INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **53**:1107–1125
Published online 20 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1357

An unstructured/structured multi-layer hybrid grid method and its application

Weimin Sang*, †, § and Fengwei Li‡, ¶

Northwestern Polytechnical University, Xi'an, Shaanxi 710072, People's Republic of China

SUMMARY

A multi-layer hybrid grid method is constructed to simulate complex flow field around 2-D and 3-D configuration. The method combines Cartesian grids with structured grids and triangular meshes to provide great flexibility in discretizing a domain. We generate the body-fitted structured grids near the wall surface and the Cartesian grids for the far field. In addition, we regard the triangular meshes as an adhesive to link each grid part. Coupled with a tree data structure, the Cartesian grid is generated automatically through a cell-cutting algorithm. The grid merging methodology is discussed, which can smooth hybrid grids and improve the quality of the grids. A cell-centred finite volume flow solver has been developed in combination with a dual-time stepping scheme. The flow solver supports arbitrary control volume cells. Both inviscid and viscous flows are computed by solving the Euler and Navier–Stokes equations. The above methods and algorithms have been validated on some test cases. Computed results are presented and compared with experimental data. Copyright © 2006 John Wiley & Sons, Ltd.

Received 27 May 2005; Revised 1 July 2006; Accepted 6 July 2006

KEY WORDS: multi-layer hybrid grid method; cell-centred finite volume method; dual-time stepping scheme; Navier–Stokes equations; Cartesian grid-method

INTRODUCTION

One of the greatest obstacles to widespread use of computational fluid dynamic (CFD) methods is the time required to create the grids used by the flow field solver [1–3]. For a grid-based approach,

*Correspondence to: Weimin Sang, School of Aeronautics, Northwestern Polytechnical University, P. O. Box 114, Xi'an, Shaanxi 710072, People's Republic of China.
†E-mail: xhasang@sina.com
‡E-mail: fwli@nwpu.edu.cn
§Associate Professor.
¶Professor, Member AIAA.

Contract/grant sponsor: Northwestern Polytechnical University

the effort involved in generating a grid for a complex CFD solution is an important factor to be considered.

At present, there are three approaches for dealing with the grids of complicated flow problems [1, 4, 5]: (1) the body-fitted structured grid can be used to map a complicated domain using a multi-block structure, a composite overlapping block structure or flexible grid embedding technique, (2) the unstructured grid method in which there is no concept of global or local coordinate directions, and which takes advantage of the geometric flexibility of triangles when defining a computational domain, (3) the Cartesian grid approach uses a recursive subdivision, and has the advantage of allowing the use of high accuracy solution methods that are difficult to develop for unstructured grids.

Coupled with grid adaptation, the Cartesian grid approach has been demonstrated to be a very viable tool for inviscid flows. The main advantages of the Cartesian grid methods [1, 2, 6] are: (1) automatic and straightforward grid generation; (2) easy incorporation in a grid adaptive refinement strategy to provide high accuracy solution of the flow features; (3) simple data structures; (4) relative simplicity of the resulting grids. Also, cell anomalies, such as highly distorted cells, can be easily eliminated.

The Cartesian grid generation process reduces the time and effort required to develop suitable computation grids for flow simulations. However, its wide application to viscous flow problems has suffered a setback because it is difficult to generate high-quality grids, with desired cell-aspect ratios, for the flow solver. Some alternative technologies have been sought to overcome these drawbacks because almost all engineering flow problems are viscous. One of the most notable is the hybrid grid approach [2, 3, 7]. Wang [8] developed a Cartesian/quad grid flow solver for 2-D viscous flow. Stolcis and Johnston [9] developed a flow solver for the Euler equations of 2-D compressible flow using unstructured grids. In our study, a Cartesian hybrid grid method and flow solver are developed, and extended for 3-D flow problems by solving the Euler and Navier–Stokes equations. The approach seeks to combine Cartesian grids with structured and triangular grids in order to exploit the advantage of each type of grid, and to avoid problems where the Cartesian grids are not efficient for viscous boundary layers.

In this paper, a flow solver for the computation of 2-D and 3-D flows is presented, based on a cell-centre finite volume method (CCFVM) and a dual-time stepping scheme. In the following sections, we will introduce the Cartesian grid generation and the hybrid grid technique, and show several flow problems.

CARTESIAN GRID GENERATION

The Cartesian grid method uses a non-body-fitted grid to discretize the flow field about an object. The Cartesian grid is generated from one large root cell that covers the entire computational domain. The root Cartesian cell is subdivided recursively until a user-specified minimum grid solution is obtained. After creating this surrounding grid, the surface geometry is simply cut out from the intersecting cells, generating a border of irregular cells surrounding the object surface.

Tree data structure

The tree data structure is the roadmap for the information during the course of the Cartesian grid generation [10].

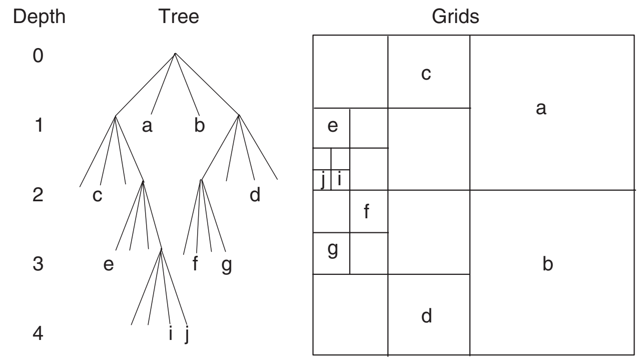

Figure 1. Tree data structure in two-dimensional grid generation.

A quadtree-based data structure is ideally suited for the two-dimensional Cartesian grid scheme. This concept is illustrated in Figure 1. The tree data structure starts with one root cell defined at depth 0. When this cell is refined, four children cells of equal size are created at one higher depth 1. In turn, these cells are refined over and over again until a suitable grid has been created. From this basic structure, all other necessary information about the relative position of any cell can be determined. In the three-dimensional Cartesian grid generation, we use an Octree data structure to store data information about nodes, lines, faces, and cells.

Cell-cutting algorithm

We generate the Cartesian grid using a cell-cutting algorithm. For computational efficiency, the grid generation is performed in six stages, and the following process is designed and implemented in a 3-D grid generation [11]. This process can also be used for two-dimensional grid generation.

