Journal of Aircraft
Vol. 44, No. 5, September–October 2007

Cartesian-Like Grids Using a Novel Grid-Stitching Algorithm for Viscous Flow Computations

Partha Mondal,* N. Munikrishna,* and N. Balakrishnan†
Indian Institute of Science, Bangalore 560 012, India

DOI: 10.2514/1.27854

A novel grid-stitching algorithm has been developed for generating Cartesian-like grids for viscous flow calculations. A grid is generated by recursive division of Cartesian cells. Unlike the conventional Cartesian-mesh calculations that involve unit aspect ratio cells, stretching is used to get the high aspect ratio cells and smooth grid near the body. The grid data are collected in an unstructured format. This novel approach, along with an unstructured mesh-based flow solver (HIFUN-2D, developed in-house), has been successfully used for numerical simulations. This grid is successfully used for Euler and laminar flow computations. Partial success has been achieved for turbulent flow simulations.

I. Introduction

THE developments made in the past decade in the area of unstructured mesh calculations have made computational fluid dynamics (CFD) sufficiently mature in providing the designer with quick estimates of design parameters. In spite of this advancement, there is sufficient scope for further enhancing the efficiency of the CFD tools, particularly in the area of grid generation. Indeed, any cut in time to generate a good grid can make CFD more useful in offering quicker estimates of design parameters. This requirement made the CFD community reinvent the Cartesian grid [1] and is the subject matter of this work. The foremost advantage of using a Cartesian mesh is the automation that the procedure offers in grid generation. Apart from this, Cartesian mesh in conjunction with tree data structure becomes a natural choice for solution-adaptive grids and dynamic flow computations involving moving bodies. Several inviscid [2–5] and viscous [2,4,6–15] flow computations have been reported using Cartesian meshes. In spite of the success involving inviscid flow computations, the use of Cartesian meshes for viscous flows was not straightforward and involved several modifications of the basic scheme. The conventional Cartesian-mesh calculations involve unit aspect ratio cells and result in the appearance of small cut cells close to the body. A jump in cell volumes (particularly, close to the body) can lead to oscillatory wall data [2]. Apart from this, it is also not practical to fill the boundary layer with unit aspect ratio cells, particularly, for 3-D turbulent flows. More important, the lack of positivity of the viscous discretization procedure can lead to loss in monotonicity, not just in the wall data, but also in regions of embedded mesh [2]. One of the means to overcome this problem is the use of hybrid grids, wherein the region of viscous layer is filled with structured grids comprising high aspect ratio quadrilateral volumes, and the potential flow region is filled with unit aspect ratio Cartesian grids [4,11]. Other methods to generate grids that look like hybrid grids [7,12] involve generating a Cartesian-grid front close to the body (by employing some strategy for volume deletion) and obtaining the viscous grid close to the body by a suitable projection procedure. In our view, any of the aforesaid strategies can seriously hamper automated grid generation.

More recently, there have also been attempts to use Cartesian grids in conjunction with meshless solvers [16–19]. In [17,18], although finite volume-based computations are made for most of the computational domain, in a layer in the vicinity of the wall boundary, meshless solvers are used. These attempts are limited to inviscid calculations. It has been demonstrated in [20] that meshless solvers also suffer from a lack of positivity of viscous discretization akin to the finite volume method for Cartesian-mesh calculations. The development of robust meshless solvers for viscous flow computations continues to be a topic of research [19]. Therefore, the success of procedures combining the finite volume and meshless solver methodologies for viscous Cartesian-mesh calculations can critically depend on further development in algorithms for meshless solvers.

One of the interesting advancements in Cartesian-grid calculations pertains to the development of immersed boundary techniques [21–25]. These methodologies, primarily developed for incompressible flow calculations, have recently been extended to compressible flows involving moderate Reynolds numbers [26]. Further research in this area is required for exploiting its potential for industrial flow computations.

Therefore, the search is still on for superior Cartesian-grid strategies for obtaining solution to viscous flows. The novel grid-stitching algorithm proposed in this work for generating what are referred to as Cartesian-like grids is an effort in this direction. In this paper, we discuss the use of Cartesian-like grids in conjunction with the finite volume method.

There are two key features of structured mesh that have made it suitable for viscous flow computations. They are 1) high aspect ratio volumes and 2) a positive viscous discretization procedure resulting from simple central differencing. Although the former restricts the total number of cells required for resolving the viscous layer, the latter renders code the required robustness. The past experience that the CFD community has gained in terms of using the hybrid grid for both unstructured triangular mesh and Cartesian-mesh calculations clearly reveals the importance of employing a structured grid in the viscous layer. Inspired by the success of the structured grid methods, we have evolved a new grid-generation strategy, called the grid-stitching procedure [13–15], that obviates any special strategy for filling high aspect ratio quadrilateral volumes in the viscous layer. The present strategy employs a stretched Cartesian mesh over streamlined bodies, unlike the conventional procedures that employ unit aspect ratio cells. This feature is particularly useful for resolving the viscous layer. In addition to this, we have employed a point-movement strategy, by which certain Cartesian-grid points are moved onto the wall; this avoids small cut cells. We have demonstrated that such a grid-generation strategy can generate a sufficiently smooth grid close to the wall. We distinguish the grids generated using the present procedure from those obtained using

Received 16 September 2006; revision received 17 November 2006; accepted for publication 30 November 2006.

*Research Scholar, Computational Aerodynamics Laboratory, Department of Aerospace Engineering.

†Assistant Professor, Computational Aerodynamics Laboratory, Department of Aerospace Engineering; nbalak@aero.iisc.ernet.in (corresponding author).

conventional Cartesian-mesh generators by calling them Cartesian-like grids. The efficacy of the procedure for solving laminar flows has been clearly brought out by solving a number of test cases. In our view, this is a significant development, particularly considering the renewed interest of the aerospace community in laminar flows past streamlined bodies. We have had limited success in solving turbulent flows. At this stage, it is worthwhile to remark that it is impossible to get physically meaningful wall data from a viscous computation, even for laminar flows using conventional all-Cartesian-grid algorithms [6]. Though, in spirit, the present work is similar to one of the earlier works of Hassan [9], it is different in algorithmic details. The readers are also referred to an interesting development of a mesh adjustment scheme for embedded boundaries [27] involving a point-movement strategy, similar to the one presented in this work.

In the following section, we present the finite volume methodology used in computations. The grid-stitching algorithm is described in Sec. III. Section IV is devoted to numerical results and discussions. Conclusions and future directions are presented in Sec. V.

II. Numerical Method

In the finite volume method, we discretize the computational domain into a set of nonoverlapping volumes called the finite volumes, and the governing equations are integrated over the finite volumes. The governing equations in conservation form can be written as

$$\frac{\partial U}{\partial t} + \nabla \cdot \boldsymbol{F} = 0 \tag{1}$$

where $\boldsymbol{U}$ is the vector of conserved variables, and $\boldsymbol{F}$ is the flux vector. Expressing Eq. (1) in integral form over an arbitrary finite volume Ω_i, we obtain the following space-discretized equation:

$$\frac{\mathrm{d}\bar{U}_i}{\mathrm{d}t} = -\frac{1}{\Omega_i}\sum_J F_{\perp J}\triangle S_J \tag{2}$$

where J refers to the interface between cell i and its neighboring cell j, $F_{\perp J}$ is the normal flux through the Jth interface, and $\triangle S_J$ is the area of the Jth interface. The solution is updated using an implicit time-integration procedure [28].

In the finite volume method, the gradient at the cell centroid is required for the purpose of linear reconstruction of the solution to be used for inviscid flux calculations. The gradient at the face midpoint is also required for viscous flux computation. In this work, we have used a diamond path reconstruction procedure [29,30]. In this method, a covolume is formed around an edge, using the points forming that edge and the centroids of the cells sharing that edge. In Fig. 1, covolumes around the edges of a typical Cartesian cell resulting from the present grid-generation strategy are presented. The gradient of any scalar ϕ at the centroid of the covolume can be obtained by the discrete form of Green's theorem:

$$\nabla\phi_J = \frac{1}{\Omega_J}\sum_k \phi_k \hat{n}_k \Delta s_k \tag{3}$$

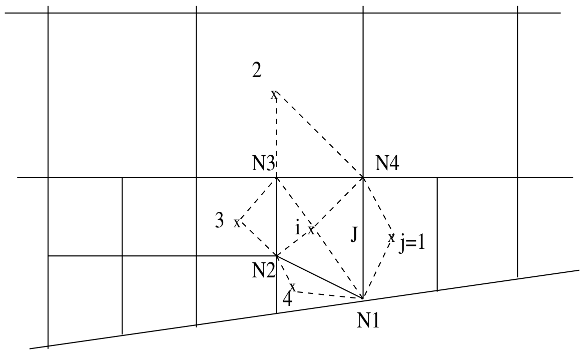


Fig. 1 Diamond path reconstruction.

where Ω_J is the area of the covolume around face J, and $\hat{n}_k$ and Δs_k are the unit normal and length of the kth edge of the covolume, respectively. To get the ϕ value at the Gaussian point of the edge forming the covolume, we have to compute the value of ϕ at the nodes. The values of scalar ϕ at the nodes can be computed by using different interpolation strategies such as volume-weighted averaging or inverse volume-weighted averaging or the use of a pseudo-Laplacian [29–31]. For the purpose of inviscid flux computations, the gradient at cell i is obtained by the area averaging of the gradients at covolumes associated with each of the edges forming the cell under consideration [29,30]. The computed gradient is limited to satisfy the monotonicity condition by using a Venkatakrishnan limiter [32].

