Quad-Tree Based Geometric-Adapted Cartesian Grid Generation

EMRE KARA1, AHMET İHSAN KUTLAR1, MEHMET HALUK AKSEL2
1Department of Mechanical Engineering
University of Gaziantep
27310 Gaziantep
TURKEY
2 Mechanical Engineering Department
Middle East Technical University
Çankaya, 06800 Ankara
TURKEY
emrekara@gantep.edu.tr   http://www.gantep.edu.tr

Abstract: - Cartesian grid generation method is a special class of unstructured techniques under the general grid generation methodology. In this paper, examples of the Cartesian grid are presented which were obtained by using quad-tree approach. Quad-tree data structure is utilized in order to connect the Cartesian cells to each other. The dynamic data structure stores connectivity information for each cell and can create or destroy cells repeatedly, anywhere in the flow domain as the programmer wills. Another benefit of Cartesian grid formulation is the requirement of minimal input to describe the geometry. In this study, the solution algorithm is implemented in FORTRAN programming language and the code employs Cartesian grid technique to model complex geometries. Two distinct test cases are presented at the end of the paper to clarify the use of the methodology.

Key-Words: - Cartesian grid generation, Geometry adaptation, Cut-cell method, Curvature adaptation.

1 Introduction

In computational fluid dynamics (CFD), a suitable grid is to be produced on the physical domain of the problem. Basically, there are three main grid generation techniques employed: structured, unstructured and Cartesian methods. Cartesian methods are simply a special class of unstructured methods. Historically, they were first proposed at the beginning of the 1970s as an alternative to structured and ordinary unstructured methods. The aim was to enhance automatic grid generation and facilitate solution adaptation. On the other hand, Cartesian methods involve complicated data structures requiring more efficient computers.

Peskin [1] reported simulations of the blood flow in the heart/mitral-valve system by using a two-dimensional very low Reynolds number flow. Three-dimensional heart flows caring the elastic nature of the boundaries were also considered successively by Peskin. In Peskin's formulation, the governing equations (incompressible Navier-Stokes equations) are solved on uniform Cartesian grids and the elastic fibers of the heart walls are immersed in the flow: Fluid and fibers exert time varying forces on one another. The obvious advantage of these methods over the conventional body-conformal approach is that irrespective of the geometric complexity of the immersed boundaries, the computational grid remains unchanged. Cartesian grid methods free the underlying structured computational grid from the task of adapting to the moving boundary, thus allowing large changes in the geometry due to boundary evolution.

An important advantage of Cartesian methods is that any kind of adaptation is very easy to implement. By means of solution adaptation, for instance, a finer grid can be obtained around a shock wave. Hence, very high accuracy levels can be obtained without increasing number of cells and the computational time required significantly. The governing equations are discretized on a Cartesian grid which does not conform to the immersed boundaries. This greatly simplifies grid generation task and also retains the relative simplicity of the governing equations in Cartesian coordinates. In addition, this method also has a significant advantage over the conventional body-fitted approach in simulating flows such as with moving boundaries, complicated shapes, or involving topological changes.

Structured and ordinary unstructured methods require user interference to some extent. Cartesian methods, on the other hand, permit automatic grid generation. Therefore, in order to handle problems and reduce the user intervention through the grid generation process, both the grid generation and adaptation processes are automated. In this respect, what remains to the user as a task is the proper definition of the problem.

2 Cartesian Grid Generation

The code utilizes Cartesian grid techniques to model complex geometries, regardless of body shape. An important advantage of Cartesian methods is that any kind of adaptation can easily be implemented. Hence, relatively high accuracy levels can be obtained with a relatively low number of cells. This saves computational time and memory allocation required significantly.

The multiplication of the maximum length by the user-defined size factor defines the domain size. This domain size is actually the length of the each edge of the mother cell. A uniform grid for the two dimensional Cartesian geometry is obtained by dividing squares successively starting from this mother cell to the smallest child cells so that these child cells have an appropriate cell size near the solid body. This gives a multi-grid system with fine cells near the solid body, where higher resolution is needed for high gradients. The one-level rule is applied between each level of the successive divisions [2]. The mother-children connectivity is provided by the quad-tree approach.

2.1 Quad-tree Data Structure

In the literature, there are various methods used for fluid flow problems to identify connectivity information such as two dimensional arrays, linked list, binary tree, quad-tree data structures [2].