(1) *Generating surface facets for a given body*: The facets are constructed in the form of triangles and planar polygons. All operation can be defined easily after the geometric entity has been implemented. The use of polygons eliminates the tedious and time-consuming tasks of intersection between surface facets and grid cells. The geometry facets are oriented such that the surface normal points into the computational domain.
(2) *Create coarse Cartesian background grid that encompasses the surface geometry*: Starting from the root cell, the child cells are recursively subdivided using a criterion based on the intersection with the defined geometry facets. Each face of every cell must be checked for the intersection. The cells that intersect the surface are subdivided. In the process of performing the cell subdivision, a tree data structure is established, linking the parent cells to the child cells.
(3) *Classify the types of cells*: Three main types of cell are defined in the computation domain. The Cartesian grid cells that intersect the body surface are called cut cells (CC), and those cells that are located inside the body surfaces are named solid cells (SC) and the rest of the cells are called flow cells (FC).
(4) *Form the closed cut cells*: A cell-cutting process is performed to generate a list of boundary facets, on which the boundary conditions are imposed. This process involves computing the

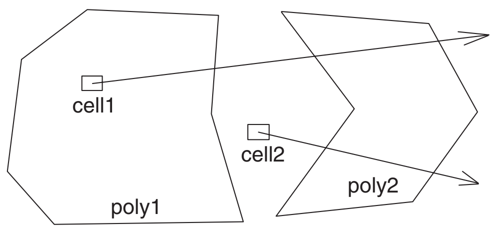

Figure 2. Inside/outside determination (2-D).

intersection of each geometry facet with each cell and generating boundary facets from the resulting intersection points. The final computational grid is then automatically produced through cell cutting.

(5) *Finish the geometry-based refinement*: The Cartesian cells are automatically refined, based only on the curvature of the geometry, until the appropriate solution is reached. We use a curvature rule to refine cells in the initial background grid. The rule is based on the maximum difference in the surface normal between two neighbouring cells. If the difference exceeds a user-specified tolerance, the two cells are marked for additional refinement.

(6) *Recursively repeat steps* (*4*) *and* (*5*) *until the maximum desired refinement level is reached*: The final grid cells with no children are the cells on which the flow calculation takes place. Solid Cells that lie completely inside the body are excluded from the computational domain. This is done using the two-step ray-casting algorithm, illustrated in two-dimensions in Figure 2.

First, a ray emanating from one point inside the cell in question is cast in an arbitrary direction. Second, the number of intersections that occur between the ray and a polyhedron is determined. If the ray intersects this polyhedron at even number of times, then the point must lie outside of the polyhedron (cell2 to poly2 in Figure 2). If the number of intersections is odd, then the point lies within the polyhedron (cell1 to poly1 in Figure 2). Note that this process may be used for both concave and convex polyhedron in three-dimension grid system.

HYBRID GRID GENERATION METHOD

The proposed method is designed to generate Cartesian grids, structured grids and triangular meshes. Each grid is constructed from an arbitrary number of face elements, allowing the unified treatment of cells. The structured grids may be better suited to the implementation of certain simulation features than unstructured grids. An example is highly stretched viscous grids and grid information in turbulence models.

Structured grid generation

The generation of high-aspect-ratio cells is only performed in the field region near the wall boundary. The grid generation process is done by advancing one layer of cells at a time, and carried out in the following steps.

First, the wall boundary is discretized base on a user-specified length scale.

Second, the surface vectors are determined at each point by first averaging the unit normal vectors of the faces sharing the point, and then smoothing the vectors by an operation through the following iterative scheme:

$$\mathbf{v}_F^{t+1} = (1-\omega)\mathbf{v}_F^t + \frac{\omega}{N}\sum_{i=1}^{N}(\mathbf{v}_F^t)_i \tag{1}$$

where $\mathbf{v}_F$ is a surface vector at point F, t represents the latest iteration level, ω is a relaxation parameter ($0<\omega<2$), N is the number of points connected to point F, and $(\mathbf{v}_F)_i$ is a vector at the neighbour point i.

Third, the distribution of grid points is generated along the surface vectors by a stretching function. The following geometric function is used to determine the grid spacing for each layer of cells:

$$\delta_m = \delta_1(1+\gamma)^{m-1} \tag{2}$$

where δ_m is the spacing for the mth layer, δ_1 is the spacing for the first layer of cells, and γ is a prescribed rate of stretching defined by the user.

Next, each layer of cells is constructed according to the above point distribution until crossings occur or a user-specified grid scale is obtained.

Finally, the structured grids are smoothed so that the expected computation grids are generated for the flow solver.

Unstructured triangular mesh

Unstructured meshes have been extremely successful in simplifying the complexity of discretizing inviscid flow fields. An unstructured grid is an arbitrary set of points which are connected to form cells which fill a region without overlaps. These cells can be made up of any number of points, although the simplest approach is to connect those sets of three points to form triangles in 2-D. Using an advancing-front method, the unstructured grids are typically generated by forming cells starting from the initial boundary and marching toward the interior of the computational domain.

The unstructured triangular grid generation algorithm was designed and implemented below:

(1) Create a background grid to control the distribution of the grid points on the surface and in the field domain.
(2) Discretize the boundaries to generate the initial advancing front according to the above background grids.
(3) Generate triangular grids and advance the front into the field until the entire domain is filled with contiguous cells, using the advancing-front method.
(4) Smooth and adjust these triangular cells.

Details of the generation algorithm are given by Kyle [4] and Pan and Cheng [12].

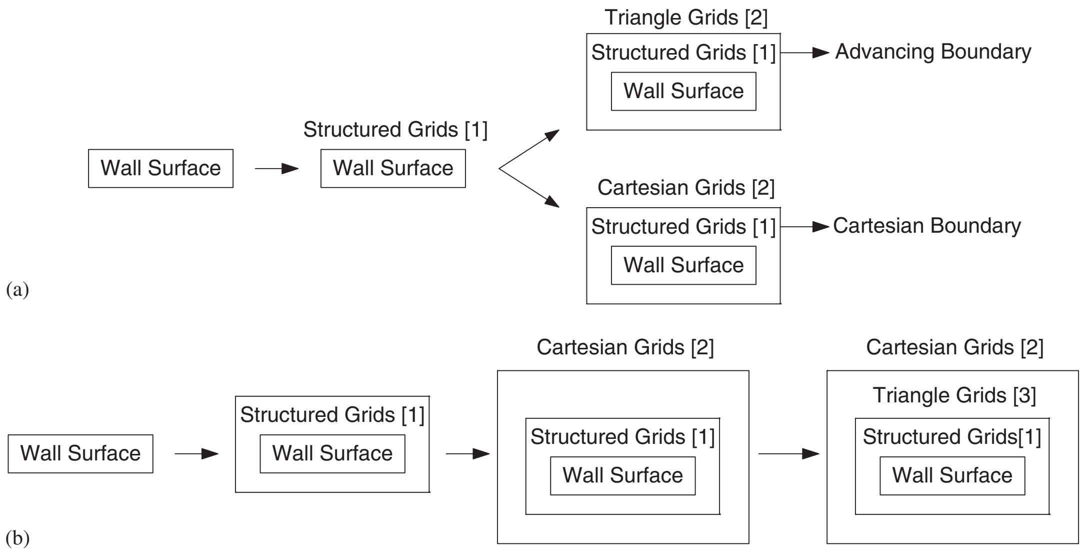

Figure 3. Illustration for generated scheme of hybrid grids: (a) hybrid scheme 1; and (b) hybrid scheme 2.