In the cell-centered finite volume framework, boundary conditions are satisfied by computing appropriate fluxes for the cell faces falling on the boundary. For inviscid flows, a mirror boundary condition [33] is applied on the wall. For viscous flows, a no-slip condition is enforced on the wall. Also, the wall is assumed to be adiabatic. The pressure on the wall boundary faces is obtained by extrapolation from the interior. At the far field, the Reimann boundary condition [34] is used.

III. Cartesian Grid-Stitching Algorithm

To begin with this strategy, a coarse Cartesian grid is generated. This grid is adapted recursively, until the grid is fine enough to resolve a predefined geometric length scale (associated with body curvature) and a physical length scale (associated with the flow). The point movement is effected on such a fine grid. The procedure is presented in the following sections.

A. Creating Initial Background Cartesian Grid

Here, a background coarse Cartesian grid is generated in a box just enclosing the body of interest. This initial grid is extended to the far field using an appropriate number of blocks, allowing for a maximum of one hanging node on any given edge falling on the block boundary. This strategy effectively curbs the fineness of the body grid from getting propagated to the far field. In the first block, we may use stretching in both horizontal and vertical directions or either one of them. Also, a vertical and a horizontal line are passed through the convex corner points of the body. This initial grid is subjected to recursive division, the details of which are discussed in the next subsection. The recursive cell division module is based on a strategy employing edge- and cell-based unstructured data, unlike the routinely used tree data. Such a strategy has been chosen to exploit some of the preexisting unstructured-data-based adaptive routines available in the lab. They are also expected to be computationally more efficient than tree-data-based routines. It is emphasized that no point movement is effected at this stage. A typical background mesh over a NACA 0012 airfoil is shown in Fig. 2.

B. Recursive Cell Division

The generation of background Cartesian mesh consists of recursive division of cells. To begin with, all the Cartesian cells

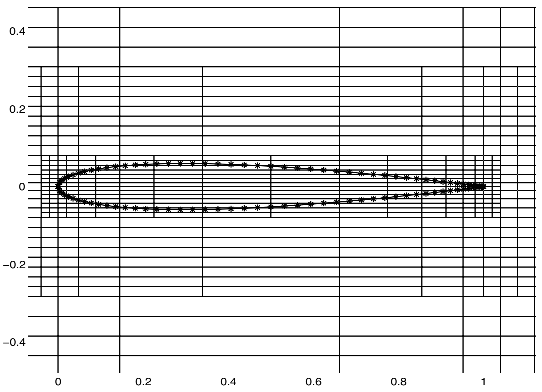


Fig. 2 Background initial Cartesian grid over a NACA 0012 airfoil.

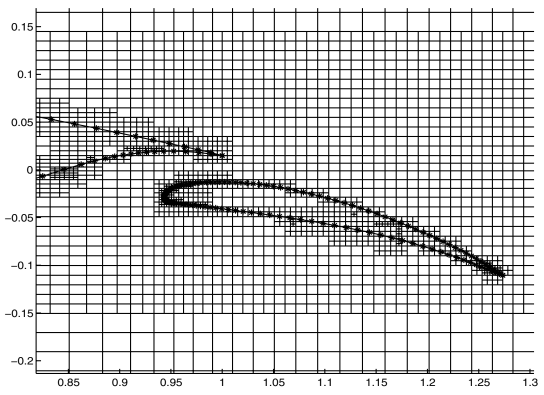


Fig. 3 Zoomed view near the flap region.

present in the initial block are considered to be at level zero. When a cell is refined, the level of all its offspring is incremented by one. Presently, the level difference between two cells sharing a common edge is not allowed to be greater than one. This restricts the number of hanging nodes on a given edge to one. Also, a maximum of one hanging node is allowed for a body-cut cell. This enforces the smoothness in the Cartesian grid all along the body. Also, it ensures a smooth transition from the fine cells near the body to coarser cells away from the body, as shown in the Fig. 3. Though the present strategy involves only isotropic refinement of cells, its extension to anisotropic refinement is trivial. In the present strategy, the cells are divided in such a way that will resolve the body geometry as well as flow physics. The division of a Cartesian cell depends on 1) a predefined length scale associated with the body geometry (e.g., the length of the segment defining the body, in a 2-D case) and 2) a physical length scale based on an estimate of boundary-layer thickness for a given flow. The second condition can be dispensed with for grids to be used in inviscid flow computations.

The division process to resolve the body geometry is as follows: All the cut cells (the Cartesian cell that intersects the body, as shown in Fig. 4) for which the size is larger than the local length scale associated with the geometry are refined. The size of the Cartesian cell is defined as the square root of its area. A Cartesian cell can have multiple body cuts; in this case, the length corresponding to the smallest body segment is chosen for comparison. The points on the body surface are distributed so that they are clustered in the regions of high curvature. After the completion of cell refinement based on geometric criterion, cells are refined for flow physics. For this purpose, a length scale associated with each cut cell is computed. The cut cell, for which this length scale is greater than a predefined length scale associated with flow physics, is flagged for refinement. Presently, the length scale associated with flow physics is obtained from an estimated boundary-layer thickness δ. Requiring m layers of grid lines within the thickness δ, the flow-associated length scale can be given by δ/m. For a conservative estimate of δ, we assume $\delta \sim Re^{-1/2}$, where Re is the Reynolds number. We suggest the following two ways of computing the length scale associated with the cut cell:

1) The corner points of the cut cell are projected onto the body surface normal defined at midpoint Q of the body segment, as depicted in Fig. 5a; the maximum of these projected lengths defines the length scale.

2) The projection of the cut cell centroid on the body surface normal defines the length scale, as shown in Fig. 5b. The latter will result in relatively less number of cells and has been used for generating viscous grids in the present work.

The preceding strategy merely provides the first grid for starting a solution-adaptive grid refinement. This procedure particularly fits in the context of solution-adaptive grid refinement/derefinement. In this paper, solution-based sensor parameters such as $\nabla.\boldsymbol{u}$ and $\nabla\times\boldsymbol{u}$ are used for identifying the cells for further refinement. Interestingly, in this procedure, refinement or derefinement is always carried out on a background Cartesian mesh and the point movement is effected after every adaptation step. It should be recognized that this involves solution-mapping from the physical grid to the Cartesian grid before every grid adaptation step.

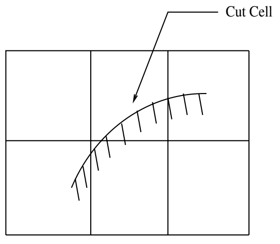


Fig. 4 Cut cell.

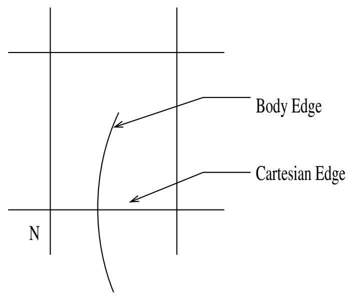


Fig. 6 Horizontal cut.

C. Identifying Body Cuts

The base grid wherein the point movement has not been effected is referred to as the Cartesian grid; the points and edges corresponding to this grid are referred to as Cartesian points and Cartesian edges, respectively. If the body cuts a horizontal Cartesian edge or a vertical Cartesian edge, it is called a horizontal cut or vertical cut, respectively. Figures 6 and 7 depict the horizontal and vertical cuts, respectively.

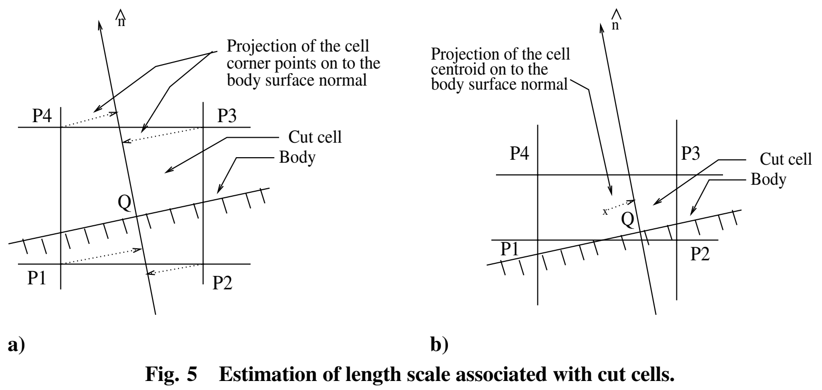

$\hat{n}$
Projection of the cell corner points on to the body surface normal
P4
P3
Cut cell
Body
Q
P1
P2
a)
$\hat{n}$
Projection of the cell centroid on to the body surface normal
P4
P3
Cut cell
Body
x
Q
P1
P2
b)

Fig. 5 Estimation of length scale associated with cut cells.

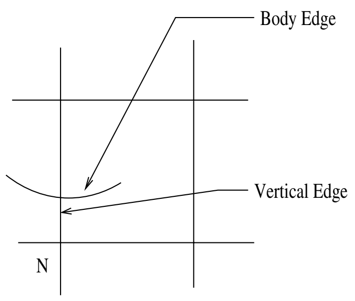


Fig. 7 Vertical cut.

