A Cartesian Grid Generation Method Considering a Complicated Cell Geometry at the Body Surface*¹

By Paulus R. LAHUR*² and Yoshiaki NAKAMURA*³

Key Words: Grid Generation, Cartesian Grid, Thin Body, Sharp Edge, Cell Splitting

Abstract

A cell-splitting method for Cartesian grid generation that has the capability of taking into account the cases of thin body and sharp edge is proposed in this paper. Such cases are frequently found when solving the flow around a very thin wing, such as that of a supersonic transport (SST). The method has also been extended to treat the problem of multiple solid regions within a cell, which is sometimes encountered at a highly curved body surface. Validation of the method proposed here is carried out on a sharp, thin double wedge in a supersonic flow, where significant improvements in accuracy are achieved at the cost of a small increase in the number of cells. Furthermore, application of the present method to a model of SST shows its effectiveness on a three-dimensional, realistic geometry. As a result of making a pseudo-planar approximation for body surface elements, the total number of body surface elements was reduced by a factor of about 3.2 in this application. Local grid refinement by relocating grid cells to a curved surface is also proposed, so that a more accurate solution is obtained with a reasonable number of cells.

1. Introduction

The present study was motivated by an initial attempt to compute the inviscid flow around a model of supersonic transport (SST) using a Cartesian grid. This type of grid, known as non-body-fitted, was selected because of its inherent capability to treat a complicated geometry with less time and effort.

Cartesian grids have several weak points, one of which is seen in the very thin part of a body, such as the outboard wing of a SST. There, a cell might contain two flow regions separated by the thin body, each of which belongs to a different wing surface: upper and lower surfaces. If no distinctions between those two flow regions are made, a flow solver will mistakenly consider them as one flow region, leading to an erroneous solution.

Several approaches to overcome this problem have been explored in other researchers' studies. One way is to use multiple overlapping grids that are fitted to parts of the body.¹⁾ Since this requires additional user-defined grids, the advantage of simplicity inherent in Cartesian grids will be lost. The simplest approach is to refine the grid until each cell has a single flow region.²⁻⁵⁾ However, this approach is impractical as a large number of cells are produced and the cell size becomes prohibitively small, causing a steep rise in computational cost. Another approach is to use a hybrid of Cartesian and prismatic grids. The latter grid is employed to discretize the boundary layer, which intersects with a Cartesian grid outside it.⁶⁻⁸⁾ This approach also has its own problems, some of which stem from its dependency on body surface

mesh.

An efficient approach is to split the problematic cell into two sub-cells in such a way that each sub-cell contains a unique flow region. This type of cell-splitting approach was first introduced by Melton, Berger, Aftosmis and Wong.⁹⁾ An alternative algorithm to it is proposed in this study, where intersection between a body surface and a cell is simplified by employing a pseudo-plane approximation.¹⁰⁾ Moreover, the method will be extended to consider a cell with a sharp edge as well as a cell with two solid regions. Further improvement due to local grid refinement around a curved surface will also be discussed.

2. Grid Generation Method

The grid generation method proposed in this study assumes that a body surface consists of a collection of interconnected triangular body panels which form a closed, non-intersecting polyhedron within a computational domain, although the method can be adapted to other types of surface representation.

Generally, an intersection between a grid cell and a body surface has many levels of complexity which can be classified by the number of fluid or solid regions existing inside a cell (see Fig. 1). The present method can treat the cases ranging from classes 1 to 3, where a cell contains 1 to 3 distinct regions, respectively. Since the cases beyond class 3 are extremely rare and too complicated to manage, those cells will be further refined.

In this study, the coarsest grid cell is in the far field,

*¹ Received August 18th, 1999.

*² School of Engineering and *³ Department of Aerospace Engineering, Nagoya University, Nagoya, Japan.

