AIAA 2001–2656

A Solution-Adaptive Technique using Unstructured Hexahedral Grids

M. Sun and K. Takayama
Shock Wave Research Center, Institute of Fluid Science, Tohoku University, Sendai, 980-8577, JAPAN

Anaheim Summer Co-Located Conferences
11-14 June 2001
Anaheim, CA

A SOLUTION-ADAPTIVE TECHNIQUE USING UNSTRUCTURED HEXAHEDRAL GRIDS

M. Sun * and K. Takayama †

Shock Wave Research Center, Institute of Fluid Science,
Tohoku University, Sendai, 980-8577, JAPAN

ABSTRACT

This paper reports the development of a solution-adaptive technique using unstructured hexahedral meshes for unsteady compressible flow simulation. A simple data structure that uses only two topological entities, cell and face, is proposed to avoid data dependency that is encountered when implementing unstructured data for vector and parallel processing. Two flow solvers are constructed based on two different schemes. One scheme belongs to the MUSCL family, while another employs a second-order central-difference scheme with artificial viscosity. The parallel performance on three supercomputers is reported. A numerical example of unsteady shock motion from a square tube to a circular tube is demonstrated.

INTRODUCTION

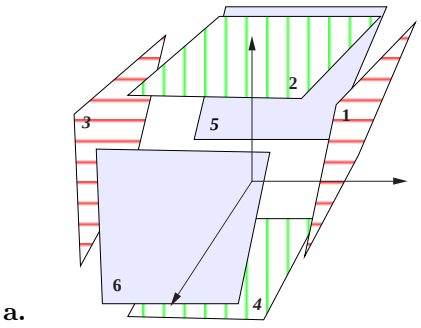
Unstructured meshes have been widely used in computation fluid dynamics.[10] In order to improve the accuracy of numerical solutions with less computer time, various grid adaptation methods have been developed and applied to the simulation of flows that contain disparate length scales, such as flows with shock waves. Most schemes were proposed for triangles and tetrahedra.[4, 11] 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 triangle/tetrahedral 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 therefore worthwhile to construct an unstructured quadrilateral and hexahedral technique for efficient and accurate flow analysis. We have presented a 2-D vectorized adaptive solver (VAS2D) using all-quadrilateral grids,[5, 8] and successfully applied to the simulation of a variety of problems, such as interfacial instability, airfoil flow, nozzle starting process, shock-transition over a cylinder, shock/boundary-layer interaction. The present paper reports our progress on the development of an algorithm for three-dimensional flows using hexahedral grids. The paper is organized as follows. The basic ideas of the present data structure and grid adaptation are discussed at first, and then two types of numerical schemes employed in this study are described. Parallel performance and a numerical result are presented before concluding remarks are given.

DATA STRUCTURE AND GRID ADAPTATION

The data structure developed for adaptive unstructured quadrilateral and hexahedral meshes is carefully optimized for the finite volume method, in order to enhance computational efficiency and

*Research Associate, email: sun@ceres.ifs.tohoku.ac.jp: †Professor, email: takayama@ifs.tohoku.ac.jp.



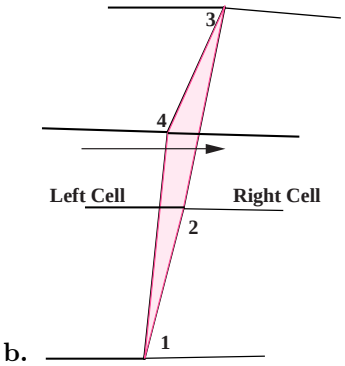


Figure 1: Cell-face data structure: **a.** every cell stores its location and points to six faces; **b.** every face stores the locations of its four vertices and points to two neighboring cells.

to minimize memory requirement. The finite volume method evaluates the change of the conservative values of all control volumes by integrating their interface fluxes. From the point view of programming it consists of two steps, calculating fluxes at every interface and gathering interface fluxes for every control volume. Therefore, the data structure is designed as having two primary arrays, one for control volumes, another for faces, with a bi-directional reference between them,

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

The two steps can be easily programmed and calculated without causing reconcurrence problem that inhibits vector processing. 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.

A basic nature of the data structure for three dimensions is that every cell points to its six faces,

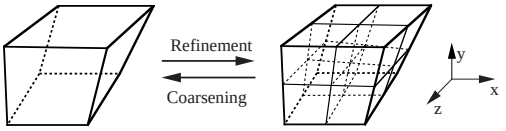


Figure 2: Strategy for grid adaptation: a father cell is divided into eight sons.