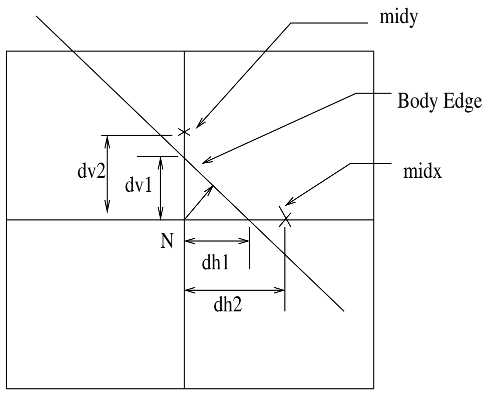


Fig. 8 Condition for diagonal movement.

D. Moving Cartesian Points onto Body Edges

Unstructured data are used for moving the Cartesian points. Hence, the information about Cartesian edges passing through a Cartesian point is defined explicitly and stored. A Cartesian point can undergo the following types of movements in a hierarchical order: 1) diagonal movement, 2) horizontal movement, and 3) vertical movement.

1. Diagonal Movement

A Cartesian point is considered as a candidate for diagonal movement if 1) a horizontal Cartesian edge and a vertical Cartesian edge passing through that Cartesian point are intersected by the body and 2) the body cut point is within 50% of the Cartesian edge length. The condition for the diagonal movement is illustrated in Fig. 8. From this figure, it can be said that point N will be moved diagonally if $dh1 < dh2$ and $dv1 < dv2$. The Cartesian point is moved onto the midpoint of the diagonal body cut, which may not be a straight line. This midpoint is identified by using the bisection method.

2. Horizontal Movement

When the body cuts a horizontal Cartesian edge, then the Cartesian point closest to the body cut is moved horizontally onto the body, as shown in Fig. 9. A Cartesian point can also undergo multiple movements. For example, close to the convex corners, the same Cartesian point can become a candidate for horizontal movement with respect to two body cuts. These possibilities are shown in Fig. 10. Under such a circumstance, a special point (called a dual point) is created with two sets of coordinates assigned to it.

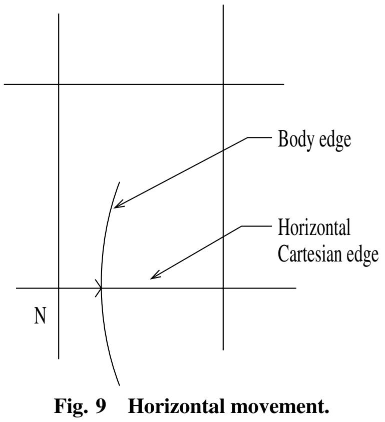


Fig. 9 Horizontal movement.

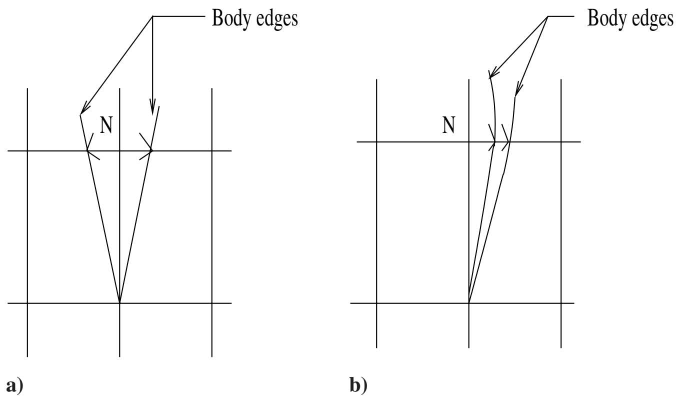


Fig. 10 Dual point for horizontal movement.

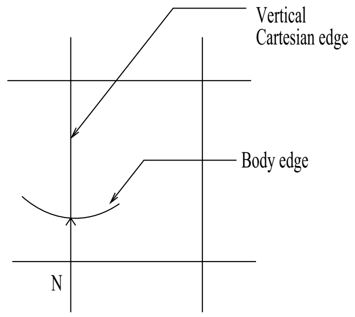


Fig. 11 Vertical movement.

3. Vertical Movement

Finally, we allow the points for vertical movement. When the body cuts a vertical Cartesian edge, then the Cartesian point closest to the body cut is moved vertically onto the body, as shown in Fig. 11. Similar to the previous case, close to the convex corners, the same Cartesian point can become a candidate for vertical movement with respect to two body cuts, as shown in Fig. 12. This special point is also called a dual point and two sets of coordinates are assigned to it.

E. Identification of the Location of Cartesian Points

In any Cartesian-grid algorithm, it is important to distinguish points that fall in the computational domain from the nonphysical points falling within the body contour. To determine such points, the ray-casting approach [35] is employed. This approach is depicted in Figs. 13 and 14. A ray is emanated from Cartesian point p in a particular direction. The number of intersections of the ray with the contour of the body indicates whether point p lies inside the body or

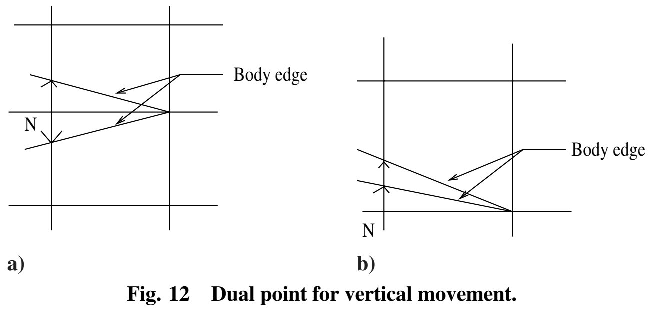


Fig. 12 Dual point for vertical movement.

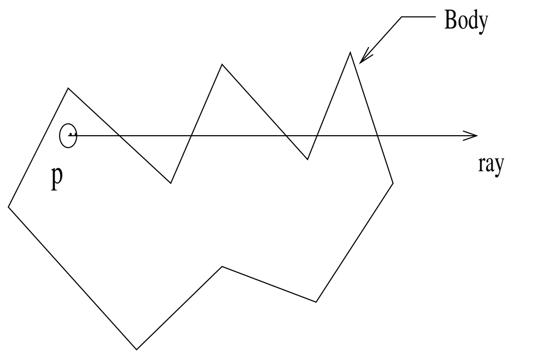


Fig. 13 Cartesian point p lies inside the body (five intersections).

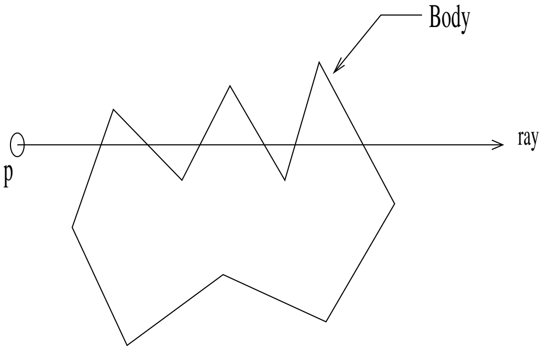


Fig. 14 Cartesian point p lies outside the body (six intersections).

not. If the number of cuts is even, then the point lies outside and if it is odd, then the point lies inside the body.

F. Data Collection

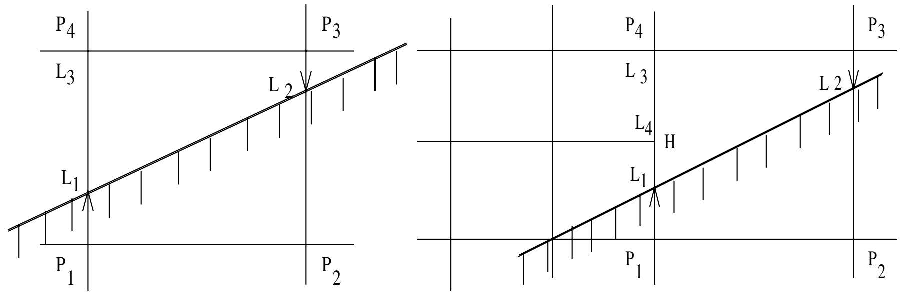
Here, the grid data are collected in a format suitable for a flow solver based on unstructured data and further post processing. These include coordinates of nodes, nodes constituting a cell, nodes constituting a face, and left and right cells of a given face.

1. *Node Data Collection*

The node data are collected by looping over the Cartesian points. A point is ignored if it lies inside the body. Otherwise, a new node number and boundary code are attributed to the point and coordinates are collected. For a dual point, an additional number with boundary code and coordinates is stored.