Hybrid scheme of grids

In this paper, we have two different schemes for generating hybrid grids. According to the characteristics of the configuration, either of the following two approaches is selected:

(1) Only two different grid types are generated in the computational domain as shown in Figure 3(a). A structured grid is generated close to the wall surface, typically consisting of several layers of quadrangle or prism grid. The rest of the domain is filled with a Cartesian grid or an unstructured triangle mesh.
(2) The concept of a hybrid scheme is illustrated in Figure 3(b). After a structured grid is created around the geometry, a Cartesian grid is generated in the far field based on the advancing boundary. Then, the void between the structured and Cartesian grids is filled with triangular cells. The hybrid approach can automatically trim the grid boundaries so that the Cartesian grid cutting process is eliminated.

Grid merging techniques

In this paper, grid merging techniques are implemented so that unwanted grid properties are eliminated. These properties can cause poor quality grids, and numerical stability problem. The grid merging techniques include two parts in the following. Since the update for the merged cell is based on the overlap area-weighted reconstruction (OAWR), the merging processes can guarantee the global flux conservation in CCFVM. Details of the OAWR can be found in Reference [13].

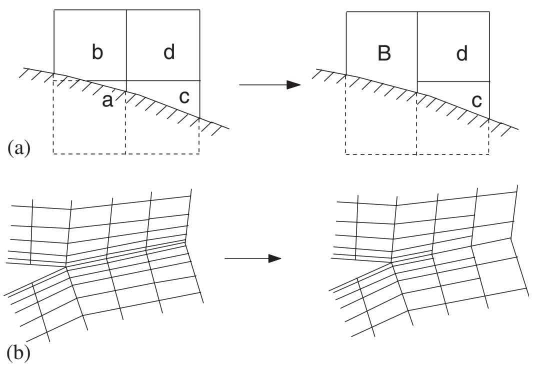

Figure 4. (a) Merging technique of two-dimensional cells; and (b) combination scheme of two-dimensional grids.

This process is illustrated with two-dimensional cells, but the process also works for three-dimensional cells [11]:

(1) Since a cut cell can be arbitrarily small in the Cartesian grid generation, a stability problem may arise at arbitrarily small cut cells. To avoid numerical instability, we need to eliminate these small cells. The idea is to merge a small cell with a neighbouring cell if their shared face is the largest face in the small cell. In Figure 4(a), a suitable merging cell for small cell a is cell b. After merging, cell a will disappear, and a larger cell B will be generated.
(2) In structured grid generation, the advancing layer process can create a grid-converging zone at the trailing edge of the configuration. The converging faces and points cause difficulty with the outside grid generation, and debase the grid quality. By using a combination process of layers, we can improve the advancing boundary and overcome this problem. Figure 4(b) illustrates the process. The present method can truncate the grid clustering at the trailing edge.

FINITE VOLUME FLOW SOLVER

A cell-centred finite volume flow solver, supporting arbitrary cells, was used to perform the flow simulation. In our case, this is a Jameson scheme [9, 12, 14] using Multi-stage Runge–Kutta integration and central differencing with added second- and fourth-order artificial viscosity [15, 16]. A cell line with a hanging node is actually treated as two separate lines, so the hanging nodes are invisible to the flow solver.

Our hybrid grid approach to solve the Navier–Stokes equations is based on the finite volume form of the integral equations. The equations are the expression of the conservation principle for mass, momentum and energy. In a domain of volume Ω with boundary S, the equations may be written in the following form

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{U}\,\mathrm{d}\Omega + \oint_S \mathbf{F}\cdot\mathbf{n}\,\mathrm{d}s = \frac{1}{Re}\oint_S \mathbf{F}_v\cdot\mathbf{n}\,\mathrm{d}s \tag{3}$$

where the vector of the conserved variables and both the inviscid and viscous flux vectors are given by

$$\mathbf{U}=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho E\end{pmatrix},\quad \mathbf{F}\cdot\mathbf{n}=\begin{pmatrix}\rho\mathbf{V}\cdot\mathbf{n}\\ \rho u\mathbf{V}\cdot\mathbf{n}+pn_x\\ \rho v\mathbf{V}\cdot\mathbf{n}+pn_y\\ \rho w\mathbf{V}\cdot\mathbf{n}+pn_z\\ (\rho E+p)\mathbf{V}\cdot\mathbf{n}\end{pmatrix},\quad \mathbf{F}_v=\begin{pmatrix}0 & 0 & 0\\ \tau_{xx} & \tau_{yx} & \tau_{zx}\\ \tau_{xy} & \tau_{yy} & \tau_{zy}\\ \tau_{xz} & \tau_{yz} & \tau_{zz}\\ \Phi_x & \Phi_y & \Phi_z\end{pmatrix} \tag{4}$$

where ρ, p, H and E are the density, pressure, total enthalpy and total energy per unit mass, respectively, and u, v, w are three components of the velocity vector $\mathbf{V}$. The variables in $\mathbf{F}_v$ can be found in Reference [17].

Note that the governing equation (3) becomes the Euler equation if the right-hand side of Equation (3) is equal to zero.

For the Cartesian or hybrid grids, the construction of the spatial discretization is based on the faces of a cell. Since the volume of any cell is supposed to be constant with respect to time, Equation (3) can be written as

$$\frac{\partial\mathbf{U}}{\partial t}=-\left(\oint_S\mathbf{F}\cdot\mathbf{n}\,\mathrm{d}s-\frac{1}{Re}\oint_S\mathbf{F}_v\cdot\mathbf{n}\,\mathrm{d}s\right)\Big/\!\!\int_\Omega \mathrm{d}\Omega \tag{5}$$

Performing the spatial discretization of the above equation leads to a set of ordinary differential equations with respect to time

$$\mathrm{d}U_k/\mathrm{d}t=-Q_k/\Omega_k \tag{6}$$

where index k denotes cell k. Ω_k is the volume of the cell k. U_k is the vector of the conserved variables. Q_k is a discrete expression of the flux integral, including inviscid flux and viscous flux, and is given by

$$Q_k=\sum_{i=1}^{I_{\text{edges}}}(Q_k^{\text{invis}}+Q_k^{\text{viscous}})_i \tag{7}$$

Note that the summation is over the edge faces (I_{edges}) forming the cell k. Using the cell-centred finite volume approach, the flux across the i edge face is calculated by simple averages of variables at the two neighbouring cell.

