DEVELOPMENT OF A TWO-DIMENSIONAL NAVIER-STOKES SOLVER FOR LAMINAR FLOWS USING CARTESIAN GRIDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY
MEHMET SERKAN ŞAHİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

MARCH 2011

Approval of the thesis:

DEVELOPMENT OF A TWO-DIMENSIONAL NAVIER-STOKES SOLVER FOR LAMINAR FLOWS USING CARTESIAN GRIDS

submitted by **MEHMET SERKAN ŞAHİN** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

ABSTRACT

DEVELOPMENT OF A TWO-DIMENSIONAL NAVIER-STOKES SOLVER FOR LAMINAR FLOWS USING CARTESIAN GRIDS

Şahin, Mehmet Serkan
M.Sc., Department of Mechanical Engineering
Supervisor : Prof. Dr. M. Haluk Aksel

March 2011, 202 pages

A fully automated Cartesian/Quad grid generator and laminar flow solver have been developed for external flows by using C++. After defining the input geometry by nodal points, adaptively refined Cartesian grids are generated automatically. Quadtree data structure is used in order to connect the Cartesian cells to each other. In order to simulate viscous flows, body-fitted quad cells can be generated optionally. Connectivity is provided by cut and split cells such that the intersection points of Cartesian cells are used as the corners of quads at the outmost row. Geometry based adaptation methods for cut, split cells and highly curved regions are applied to the uniform mesh generated around the geometry. After obtaining a sufficient resolution in the domain, the solution is achieved with cell-centered approach by using multistage time stepping scheme. Solution based grid adaptations are carried out during the execution of the program in order to refine the regions with high gradients and obtain sufficient resolution in these regions. Moreover, multigrid technique is implemented to accelerate the convergence time significantly. Some tests are performed in order to verify and validate the accuracy and efficiency of the code for inviscid and laminar flows.

Keywords: Cartesian Grid Generation, Quad Grid Generation, Navier-Stokes Equations, Least Squares Reconstruction, Flux Vector Splitting, Approximate Riemann Solver of Roe, Multigrid

ÖZ

KARTEZYEN HESAPLAMA AĞLARI KULLANILARAK LAMİNER AKIŞLAR İÇİN İKİ BOYUTLU BİR NAVIER-STOKES ÇÖZÜCÜSÜ GELİŞTİRİLMESİ

Şahin, Mehmet Serkan
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. M. Haluk Aksel

Mart 2011, 202 sayfa

Dış akış için tamamen otomatikleştirilmiş Kartezyen/Dörtgen hesaplama ağı üreticisi ve laminer akış çözücüsü, C++ programlama dili kullanılarak geliştirilmiştir. Ağsal noktalar ile geometri tanımlandıktan sonra, uyarlamalı Kartezyen hesaplama ağları otomatik olarak yaratılmıştır. Kartezyen hücreleri birbirine bağlamak için dörtlü ağaç veri yapısı kullanılmıştır. Viskoz akışları simule etmek için, gövde uyumlu dörtgen hücreler isteğe bağlı olarak yaratılmıştır. Hücreler arasındaki ilişki, şişirilmiş geometrinin çevresindeki Kartezyen hücrelerin kesim noktalarını en dış sıradaki dörtgenlerin köşeleri ile çakıştırarak kurulmuştur. Geometri çevresinde ve bu bölgedeki çok eğimli yerlerde, düzenli hesaplama ağına geometri bazlı uyarlamalar uygulanmıştır. Çalışma alanında yeterli bir çözünürlük elde edildikten sonra, çözüm hücre merkezli bir yaklaşımla, çok kademeli zaman uygulaması kullanılarak elde edilmiştir. Yüksek gradyanlı bölgeleri sıklaştırmak ve buralarda yeterli bir çözünürlük elde etmek için çözüme bağlı uyarlamalar program çalışırken gerçekleştirilmiştir. Ayrıca, yakınsamanın arttırılması için çoklu ağ yöntemi koda eklenmiştir. Kodun doğruluğu ve verimliliğini doğrulamak için viskoz olmayan akışlar ve laminer akışlar için bazı testler yapılmıştır.

Anahtar Kelimeler: Kartezyen Ağ Üretimi, Dörtgen Ağ Üretimi, Navier-Stokes Denklemleri, Yeniden Yapılandırma, Akı Vektör Ayrıştırması, Roe'nun Yaklaşık Riemann Çözücüsü, Çoklu Ağ Yöntemi

Dedicated to my family, Necla, Müslüm, Sercan Şahin
and Ayşegül Baylas...

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my thesis advisor, Prof. Dr. Haluk AKSEL for their guidance and supports throughout my research. Besides, I would like to give my gratitude to Mehtap Çakmak and Bercan Siyahhan. Finally, I thank to my parents, my brother and my friends (especially Ayşegül Baylas) for supporting me throughout my whole education life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

ALPHANUMERIC SYMBOLS

l	level of a cell
D_x, D_y	distance between body and box in x and y directions
D	domain size
k	boundary size factor, coefficient of thermal conductivity
n	body division factor
Re	Reynolds number
M	Mach number
Pr	Prandtl number
$\boldsymbol{Q}$	conserved variables vector
$\boldsymbol{F}$	inviscid flux vector
$\mathbf{G}$	viscous flux vector
dS	surface area element
u	velocity in x direction
v	velocity in y direction
E	specific total energy
p	static pressure
H	specific total enthalpy
q_x, q_y	heat flux terms in x and y directions
R	specific gas constant
T	temperature
c_v	specific heat for a constant volume
c_p	specific heat for a constant pressure, pressure coefficient
c_f	skin friction coefficient
e	specific internal energy
c	speed of sound
t	time

L_∞	reference length
c_∞	free stream speed of sound
v_∞	free stream velocity
Re_∞	free stream Reynolds number
M_∞	free stream Mach number
Res	residual vector
A	area of a cell
S_x, S_y	projections of edges
Δt_c	convective time step
Δt_v	viscous time step
$\boldsymbol{T}$	transformation matrix
L	non-linear differential space operator
e	error function

GREEK SYMBOLS

δ	boundary layer thickness
θ_{curv}	curvature angle
ρ	density
τ_{xx}, τ_{xy}, τ_{yy}	stresses
γ	ratio of specific heats
μ	laminar dynamic viscosity
ρ_∞	free stream density
μ_∞	free stream laminar dynamic viscosity
υ	CFL number
α_k	stage coefficient
α	angle of attack
Ψ_x, Ψ_y	convective spectral radii
λ_v	maximum eigenvalue of the diffusive operator
ε_p	relative change in pressure
ε_ρ	relative change in density

CHAPTER 1

INTRODUCTION

Fluid flow problems are generally governed by complex equations. Because of the nonlinearity in these equations, most problems cannot be solved by analytical techniques. Analytical methods are used for some problems where nonlinear terms are negligible. However, these terms are not negligible in general so that these problems must be solved by using numerical methods.

Computational Fluid Dynamics (CFD), is a branch of fluid mechanics that enables solution and analyses of fluid problems by using numerical methods and algorithms. Some problems such as problems having high Mach numbers or high temperatures cannot be simulated in laboratory conditions using wind tunnels. These problems and flows around multi-element complex geometries can be handled by using CFD. In the beginning of 1970's, CFD is started to be utilized for the solution of fluid flow problems with the evolution of computer technology. The simulations of transonic flows based on the non-linear potential equation were the first applications of CFD. In the early 1980's, two and three-dimensional Euler equations were solved. The rapidly increasing speed of computers and the development of acceleration techniques like multigrid enabled to solve inviscid flows around airfoils and inside of turbomachines. In the late 1980's, the focus was shifted to viscous flows. Navier-Stokes equations were solved with the improvement of different turbulence modelling techniques such as the direct numerical simulation and large eddy simulation in order to analyze the turbulence flows accurately [1].

Nowadays, due to the high speed and large memory computers, it is possible to analyze the inviscid or viscous flows in two or three dimensional space around multi-element, complex geometries. In order to obtain accurate results by CFD, it is important to choose the right numerical technique for flux computations, to choose the suitable turbulence model, to have high algorithmic efficiency and to generate a grid having sufficient resolution around critical regions such as shock locations, high gradient locations, wakes etc.

A CFD code consists of three main elements, namely pre-processor, flow solver and post-processor. A pre-processor generates the grid around the geometry which is specified by the inputs. In addition, the flow parameters and boundary conditions are defined in the pre-processor. Then, flow solver uses the governing equations in order to solve the flow around the geometry subjected to the defined pre-conditions by one of the three common solution methods which are finite difference, finite element and finite volume method. Finally, post-processor forms the output files where results are shown in graphical and easy to read format [2].

1.1 MESH GENERATION

Mesh generation is a crucial step of CFD. In order to get accurate results, mesh resolution must comply with the solution schemes. In other words, an efficient grid must be generated in order to minimize the errors while resolving the physical properties of the flow. This grid must use as fewer grid points as possible in order to use the memory efficiently [3]. In general, there are two types of meshes; structured and unstructured meshes.

1.1.1 Structured Meshing

A structured mesh consists of quadrilaterals in two-dimensional space and hexahedra in three-dimensional space. The grid points are located in sequentially with the aid of an array (i, j, k) so that the connectivity information among them is provided implicitly. To illustrate, a neighbor of a grid point which is in the direction of i, j or k

indexes can be reached by adding or substracting an integer to or from i, j or k index of the grid point itself [4].

Structured grids have some advantages compared to unstructured meshes. Data structure is less complex due to the implicit connectivity information. Moreover, memory usage is less due to the simpler connectivity structure. As a result, more efficient and simpler codes can be developed by using structured grids. It is also advantageous for viscous flows. By adjusting the grid spacing, high resolution can be obtained in the boundary layer which is the most important region for viscous flows.

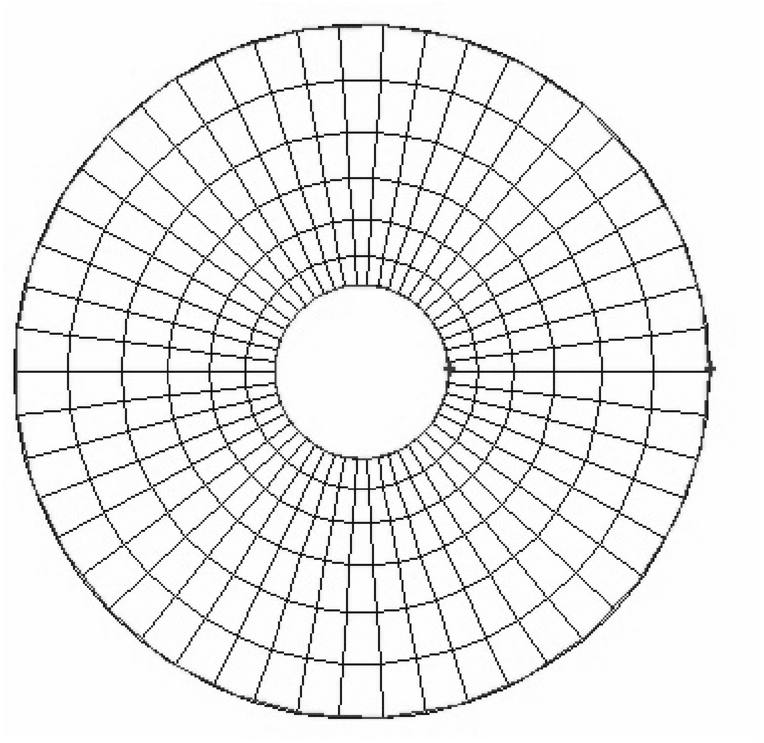


Figure 1.1 Illustration of a structured mesh

Besides its advantages, there are some disadvantages in comparison to unstructed grids. Since the edges of the geometry are not in the direction of the primary coordinate axes, transformation from physical to computational space is necessary. This task needs more computational power. Furthermore, grid generation around complex, multi-element geometries is a very complex problem. In order to eliminate this problem, some techniques such as Chimera and multi-block is used. However, these are so complicated that the advantages of structured grids diminish [4]. Whereas structured meshes has numerous advantages, they are not generally preferred due to its disadvantages, especially the one that meshing cannot easily be applied to complex geometries.

1.1.2 Unstructured Meshing

An unstructred mesh consists of triangular or quadrilateral cells in two-dimensional space, hexahedral, prismoidal, pyramidal and tetrahedral cells in three-dimensional space, in an irregular pattern. Unlike structured grids, unstructured grids require a list of the connectivity, which leads to a more complex data structure. Moreover, this complex data structure causes higher memory usage.

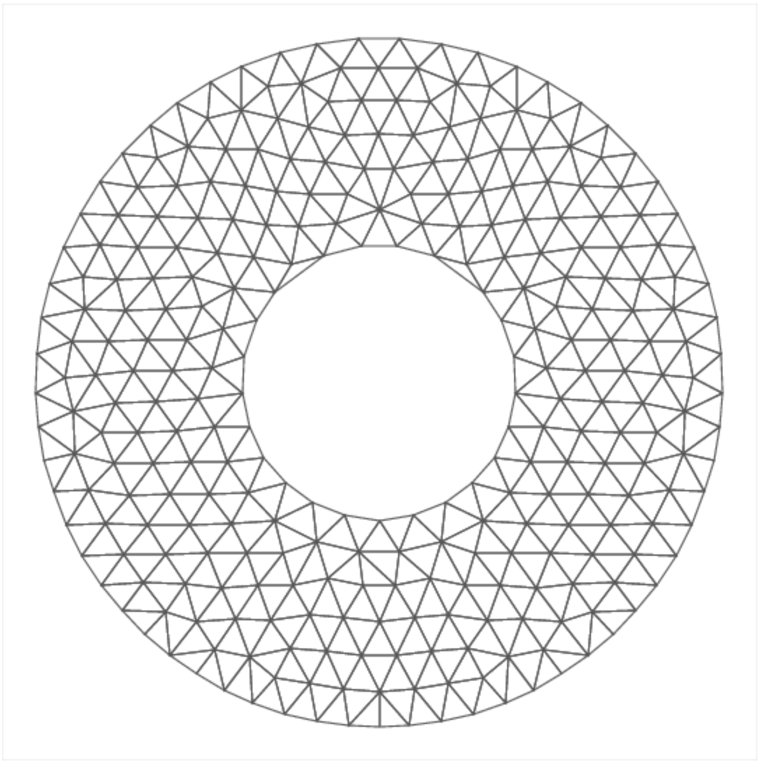


Figure 1.2 Illustration of an unstructured mesh

With the recent advancements in computer technology, efficiency of a CFD code is not affected very much by the high memory requirements. In addition to this, due to the capability of handling complex geometries easily, the popularity of unstructured meshing is increased among the meshing techniques.

Advancing front and Delaunay triangulation methods are the most widely used methods in unstructured meshing. In reference [5], detailed information can be found.

1.1.3 Cartesian Meshing

A Cartesian mesh is a special type of unstructured meshing where the cells are squares in two-dimensional space and cubes in three-dimensional space. Quadtree and octtree data structures are used for two-dimensional and three-dimensional spaces, respectively. It was not preferred in the past since it is very hard to handle curved boundaries. However, with the recent developing techniques dealing with these curved regions, Cartesian meshing becomes more popular.

One important advantage of Cartesian grids is that it requires hardly any user interference so that automatic meshes can be generated around even complex and multi-element airfoils easily. Denser meshes around shocks, shear layers and wakes can be obtained easily without user interference by using solution adaptation. In addition, multigrid technique which accelerates the convergence rate can be applied very easily since quadtree data structure is used for a two-dimensional Cartesian grid. Furthermore, the transformation of physical variables from computational space to physical space is applied only for the cells near the boundary since the other cells are in the direction of the primary coordinate axes.

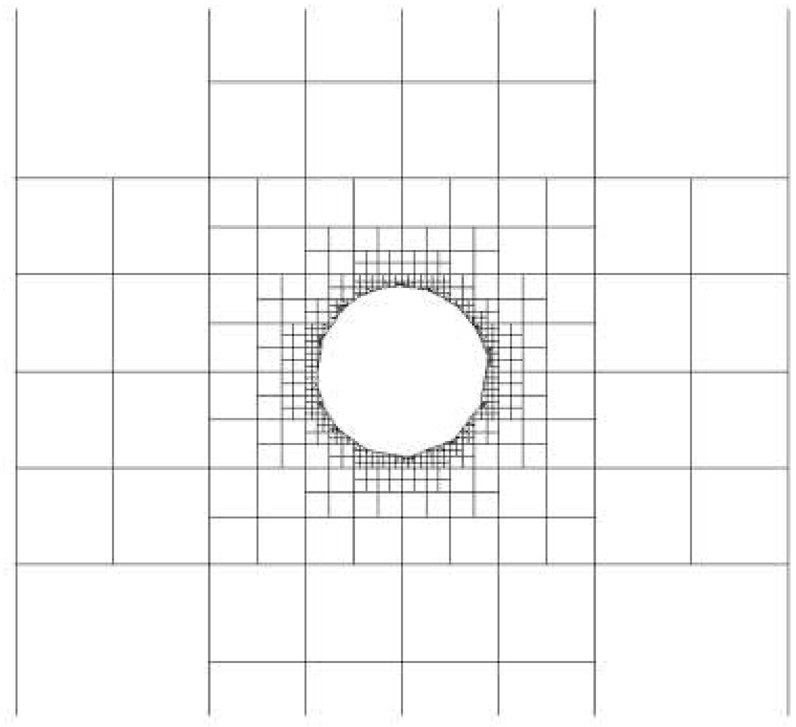


Figure 1.3 Illustration of a Cartesian mesh

Besides its advantages, there are some disadvantages, as well. One of the most difficult parts of Cartesian meshing is to deal with the curved parts of the geometry. The shapes of the cells which are intersected by the geometry are modified. The shape of these cells are not square and they are called irregular cells. It is very important to resolve the physical quantities at the irregular cells correctly in order to have accurate results. To do this, physical variables must be rotated into physical space. Moreover, the smaller sizes in those cells compared to regular Cartesian cells can cause deceleration of convergence rate. Local time stepping and multigrid technique can eliminate these problems. In addition, in order to model the viscous flows, the Cartesian meshing is not sufficient [6]. To have sufficient resolution in boundary layer, hybrid grid, which is composed of Cartesian mesh outside the boundary layer and body-fitted quad grid inside the boundary layer, is generated.

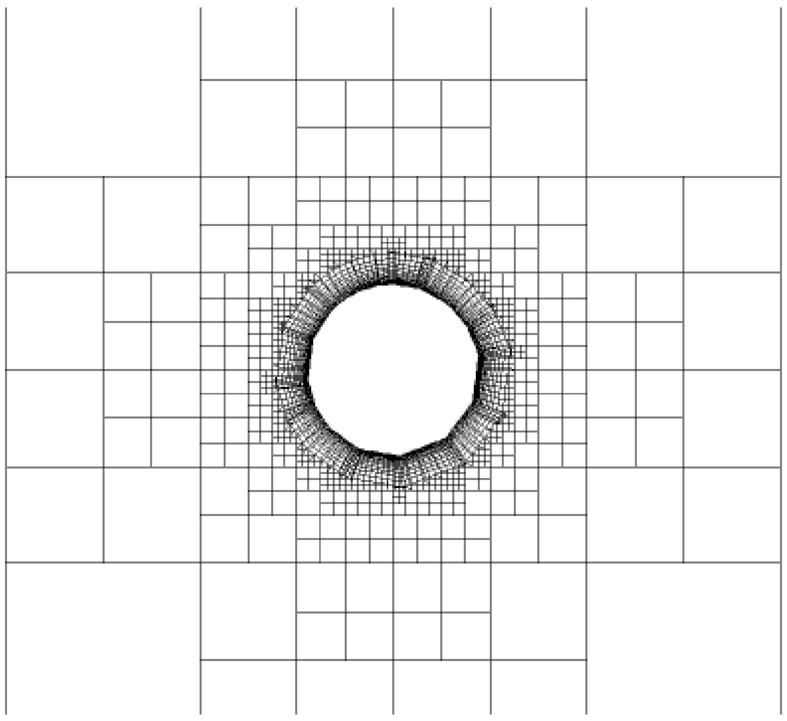


Figure 1.4 Illustration of a hybrid mesh

1.2 LITERATURE REVIEW

Wang (1998) developed a second-order accurate, cell-centered viscous flow solver by using a quadtree-based adaptive Cartesian/quad grid. In mesh generation part, first, geometrically-adaptive, body-fitted grids are automatically generated. After obtaining a user specified minimum grid resolution by recursively Quadtree divisions of a large root cell, Cartesian cells are intersected by the outer boundary of the Quad cells. By using cell-cutting method, final computational grid is produced automatically. In the solver part, after obtaining converged solutions in a given grid, solution-based adaptations are performed [7].

Ye, Mittal, Udaykumar and Shyy (1999) developed a Cartesian grid method for two-dimensional, unsteady, viscous, incompressible flows around complex immersed boundaries. In this method, finite volume method based on second order central difference scheme and a two-step fractional-step procedure are used. An interpolation procedure is applied for accurate discretization of the governing equations in the boundary cells. This procedure allows systematic development of a spatial discretization scheme that preserves second-order spatial accuracy of the solver. Since the iterative solution is slowing down by the fact that conditioning of the linear operators are changed with the presence of immersed boundaries, the convergence is accelerated by using a preconditioned conjugate gradient method [8].

Wang, Cphen, Hariharan, Przekwas and Grove (1999) developed a 2^n tree based viscous Cartesian grid generation method for complex geometries. With 2n data structure, it is easy to handle complex geometries and deal in shocks, shear layers and wakes since it supports anisotropic grid adaptations in any of the coordinate directions. To resolve boundary layer for viscous flows, a viscous layer grid whose thichness is determined according to the expected thickness is added between Cartesian grid and body surface through a projection technique. Furthermore, an algorithm which detects critical regions has developed and good quality computational grids has been produced by avoiding cell-cutting completely [9].

Tucker and Pan (2000) implemented a Cartesian cut cell method to incompressible viscous laminar flows. In this method, some cut cells are created at solid boundary surfaces. For these cells, a novel hybrid technique is applied while integrating the governing Navier-Stokes equations. This technique consists of surface cell trimming and interpolation [10].

Wang (2000) developed a nested multi-grid solution algorithm for an adaptive Cartesian/Quad grid viscous flow solver. Body-fitted quadrilateral grids are produced around the solid geometry by the method of surface extrusion. After overlapping Quad grids with Cartesian grids, cell-cutting is performed in order to obtain the final computational grid. While the Cartesian grid is obtained by a single root using

Quadtree data structure, Quad grids are generated from multiple roots which are termed as a forest of Quadtrees representing the coarsest possible Quad grids. The coarsening algorithm, which is necessary to produce multi grids, is based on the reverse tree of Quadtree data structure. The flow solver is based on Roe's flux splitting, finite volume discretization with a cell-centered method, least-squares reconstruction and a differentiable limiter. In order to handle very small cut cells, local time stepping scheme is used as a time. For multigrid strategy, several cycling techniques such as Saw-Tooth Cycle, W-Cycle and V-Cycle are used [11].

Kirkpatrick, Armfield and Kent (2003) presented a method for representing curved boundaries in order to solve the viscous governing equations on a non-uniform, staggered, three-dimensional Cartesian grids. The method proposes that Cartesian cells at the boundary surface are truncated so that new cells are created and the boundary grid fits the shape of the surface completely. In the paper, some problems related to the development of a cut cell in staggered grid are discussed in a detailed manner. Second order accuracy is provided with the derived flux calculation methods through the boundary cell faces. On top of that, a method called "cell-linking" is developed in order to overcome the problems originated from the creation of small cells while avoiding the complexities resulted from the cell-merging operations [12].

Russell and Wang (2003) developed a Cartesian grid method for multiple moving objects in an incompressible two-dimensional viscous flow. The system is generated by regular Cartesian grid and solved by using a vorticity-stream function formulation. The no-penetration condition for the moving object and no-slip condition are provided by superposing a homogenous solution to the Poisson's equation for the stream function and producing vorticity on the surfaces of the moving objects [39].

Gilmanov, Sotiropoulos and Balaras (2003) presented an algorithm for a general reconstruction, while analyzing flows with complex three-dimensional immersed boundaries using Cartesian grids. In this algorithm, solution in the Cartesian grid nodes near the interface of the unstructured, triangular mesh generated by discretizing three-dimensional immersed solid surface is reconstucted by using linear interpolation along the local normal to the body. As a result, the overall accuracy of the solver is second order [14].

Sanmiguel-Rojas, Ortega-Casanova, del Pino and Fernandez-Feria (2005) developed a method for incompressible two-dimensional viscous flows arround irregular geometries which generates a non-uniform Cartesian grid such that all boundary points are regular mesh points. The generated non-uniform grid is solved by the Navier-Stokes equations using finite difference methods [15].

Verstappen and Dröge (2005) developed a numerical method for solving unsteady, incompressible Navier-Stokes equations on Cartesian grids for arbitrarily-shaped boundaries. A novel cut-cell discretization method is introduced. This method provides the preservation of the spectral properties of convection and diffusion. A skew symmetric operator is used while discretizing convection and a symmetric, positive-definite coefficient matrix is used while approximating diffusion. This coefficient matrix conserves kinetic energy on any grid if the dissipation is turned off [16].

Singh and Shy (2007) presented three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction. In this method, multiphase flows and moving boundaries between different phases are considered. The moving boundary is tracked using triangulated surface grids and the flow is solved by using governing equations on a stationary Cartesian grid. This grid is locally adaptive so that the resolution requirements can be provided. The interface resolution is controlled via a conservative restructuring technique which satisfies conservation of mass. In addition, a reconstruction algorithm for topology change is implemented [17].

Ito, Lai and Li (2009) developed an augmented method based on a Cartesian grid for solving Navier-Stokes equations in irregular domains. A fast Poisson solver is utilized in the projection method after embedding the irregular domain into a rectangular one. The jump in the normal derivative of the velocity is set as the augmented variable so that ill conditioned system, which is usually produced by the methods setting force strengths as unknowns, is avoided. With this approach, condition number of the system is improved significantly for the augmented variable. In addition, the second order accuracy is provided for the velocity by using immersed interface method [18].

Karagiozis, Kamakoti and Pantano (2010) proposed a numerical method in order to solve the compressible Navier-Stokes equations on Cartesian grids. In this method, an embedded geometry representation of the objects is used and the governing Navier-Stokes equations are approximated with a low numerical dissipation centered finite-difference discretization. This method is useful for immersed boundaries, not suitable for compressible flows with shocks [19].

Hartmann, Meinke and Schröder (2010) developed a strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids. In this approach, finite volume method is used allowing the conservation of mass, momentum and energy at the boundaries. Up to 2010, there is not such a proposed method in literature for three dimensional compressible flows. While solving the mesh, a linear-least squares reconstruction is used to rebuild the gradients of the cell centers in irregular regions of the mesh and those are employed while calculating the flux at the surface. As a result, the accuracy of the solution is second order [20].

In addition, several researches were done about Cartesian meshing in Department of Mechanical Engineering in METU. Siyahhan (2008) solved two-dimensional Euler equations by using flux vector splitting methods which are AUSM, AUSMD, AUSMV and Van Leer in addition to Roe’s method while the mesh is generated by using Cartesian grids. Multistage time stepping is used for temporal discretization. Moreover, the flow variables are reconstructed in order to increase the accuracy [22].

Çakmak (2009) developed an Euler solver on adaptively refined two and three dimensional Cartesian grids. The solution is obtained by cell-centered finite volume method. While calculating inviscid fluxes, flux vector splitting and flux difference splitting methods are used. In the mesh generation part, a dynamic data structure is used together and geometric based adaptations are applied. In addition, solution adaptation is applied to the mesh in order to refine the regions with high gradients. In order to accelerate the convergence rate, local time stepping and multigrid techniques are embedded to the developed code [21].

1.3 ORGANIZATION OF THE THESIS

In this thesis, a flow solver with an adaptive Cartesian or hybrid grid generated automatically around simple and complex, one or multi element airfoils is developed. The flow solver is capable of analyzing the compressible inviscid or laminar external flows. As a solution method, finite volume technique is used.

In Chapter 2, mesh generation is discussed in detail. After quadtree data structure is introduced, the steps for Cartesian grid generation which are uniform mesh generation, cell type determination and geometric adaptations are discussed in detail. Finally, quad grid generation is introduced and hybrid grid generation is explained.

In Chapter 3, governing equations in integral form are introduced for viscous flows. Next, discretization of governing equations temporally and spatially are presented. Then, inviscid and viscous flux computations are discussed in detail. Reconstruction of flow variables is explained. After discussing calculation of non-dimensional coefficients, pressure and skin friction coefficients, refinement based on solution adaptation is introduced.

In Chapter 4, multigrid method, which is an acceleration technique, is introduced and the steps for the application to non-linear problems are discussed in detail. Coarsening process of Cartesian and quad grids are explained, which is necessary for multigrid applications. Then, the effect of multigrid on inviscid and low Reynolds number flows is discussed with tables and graphs.

In Chapter 5, inviscid flow around a single element airfoil is validated and the results are discussed. Next, low Reynolds number flow is considered by testing two different problems. Finally, a multi-element airfoil is considered at a high Reynolds number to show the hybrid grid effect.

In Chapter 6, techniques used in the developed code and the obtained results are discussed. Then, some suggestions are made, for future works.

CHAPTER 2

MESH GENERATION

In this chapter, data structure used in the code for Cartesian meshing discussed. While quadtree data structure is introduced, stored variables and connectivity information for Cartesian cells are given.

Next, generation of Cartesian mesh around an airfoil is explained. Uniform mesh generation, cell type determination and geometric adaptations are mentioned in detail. As a result of these processes, a good resolution around the geometry is obtained and all necessary properties of cells are stored in order to use them in solution.

Finally, quad grid generation is introduced. Before quad cells are created, boundary layer is set by puffing the geometry up. During this process, some unwanted situations are eliminated. After obtaining a good boundary layer, the quad cells are created in the layer and the quad cells are connected to Cartesian cells and each other. As a result, a hybrid mesh having sufficient resolution at critical regions is formed.

2.1 DATA STRUCTURE

In this code, domain is divided into cells with Cartesian meshing. Since Cartesian meshing is a type of unstructured mesh, the connectivity information between cells is not provided simply like structured meshes. In order to store the data of cells and provide connectivity information successfully, an appropriate data structure should be used. For two dimensional problems, several data structures can be chosen. The main data structure types are linked list, binary tree and quadtree types.

In the developed code, the quadtree data structure is chosen since it is more advantageous than the others. It is easy to apply solution adaptation, where the dynamic cell number is required; i.e. e. number of cells is changing with solution adaptation. Furthermore, multigrid adaptation can be applied without creating new coarser grids unlike the others.

In this section, the main properties of quadtree data structure is introduced, first and then, the connection of the cells in the domain with each other is discussed. Finally, the variables that must be stored are explained in detail.

2.1.1 Quadtree Data Structure

Quadtree data structure is a tree data structure in which each cell has four children. The two dimensional space is partitioned recursively by subdividing it into four equal quadrants from the mid points of edges, so that four equal size squares are generated, until the desired resolution is obtained.

The largest cell covers the whole domain and it is called "root cell". Root cell has four children and each of these children have four children and so on. Children are separated from each other by naming it according to their location in the larger cell. They are named as top left, top right, bottom left and bottom right. The cells which have no children are called "leaf cells". These cells are used for solution calculations and also called "computational cells".

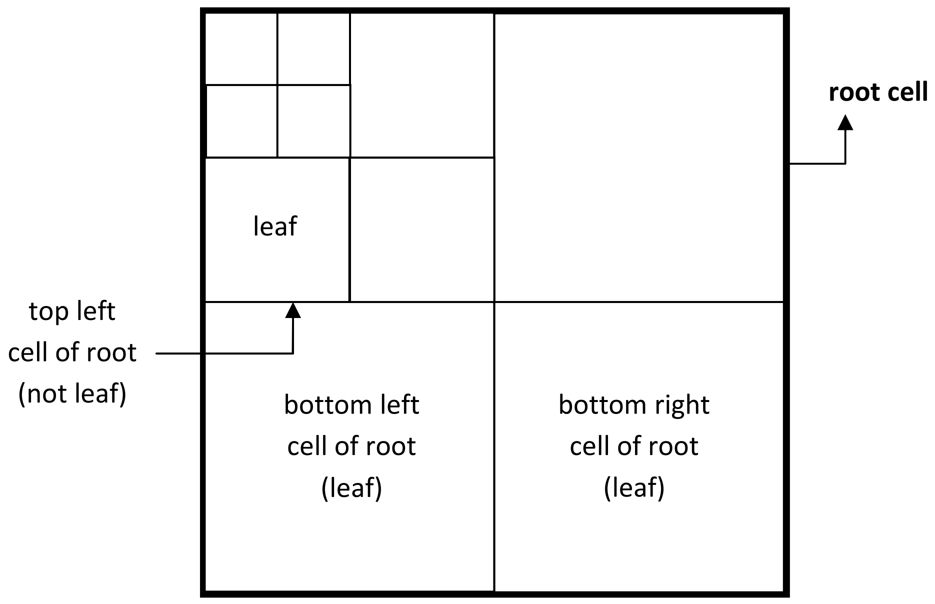


Figure 2.1 Illustration of root cell and its children

While the relationship between cells from the largest to smallest is supplied by children phenomenon, the inverse relation from the smallest to the largest cells is provided by the word "parent". For example, a cell has four children and these four children have a parent which is the mentioned cell. Moreover, for providing relationship accurately, there is a level concept that every cell has. It shows the number of divisions until the cell under consideration is obtained from the root cell. The root cell has a level of 0. The level of four children of a cell is assigned a level of one higher of their parent cell. While Figure 2.1 illustrates root cell and leaf cells considering children parent relations in a 2-D cell, the same relations is shown in a tree view with their levels in Figure 2.2.