The Cartesian grid is obtained by using the quad-tree approach in two dimensions. In the quad-tree approach, a square containing the whole domain is divided into its four quadrants to form the grid in a tree structure. Hence, the procedure followed is much simpler than the methods for generating triangular unstructured meshes. In this paper, the basic data structure used is a cell-based quad-tree structure; “mother” cells are refined by division into four ”children” cells which is done in coarsening part of the FORTRAN program. The information is treated in allocatable arrays using pointers.

The grid typically begins with a single mother cell, and grows by a recursive subdivision of each cell into its four children. Each child is geometrically contained within the boundaries of the parent cell, and is located logically below the parent cell in the tree. Figure 1 illustrates this concept for the subdivision of a single cell, child 4, isotropically into 4 cells, cells D, E, B and C.

Arbitrary subdivisions of the cells are allowed during the process, only requiring that the newly created cells be non-overlapping polygons that fill the space occupied by the mother cell. To take advantage of the smooth grid that can be achieved, the root cell is taken to be a square Cartesian cell of having unit aspect ratio, and cell division is obtained by isotropically splitting each cell into four, equal area children. Since N-sided cut cells are obtained by “cutting” them out of their background, Cartesian cell, they are always logically locatable in the tree [2].

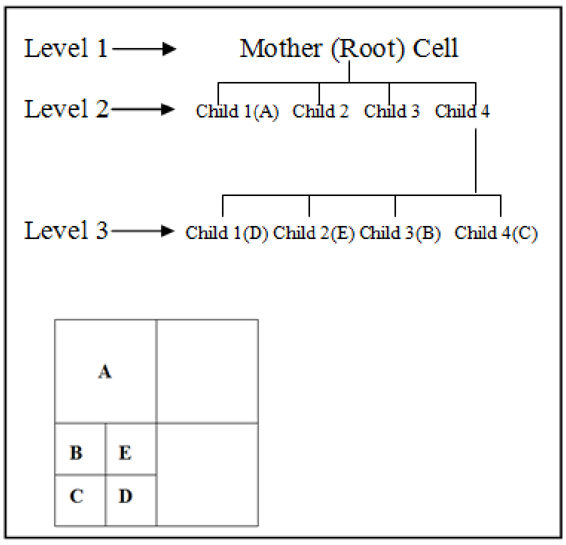

Fig. 1: An illustration of quad-tree data structure.

2.2 Geometry-Based Adaptive Grid Refinement

Generation of the initial grid begins with the creation of the ‘root’ cell. Its size is determined from the size of the flow field and by what the user determines to be the coarsest acceptable grid for that flow field. Then, cells without children, initially just the root cell, are refined until the grid reaches the coarsest acceptable grid. This grid serves as the initial grid for cases in which no body is cut out of the grid. The procedure for computing the intersections of the body with the grid depends upon how the body has been defined.

Each cell in the Cartesian grid is classified as;

- inside the body
- cut by the body
- outside of the body, or in the outer boundary based on the information found at the corners of that cell. With this approach, very small cells can appear in the grid. The only limit placed on

the cell size is that a node that is within a very small (machine-zero level) tolerance of a body is considered on that body, which in effect ‘pulls’ the body out to the node. The remaining cell areas vary as much as six orders of magnitude from one cell to its immediate neighbor.

2.2.1 Initial grid refinement

An initial uniform grid is formed in the solution domain which will be refined in the later stages to form the grid on which the solution is to be performed. The reason for forming this initial grid is to have a prescribed resolution at the outer boundaries of the domain.

Uniform grid generation is fairly straightforward: Starting from the root cell, refinement is performed when the cell level is less than the desired level. If a cell is to be refined, first the level of the edge and vertex neighbors must be investigated. If a neighbor is at a lower level, then that neighbor must be refined prior to the refinement of the cell. In this way, the one level rule is preserved in the process of;

$$x_c = x_{mother} + \frac{(d/2)}{2^l}, \; y_c = y_{mother} + \frac{(d/2)}{2^l} \qquad (1)$$

$$x_c = x_{mother} - \frac{(d/2)}{2^l}, \; y_c = y_{mother} + \frac{(d/2)}{2^l} \qquad (2)$$

$$x_c = x_{mother} - \frac{(d/2)}{2^l}, \; y_c = y_{mother} - \frac{(d/2)}{2^l} \qquad (3)$$

$$x_c = x_{mother} + \frac{(d/2)}{2^l}, \; y_c = y_{mother} - \frac{(d/2)}{2^l} \qquad (4)$$

for the 1st quadrant to 4th quadrant children, respectively, where d is the domain size and l is the level of the cell. Once this information is obtained, the neighbors of each cell must also be determined. The procedure followed is similar to the one outlined by De Zeeuw [3]. For any cell two edge neighbors and one vertex neighbor will be known automatically since they will be the children of the cell’s parent.