Class	Schematic view of cell
1	<div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: white; margin-right: 10px;"></div> fluid <div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: gray; margin-left: 10px;"></div> solid
2	<div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: white; position: relative; margin-right: 10px;"> <div style="position: absolute; bottom: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to top right, transparent 49%, black 49%, black 51%, white 51%);"></div> </div> normal cut <div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: white; position: relative; margin-left: 10px;"> <div style="position: absolute; top: 0; right: 0; width: 100%; height: 100%; background: linear-gradient(to bottom left, transparent 49%, black 49%, black 51%, white 51%);"></div> </div> sharp edge
3	<div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: white; position: relative; margin-right: 10px;"> <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to top right, transparent 49%, black 49%, black 51%, white 51%);"></div> </div> thin body <div style="display: inline-block; border: 1px solid black; width: 40px; height: 40px; background-color: white; position: relative; margin-left: 10px;"> <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background: linear-gradient(to top right, transparent 49%, black 49%, black 51%, white 51%);"></div> </div> 2 solid regions
4	too complex to consider here

Fig. 1. Cases of intersection between grid cell and body surface according to the number of regions (fluid and solid) inside a cell.

which is called level 0. Approaching a body, its level increases, for example, up to level 7 for the double-wedge case shown later. At every increment in level, a cell is isotropically divided into 8 sub-cells. An octree data structure is thus employed to store the relations between a cell and its sub-cells, which is particularly convenient in conducting cell search operation.

The following sections describe the two major issues of the present Cartesian grid: (1) treatment of irregularly shaped cells at the body surface, and (2) local grid refinement around a curved body surface.

2.1. Treatment of irregular cells at the body surface A cell-centered, finite-volume flow solver, that requires the following information on cell geometrical properties is employed in this study: (1) volume, (2) face area, and (3) face normal vector. The computational algorithm of these properties varies from one class of complexity to another, as shown in Fig. 2. Note that pure flow cells with no solid part inside them can easily compute those properties owing to the simplicity of the Cartesian grid, whereas cells at the body surface, which partly contain a solid region, need complicated calculation, especially for three-dimensional (3D) form. Therefore, we propose a method for this treatment based on the following two assumptions:

1. A body surface is approximated to be pseudo-planar within a cell (see Fig. 3).
2. One cell contains one solid region and one flow region.

2.1.1. Pseudo-plane approximation The first assumption simplifies body surface representation within a cell so that the number of body panels can be minimized. The advantage of fewer body panels is reduction in the computation needed during the flow-solving stage. Due to the iterative nature of the solver, the saving of computation effort will increase with the number of

Fig. 2. Calculation procedure of cell geometrical properties.

Fig. 3. Pseudo-plane approximation of body surface within a cell: (a) original body elements, (b) intermediate triangles, (c) pseudo-plane.

iterations. Although fewer body panels also seem to reduce memory requirement, the overall memory of the cell-splitting method might actually increase as compared with non-cell-splitting methods since the cell-splitting method requires additional memory to store the coordinates of intersection between body panels and cell edges.

In other research dealing with the thin-body problem, where all body elements inside a cell are stored (see Fig. 3a), such simplification is not made. Therefore, more computational effort will be required.^{9,11-14)}

It should be noted that the pseudo-plane is an arbitrary polygon which is not planar generally. Thus, to calculate its area and normal vector, it needs some treatment. Here,

Fig. 4. Cell-splitting procedure: (a) identification of the number of body surfaces inside a cell, (b) sharp-edge treatment.

in this study, we first divide the plane into several intermediate triangles (Fig. 3b), and the values from those triangles are then averaged (Fig. 3c) to obtain the properties of the pseudo-plane.

The second assumption mentioned above makes the relation between cell and body surface unique, leading to efficient computation. However, the flow region is sometimes bounded by two body surfaces instead of one, as seen in the case of a highly concave body. This is quite rare, and the treatment of such will be discussed in sub-section 2.1.4.

2.1.2. Identification of the number of body surfaces within a cell Though normally only one body surface is contained within a cell at body surface, there might be two in the case of a thin body (see Fig. 1). In order to identify the number of body surfaces, first intersection points between the body surface and the cell edges are located (see Fig. 4a). These points will occur in pairs. After repeating this procedure for all six faces of the cell, those intersection points are interconnected to form a loop, which approximates a group of panels representing one body surface.