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 neighboring cells, *left* and *right*, according to the direction as shown in Fig. 1b. The six faces of each cell is ordered following rules: 1) two opposite faces must have a same direction; 2) the directions of three pairs of opposite faces must constitute a local coordinate frame (the right-hand rule) in which six faces is unambiguously ordered. These strict definitions in the data structure reduce the complexity of the unstructured adaptive solver without losing generality. Actually the whole solver, both grid adaptation and flow analysis, is freed of any searching which is commonly used for an unstructured grid allowing mesh adaptation. The existence of the ordering method for any unstructured quadrilateral and hexahedral grids has been proven.[5] 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 are prescribed. When either the level or the volume of cell reaches the given limit, further refinement is prohibited.

The criterion used for grid adaptation is based on the truncation error of the Taylor series expansion of density. The criterion is, for cell i

$$Refine = \text{Largest } \varepsilon_{\mathrm{T}} \text{ of } i'\text{s six faces} > \varepsilon_r,$$

$$Coarse = \text{Largest } \varepsilon_{\mathrm{T}} \text{ of } i'\text{s six faces} < \varepsilon_c,$$

where $Refine$ and $Coarse$ are logical flags which indicate whether the cell needs to be refined or

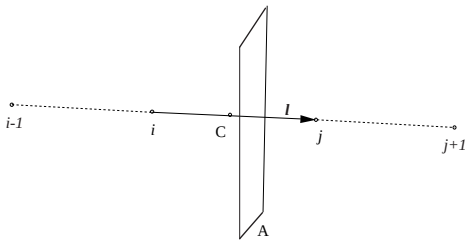


Figure 3: Sketch of an interface: face A and its two neighboring cells i and j; $i-1$ and $j+1$ are ghost cells.

coarsened respectively. The truncation error indicator ε_T is defined for every face, and given by the ratio of the second-order derivative term to the first-order one of the Taylor series of density, so that

$$\varepsilon_T = \mathrm{Max}\left(\frac{|(\nabla_l\rho)_C - (\nabla_l\rho)_i|}{\alpha_f\rho_c/dl + |(\nabla_l\rho)_i|}, \frac{|(\nabla_l\rho)_C - (\nabla_l\rho)_j|}{\alpha_f\rho_c/dl + |(\nabla_l\rho)_j|}\right) \tag{1}$$

where ∇_l denotes the gradient along direction l as shown in Fig. 3, $(\nabla_l\rho)_C = (\rho_j - \rho_i)/dl$. The subscripts C, i, and j represents the locations of the gradients, as shown in Fig. 3. It can be proven that the two ratios in (1) are exactly the central-difference formulation of the ratio of the second-order term to the first-order one

$$\left|\frac{1/2\rho''\Delta x^2}{\rho'\Delta x}\right|$$

at points i and j for α_f=0, by considering a 1-D problem along l. The logical operation Max in (1) just chooses a larger value of the two ratios.

There are three parameters in the criterion. α_f is initially designed to prevent a zero denominator, and it can also filter extremely small flow variations. For example, if the amplitude of the variations in some regions is much less than $\alpha_f\rho_c$, then the truncation error indicators are also small. Consequently cells will not be refined there. The subscript f therefore denotes "filter". ρ_c/dl which follows α_f is used to make the indicator dimensionless. ε_r and ε_c are threshold values for refinement and coarsening. Numerical experiments show that three parameters are nearly problem-independent. Parameters $\varepsilon_r = 0.08$, $\varepsilon_c = 0.05$ and $\alpha_f = 0.03$ are generally used in most 2-D and 3-D applications. The indicator in its structured formulation has been tested by a variety of gas-dynamic problems with shock waves in our early study.[6] It was noticed that the indicator is sufficient to detect most important phenomena in compressible flows, such as shock wave, contact surface, the front of expansion wave, and vortex. The indicator is also consistent with the limiting procedure in flow solvers so that any cell that turns on the artificial viscosity or whose slope is limited must be refined. In coarsening procedure only when all sons are flagged $coarse = true$, are they deleted. In Fig. 2 all sons and inside edges are removed, and recovered to their father cell.

In the refinement procedure, it is required that no two neighboring cells differ by more than one refinement level. This one-level-difference rule has been accepted by many authors because it simplifies adaptation procedure and prevents pathologically large volume ratios under certain circumstances. However, in unsteady calculations, the rule may mismatch moving refined regions, say shock wave regions. To overcome this problem, *pre-refinement* is introduced.[6] Once a cell cannot be refined due to the level difference between itself and one or a few neighboring cells, the neighboring cells are refined no matter what values of their $refine$ and $coarse$ are.