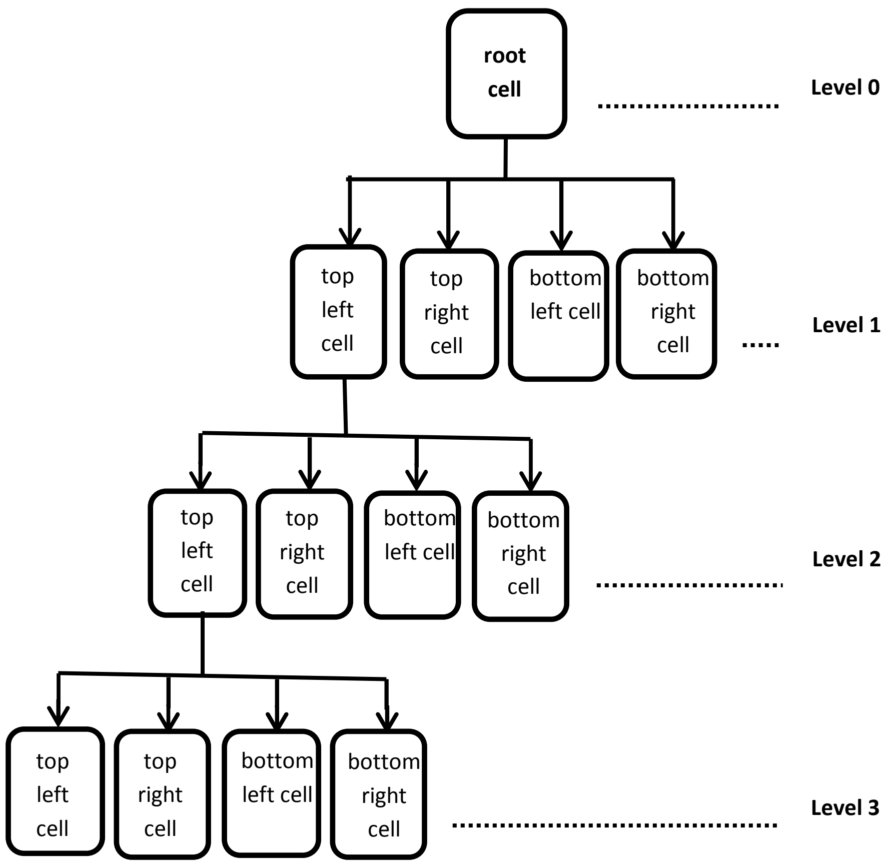


Figure 2.2 Children-parent relationship in a tree view

2.1.2 Connectivity

Connectivity of cells is provided by not only children-parent relation, but also by the neighborhood relationship. Each cell has four neighbors, namely top neighbor, right neighbor, left neighbor and bottom neighbor. In other words, for each cell, four more pointers are required in addition to children and parent pointers. Totally, 9 pointers are necessary in order to provide the connectivity accurately [21], [22].

While finding neighbors, the relation between a parent and their children is used. Starting from the root cell, neighbors are found. At first, neighbors of children of the root cell are found. It can easily be determined by considering the locations of children in the root cell. The determination of the top left child of the root cell is shown in Figure 2.3.

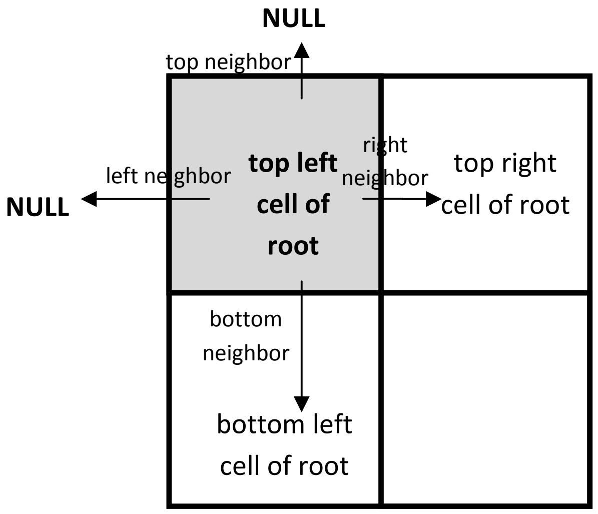


Figure 2.3 Neighbors of top left cell of root

When neighbors of first level cells are found, their parents and location of their children places are used as shown above. For higher levels, in addition to the location of the children of a parent, parent's neighbors are also used. Below, determination of neighbors of a second level cell, shown in Figure 2.4, is explained in Table 2.1.

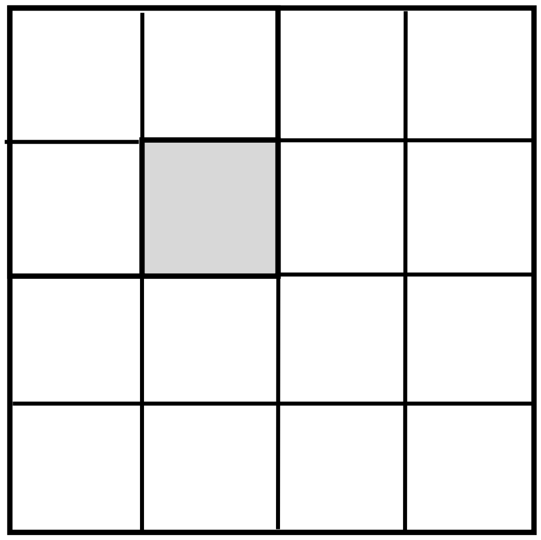


Figure 2.4 Neighbors of a second level cell

Table 2.1 Determination of neighbors of a second level cell

NEIGHBORS	DETERMINATION
Top Neighbor	Parent→Top Right Child
Left Neighbor	Parent→Bottom Left Child
Bottom Neighbor	Parent→Bottom Neighbor→Top Right Child
Right Neighbor	Parent→Right Neighbor→Bottom Left Child

Since the entire domain is not at the same level, the above illustration is not the only case one may meet. The neighboring cells may have lower or higher levels. It is very difficult to handle neighboring cells which have more than one level difference. Therefore, the code is adjusted that only one level difference can exist between the two neighboring cells. This fact is explained in more detailed fashion in Section 2.2.3.4.

When a cell has a neighbor which is one level higher than itself, nothing changes during the determining neighbors. The neighbor of the cell will be the parent of that cell having the lower level, since the parent shares the same edge with the cell and has the same level. However, when one considers a cell whose neighbor is at one level lower less than itself, than a slight change is necessary. The neighbor will not be the appropriate child of the appropriate neighbor of the its parent. Since it has no child, the neighbor will be directly the parent's appropriate neighbor.

2.1.3 Stored Variables

Storing the correct variables is quite important in order to use the memory efficiently. Excessive storage results in inefficient memory usage and slows down the computations. On the contrary, while trying to decrease the number of stored variables, to calculate the same variable again and again slows down the calculations. Therefore, optimization according to today's memory technology is necessary.

The pointers identifying the stored variables can be classified into six groups. These are geometric pointers that define the cell geometry, connectivity pointers that relate the cell with the others, cell type pointers which are used to determine the cell type, solution pointers which are necessary for solving the governing equations, solution adaptation pointers which are required for the solution refinement and multigrid pointers in order to create coarser grids to be used in the solution. In addition to primitive types in C++, some classes are defined in order to handle these pointers more easily. These user-specified classes have also some stored variables.

While the cells have these pointers, some static variables are also used. With the use of these, only one variable is stored instead of a number of cell variables. In other words, for each cell, these static variables are calculated and they are used during the application of the necessary methods. After that, instead of creating a new variable for a new cell, the necessary quantity is recalculated for the new cell and stored at the same variable since the old one is no longer used.

In the developed code, all cells have some common geometrical variables. All cells have four corners, centroidal coordinates, center coordinates and area. The center and the centroid are similar to each other for out cells. However, they are different for cut and split cells. While center means the middle point of the Cartesian square cell whether the shape of the cell is square or an arbitrary shape, the centroid represents the mass center of the shape covering outside the geometry of the Cartesian cell. Furthermore, in the developed code, the flux calculations are done through the faces so that storage of the faces is very important for leaf cells.

Table 2.2 Geometric pointers

POINTER NUMBER	**POINTER TYPE**	**POINTER NAME**	**POINTER FOR**	**EXPLANATION**
1	CornerPt	topLeftCorner	All cells	Corner point at top left
1	CornerPt	topRightCorner	All cells	Corner point at top right
1	CornerPt	bottomLeftCorner	All cells	Corner point at bottom left
1	CornerPt	bottomRightCorner	All cells	Corner point at bottom right
1	double	area	All cells except incells	Area
1	Pt	center	All cells	Center point
1	Pt	centroid	All cells except incells	Centroid point
vector	Face	faces	Leaf cells	Face vector

Connectivity information between cells is provided with totally 16 pointers. Four of them represent the children of the cell while the other four of them denotes the side neighbors of the cell. In addition, as discussed earlier, the number of divisions is stored with the aid of the "level" pointer and the inverse connection is provided with the "parent" pointer. Furthermore, the remaining six pointers are necessary to relate the splitToCut type cells with their inclusive cell and quad type cells with their inclusive cell directly and inversely. These are mentioned in detailed in Section 2.2 and 2.3, respectively.

Table 2.3 Connectivity pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
1	Cell	topLeft	All cells	Child cell at top left location
1	Cell	topRight	All cells	Child cell at top right location
1	Cell	bottomLeft	All cells	Child cell at bottom left location
1	Cell	bottomRight	All cells	Child cell at bottom right location
1	Cell	topNeighbor	All cells	Neighbor cell at top side
1	Cell	bottomNeighbor	All cells	Neighbor cell at bottom side
1	Cell	leftNeighbor	All cells	Neighbor cell at left side
1	Cell	rightNeighbor	All cells	Neighbor cell at right side
1	Cell	parent	All cells	Parent cell
1	Cell	splitToCut1	Split cells having 2CV	SplitToCut cell forming with the first control volume of split cell having 2 control volumes
1	Cell	splitToCut2	Split cells having 2CV	SplitToCut cell forming with the second control volume of split cell having 2 control volumes
1	Cell	quad1	Cells except out and in cells	First quad cell
1	Cell	quad2	Cells except out and in cells	Second quad cell
1	Cell	inclusiveOfSplits	Split cells having 2CV	Inclusive cell of the splitToCut cell
1	Cell	inclusiveOfQuads	Cut, split and splitToCut cells	Inclusive Cartesian cell of the quad cell
1	int	level	All cells	Division level

Third group pointers are used in order to determine cell type. "type" pointer is an enumerator type and determines the cell whether it is an out, in, cut, split, splitToCut, quad or notDefined. If the cell does not have some specific properties, than its type is set to notDefined so that it should be refined or the geometry must be shifted in order to eliminate these cells. For some split cells, "nodeIn" pointer is used in order to present geometry correctly. This pointer defines a nodal point of the input geometry in the cell. It is used especially for split cells at highly curved parts of the geometry. The pre-determined "IntPt" type vector of "intersections" give the points intersected with the geometry. Square and split indices are used for split and cut cells and they determine the sub-type of the split or cut cells. In Appendix A, one can see the sub-types of these cells. Moreover, the usage of these indices is presented in Section 2.2.2.2.

Table 2.4 Cell type pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
1	enum	type	All cells	Type (cut, split, in, out, quad, splitToCut, notDefined)
1	IntPt	nodeIn	Split cells	Node point of the geometry in the cell
vector	IntPt	intersections	Cut, split and splitToCut cells	Intersection points with the geometry
1	int	squareIndex	All cells	determines the sub-type
1	int	splitIndex	Cut, split and splitToCut cells	determines the sub-type

Next group is the solution pointers. In this group, two pointers are used to store the conserved variables of the cell at the centroid, before and after the iteration. Since the one before the iteration is necessary for the new calculations, two of them must be stored separately. Additionally, residuals for those variables are stored using "res" pointer. The gradients of these variables in x and y directions and viscosity are stored with totally 9 pointers. Finally, if reconstruction and gradient limiting are chosen (they are mentioned in Chapter 3), then 4 additional pointers are needed for the limiters of the four conserved variables.

Table 2.5 Solution pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
4	double	qOld	Leaf cells	conserved variables before the iteration
4	double	qNew	Leaf cells	conserved variables after the iteration
4	double	res	Leaf cells	residuals of conserved variables
4	double	dqdx	Leaf cells	x gradient of conserved variables
4	double	dqdy	Leaf cells	y gradient of conserved variables
4	double	limiter	Leaf cells	gradient limiters if order of scheme is 2.
1	double	viscosity	Leaf cells	laminar non-dimensional viscosity

As the solution adaptation pointers, two pointers are used. These pointers are for the curl and divergence criteria of the solution adaptation for each leaf cell, as mentioned in Chapter 3.

Table 2.6 Solution adaptation pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
1	double	tau	Leaf cells	divergence criteria for solution adaptation
1	double	ksi	Leaf cells	curl criteria for solution adaptation

The final group is for the pointers required for the multigrid applications. For the application of the multigrid technique, a total of 8 pointers are required. Multigrid is a very detailed convergence acceleration technique so that it is explained separately in Chapter 4. To summarize, the words "perform" and "meshSpacing" are used for the coarsening of the finest mesh. The word "compCell" determines whether the cell is a computational cell or not in a given computational grid. Forcing function is used in order to correct the residuals using coarser grids.

Table 2.7- Multigrid pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
1	int	perform	All cells	determines whether the cell is coarsened or not
1	int	meshSpacing	All cells	determines the step number of mesh
1	int	compCell	All cells	determines whether the cell is computational cell
4	double	FF	All cells	forcing function of conserved variables

As mentioned before, some static pointers are used to avoid excessive storage. Eight of these are the conserved variables, which are transformed according to the face for left and right states. The "stress" pointer determines three stresses, τ_{xx}, τ_{yy} and $\tau_{xy,}$ and includes the heat fluxes terms, q_x and q_y. Although the heat fluxes are not stresses, in the developed code heat terms are added to stress pointer in order to handle them more easily. Finally, the remaining 8 pointers are used for inviscid and viscous fluxes at the face of the cell.

Table 2.8 Static pointers

POINTER NUMBER	POINTER TYPE	POINTER NAME	POINTER FOR	EXPLANATION
4	double	qLeftBar	Leaf cells	transformed conserved variables of left state at the face
4	double	qRightBar	Leaf cells	transformed conserved variables of right state at the face
5	double	stress	Leaf cells	stresses and heat flux terms in x and y at the face
4	double	faceFlux	Leaf cells	inviscid fluxes at the face
4	double	faceViscousFlux	Leaf cells	viscous fluxes at the face

2.2 CARTESIAN GRID GENERATION

In the developed code, as mentioned earlier, Cartesian meshing is used. This grid is adapted to the code with the quadtree data structure. While generating the Cartesian mesh, totally three steps are applied in order. First, uniform mesh is generated around the created domain. Then, the types of the cells are found using intersection methods and indices. Finally, the geometric adaptation is applied to the uniform mesh so that the grid around the geometry becomes finer in order to get accurate results.

2.2.1 Uniform Mesh Generation

The input geometry is specified in terms of the nodal points. By connecting the consecutive nodes, the geometry can be obtained. The first and last nodes of a body are the same so that a closed loop can be obtained. As a first step, the domain around the geometry is built. After the maximum length in *x* and *y* directions are obtained by subtracting minimum values of *x* and *y* coordinates, from their corresponding maximum values, the maximum length, whether it is along *x* or *y* axis, is multiplied with the input outer size factor input to obtain the domain size is calculated. Since far-field boundary conditions are simply the free stream values that are mentioned in Chapter 3, it is important to set the outer boundary far away from the given geometry. Thus, a factor of 18 is taken as the minimum sufficient condition for this case.

The geometry is placed at the middle of the domain. The center of the root cell is determined according to the center of the geometry which is formed through the averaging of minimum and maximum *x* and *y* coordinates of the geometry. Using the domain size, the corners of the root cell are obtained.

After the creation of the root cell, the uniform mesh can be formed by dividing cells successively until the division level of the finest cells reaches the input uniform division level. At each cycle, levels of new formed cells are increased by 1, centers and corners of these cells are determined according to the location of its children place of their parent. Below, one can see the equations used for setting center coordinates of those cells,

$$x_c^{topLeft} = x_c^{parent} - \frac{d}{2^{l+1}} \qquad y_c^{topLeft} = y_c^{parent} + \frac{d}{2^{l+1}} \tag{2.01}$$

$$x_c^{topRight} = x_c^{parent} + \frac{d}{2^{l+1}} \qquad y_c^{topRight} = y_c^{parent} + \frac{d}{2^{l+1}} \tag{2.02}$$

$$x_c^{bottomLeft} = x_c^{parent} - \frac{d}{2^{l+1}} \qquad y_c^{bottomLeft} = y_c^{parent} - \frac{d}{2^{l+1}} \tag{2.03}$$

$$x_c^{bottomRight} = x_c^{parent} + \frac{d}{2^{l+1}} \qquad y_c^{bottomRight} = y_c^{parent} - \frac{d}{2^{l+1}} \tag{2.04}$$

where *d* is the domain size and *l* is the level of the considered cell. In addition to the center calculations, the corners are also computed by using the division level and domain size. However, instead of center coordinates of the parent, the center of the considered cell is used. The calculations of corner coordinates of a cell can be expressed as follows:

$$x_{topLeftCorner} = x_c - \frac{d}{2^{l+1}} \qquad y_{topLeftCorner} = y_c + \frac{d}{2^{l+1}} \tag{2.05}$$

$$x_{topRightCorner} = x_c + \frac{d}{2^{l+1}} \qquad y_{topRightCorner} = y_c + \frac{d}{2^{l+1}} \tag{2.06}$$

$$x_{bottomLeftCorner} = x_c - \frac{d}{2^{l+1}} \qquad y_{bottomLeftCorner} = y_c - \frac{d}{2^{l+1}} \tag{2.07}$$

$$x_{bottomRightCorner} = x_c + \frac{d}{2^{l+1}} \qquad y_{bottomRightCorner} = y_c - \frac{d}{2^{l+1}} \tag{2.08}$$

The uniform mesh is generated with the above calculations and by setting the neighbors told at Section 2.1.2, connectivity. With the uniform mesh, a default resolution is obtained for the outer cells. The cells near the geometry are then refined by geometric adaptation.

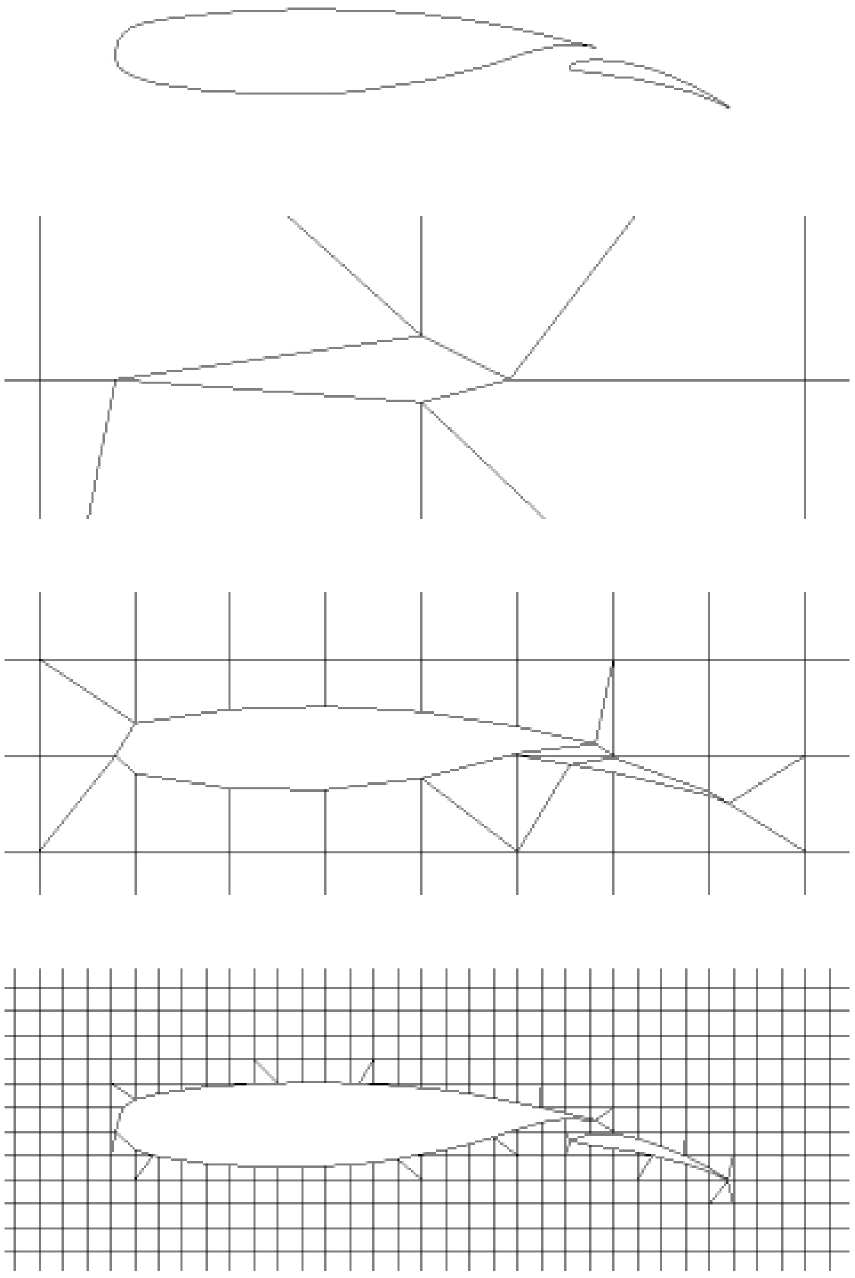


Figure 2.5 Uniform meshes around a two-element airfoil with 5, 7 and 9 cycles

It is important to obtain a sufficient resolution with the uniform mesh. If division number for uniform mesh is small, than the smaller geometries, especially flap or slat parts of a multi-element airfoil cannot be captured accurately. However, the geometric adaptation that will be applied after uniform meshing can solve this problem by refining the cells near the geometry. Yet, it is important to get a good resolution at the out cells to get accurate results. On the contrary, a very fine uniform mesh leads to a high number of cells so that the solution converges very slowly, since the number of cells doubles with one uniform mesh cycle. In the analyses, a uniform division level of 4 is used for most of the cases.

Figure 2.5 shows uniform meshes around two-element airfoil with 5, 7 and 9 cycles, without applying any geometric adaptation. In the mesh with 5 cycles, the flap is not captured totally and the main body is very different than the original one. As shown, increase in the number of cycles results in more accurate capture of the given geometry. However, cell number increases excessively. In the above figure, since comparison between uniform meshes is done, no geometric adaptation is applied. As a result, a high level of uniform mesh generation is needed in order to capture the geometry accurately.

2.2.2 Cell Type Determination

Type determination is crucial in Cartesian grid for capturing the geometry accurately, refining the critical cells near the geometry and multigrid application. While determining the types of cells, a number of steps are applied sequentially. First, it is determined whether the corner of the cell is inside or outside the geometry by using the Ray-Casting technique. Then, intersection points are found and sorted. As a result, the type of the cell is determined roughly. According to sorted intersection points and in-out indices of corners, square and split indices of the cell are set. By using all of this information, the final type of the cell is determined.

2.2.2.1 Corner Index Determination

Each corner has an index in the developed code. This index determines whether the point is inside or outside the geometry. By determination of all corners of a cell, the type of the cell can be determined roughly. In other words, if all corners are outside the geometry, then the cell is an out-cell. If they are inside the given geometry, the cell type is set to an in- cell. If all corners are neither outside nor inside, than the type of the cell may be a cut or a split cell.

While determining this index, there are two common techniques, namely winding number method and ray-casting method. Ray-casting method has numerous advantages compared to the winding number method. First, the winding method works by considering all the line segments of the geometry. However, it is not required to visit all segments in ray-casting. It is sufficient to consider only the line segments that the considered point is between its start and end nodes in *y* direction. Moreover, unlike the winding method, round-off errors of the floating points do not harm ray-casting method [23]. Due to its advantages, Ray-Casting method is chosen for inside-outside determination.

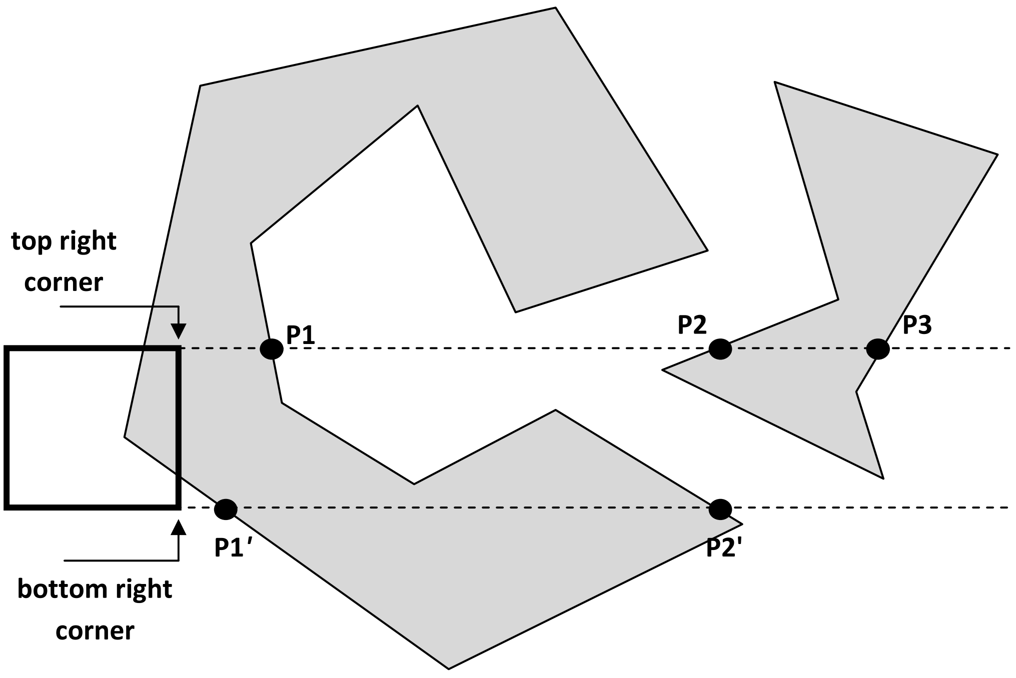


Figure 2.6 Ray-casting method

In the ray-casting method, there is a restriction that the bodies to be examined must be closed loop. Since the given geometry is formed with closed-loop bodies, this method is suitable for the developed code. In this technique, a ray is casted from a point along *x* direction generally. If this point intersects the given geometry odd number times, then this point lies in the geometry. On the other side, if an even number is found with the intersections of the cell with the geometry, then the point must be outside the geometry. No matter how many bodies there are in the domain, this method works successfully. Figure 2.6 summarizes the ray-casting method with an example of a cell around a geometry formed by two bodies whose one corner is inside and one corner is outside the geometry. The index of corners inside the geometry is set to -1, whereas for outside corners, the index is set to 1.

2.2.2.2 Square and Split Indexes

After ray-casting method is applied and the corner indexes are determined, intersection points are found by considering horizontal and vertical edges separately. In the developed code, intersection points are stored with the help of "intersections" vector and this vector has an object of user-defined class "IntPt". This class has also some stored variables. These are two doubles for x and y coordinates, one string for its location. The location of a point may be on the edges or on a corner. After finding of coordinates and locations of these intersection points, they should be sorted according to an order. This order is significant in order to be able to finalize the true type of the cell. Sorting of those points is started from the right edge and continues in a counterclockwise direction.

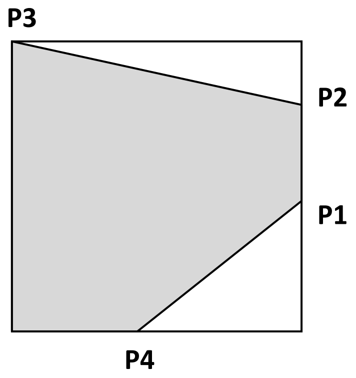


Figure 2.7 Illustration of sorting intersection points

The next step after sorting is setting the square index of the cell. The integers from 0 to 3 are first assigned to the corners starting from the bottom right corner and continuing in counterclockwise direction. In other words, this integer is 0 for bottom right corner, while it is 3 for bottom left corner. The square index is then found by summing two raised to the power of the index of that corner (an integer 0 to 3) for all corners whose in-out index is -1. In the below figure, an example is given for finding square index of a cell. The gray region indicates the part inside the geometry.

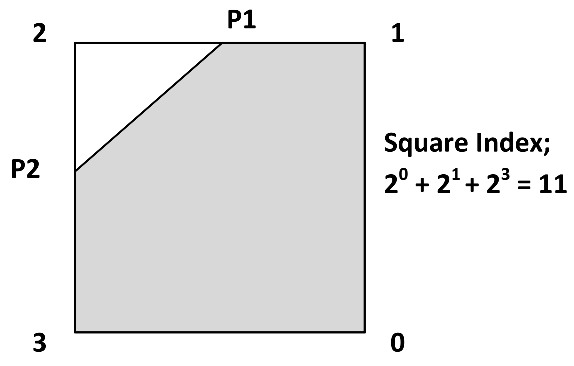


Figure 2.8 Determination of square index

The bottom right, top right and bottom left corners are inside the geometry. If two is raised to the power of their corresponding indexes 1, 2 and 8 are obtained, respectively. If they are summed up, total square index of this cell is found as 11. If all the corners are inside the geometry, the square index can be calculated as 15 by using this relation. On the contrary, for out cells whose corners have an in-out index of 1, the square index is set to 0 [21].

For all leaf cells in a Cartesian mesh, square indices are calculated by this way. Then using the intersection vector and square index, the general type of the cell can be determined. If a cell has no intersection point and has a square index of 0, then this cell must be an out cell. On the contrary, if a square index of 15 is assigned to a cell whose intersection vector is empty, then type of this cell is set to an in cell. If a cell has an intersection vector whose size is 1 or 2, and all corners of the cell has an index of neither -1 nor 1, then type of the cell is assigned to a "cut". For the other situations, except when the number of intersection points is greater than 4, the type of the cell is set to "split". Finally, for the exception case, the type is assigned as "notDefined". notDefined type is assigned to cells which cannot be considered in the other types. The code does not give an error if these are computational cells. If not, then there is a need to modify the mesh generation with some input change.

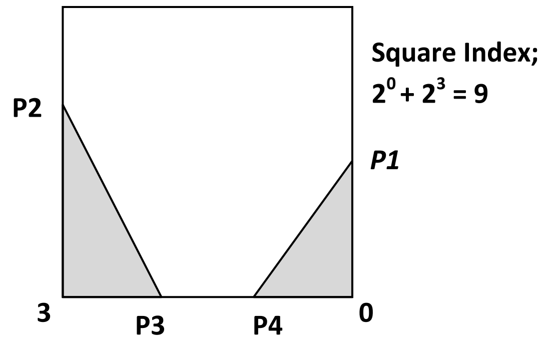


Figure 2.9 Example to a split cell

The cells having 4 intersection points are assigned to split cells. For a special case, the intersection point number may be 2 if both intersection points are on the same edge. While one or more corner has an in-out index of -1, the square index calculation can be done similar to the calculation above, as can be shown in Figure 2.9. However, it may be possible to have split cells whose all corners are either outside or inside the given geometry. Therefore, additional minus square indexes are assigned to these cells, as indicated in Figure 2.10.

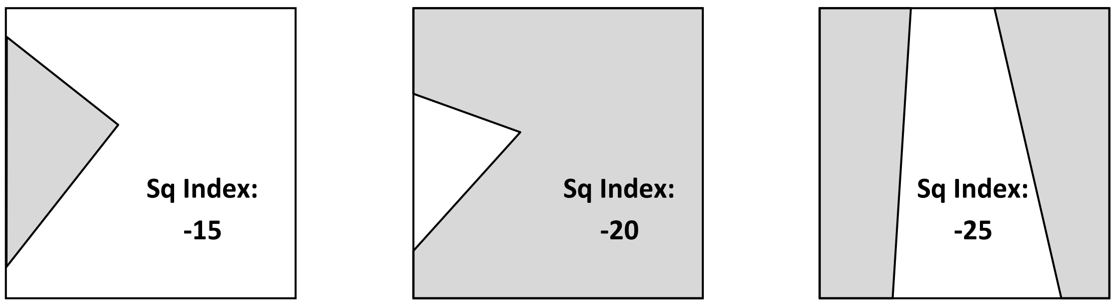


Figure 2.10 Split cells having minus square indices

In addition to square index, one more index should be used to understand the shape of the cell exactly. This index is called "split index". For one square index of a type, there are various alternatives that a cell has. While some cut cells may have 4 different alternatives, it may be increased to 18 for a split cell. These alternatives may come from the intersection points at the corners since the location variable of an intersection point is changed for a corner point. Alternatives of a cut cell having a square index of 6 is indicated in Figure 2.11.