2.1.3. A cell containing normal geometry or sharp edge (Class 2) When there is only one body surface within a cell, the body geometry is classified into either normal geometry or sharp edge (see Fig. 2). Here a sharp edge is defined as a pair of neighboring body panels with $\theta < 30^\circ$, where θ is the angle made between these panels.

For the case of normal geometry, computing cell geometrical properties is simple since a polyhedron representing the flow region inside the cell can readily be constructed. For the sharp edge case, however,

Fig. 5. Two solid regions within a cell.

according to the first assumption mentioned above, the body inside the cell will be truncated (see Fig. 4b). This is undesirable since it deteriorates solution accuracy. To preserve the original shape as much as possible, the cell should be split along the sharp edge by placing a splitting panel, which is not a solid panel but a flow-through surface, in order not to change the physical flow. The splitting panel is constructed by extending a panel from the sharp edge so as to halve the edge angle. The geometrical properties are computed twice for this cell. That is, when one split flow region is considered, the rest of the original cell including the other split flow region is assumed to be a solid region.

2.1.4. A cell containing two flow regions or two solid regions (Class 3) A cell contains two body surfaces in the following two cases: (1) one solid region and two flow regions, or (2) two solid regions and one flow region (see Fig. 1). In case (2), the two body surfaces face each other (see Fig. 5). This condition can be written as follows:

$$\hat{n}_1 \cdot (\vec{X}_{c2} - \vec{X}_{c1}) > 0, \quad \hat{n}_2 \cdot (\vec{X}_{c1} - \vec{X}_{c2}) > 0 \quad \dots (1)$$

where $\vec{X}_c$ and $\hat{n}$ are the position vector of the centroid and normal vector of the pseudo planar body surface, respectively. Subscripts 1 and 2 denote each body surface.

In both cases, the cell is split into two sub-cells, where only one interface between fluid and solid regions is considered in the cell following the second assumption that there can only be one flow region and one solid region in a cell. The geometrical properties of the polyhedron of each flow region are then computed as in the case of normal cell geometry.

Although in case (2) the cell should not be split since it contains only one flow region, it is actually split to be consistent with the second assumption, where geometry properties in each sub-cell are modified so as to become the same as those of the original cell. However, they are re-merged in the stage of solving to give an identical solution equivalent to the original one (see Fig. 2). The merging scheme is outlined in section 3.

2.2. Local grid refinement around a curved body surface As the pseudo-planar approximation becomes less accurate at a curved body surface, more refinement is needed with increase in curvature. However, as the curvature approaches infinity at a sharp edge, excessive refinement will be required so that a large number of very small cells will be produced. Though this has to be

avoided, in other researchers' methods, local grid is more refined as the body surface becomes more curved. Those methods use the following criteria: (1) a cell size is larger than the characteristic size of the body panels it intersects,⁶⁻⁸⁾ (2) the angle between neighboring body panels within a cell is less than a certain limit,⁹⁻¹²⁾ and (3) a cell has more than one vertex of body panel.^{13,14)} These criteria cannot treat a sharp edge efficiently. Therefore, in this paper, we propose a new criterion to reduce the number of cells around a sharp edge where a cell with the body panel angle between two limits is refined. If the angle is less than the lower limit, it is split. To implement this, the absolute value of the cosine of angle θ between two body panels is employed, since it reaches a maximum of 1 at $\theta=0^\circ$ (sharp edge) and 180° (flat surface), and a minimum of 0 at $\theta=\pm 90^\circ$.

