39

A Solution-Adaptive Technique using Unstructured Hexahedral Grids

M. Sun and K. Takayama [1]

Abstract This paper summarizes our work on the development of a solution-adaptive flow solver using unstructured hexahedral meshes for thress dimensional computation. A cell-face data structure is proposed to avoid data dependency that is encountered when applying unstructured data for high performance computation. Two flow solvers are constructed using two different schemes. One solver uses the MUSCL-Hancock scheme, a second-order upwind scheme, another employs a second-order central-difference scheme with artificial viscosity. The parallel performance of the solvers on three supercomputers is reported. Numerical result of shock motion from a square tube to a circular tube is also presented.

Introduction

Unstructured meshes have been widely used in computation fluid dynamics [11]. In order to improve the accuracy of numerical solutions with less computer time, various grid adaptation methods (*e.g.* [5, 12]) have been developed and applied to the simulation of flows that contain disparate length scales, such as flows with shock waves. Most schemes were proposed for triangles and tetrahedra. Since a triangular mesh contains more edges than the quadrilateral counterpart, it consumes more storage and flux evaluations. Aftosmis et al [1] compared a variety of schemes on both meshes, and concluded that on regular and stretched meshes the additional edges do not lead to apparent accuracy advantage. For three dimensional meshes, Biswas and Strawn [2] also concluded that hexahedral meshes utilize computer resources more efficiently than tetrahedral meshes for the same level of accuracy by comparing the solutions of the Euler equations. In addition, unlike the hybrid methods that combine structured quadrilaterals/prisms around body with unstructured triangles/tetrahedrals outside in the computation of viscous flows, a technique using unstructured quadrilateral or hexahedral meshes will neatly generate a layer of body-fitted mesh by repeatedly refining the cells in the boundary layer. It is a real bonus that hexahedral grid generation becomes rather easy especially due to rapid progress on the automation of unstructured hexahedral meshing. [1]

The 2-D vectorized adaptive solver (VAS2D) using a grid containing uniform quadrilateral cells has been presented in [9], and applied to the solution of a variety of problems. The present work aims to develop an algorithm for three-dimensional flows using hexahedral cells. The paper is organized as follows. Section 2 is devoted to the basic ideas of the present data structure and grid adaptation. Section 3 reviews the main points of the present flow solver. Section 4 reports numerical results.

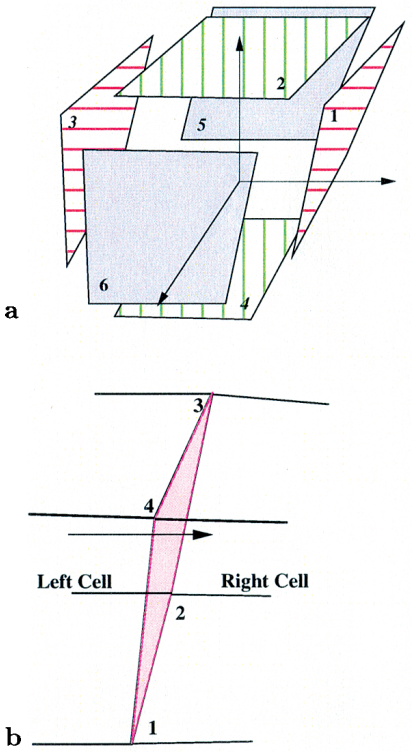

Figure 1: Cell-face data structure: **a.** every cell stores its location and points to six faces; **b.** every face stores the locations of its four vertices and points to two neighboring cells.

Data structure and grid adaptation

The data structure used for unstructured quadrilateral and hexahedral meshes is optimized for the finite volume method, in order to enhance computational efficiency and to minimize memory requirement. The finite volume method evaluates the change of the conservative values of all control volumes by integrating

[1]Shock Wave Research Center, Institute of Fluid Science, Tohoku University, Katahira 2-1-1, Aoba, Sendai 980, Japan

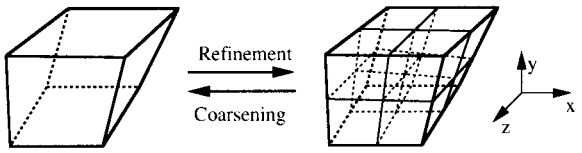

Figure 2: Strategy for grid adaptation: a father cell is divided into eight sons

their interface fluxes which consists of two steps, calculating fluxes at every interface and gathering interface fluxes for every control volume. The data structure is designed as having two primary arrays, one for control volumes, another for faces, with a bi-directional reference between them,

$$\text{control volume} \Longleftrightarrow \text{face}.$$

Thus the data structure is called a *cell-face* one. The data structure actually degenerates to a *cell-edge* one for 2-D quadrilateral grids. There are no reconcurrence appearing in the bi-directional reference, so that it naturally avoids sorting or coloring for vector/parallel supercomputing. Furthermore, because of the similarity of the data structure for two dimensions and three dimensions, it is rather convenient for programming.