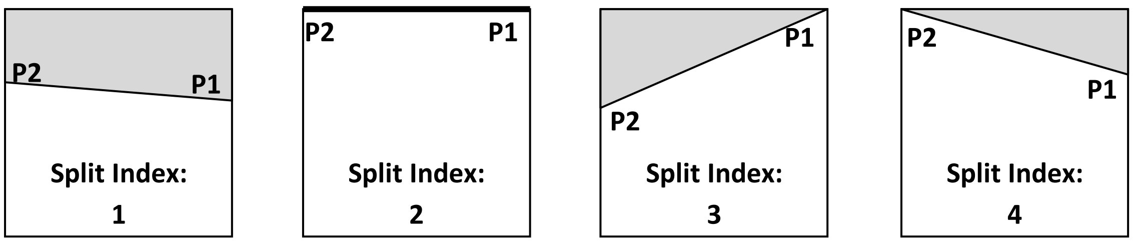


Figure 2.11 Alternatives of a cut cell having a square index of 6

Moreover, these may arise from intersection points on different edges for a split cell. In Figure 2.12, two alternatives of a split cell are shown with a square index of 6. Since the other alternatives have the same logic as the cut cell above, in other words, corner intersection points create the other alternatives, they are not illustrated.

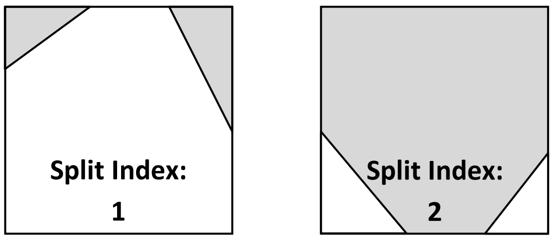


Figure 2.12 Two alternatives of a split cell having square index of 6

All alternatives that a cell may have are shown explicitly in Appendix A.

2.2.2.3 Split Cells Having Two Control Volumes

Some split cells have two separate control volumes like the cell having a split index of 2 in Figure 2.12. When these cells are encountered, two different cells are created, stored with the cell type pointers of "splitToCut1" and "splitToCut2". Each control volume is converted to a cut cell and all the calculations are carried out using these new cells [21]. The inverse relation among these cells is provided with the "inclusiveOfSplits" pointer. This word points the cell having those splitToCut cells. This relation is necessary especially for multigrid applications since coarsening is required which is discussed in Chapter 4 in detail. In addition, it is important to pay attention to order of intersection points for new cells. In Figure 2.13, the conversion of a split cell into two cut cells is illustrated.

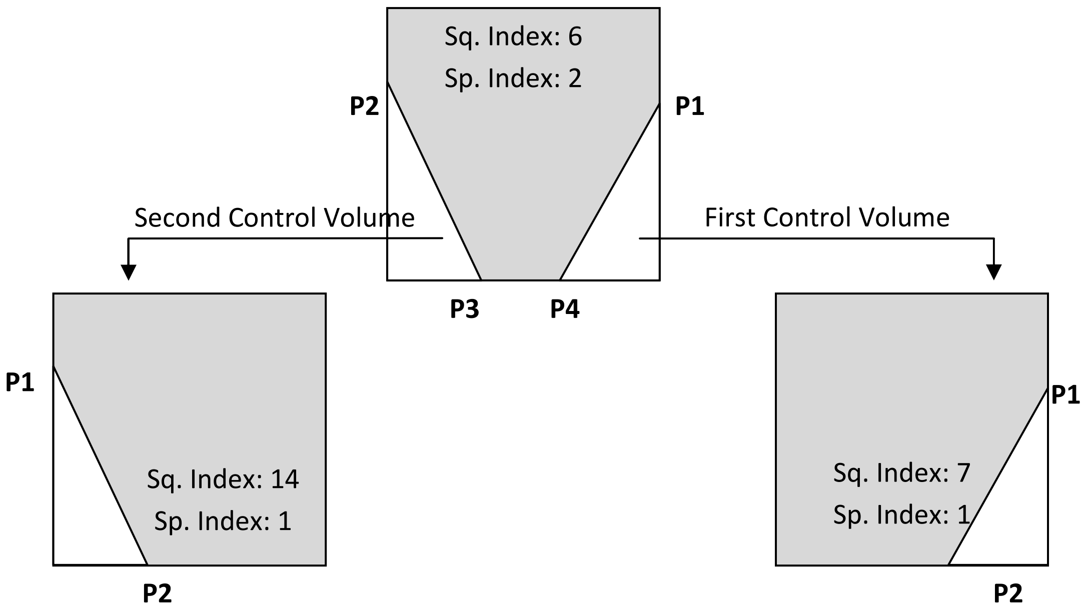


Figure 2.13 Conversion of split cell into two cut cells

2.2.3 Geometric Adaptations

Geometric adaptation allows high resolution grids around the input geometry. Three different adaptations can be applied to the uniform mesh, sequentially. First, box adaptation is applied to the mesh. Then, cut and split cells around the input geometry can be refined more cut and split adaptation. Finally, highly curved parts can become finer with curvature. The amount of these adaptations can be controlled by inputs.

2.2.3.1 Box Adaptation

In box adaptation, a rectangular box is first determined around the given geometry, the size of which is specified by the user. The size of box is specified with two inputs, boundary size factor in x and y coordinates. With this factor, x and y coordinates of the box can be found using the maximum and minimum coordinates of the whole geometry. The distance between the body and box can be found using the following relations;

$$D_x = (k_x - 1)\frac{x_{max} - x_{min}}{2} \tag{2.09}$$

$$D_y = (k_y - 1)\frac{y_{max} - y_{min}}{2} \tag{2.10}$$

where k_x and k_y are boundary size factors in x and y directions, respectively, subscripts "max" and "min" represent the maximum and minimum coordinates among all bodies in the geometry, respectively.

After that, the cells in this box are refined to the desired level until the desired resolution around the geometry is obtained. This desired level is controlled with an input of body division factor. Maximum body dimension, which is either on x axis or y axis, is multiplied by this factor. If the sizes of cells are larger than this determined size by the multiplication, then the cells in the box are refined. The following relation is used as the main criteria;

$$If\ \left(d_{max} n \le \frac{D}{2^l}\right)\ Do(Refine) \tag{2.11}$$

where D is the domain size, l is the level of the cell with minimum size in the domain, n is the body division factor and d_{max} is expressed as;

$$d_{max} = \begin{cases} x_{max} - x_{min} & (y_{max} - y_{min}) < (x_{max} - x_{min}) \\ y_{max} - y_{min} & (y_{max} - y_{min}) \geq (x_{max} - x_{min}) \end{cases} \quad (2.12)$$

The mesh after the application of box adaptation is shown in Figure 2.14.

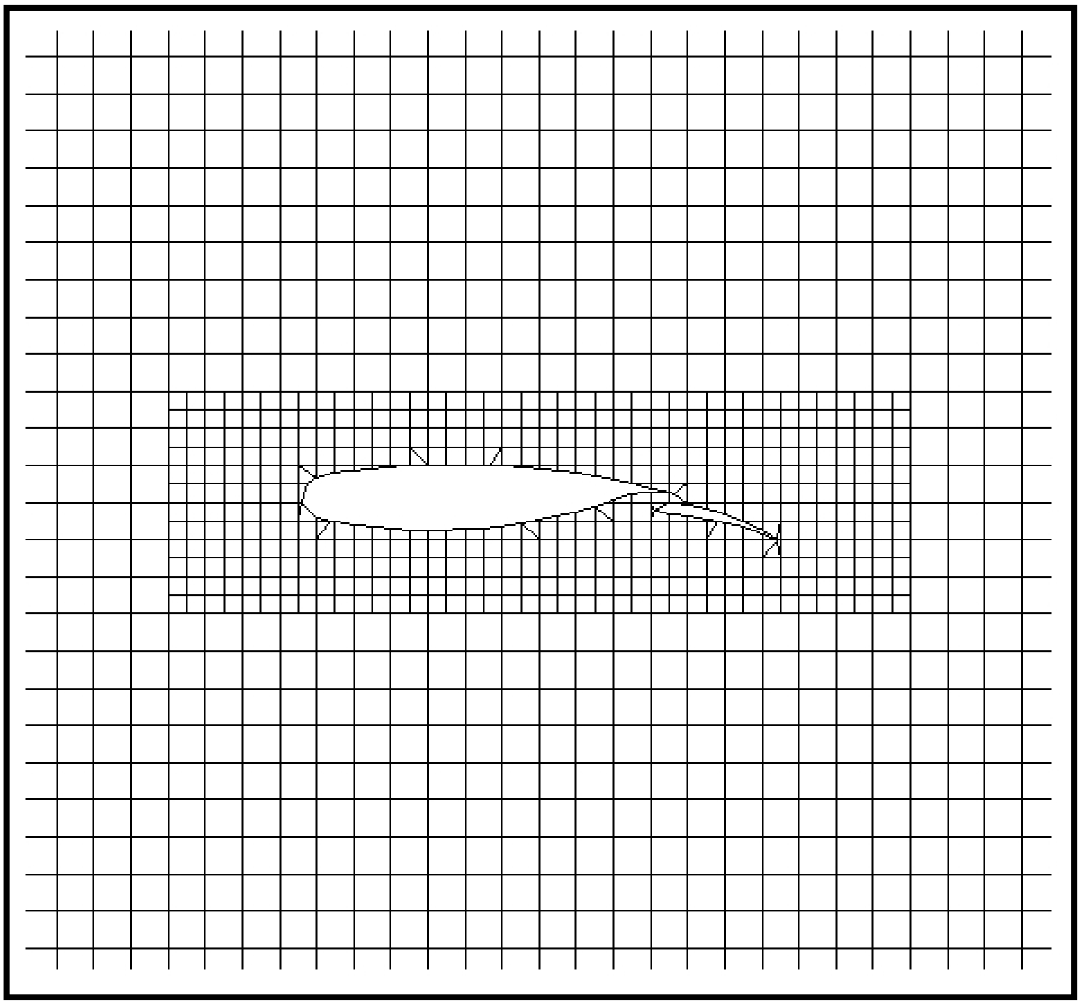


Figure 2.14 Box adaptation around a two-element airfoil

2.2.3.2 Cut-Split Adaptation

Since the cells near the given geometry can be small enough to get accurate results, these can become finer by the use of cut-split adaptation. As it can be understood from the name of the adaptation, cut and split cells are considered. In addition, the neighbors of these cells are also refined even if they are out cells. As a result, a smooth resolution around the geometry is obtained [21]. The user can specify the number of cycles that should be applied in this adaptation according to the desired level. In Figure 2.15, one cycle of cut-split adaptation is illustrated after the application of box adaptation to a two-element airfoil, NLR 7301.

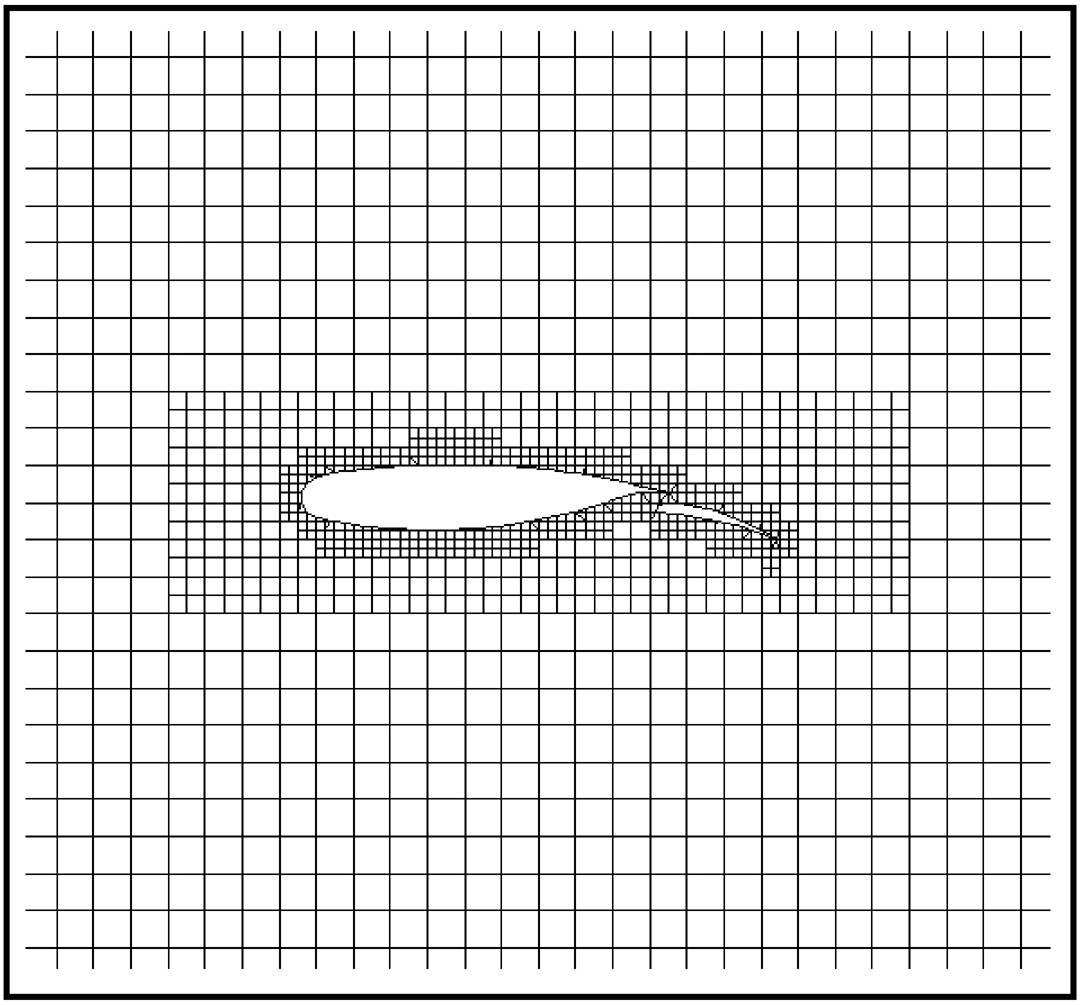


Figure 2.15 Cut-split adaptation around a two-element airfoil

2.2.3.3 Curvature Adaptation

Some regions of the geometry have highly curved parts. In these parts, there may be shear layers, vortices, wakes and similar events like these. Therefore, more resolution is required at these locations and this is provided by curvature adaptation.

In curvature adaptation, two neighboring cells near the wall boundary are considered. If the curvature formed by these geometrical parts in these cells is large enough, then these two cells are refined once in one cycle. The amount of curvature is determined by the angle between the intersection lines of the cells [22]. Since this angle is found by a triangle formed by three different intersection points at two cells, this angle is always less than 180 degrees, sometimes directing the outside of the geometry, sometimes inside of the geometry. Two examples illustrating these two different cases are shown in Figure 2.16, respectively. Note that the gray parts represent the geometry and T1, T2 and T3 are the corners of the triangle which is used to determine the curvature angle, θ_{curv}, by cosine theorem.

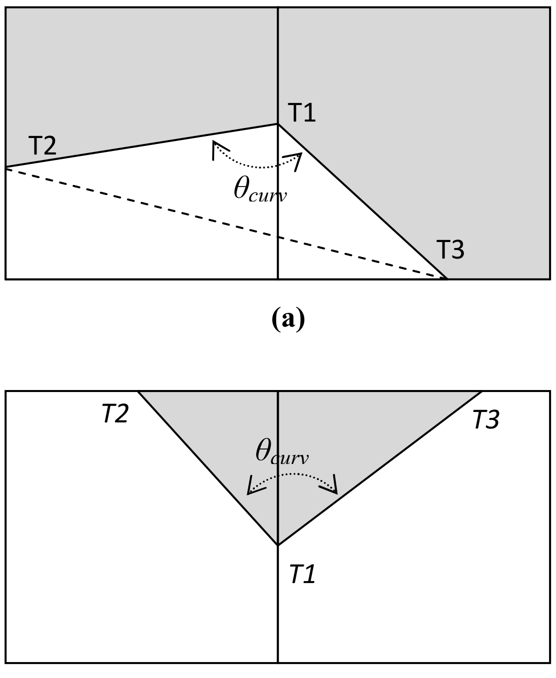


Figure 2.16 Curvature angle determination directing outside (a) and inside (b) of the geometry

After determining the curvature angle, it is compared with a threshold angle which is set by the user. If this angle is less than this specified threshold angle, then both of the cells are refined. Moreover, the cycle of the adaptation can be controlled by another input. Below, curvature adaptation is illustrated after applying box and cut-split adaptations to uniform mesh around NLR7301.

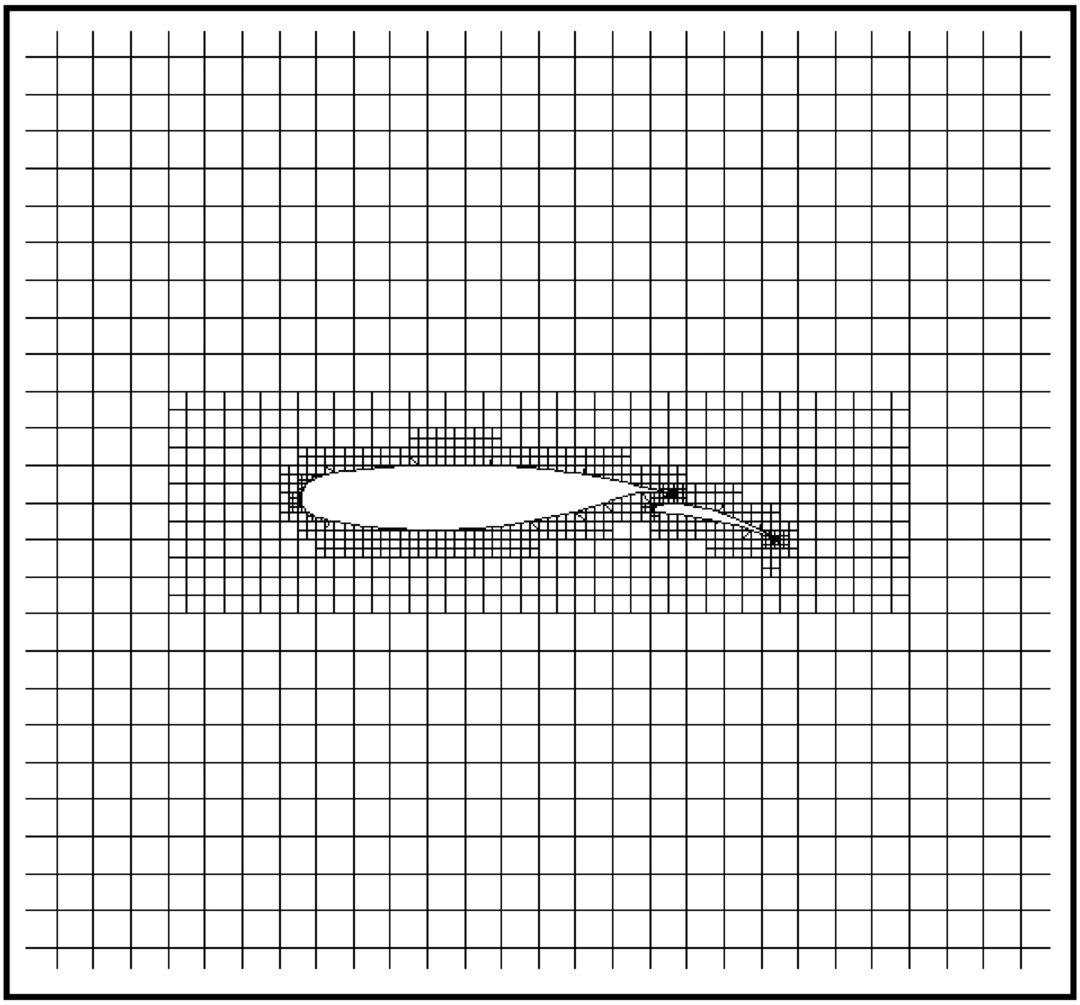


Figure 2.17 Curvature adaptation around a two-element airfoil

2.2.3.4 One Level Rule

One level rule sets the level difference between two neighborhood cells to 1 at maximum. This rule is provided to avoid the complexity of data structure and to facilitate the connectivity handling [21]. During the flux computations, reconstruction scheme can easily be applied to a mesh generated with the help of one level rule. The grid smoothness is also provided by this rule. In the geometric adaptations, which are discussed above, only the single cycle of each adaptation is shown since this rule is not yet introduced. For more cycles of a geometric adaptation, the grid becomes smoother with the aid of the one level rule. In Figure 2.18, the level difference between the uniform mesh and cells in the adapted box exceeds unity. The cells between box and uniform mesh are refined according to the one level rule. Moreover, cut cell adaptation cycle is set to 2 and curvature adaptation cycle is set to 4 in this example. Therefore, the effect of the one level rule can clearly be seen in the cells neighboring to the adapted cells and this is illustrated in Figure 2.19.

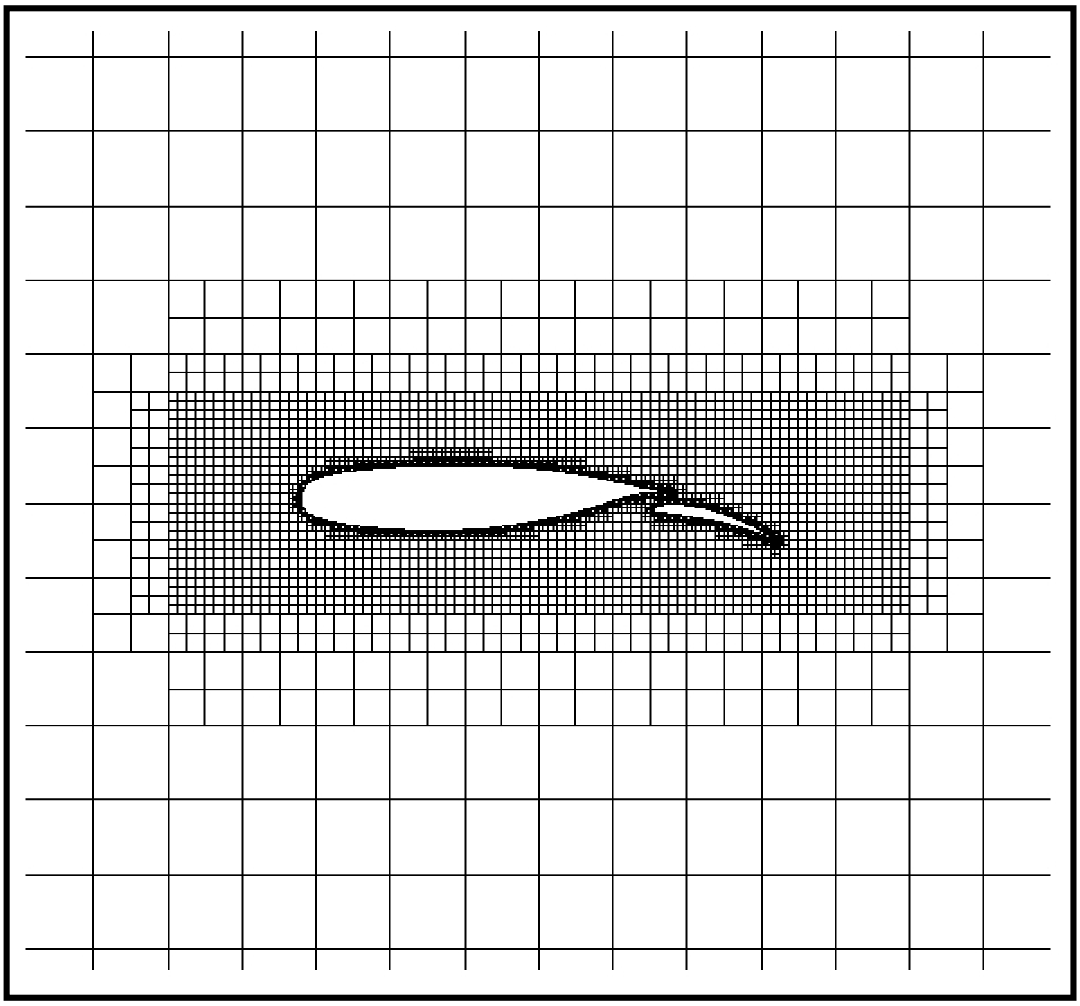


Figure 2.18 One level rule

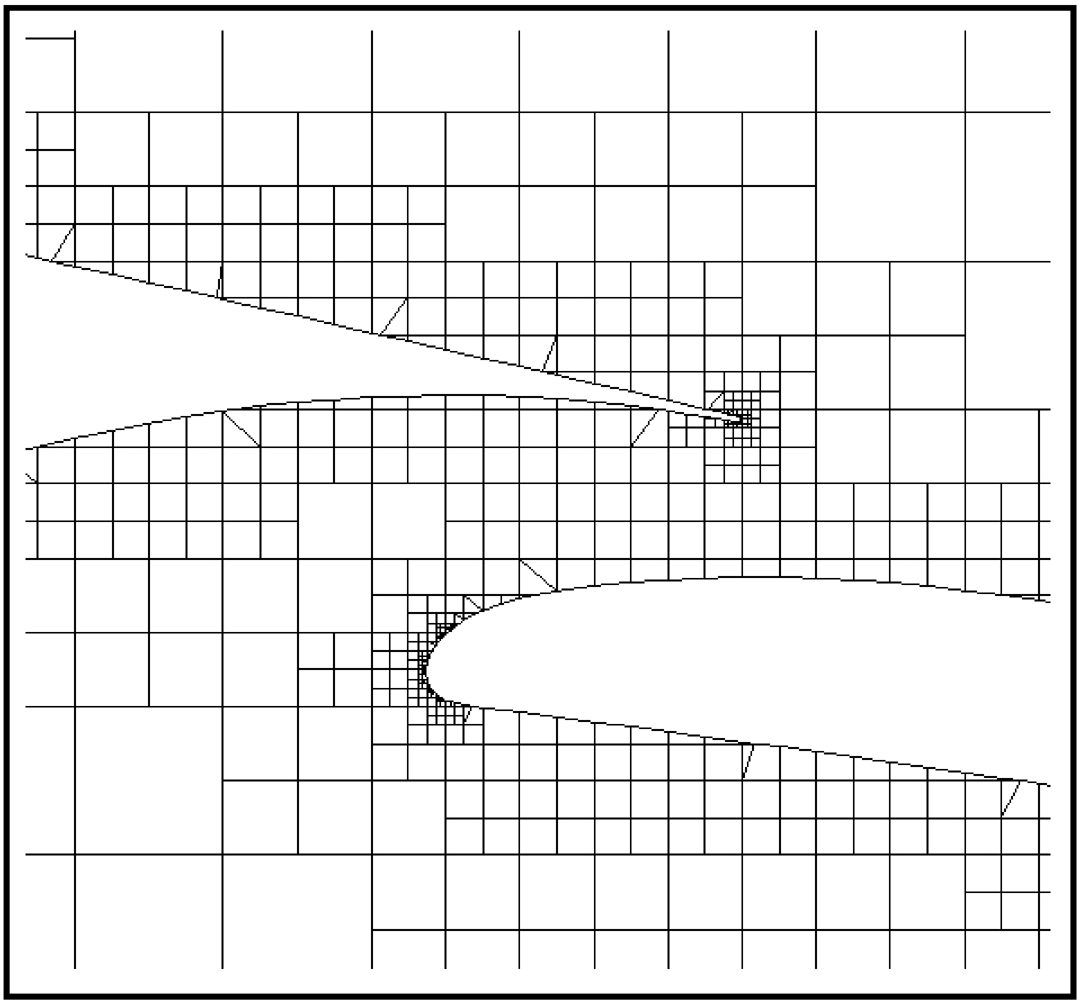


Figure 2.19 Closer look to the geometry to illustrate one level rule for cut-split and curvature adaptations

2.3 QUAD GRID GENERATION

For viscous flows, in order to obtain sufficient resolution in the boundary layer, quad grids can be used optionally. Before the generation of quad cells, the geometry is first puffed up by a specified amount. This puffed geometry becomes the geometry input for the Cartesian meshing and Cartesian cells are generated outside this puffed geometry. In the space lying between the original geometry and the puffed geometry, quad cells are created.

2.3.1 Boundary Layer Setting

The boundary layer thickness is set according to Reynolds number of the flow. Using the following relation, thickness, δ, can be determined for laminar flows [7].

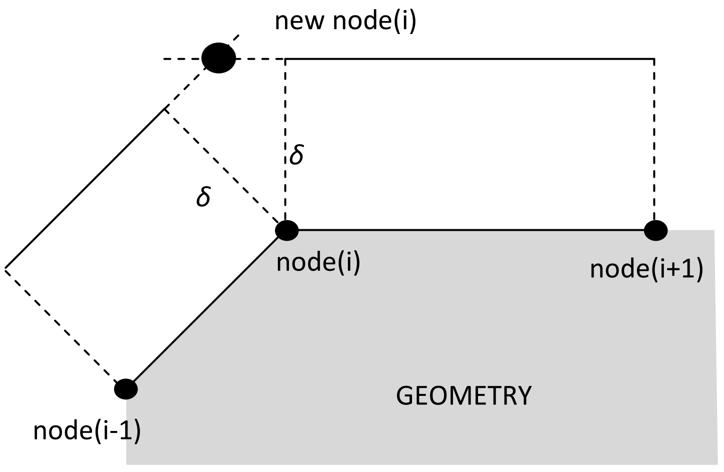
$$\delta = \frac{5}{\sqrt{Re}} \tag{2.13}$$

For the turbulent or separated with a thicker boundary layer, this thickness is multiplied by a factor which is greater than 1 can be multiplied with the thickness. Whereas the developed code considers only laminar flows, the thickness found by using the relation above can also be multiplied by a factor as a safety factor.

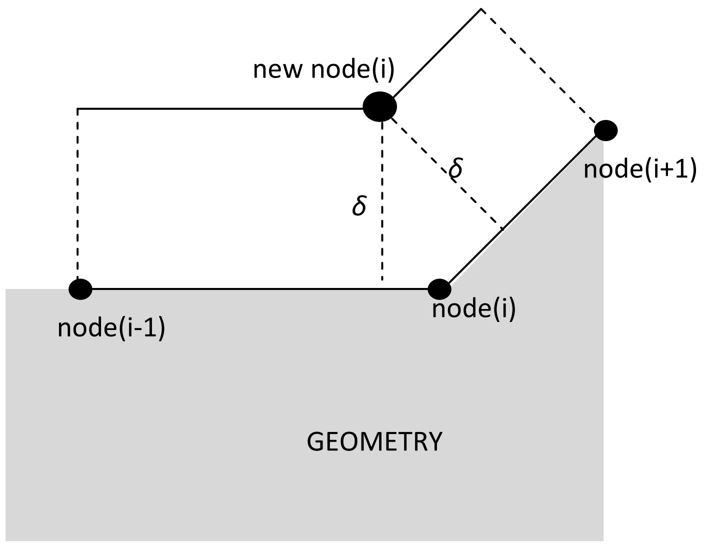
While setting the boundary layer, some corrections may be required to puff the geometry up correctly. Highly curved parts must be handled so that thickness has the same quantity at all points. In addition, negative volumes should be eliminated, which can be formed at some concave surfaces. After the puffing up process, quad cells are generated according to the Cartesian cells near the puffed geometry.

2.3.1.1 Setting Puffed Geometry

The geometry is specified with the nodal points, as mentioned earlier. While setting puffed geometry, line segments which are formed by two consecutive nodes are used. After forming a line segment, starting and end point for new line segment can be created by shifting the nodal points along the normal direction of the line segment by an amount equivalent to the determined boundary layer thickness. After forming all new line segments, the location of new puffed nodes are found by intersecting the two consecutive new line segments. Below, one can see two examples about the creation of new puffed node. While elongation of line segments are required to obtain the intersection point at the first one, shortening of new line segments is necessary for the second one.



(a)



(b)

Figure 2.20 Creation of a new node for puffed geometry

2.3.1.2 Handling of Highly Curved Parts

As it can be seen in Figure 2.20, new line segments must be elongated or shortened in order to find the new puffed node. While, the thickness between the original line segment and the puffed line segment is the same, the shortened or elongated part of line segment has a smaller or greater amount of thickness, respectively. Although this is not a problem for slightly curved parts, it leads to excessive amount of thickness for highly curved parts at convex regions of the geometry. This situation is illustrated at the trailing edge of the airfoil in Figure 2.21.