A curved body surface within a cell may be finely discretized such that the angle between neighboring panels tends to that of a flat surface. If the above criterion is applied only on adjacent body panels, the real body surface with curvature is truncated to a flat plate. To avoid this, all combinations of body panels within the cell whose minimum values are selected to judge the cell refinement are considered (see Eq. (2)). A limiting angle θ_{lim} is set to 30° in this study, that is, an edge with $\theta < \theta_{\text{lim}}$ is considered a sharp edge, which is treated by the method described in section 2.1.3. On the other hand, a cell containing body panels with $30^\circ \leq |\theta| \leq 150^\circ$ is refined. This is local grid refinement:

$$\min |\hat{n}_i \cdot \hat{n}_j| < \cos \theta_{\text{lim}} \quad \text{for } 1 \leq i, j \leq N_p, \text{ and } i \neq j \quad \dots (2)$$

where all combinations of i and j are considered. $\hat{n}_i$ and $\hat{n}_j$ are the normal vectors to body panels i and j , respectively, and N_p is the total number of body panels within a cell.

The methods described above will be used to generate a Cartesian grid for the problems of thin body, sharp edge and two solid regions.

3. Flow Solver

The Euler equations are solved to calculate inviscid, compressible flows on a Cartesian grid generated by the present method. The flow solver employed here is based on a cell-centered finite-volume scheme (see Eq. (3)) where the numerical flux is calculated using Hännel's flux-vector-splitting scheme:¹⁵⁾

$$\Omega \frac{\partial \vec{U}}{\partial t} + \sum_{j=1}^N \mathcal{F} \cdot \hat{n}_j S_j = 0 \quad \dots (3)$$

where Ω is the cell volume, $\vec{U}$ the solution vector, $\mathcal{F}$ the flux tensor, and N the number of cell surfaces. $\hat{n}_j$ and S_j are the unit normal vector and area regarding j -th cell face, respectively. The solution is advanced in time using a

three-stage Runge-Kutta method. Furthermore, convergence is accelerated using local time stepping.

To achieve a reasonable time step, an extremely small and irregular cell produced by the intersection of Cartesian grid with body surface is merged with its neighboring cell of larger size. A cell is defined as a small cell when the volume of the remaining part after being cut is less than 5% of that of the original cell. Merging is carried out by taking the summation with regard to mass, momentum and energy at all boundaries of the two cells, as shown in Eq. (4), by which the quantities passing through their common boundary are automatically cancelled. By substituting this into Eq. (3), the flow solution for the merged cell can be computed (see Eq. (5)), which is identical to the solution of the cell (see Eq. (6)):

$$\vec{F}_m = \left(\sum_{j=1}^{N_m} \mathcal{F} \cdot \hat{n}_j S_j \right)_m = \left(\sum_{j=1}^{N_1} \mathcal{F} \cdot \hat{n}_j S_j \right)_1 + \left(\sum_{j=1}^{N_2} \mathcal{F} \cdot \hat{n}_j S_j \right)_2 \quad \dots (4)$$

$$(\Omega_1 + \Omega_2) \frac{\partial \vec{U}_m}{\partial t} + \vec{F}_m = 0 \quad \dots (5)$$

$$\vec{U}_1 = \vec{U}_2 = \vec{U}_m \quad \dots (6)$$

where subscripts 1, 2 and m denote cells 1, 2 and the merged cell, respectively.

The same merging procedure is also used to merge the flow regions inside the cell for the case of two solid regions.

4. Test Cases

4.1. Double wedge To validate the present cell-splitting method, a double wedge with a thickness-to-chord ratio of 5% is employed. The flow conditions are Mach number 1.5 and zero attack angle. The grid around the double wedge is shown in Fig. 6, where the grid has 128 cells along its chord, which corresponds to refinement level 7.

To assess the effect of cell splitting, three cases are considered as shown in Table 1. Case A is a baseline method, where no cell splitting is performed.⁵⁾ In case B, the cell-splitting method is applied only to a cell with two flow region cells. In case C, the cell-splitting method is also applied to cells with a sharp edge.

It is seen from Table 1 that cell splitting increases the number of cells only about 0.4% compared with the baseline method, which is not significant.