A basic nature of the data structure for three dimensions is that every cell points to its six faces, and every face points to its two neighboring cells, as shown in Fig. 1. Physical variables are stored at cells, therefore it is a cell-centered data structure. Every face is set to have a direction, so that we may label two neibhouring cells, *left* and *right*, according to the direction. The six faces of each cell is ordered following these rules: two opposite faces must have the same direction, and three pairs of opposite faces must constitute a local coordinate system (the right-hand rule). These strict definitions in the data structure reduce the complexity of logic in the unstructured adaptive solver without losing generality. The existence of the ordering method for any unstructured quadrilateral and hexahedral grids has been proven in [6, 9]. Note that the data structure does not require additional memory to store the definitions, but stores the neighboring information following the rules.

A cell to be refined is divided into eight subcells or *sons* as shown in Fig. 2. To avoid unlimitedly adding fine cells around shock waves, either the maximum level of refinement or the minimum cell volume or both should be prescribed. When either the level or the volume of cell reaches the given limit, further refinement is prohibited. This strategy is similar to many other h-adaptive schemes. The criterion used for grid adaptation is based on the truncation error of the Taylor series expansion of density. The truncation error indicator is given by the ratio of the second-order derivative term to the first-order one in the Taylor series. The indicator is consistent with the limiting procedure in flow solvers so that the cells that turn on the artificial viscosity or the slope limiter must be refined.

Refinement and coarsening procedures are handled separately. Both procedures have similar steps for vector/parallel processing:

1. handling the inside of cells which are flagged to be refined or coarsened,
2. handling the faces of the flagged cells,
3. arranging memory.

Step 1 is conducted based on cells, and updates all information inside, such as face deletion and adding of finer cells, which is independent of other cells. Step 2, based on faces, renews every face of refined or coarsened cells and its two neighboring cells, which may be done without influencing other faces. In this way the adaptation procedure has no data dependence and can be naturally vectorized. Step 1 and 2 change the status of some cells and faces, for example from sons to fathers. Step 3 fills the "holls" that appear due to the deletion of cells, and make the memory easily to be accessed by other subroutines such as flow solvers.

Either the conservative or the primitive values at newly appeared cells in the adaptation procedure have to be imposed by those on the old mesh. In the refinement procedure the conservative variables of new sons are linearly interpolated from those of their father, while in coarsening procedure flow variables of a coarsened father cell are the volume-weighted average of these of its "dead" sons. The interpolation and the weighted average for new cells preserve conservation [6].

Flow solver

The conservation laws are solved by the finite volume method. The method solves the conservation laws by directly applying them to every non-overlapping discrete volume the summation of which covers the whole computational domain. The conservation laws written for a discrete control volume are, for a second-order scheme in both time and space,

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i - \frac{\Delta t}{\Delta \Omega_i} \sum_{k=1}^{\text{faces}} \hat{\mathbf{F}}_k^{n+1/2}, \qquad (1)$$

where $\hat{\mathbf{F}}_k^{n+1/2}$ are fluxes evolved by half a time step, and locate at the center point of the interface k. Since the values are unknown at the center point, they are interpolated from the centroid (cell-centered data are considered here). Difference between all conservative schemes is just how they calculate the numerical fluxes. Two schemes that belongs to the central-difference scheme and the upwind scheme respectively are used to calculate the fluxes.

For the solver using the central-difference scheme, a TVD smoothing scheme is used [6]. A predictor step evolves the solution by a modified half time step, $\frac{\Delta t}{2}(1 - 2\mu)$, by locally solving the two-dimensional Lax-Friedrichs scheme at the interface, where μ is a

smoothing coefficient (see [8] for details). Both the interpolation and the predictor step necessitate estimation of the gradients at every cell which are given by the least-square method. In solving the compressible Euler equations, the fluxes through cell faces, which consist of inviscid fluxes and smoothing fluxes, are

$$\hat{\mathbf{F}} = \hat{\mathbf{F}}_{\text{inviscid}} + \hat{\mathbf{F}}_{\text{smoothing}}. \tag{2}$$