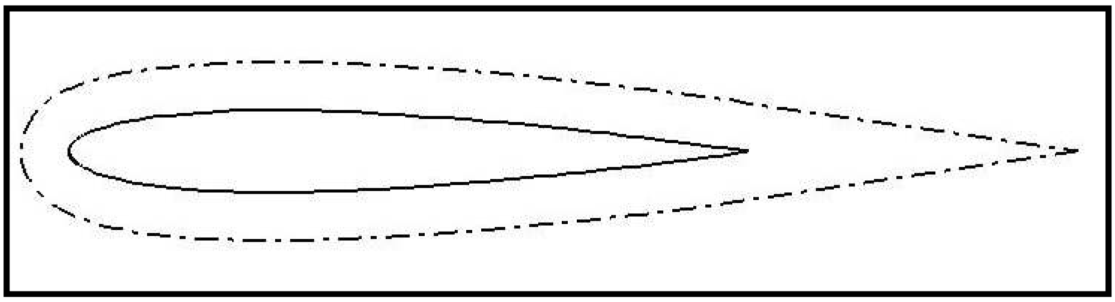
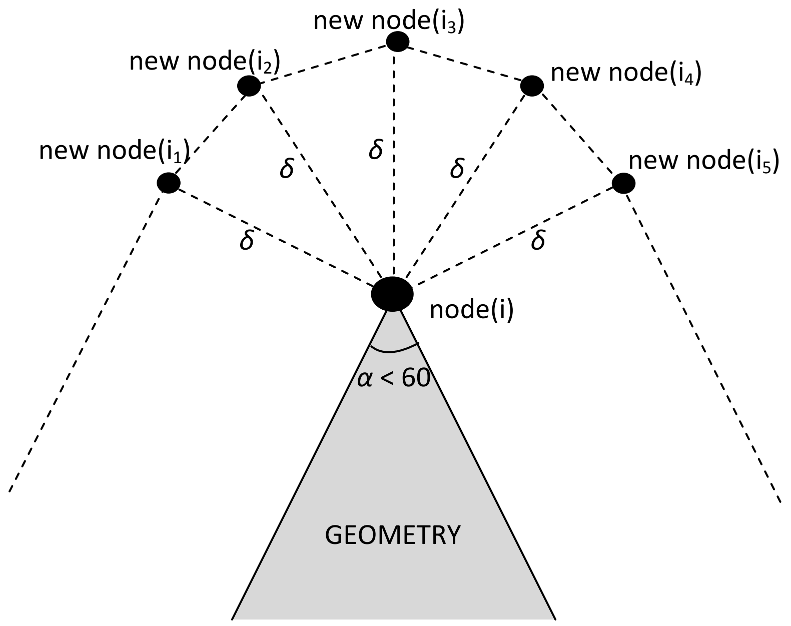
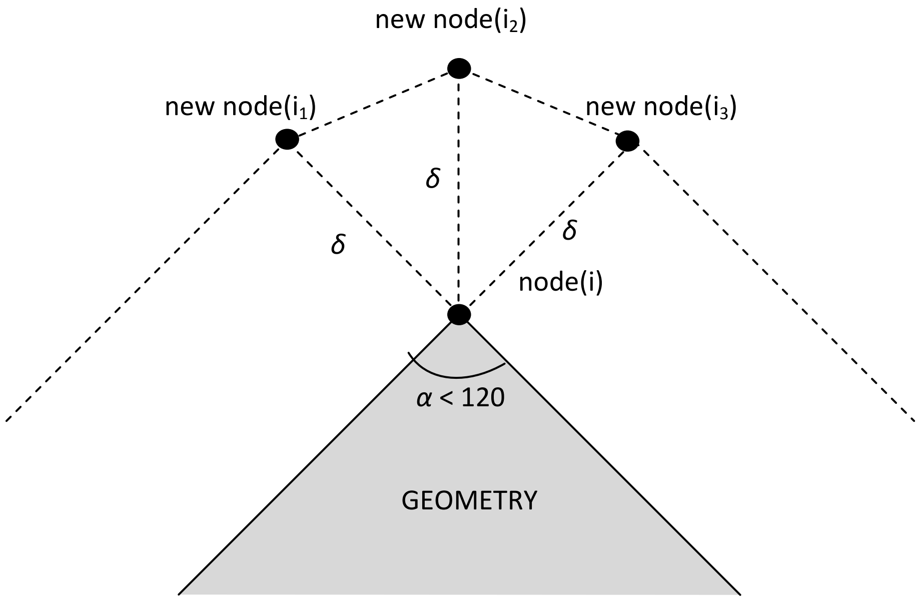


Figure 2.21 Original and puffed geometry without handling convex parts



(a)



(b)

Figure 2.22 Creation of a puffed node for highly curved part

In order to get a good puffed geometry at convex parts, the node is shifted not only to one location but also to several other locations by the boundary layer thickness, as shown in Figure 2.22. The number of these locations is determined by the angle of the convex part. If this angle is less than 60 degrees, five different nodal points are created from the original node of the geometry. If the angle is between 60 and 120 degrees, three different nodes are created. In this case, these three nodes are sufficient for obtaining uniform thickness at all points of the boundary layer. If the angle is greater than 120 degrees, one puffed node is sufficient since the curved region is not sharp enough. Figure 2.22 illustrates the two situations having an angle less than 60 degrees and between 60 and 120 degrees at the convex parts.

By shifting the nodal points to several different locations for convex parts, the puffed geometry shown in Figure 2.21 can be modified to the one indicated in Figure 2.33.

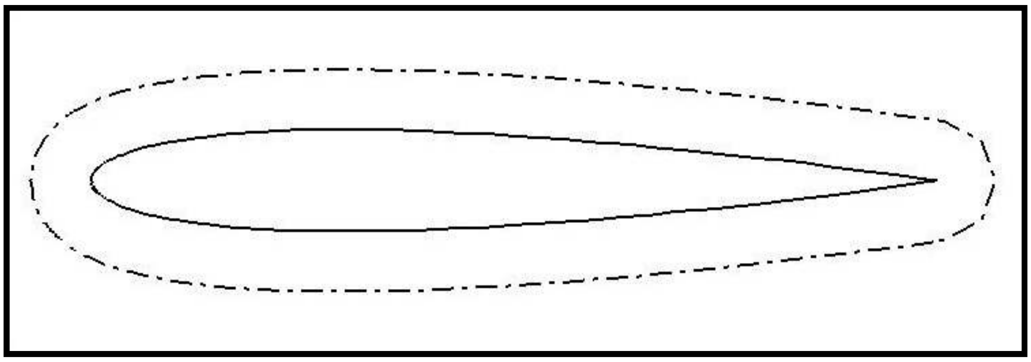


Figure 2.23 Original and puffed geometry with handling convex parts

2.3.1.3 Negative Volume Elimination

On some concave surfaces, it is possible to have negative volumes by direct extrusion of the geometry. The reason is that the new line segments are formed by connecting wrong nodes after shifting of two consecutive line segments at a certain amount, resulting in negative volumes. This situation can be exemplified in Figure 2.24.

In order to eliminate negative volumes, the intersection point of the intersected line segments is accepted as the new puffed node. However, this may cause new negative volumes at the parts near to the fixed region. Therefore, elimination of negative volumes using this method is continued until none of the line segments intersect each other. After elimination process, the airfoil shown in Figure 2.24 has a good puffed geometry, as shown in Figure 2.25.

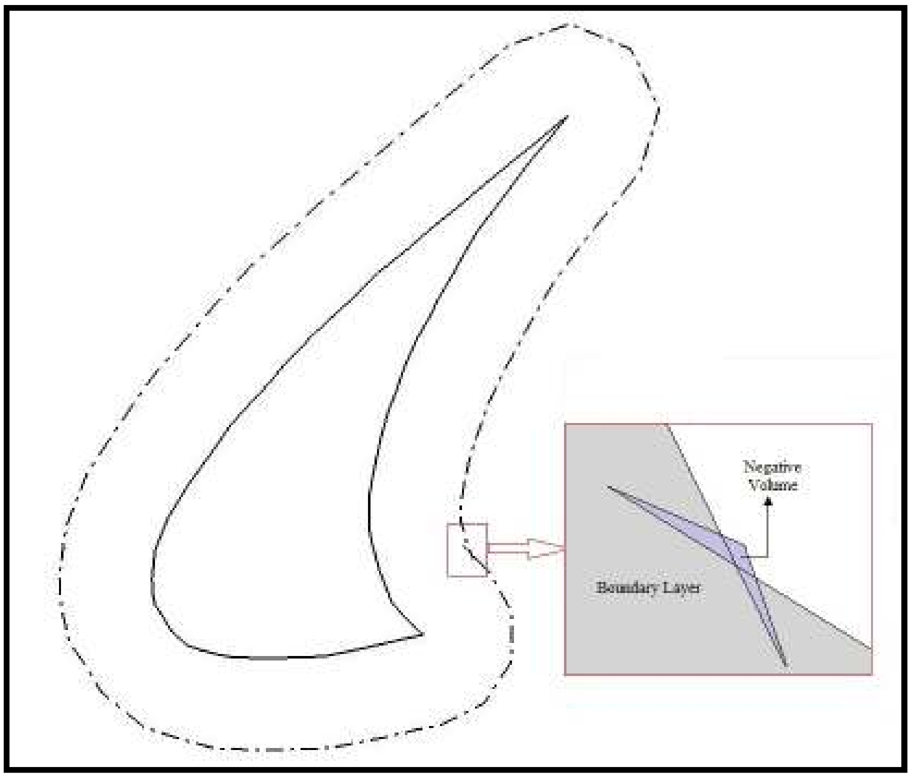


Figure 2.24 Negative volume at concave region

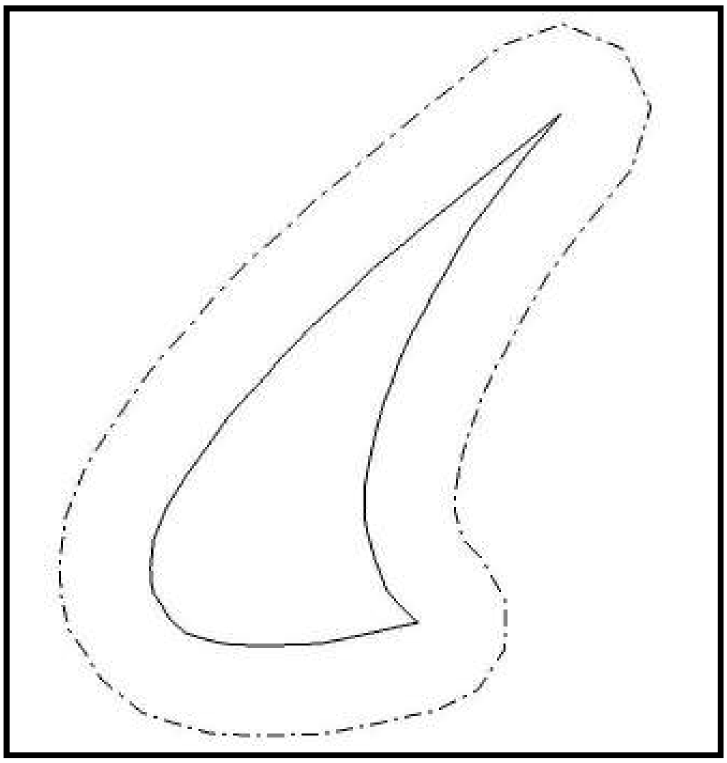


Figure 2.25 Boundary layer after elimination of negative volumes

2.3.2 Quad Cell Generation

After setting the boundary layer using the puffed geometry, the spacing between two geometries is filled with quad cells. Quad grids are connected to the Cartesian cells formed outside the boundary layer. With the connectivity information, a smooth hybrid grid can be generated.

While generating quad grids, two inputs are used, namely row number and stretch factor. The number of rows in the boundary layer can be specified by user. The thicknesses of the quad cells are determined according to the stretch factor which is the ratio between quad cells at two consecutive rows.

2.3.2.1 Connectivity

After row number and thicknesses of quad cells are set, the quad cells can be generated with the connection to the Cartesian cells. The connection between quad cells and Cartesian cells are provided with two pointers, "quad1" and "inclusiveOfQuads". A Cartesian cell may have a quad cell and if it has, then this quad cell is stored at "quad1". The opposite connectivity relation is obtained by "inclusiveOfQuads" pointer. With this technique, the corner points are forced to coincide with the intersection points of the Cartesian cell. In other words, interpolation of the flow variables from the Cartesian cells to the quad cells is not necessary since fluxes can simply be calculated along the common faces.

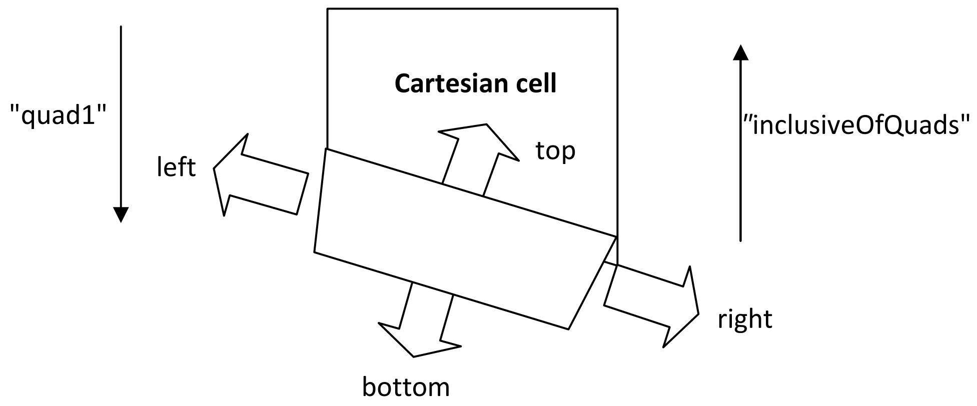


Figure 2.26 Relation between a quad cell and a Cartesian cell

Sometimes, a Cartesian cell may have two faces neighboring to the puffed geometry, which possesses one control volume. Then, a second pointer, "quad2", is used for these situations. While first quad cell is stored at "quad1", "quad2" is used for the second one. On the contrary, it is sufficient to use one pointer, "inclusiveOfQuads", for the inverse relation, as described before.

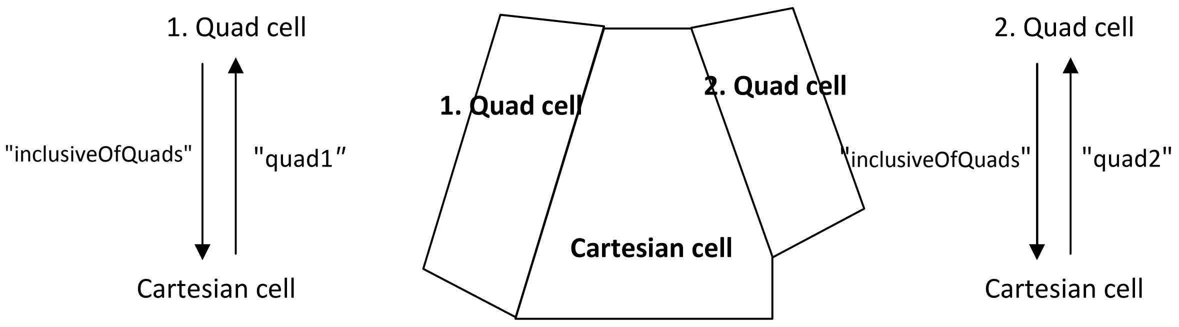


Figure 2.27 Relation between a Cartesian cell and its two quad cells

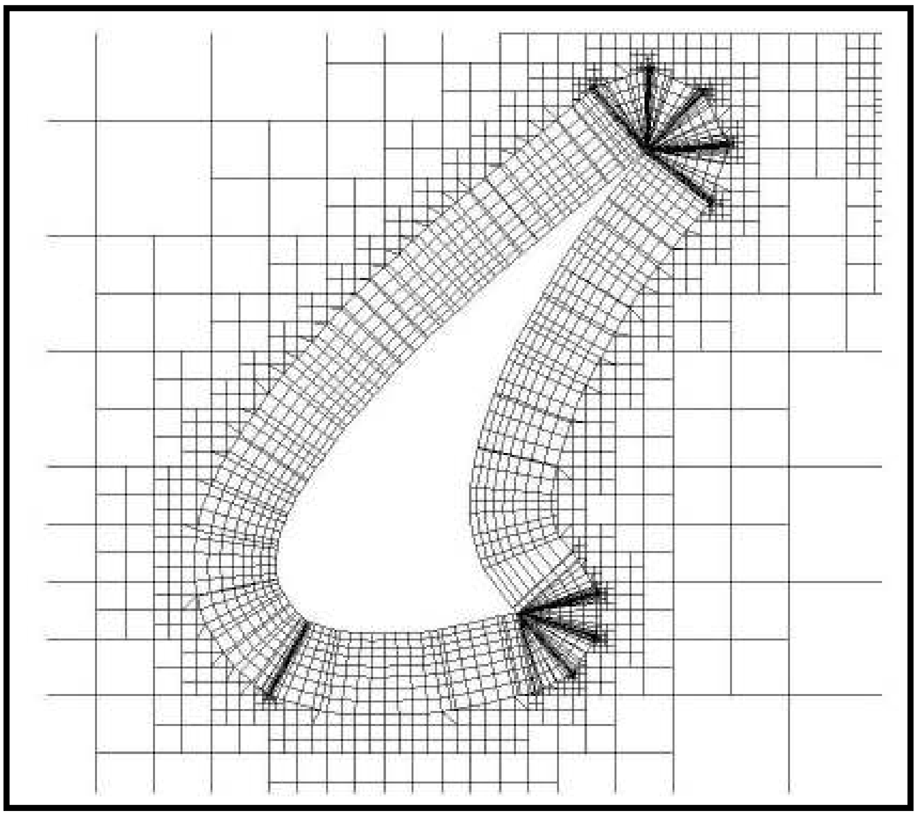


Figure 2.28 Hybrid mesh around slat of a three-element airfoil

After providing the connectivity between Cartesian and quad cells, the connectivity between quad cells is provided by the neighborhood information. As shown in Figure 2.26, the neighbors are found according to the specified directions. Thus, all cells including quad and Cartesian cells are connected to each other. In Figure 2.28, one can see an example to hybrid mesh around a slat of a three element airfoil.

It can be shown in Figure 2.28 that the size of the quad cells is very small near the smaller Cartesian cells. Although cut-split and curvature adaptations are applied only to Cartesian mesh, since smaller cells lead to smaller quad cells, quad cells become automatically finer at the highly curved regions.

It is also important to note that the quad cells are not refined directly during the solution adaptation, which is discussed in Chapter 3.7. If a region where a quad cell exists needs to be refined through solution adaptation, the inclusive Cartesian cell of this quad cell is refined at first. Then, quad cells of the Cartesian cell are deleted and new quads are regenerated according to children of the refined Cartesian cell. As a result, those regions become finer without refining quad cells. However, with solution adaptation, the number of rows is not changed.

CHAPTER 3

NUMERICAL SOLUTION

In this chapter, first of all, the governing equations are explained in detail. While these equations are presented, two dimensional Navier-Stokes equations in integral form are introduced. These equations are then non-dimensionalized with suitable reference values. Finally, wall and far-field boundary conditions are explained for both inviscid and viscous flows.

Secondly, the discretization of these governing equations is discussed. After spatial discretization is introduced, the temporal discretization is told by using multistage time stepping. Time step calculations are explained later while solving inviscid and viscous flows. Furthermore, a cut-back procedure for CFL number is described in order to avoid instability in earlier iterations of the execution of the code.

Thirdly, inviscid flux computations are mentioned. Flux vector splitting methods like AUSM and its derivatives are discussed. Furthermore, approximate Riemann solver of Roe is described.

Fourthly, reconstruction of the primitive flow variables are explained using the least squares method. To get more accurate result, this technique is used in some of the analyses. However, computational time increases as expected. In addition to this, to get more stable results, gradient limiting procedure is introduced.

Fifthly, viscous flux computations are discussed. One reconstruction technique is used while calculating viscous fluxes. In this viscous reconstruction technique, viscous flux is computed using both flow variables and gradients obtained by inviscid reconstruction at cell centroids.

Sixthly, how to calculate the coefficients of pressure and skin friction are presented. These coefficients are used to compare the results with the available data in the literature.

Finally, solution adaptation is discussed in detail. With this adaptation, the critical grids in the domain become finer so that more accurate solutions can be obtained.

3.1 GOVERNING EQUATIONS

Navier-Stokes equations are the governing equations for the flow around bodies. These equations can be in integral form or differential form. These equations are derived from the conservation of mass, momentum and energy. In the present code, non-dimensionalized Navier-Stokes equations are used in integral form, with appropriate wall and far-field boundary conditions.

3.1.1 Two-Dimensional Governing Equations in Integral Form

The general compressible integral form of these equations can be represented as;

$$\frac{\partial}{\partial t}\int_V \boldsymbol{Q}\,\mathrm{d}V + \int_S (\mathbf{F}\cdot\mathbf{n})\,\mathrm{d}S = \int_S (\mathbf{G}\cdot\mathbf{n})\,\mathrm{d}S \qquad (3.01)$$

In this equation, **Q** contains the vector of conserved variables of density, momentum and total energy. **F** is the inviscid flux vector while **G** is the viscous flux vector. **n** represents the unit vector in the normal direction to the differential area, d*S*. In two-dimensional Cartesian coordinates, the conserved variables vector, **Q**, inviscid flux vector, **F** and viscous flux vector, **G** can be represented as below.

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \tag{3.02}$$

$$\mathbf{F} = \begin{bmatrix} \rho u\,\mathbf{i} + \rho v\,\mathbf{j} \\ (\rho u^2 + p)\,\mathbf{i} + \rho uv\,\mathbf{j} \\ \rho uv\,\mathbf{i} + (\rho v^2 + p)\,\mathbf{j} \\ \rho uH\,\mathbf{i} + pvH\,\mathbf{j} \end{bmatrix} \tag{3.03}$$

$$\boldsymbol{G} = \begin{bmatrix} \mathbf{0} \\ \tau_{xx}\,\boldsymbol{i} + \tau_{yx}\,\boldsymbol{j} \\ \tau_{xy}\,\boldsymbol{i} + \tau_{yy}\,\boldsymbol{j} \\ (u\tau_{xx} + v\tau_{xy} - q_x)\boldsymbol{i} + (u\tau_{yx} + v\tau_{yy} - q_y)\,\boldsymbol{j} \end{bmatrix} \tag{3.04}$$

Since the unit normal vector can be defined using angle θ between the unit vector and x-axis, the dot products of inviscid flux vector and viscous flux vector with the unit normal can be written as;

$$\boldsymbol{F}\cdot\boldsymbol{n} = \begin{bmatrix} \rho u\,cos\theta + \rho v\,sin\theta \\ (\rho u^2 + p)\,cos\theta + \rho uv\,sin\theta \\ \rho uv\,cos\theta + (\rho v^2 + p)\,sin\theta \\ \rho uH\,cos\theta + \rho vH\,sin\theta \end{bmatrix} \tag{3.05}$$

$$\boldsymbol{G}\cdot\boldsymbol{n} = \begin{bmatrix} 0 \\ \tau_{xx}\,cos\theta + \tau_{yx}\,sin\theta \\ \tau_{xy}\,cos\theta + \tau_{yy}\,sin\theta \\ (u\tau_{xx} + v\tau_{xy} - q_x)\,cos\theta + (u\tau_{yx} + v\tau_{yy} - q_y)\,sin\theta \end{bmatrix} \tag{3.06}$$

Descriptions of the variables used in Equations (3.02) to (3.05) are as follows. ρ is the fluid density, u and v are the x and y components of the fluid velocity, respectively. p represents the fluid static pressure, E is the specific total energy while H is the specific total enthalpy. τ_{xx}, τ_{xy}, τ_{yx}, and τ_{yy} are the stresses. Finally, q_x and q_y represent the heat flux terms in x and y directions, respectively.

In order to be capable of solving the above equations, some additional relations are required. These relations are formed using thermodynamic relations and the perfect gas assumption. Following equations are used to close the system of equations.

$$p = \rho R T \tag{3.07}$$

$$e = c_v T \tag{3.08}$$

$$R = c_p - c_v \tag{3.09}$$

$$\gamma = \frac{c_p}{c_v} \tag{3.10}$$

$$E = e + \frac{u^2+v^2}{2} \tag{3.11}$$

In the above equations, R is the specific gas constant, c_p is the specific heat for a constant pressure, c_v is the specific heat for a constant volume, e is the specific internal energy, T is temperature and γ represents the specific heat ratio. Using these equations, specific total enthalpy and static fluid pressure can be expressed as;

$$H = E + \frac{p}{\rho} \tag{3.12}$$

$$p = \rho(\gamma - 1)\left(\rho E - \frac{\rho(u^2+v^2)}{2}\right) \tag{3.13}$$

Since the fluids used in this code are restricted to the Newtonian fluids, the viscous stresses are related to the laminar dynamic viscosity, μ, and the velocity gradients through the following relations.

$$\tau_{xx} = -\frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + 2\mu\frac{\partial u}{\partial x} \tag{3.14}$$

$$\tau_{yy} = -\frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + 2\mu\frac{\partial v}{\partial y} \tag{3.15}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{3.16}$$

In the above equations, since the fluid used in the analyses is air, laminar dynamic viscosity is calculated using Sutherland's law [25], which the viscosity is accurately related to the temperature.

$$\frac{\mu}{\mu_\infty} = \left(\frac{T}{T_\infty}\right)^{3/2}\left(\frac{T_\infty + 110.4}{T + 110.4}\right) \tag{3.17}$$

In this equation, μ_∞ is the reference dynamic viscosity at the temperature T_∞ which is taken as 273.15 K. In addition to these, the heat flux components are calculated using Fourier's heat conduction law

$$q_x = -k\frac{\partial T}{\partial x} \tag{3.18}$$

$$q_y = -k\frac{\partial T}{\partial y} \tag{3.19}$$

where k is the coefficient of thermal conductivity.

3.1.2 Non-Dimensionalization

The non-dimensionalization is advantageous since it prevents numerical errors resulting from the disparity in scale of the conserved variables [26]. Moreover, it decreases the number of parameters to be handled and ease the handling of the equations. In the present code, the governing equations are made non-dimensional with the suitable reference values as follows;

$$\begin{array}{lll} x' = \frac{x}{L_\infty} & y' = \frac{y}{L_\infty} & t' = \frac{t\,c_\infty}{L_\infty} \\ u' = \frac{u}{c_\infty} & v' = \frac{v}{v_\infty} & p' = \frac{p}{{c_\infty}^2} \\ \rho' = \frac{\rho}{\rho_\infty} & E' = \frac{E}{{c_\infty}^2} & \mu' = \frac{\mu}{\mu_\infty} \end{array} \tag{3.23}$$

where superscript (') denotes non-dimensional variable. With the use of non-dimensional Reynolds number, the governing equations can be made non-dimensional as shown below;

$$\int_V \frac{\partial \boldsymbol{Q}'}{\partial t'} dt' + \int_S (\boldsymbol{F}' \cdot \boldsymbol{n}') dS' = \frac{M_\infty}{Re_\infty} \int_S (\boldsymbol{G}' \cdot \boldsymbol{n}') dS' \tag{3.24}$$

where Re_∞ is the Reynolds number based on fluid velocity, V, and reference length, L_∞,

$$Re_\infty = \frac{\rho_\infty V L_\infty}{\mu_\infty} \tag{3.25}$$

and M_∞ is the free stream Mach number.

$$M_\infty = \frac{V}{c_\infty} = \frac{\sqrt{u^2+v^2}}{c_\infty} \tag{3.26}$$

Non-dimensional conserved variables and dot products of inviscid flux vector and viscous flux vector with the unit normal vector can be explicitly written as;

$$\boldsymbol{Q}' = \begin{bmatrix} \rho' \\ \rho' u' \\ \rho' v' \\ \rho' E' \end{bmatrix} \tag{3.27}$$

$$\boldsymbol{F}' \cdot \boldsymbol{n}' = \begin{bmatrix} \rho' u' \cos\theta + \rho' v' \sin\theta \\ (\rho' u'^2 + p') \cos\theta + \rho' u' v' \sin\theta \\ \rho' u' v' \cos\theta + (\rho' v'^2 + p') \sin\theta \\ \rho' u' \left(E' + \frac{p'}{\rho'}\right) \cos\theta + p' v' \left(E' + \frac{p'}{\rho'}\right) \sin\theta \end{bmatrix} \tag{3.28}$$

$$\boldsymbol{G}' \cdot \boldsymbol{n}' = \begin{bmatrix} 0 \\ \tau_{xx}' \cos\theta + \tau_{yx}' \sin\theta \\ \tau_{xy}' \cos\theta + \tau_{yy}' \sin\theta \\ (u'\tau_{xx}' + v'\tau_{xy}' - q_x') \cos\theta + (u'\tau_{yx}' + v'\tau_{yy}' - q_y') \sin\theta \end{bmatrix} \tag{3.29}$$

As it can be seen, non-dimensionalized inviscid flux vector is not different than the dimensional one. For inviscid solutions where the viscous flux vector is zero, there is no need to use any additional terms. However, the initial guesses should be appropriate.

As non-dimensional free-stream values, density is chosen as 1. Static pressure is chosen as 1/γ in order to equalize speed of sound to 1. With these initial guesses, it is not required to add new terms to non-dimensionalized equations [21]. The initial guesses are given below. The subscript "in" denotes the free-stream values.

$$\rho_{in}' = 1 \qquad p_{in}' = {}^{1}/_{\gamma} \qquad c_{in}' = \sqrt{\frac{p_{in}'\gamma}{\rho_{in}'}} = 1 \tag{3.30}$$

When considering viscous terms, non-dimensional stresses are very similar to dimensional ones. The ratio of the Mach number to Reynolds number is required for the conserved equations. However, heat flux terms are different than the dimensional ones since thermal conductivity is non-dimensionalized with another non-dimensional parameter, i.e. Prandtl number. Prandtl number can be defined as;

$$Pr = \frac{\mu c_p}{k} \tag{3.31}$$

With the use of Prandtl number and non-dimensional variables, heat flux components are made non-dimensional as shown below. Since gradient of pressure is calculated instead of temperature gradient in the code, the equations are revised according to this gradient.

$$q_x' = -\frac{\gamma}{(\gamma-1)Pr}\frac{\partial\left({}^{P'}/_{\rho'}\right)}{\partial x'} \tag{3.32}$$

$$q_y' = -\frac{\gamma}{(\gamma-1)Pr}\frac{\partial\left({}^{P'}/_{\rho'}\right)}{\partial y'} \tag{3.33}$$

From now on, superscript (') is not used for simplicity. The variables without any superscripts denote non-dimensional variables unless it is particularly mentioned.

3.1.3 Boundary Conditions

There are two types of boundary conditions for external flow. These are far-field boundary conditions and wall boundary conditions.

3.1.3.1 Far-Field Boundary Conditions

Far-field boundary conditions are used for the outermost cells in the domain. These conditions are applied at the faces not having any neighbors. Since in the analyses, far-field boundary is located at least 18 chords ahead of the analyzed airfoil, boundary conditions here are simply calculated using free-stream values, as shown below. These free-stream values are equated to the ghost cell which is created as a neighbor to the face having no real neighbor. Moreover, this ghost cell has the same size as the considered cell.

$$\rho_{ghost} = \rho_{in}, \qquad p_{ghost} = p_{in} \qquad c_{ghost} = c_{in} \tag{3.34}$$

Using these conditions, velocity components and specific total energy for far-field faces can be computed as;

$$u_{ghost} = M_\infty cos(\theta_{face}) \tag{3.35}$$

$$v_{ghost} = M_\infty sin(\theta_{face}) \tag{3.36}$$

$$E_{ghost} = \frac{p_{in}}{\rho_{in}(\gamma-1)} + \frac{{u_{ghost}}^2 + {v_{ghost}}^2}{2} \tag{3.37}$$

In the figure below, ghost cell of an outermost cell can be seen.

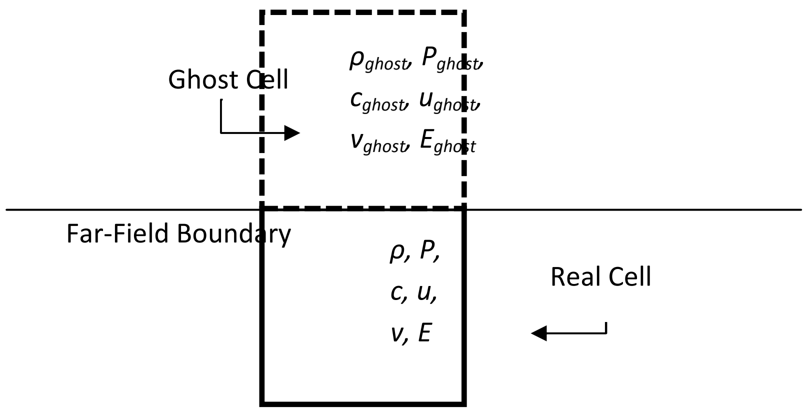


Figure 3.1 Far-field boundary conditions

3.1.3.2 Wall Boundary Conditions

Wall boundary conditions are used for the cells near the wall boundary. These cells are cut and split cells for inviscid flows, while they can be quad cells or cut and split cells for viscous flows depending on quad cell usage. The flux through the interface between the wall and fluid is calculated by using the ghost cell technique. The created ghost cell has same size as the real cell. Moreover, both for inviscid and viscous flows, pressure and density are taken as the same as the ones in the real cell. The velocity components on the interface are changed according to the flow type.

While solving inviscid flows, the velocity components at the interface of the real cell are found by using the normal angle. Then, the tangential velocity component on the interface of the ghost cell is taken as the same as the one in the real cell, whereas the normal velocity component has the same size as the one in the real cell, but it is in the opposite direction. With these velocity components, the cell-centered components of the velocity can easily be calculated using the face normal angle.