In order to eliminate the oscillations in the cell-centred schemes, artificial dissipative terms are added to the right-hand side of Equation (6) for the cell k, be written as

$$\mathrm{d}U_k/\mathrm{d}t=-(Q_k-D_k)/\Omega_k \tag{8}$$

Within the finite-volume approach, the dissipative function D_k is computed by a summation of the second and fourth terms across the edge faces

$$D_k=\sum_{i=1}^{I_{\text{edges}}}d_i^{(2)}+\sum_{i=1}^{I_{\text{edges}}}d_i^{(4)} \tag{9}$$

where

$$d_i^{(2)} = \alpha_i \varepsilon_i^{(2)} (U_r - U_k)_i$$

$$d_i^{(4)} = -\alpha_i \varepsilon_i^{(4)} (\nabla^2 U_r - \nabla^2 U_k)_i$$

where index i denotes the edge face delimiting the cell k and its neighbour cell r, and α_i, ε_i are the adaptive coefficient and scaling factor, respectively. Also,

$$\nabla^2 U_k = \sum_{i=1}^{I_{\text{edges}}} (U_r - U_k)_i$$

$$\varepsilon_i^{(2)} = k^{(2)} \max(v_r, v_k)_i$$

$$\varepsilon_i^{(4)} = \max(0, k^{(4)} - \varepsilon_i^{(2)})$$

$$v_k = \left| \sum_{i=1}^{I_{\text{edges}}} (p_r - p_k)_i \right| \Bigg/ \left| \sum_{i=1}^{I_{\text{edges}}} (p_r + p_k)_i \right|$$

$$\alpha_i = (|\mathbf{V} \cdot \mathbf{n}\, \mathrm{d}s| + c|\mathbf{n}\, \mathrm{d}s|)_i$$

where c is the local speed of sound, and $k^{(2)}$, $k^{(4)}$ are two chosen constants, which have values in the range $0.5 < k^{(2)} < 1$ and $1/256 < k^{(4)} < 1/32$.

The Baldwin–Lomax algebraic model is implemented to handle turbulence flows. This simple and common model shows fairly good convergence, and allows the use of comparatively coarse grids. Details of the model can be found in Reference [18].

DUAL-TIME STEPPING SCHEME

A dual-time stepping algorithm is used to improve the robustness, efficiency and accuracy of the flow solver. We can solve Equation (8) using the explicit four-stage Runge–Kutta time stepping scheme [9, 14]. At physical time step $n+1$, this equation can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}(U_k^{n+1}) + \frac{Q_k^{n+1} - D_k^{n+1}}{\Omega_k} = 0 \tag{10}$$

According to References [19, 20], a Newton-like subiteration is introduced through a pseudo-time τ to allow the flow problem for the pseudo steady state to be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}\tau}(U_k) + \frac{\mathrm{d}}{\mathrm{d}t}(U_k^{n+1}) + \frac{Q_k^{n+1} - D_k^{n+1}}{\Omega_k} = 0 \tag{11}$$

Using backward first-order and second-order accuracy time-difference formulae to discretize the above derivatives, Equation (11) becomes

$$\frac{U_k^{m+1} - U_k^m}{\Delta\tau} + \frac{3U_k^{n+1} - 4U_k^n + U_k^{n-1}}{2\Delta t} + R_k^{n+1} = 0 \tag{12}$$

where

$$R_k^{n+1} = \frac{Q_k^{n+1} - D_k^{n+1}}{\Omega_k}$$

then

$$\frac{U_k^{m+1} - U_k^m}{\Delta\tau} = -R_k^{m+1} \quad \text{where } R_k^{m+1} = \frac{3U_k^{n+1} - 4U_k^n + U_k^{n-1}}{2\Delta t} + R_k^{n+1} \tag{13}$$

where m denotes the pseudo-time.

The initial values for the subiteration are taken as $U_k^m = U_k^n$. Starting from $m = 1$, $U_k^1 = U_k^n$, the sequence of iterations U_k^m, $m = 1, 2, 3, \ldots$ converges to U_k^{n+1} when the right-hand side residual is equal to zero. Thus, through the subiteration in the pseudo-time, the following equation is valid:

$$\frac{3U_k^{n+1} - 4U_k^n + U_k^{n-1}}{2\Delta t} + R_k^{n+1} = 0 \tag{14}$$

At convergence of the pseudo-time iterations $\Delta U^m \to 0$, the accuracy of the solution at each physical time step is the accuracy of the discretized governing equations. That is to say, in the case of convergence, $U_k^{m+1} \to U_k^{n+1}$ and $R_k^{m+1} \to R_k^{n+1}$.

Substituting $R_k^{n+1} = (Q_k^{n+1} - D_k^{n+1})/\Omega_k$ into the above equation yields a fully second-order implicit scheme in time for the governing equation

$$\Omega_k \frac{3U_k^{n+1} - 4U_k^n + U_k^{n-1}}{2\Delta t} + (Q_k^{n+1} - D_k^{n+1}) = 0 \tag{15}$$

Experience with the method shows that the convergence rate within the pseudo-time process is very fast, and only a few subiteration steps are needed. For some steady-state flow problems, this dual-time stepping scheme allows for a large pseudo-time step enabling fast convergence.

GRID INDEPENDENCE TEST

The two-dimensional grid used in this test for modelling the region around the RAE 2822 airfoil. The grid includes a single Cartesian grid and a Cartesian/structured hybrid grid, shown in Figures 5(a) and (b). Both grids have far boundary set at 10c, where c is the chord, from the airfoil surface. The hybrid grid is composed of two blocks where the inner and outer blocks have an interface between 0.1c and 0.2c from the wall surface.

To develop the high quality simulation of aerodynamic properties, a series of grids having different resolutions are constructed. For the Euler solution of single Cartesian grids, based-geometry refinement time and minimum cell size before refinement are two important factors. For the N–S solution of hybrid grids, we will demonstrate the effect of minimum wall spacing and normal dimension in the inner block (the streamwise dimension is fixed with 110).

The first set of grids constructed (Table I) is used for an investigation of the single Cartesian grid independence, which was followed by a study on the hybrid grid (Table II). Table I shows that the lift values greatly change after based-geometry refinement. This suggests that grid refinement has a sufficient number of points to predict the lift value and also shows that any further refinement

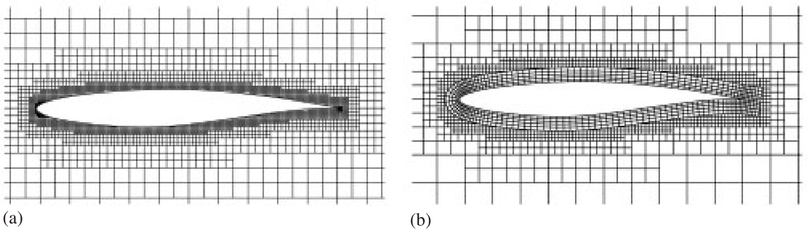

Figure 5. Computed grids used for the independence test: (a) single Cartesian grids; and (b) Cartesian/structured hybrid grids.