In programming, refinement and coarsening procedures are handled separately. Both procedures have similar steps:

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 makes the memory easily to be accessed by other subroutines such as flow solvers.

The physical 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 deleted sons. The interpolation and the weighted average for new cells preserve conservation.[5]

FLOW SOLVER

The conservation laws for non-reactive compressible flows 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,

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

where $\hat{\mathbf{F}}_k^{n+1/2}$ are fluxes evolved by half a time step, and locate at the centroid of interface k. Since the values are unknown at the centroid, they are interpolated from that at cells (cell-centered data are used here). Difference between conservative schemes is just the way to 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.[5] In solving the compressible Euler equations, the fluxes through cell faces, which consist of inviscid fluxes and smoothing fluxes, are

$$\hat{\mathbf{F}} = \hat{\mathbf{F}}_{\text{inviscid}} + \hat{\mathbf{F}}_{\text{smoothing}}. \qquad (3)$$

The inviscid fluxes are convection terms and pressure surface terms, or those in the Euler equations. 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. Both the interpolation and the predictor step necessitate estimation of the gradients at every cell which are given by the least-square method. This procedure for $\mu = 0$ is equivalent to a predictor-corrector Lax-Wendroff scheme.

The smoothing fluxes which are added to suppress spurious oscillations, are defined by the following equation

$$\hat{\mathbf{F}}_{\text{smoothing}} = -\mu_1 \mathbf{U}_x - \mu_2 \mathbf{U}_y - \mu_3 \mathbf{U}_z, \qquad (4)$$

where μ_1, μ_2 and μ_3 are nonlinear coefficients of artificial viscosity. Artificial viscosity coefficient is actually a tensor in multi-dimensional flows. Since it should be as less diffusive as possible, it is set to be zero in directions other than the normal direction of a wave front (k_1, k_2, k_3). The non-zero element, μ, is equal to the coefficient in the one-dimensional case. It is then rotated to the Cartesian coordinates, and become a diagonal matrix with three diagonal elements $(\mu_1, \mu_2, \mu_3) = (\mu|k_1|, \mu|k_2|, \mu|k_3|)$, where all elements are forced to be positive. Having obtained the artificial viscosity coefficient in x, y and z directions, one may easily design a smoothing flux as that in (4). The direction of the wave front is estimated by velocity difference between two cells. The nonlinear coefficient μ is limited according to one dimensional analysis so that the TVD constraint can be satisfied.[5, 7]

For the solver using the upwind scheme, the MUSCL-Hancock scheme (see Toro's book[9] 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 that is needed to suppress oscillations around sharp changes for high-order schemes, only gradients in the direction along two neighboring cells are modified by limiters. One may also limit the slopes on one cell, i.e. limit the slopes in all directions. We compared two limiting methods for quadrilateral meshes and found numerical difference is negligible. 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. One of the exact Riemann solver, HLL and HLLC approxi-

mate Riemann solvers[9] is chosen to determine the flux at interface depending what problem to solve.

The flow solver can be conveniently constructed under the cell-face data structure with high efficiency. Since the flow solver computes the change of the conservative values of all cells by integrating their interface fluxes. This procedure is done following a routine that consists of following basic steps:

1. Computing fluxes through non-split non-boundary edges;
2. Computing fluxes through non-split boundary edges;
3. Computing fluxes through split faces by adding their subfaces' fluxes;
4. Computing the sum of six fluxes for every cell.

The computation of fluxes through a face requires only its two neighboring cells whose indices are explicitly saved for every face. This property is similar to an edge-based data structure. Since all information of adjacent connectivity is directly accessible, the flux evaluation can be simply vectorized. † However, to get the same information, the quadtree or octree structure often needs to climb up to the root of the branch and then climb down to neighboring cells, which is difficult to vectorize if not impossible. The searching process is avoided by using the present data structure. This strong point is shared by other steps under the data-face data structure. Note that because of step 3, step 4 simply accumulates the fluxes through six faces no mater whether the face is split or not while preserving conservation, which in programming means that no logical statement is necessary for judging neighboring connetivities.

It is clear that the data structure allows loop-level vector/parallel processing in both mesh adaptation and flow analysis. One may expect it is highly efficient on a shared-memory machine, but less efficient on a distributed-memory machine. The cache and communication factors will greatly affect the performance of loop-level parallelization.

†All flow solvers mentioned in the paper are vectorized except for the MUSCL scheme using the exact Riemann solver that contains uncertain iterations.