The pressure distributions along the surface for all cases are shown in Fig. 7. It is clear that the flow solution is inaccurate in the leading and trailing edge regions in case A, where no cell splitting is performed. In case B, some improvement is achieved, except at the leading edge, where the truncation of the sharp edge produces a non-physical pressure jump. In case C, a great improvement is made.

The accuracy of the solution can be quantitatively assessed using the difference between the computed value and an analytical solution obtained from the oblique shock theory and Prandtl-Meyer expansion theory. Shown in Table 1 are the maximum and rms errors, which can represent local and global accuracies,

(a) Overall grid

(b) Close-up of leading edge

Fig. 6. Cartesian grid around a double-wedge:
(a) overall grid, (b) close-up of leading edge.

respectively. It is evident that case C has a significant improvement in both errors, where the maximum and rms errors are reduced to 70 and 40% of the baseline method, respectively. The remaining error seems to be caused by the smearing of expansion waves in the mid-chord region, because the flow solver employed produces numerical dissipation due to the first-order accuracy in space.

The time required to generate the grid using the baseline method is 10 min and 16 s on a NEC EWS 4800/360MP workstation with a R 4400/100 MHz processor. Using the cell-splitting method, it increases to 10 min and 47 s, which is not very significant. On average, it takes 2.74 ms to treat a normally cut cell, whereas a multiple-flow-region cell and a sharp-edge cell take 4.05 ms and 5.42 ms, respectively.

Table 1. Three cases for a double-wedge problem.

Case	Cell splitting		Number of cells	Error	
	2 flow regions	Sharp edge		max.	rms
A	no	no	89,760	0.1406	0.02394
B	yes	no	90,080	0.2441	0.02069
C	yes	yes	90,144	0.0979	0.00963

(a) case A

(b) case B

(c) case C

Fig. 7. Pressure distribution along the surface of a double-wedge for cases A, B, and C.

Fig. 8. Grid around the SST: (a) overall view, (b) cross-sectional view at $\eta=0.5$, (c) cross-sectional view at $\eta=0.5$ with local grid refinement near leading edge.

4.2. Supersonic transport model The present method was applied to a model of a supersonic transport, shown in Fig. 8a, to determine its effectiveness in a more practical, complicated configuration. The body surface consists of 9,906 triangular panels and 5,068 nodes for a semi-span model. Since the surface mesh of the model used here is not made for CFD analysis, the triangles are of irregular size and shape. If we generate a body-fitted grid, whether structured or unstructured, based on this surface mesh, it will be a hard task; that is, the grid quality is severely affected by it. A Cartesian grid, on the other hand, can be generated around any surface mesh immediately, since it is much less dependent on the surface mesh quality. The only parameters that a user needs to set are the dimension of computational domain, the size of coarsest grid (level 0), and the highest level of grid refinement. Once started, the grid generation proceeds automatically.

Two tests are performed for this application, the objective of which is to examine: (1) the efficiency of cell splitting in the flow around a 3D geometry, and (2) the effect of local grid refinement on accuracy. The flow is calculated under the conditions of $M=2.4$ and $\alpha=1.5^\circ$.

4.2.1. Cell splitting The grid is shown in Fig. 8a and

b, where grid refinement is based only on the distance from the body surface. Regarding grid resolution, the number of cells along the axial length of the aircraft is 538, which corresponds to a refinement level of 8. In other words, assuming that the actual length is in the order of 100m, the cell size on the body surface becomes about 19 cm. For comparison, two grids are generated by the non-cell-splitting and cell-splitting methods.

Figure 9a and b shows pressure distributions on the wing surface obtained by these methods. The pressure coefficient is plotted at 4 stations in the spanwise direction, that is, at $\eta=0.3, 0.5, 0.7$ and 0.9 , where η is the ratio of the spanwise length to the half-span. The thin-geometry problem is clearly seen even at $\eta=0.3$, especially near the trailing edge, and the solution is completely impractical at $\eta=0.9$ (see Fig. 9a).

By employing the cell-splitting method, a much more improved solution is obtained over the whole wing (see Fig. 9b). The flow solution near the leading edge, however, is less satisfactory, which is mainly due to low grid resolution in this region.