Table I. Computed lift coefficients for single Cartesian grids and euler equations ($M_\infty = 0.75,\ \alpha = 2.72°$).

		Minimum cell size before refinement (based on the chord length)			
Grid independence test		4.0×10^{-2}	2.0×10^{-2}	1.0×10^{-2}	0.5×10^{-2}
Based-geometry	0	0.80401	0.79512	0.79135	0.78810
refinement time	1	0.84974	0.84634	0.84621	0.84603
	2	0.85103	0.84641	0.84630	0.84612
	3	0.85059	0.84644	0.84631	0.84614

Table II. Computed lift coefficients for hybrid grids and N–S equations ($M_\infty = 0.75$, $\alpha = 2.72°,\ Re = 6.2 \times 10^6$).

		Minimum wall spacing (the first layer spacing of cells, δ_1)			
Grid independence test		10.0×10^{-5}	5.0×10^{-5}	1.0×10^{-5}	0.5×10^{-5}
Dimension of the	110×10	0.67552	0.66541	0.65801	0.65524
inner block	110×20	0.70768	0.70242	0.69500	0.68911
(streamwise	110×30	0.72253	0.71604	0.71214	0.71209
and normal)	110×40	0.72611	0.72043	0.71213	0.71211

beyond 2 times was not necessary. From this test, we decided to use the 1 time refinement with the minimum cell of 2.0×10^{-2} for the single Cartesian grids, and use the dimension of 110×30 with the minimum wall spacing of 1.0×10^{-5} for the hybrid grids, as the base line.

TEST CASES AND RESULTS

The main purpose of the test cases is to demonstrate the capability of the developed overall algorithms (grid generation, flow solver and hybrid technique). Four test cases are presented. We demonstrate our methods by showing the numerical solutions for these 2-D and 3-D flows and comparisons with the experimental data.

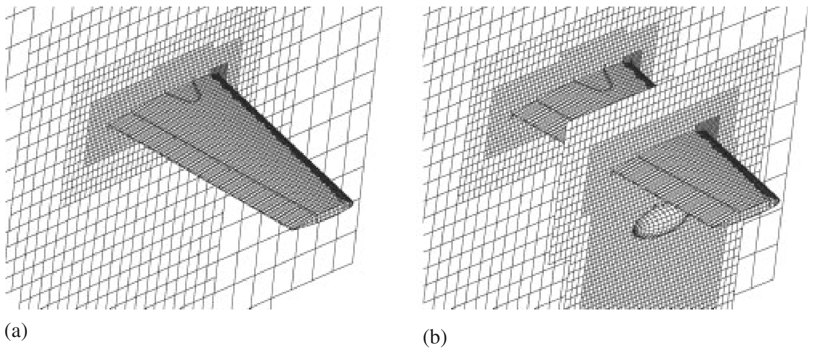
(a) (b)

Figure 6. Computed Cartesian grids: (a) without store; and (b) with store.

The four numerical calculations in the current work are performed on a Pentium 4 2.0 GHz class personal computer with 1 GB main memory. The convergence to a steady-state solution is assumed when there are no changes in the lift and drag coefficients to five significant digits with subsequent iterations.

Complex flow over wing/store configuration

This test case involves transonic flow and is used for the validation of the Cartesian grid generation method and flow solver. The geometry consists of a wing with an attached store. The store is a solid ellipsoid, with a long to short axis ratio of 3.0. The wing configuration is ONERA M6. The flow was computed by solving the Euler equations at a Mach number of 0.84, and an angle of attack of 3.06. The generated grids for the single wing configuration have 110 174 cells and are shown in Figure 6(a). When the store is located at the half semispan distance under the mid-section of the wing, the generated grids have 171 018 cells and are presented in Figure 6(b).

The computed pressure coefficient distributions are compared with experimental data in Reference [21] at six spanwise stations of the wing surface in Figure 7. We observe that results of the single wing simulation, without store, agree well with experimental data on the whole. Some disagreements exist at the upper surface of the wing, in particular at the station $2y/B = 80\%$. It is believed that the present flow solver is inadequate to calculate the flow using the Euler equations without viscous terms. In Figure 7, the computed pressure coefficient profiles with the store are shown at six stations of the wing surface. The store is shown to have a large influence on the pressure at the stations close to the store. On existing computer conditions, it takes about 1 and 1.5 h of CPU time to run 1000 iterations to converge for the two configurations, respectively.

Complex viscous flow over four-element airfoil

The second example considers flow over a four-element airfoil. The test flow conditions are $M_\infty = 0.2$, $Re = 9.0 \times 10^6$, $\alpha = 0.0°$ and $\alpha = 20.318°$ based on the chord length. The computational grid is shown in Figure 8, including three partial grid figures. It is obvious that the grid merging techniques improve the whole grid quality. The structured grids and unstructured triangle and Cartesian cells were generated in different domains. The above three different grids have 8056, 6062 and 916 cells, respectively. The body-fitted structured grid near the wall surface and the

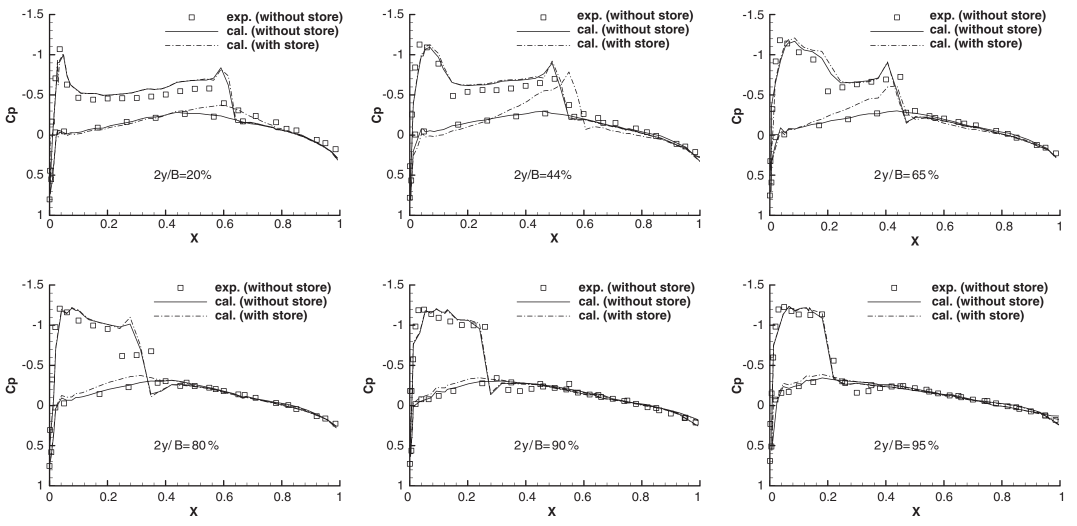

Figure 7. Comparison between computed and experimental wing surface pressure coefficient ($M_{\infty} = 0.84$, $\alpha = 3.06°$).

