes Computations of Multi-element Airfoils," *Computational Fluid Dynamics Journal*, Vol.12, No.13, pp. 107~114.

Matsuno, K., 1993, "Improvement and Assessment of an Arbitrary-Higher-Order Time-Accurate Algorithm," *Computer and Fluids*, Vol. 22, No. 2/3, pp. 311~322.

Mellen, C. P., Frohlich, J. and Rodi, W., 2003, "Lessons from LESFOIL Project on Large-Eddy Simulation of Flow around an Airfoil," *AIAA J.*, Vol. 41, No. 4, pp. 573~581.

Nakahashi, K. and Egami, K., 1991, "An Automatic Euler Solver Using the Unstructured Upwind Method," *Computers & Fluids*, Vol. 19, No. 3/4, pp. 273~286.

Nakahashi, K. and Saitoh, E., 1997, "Space-Marching Method on Unstructured Grid for Supersonic Flows with Embedded Subsonic Regions," *AIAA Journal*, Vol. 35, No. 8, pp. 1280~1285.

Nakahashi, K., 2003, "Building-Cube Method for Flow Problems with Broadband Characteristic Length", *Computational Fluid Dynamics* 2002, Eds. S. Armfield, R. Morgan, K. Srinivas, Springer, pp. 77~81.

Nakahashi, K., Ito, Y. and Togashi, F., 2003, "Some Challenges of Realistic Flow Simulations by Unstructured Grid CFD," *Int. J. for Numerical Methods in Fluids*, Vol. 43, pp. 769~783.

Obayashi, S. and Guruswamy, G. P., 1995, "Convergence Acceleration of a Navier-Stokes Solver for Efficient Static Aeroelastic Computations," *AIAA Journal*, Vol. 33, No. 6, pp. 1134~1141.

Omar et al., 1979, "Two-dimensional Wind-Tunnel Tests of a NASA Supercritical Airfoil with Various High-Lift Systems Vol. 2-Test Data," NASA CR-2215.

Yamamoto, S. and Daiguji, H., 1993, "Higher-Order-accurate Upwind Schemes for Solving the Compressible Euler and Navier-Stokes Equations," *Computer and Fluids*, Vol. 22, No. 2/3, pp. 259~270.