Grid generation for two-dimensional Cartesian grids can be achieved in three steps. Initial step is the creation of the domain and uniform grid generation. When sufficient resolution around the geometry is not obtained during the uniform grid generation, sometimes small parts of multi-component geometries are not realized; therefore, adaptation steps do not notice these parts due to the insufficient resolution. As a result, incorrect grid generations may result. In order to overcome this sort of shortcomings, some identification and curing techniques are implemented in the developed code. However, this does not mean, in no way, that these implementations will prevent all possible illnesses that may come through. Stating simply, the user should always be on alert about the possible emergence of these types of poor solutions.

The growth of the total number of cells during the construction of the uniform grid is exponential. However, in the developed code, there is no restriction on the desired number of levels for the uniform grid formation because coarsening of cells is allowed in this code. Higher levels may be desired and possibly necessary in some cases where small geometry components exist [4].

2.2.2 Box adaptation

The next step after the initial grid generation is the box adaptation. An imaginary rectangular box is generated around the input geometry and finer grids are flagged for refinement near the input body. An illustration of box adaptation around NACA0012 airfoil geometry is shown in Figure 2.

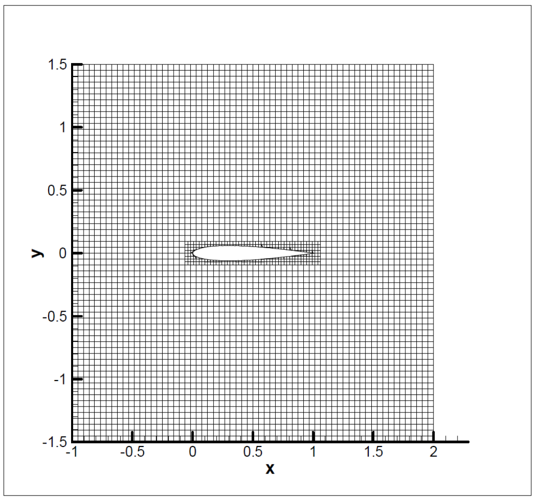

Fig. 2: Box adaptation to NACA0012 airfoil.

2.2.3 Cut-cell adaptation

Marching squares method is used to determine interfaces of cut and split cells [4]. The corners of sufficiently small cells are tested whether they are inside the boundary of the given geometry or not. This is called inside-outside testing. This step is obligatory for Cartesian grids because the cells that are cut by the given geometry are determined by this test, see Figure 3.

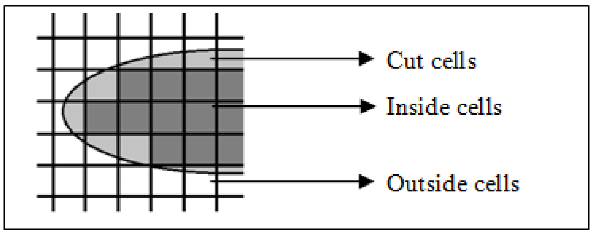

Fig. 3: Example of types of cells.

2.2.4 Curvature adaptation

The last step in the grid generation process is the curvature adaptation. The purpose of the curvature adaptation is to ensure that regions of a body that have high curvatures are resolved enough to be represented accurately (Since these regions will most probably be associated with high gradients they deserve special treatment, see Reference 5 for more detailed information). The result of curvature adaptation, generated by the developed code, to the nose of RAE2822 airfoil is shown in Figure 4.