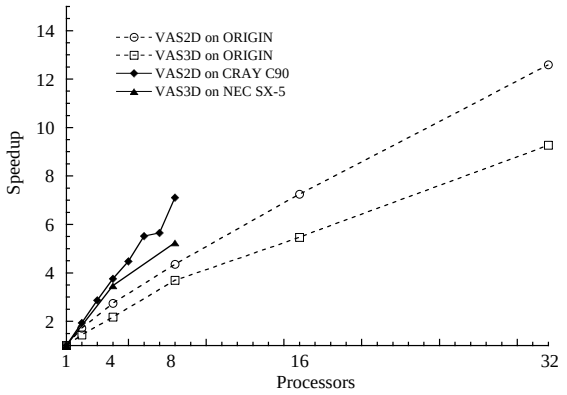


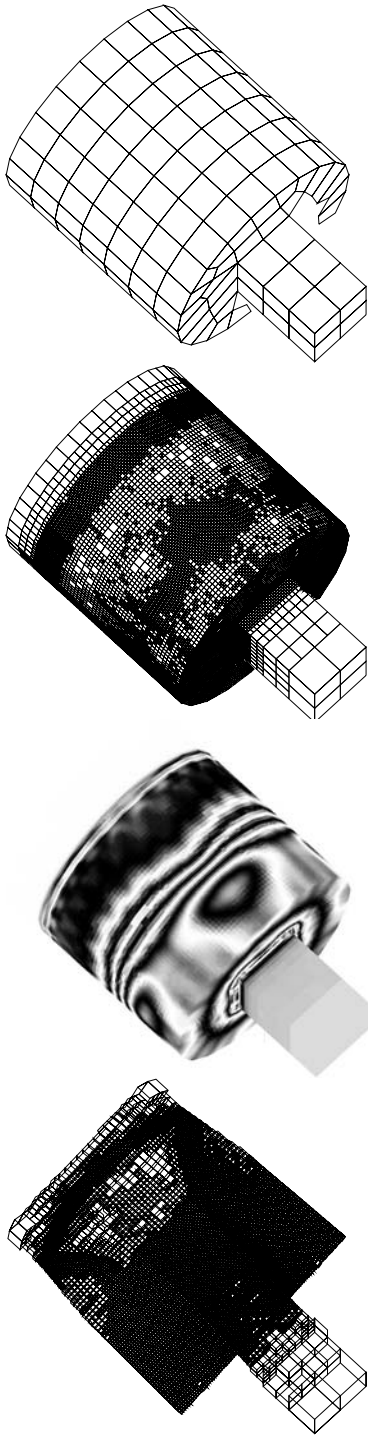
Figure 4: Performance of the 2-D and the 3-D solvers on machines, Cray C90, Origin 2000, and NEC SX-5

This will be shown by numerical tests in the next section.

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 4. On the shared-memory machines, NEC SX-5 and Cray C90, the data are collected by testing an unsteady computation in which grid adaptation is performed at every iteration. For eight processors the speedup is approximately six, which is not bad because the two solvers are automatically parallelized by the compliers coming with the machines. On the NEC SX-5, the speed per processor is about 2.1GFlops, which is 26% of the machine peak performance. On the distributed-memory machines, only steady flow computation is tested, and grid adaptation is conducted every 50 steps. It seems that the speedup achieved is not as high as that of other parallel unstructured codes that use 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.

Numerical results of a 3-D case are shown in



a.

b.

c.

d.

Figure 5: Shock wave entering a large 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 inside.

Fig. 5, 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 Figs. 5b and c; the total number of cells is increased to about 0.78 million. It takes about three minutes to run this computation on the 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. 5d; 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 successfully applied to the solution of the unsteady Euler equations. The high vector and parallel efficiency has been obtained on shared-memory machines. For cache-based distributed-memory machines, although the present technique allows easy loop-level parallelization without domain decomposition, it fails to take full advantage of the high-speed cache memory. We have obtained some ideas how to improve the situation, and will discuss them in future.

One factor that may hinder applying the technique to general problems is the level of automation of hexahedral meshing for arbitrarily complex geometry. Although it is not a problem for a simple tube geometry discussed in this paper, the amount of human effort necessary to discretize a general three dimensional domain possibly outweighs what one gains on accuracy and computational speed.[3] Much more work on the automation of hexahedral meshing is still needed.

References

[1] Aftosmis MJ, Gaitonde D, Sean Tavares T (1994) On the accuracy, stability, and mono-

tonicity 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] Blacker T (2000) Meeting the challenge for automated conformal hexahedral meshing, 9th Intl. Meshing Roundtable, New Orleans

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

[5] 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*

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

[7] 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.

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

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

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

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