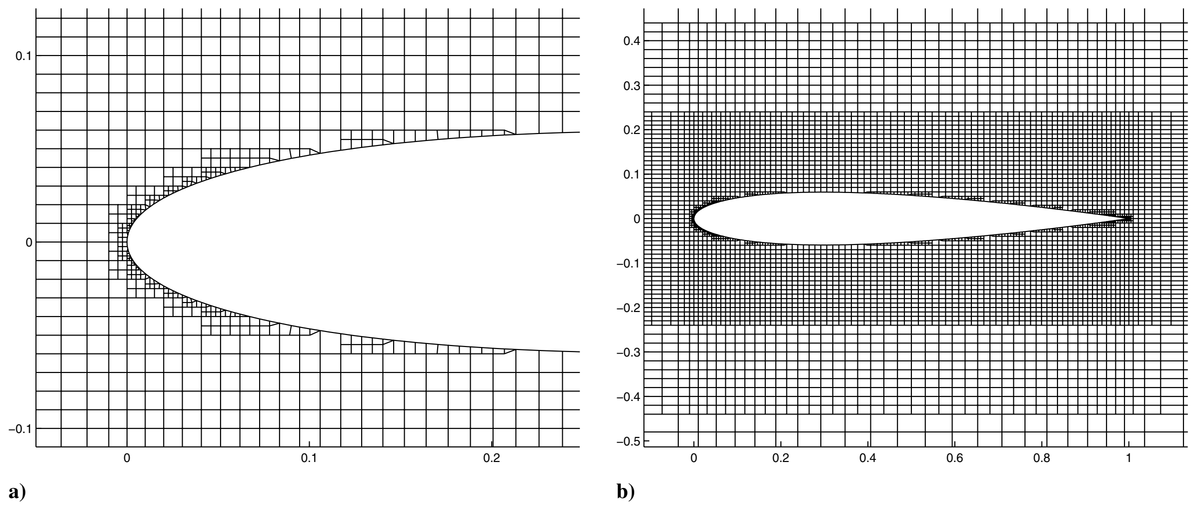
2. *Cell and Face Data Collection*

The cell and face data are collected simultaneously by looping over the Cartesian cells. It is noted that the maximum level difference between two cells sharing a face cannot be more than one for the background Cartesian mesh. Typical body cut cells resulting from such a procedure are presented in Fig. 15. For a body-cut cell, the nodes constituting the cell (including the hanging nodes) are identified along with their new locations after effecting point movement. For the cell, the nodes not falling inside the body (i.e., physical nodes) are ordered for contiguity. For example, in Fig. 15a, P_1, P_2, P_3, and P_4 are the points constituting a Cartesian cell without hanging node, whereas L_1, L_2, and L_3 are the points forming the new cell in the computational domain after effecting point movement. This would result in a new cell with L_1, L_2, and L_3 as its nodes. Similarly, in Fig. 15b, P_1, P_2, P_3, and P_4 are the corner points and H is



a) b)

Fig. 15 Locally renumbering the cell points.



a) b)

Fig. 16 Grid 1-0 a) zoomed view near leading edge and b) initial grid.

Table 1 Grid details of different geometries

Grid	Configuration	N_p	N_c	N_e	Number of body points	Figure no.
Grid 1-0	NACA 0012	9291	8664	17955	404	16a and 16b
Grid 1-1	NACA 0012	16,855	15,688	32,543	752	-
Grid 1-2	NACA 0012	23,617	21,964	45,581	991	19a
Grid 2	NLR 7301	10805	10,089	20,894	465	22a and 22b
Grid 3	NACA 0012	9038	8427	17,465	382	25a–25c
Grid 4	NACA 0012	3798	3509	7307	108	27
Grid 5	RAE 2822	34,309	32,718	67,027	1295	29

Table 2 Flow conditions on different geometries

Case	Configuration	Grid	Re	M_∞	α	Standard/exp result	Figure no.
Case 1	NACA 0012	Grid 1-0	-	0.63	2^0	Fine triangulated grid (HIFUN-2D)	17 and 20
Case 2	NACA 0012	Grid 1-0, Grid 1-1, Grid 1-2	-	0.85	1^0	Fine triangulated grid (HIFUN-2D)	18, 19b, and 21
Case 3	NLR 7301	Grid 2	-	0.185	6^0	Van den Berg [39]	23
Case 4	NLR 7301	Grid 2	-	0.185	13.1^0	Van den Berg [39]	24
Case 5	NACA 0012	Grid 3	5000	0.5	0^0	Venkatakrishnan [40]	26a and 26b
Case 6	NACA 0012	Grid 4	500	0.85	0^0	Fortunato [41]	28a and 28b
Case 7	RAE 2822	Grid 5	5.7×10^6	0.676	1.92^0	Experiment [42]	30a and 30b
Case 8	RAE 2822	Grid 5	6.5×10^6	0.73	2.79^0	Experiment [42]	31a and 31b

Table 3 Grid details for reference data

Reference grid	Configuration	N_p	N_c	N_e	Number of body points
Reference grid 1	NACA 0012	5864	11,288	17,152	400

the hanging node between points P_1 and P_4 associated with a Cartesian cell. Here, a new cell is formed in the computational domain with the physical points L_1, L_2, L_3, and L_4. The cell is given a new number.

To collect the face data, the edges forming the background Cartesian cell are explicitly defined and stored. We associate a flag with each of these Cartesian edges. To begin with, all the Cartesian edges are given flag 0. The face data are collected by looping over the edges forming the Cartesian cell under consideration. Two consecutive points of newly formed cell are compared with the points constituting the edges of the Cartesian cell. The Cartesian edge for which the points match with those consecutive points become a face in the computational domain. The consecutive points constitute a newly formed face. The newly formed cell becomes one of the cells sharing the face. The flag of the Cartesian edge is set to one to avoid duplication. If a Cartesian edge with flag 1 is encountered, then the Cartesian cell under consideration becomes the other neighbor cell sharing that face. If two consecutive points of a newly formed cell do not match with points constituting any of the edges forming the Cartesian cell under consideration, a new face is formed with those points. This face is a body face, and one such face with points L_1 and L_2 is shown in Fig. 15b. In Fig. 16, the zoomed view of the Cartesian grid generated by this strategy for a NACA 0012 airfoil is shown.

IV. Numerical Results

In this section, various flow simulations for different geometries are presented. The results have been generated with a cell-centered, finite volume, two-dimensional high-resolution flow solver on unstructured meshes (HIFUN-2D). To compute inviscid fluxes, van Leer [36] and Roe [37] flux formulas are used for inviscid and viscous flow computations, respectively. Convergence acceleration is attained by using the symmetric Gauss–Seidel (SGS) implicit relaxation procedure [28,38]. The different grids and flow conditions used in validating the algorithm are presented in Tables 1 and 2, respectively. The outer boundary is located around 10 chords away from the airfoil. The grid used for generating reference data for inviscid flow past a NACA 0012 airfoil is summarized in Table 3. In the tables, N_p, N_c, and N_e stand for the number of points, number of cells, and number of faces, respectively.

A. Inviscid Flow Computations

Two standard test cases (cases 1 and 2 in Table 2) corresponding to inviscid flow past a NACA 0012 airfoil have been considered for validation. A Cartesian mesh is easily generated around this geometry. Figures 16a and 16b show the zoomed view at the leading edge and the close-up view of grid 1-0, respectively. Mach contours obtained for these test cases in a zero-level grid are shown in Figs. 17

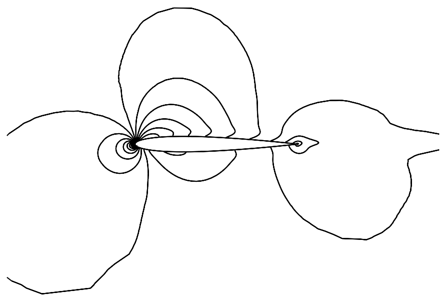


Fig. 17 Case 1 Mach contours over a NACA 0012 airfoil using grid 1-0; $M_\infty = 0.63$ and $\alpha = 2^0$.

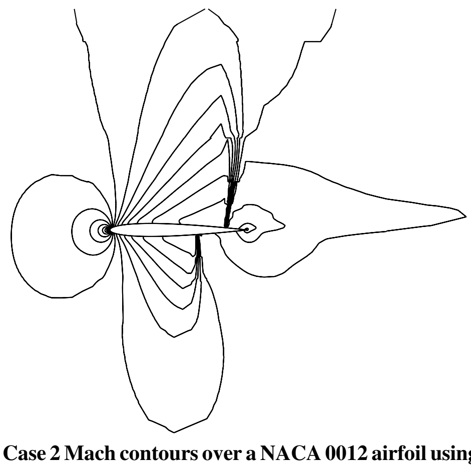
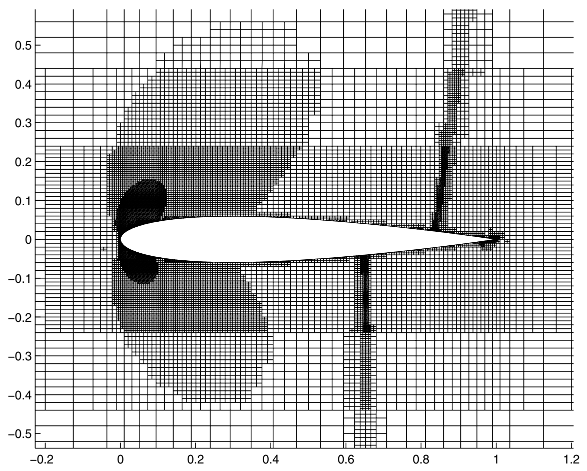


Fig. 18 Case 2 Mach contours over a NACA 0012 airfoil using grid 1-0; $M_\infty = 0.85$ and $\alpha = 1^0$.


a)

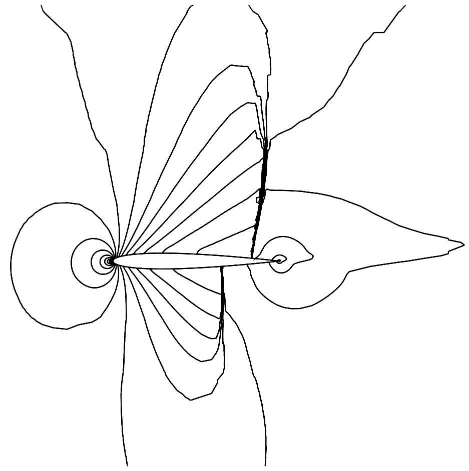

b)

Fig. 19 Grid 1-2 a) level-2 grid and b) Mach contours; $M_\infty = 0.85$ and $\alpha = 1^0$.

and 18. Because of the poor grid resolution in the shock region for the transonic test case, the shock is smeared. Therefore, two levels of solution-based adaptation are considered. The final grid after two levels of adaptation is shown in Fig. 19a. Figure 19b shows the Mach contours obtained on a level-2 grid. The computed C_p distribution for test case 1 is compared with reference data in Fig. 20. In Fig. 21, C_p distributions of level 0, level 1 and level 2 are shown with reference data. It is noted that the reference data are obtained by using the HIFUN-2D solver on a fine triangular grid (reference grid 1 in Table 3). The computed results compare well with the reference data, and the upper and lower surface shocks are captured accurately. Table 4 presents a comparison of the C_L and C_D predicted by the present method and the benchmark AGARD [43] data. The computed results compare well with the standard results.