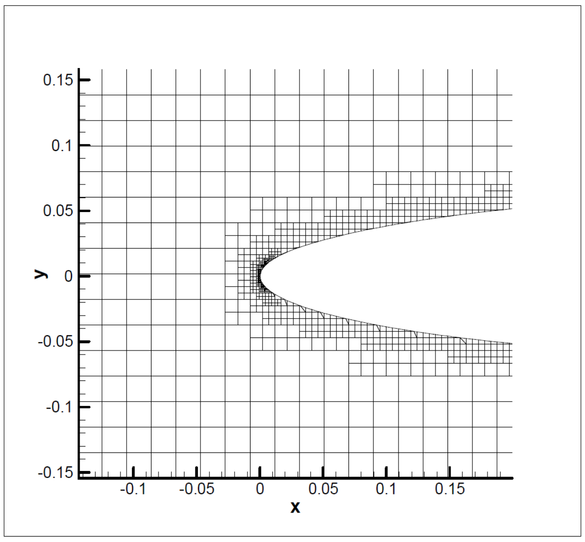

Fig. 4: Curvature adaptation to RAE2822 airfoil.

3 Example Grids

Two cases are exemplified to demonstrate the effectiveness of the grid generation scheme in symmetric and complex geometries.

3.1 NACA0012 Airfoil

The applications of cut-cell refinement and the curvature adaptation processes to NACA0012 airfoil are given in Figures 5 and 6, respectively. The remarkable change in mesh resolution around the nose and tail of the profile can easily be seen in these figures. Note also the formation of symmetrical mesh distribution near the tail of the profile, Figure 7.

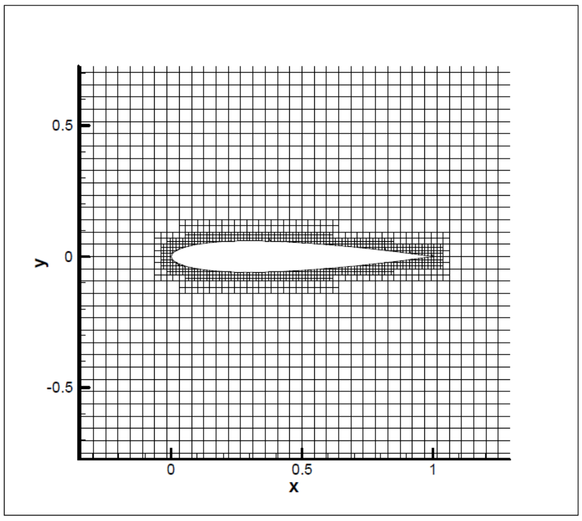

Fig. 5: NACA0012 cut-cell adapted grid.

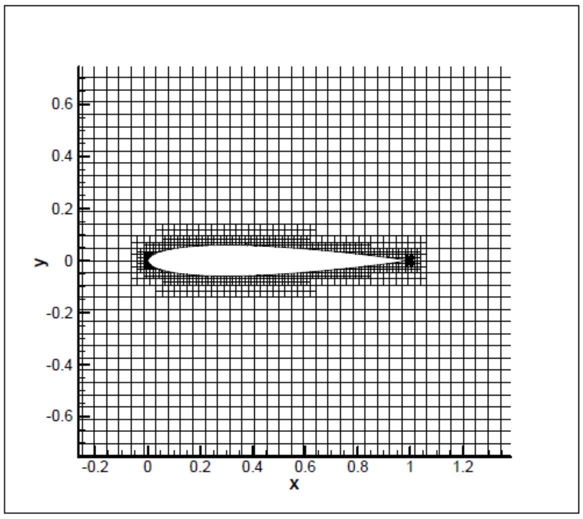

Fig. 6: NACA0012 curvature adapted grid.

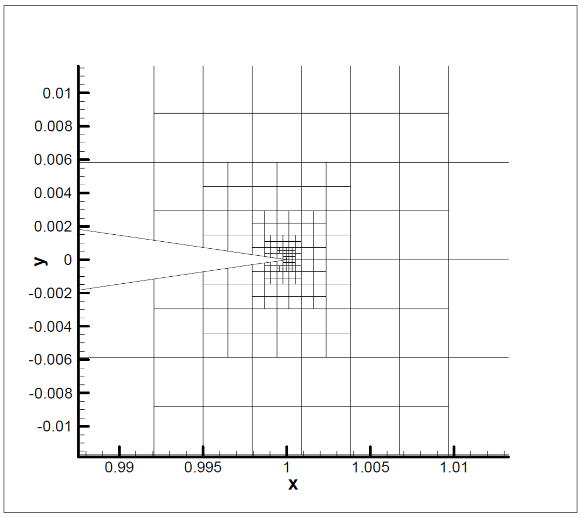

Fig. 7: NACA0012 symmetric grid near the tail.