The inviscid fluxes are convection terms and pressure surface terms, or those in the Euler equations. The smoothing fluxes which are added to suppress spurious oscillations, read

$$\hat{\mathbf{F}}_{\text{smoothing}} = \begin{pmatrix} -\rho_x\mu_1 - \rho_y\mu_2 - \rho_z\mu_3 \\ -(\rho u)_x\mu_1 - (\rho u)_y\mu_2 - (\rho u)_z\mu_3 \\ -(\rho v)_x\mu_1 - (\rho v)_y\mu_2 - (\rho v)_z\mu_3 \\ -(\rho E)_x\mu_1 - (\rho E)_y\mu_2 - (\rho E)_z\mu_3 \end{pmatrix}, \tag{3}$$

where μ_1, μ_2 and μ_3 are nonlinear coefficients of artificial viscosity. In multi-dimensional flow, the artificial viscosity coefficient is actually a tensor. Since it should be as less diffusive as possible, it is set to be zero in directions other than the normal direction of the wave front. It is then rotated to the Cartesian coordinates, and become $\begin{vmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{vmatrix}$. Having obtained the artificial viscosity coefficient in x, y and z directions, one may easily design a smoothing flux as that in (3). The direction of the wave front is estimated by velocity difference between two cells.

For the solver using the upwind scheme, the MUSCL-Hancock scheme (see, [10] for formulations on structured grids) is extended to unstructured grids. Primitive variables are first reconstructed by the least square method at cells, and other procedures are all conducted at interfaces. In slope limiting procedure which aims to maintain TVD conditions around sharp gradients, only gradients in the direction along two neighboring cells beside an interface are modified by limiters. The minmod limiter is used in most applications. The limited slopes are then used to evolve solutions at the two cells respectively by half a time step. A HLL approximate Riemann solver (see, [10]) is chosen to determine the flux at interface because of its simplicity and generality.

Numerical results

The data structure proposed contains no data dependency, so that it allows easy vector/parallel processing without recourse to any sorting or coloring procedure that most unstructured data structures require. The parallel performance on three different machines of our 2-D and 3-D solvers is summarized in Fig 3. On the shared-memory machines, SX-5 and Cray C90, the data are collected by testing an unsteady computation in which grid adaptation is performed at every

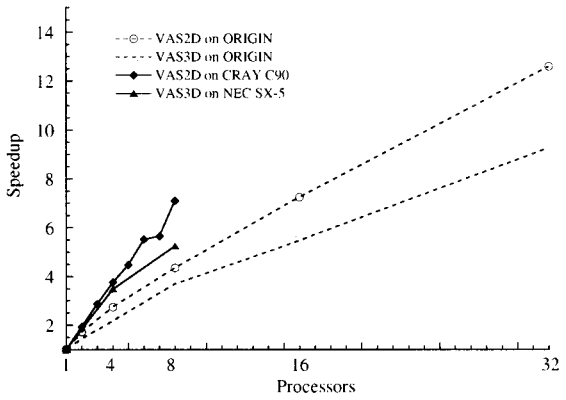

Figure 3: Performance of the 2-D and the 3-D solvers on machines, Cray C90, Origin 2000, and SX-5

iteration. For eight processors the speedup is approximately six, which is not bad for the two solvers are automatically parallelized by the compliers coming with the machines. On the distributed-memory machines, only steady flow computation is tested, and very few adaptations are conducted. It seems that the speedup achieved is not as high as well-tuned parallel codes using domain decomposition method. However, if the overheads of conducting domain decomposition is taken into account, the speedup is rather satisfactory because the data structure allows grid adaptation that will decrease computer time dramatically (but not the rate of speedup).

Numerical results of a 3-D case are shown in Fig. 4, in which an incident shock wave with M_s=2.0 moves from a square cross-section shock tube into a larger cylindrical tube. Initial mesh contains 299 cells. Notice that unstructured cells are used in the cylindrical cross-section. In this unsteady computation, because of the existence of the moving shock wave and complicated vortex formation whose locations are unknown in advance, the mesh adaptation is done at every time step. An adaptive mesh and the density distribution on the surface at some instant after shock wave entering the circular tube are shown in Fig. 4bc; the total number of cells is increased to about 0.78 million. It takes about three minutes to run this computation on SX-5 using one processor. It is seen that basic flow features, such as incident shock and vortices, are well reproduced. Hexahedral cells on a cut plane inside the tube are shown in Fig. 4d; fine cells are clearly distributed around shocks and vortex regions.

Concluding remarks

A solution-adaptive technique for unstructured hexahedral meshes has been proposed using a vectorizable data structure, and it is successfully applied to the solution of the unsteady Euler equations. The efficiency of vector processing is comparable with that of a structured solver.