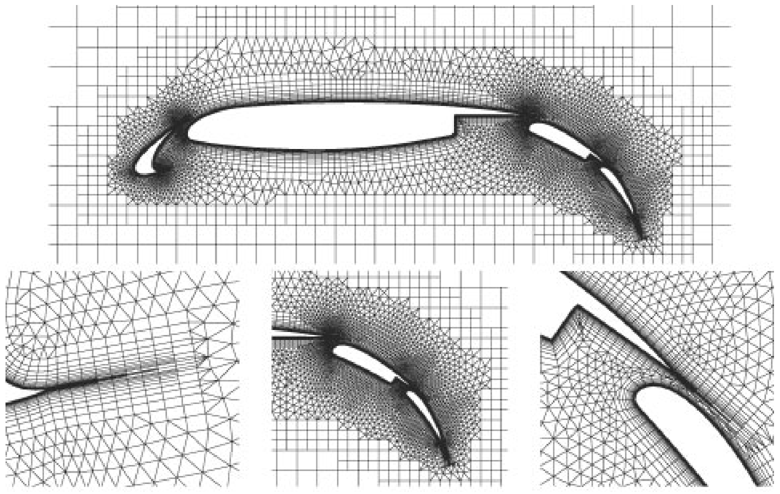

Figure 8. Cartesian/triangular/structured hybrid grids for four-element airfoil (including three partial grid figures).

Cartesian grid in the far field were generated as separate regions. The triangular mesh acts as an adhesive to link each part of the grid.

In Figure 9, the computed pressure coefficient profiles at the airfoil surfaces are compared with the experimental data in Reference [22]. It is observed that good agreement is obtained. In Figures 10(a) and (b), we can observe that contour lines and streamlines pass very smoothly

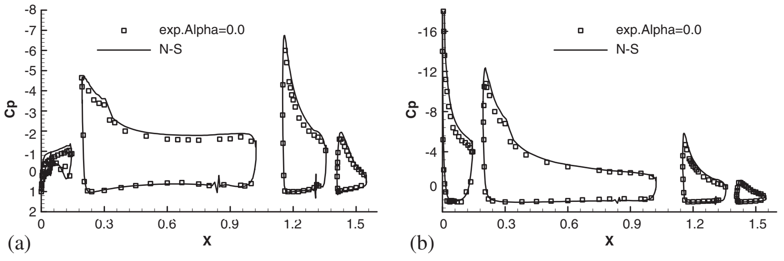

Figure 9. Computed surface pressure distributions on four-element airfoil ($M_\infty = 0.2$, $Re = 9.0 \times 10^6$): (a) $\alpha = 0.0°$; and (b) $\alpha = 20.318°$.

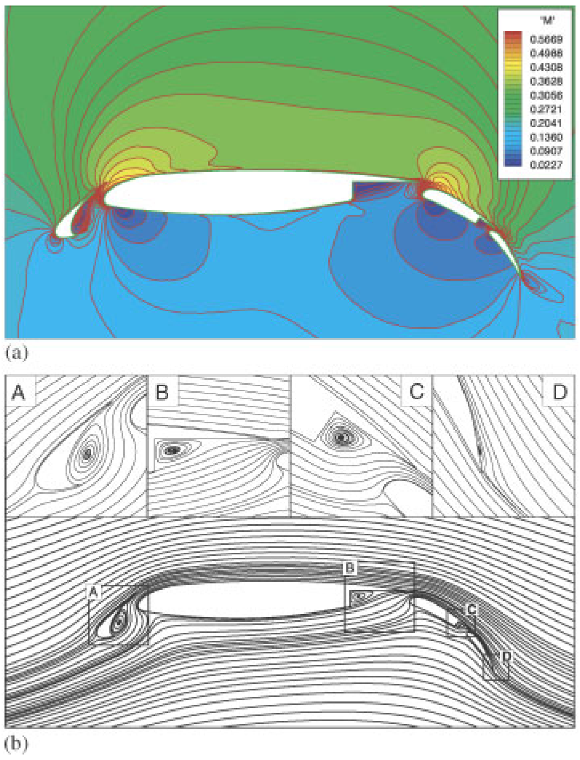

Figure 10. Computed contours and streamlines for four-element airfoil ($M_\infty = 0.2$, $\alpha = 0.0°$, $Re = 9.0 \times 10^6$): (a) Mach number contours; and (b) streamlines (including four partial figures).

from the structured grids to the Cartesian cells even in the presence of the triangular meshes. For the results figures, the structured and Cartesian cells were cut into triangles for visualization purpose [8]. The flow simulation takes about 25 minutes for approximately 1000 time steps to converge on existing computer conditions.

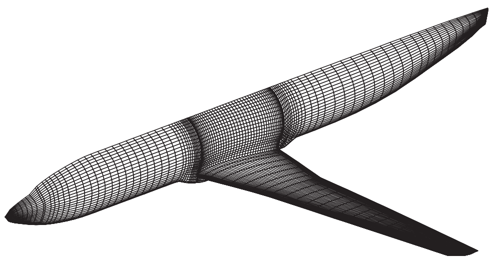

Figure 11. Surface grids for wing-body combination.

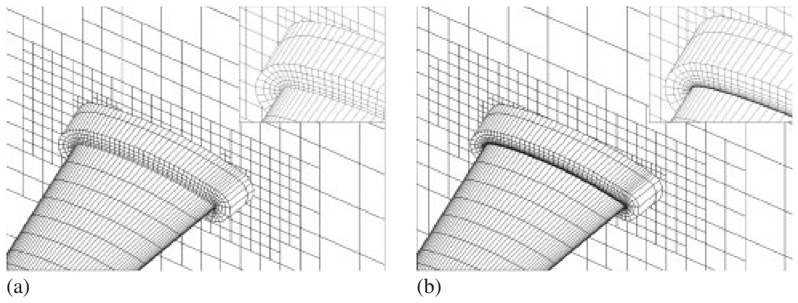

Figure 12. Cartesian/structured hybrid grids for wing-body combination (partial grids for wing-body configuration): (a) grids for Euler equations; and (b) grids for Navier–Stokes equations.

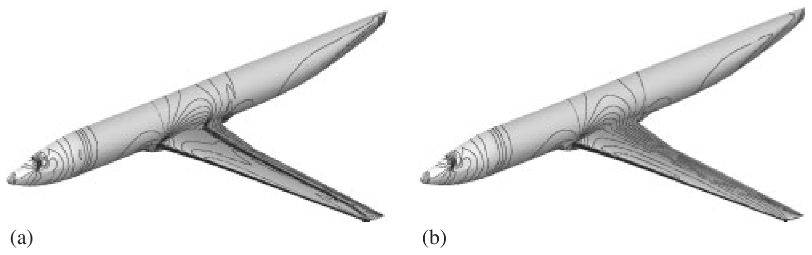

Figure 13. Computed surface pressure contours on wing-body combination ($M_\infty = 0.78$, $\alpha = 2.0°$, $Re = 2.5 \times 10^6$): (a) result of the Euler equations; and (b) result of the Navier–Stokes equations.