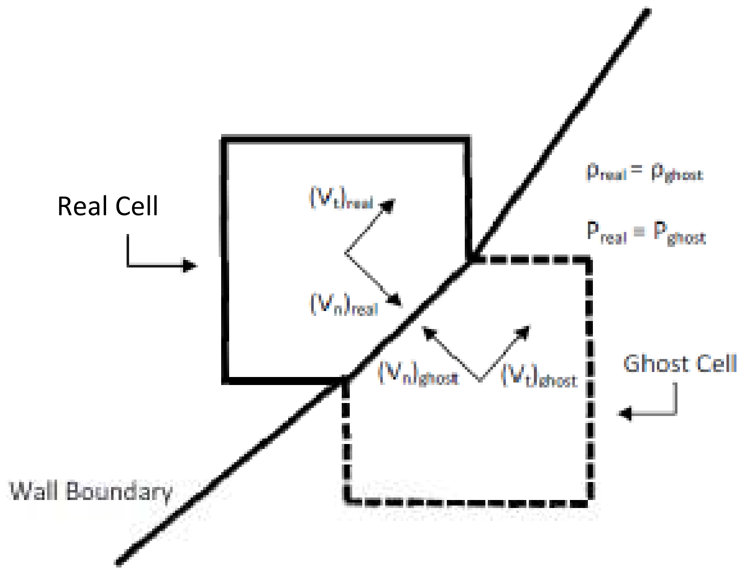


Figure 3.2 Wall boundary conditions for inviscid flow

For viscous flows, interface velocity components of the ghost cell should be reversed in order to provide no-slip condition. In addition to the reversed normal velocity of inviscid flow, the tangential velocity of the ghost cell should also be reverse of the one in the real cell. Furthermore, constant wall temperature is used while computing heat flux terms. In other words, temperature is taken as the same as the one in the real cell.

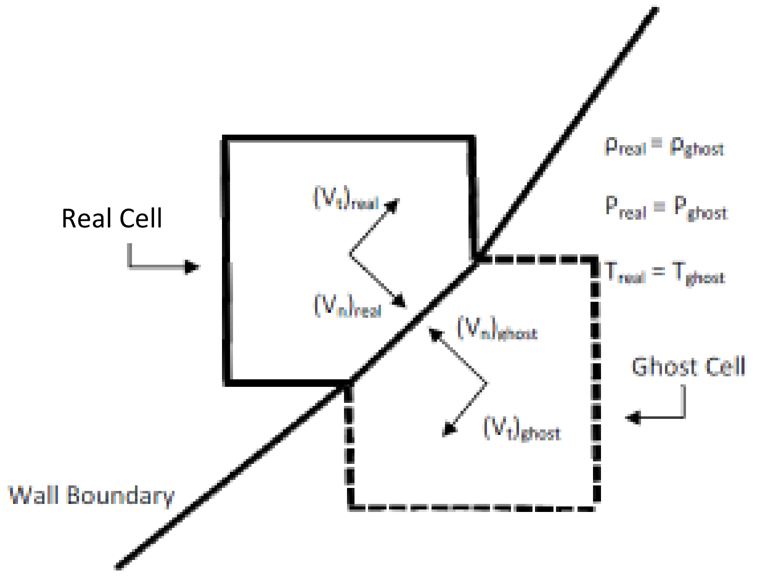


Figure 3.3 Wall boundary conditions for viscous flow

3.2 SPATIAL AND TEMPORAL DISCRETIZATION

After obtaining non-dimensional Navier-Stokes equations with appropriate boundary conditions, some discretization in space and time should be done in order to be capable of solving these equations. Finite volume method is used when discretizing these equations spatially. Although steady flows are considered, there is a need for discretizing time derivative of conserved variables in time in order to equalize it to the residuals. Furthermore, time step calculations should be done accurately by considering the flow type; inviscid or viscous. In addition to these, a cut-back procedure for Courant number is introduced in order to avoid some start-up stability problems that may exist during the execution of the code.

3.2.1 Spatial Discretization

By using finite volume method, integral form of Navier-Stokes equation can be solved easily. Domain is divided into cells, firstly. These cells become the control volumes that do not changed in time. The conserved variables are stored at the cell centroids and it can be assumed that variables of a cell remain the same throughout the whole cell. In addition to this, integrals of inviscid and viscous fluxes can be written as the sum of the fluxes through each face of a cell. Using these assumptions, Equation (3.24) can be written in two-dimensions as;

$$A\frac{\partial \boldsymbol{Q}}{\partial t} + \sum_{faces}\left[\left(\boldsymbol{F} - \frac{M_\infty}{Re_\infty}\boldsymbol{G}\right)\cdot \boldsymbol{n}\right]\Delta s = 0 \tag{3.38}$$

where A is the area of the cell and Δs is the edge length of the face. Using the above equation, residuals of the cell can be defined as;

$$Res(\boldsymbol{Q}) = \sum_{faces}\left[\left(\boldsymbol{F} - \frac{M_\infty}{Re_\infty}\boldsymbol{G}\right)\cdot \boldsymbol{n}\right]\Delta s \tag{3.39}$$

If one combine Equation (3.38) and (3.39), spatially discretized governing equation can be rewritten in terms of residuals.

$$\frac{\partial \boldsymbol{Q}}{\partial t} = -\frac{1}{A} Res(\boldsymbol{Q}) \tag{3.40}$$

As a result, spatially discretized governing equations in compact form are obtained in terms of residuals, area of the cell and time derivative of the conserved variables.

3.2.2 Temporal Discretization

After the spatial discretization, time derivative of the conserved variables should also be discretized. This discretization is called temporal discretization. Although the code solves steady-state flow, temporal discretization is necessary in order to obtain zero residuals by iterative method. Time derivative can be discretized as the difference of the conserved variables of *n*+1'th time step and *n*'th time step divided by specified time step.

$$\frac{\partial \boldsymbol{Q}}{\partial t} = \frac{\boldsymbol{Q}^{n+1} - \boldsymbol{Q}^{n}}{\Delta t} \tag{3.41}$$

This discretized equation can simply be equalized to the residuals of the conserved variables divided by the cell area by using Equation (3.40). While equalizing, two different schemes can be used. If the residuals are calculated using the *n*'th time step, then only unknown will be the conserved variables at the (*n*+1)'th time step. This is called **explicit time scheme**.

$$\frac{\boldsymbol{Q}^{n+1} - \boldsymbol{Q}^{n}}{\Delta t} = -\frac{1}{A} Res(\boldsymbol{Q}^{n}) \tag{3.42}$$

If it is desired to use residuals at the (*n*+1)'th time step, then unknowns are placed at both sides of the equations. This is called **implicit time scheme**.

$$\frac{\boldsymbol{Q}^{n+1} - \boldsymbol{Q}^{n}}{\Delta t} = -\frac{1}{A} Res(\boldsymbol{Q}^{n+1}) \tag{3.43}$$

In the implicit scheme, the residuals at the (*n*+1)'th time step are found using Taylor series expansion by neglecting of the higher order derivatives.

$$Res(\boldsymbol{Q}^{n+1}) = Res(\boldsymbol{Q}^{n}) + \frac{\partial Res(\boldsymbol{Q}^{n})}{\partial \boldsymbol{Q}}(\boldsymbol{Q}^{n+1} - \boldsymbol{Q}^{n}) \tag{3.44}$$

In the developed solver, explicit time scheme is used.

3.2.2.1 Multistage Time Stepping

The discretized equations are solved using multistage time stepping method. In order to use this method, initial guesses should be made. As initial guesses, conserved variables of all cells are taken as the free stream values. Then, using multistage time stepping at each iteration, residuals are found. The general *m*-stage scheme is defined as;

$$\begin{aligned}
&\boldsymbol{Q}^{(0)} = \boldsymbol{Q}^{n} \\
&\boldsymbol{Q}^{(k)} = \boldsymbol{Q}^{(0)} - \upsilon\frac{\alpha_k \Delta t}{A} Res\big(\boldsymbol{Q}^{(k-1)}\big) \qquad k = 1, \dots, m \\
&\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^{(m)}
\end{aligned} \tag{3.45}$$

where υ is the Courant number (CFL Number), α_k is the stage coefficient at the *k*'th stage.

In the developed code, three, four and five stage time stepping can be used with the first order and second order scheme. In analyses, generally three stage time stepping is used. Below, one can see the CFL numbers and stage coefficients according to the stage number and scheme type [38].

Table 3.1 CFL numbers and stage coefficients
for the first order scheme

stages	υ	α_1	α_2	α_3	α_4	α_5
3	1.5	0.1481	0.4000	1.0000		
4	2.0	0.0833	0.2069	0.4265	1.0000	
5	2.5	0.0533	0.1263	0.2375	0.4414	1.0000

Table 3.2 CFL numbers and stage coefficients
for the second order scheme

stages	υ	α_1	α_2	α_3	α_4	α_5
3	0.6936	0.1918	0.4929	1.0000		
4	0.9214	0.1084	0.2602	0.5052	1.0000	
5	1.1508	0.0695	0.1602	0.2898	0.5060	1.0000

3.2.3 Time Step Calculations

Calculation of the time step is very important to obtain fast and stable solutions. It depends on the cell size and the flow properties directly. If it is chosen very small, then solution converges very slowly. On the contrary, if it is taken very large, then solution may diverge easily. In addition to this, calculation method is significant in order to determine the appropriate time step. There are two methods for the calculation of time steps, namely global and local time stepping methods.

In the global time stepping method, all cells in the domain should be examined and minimum time step must be used for all cells. It is necessary while solving unsteady flows in order to obtain logical results at any time step. For steady flows, this method is impractical since only the final solution is considerable, in other words, the solution at any time step is not important. Moreover, with this method, convergence time significantly increases.

In the local time stepping method, every cell has its own time step. In order to solve steady flows, this method is very useful. While the larger cells have greater time steps, the smaller cells have lower time steps. This brings faster convergence to larger cells. Since the solutions at the mid-stages are not required to be accurate, this method provides an important advantage for the convergence time. Moreover, in the code, local time stepping method is used while dealing with steady flows. It is very advantageous since Cartesian mesh has large cell size differences. It is important to note that the time step for each cell is computed at every iteration since flow properties on which the time step calculation depends are changing from one iteration to the other.

Two different calculations are used for the local time stepping method. First one is used when dealing with inviscid flows, while second one is introduced when dealing with viscous flows. In addition to these, a CFL cut-back procedure is used in order to eliminate stability problems especially in earlier iterations of the solution if any.

3.2.3.1 Inviscid Time Step Computation

Local time step of each cell can be computed using the relation below for two-dimensional inviscid problems [27].

$$\Delta t = \frac{A}{\varphi_x + \varphi_y} \tag{3.46}$$

Here, φ_x and φ_y denotes the convective spectral radii and the absolute values of the projection of edges, S_x and S_y, in x and y directions, respectively, are used while computing them.

$$\varphi_x = \frac{1}{2}(|u| + c)\sum_{faces}|S_x| \tag{3.47}$$

$$\varphi_y = \frac{1}{2}(|v| + c)\sum_{faces}|S_y| \tag{3.48}$$

3.2.3.2 Viscous Time Step Computation

In order to avoid stability problems in viscous flows, both convective and diffusive characteristics of the flow must be considered. Thus, the local time step for each cell can be calculated as;

$$\Delta t = \frac{\Delta t_c \, \Delta t_v}{\Delta t_c + \Delta t_v} \tag{3.49}$$

where Δt_c is the convective time step and Δt_v is the viscous time step [28]. Convective time step is calculated similar to the inviscid time step calculation in the previous section. While computing viscous time step, following relation is used.

$$\Delta t_v = K_v \frac{A}{\lambda_v} \tag{3.50}$$

In this relation, K_v is an empirically determined coefficient which considers the relative importance of viscous effects for the final time step expression. It is chosen as 0.25 for most cases. For low Reynolds number flows, since viscous effects are more dominant, this coefficient may be increased to get more stable results, when there are stability problems. The other variable, λ_v, represents the maximum eigenvalue of the diffusive operator of the Navier-Stokes equations and it is a discretized and averaged quantity about the boundary of the control volume and expressed as:

$$\lambda_v = \frac{\gamma\, M_\infty}{Re\, \Pr A} \sum_{faces} \frac{\mu}{\rho} \Delta s^2 \tag{3.51}$$

where dynamic viscosity and density are computed at the face boundary and Δs denotes the face length.

3.2.3.3 CFL Cut-Back Procedure

Sometimes, initial guesses at the critical locations can cause negative pressure and temperatures at the early iterations of the execution. This problem can be solved by decreasing the CFL number. However, this increases the solution time considerably. To avoid this convergence time increase and also stability problems at the start-up, a CFL cut-back procedure may be applied, which limits the maximum relative change in density and pressure per time step [29].

In this procedure, first, the maximum relative change in conserved variables of a cell is found using residuals at the beginning of each time step.

$$\Delta \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} = \frac{\Delta t}{A} \begin{bmatrix} Res(\rho) \\ Res(\rho u) \\ Res(\rho v) \\ Res(\rho E) \end{bmatrix} \tag{3.52}$$

The relative change in pressure can be calculated using the relative changes of specific total energy, density and velocity components

$$\Delta p = (\gamma - 1)\left[\Delta(\rho E) - \big(u\,\Delta(\rho u) + v\,\Delta(\rho v)\big) + \Delta\rho \frac{u^2+v^2}{2}\right] \tag{3.53}$$

The relative change in pressure and density can then be written as:

$$\varepsilon_\rho = \frac{\Delta\rho}{\rho} \tag{3.54}$$

$$\varepsilon_P = \frac{\Delta \text{p}}{p} \tag{3.55}$$

The CFL number may be cut back by making maximum change per time step in either the density or pressure to be less than some specified tolerance, ε_{cut}.

$$\upsilon_{cut} = \frac{\varepsilon_{cut}}{\max\left(\varepsilon_{\rho}, \varepsilon_{p}\right)} \tag{3.56}$$

Then the new CFL number can be obtained by taking the minimum of the original CFL number and cut-back CFL number.

$$\upsilon_{new} = \min\left(\upsilon, \upsilon_{cut}\right) \tag{3.57}$$

While finding cut-back CFL number, specific tolerance is taken as 0.1. It may be thought that convergence time is increased with this procedure. However, it is observed that CFL number is cut back at the very early stages of the run. After these early stages, CFL number quickly increases back to the maximum allowed.

3.3 INVISCID FLUX CALCULATIONS

Inviscid flux calculations play very important role while analyzing a problem. In this study, four different techniques are used. One Riemann solver and three flux-vector splitting methods are used. As Riemann solver, approximate Riemann solver of Roe is used. On the other hand, Liou's Advection Upstream Splitting Method, in short, AUSM, and two derivatives of it, namely AUSMV and AUSMD are embedded into the code.

In these methods, it is required to interpolate the variables of the cell ,whose flux value is calculated, to the midpoint of each face. In addition, the neighboring cell values should also be interpolated to the values at the face. These cells are denoted as left and right cells, respectively. After the variables are moved accurately to the interface of the left and right cells, the inviscid flux vector can be calculated by using one technique described below in detail. Finally, found face flux values according to the face direction must be transformed to the Cartesian coordinates.

Using the rotational invariance of the governing equations as shown in Equation (3.58), one can find the conserved quantities and flux vectors in the normal and tangential directions to the face [24].

$$\boldsymbol{F} \cdot \boldsymbol{n} = \boldsymbol{T}^{-1}\boldsymbol{F}(\boldsymbol{TQ}) = \boldsymbol{T}^{-1}\overline{\boldsymbol{F}}(\overline{\boldsymbol{Q}}) \tag{3.58}$$

In the above equation, the overbar symbol "−" denotes that the quantity or the vector is transformed to the face direction. In addition, **T** and **T**$^{-1}$ are the transformation matrix and its inverse respectively, which can be written in explicit form by using the face normal angle, θ, as;

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.59}$$

$$\boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.60}$$

As a result, transformed conserved quantities and transformed inviscid flux vector become;

$$\overline{\boldsymbol{Q}} = \begin{bmatrix} \rho \\ \rho\overline{u} \\ \rho\overline{v} \\ pE \end{bmatrix} \tag{3.61}$$

$$\overline{\boldsymbol{F}}(\overline{\boldsymbol{Q}}) = \begin{bmatrix} \rho\bar{u} \\ \rho\bar{u}^2 + p \\ \rho\overline{uv} \\ p\bar{u}H \end{bmatrix} \tag{3.62}$$

where $\bar{u}$ and $\bar{v}$ are the normal and tangential velocities to the face direction and can be expressed as;

$$\bar{u} = u\cos\theta + v\,\sin\theta \tag{3.63}$$

$$\bar{v} = v\cos\theta - u\,\sin\theta \tag{3.64}$$

The calculations for inviscid flux should be done according to the transformed quantities and vectors using a technique available. Then, the found vector must be transformed back to the Cartesian coordinates as shown below;

$$\boldsymbol{T}^{-1}\overline{\boldsymbol{F}}(\overline{\boldsymbol{Q}}) = \boldsymbol{F}\cdot\boldsymbol{n} \tag{3.65}$$

<u>3.3.1 Approximate Riemann Solver Of Roe</u>

In the approximate Riemann solver of Roe [30], the following equation is used in general.

$$\overline{\boldsymbol{F_k}}(0) = \frac{1}{2}\left(\overline{\boldsymbol{F_k}}(\overline{\boldsymbol{Q_L}}) + \overline{\boldsymbol{F_k}}(\overline{\boldsymbol{Q_R}})\right) - \frac{1}{2}\sum_{i=1}^{4}|\boldsymbol{\lambda}_i|\boldsymbol{r}_{ik}\Delta\boldsymbol{v}_i \qquad k = 1,\dots,4 \tag{3.66}$$

where subscript *L* denotes the left cell, in other words, the cell whose flux will be calculated, subscript *R* denotes the right cell, i.e. neighboring cell, λ is the eigenvalue 1x4 matrix, *r* is the right eigenvector 4x4 matrix and Δv is the wave strength 1x4 matrix, *k* denotes the row number of the flux vector.

The eigenvalue, the right eigenvector and wave strength matrices are calculated by using Roe's averaged quantities. These quantities are given below.

$$\rho_{RL} = \sqrt{\rho_L \rho_R} \tag{3.67}$$

$$u_{RL} = \frac{\bar{u}_L\sqrt{\rho_L}+\bar{u}_R\sqrt{\rho_R}}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{3.68}$$

$$v_{RL} = \frac{\bar{v}_L\sqrt{\rho_L}+\bar{v}_R\sqrt{\rho_R}}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{3.69}$$

$$H_{RL} = \frac{H_L\sqrt{\rho_L}+H_R\sqrt{\rho_R}}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{3.70}$$

$$c_{RL} = \sqrt{(\gamma-1)\left(H_{RL} - \frac{{u_{RL}}^2+{v_{RL}}^2}{2}\right)} \tag{3.71}$$

Using these averaged quantities, vectors at the right hand side of Equation (3.66) can be calculated using the following relations,

$$\boldsymbol{\lambda} = \begin{bmatrix} u_{RL} - c_{RL} \\ u_{RL} \\ u_{RL} \\ u_{RL} + c_{RL} \end{bmatrix} \tag{3.72}$$

$$\boldsymbol{r} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ u_{RL} - c_{RL} & u_{RL} & 0 & u_{RL} + c_{RL} \\ v_{RL} & v_{RL} & 1 & v_{RL} \\ H_{RL} - u_{RL}c_{RL} & \frac{{u_{RL}}^2+{v_{RL}}^2}{2} & v_{RL} & H_{RL} + u_{RL}c_{RL} \end{bmatrix} \tag{3.73}$$

$$\Delta\boldsymbol{v} = \begin{bmatrix} \frac{\Delta P - \rho_{RL}c_{RL}\Delta u}{2{c_{RL}}^2} \\ \Delta\rho - \frac{\Delta P}{{c_{RL}}^2} \\ \rho_{RL}\Delta v \\ \frac{\Delta P + \rho_{RL}c_{RL}\Delta u}{2{c_{RL}}^2} \end{bmatrix} \tag{3.74}$$

where

(a) $\Delta\rho = \rho_R - \rho_L$ (b) $\Delta P = P_R - P_L$

(c) $\Delta u = u_R - u_L$ (d) $\Delta v = v_R - v_L$ (3.75)

3.3.2 Liou's Advection Upstream Splitting Method (AUSM)

The AUSM scheme [31], [21], [22] works by splitting the advection and pressure terms in the flux of momentum while calculating face flux as;

$$\overline{\boldsymbol{F}}(0) = \frac{1}{2}\left[M_{1/2}(\psi_R + \psi_L) - \left|M_{1/2}\right|(\psi_R - \psi_L)\right] + p_{1/2} \tag{3.76}$$

where $M_{1/2}$ is the split Mach number, $p_{1/2}$ is the split pressure and Ψ is the flux vector which is obtained by removing the pressure term.

Flux vector can then be expressed as;

$$\boldsymbol{\psi} = \begin{bmatrix} \rho c \\ \rho c\bar{u} \\ \rho c\bar{v} \\ \rho cE \end{bmatrix} \tag{3.77}$$

The split Mach number can be written as follows;

$$M_{1/2} = \overline{M}_L^{+} + \overline{M}_R^{-} \tag{3.78}$$

where $\overline{M}_L^{+}$ is the positive part of Mach number using normal component of velocity in the face direction of the left state, and $\overline{M}_R^{-}$ is the negative part of Mach number using normal component of velocity in the face direction of the right state. They can be expressed as;

$$\overline{M}_L^+ = \begin{cases} \frac{1}{4}\left(\overline{M}_L + 1\right)^2 & |\overline{M}_L| \leq 1 \\ \frac{1}{2}\left(\overline{M}_L + |\overline{M}_L|\right) & |\overline{M}_L| > 1 \end{cases} \tag{3.79}$$

$$\overline{M}_R^- = \begin{cases} -\frac{1}{4}\left(\overline{M}_R - 1\right)^2 & |\overline{M}_R| \leq 1 \\ \frac{1}{2}\left(\overline{M}_R - |\overline{M}_R|\right) & |\overline{M}_R| > 1 \end{cases} \tag{3.80}$$

where $\overline{M}_L$ and $\overline{M}_R$ represents left state and right state Mach number based on transformed velocity.

$$\overline{M}_L = \frac{\overline{u}_L}{c_L} \qquad \text{and} \qquad \overline{M}_R = \frac{\overline{u}_R}{c_R} \tag{3.81}$$

The split pressure can be written as follows;

$$\boldsymbol{p}_{\mathbf{1/2}} = \begin{bmatrix} 0 \\ p_L^+ + p_R^- \\ 0 \\ 0 \end{bmatrix} \tag{3.82}$$

where

$$p_L^+ = p_L\overline{M}_L \begin{cases} 2 - \overline{M}_L & |\overline{M}_L| \leq 1 \\ 1/\overline{M}_L & |\overline{M}_L| > 1 \end{cases} \tag{3.83}$$

$$p_R^- = p_R\overline{M}_R \begin{cases} -2 - \overline{M}_R & |\overline{M}_R| \leq 1 \\ 1/\overline{M}_R & |\overline{M}_R| > 1 \end{cases} \tag{3.84}$$

3.3.3 AUSMD Method

AUSMD method [32], [22] is a derivative of AUSM method. This is referred to as AUSMD since the numerical flux is calculated similar to the finite difference splitting scheme's (FDS). In this method, splitting of mass flux is used instead of mach number splitting as well as the flux vector is modified. In addition to this, pressure is also split similar to AUSM method. In this scheme, the interface flux is calculated using the following relation;

$$\overline{\boldsymbol{F}}(0) = \frac{1}{2}\left[(\rho U)_{1/2}(\psi_R + \psi_L) - \left|(\rho U)_{1/2}\right|(\psi_R - \psi_L)\right] + p_{1/2} \tag{3.85}$$

where $(\rho U)_{1/2}$ is the splitted mass flux based on splitted velocity defined according to left and right values so that the best resolution can be obtained for shock discontinuity. φ is the modified flux vector and $p_{1/2}$ is the split pressure.

The modified flux vector is defined as;

$$\boldsymbol{\psi} = \begin{bmatrix} 1 \\ \overline{u} \\ \overline{v} \\ H \end{bmatrix} \tag{3.86}$$

The mass flux can be expressed in split form as follows;

$$(\rho U)_{1/2} = U_L^+ \rho_L + U_R^- \rho_R \tag{3.87}$$

where

$$U_L^+ = \begin{cases} \alpha_L \left[\frac{(\overline{u}_L + c_{max})^2}{4c_{max}}\right] + (1 - \alpha_L)\left[\frac{\overline{u}_L + |\overline{u}_L|}{2}\right] & |\overline{u}_L| \le c_{max} \\ \frac{\overline{u}_L + |\overline{u}_L|}{2} & |\overline{u}_L| > c_{max} \end{cases} \tag{3.88}$$

$$U_R^- = \begin{cases} \alpha_R \left[-\frac{(\overline{u}_R - c_{max})^2}{4c_{max}}\right] + (1 - \alpha_R)\left[\frac{\overline{u}_R - |\overline{u}_R|}{2}\right] & |\overline{u}_R| \le c_{max} \\ \frac{\overline{u}_R - |\overline{u}_R|}{2} & |\overline{u}_R| > c_{max} \end{cases} \tag{3.89}$$

In these relations, α_L, α_R and maximum interface sound speed are defined as follows;

$$\alpha_L = \frac{2\left(^{P_L}/_{\rho_L}\right)}{\left(^{P_L}/_{\rho_L}\right)+\left(^{P_R}/_{\rho_R}\right)} \tag{3.90}$$

$$\alpha_R = \frac{2\left(^{P_R}/_{\rho_R}\right)}{\left(^{P_L}/_{\rho_L}\right)+\left(^{P_R}/_{\rho_R}\right)} \tag{3.91}$$

$$c_{max} = max(c_L, c_R) \tag{3.92}$$

The split pressure can be written as follows;

$$\boldsymbol{p_{1/2}} = \begin{bmatrix} 0 \\ p_L^+ + p_R^- \\ 0 \\ 0 \end{bmatrix} \tag{3.93}$$

where

$$p_L^+ = p_L \begin{cases} \frac{(\overline{u}_L + c_{max})^2}{4c_{max}{}^2}\left(2 - \frac{\overline{u}_L}{c_{max}}\right) & |\overline{u}_L| \le c_{max} \\ \frac{\overline{u}_L + |\overline{u}_L|}{2\overline{u}_L} & |\overline{u}_L| > c_{max} \end{cases} \tag{3.94}$$

$$p_R^- = p_R \begin{cases} \frac{(\overline{u}_R - c_{max})^2}{4c_{max}{}^2}\left(2 + \frac{\overline{u}_R}{c_{max}}\right) & |\overline{u}_R| \le c_{max} \\ \frac{\overline{u}_R - |\overline{u}_R|}{2\overline{u}_R} & |\overline{u}_R| > c_{max} \end{cases} \tag{3.95}$$

3.3.4 AUSMV Method

AUSMV method [32], [22] is another derivative of AUSM scheme referring to finite volume splitting (FVS). This scheme is very similar to the AUSMD except normal momentum calculations and splitting computations of pressure and velocity.

Equation (3.85) is used for the calculation of interface flux, again. However, in the second column of face flux vector, instead of face flux calculation terms without pressure, a new normal momentum calculation is used as follows;

$$\tfrac{1}{2}\left[(\rho U)_{1/2}(\psi_R + \psi_L) - \left|(\rho U)_{1/2}\right|(\psi_R - \psi_L)\right] \Rightarrow (\rho U^2)_{1/2} \tag{3.96}$$

where normal momentum flux at the interface, $(\rho U^2)_{1/2}$, can be expressed as;

$$(\rho U^2)_{1/2} = U_L^+ \rho_L \overline{u}_L + U_R^- \rho_R \overline{u}_R \tag{3.97}$$

In this scheme, the velocity can be split as;

$$U_L^+ = \begin{cases} \frac{(\overline{u}_L + c_L)^2}{4c_L} & |\overline{u}_L| \le c_L \\ \frac{\overline{u}_L + |\overline{u}_L|}{2} & |\overline{u}_L| > c_L \end{cases} \tag{3.98}$$

$$U_R^- = \begin{cases} -\frac{(\overline{u}_R - c_R)^2}{4c_R} & |\overline{u}_R| \le c_R \\ \frac{\overline{u}_R - |\overline{u}_R|}{2} & |\overline{u}_R| > c_R \end{cases} \tag{3.99}$$

For the interface split pressure, Equation (3.93) can be used with the new split computations as shown below;

$$p_L^+ = p_L U_L^+ \begin{cases} \frac{1}{c_L}\left(2 - \frac{\overline{u}_L}{c_L}\right) & |\overline{u}_L| \le c_L \\ \frac{1}{\overline{u}_L} & |\overline{u}_L| > c_L \end{cases} \tag{3.100}$$

$$p_R^- = p_R U_R^- \begin{cases} \frac{1}{c_L}\left(-2 - \frac{\overline{u}_R}{c_R}\right) & |\overline{u}_R| \le c_R \\ \frac{1}{\overline{u}_R} & |\overline{u}_R| > c_R \end{cases} \tag{3.101}$$

3.4 RECONSTRUCTION

In the code, cell centered approach is used. In other words, the primitive and conserved flow variables are calculated and stored at the centroids of each cell. For the calculation of inviscid fluxes, primitive flow variables should be estimated at both sides of the interface between two cells. As mentioned before, the cell whose flux will be calculated is referred to as the left state, while neighboring cell is named as the right state.

As estimation methods of variables at the interface, two schemes can be used, namely first order and second order schemes. In first order schemes, the flow variables at the cell centroids are simply taken as the flow variables at the face for both left and right states. In second order schemes, the cell-centered flow variables should be reconstructed in order to use them at the interface. With the reconstruction, one may obtain more accurate results whereas the solution time considerably increases since gradients must be calculated for all cells at each iteration.

As reconstruction scheme, least squares reconstruction method is used in order to calculate the gradients of flow variables at the cell centroids. After gradients are found, they are used to estimate the primitive flow variables at the interfaces of the cells.

3.4.1 Least Squares Reconstruction

There are two popular reconstruction schemes available in the literature. The least squares reconstruction method [7] is used in the developed code since it gives more accurate results compared to the second scheme, path integral method [33]. The variables at a certain point in a cell can be calculated using primitive variables and their gradients at the cell centroids as shown below;

$$\boldsymbol{q}(x,y) = \boldsymbol{q}_{cell} + \frac{d\boldsymbol{q}}{dx}(x - x_c) + \frac{d\boldsymbol{q}}{dy}(y - y_c) \tag{3.102}$$

where subscript '*cell*' denotes the centroid of the cell whose gradients are sought, **q** is the vector of primitive variables and can be expressed as;

$$\boldsymbol{q} = \begin{bmatrix} \rho \\ \overline{u} \\ \overline{v} \\ P \end{bmatrix} \tag{3.103}$$

The gradients of primitive variables can be calculated using the primitive variables at the cell centroids of the cell whose gradients are calculated and neighbor cells. In the following relations, subscript 'n' denotes the neighbor cells.

$$\frac{d\boldsymbol{q}}{dx} = \frac{1}{\Delta}\left[I_{yy}\sum_n(\boldsymbol{q}_n - \boldsymbol{q}_{cell})(x_n - x_{cell}) - I_{xy}\sum_n(\boldsymbol{q}_n - \boldsymbol{q}_{cell})(y_n - y_{cell})\right] \tag{3.104}$$

$$\frac{d\boldsymbol{q}}{dy} = \frac{1}{\Delta}\left[I_{xy}\sum_n(\boldsymbol{q}_n - \boldsymbol{q}_{cell})(x_n - x_{cell}) - I_{xx}\sum_n(\boldsymbol{q}_n - \boldsymbol{q}_{cell})(y_n - y_{cell})\right] \tag{3.105}$$

where

$$I_{xx} = \sum_n(x_n - x_{cell})^2 \tag{3.106}$$

$$I_{yy} = \sum_n(y_n - y_{cell})^2 \tag{3.107}$$

$$I_{xy} = \sum_n(x_n - x_{cell})\,(y_n - y_{cell}) \tag{3.108}$$

$$\Delta = I_{xx}I_{yy} - {I_{xy}}^2 \tag{3.109}$$

3.4.2 Gradient Limiting

In order to avoid numerical oscillations at steep gradients which may lead to stability problems, a limiter can be used for the gradients. With the use of limiter, calculation of primitive variables at a certain point in the cell may be modified as follows;

$$\boldsymbol{q}(x,y) = \boldsymbol{q}_{cell} + \boldsymbol{\phi}\left[\frac{d\boldsymbol{q}}{dx}(x - x_c) + \frac{d\boldsymbol{q}}{dy}(y - y_c)\right] \tag{3.110}$$

where $\boldsymbol{\phi}$ is the limiter vector which is a 1x4 matrix for gradients of four primitive variables. The limiter value must be between 0 and 1. In order to determine its value, one should need the maximum and minimum quantities of primitive variables among the considered cell and its neighbors,

$$\begin{aligned} \boldsymbol{q}^{max} &= \max(\boldsymbol{q}_{cell}, \boldsymbol{q}_n) \\ & \qquad\qquad n = 1, \dots, m^{th}\ neighbor \\ \boldsymbol{q}^{min} &= \min(\boldsymbol{q}_{cell}, \boldsymbol{q}_n) \end{aligned} \tag{3.111}$$

where m is the number of neighbors of the cell under consideration. To compute exact value of the limiter, it is also necessary to know the maximum and minimum quantities of primitive variables in the cell. For outside cells, these points are usually in the corners. However, for cut and split cells, these points may also be at the intersection locations. For each point in a cell, the limiter value is calculated as shown below [34],

$$\phi_i = \begin{cases} 1 & \boldsymbol{q}_i = \boldsymbol{q}_{cell} \\ \min\left(1, \frac{(\boldsymbol{q}^{min} - \boldsymbol{q}_{cell})}{(\boldsymbol{q}_i - \boldsymbol{q}_{cell})}\right) & \boldsymbol{q}_i < \boldsymbol{q}_{cell} \\ \min\left(1, \frac{(\boldsymbol{q}^{max} - \boldsymbol{q}_{cell})}{(\boldsymbol{q}_i - \boldsymbol{q}_{cell})}\right) & \boldsymbol{q}_i > \boldsymbol{q}_{cell} \end{cases} \qquad i = 1, \dots, k \tag{3.112}$$

where k is the number of points which are examined in the cell to determine maximum and minimum quantities.