The computations have also been performed for inviscid flow past an NLR 7301 airfoil with slotted flap (test cases 3 and 4 in Table 2). The close-up view of the airfoil and zoomed view in the slotted region between the main airfoil and flap are shown in Figs. 22a and 22b, respectively. The pressure distributions obtained for these test cases are shown in Figs. 23 and 24, along with the standard experimental results [39].

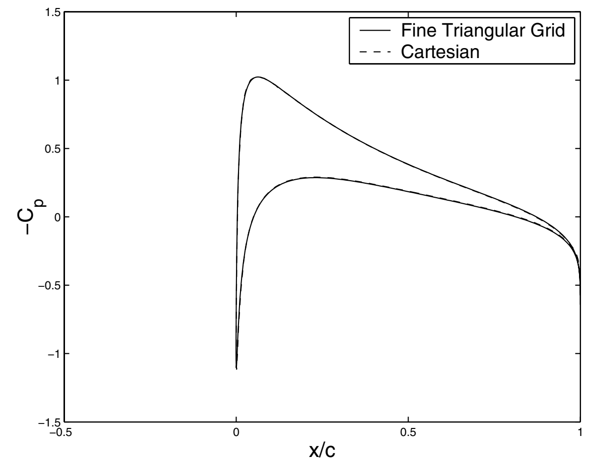


Fig. 20 Case 1 C_p distribution for a NACA 0012 airfoil ($M_\infty = 0.63$ and $\alpha = 2^0$).

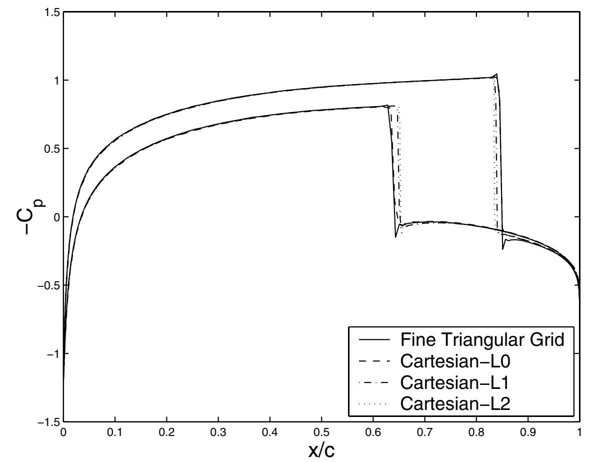


Fig. 21 Case 2 C_p distribution on different levels of grid over a NACA 0012 airfoil ($M_\infty = 0.85$ and $\alpha = 1^0$).

B. Laminar Flow Computations

Two test cases (cases 5 and 6) involving laminar flow past a NACA 0012 airfoil have been considered. For the validation of case 5, grid 3, involving a maximum aspect ratio of 30, has been used. Zoomed views of the grid at the leading-edge, midchord, and trailing-edge locations are shown in Figs. 25a–25c, respectively. The computed pressure and skin friction coefficients are compared with the standard results in Figs. 26a and 26b. The aerodynamic coefficients and separation point for the preceding case are compared with the standard data [40] in Table 5. The flow separates at 81.3% from the leading edge and this is in good agreement with Venkatakrishnan's [40] result. The second test case (case 6) involves transonic laminar flow past a NACA 0012 airfoil. The zoomed view of grid 4 used in this computation is shown in Fig. 27. Interestingly, the present procedure involving a flow-based adaptation strategy results in a coarser grid for this low Reynolds number case, compared with the previous grid 3. The maximum aspect ratio in this case is around 10 for the body cells. It is important to note that the grid adaptation is performed at a preflow solver stage itself and no solution-based adaptation is effected. Computed pressure and skin friction coefficients distributions compare well with the standard data [41], as shown in Figs. 28a and 28b.

The excellent comparison of the results obtained for the laminar flow cases with the standard cases clearly establishes the efficacy of

Table 4 Comparison of computed aerodynamic coefficients with standard data [43] for cases 1 and 2

Cases	AGARD [43]		Present Method	
	C_L	C_D	C_L	C_D
Case 1	0.3335	-	0.3047	-
Case 2	0.3790	0.0576	0.3313	0.05484

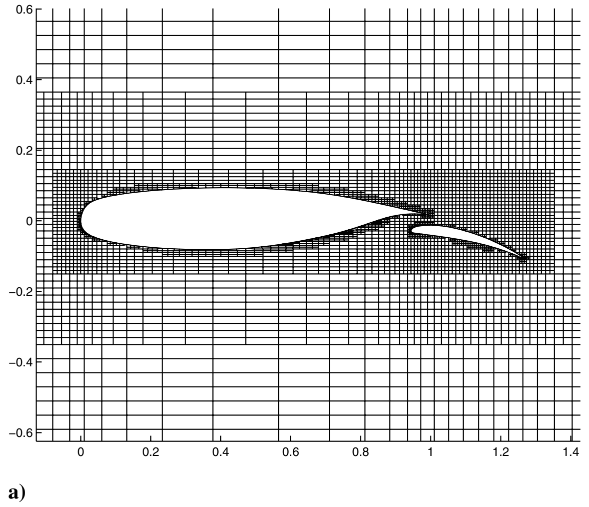
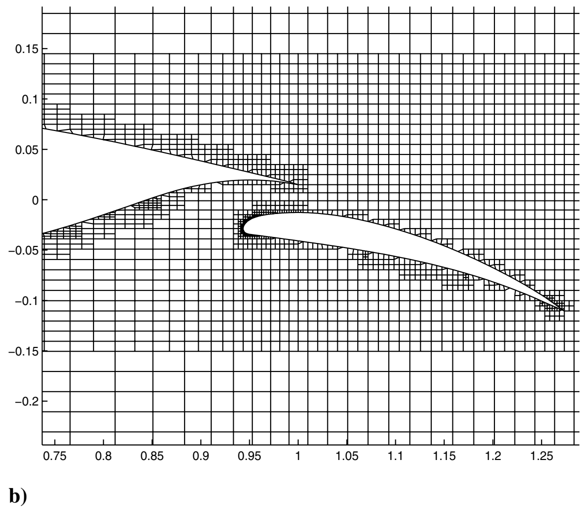



Fig. 22 Grid 2 a) around an NLR 7301 airfoil and flap and b) zoomed view in the slotted region.

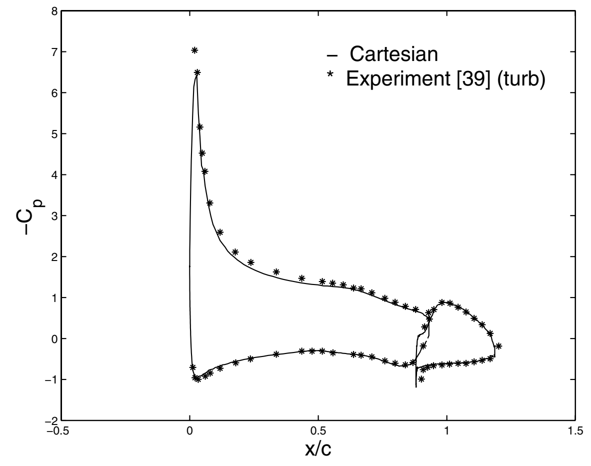


Fig. 23 Case 3 C_p distribution over an NLR 7301 airfoil and flap; $M_\infty = 0.185$ and $\alpha = 6^0$.