Using the cell-splitting method, the number of cells increases by 1.3%, that is, from 262,056 to 265,418, where the cells placed on the body surface account for 19 and

(a)

(b)

(c)

Fig. 9. Pressure distribution along the SST wing: (a) without cell splitting, (b) with cell splitting, (c) with cell splitting and local grid refinement.

20%, respectively, which is not significant (see Fig. 10). On the other hand, a considerable savings is achieved by approximating body surface inside a cell by a pseudo-plane, where the number of body elements has been reduced by a factor of 3.2 from 171,104 to 53,039. This implies that a cell on the body surface contained 3.2 original body elements on average.

The total number of flux surfaces which are composed of cell surfaces and body elements is 793,797 for the baseline method, whereas it is 681,268 for the cell-splitting method, which is a reduction of 14.2%. However, this advantage does not seem to lead to a significantly faster flow computation since there are other opposing factors that have a greater effect on the speed

Fig. 10. Computational costs in terms of grid cells and body surface elements.

of computation. One such factor is the size and irregularity of cells on the body surface, which lowers the CFL number, leading to slow computation. The typical flow computation time for the present configuration is on the order of one or two days.

To generate a grid, the current method takes a total of 1 h 24 min of CPU time, where the time for cell cutting accounts for 67.6%. In this procedure, all body panels are checked whether they intersect with a particular cell on the body surface. Since there are 53,039 cells on the body surface and 9,906 original body panels, the checking algorithm is invoked 525 million times, which accounts for 62% of the time for the cell-cutting procedure. A better approach would be to check only the panels in the vicinity of a particular cell. One such approach was proposed in Ref. 11, using a special tree data structure called an Alternating Digital Tree (ADT) to store the panels.

4.2.2. Local grid refinement Regarding further improvement of accuracy and efficiency, the grid is refined locally around the curved body surface, so that the cell refinement level varies from 7 to 9 along the surface, which corresponds to cell sizes of about 37 and 9 cm, respectively, for a SST with a length of 100 m. It is clear from Fig. 8b and c that the grid resolution is one level higher near the leading-edge portion, while one level lower around more flat portion.

The grid thus obtained contains 175,810 cells, which is a reduction of 34% over the cell-splitting method without local grid refinement. Of all the cells, 32,616, or 18.6%, intersect with the body surface. Regarding the original body surface elements, 87.2% of these cells have body panels with a connecting angle θ between 150 and 180°, which are considered to be nearly flat. On the other hand, the cells including a sharp edge with $0^\circ \leq \theta \leq 30^\circ$ account for 3%, all of which were split by the sharp-edge splitting method. Cells with $30^\circ \leq \theta \leq 150^\circ$ account for 9.8%, which are most refined. In particular, cells with

$80^\circ \leq \theta \leq 100^\circ$ account for 4.8%, which are considered to be the least accurate regarding planar approximation. From a small percentage of such cells and the fact that they are most refined, it is expected that the present grid can capture the flow properties efficiently and effectively, the results of which will be shown in the following.

By comparing Fig. 9b and c, it is evident that pressure rise near the leading edge becomes much sharper, whereas pressure distribution along the rest of the wing remains almost the same. The number of body elements is also reduced to 32,616, or 61% of the case without local grid refinement. It takes 1 MB to store the data on area and normal vector for all panels, whereas in the case without local grid refinement around a curved body surface, which was mentioned previously, it takes about 1.7 MB. This means that local grid refinement can distribute the cells much more efficiently.

However, it should be noted that even with the current refinement level, the grid resolution around the leading edge is not yet sufficient. There, the average cell size is about 9 cm, whereas the curvature radius of the leading edge is on the order of 1 cm. Thus, cells in this region will have to be refined further for practical use.