After computing all limiter values for all points, the minimum of all is chosen as the exact limiter value,

$$\phi = \min\,(\phi_1, \phi_2, \dots, \phi_k) \tag{3.113}$$

For all primitive variables, the same procedure is applied and the limiter vector is obtained.

3.5 VISCOUS FLUX CALCULATIONS

The viscous flux at a face, denoted as $\boldsymbol{G}$, can be expressed as a function of flow variables at the face and their gradients,

$$\boldsymbol{G} = f(\boldsymbol{q}_f, \nabla \boldsymbol{q}_f) \tag{3.114}$$

The flow variables at the face can be obtained through the averages of left and right state flow variables at the cell centroids. However, for calculating face gradients of these variables, different methods are available, some of which gives fast results but less accurate, some of which gives more accurate but slower results.

3.5.1 Reconstruction for Viscous Flux

The face gradients can be obtained by using cell-centered gradients computed by inviscid reconstruction and cell-centered flow variables. By adding gradients of inviscid reconstruction to the viscous flux computation, data further away from the interface are considered so that more accurate results may be acquired [7].

Along the direction between left and right cells of the interface, the derivative of a variable is obtained through transforming the gradients into *x* and *y* coordinates, which are calculated by averaging the gradients computed by inviscid reconstruction at cell centroids.

$$\frac{dq}{dt} = \frac{1}{2}\left\{\left[\left(\frac{dq}{dx}\right)_L \cos\theta_t + \left(\frac{dq}{dy}\right)_L \sin\theta_t\right] + \left[\left(\frac{dq}{dx}\right)_R \cos\theta_t + \left(\frac{dq}{dy}\right)_R sin\ \theta_t\right]\right\} \tag{3.118}$$

In the above equation, *t* is the unit vector along the interface, *n* is the unit vector along the direction between left and right cells and θ_t is the angle between vector *t* and *x* axis. In Figure 3.4, these vectors are illustrated for the interface between two regular cells.

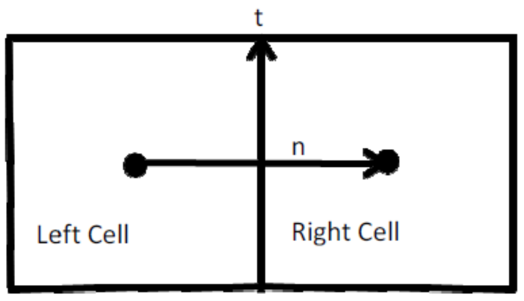


Figure 3.4 Schematic view of viscous flux computation at a face

The derivative of *q* along *n* direction can simply be computed using central difference as

$$\frac{dq}{dn} = \frac{q_{neigh} - q_{cell}}{|r_{neigh} - r_{cell}|} \tag{3.119}$$

Note that the above relations can also be written in terms of face gradients as follows;

$$\left(\frac{dq}{dx}\right)_f \cos\theta_n + \left(\frac{dq}{dy}\right)_f \sin\theta_n = \frac{dq}{dn} \tag{3.120}$$

$$\left(\frac{dq}{dx}\right)_f \cos\theta_t + \left(\frac{dq}{dy}\right)_f \sin\theta_t = \frac{dq}{dt} \tag{3.121}$$

where θ_n is the angle between vector *n* and *x* axis. In these relations, the only two unknowns are the *x* and *y* derivatives of face gradients, so that they can be found easily.

For calculating face gradients at the wall boundary, right state is taken at the wall. Here, the primitive variable vector can be written as;

$$\boldsymbol{q}_R = \begin{bmatrix} \rho_{cell} \\ 0 \\ 0 \\ p_{cell} \end{bmatrix} \tag{3.122}$$

The inviscid reconstructed gradients at right state are also necessary in order to compute the viscous flux. These are taken as same as inviscid reconstructed gradients of the cell itself in order to have the same gradients at the face

3.6 CALCULATION OF THE COEFFICIENTS

In order to verify the code's accuracy, two coefficients are used along the chord length. While the skin friction coefficient are calculated for viscous flows, pressure coefficient are used for both inviscid and viscous flows. These coefficients are calculated for the cells near the wall boundary. Then, the graph created from these data can be used to compare with the available numerical or experimental data in the literature.

3.6.1 Pressure Coefficient

Pressure coefficient is a non-dimensional quantity which describes the relative pressure along the chord length of the airfoil. The difference between stagnation and static pressure is non-dimensionalized by the dynamic pressure.

$$c_P = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty {V_\infty}^2} \tag{3.123}$$

While the dynamic pressure can be used for inviscid flows, it is not an accurate measure for viscous flows. The free-stream Mach number can be used for compressible viscous flows.

$$c_P = \frac{2}{\gamma {M_\infty}^2}\left(\frac{p}{p_\infty} - 1\right) \tag{3.124}$$

It is important to note that both equations for pressure coefficients lead to same result with the non-dimensionalized free-stream values and boundary conditions. Therefore, in the code, Equation (3.123) is used. If non-dimensionalization was not used in the code, then it would be necessary to compute the coefficient by using Equation (3.124).

3.6.2 Skin Friction Coefficient

Skin friction comes from the friction of the "skin" of the wall against the moving fluid on it. While calculating skin friction, local wall shear stress, τ_w, is used. Non-dimensionalization is realized by dynamic pressure similar to the pressure coefficient.

$$c_f = \frac{\tau_w}{\frac{1}{2}\rho_\infty {V_\infty}^2} \tag{3.125}$$

The local shear stress should be taken along the tangential direction to the wall. The normal and shear stresses along the faces are calculated with the viscous reconstruction as mentioned before. After the shear stresses in the nearest cells to the wall are computed, these must be converted to the tangential direction from Cartesian coordinates. Mohr circle can be used for this conversion.

Mohr circle can be established by the planar normal stresses, τ_{xx} and τ_{yy}, and shear stress, τ_{xy}. Later, using this circle, the transformed stresses can be found at any point on the circle. The following relation can be obtained from the Mohr circle in order to compute the transformed shear stress at a point which is at an angle of θ away from the x axis.

$$\tau_{x'y'} = -\frac{\tau_{xx}-\tau_{yy}}{2}\sin 2\theta + \tau_{xy}\cos 2\theta \tag{3.126}$$

τ_{xv}

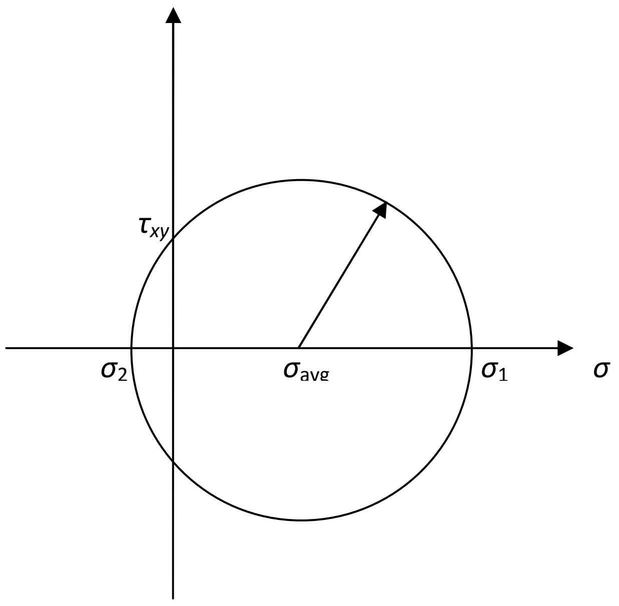


Figure 3.5 Mohr circle

3.7 SOLUTION ADAPTATION

Solution adaptation is an important grid adaptation method which is applied during the execution of the program when a specified level of convergence is achieved. It is applied according to the compressibility and rotationality principles of the flow. The critical regions containing discontinuities due to shocks and stagnation points are refined so that resolution at these locations is increased to get more accurate results.

The criterion for solution adaptation is based on divergence and curl of velocity [38], [22], for determining shock locations and shear layers accurately. A characteristic length is used while using these criterion as shown below;

$$\tau_D = |\nabla \cdot \boldsymbol{V}| {A_{cell}}^{3/2} \tag{3.127}$$

$$\tau_C = |\nabla \times \boldsymbol{V}| {A_{cell}}^{3/2} \tag{3.128}$$

For each cell, these criteria are checked. If one of these criteria is greater than the standard deviations of these quantities, σ_D and σ_C which are given below, then the cell is refined.

$$\sigma_D = \sqrt{\frac{\sum_{i=1}^{n}(\tau_D)^2}{n}} \tag{3.129}$$

$$\sigma_C = \sqrt{\frac{\sum_{i=1}^{n}(\tau_C)^2}{n}} \tag{3.130}$$

where n is the total number of cells. In Figure 3.6, one can see an example to the solution adaptation. Here, the grids at the shock location, stagnation points and shear layers become finer with six cycles of solution adaptation.

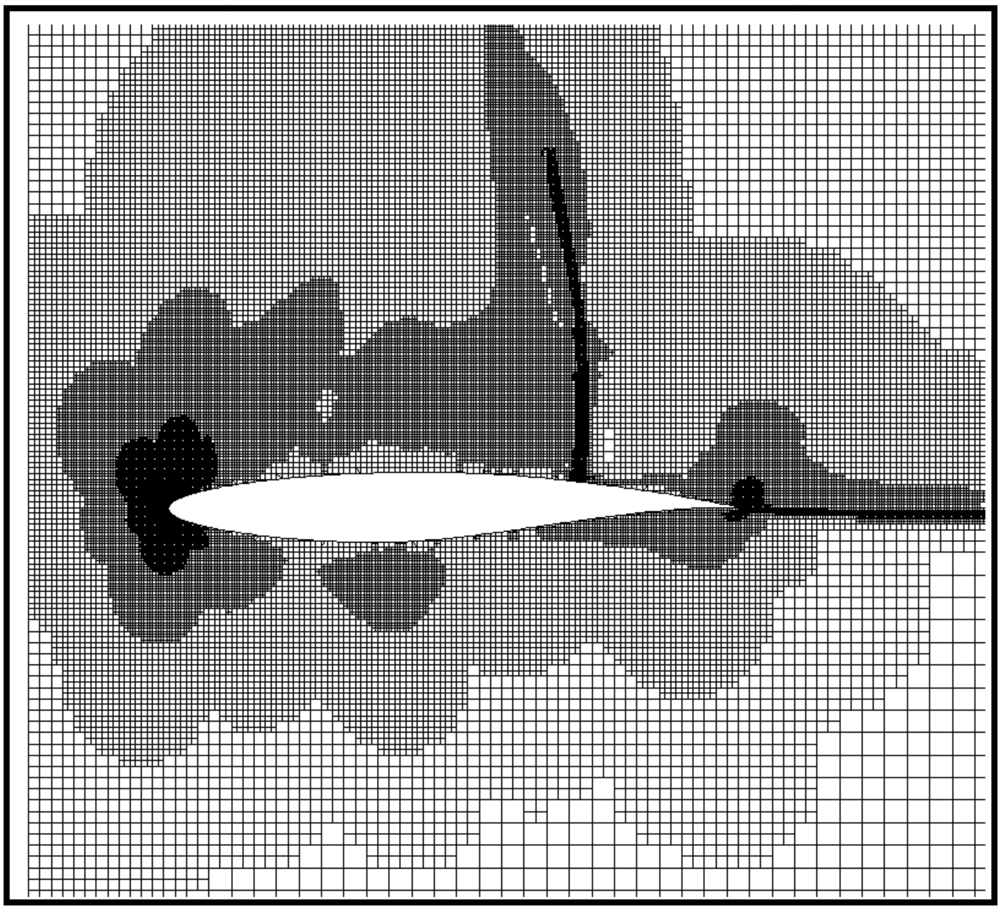


Figure 3.6 An example of solution adaptation

CHAPTER 4

MULTIGRID METHOD

Multigrid is a technique that accelerates the convergence rate by using coarser grids in order to eliminate the low frequency errors. It is based on two principles, error smoothing and coarse grids. In the first principle, some iterations are performed on the finest grid in order to eliminate the high frequency errors. These iterations cannot reduce the low frequency errors significantly. In order to smooth the low frequency errors, coarse grids are used. The solutions on the finest mesh are transformed to the coarser meshes and some iterations are performed on these meshes. As a result, high frequency errors for coarser grids are improved. Since these high frequency errors are low frequency errors for finest mesh, one may reduce low frequency errors by transforming solutions back to the finest grid. Therefore, high and low frequency errors are eliminated by multigrid. [35]

Multigrid method can be used for linear and non-linear problems. Brandtl [36] developed an effective multigrid method for non-linear problems. This is called Full Approximate Storage (FAS) scheme. Then, Jameson [37] and De Zeeuw [38] implemented this scheme into Euler solvers. In the developed code, this scheme is used.

In this chapter, multigrid concept and its steps are introduced for non-linear problems, at first. Next, the coarsening process for Cartesian and quad grids are explained in detail. Finally, the effect of multigrid technique both for inviscid and viscous flows are investigated with some tables and graphs.

4.1 MULTIGRID CONCEPT FOR NON-LINEAR EQUATIONS

The form of a non-linear problem may be presented as shown by the following equation , in a discretized way.

$$L_h \breve{\boldsymbol{Q}}_h = 0 \tag{4.01}$$

In this equation, L represents the non-linear differential space operator, $\breve{\mathbf{Q}}$ is the converged discrete solution and subscript h denotes the mesh spacing for grids. While h is the finest step size, $2h$, $4h$ ... and nh represent the coarser step sizes. If one use the approximate discrete solution, $\widehat{\mathbf{Q}}$, the following relation is obtained.

$$L_h \widehat{\boldsymbol{Q}}_h = R_h(\widehat{\boldsymbol{Q}}_h) \tag{4.02}$$

In this relation, R denotes the residual function. If Equation (4.01) is subtracted from Equation (4.02), one can acquire the following equation:

$$L_h \widehat{\boldsymbol{Q}}_h - L_h \breve{\boldsymbol{Q}}_h = R_h(\widehat{\boldsymbol{Q}}_h) \tag{4.03}$$

Since the error function is the difference between the approximate solution and the exact solution, this equation can be approximated by using the solution which is obtained at the coarser grid one step away from the initial grid. To do this, a restriction operator which transfers the information from the finer to coarser grid is used both for residual function and the approximate solution.

$$L_{2h}(I_h^{2h} \breve{\boldsymbol{Q}}_h + \boldsymbol{e}_{2h}) - L_{2h} I_h^{2h} \breve{\boldsymbol{Q}}_h = I_h^{2h} \boldsymbol{R}_h \tag{4.04}$$

The error function at the step $2h$, which is the only unknown in the equation above, can be found. Then, by using the prolongation operator which transfers the information from coarser to finer grid, the improved approximate solution is obtained.

$$\breve{\boldsymbol{Q}}_h^{new} = \breve{\boldsymbol{Q}}_h + I_{2h}^h \boldsymbol{e}_{2h} \qquad (4.05)$$

These equations are adapted to the code in four steps. These are fine grid iterations, restriction, prolongation and final iterations with correction. When the second order scheme is used, some problems occurred during the application of the multigrid scheme. After describing multigrid steps, the modifications done for the second order scheme are explained.

4.1.1 Fine Grid Iterations

Some iterations are performed initially in order to decrease the high frequency errors. These iterations are done in the finest mesh and simply multistage time stepping is used, as described in Chapter 3.

$$\boldsymbol{Q}_h^{(0)} = \boldsymbol{Q}_h^n$$

$$\boldsymbol{Q}_h^{(k)} = \boldsymbol{Q}_h^{(0)} - \upsilon \frac{\alpha_k \Delta t}{A}\left[Res\left(\boldsymbol{Q}_h^{(k-1)}\right) + \boldsymbol{FF}_h\right] \qquad k = 1, \ldots, m \qquad (4.03)$$

$$\boldsymbol{Q}_h^{n+1} = \boldsymbol{Q}_h^{(m)}$$

In the equation set above, subscript *n* denotes the time step and *FF* represents the forcing function. For the fine grid iterations step, forcing function is zero for all conserved variables. By using the solutions in the *n*'th time step, the solutions in the (*n*+1)'th time step are simply found. The high frequency errors are considerably reduced while low frequency errors decrease slightly. The number of iterations in this step is determined by an input. After all fine grid iterations are performed in one cycle, the residuals obtained from the latest results, i.e. $Res(\mathbf{Q}_h{}^m)$ are found in order to use them in restriction part.

4.1.2 Restriction

In this step, the results obtained in the finest mesh are transferred to the coarser meshes with the use of the restriction operator. The obtained approximate solution and the final residual in the finest mesh are used to determine the initial guess for the computational cells in the coarser grid. Before these, coarser meshes must be obtained and the cell relations between the grids must be determined. This process is described in Section 4.2 in a detailed manner. At the moment, one may assume that the coarser grids are obtained and the cell relations are set. The equivalent cells at coarser grids are obtained with coarsening process. If one looks at a coarser cell, it may cover four children or one cell which is the coarsened cell itself. In this section, each of these finer cells is referred to as the equivalent finer cell. If one looks at a finer cell from a coarser cell, the equivalent coarser cell may be the cell itself or the parent of the cell according to the coarsening process. These naming is expressed in Figure 4.1 in order to be understood well.

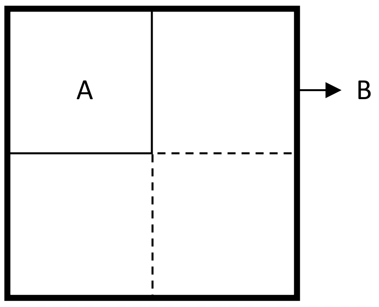


Figure 4.1 Illustration of "equivalent cell" term

In Figure 4.1, cells A and B are shown. Cell A is a computational cell in the mesh spacing *h* while cell B is a computational cell in the mesh spacing 2*h*. In terms of cell relations between two mesh spacings, cell A is an equivalent finer cell for cell B. On the other hand, cell B is an equivalent coarser cell for cell A. It can be seen that cell B has four equivalent finer cells.

After this brief explanation, the equations which are used to calculate the initial guesses of the solutions and the forcing function at the coarser step can be written as;

$$\boldsymbol{Q}_{2h}^{(0)} = I_h^{2h}\boldsymbol{Q}_h^{(m)} \tag{4.04}$$

$$\boldsymbol{FF}_{2h} = \hat{I}_h^{2h}\left[Res\left(\boldsymbol{Q}_h^{(m)}\right) + \boldsymbol{FF}_h\right] - Res\left(\boldsymbol{Q}_{2h}^{(0)}\right) \tag{4.05}$$

where I_h^{2h} is the volume weighted restriction operator and $\hat{I}_h^{2h}$ is the residual collection operator, which can be expressed as;

$$I_h^{2h}\boldsymbol{Q}_h^{(m)} = \frac{\sum_{\substack{equivalent \\ finer\ cells}}\left[\boldsymbol{Q}_h^{(m)}A\right]}{\sum_{\substack{equivalent \\ finer\ cells}}[A]} \tag{4.06}$$

$$\hat{I}_h^{2h}[\alpha] = \sum_{\substack{equivalent \\ finer\ cells}}[\alpha] \tag{4.07}$$

After setting the initial values and forcing functions for coarser grids, some mid step iterations are performed, whose number is determined according to the input specified by the user. These lead to the improvement of approximate solutions at coarser grids. In the equations above, transformation from h to $2h$ is illustrated. If the level number is higher than 1, the approximate solutions for more coarser grids such as $4h$, $8h$, $16h$ etc. can be obtained using the same methodology after obtaining the improved solutions at one mesh spacing size before. In this code, one can use 7 levels at maximum, providing a coarse mesh of a spacing of $128h$.

4.1.3 Prolongation

The aim in this step is to transfer the results obtained in coarser meshes to the actual finest mesh and to get an improved result. To do this, a prolongation operator is used with the approximate results found in the coarser meshes, as shown by the following equation;

$$\boldsymbol{Q}_h^{new} = \boldsymbol{Q}_m^h + I_{2h}^h\left(\boldsymbol{Q}_{2h}^{(new)} - I_h^{2h}\boldsymbol{Q}_{2h}^{(m)}\right) \tag{4.08}$$

where I_{2h}^h is the prolongation operator and can be written as;

$$I_{2h}^h(\alpha) = \alpha \tag{4.09}$$

This is referred to as the injection operator for prolongation. As a prolongation operator, a gradient operator which is the dot product of the gradient and direction vector may be used, as well. However, for simplicity, injection operator is used in the developed code. The "*new*" superscript that is seen in Equation (4.08) represents the new results after interpolating the approximate results to the mesh spacing that the results are sought. It can be obtained from prolongation. However, if ,for example, the level is set to 1, then the new values are taken from the restriction part after mid step iterations are performed.

In the developed code, two different cycles are used. These are the Saw-Tooth and V-Cycle. The only difference between them are the iterations performed after prolongation is applied in the V-Cycle. In the V-Cycle, some iterations are performed in each level after the new improved solutions are acquired with the transfer of the solutions to the coarser level. Since forcing functions are necessary to perform iterations, it is necessary to store all forcing functions found in the restriction part for all levels. Since it brings a low efficiency for memory with the storage of excessive number of variables, it is expected to have a lower convergence rate compared to the Saw-Tooth cycle. The comparison between these two cycles are done in the next two sections of this chapter.

<u>4.1.4 Correction and Final Iterations</u>

The final step in the multigrid technique is the correction and final iterations. In this step, the improved approximate solutions are corrected with some iterations, like fine grid iterations. As an initial guess, the found improved solution after prolongation, i.e. $\boldsymbol{Q}_h^{new}$, is used. This step is applied only for the finest mesh with a number of iterations determined by a seperate input. As a result of all these steps, the solutions at the finest mesh are obtained with the elimination of low and high frequency errors as much as possible.

4.1.5 Modifications for the Second Order Scheme

While solving flows with a second order and multigrid technique, some problems were encountered. In these attempts, the pressure and/or the density were decreased to negative values in coarser grids. Therefore, the solutions in the coarser grids are obtained by using a first order scheme even if the scheme is second order for the finest grid. Only at the fine grid iterations step and the correction and final iterations step, the second order scheme is used. With this slight modification, stability during the execution of the program is provided.

Whereas solutions are verified in Chapter 5, in order to validate that the pure second order solution and the modified solution give the same result, a comparison is done around RAE 2822 airfoil, using an inviscid flow at a Mach number of 0.75 and an angle of attack of 3 degrees. Furthermore, five cycles of solution adaptation is applied to the grid. The pressure coefficient distribution are depicted in Figure 4.2.

As it can be seen, nearly the same distribution is observed for two cases. Moreover, the drag coefficient is calculated as 0.0427 for both cases. A slight difference is formed for lift coefficient such that seven level multigrid solution calculates the lift coefficient as 0.9725 while solution without multigrid gives a lift coefficient of 0.9720. Since this very small effect can be neglected while using the second order scheme, it can be concluded that modifications are useful if the second order scheme is used together with the multigrid technique.

Moreover, the multigrid scheme affects solution time significantly. A speed up ratio of 19 is obtained in this problem. As described in the next section, a maximum ratio of 17 is acquired for this inviscid flow using the first order scheme. The difference can be due to the difference in time required for the execution between the first and second order schemes. Second order schemes take more time in comparison to first order schemes since gradients and limiters are calculated at each iteration for each cell. In multigrid solutions, the second order scheme is used partly. Therefore, the convergence time decreases significantly by using with the first order scheme.

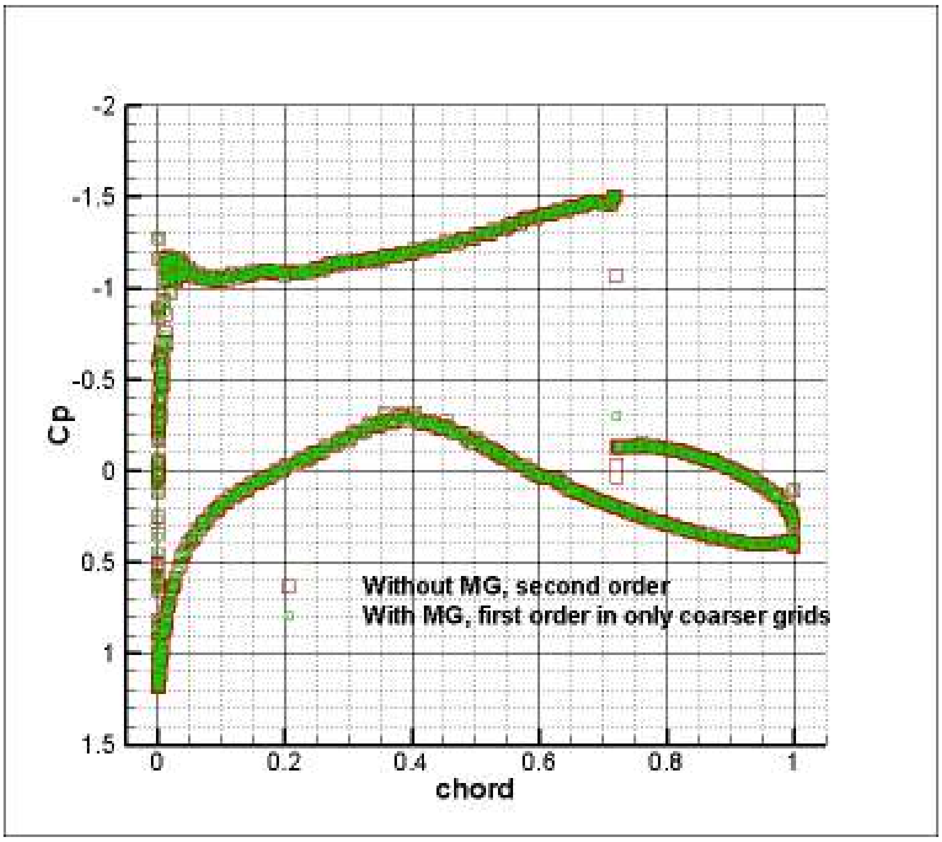


Figure 4.2 Comparison of pressure coefficient distribution obtained by using partly and purely second order scheme around RAE 2822 airfoil

4.2 COARSENING PROCESS

The coarsening process, which is applied in the restriction part of multigrid technique, is presented in detail in this section. First, how to coarsen Cartesian cells are presented. Then, the Quad cells are considered for the coarsening process.

4.2.1 Coarsening of Cartesian Cells

During coarsening process, three pointers are used, as discussed briefly in Section 2.1.3. These are “perform”, “meshSpacing” and “compCell” words. First, the cells are flagged with the perform word if all of their children are computational cells for one finer grid. As a result of this step, the cells which can be coarsened successfully are found. Then, according to the mesh spacing of the grid, the cells’ “meshSpacing” word is determined. Finally, using “compCell” pointer, computational cells for determined mesh spacing is set according to the one level rule. It is important to note that cells in the finer grids are not deleted. By using “compCell” and “meshSpacing” words, these finer cells are ignored for the coarser grids so that new computational cells are created without any deletion. Figure 4.3 illustrates a grid with one cycle of solution adaptation around the RAE 2822 airfoil, which is coarsened to 7 levels. In Table 4.1, the cell numbers are presented according to cell types for all grids used in multigrid scheme. It can be seen that with the increase in mesh spacing, the total coarsening ratio increases significantly whereas the relative coarsening ratio decreases and remains at a value of 1.3.

Table 4.1 Cell numbers of grids used in multigrid for RAE 2822 airfoil

Mesh Spacing	**Out Cell No**	**Cut Cell No**	**Split Cell No**	**Total Cell No**	**Coarsening Ratio Acc. To *h*-grid**	**Coarsening Ratio Acc. To One Finer Grid**
h	5804	525	5	6334	-	-
2*h*	1707	263	4	1974	3.2	3.2
4*h*	708	134	3	845	7.5	2.3
8*h*	326	69	2	397	16.0	2.1
16*h*	234	36	2	242	26.2	1.6
32*h*	163	19	2	184	34.4	1.3
64*h*	129	14	2	145	43.7	1.3
128*h*	98	11	0	109	58.1	1.3

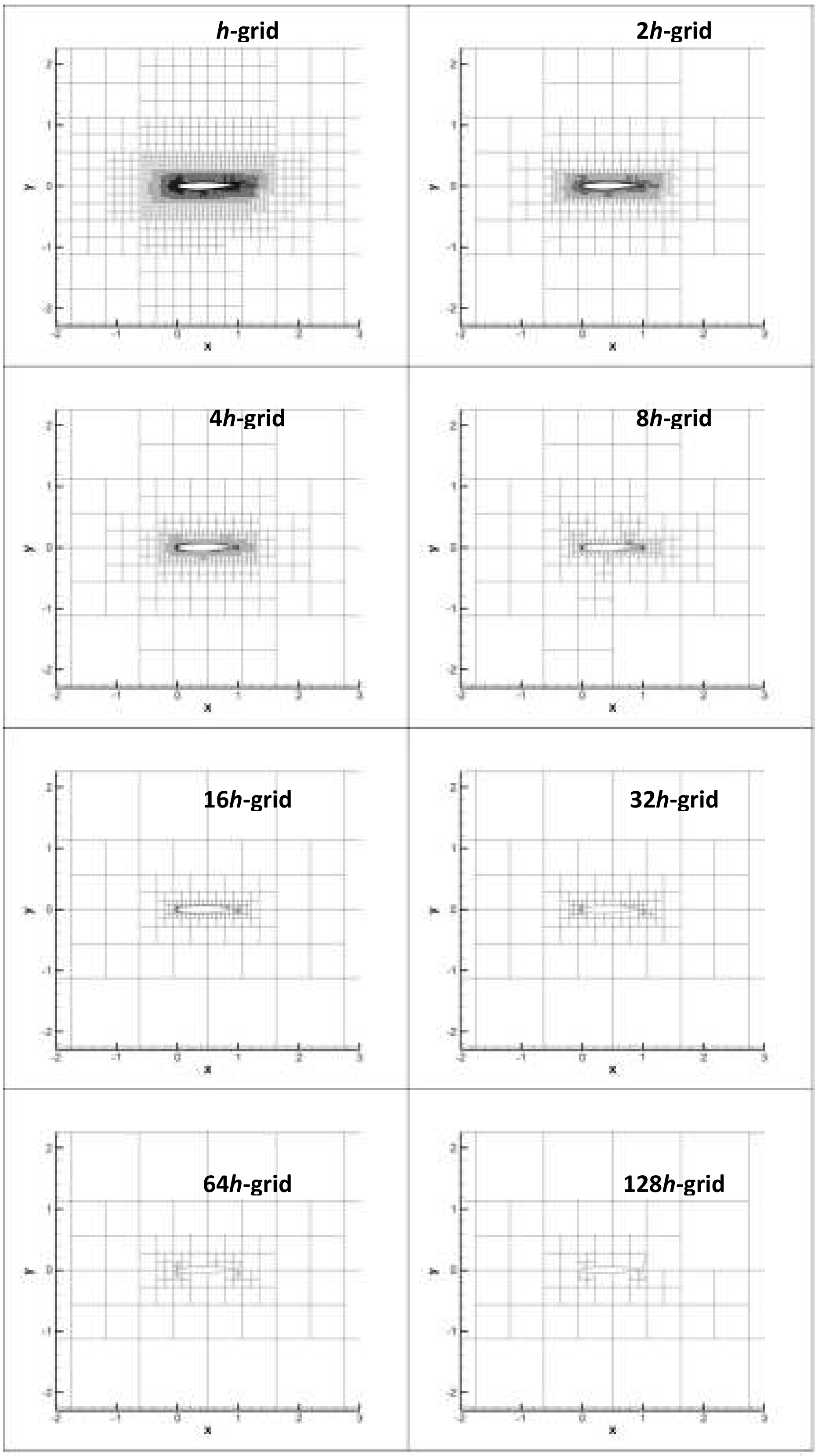
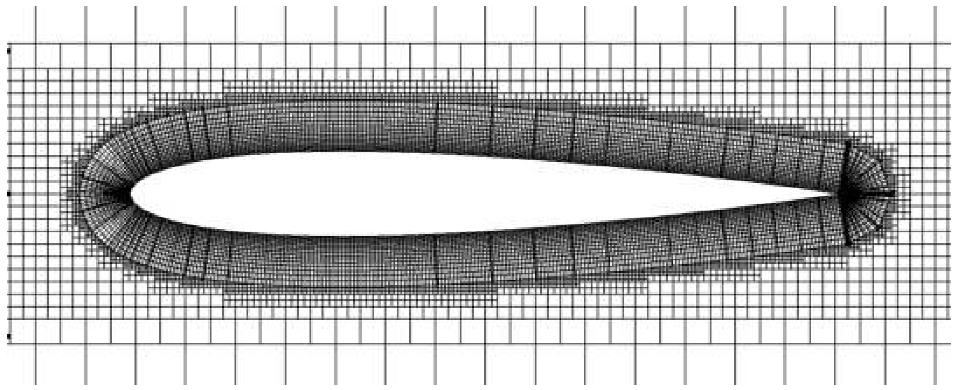
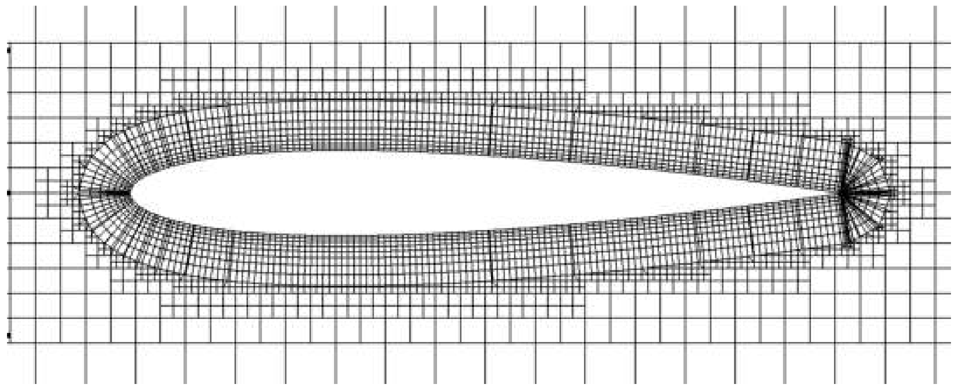


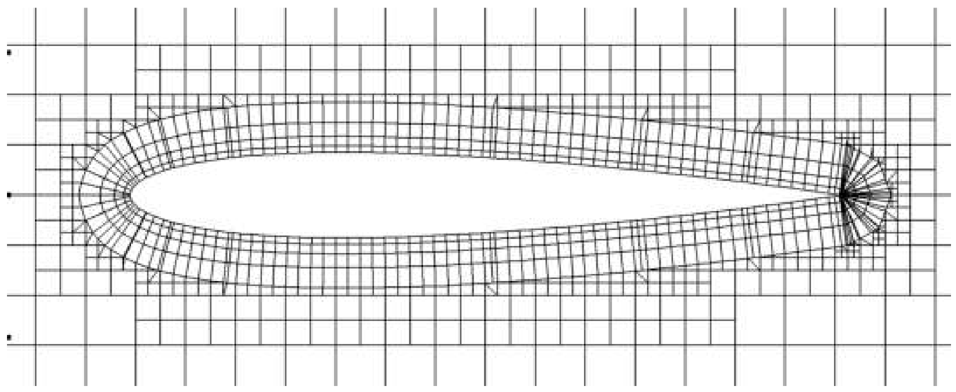
Figure 4.3 Illustration of grids used in multigrid scheme for RAE 2822 airfoil

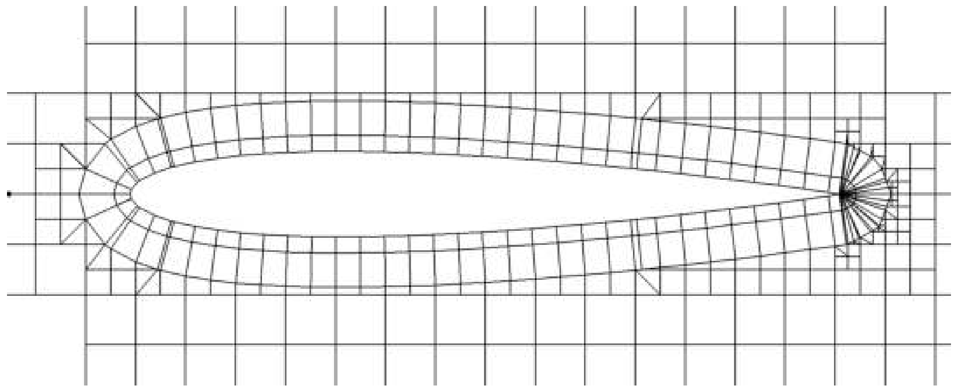
4.2.2 Coarsening of Quad Cells

Coarsening of quad cells are different than the Cartesian cells since quads are formed according to the Cartesian cells. In a hybrid mesh, Cartesian cells are first coarsened at the restriction step. Since cut and split cells are changed compared to the finer grid, quad cells which are generated from these newly coarsened cells near the wall are automatically coarsened. However, this is not sufficient. The row number in the boundary layer is divided by 2 for each mesh spacing so that quad cells are coarsened not only horizontally but also vertically. Since quad cells are coarsened by dividing the row number by 2 at each level, the user-defined row number is restricted such that it must be a power of 2. Moreover, the stretch factor is squared so that the coarsened quad cells cover the finer ones completely. When the row number is 1 for quad cells at a level, then it remains the same for all coarser levels in order not to lose the body-fitted geometry. In Figure 4.4, the hybrid mesh with 16 rows around NACA 0012 airfoil can be seen with the other grids used in the multigrid scheme.