In solving the Navier-Stokes equations, unlike the

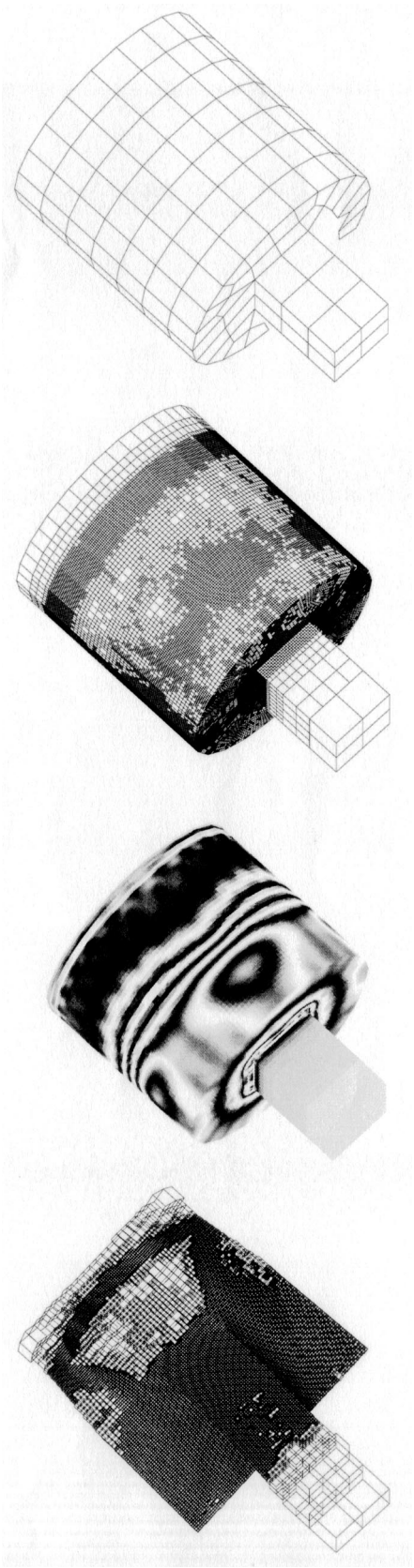

Figure 4: Shock wave entering a cylindrical tube from a smaller square tube: **a.** initial unstructured mesh, 299 cells; **b.** adaptive surface meshes at $t = 1.4$, totally 776,550 cells and 2,412,340 faces; **c.** corresponding density contours; **d.** adaptive cells on a cut plane.

hybrid methods which combine structured quadrilaterals around body with unstructured triangles outside, the present unstructured adaptation method adopts uniformly unstructured quadrilateral and hexahedral meshes. It neatly generates a layer of body-fitted orthogonal mesh by repeatedly refining the cells in the boundary layer, and allows easy directional refinement. Extension of the technique for solving Navier-Stokes equations is now being under construction.

References

[1] Aftosmis MJ, Gaitonde D, Sean Tavares T (1994) On the accuracy, stability, and monotonicity of various reconstruction algorithms of unstructured meshes, AIAA paper 94-0415.

[2] Biswas R, Strawn RC (1997) Tetrahedral and hexahedral mesh adaptation for CFD problems, NAS-97-007, NASA Ames Research Center.

[3] Domel ND (1996) Research in parallel algorithms and software for computational aerosciences, NAS-96-004, NASA Ames Research Center.

[4] Kallinderis Y, Vijayan P (1993) Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes. AIAA J. 31 : 1440-1447.

[5] Löhner R (1987), *Comput. Meths. Appl. Meh. Engrg.* V61, pp.323-338.

[6] Sun M (1998) Numerical and experimental studies of shock wave interaction with bodies, Ph.D. Thesis, Tohoku University, Japan. *http://ceres.ifs.tohoku.ac.jp/~sun/thesis.html*

[7] Sun M, Takayama K (1999) Conservative smoothing on an adaptive quadrilateral grid, *J. Comp. Phys*, V150 : 143-180.

[8] Sun M, Takayama K (1999) A simple smoothing TVD scheme on structured and unstructured grids, Godunov methods theory and applications, Oxford University, Oct. 18-22.

[9] Sun M, Takayama K (1999) H-adaption on unstructured quadrilaterals, 8th Intl. Symp. on Comput. Fluid Dynamics, Bremen, Germany.

[10] Toro EF (1999) Riemann solvers and numerical methods for fluid dynamics, 2nd edition, Springer.

[11] Venkatakrishnan V (1995) A perspective on unstructured grid flow solvers, AIAA paper 95-0667. *see also AIAA J.* V34 : 533-547, 1996.

[12] Voinovich P, Timofeev E, Takayama K, Saito T, Galyukov A (1998), AIAA paper 98-0540.