5. Concluding Remarks

A cell-splitting method in the Cartesian grid was proposed in this study to deal with multiple-flow or solid regions within a cell, which occurs in the thin part of a body. One of the features of the present method is to use pseudo-planar approximation for body elements within a cell, which can remarkably reduce computational cost. The method was further improved by considering the case of a sharp edge as well as the case of two solid regions. Moreover, a local grid refinement method was also introduced at curved surfaces for efficient use of the grid.

The present method was first verified for flow over a 5%-thick double wedge at $M=1.5$. The results showed good agreement with the theory. For this particular case, the cell splitting reduced local and global errors to 70 and 40% of those of the baseline non-splitting method, respectively.

The present method was also applied to realistic 3D geometry such as a SST configuration. The flow solution in the thin geometry region was improved by using only about 1% additional cells. Moreover, the number of body elements intersecting with the grid cells was reduced by a factor of 3.2. Finally, by using local grid refinement to relocate the grid to a highly curved surface, a further improved solution was obtained with even fewer cells.

Acknowledgement

The authors gratefully acknowledge that the SST surface geometry used in this study was provided by Kawasaki Heavy Industries, Ltd.

References

- 1) Epstein, B., Luntz, A. L. and Nachshon, A.: Cartesian Euler Method for Arbitrary Aircraft Configurations, *AIAA J.*, **30**, 3 (1992), pp. 679–687.
- 2) De Zeeuw, D. and Powell, K.G.: An Adaptively Refined Cartesian Mesh Solver for the Euler Equations, *J. Comput. Physics*, **104** (1993), pp. 56–68.
- 3) Pember, R. B., Bell, J. B., Colella, P., Crutchfield, W. Y. and Welcome, M. L.: An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions, *J. Comput. Physics*, **120** (1995), pp. 278–304.
- 4) Coirier, W. J. and Powell, K. G.: A Cartesian, Cell-Based Approach for Adaptively-Refined Solutions of the Euler and Navier-Stokes Equations, NASA CP-3291 (1995), pp. 207–224.
- 5) Ochi, A.: Numerical Calculation of SST Using Unstructured Grid, Master Thesis, Dept. of Aerospace Eng., Nagoya Univ., Japan, 1996 (in Japanese).
- 6) Finley, D. B.: Euler Technology Assessment Program for Preliminary Aircraft Design Employing SPLITFLOW Code With Cartesian Unstructured Grid Method, NASA CR-4649 (1995).
- 7) Finley, D. B. and Karman, S. L., Jr.: Euler Technology Assessment for Preliminary Aircraft Design—Compressibility Predictions by Employing the Cartesian Unstructured Grid SPLITFLOW Code, NASA CR-4710 (1996).
- 8) Karman, S. L., Jr.: Unstructured Cartesian/Prismatic Grid Generation for Complex Geometries, NASA CP-3291 (1995), pp. 251–270.
- 9) Melton, J. E., Berger, M. J., Aftosmis, M. J. and Wong, M. D.: 3D Applications of a Cartesian Grid Euler Method, AIAA-95-0853 (1995).
- 10) Lahur, P. R. and Nakamura, Y.: A New Method for Thin Body Problem in Cartesian Grid Generation, AIAA-99-0919 (1999).
- 11) Aftosmis, M. J.: Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries, Computational Fluid Dynamics VKI Lectures Series 1997-05 (1997).
- 12) Aftosmis, M. J., Berger, M. J. and Melton, J. E.: Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry, *AIAA J.*, **36**, 6 (1998), pp. 952–960.
- 13) Deister, F. and Hirschel, E. H.: Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometries, AIAA 99-0782 (1999).
- 14) Deister, F., Rocher, D., Hirschel, E. H. and Monnoyer, F.: Three-Dimensional Adaptively Refined Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries, 4th European CFD Conference, Greece, vol. 1, part 1, 1998, pp. 96–101.
- 15) Hännel, D., Schwane, R. and Seider, G.: On the Accuracy of Upwind Schemes for the Solution of the Navier-Stokes Equations, AIAA Paper 87-1105, *Proc. AIAA 8th Computational Fluid Dynamics Conference* (1987), pp. 42–46.