(a) h-grid


(b) $2h$-grid


(c) 4h-grid


(d) 8h-grid

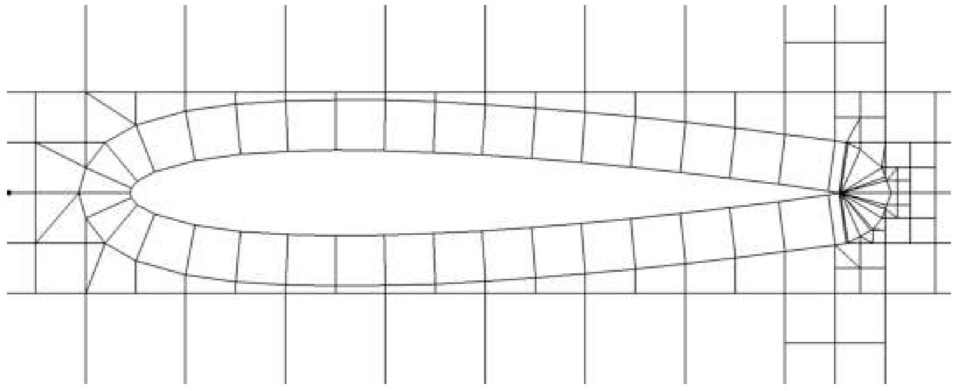

(e) 16h-grid

Figure 4.4 Illustration of hybrid grids used in multigrid for NACA 0012 airfoil

4.3 MULTIGRID EFFECT ON INVISCID FLOW

In this section, the effect of multigrid method is investigated using the transonic flow around RAE 2822 at a Mach number of 0.75 and an angle of attack of 3 degrees. Since the results are discussed in Chapter 5, only the residuals are examined in order to see the effect of multigrid on the convergence time. For the results in this section and in the next section, a work-station is used. This work-station has a four core processor at 2.33 GHz and 32 GB Ram.

Four different problems are discussed in this chapter. 25 cases are tested for these four problems. In the first problem, the solution adaptation is not used and the multigrid level on this coarse mesh are discussed. Secondly, again the level of multigrid is examined at a mesh after the application of five cycles of solution adaptation. Thirdly, the difference between Saw-Tooth and V-Cycle are shown both for solution adapted and non-adapted cases. Finally, the iterations at the steps of the multigrid technique are changed and the results are compared. The results are taken as a form of a data set formed by the logarithm of the normalization of root mean square of continuity residuals and the CPU time. Normalization is done by dividing the root mean square to the difference between the maximum and the minimum continuity residuals.

4.3.1 Level Test Without Solution Adaptation for Inviscid Flow

In the first problem, the first mesh created before the solution is used. In other words, no cycle of solution adaptation is used. Totally 8 tests were done. The number of cells for all test cases is 4055. The solver is iterated until the normalized residual reaches -10. The flux method is approximate Riemann solver of Roe. The iterations for all steps are set to 10 for this problem. Saw-Tooth cycle is used. The only different parameter in test cases is the level number of the multigrid. This number is changed from 0 to 7 in these cases. The results that are obtained, are shown in Table 4.2. In addition, the residuals of these tests with respect to the CPU time are presented in Figure 4.5.

Table 4.2 Level test results without solution adaptation for inviscid flow

Case No	Description	Time (sec)	Speed Up Ratio
1	No multigrid	458	-
2	One level multigrid	298	1.54
3	Two level multigrid	187	2.45
4	Three level multigrid	116	3.95
5	Four level multigrid	85	5.39
6	Five level multigrid	74	6.19
7	Six level multigrid	73	6.27
8	Seven level multigrid	73	6.27

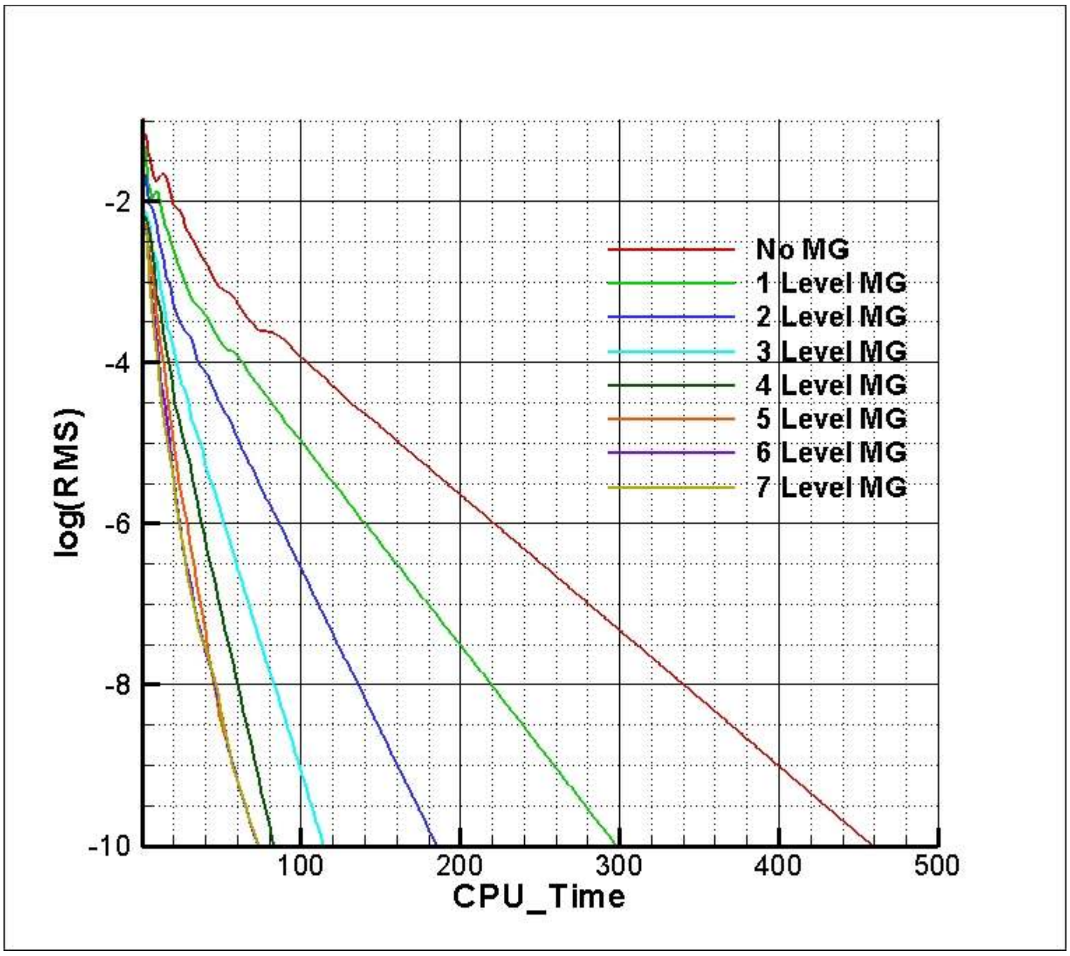


Figure 4.5 Residuals with respect to CPU time using a non-solution adapted mesh around RAE 2822 airfoil

As shown above, the solution speeds up with the increase in level. However, when the last three cases having levels of 5, 6 and 7 are considered, it can be seen that there is not too much difference between them in terms of the convergence time. Nonetheless, the maximum speed up ratio is obtained from the seventh and the eighth cases, which is 6.27.

4.3.2 Level Test With Solution Adaptation for Inviscid Flow

In this problem, the solution adapted mesh is used while examining the effect of levels. Five cycles of solution adapted mesh are used. The cases and the results are tabulated in Table 4.3.

Table 4.3 Level test results with solution adaptation for inviscid flow

Case No	Description	Time (s)	Speed Up Ratio
9	No multigrid	36965	-
10	One level multigrid	23327	1.58
11	Two level multigrid	14356	2.57
12	Three level multigrid	7510	4.92
13	Four level multigrid	5631	6.56
14	Five level multigrid	2943	12.56
15	Six level multigrid	2423	15.26
16	Seven level multigrid	2177	16.98

As it can be seen, the level increase has a great effect on the acceleration for solution adapted mesh. Up to five levels, the speed up ratio is doubled approximately. After that, while the increase in the ratio is decreasing, the ratio is continuously raising so that the maximum ratio, which is nearly 17, is obtained from the final case having a level of seven.

In Figure 4.6, the residuals are expressed with respect to the CPU time graphically. One can see the closeness of residuals for the 14th, 15th and 16th cases. If one consider the results in the previous section, the level increase brings an enormous acceleration rate especially for solution-adapted grids while non-solution adapted grids are accelerated at a maximum ratio of 7 with the level increase.

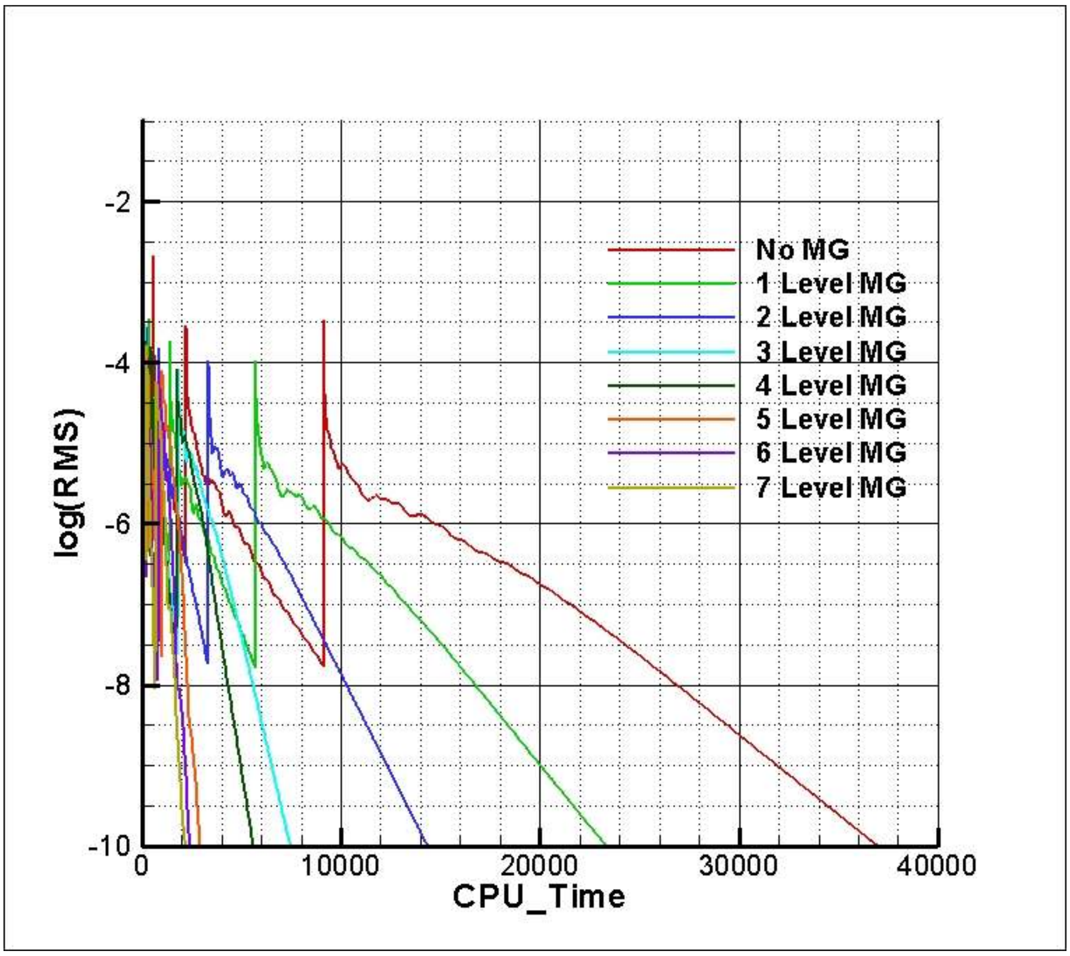


Figure 4.6 Residuals with respect to CPU time using a solution adapted mesh around RAE 2822 airfoil

4.3.3 Cycle Test for Inviscid Flow

In this problem, two different cycles, namely Saw-Tooth and V-Cycle, are tested with and without applying solution adaptation. For all cases, seven level of multigrid is used. In the first two cases, the solution adaptation is not used. Then, for the last two cases, five cycles of solution refinement are applied to the mesh. The results obtained for these cases are presented in Table 4.4.

Table 4.4 Cycle test results for inviscid flow

Case No	Description	Time (s)	Speed Up Ratio
17	Saw-Tooth without solution adaptation	73	6.27
18	V-Cycle without solution adaptation	98	4.67
19	Saw-Tooth with five cycles of solution adaptation	2177	16.98
20	V-Cycle with five cycles of solution adaptation	2365	15.63

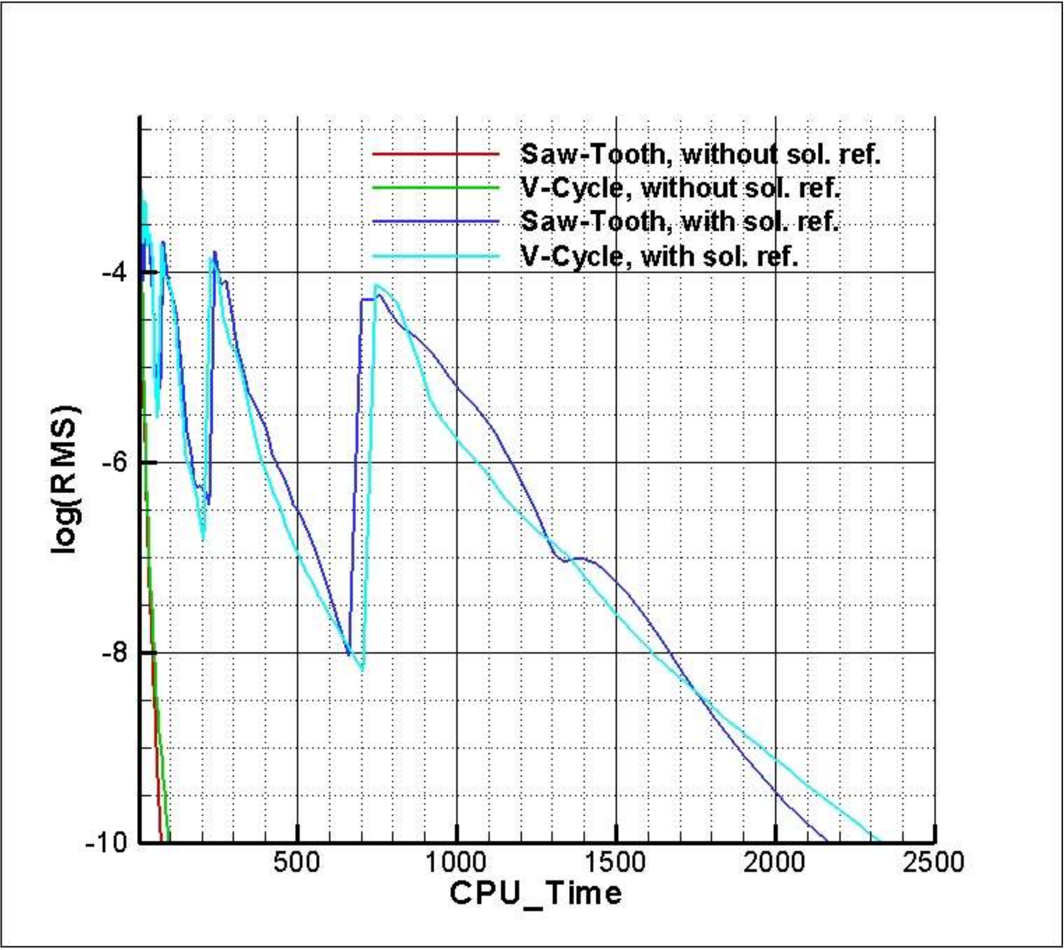


Figure 4.7 Residuals with respect to the CPU time for cycle testing around RAE 2822 airfoil

As one can see in Figure 4.7, V-Cycle leads to a more slower convergence compared to Saw-Tooth cycle for both grids with and without solution adaptation. As expected, the difference between these cycles are not much higher. The reason, why V-Cycle converges slower, may be the increase in the required memory since the forcing functions must be stored in order to use them for the iterations at the prolongation step.

4.3.4 Iteration Test for Inviscid Flow

Another important input, which affects the convergence time while applying multigrid technique is the iteration number in the multigrid steps. In this section, the iteration numbers are changed using five cases in order to find the optimum numbers for inviscid flow. The same grid with five cycles of solution adaptation and the same inputs are used for all cases so that the pure effect of iteration numbers can be observed.

Table 4.5 Iteration test results for inviscid flow

Case No	**Description**	**Time (s)**	**Speed Up Ratio**
21	5 iterations	2165	17.07
22	10 iterations	2177	16.98
23	15 iterations	2182	16.94
24	20 iterations	2418	15.29
25	25 iterations	2728	13.55

In the fine-grid, mid and final steps, the same number of iterations are used. As shown in Table 4.5 and Figure 4.8, the decrease in the iteration number enables faster convergence rates. Thus, the optimum number for this inviscid flow is 5 iterations at each step, providing a speed up ratio of 17.07. However, the amount of increase is just a few seconds especially between cases 21, 22 and 23. One can infer that, it is not necessary to use a large number of iterations since the low frequency errors can be eliminated using a few iterations for inviscid flows.

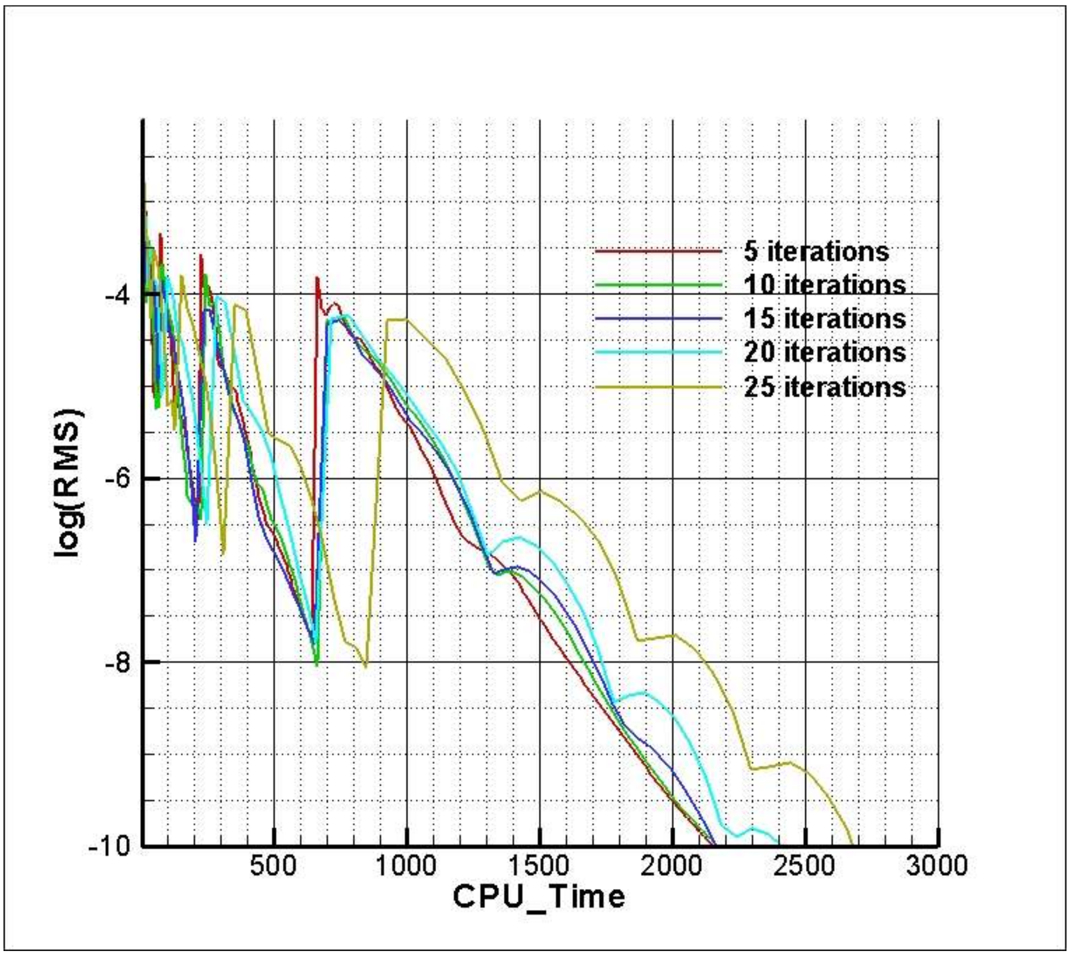


Figure 4.8 Residuals with respect to the CPU time for testing the number of iterations around the RAE 2822 airfoil

4.4 MULTIGRID EFFECT ON VISCOUS FLOW

In this section, the effect of multigrid is investigated using the transonic flow around NACA 0012 at a Mach number of 0.5, an angle of attack of 0 degree and a Reynolds number of 5000. Only the residuals are presented in this section since the results are discussed in the next chapter.

This time, five problems are considered. Firstly, the grid without applying solution adaptation is used and the effect of multigrid level number is examined with several test cases. Secondly, the level number is tested using a mesh where three cycles of solution adaptation is applied. Thirdly, the difference in cycles are presented both for solution adapted and non-solution adapted meshes. Fourthly, the iteration numbers in the steps are changed to find the optimum numbers for three cycles of solution adapted mesh. Finally, the effect of multigrid is discussed for an hybrid mesh.

4.4.1 Level Test without Solution Adaptation for Viscous Flow

In the first problem, eight cases are tested. The only variable parameter is the multigrid level number. The purpose is to see the effect of level increase when solution adaptation is not used for viscous flows. While the iteration number at fine grid step is 15, 10 is used for the other steps. Saw-Tooth cycle is used for all cases. As inviscid flux method, AUSMV is used. As convergence criteria, the solver is iterated until the normalized continuity residual reaches -9. As shown in Table 4.6 and Figure 4.9, the level increase does not provide a significant acceleration rate even if seven levels are used. While a speed up ratio up to 6 can be obtained for the inviscid flow using a mesh without solution adaptation , a maximum speed up ratio of 1.5 is obtained for the laminar flow.

Table 4.6 Level test results without solution adaptation for viscous flow

Case No	Description	Time (s)	Speed Up Ratio
1	No multigrid	2958	-
2	One level multigrid	2480	1.19
3	Two level multigrid	2364	1.25
4	Three level multigrid	2280	1.30
5	Four level multigrid	2232	1.33
6	Five level multigrid	2251	1.31
7	Six level multigrid	2091	1.41
8	Seven level multigrid	2002	1.48

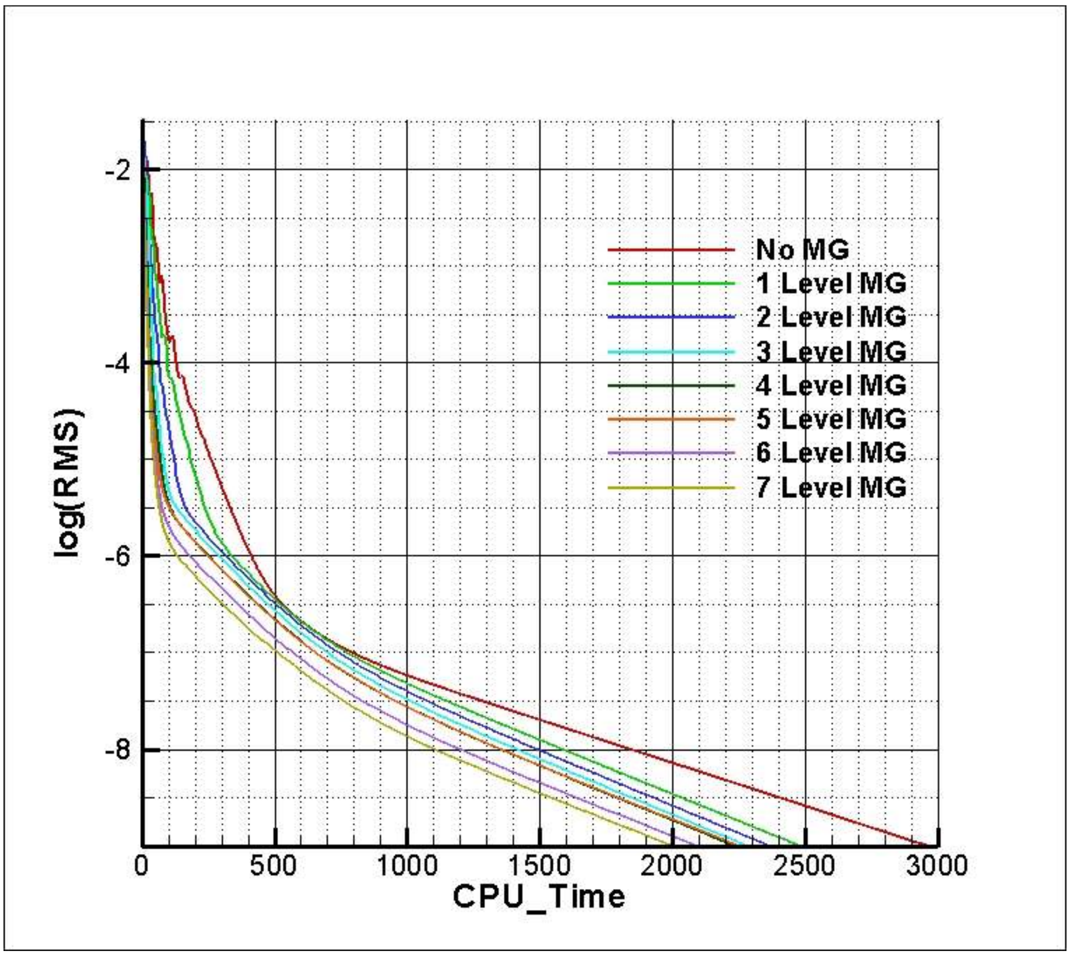


Figure 4.9 Residuals with respect to CPU time using a non-solution adapted mesh around NACA 0012 airfoil

<u>4.4.2 Level Test with Solution Adaptation for Viscous Flow</u>

The effect of level increase is also tested in this problem. However, the grid is changed such that three cycles of solution adaptation is applied. The results are tabulated in Table 4.7.

Table 4.7 - Level test results with solution adaptation for viscous flow

Case No	**Description**	**Time (s)**	**Speed Up Ratio**
9	No multigrid	67660	-
10	One level multigrid	42862	1.58
11	Two level multigrid	23558	2.87
12	Three level multigrid	13169	5.14
13	Four level multigrid	10838	6.24
14	Five level multigrid	9859	6.86
15	Six level multigrid	10284	6.58
16	Seven level multigrid	9582	7.06

As it can be seen, level increase leads to a higher speed up ratio for solution adapted mesh. Except the 15th case, the convergence time decreases with the increase in the level number. In comparison to the previous problem, one can say that the multigrid is able to accelerate the solution time significantly by increasing the level number. While the ratio is around 1.5 for the coarsest mesh, it can reach 7 for finer meshes, as one can see in Table 4.7 and Figure 4.10.

In addition, the increase after five levels does not provide a significant difference in terms of the convergence time. While the solution converges after 9859 seconds for the 14th case, the final case provides a convergence in 9582 seconds. The difference is small if one compares this amount with 67660 seconds which is obtained for the case without the application of multigrid.

If the finer mesh while solving the inviscid flow is reconsidered, it can be concluded that the multigrid does not provide a sufficient acceleration since a speed up ratio of 17 can be obtained for inviscid flows. However, it must be considered that the viscous flows converge so slowly in contrast to the inviscid flows. Thus, a speed up ratio of 7 gives a significant improvement in the convergence time.