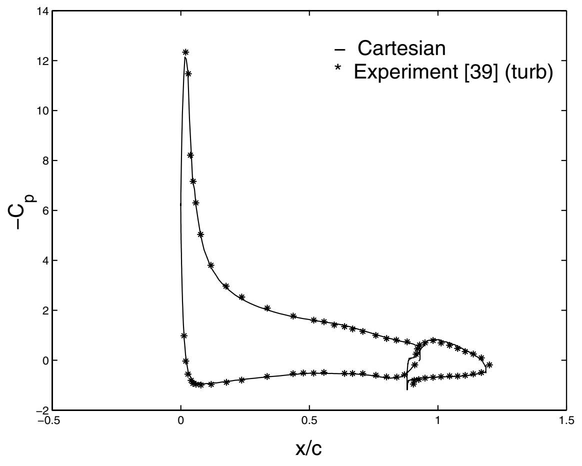


Fig. 24 Case 4 C_p distribution over an NLR 7301 airfoil and flap; $M_\infty = 0.185$ and $\alpha = 13.1^0$.

the present grid-generation procedure for computing such flows. In our view, it has been possible to generate these solutions because of the smoothness of the grids generated using the grid-stitching procedure. Also, it is expected that the requirement on grid quality for turbulent flow calculations using large-eddy simulation (LES) tools is similar to the one presented for the laminar flows, although a much finer grid may be required. In such a case, employing the present methodology, it is possible to simulate laminar separated flows at moderate Reynolds numbers, encountered in variety of aerospace applications such as mini-aerial vehicles and flapping wings. In our view, the real challenge in extending the utility of the Cartesian-grid algorithms lies in computing turbulent flows by solving Reynolds-averaged Navier–Stokes (RANS) equations. This invariably involves higher aspect ratio cells on the walls. The extension of the present methodology for RANS calculation is presented in the next section.

C. Turbulent Flow Computations

Two standard turbulent flow test cases involving a RAE 2822 supercritical airfoil are considered for validating the algorithm. Turbulent flow computations have been made using the Baldwin and Lomax turbulence model [44] and are compared with standard experimental data [42]. The grid used for these computations (grid 5) is presented in Fig. 29. For this class of computations, a structured mesh calculation would typically involve cells with an aspect ratio of

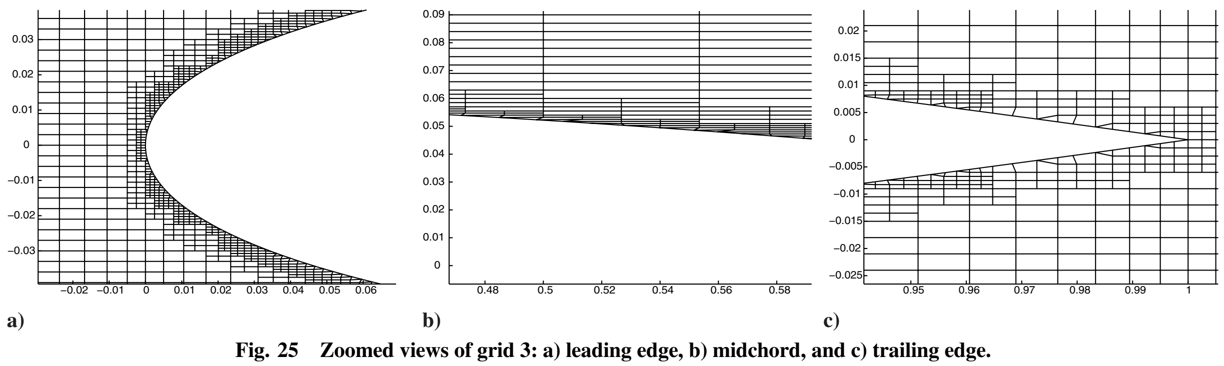


Fig. 25 Zoomed views of grid 3: a) leading edge, b) midchord, and c) trailing edge.

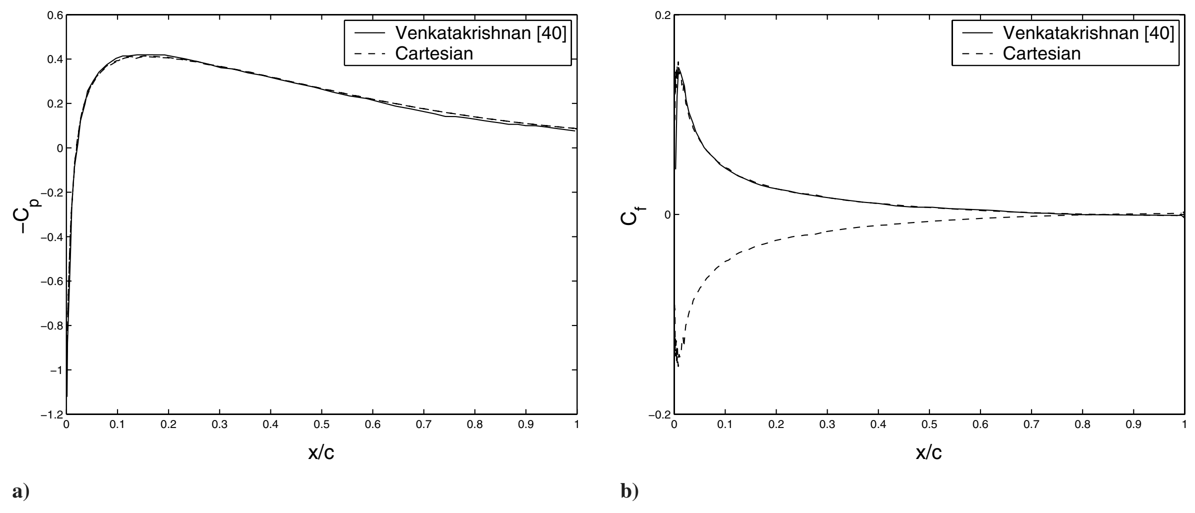


Fig. 26 Case 5 a) C_p distribution and b) C_f; $R_e = 5000$, $M_\infty = 0.5$, and $\alpha = 0^0$.

the order of 1000. This can be easily seen by the fact that the first point in the boundary layer is placed at a distance of 10^{-5} for an airfoil of unit chord, with approximately 100 stations in the horizontal direction. As indicated before, any turbulent flow calculation involving Cartesian mesh should necessarily mimic this feature of the structured mesh for the resulting grid to have a reasonable number of cells. In our attempt, we have generated the zero-level grid by invoking about 80 horizontal stations along the x direction and holding the y-grid spacing uniform in the body block. Isotropic refinement of the resulting zero-level grid is effected by using both the geometric and flow-based refinement criteria, as indicated in Sec. III.B. This results in a grid with 1295 points on the wall and a maximum aspect ratio of about 100 for the wall cell. It is important to note that anisotropic refinement would have resulted in a lesser number of wall cells. It is easy to see that use of unit aspect ratio cells close to the wall would have resulted in as many as 10,000 wall cells for the same kind of grid resolution. This observation has serious implications in 3-D turbulent flow computations. It is also worthwhile to compare the present grid with a body-fitted grid; the same kind of grid resolution could have been attained with just about 200 wall cells. It also has an advantage of resolving the boundary layer in a direction normal to the wall. In contrast, in the present grid, the normal distance of the centroids of the wall cells from the wall resolves the boundary layer, resulting in a greater number of wall cells. Nevertheless, this may be considered as a best compromise for

Table 5 Comparison of computed aerodynamic coefficients with standard data [40] for case 5

	C_{D_p}	C_{D_f}	C_D	Separation
Present method	0.02746	0.03411	0.06157	81.3%
Venkatakrishnan [40]	0.02297	0.03247	0.05544	81.0%

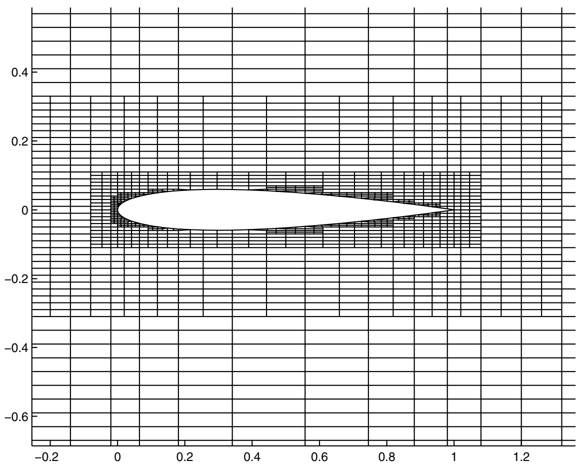


Fig. 27 Grid 4 over a NACA 0012 airfoil for viscous flow.

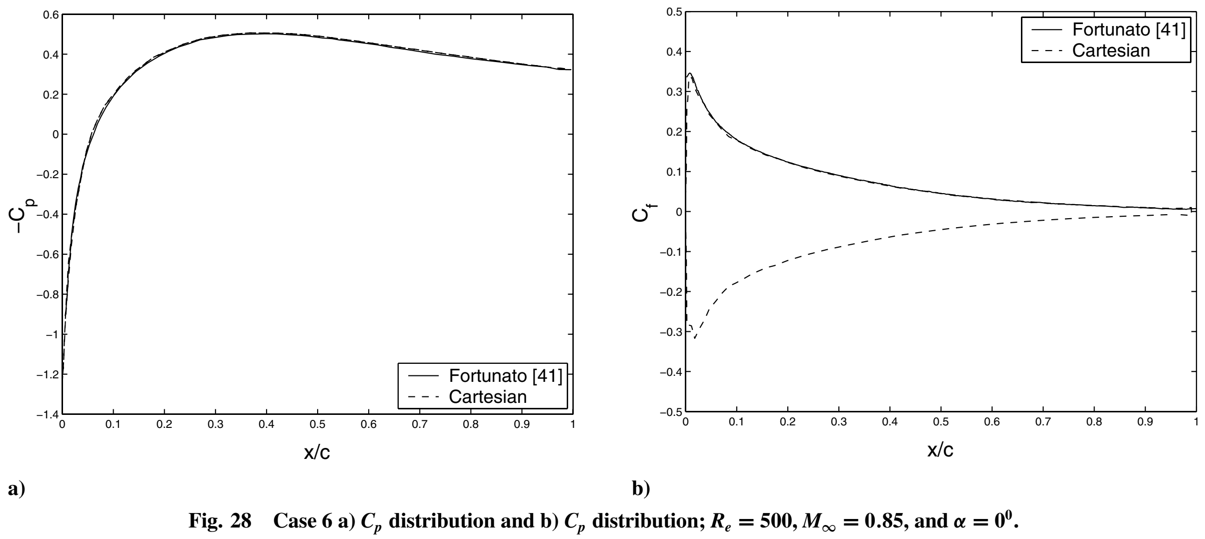


Fig. 28 Case 6 a) C_p distribution and b) C_p distribution; $R_e = 500$, $M_\infty = 0.85$, and $\alpha = 0^0$.