Transonic flow around wing-body combination

In Figure 11, the geometry and surface grids are presented for transonic flow around a wing-body geometry. Both inviscid and viscous grids for this case are shown in Figure 12 and have 230 768 and 410 334 cells, respectively, using the Cartesian/structured hybrid scheme.

The model has been experimentally studied in the high-speed wind tunnel at China Aerodynamics Research and Development Center (CARDC). The initial flow conditions are $M_\infty = 0.78$, $\alpha = 2.0°$, $Re = 2.5 \times 10^6$ based on the half span length. The pressure contours on the surface are shown in Figure 13. The computed pressure coefficients on the wing surface at several cross section stations

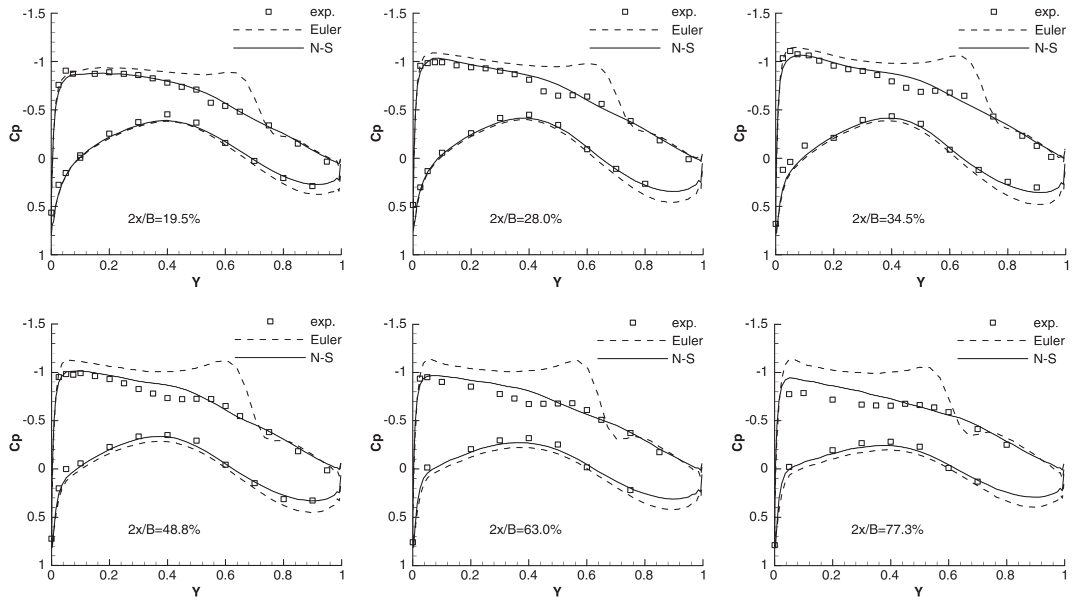

Figure 14. Comparison between computed and experimental surface pressure coefficient ($M_\infty = 0.78$, $\alpha = 2.0°$, $Re = 2.5 \times 10^6$).

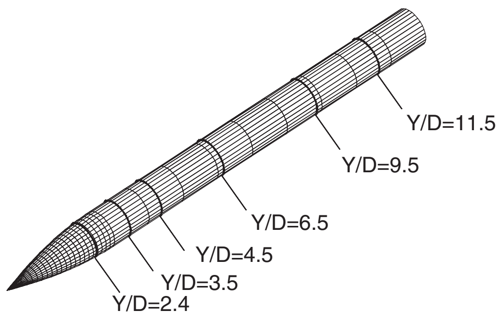

Figure 15. Ogive/cylinder geometry (including six test sections on the surface).

are compared with the experimental data in Figure 14. See Figures 13 and 14, for the shocks on the upper surface of wing, where the Euler solutions are stronger than the N–S solutions. It is observed that better agreement was obtained using N–S equations. These simulations take about 2.5 and 5 h of CPU time to run 1500 iterations to converge for the Euler and N–S equations, respectively.

The numerical results presented show the feasibility and validity of the hybrid grid generation and flow solver for the Euler and N–S equations in 3-D flow fields.

High angle of attack flow over ogive/cylinder configuration

This test case is performed for an ogive/cylinder model. The geometry of the configuration is displayed in Figure 15, including six pressure test sections on the surface. Both inviscid and viscous grids for the Euler and N–S equations are shown in Figures 16(a) and (b), and have 180 919 and 312 096 cells, respectively.

The experimental data in Reference [23] are used and conducted at a free stream Mach number 0.7 and angle of attack 14° and Reynolds number 0.6667×10^6. The flow field is characterized by large separation regions, and is a challenge for numerical simulation. Computed pressure coefficient profiles are compared with the experimental data at the six sections in Figure 17. Note that generally

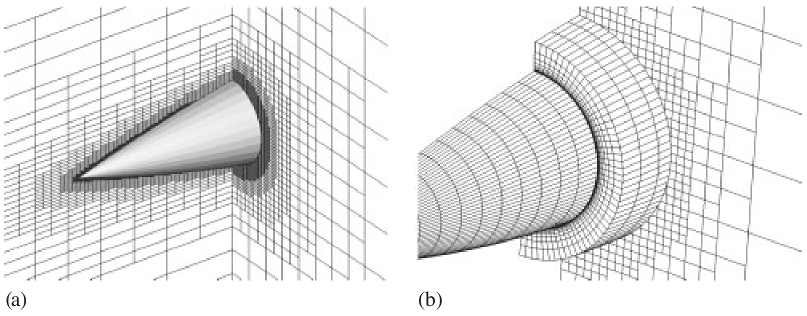

Figure 16. Computation grids for the Euler and Navier–Stokes equations (partial grids for ogive/cylinder configuration: (a) single Cartesian grids; and (b) Cartesian/structured hybrid grids.