3.2 Three-element Airfoil
The presentation purpose of this second example is to show resolution ability of the curvature adaptation technique between the elements of a complex airfoil. As it can be seen in Figure 8, the spaces between the elements are effectively meshed without any skewness or erroneous shape change. A close look at the near right-hand sided element shows that the split cells are generated effectively in Figure 9.

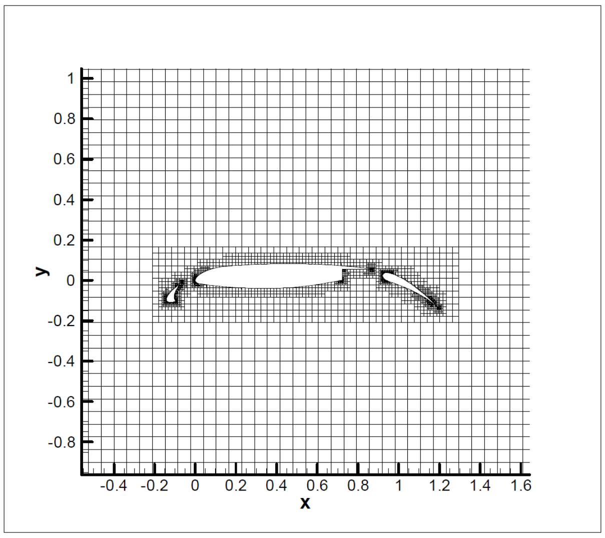

Fig. 8: Adapted grid on three-element airfoil.

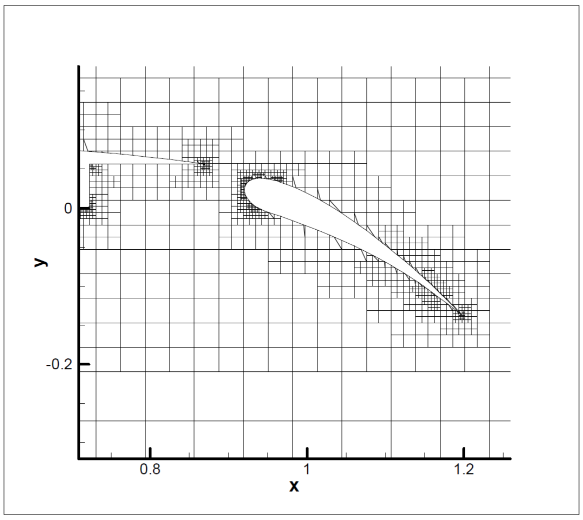

Fig. 9: Close look on three-element airfoil grid.

4 Conclusion
In the present paper, it is aimed to enhance automatic grid generation using Cartesian Methods, which will constitute the basis for the solution adaptation. Dynamic data structures are preferred in order to maintain faster computation. A cell-based quad-tree data structure is selected. A geometry-based adaptive refinement scheme for generating an initial grid is developed as the first step of the code. An adaptive refinement/coarsening scheme for generating the final grid is prepared by using some special Cartesian algorithms, namely marching squares algorithm, inside/outside test, cut/split cell determination and curvature adaptation algorithm. At the end of this work, a "hands-off" grid generator is implemented in FORTRAN programming language. Current study focuses on improving the adaptation near the highly skewed and highly curved elements. Two different cases are simulated at the end of the paper. The adaptation techniques are applied on the test domains effectively.

References:

[1] C.S.Peskin, Flow Patters Around Heart Valves: A Numerical Method, Journal of Computational Physics, Vol.10, 1972, pp. 252-271.

[2] H. Samet, The Quadtree and Related Hierarchical Data Structures, Theoretical Foundations of Computer Graphics and CAD, Vol. F40, 1988, pp. 51-69.

[3] D. De Zeeuw, *A Quad-tree Based Adaptively-Refined Cartesian-Grid Algorithm for the Solution of the Euler Equations*, PhD Thesis in the University of Michigan, 1993.

[4] M. Çakmak, *Development of a Multigrid Accelerated Euler Solver on Adaptively Refined Two- and Three-Dimensional Cartesian Grids*, PhD Thesis in the Middle East Technical University, 2009.

[5] B. Siyahhan*, A Two Dimensional Euler Flow Solver on Adaptive Cartesian Grids*, MSc Thesis in the Middle East Technical University, 2008.