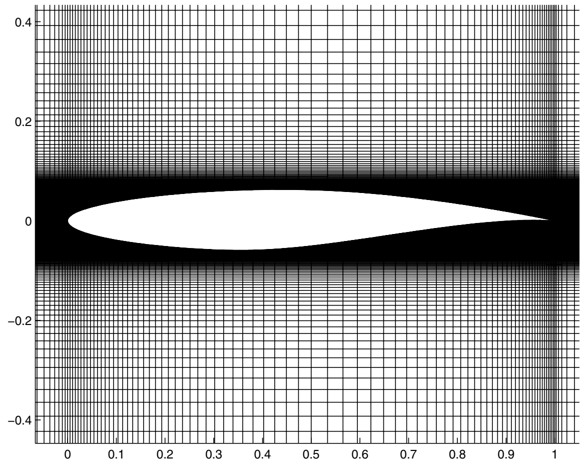


Fig. 29 Grid 5 over RAE 2822 airfoil for turbulent flow.

resolving the flow past streamlined bodies using Cartesian grids. It should also be remarked that the total number of cells resulting from the present procedure is still comparable with that of a body-fitted grid, simply because the grid fineness close to the body is not allowed to propagate to the far field.

Average y^+ values for the body cells are found to be 10 and 15 for subsonic and transonic test cases, respectively. Wall pressure distribution and Mach contours are shown in Figs. 30a and 30b, respectively, for the subsonic turbulent test case (case 7). The computed wall pressure distribution is in good agreement with the experimental data [42]. Test case 8 corresponds to the transonic turbulent flow. Referring to Fig. 31a, the computed pressure distribution is closely matching with the experimental data [42] and is devoid of oscillations. Mach contours shown in Fig. 31b indicate the formation of a shock on the upper surface of the airfoil. Table 6 presents the comparison of predicted lift and drag coefficients against the experimental data [42] for the transonic test case. The computed aerodynamic coefficients agree well with the experimental values. The good agreement of the lift coefficient does not come as a surprise, because the pressure distribution has been predicted accurately. On the other hand, the agreement of the drag coefficient is good, in spite of an oscillatory skin friction distribution. This could be because the major component of the total drag coefficient comes from the wave drag (almost 70%). The oscillatory skin friction distribution is attributed to a nonpositive viscous flux discretization procedure. In spite of this deficiency, even the fact that the pressure distribution has been captured accurately for turbulent flows with Cartesian grids clearly establishes the efficacy of the grid-stitching strategy in generating smooth grids for turbulent flows. Nevertheless, any further development in Cartesian-mesh calculations will critically

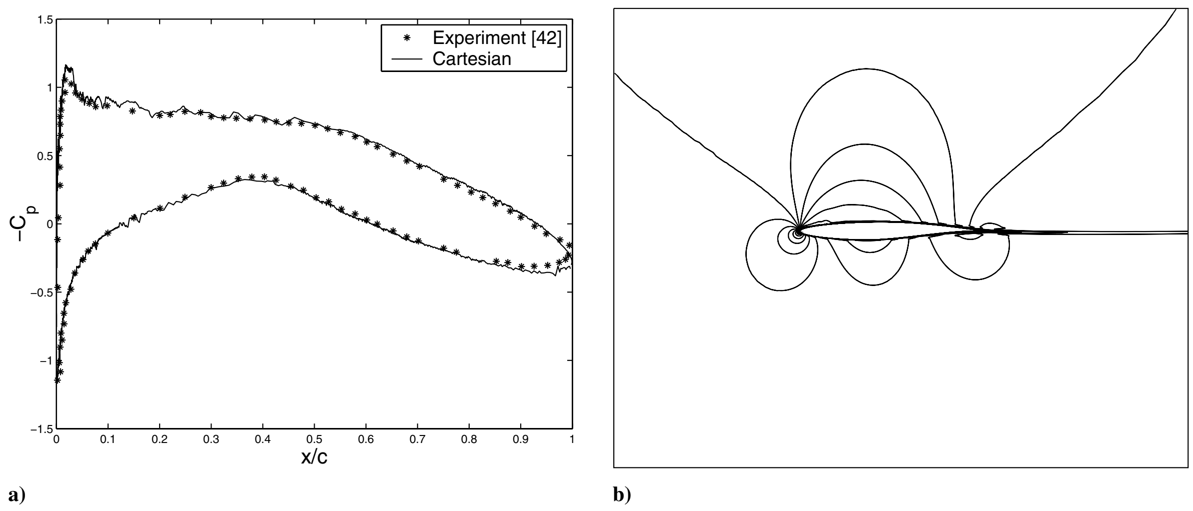


Fig. 30 Case 7 a) C_p distribution and b) Mach contours; $R_e = 5.7 \times 10^6$, $M_\infty = 0.676$, and $\alpha = 1.92^0$.

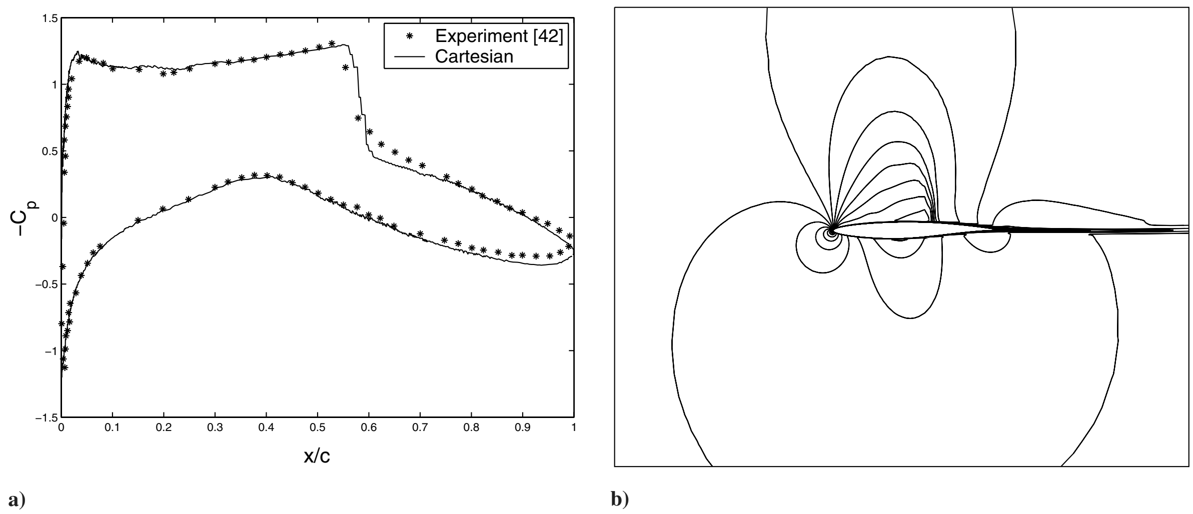


Fig. 31 Case 8 a) C_p distribution and b) Mach contours; $R_e = 6.5 \times 10^6$, $M_\infty = 0.73$, and $\alpha = 2.79^0$.

Table 6 Comparison of computed aerodynamic coefficients with standard data [42] for case 8

Case	Experiment [42]		Present method	
	C_L	C_D	C_L	C_D
Case 8	0.8030	0.0168	0.8321	0.01743

depend on evolving positive schemes for viscous flux discretization [30].

V. Conclusions

A novel grid-stitching algorithm has been developed for generating Cartesian-like grids. The procedure involves identification of the streamline direction and stretching the Cartesian grid suitably. This may be considered as an aspect that brings in human intervention, thereby reducing the possibility of automated grid generation. In our view, because of the inseparability of the flow physics and the computation, this is the minimum intervention essential for any meaningful viscous computation. It is important to note that the user intervention required for the present strategy is less intense than that required for Cartesian-mesh-based strategies employing structured grids in the viscous region [4,11]. In the present strategy, use of tree data could be an important component of the background Cartesian-mesh generation. This has not been exploited in this work. On the contrary, the grid data associated with the flow solver are recommended to be handled using unstructured data. The excellent laminar solution and an acceptable wall pressure distribution for turbulent flow presented in this paper clearly establish the efficacy of the proposed Cartesian-grid-generation procedure. Therefore, in our view, the grid-stitching algorithm provides an effective guideline for the future Cartesian-mesh-generation strategies. But we should also be reminded that any further success in all-Cartesian-grid calculations would critically depend on evolving a positive viscous flux discretization procedure.