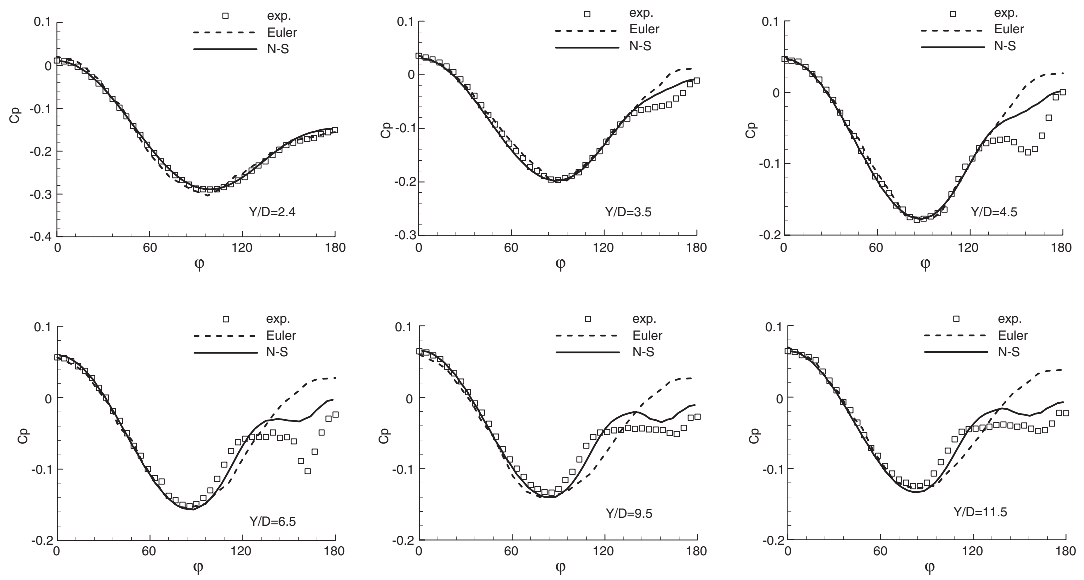

Figure 17. Comparison between computed and experimental surface pressure coefficient ($M_\infty = 0.7$, $\alpha = 14.0°$, $Re = 0.6667 \times 10^6$).

speaking the agreement is good, and the N–S solution is better than that of the Euler equations on the lee side of the model. An improvement in the turbulence models is needed to accurately capture the large separation in this case. The Euler and N–S equations computations take about 3 and 5.5 h, respectively, for approximately 2000 time steps to converge.

CONCLUDING REMARKS

The multi-layer hybrid grid method and a compatible flow solver have been developed and successfully demonstrated in this paper. The work presented here confirms the utility and flexibility of using a hybrid grid in conjunction with a finite volume method for the solution of the Euler and Navier–Stokes equations. The tree data structure and small grid merging techniques are very useful in the grid generation method. Some 2-D and 3-D test applications presented here demonstrate the accuracy and robustness of these approaches.

It is also found that the improvement in the turbulence models is needed to obtain more accurate predictions of highly separated flows according to References [24, 25]. The hybrid grid method is required for complete aircraft configuration. This work is now underway.

ACKNOWLEDGEMENTS

This study has been partially supported by the fund of 'The Developing Program for Outstanding Persons' at Northwestern Polytechnical University for the first author.

REFERENCES

1. Melton JE, Berger MJ, Aftosmis MJ, Wong MD. 3D Application of a Cartesian grid Euler method. *AIAA Paper 95-0853*, 1995.
2. Karman SL. SPLITFLOW: a 3D unstructured Cartesian/prismatic grid CFD code for complex geometries. *AIAA Paper 95-0343*, 1995.
3. Coirier WJ, Philip Jorgenson CE. A mixed volume grid approach for the Euler and Navier–Stokes equations. *AIAA Paper 96-0762*, 1996.
4. Kyle AW. A grid generation and flow solution method for the Euler equations on unstructured grids. *Journal of Computational Physics* 1994; **110**(1):23–38.
5. Thompson JF, Weatherill NP. Aspects of numerical grid generation: current science and art. *AIAA Paper 93-3539-CP*, 1993.
6. Yang G, Causon DM, Ingram DM, Saunders R, Battent P. A Cartesian cut cell method for compressible flows part B: moving body problems. *Aeronautical Journal* 1997; **101**(1002):57–65.
7. Noack RW, Steinbrebber JP. A three-dimensional hybrid grid generation technique. *AIAA Paper 95-1684*, 1995.
8. Wang ZJ. A quadtree-based adaptive Cartesian/quad grid flow solver for Navier–Stokes equations. *Computers and Fluids* 1998; **27**(4):529–549.
9. Stolcis L, Johnston LJ. Solution of the Euler equation on unstructured grids for two-dimensional compressible flow. *Aeronautical Journal* 1990; **94**(936):181–195.
10. Aftosmis MJ. Solution adaptive Cartesian grid methods for aerodynamics flows with complex geometries. von Karman Institute for Fluid Dynamics, *Lecture Series 1997-02*, Rhode-Saint-Genèse, Belgium, March 1997.
11. Sang WM. Numerical simulation of Euler and N–S equations using adaptive Cartesian hybrid grid method. *Ph.D. Thesis*, Northwestern Polytechnical University, 2003.
12. Pan D, Cheng JC. Upwind finite-volume Navier–Stokes computations on unstructured triangular meshes. *AIAA Journal* 1993; **31**(9):1618–1625.
13. Kathong M, Smith RE, Tiwari SN. A conservative approach for flow field calculation on multiple grids. *AIAA Paper 88-0224*, 1988.

14. Jameson A. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. *AIAA Paper 81-1259*, 1981.
15. Mahaian AJ, Dowell EH, Bliss DB. Role of artificial viscosity in Euler and Navier–Stokes solvers. *AIAA Journal* 1991; **29**(4):555–559.
16. Swanson RC, Eli T. Artificial dissipation and central difference schemes for the Euler and Navier–Stokes equations. *AIAA Paper 87-1107*, 1987.
17. Dome ND, Karman SL. SPLITFLOW: progress in 3D CFD with Cartesian omni-tree grids for complex geometries. *AIAA Paper 2000-1006*, 2000.
18. Baldwin B, Lomax H. Thin-layer approximation and algebraic model for separated turbulent flows. *AIAA Paper 78-0257*, 1978.
19. Gaitonde AL. A dual-time method for the solution of the unsteady Euler equations. *Aeronautical Journal* 1994; **98**(978):283–291.
20. Arnone A, Liou MS, Povinelli LA. Integration of Navier–Stokes equations using dual time stepping and a multigrid method. *AIAA Journal* 1995; **33**(6):985–990.
21. Schmitt V, Charpin F. Pressure distributions on the ONERA M6-wing at transonic Mach numbers. Experiment Data Base for Computer Program Assessment, *AGARD AR-138*, May 1979.
22. Anderson WK, Bonhaus DL. Numerical study to assess sulfur hexafluoride as a medium for testing multielement airfoils. *NASA Technical Paper 3496*, June 1995.
23. Sturek WB, Birch T, Lauzon M, Housh C, Manter J, Josyula E, Soni B. The application of CFD to the prediction of missile body vortices. *AIAA Paper 97-0637*, 1997.
24. Murman SM, Chaderjian NM. Application of turbulent models to separated high-angle-of-attack flows. *AIAA Paper 98-4519*, 1998.
25. Wang ZJ, Chen RF. Anisotropic Cartesian grid method for viscous turbulent flow. *AIAA Paper 2000-0395*, 2000.