References

[1] Arpaci, V. S., *Conduction Heat Transfer*, Addison–Wesley, London, 1966, pp. 492–493.

[2] Coirier, W. J., "An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations," Ph.D. Thesis, Department of Aerospace Engineering, Univ. of Michigan, Ann Arbor, MI, Jan. 1994.

[3] Zeeuw, D. D., and Powell, K. G., "An Adaptively Refined Cartesian Mesh Solver for the Euler Equations," *Journal of Computational Physics*, Vol. 104, No. 1, 1993, pp. 56–68.

[4] Philippe, G., "An Implicit Upwind Finite Volume Method for Compressible Turbulent Flows on Unstructured Meshes," Ph.D. Thesis, Univ. of Liège, Liège, Wallonia, Belgium, Apr. 1999.

[5] Clarke, D. K., Salas, M. D., and Hassan, H. A., "Euler Calculations for Multi Element Airfoils Using Cartesian Grids," *AIAA Journal*, Vol. 24, No. 3, 1986, pp. 353–358.

[6] Coirier, W. J., and Powell, K. G., "Solution-Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows," *AIAA Journal*, Vol. 34, No. 5, 1996, pp. 938–945.

[7] Wang, Z. J., and Chen, R. F., "Anisotropic Cartesian Grid Method for Viscous Turbulent Flow," AIAA Paper 2000-0395, 2000.

[8] Wang, Z. J., "A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equations," *Computers and Fluids*, Vol. 27, No. 4, 1998, pp. 529–549.

[9] Frymier, P. D., Jr., Hassan, H. A., and Salas, M. D., "Navier-Stokes Calculations Using Cartesian Grids, 1: Laminar Flows," *AIAA Journal*, Vol. 26, No. 10, 1988, pp. 1181–1188.

[10] Fenno, C. C., Jr., Newman, P. A., and Hassan, H. A., "Unsteady Viscous-Inviscid Interaction Procedures for Transonic Airfoils Using Cartesian Grids," *Journal of Aircraft*, Vol. 26, No. 8, 1989, pp. 723–730.

[11] Delanaye, Michel, Aftosmis, M. J., Berger, M. J., Liu, Y., and Pulliam, T. H., "Automatic Hybrid-Cartesian Grid Generation for High-Reynolds Number Flows Around Complex Geometries," AIAA Paper 99-0777, Jan. 1999.

[12] Smith, R. J., and Leschziner, M. A., "Automatic Grid Generation for Complex Geometries," *The Aeronautical Journal*, Vol. 100, No. 991, Jan. 1996, pp. 7–14.

[13] Mondal, P., "Cartesian-Like Grids Using a Novel Grid-Stitching Algorithm for Viscous Flow Computations," M.S. Thesis, Dept. of Aerospace Engineering, Indian Inst. of Science, Bangalore, India, Apr. 2005.

[14] Mondal, P., Munikrishna, N., and Balakrishnan, N., "Computation of Viscous Compressible Flows Employing Cartesian-Like Grids," *Proceedings of the 18th National Convention of Aerospace Engineers*, IIT Kharagpur, India, 2004, pp. 140–146.

[15] Mondal, P., Munikrishna, N., and Balakrishnan, N., "Viscous Flow Computations Using Cartesian-Like Grids," *Proceedings of the 8th Annual AeSI CFD Symposium*, CP25, National Aerospace Labs., Bangalore, India, 2005..

[16] Morinishi, K., "A Gridless Type Solver for Parallel Simulation of Compressible Flow," *Parallel Computational Fluid Dynamics, Development and Applications of Parallel Technology*, Elsevier Science, New York, 1999.

[17] Luo, H., Baum, J. D., and Lohner, R., "A Hybrid Cartesian Grid and Gridless Method for Compressible Flows," AIAA Paper 2005-0492, Jan. 2005.

[18] Kirshman, D. J., and Liu, F., "A Gridless Boundary Condition Method for the Solution of the Euler Equations on Embedded Cartesian Meshes with Multigrid," *Journal of Computational Physics*, Vol. 201, No. 1, 2004, pp. 119–147.

[19] Munikrishna, N., Karthikeyan, N., and Balakrishnan, N., "A Meshless Solver for Computing Viscous Flows on Cartesian-Like Grids," *Computational Fluid Dynamics 2006* (to be published).

[20] Ninawe, A., Munikrishna, N., and Balakrishnan, N., "Viscous Flow Computations Using A Meshless Solver, LSFD-U," *Computational Fluid Dynamics 2004*, edited by C. Groth and D. W. Zingg, Springer–Verlag, Berlin, 2004, pp. 509–514.

[21] Ye, T., Mittal, R., Udaykumar, H. S., and Shyy, W., "An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries," *Journal of Computational Physics*, Vol. 156, No. 2, 1999, pp. 209–240.

[22] Roma, A. M., Peskin, C. S., and Berger, M. J., "An Adaptive Version of the Immersed Boundary Method," *Journal of Computational Physics*, Vol. 153, No. 2, 1999, pp. 509–534.

[23] Tseng, Y.-H. and Ferziger, J. H., "A Ghost-Cell Immersed Boundary Method for Flow in Complex Geometry," *Journal of Computational Physics*, Vol. 192, No. 2, 2003, pp. 593–623.

[24] Majumdar, S., Iaccarino, G., and Durbin, P., "Navier Stokes Solvers Using Adaptive Boundary-Non-Conforming Cartesian Grids," *Proceedings of the 6th Annual AeSI CFD Symposium*, CP12, National Aerospace Labs., Bangalore, India, 2003.

[25] Iaccarino, G., and Verzicco, R., "Immersed Boundary Technique for Turbulent Flow Simulations," *Applied Mechanics Reviews*, Vol. 56, No. 3, 2003, pp. 331–347.

[26] De Tullio, M. D., De Palma, P., Iaccarino, G., Pascazio, G., and Napolitano, M., "Immersed Boundary Technique for Compressible Flow Simulations on Semi-Structured Grids," *Computational Fluid Dynamics 2006* (to be published).

[27] Sachdev, J. S., and Groth, C. P. T., "A Mesh Adjustment Scheme for Embedded Boundaries," *Computational Fluid Dynamics 2004*, edited by C. Groth and D. W. Zingg, Springer, Berlin, 2004, pp. 109–114.

[28] Shende, N. and Balakrishnan, N., "New Migratory Memory Algorithm for Implicit Finite Volume Solvers," *AIAA Journal*, Vol. 42, No. 9, Sept. 2004, pp. 1863–1870.

[29] Jawahar, P., and Kamath, H., "A High-Resolution Procedure for Euler and Navier-Stokes Computations on Unstructured Grids," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 165–203.

[30] Munikrishna, N., and Balakrishnan, N., "Computing Viscous Flows on Unstructured Meshes with Hanging Nodes," *Proceedings of the 7th Annual AeSI CFD Symposium*, CP13, National Aerospace Labs., Bangalore, India, 2004.

[31] Holmes, D. G., and Connel, S. D., "Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids," AIAA Paper 89-1932-CP, 1989.

[32] Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol. 118, No. 1, 1995, pp. 120–130.

[33] Balakrishnan, N., and Fernandez, G., "Wall Boundary Conditions for Inviscid Compressible Flows on Unstructured Meshes," *International Journal for Numerical Methods in Fluids*, Vol. 28, No. 10, 1998, pp. 1481–1501.

[34] Hirsh, C., *Numerical Computation of Internal and External Flows*, Vol. 2, Wiley, New York, 1988, pp. 346–347.

[35] Milgram, M. S., "Does a Point Lie Inside a Polygon?," *Journal of Computational Physics*, Vol. 84, No. 1, 1989, pp. 134–144.

[36] Van Leer, B., "Flux Vector Splitting for Euler Equations," Inst. for Computer Applications in Science and Engineering, Rept. 82-30, NASA Langley Research Center, Hampton, VA, 1982.

[37] Roe, P. L., "Approximate Reimann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[38] Yoon, S., and Jameson, A., "Lower-Upper Symmetric-Gauss-Seidel Method for the Euler and Navier-Stokes Equations," *AIAA Journal*, Vol. 26, No. 9, Sept. 1988, pp. 1025–1026.

[39] Van den Berg, B., "Boundary Layer Measurements on a Two-Dimensional Wing with Flap," Dutch National Aerospace Laboratory (NLR), Rept. TR-79009 U, 1979.

[40] Venkatakrishnan, V., "Viscous Computations Using A Direct Solver," *Computers and Fluids*, Vol. 18, No. 2, 1990, pp. 191–204.

[41] Fortunato, B., and Magi, V., "An Implicit Lambda Method for 2-D Viscous Compressible Flows," *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, Springer–Verlag, Berlin, 1994, pp. 259–264.

[42] Cook, P. H., Mc Donald, M. A., Firmin, M. C. P., "Aerofoil RAE 2822-Pressure Distribution, and Boundary Layer, and Wake Measurement," AGARD AR-138, May 1979.

[43] Anon., "Inviscid Flow Field Methods," Fluid Dynamics Panel Working Group, AGARD Advisory Rept. 211, July 1985.

[44] Baldwin, B. S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AlAA Paper 78-257, 1978.