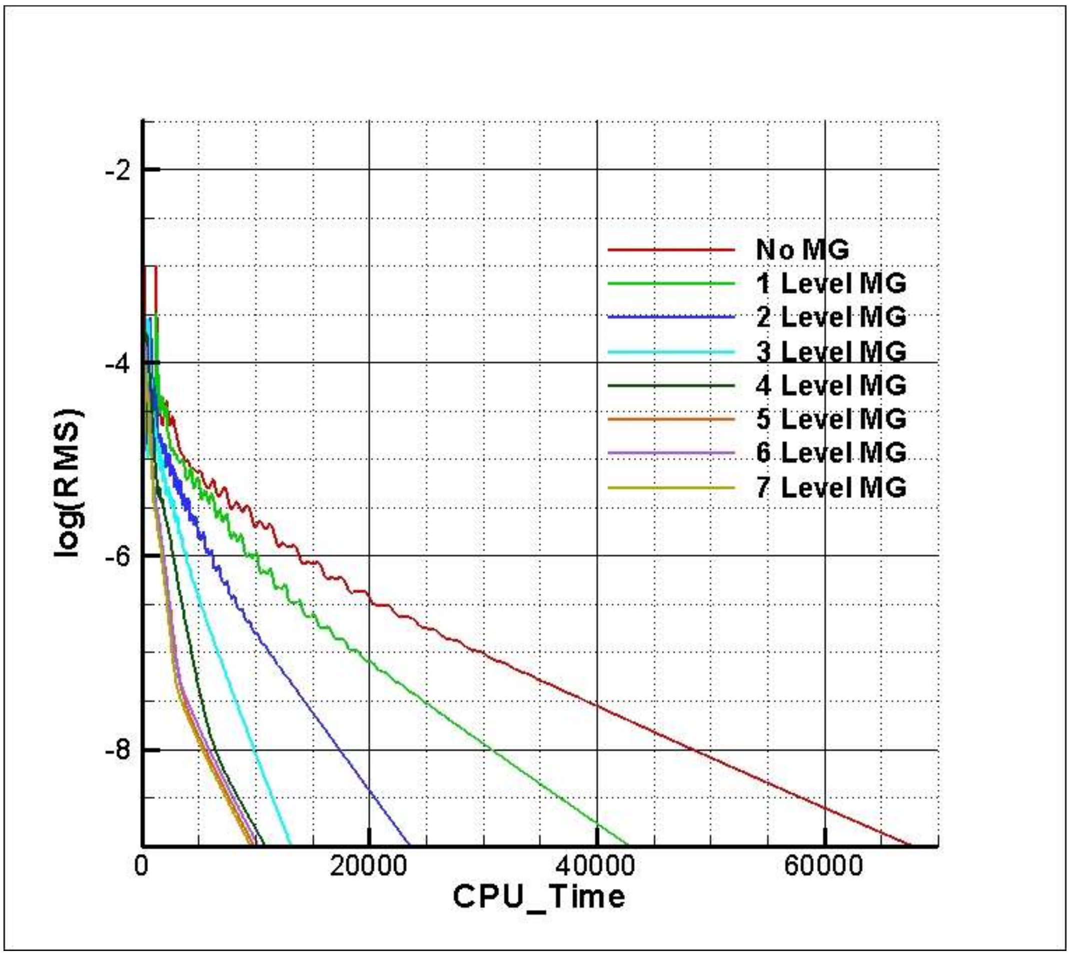


Figure 4.10 Residuals with respect to the CPU time using a solution adapted mesh around NACA 0012 airfoil

<u>4.4.3 Cycle Test for Viscous Flow</u>

In this section, V-Cycle and Saw-Tooth cycles are compared using a level of 7 for grids with and without solution adaptation. The results obtained for this problem are tabulated in Table 4.8.

Table 4.8 Cycle test results for viscous flow

Case No	Description	Time (s)	Speed Up Ratio
17	Saw-Tooth without solution adaptation	2002	1.48
18	V-Cycle without solution adaptation	2248	1.32
19	Saw-Tooth with three cycles of solution adaptation	9582	7.06
20	V-Cycle with three cycles of solution adaptation	14077	4.81

The results show that the V-Cycle converges slower than the Saw-Tooth cycle for both grids, as shown in Figure 4.11. While there is a slight difference in the grid without solution adaptation, it becomes larger when finer mesh is used.

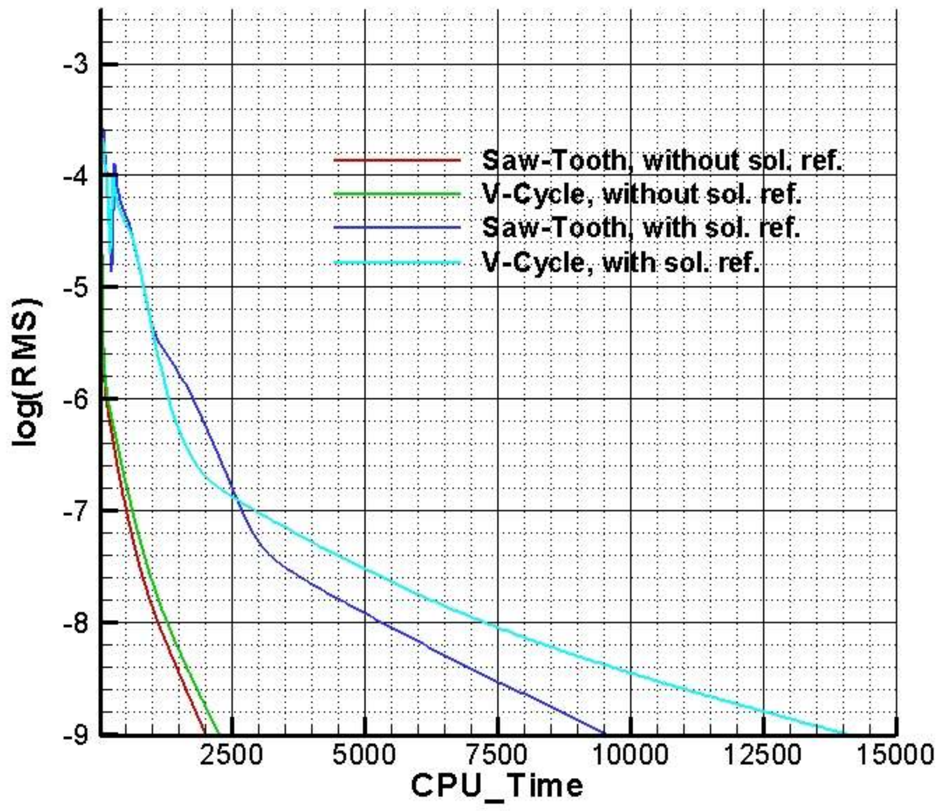


Figure 4.11 Residuals with respect to CPU time for cycle testing around NACA 0012 airfoil

4.4.4 Iteration Test For Viscous Flow

In this problem, the grid that three cycles of solution adaptation is applied is used to determine the optimum iteration numbers at the steps of multigrid. Thus, six cases are tested. On the contrary to the inviscid flow problem, all of the cases does not have the same iteration number at all steps. Some cases have more iterations for fine grid step. The case descriptions and the results are presented in the Table 4.9.

As one can see in Figure 4.12, the 26th case gives the best results for this viscous flow. Besides, the 22th case is not very much different than the 26th case. One can infer that it is necessary to iterate more in order to eliminate low frequency errors for viscous flows compared to inviscid flows.

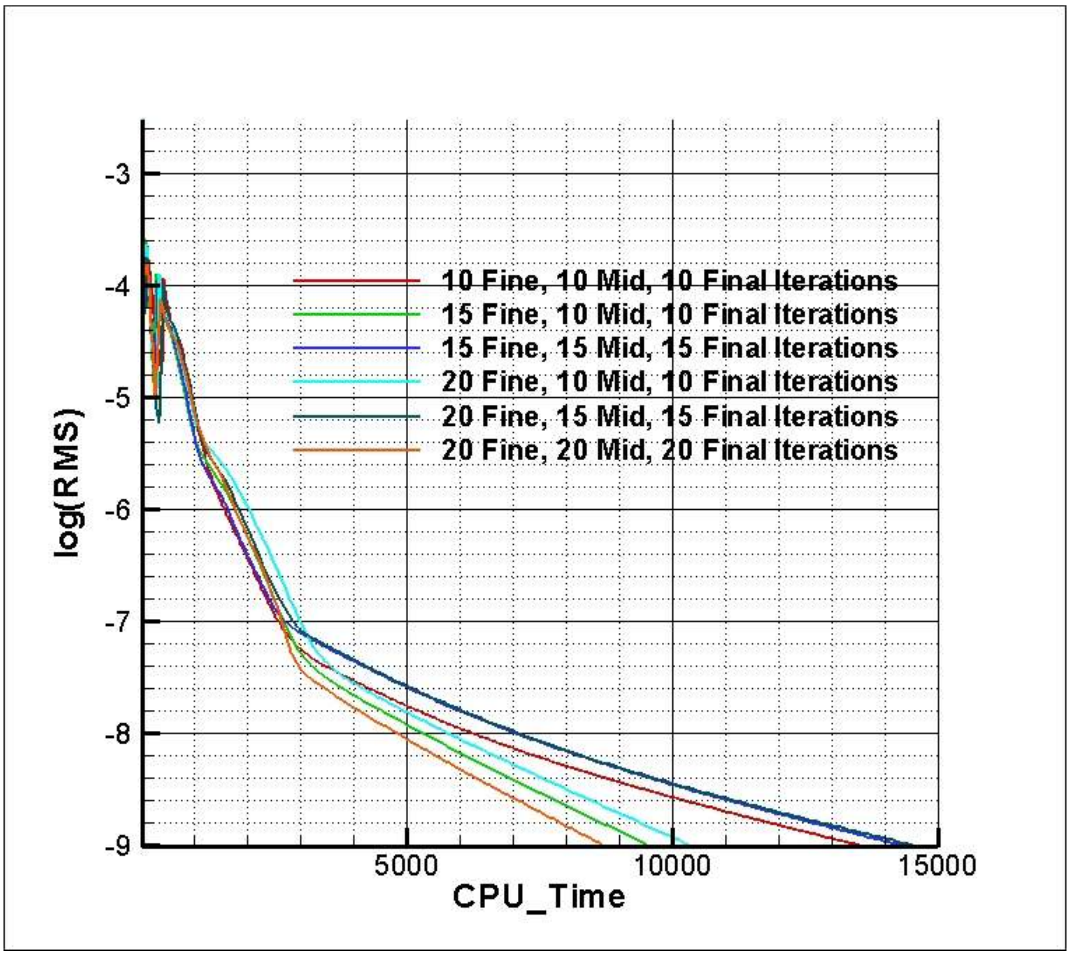


Figure 4.12 Residuals with respect to CPU time for iteration testing around NACA 0012 airfoil

Table 4.9 Iteration test results for viscous flow

Case No	Description	Time (s)	Speed Up Ratio
21	10 fine, 10 intermediate and 10 final iterations	13549	4.99
22	15 fine, 10 intermediate and 10 final iterations	9582	7.06
23	15 fine, 15 intermediate and 15 final iterations	14346	4.72
24	20 fine, 10 intermediate and 10 final iterations	10346	6.54
25	20 fine, 15 intermediate and 15 final iterations	14572	4.64
26	20 fine, 20 intermediate and, 20 final iterations	8773	7.71

4.4.5 Hybrid Mesh Test for Viscous Flow

The final problem is the test of multigrid on hybrid meshes. As the quad cells are used in the boundary layer, the number of cells increases significantly. For example, while the mesh without solution adaptation has a cell number of 4040, 14562 cells are used for a hybrid mesh when solution adaptation is not used. In other words, even the coarsest mesh includes a large number of cells with very small cells in the boundary layer. With this huge amount of cells and the large size differences, the convergence time is very long compared to the mesh formed only by Cartesian cells. Thus, multigrid is very important to decrease the solution time.

For this problem, two cases are tested; multigrid is not used for the first one and a seven level multigrid is used for the second one. Iteration numbers are set to 30 at each step since a lower number causes divergence in the solution. It can be inferred that while using very small cells in the boundary layer, the iteration step needs to be larger than the other problems in order to decrease the low frequency errors. Saw-Tooth cycle is used in the second case. The results are presented in Table 4.10 in written and in Figure 4.13.

Table 4.10 Hybrid mesh test results for viscous flow

Case No	Description	Time (s)	Speed Up Ratio
27	No multigrid	256049	-
28	Seven level multigrid	55014	4.65

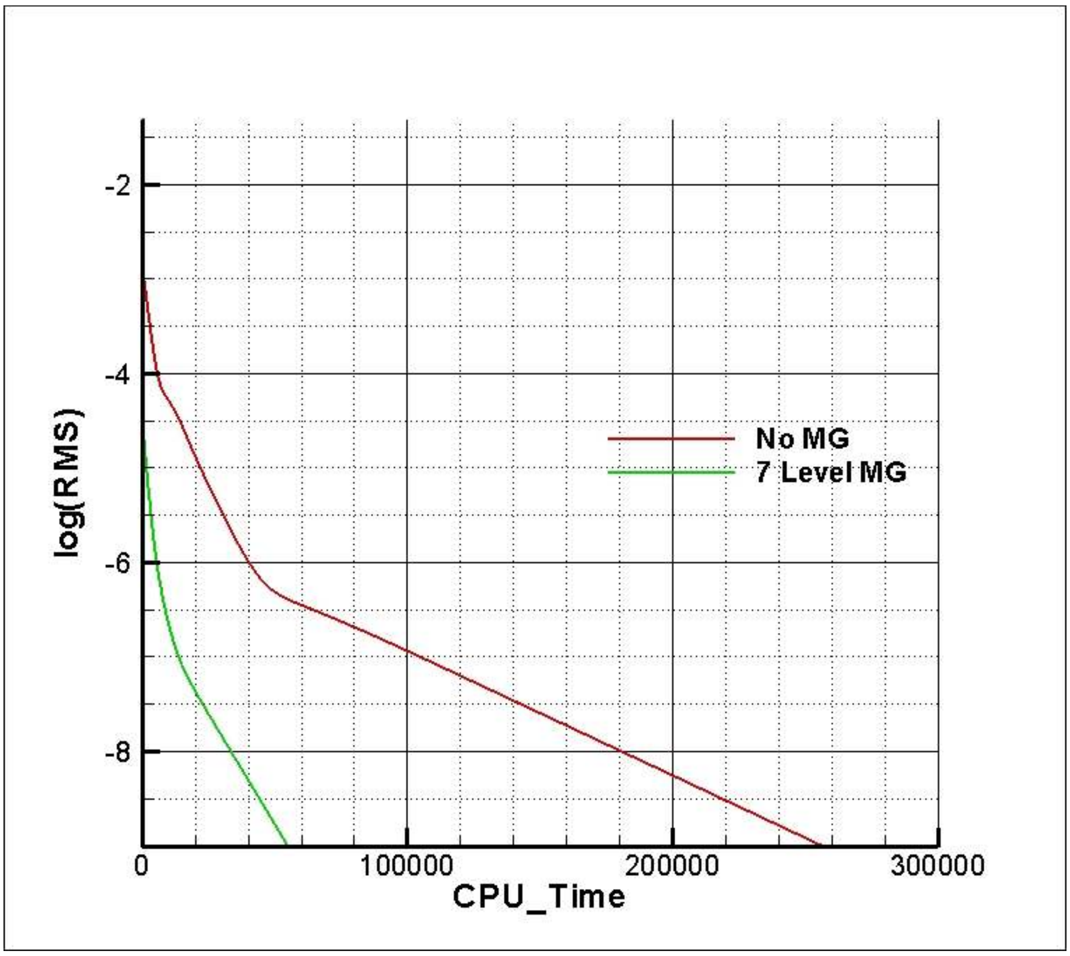


Figure 4.13 Residuals with respect to the CPU time for hybrid mesh testing around NACA 0012 airfoil

The case for which no multigrid is used converges approximately 8 times slower than the case for which quad cells are not used. Although the effect of multigrid is significantly more for finer meshes as before, a seven level multigrid leads to a speed up ratio of 4.65 for hybrid mesh and a significant difference occurs between the hybrid and Cartesian grids when solution adaptation is not used, as expected. However, the multigrid effect on solution adapted hybrid grids cannot be tested due to some problems regarding multigrid usage on hybrid meshes when solution adaptation is applied.

CHAPTER 5

RESULTS AND DISCUSSIONS

In this chapter, the developed code is validated and verified both for inviscid and viscous laminar flows with some tests. These tests are divided into three sub groups. First, inviscid flow around an airfoil is tested for two different cases. One of the cases is a transonic flow, while the other one is a supersonic flow. Secondly, one subsonic flow and one transonic flow at low Reynolds numbers are analyzed in order to verify the code for laminar flows. Finally, in order to show the validity of the hybrid mesh, one high Reynolds number flow is examined. Since the flow is turbulent for this case and the code is not designed to solve turbulence, the results are not expected to be accurate. For all problems, the results are compared with the data found in the literature which is numerical or experimental. All results are obtained by using a work-station. This computer has a single processor with 4 cores each at a speed of 2.33 GHz and 32 GB Ram. The operating system is Microsoft Windows XP 64Bit Edition.

5.1 INVISCID FLOW

In this section, two problems are considered. While one of them is a transonic flow, the other one is a supersonic flow around a one-element airfoil. The shock locations and strengths of these shocks are compared using pressure distribution graphs with the data available in the literature. The specifications of these problems are depicted in Table 5.1.

Table 5.1 Test problems for inviscid flow

SECTION	**AIRFOIL**	M_∞	α **(degrees)**
5.1.1	RAE 2822	0.75	3
5.1.2	NACA 0012	1.2	7

5.1.1 Transonic Flow Around RAE 2822

The first problem is a transonic flow around a non-symmetric airfoil, i.e. RAE 2822, with a Mach number of 0.75 and an angle of attack of 3 degrees. The reason why transonic flow is selected is to demonstrate that by using Cartesian approach, locations of shocks and strengths of shocks can be captured using a sufficiently finer mesh around the shock.

Table 5.2 Common properties for transonic inviscid flow

MESH INPUTS	
Outer boundary size factor	18

SOLUTION INPUTS	
Flux method	Roe
Refinement cycle interval	20
log(RMS) for convergence	-10
CFL safety factor	1

MULTIGRID INPUTS	
Multigrid type	Saw-Tooth
Multigrid level	7
Fine grid iterations	10
Intermediate step iterations	10
Final grid iterations	10

The results are obtained using six cases. After that, they are compared with the results from reference [13], which uses an O-type mesh with 20480 cells. For all cases, the outer boundary is located 18 chords ahead of the airfoil. As a flux calculating technique, approximate Riemann solver of Roe is used. The solver is iterated until the logarithm of root mean square of normalized continuity residual reaches -10. In addition, a 7 level multigrid is used to accelerate convergence time. The common properties for the test cases in this problem are tabulated in Table 5.2.

Two parameters are changed at the test cases. One of them is the order of the scheme and the other one is the number of refinement cycles at solution adaptation. Gradient limiting is used in the cases where second order scheme is used to calculate the face fluxes. Table 5.3 shows the calculated lift and drag coefficients for these cases and the reference case as well as the convergence time. While attempting to obtain results using the second order scheme without gradient limiters, the pressure became negative and results are not acquired. The reason is that the second order scheme leads to fluctuating residuals. While the limiters prevent the excessive change in residuals at each iteration, the solutions without using these limiters can cause to instability problems.

Table 5.3 Comparison of results for transonic inviscid flow around RAE 2822 airfoil

Case No	**Description**	C_D	C_L	**Cell Number**	**Time**
1	No solution refinement with 1st order	0.0663	0.6725	4055	0 hour 1 minute 13 seconds
2	No solution refinement with 2nd order	0.0638	0.7089	4055	0 hour 9 minutes 19 seconds
3	Three solution refinement with 1st order	0.0444	0.9267	18880	0 hour 6 minutes 31 seconds
4	Three solution refinement with 2nd order	0.0446	0.9377	17647	0 hour 24 minutes 53 seconds
5	Five solution refinement with 1st order	0.0424	0.9649	61526	0 hour 36 minutes 16 seconds
6	Five solution refinement with 2nd order	0.0427	0.9725	57496	1 hour 25 minutes 5 seconds
7	Reference results [13]	0.0448	1.1044	20480	-

As shown in Table 5.3, the results are getting closer to the reference results with the increase in the solution refinement cycle. In addition, the second order usage leads to more accurate results for a fixed solution adaptation cycle. However, with the second order scheme, the convergence time increases greatly. For example, the increasing ratio is about 9 if solution refinement is not used, while it is approximately 2.5 for five cycles of solution refinement. Nonetheless, the best result is obtained for the sixth case, where five cycles of solution adaptation and second order scheme is used. The pressure distributions of all the cases compared to the reference case are shown in Figure 5.1.

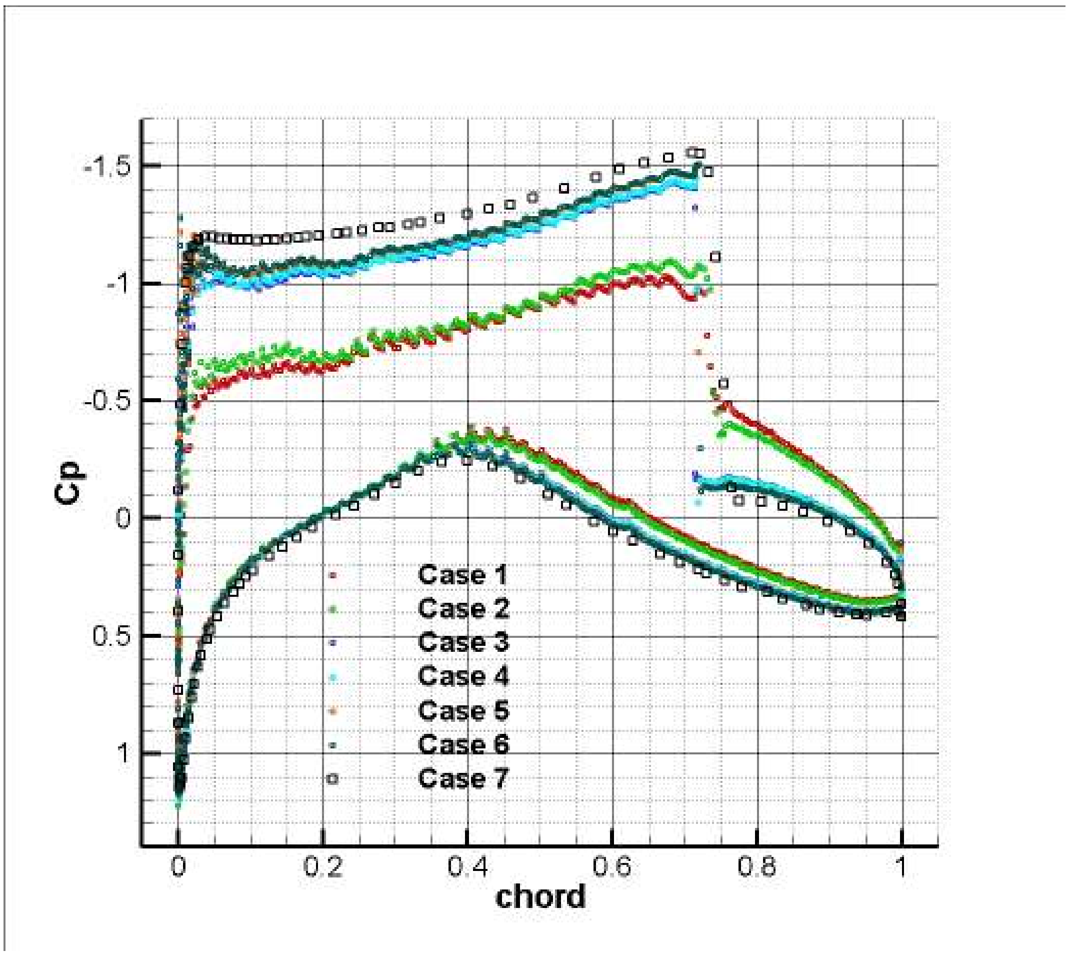


Figure 5.1 Pressure coefficient distribution for transonic inviscid flow around RAE 2822 airfoil

As observed from Figure 5.1, the pressure distribution at the upper surface, where a shock exists, are underestimated for all cases. One can say that low number of solution refinement cycles leads to big difference especially at upper surfaces. However, slight differences are occurred at the cases where a solution refinement cycle number greater than 3 is used. Nonetheless, with increasing cycle number, the solution gets closer to the reference data at a more or less amount. In addition, the shock locations are not captured well by all of the cases. Yet, the distribution at the below surface are in a good agreement with the reference data. The grids used for Case 3 and Case 5 are shown in Figure 5.2.

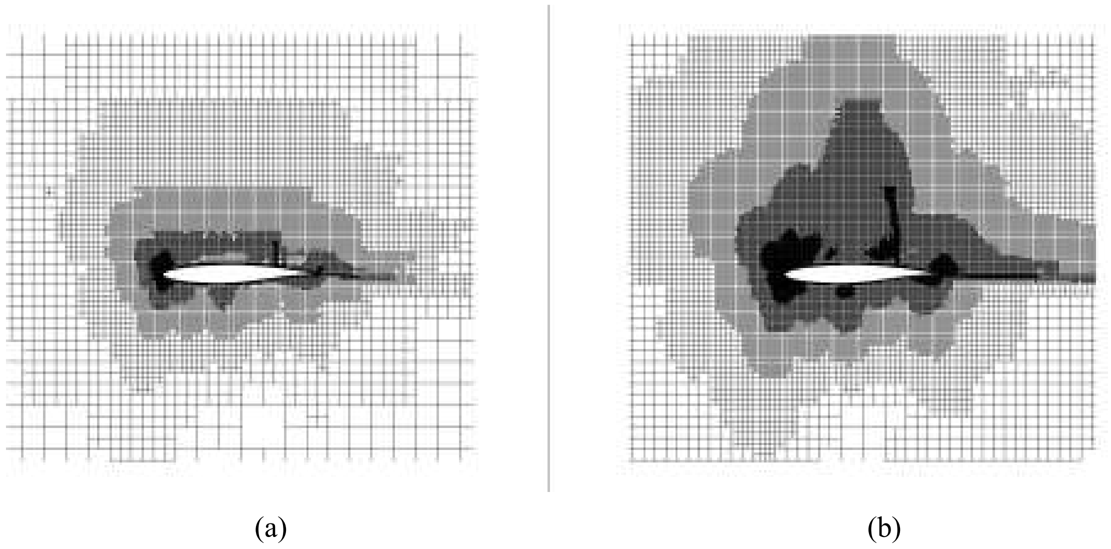


Figure 5.2 The grids around the RAE 2822 for Case 3 (a) and Case 5 (b) for the transonic inviscid flow

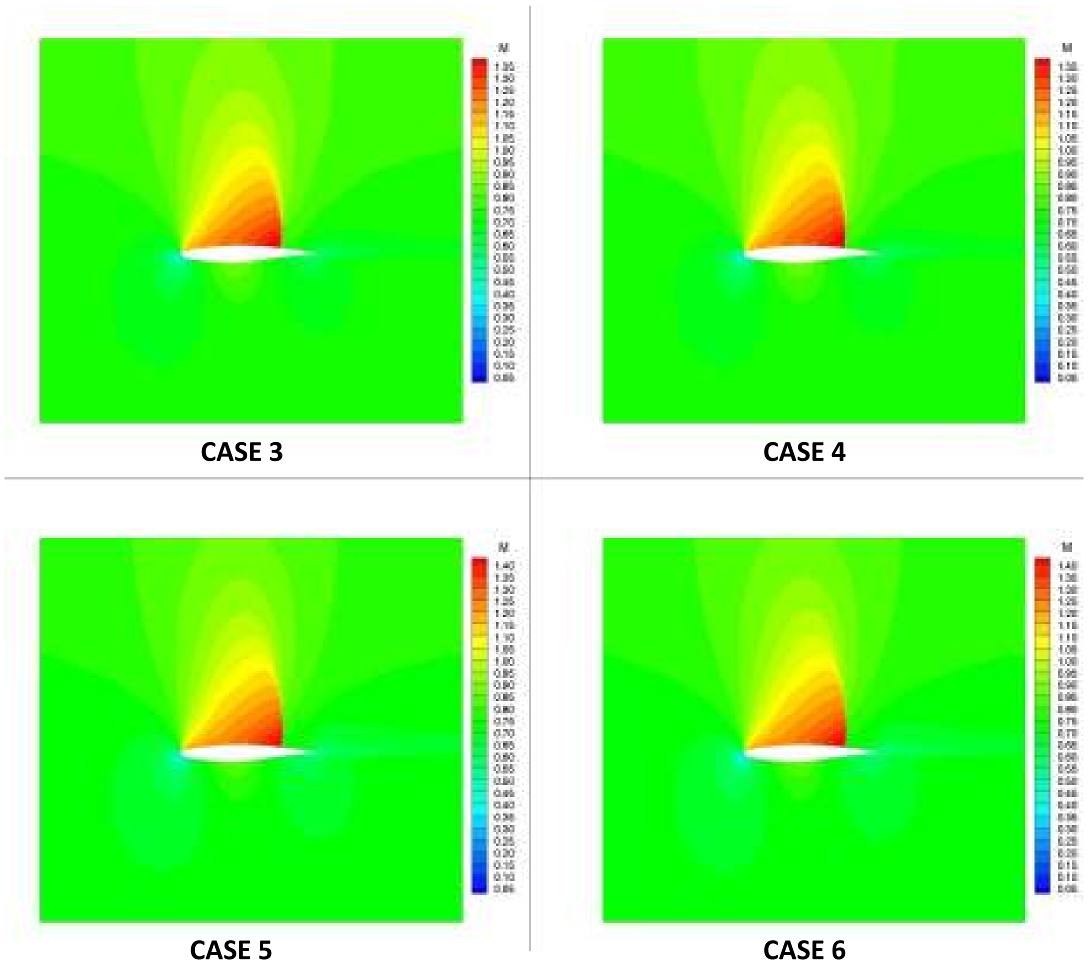


Figure 5.3 Mach contours of cases where solution adaptation is applied around RAE 2822 for the transonic inviscid flow

Besides the importance of cycle increase, the second order scheme usage leads to slightly more accurate results compared to the first order scheme for all cases, as seen in Table 5.3 and Figure 5.1. It can be seen that the best result is obtained from the sixth case again. Since it is difficult to see the effect of the order of scheme in Figure 5.1 for the cases where solution adaptation is applied, Mach and pressure contours are presented in Figures 5.3 and 5.4 both for three cycles and five cycles of solution adaptation. One can see the slight differences between the abreast figures. In the first row, cases with three cycles of solution refinement are compared while five cycles are compared in the second row.

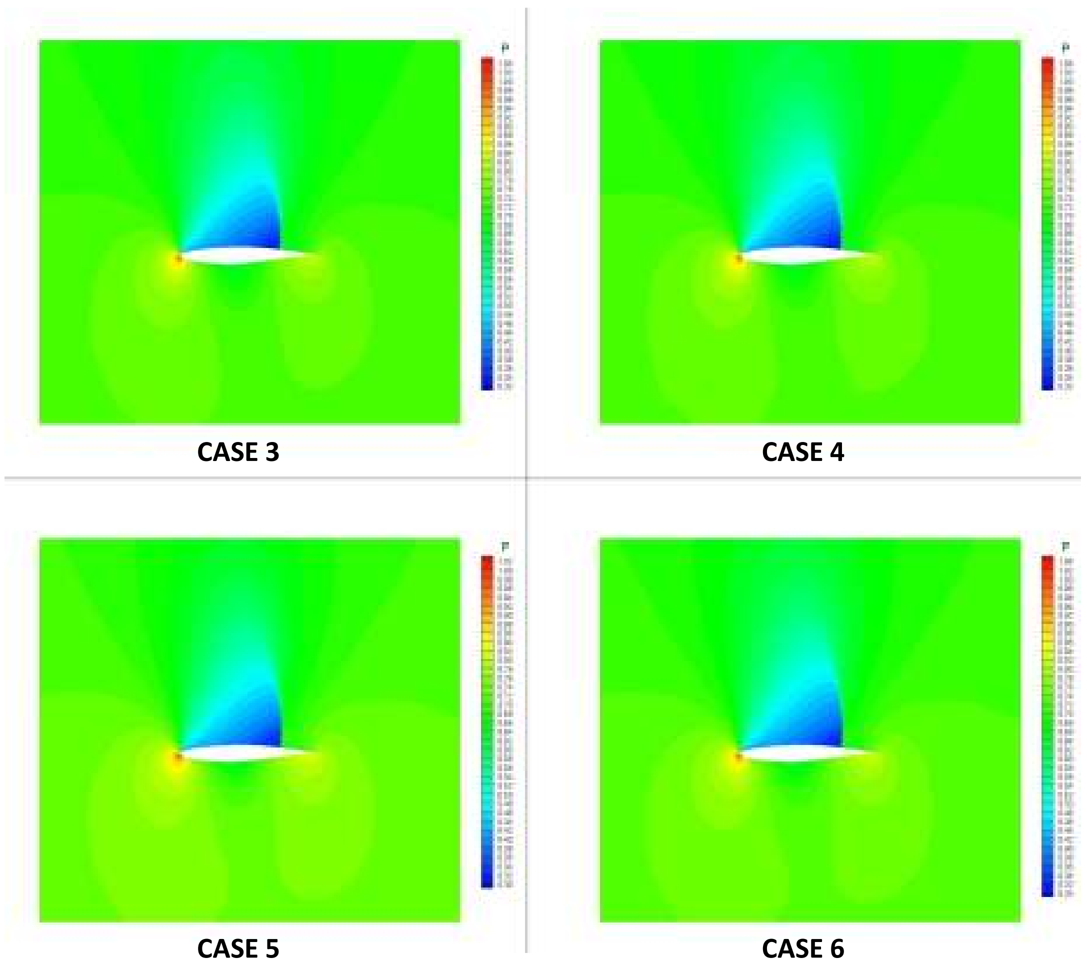


Figure 5.4 Pressure contours of cases where solution adaptation is applied around RAE 2822 for the transonic inviscid flow

5.1.2 Supersonic Flow Around NACA 0012

The second problem is a supersonic flow around a symmetric airfoil at a Mach number of 1.2 and an angle of attack of 7 degrees. The aim is to show that bow and oblique shocks can be captured accurately.

Eight tests were carried out for this problem. The far-field boundary is located 18 chords ahead of the airfoil for all cases. In addition, first order scheme is used. Multigrid technique is also used with seven levels. The only changing inputs are the solution refinement cycle and the flux calculation method. One reference case is used to compare the test cases.

Table 5.4 Common properties for supersonic inviscid flow

MESH INPUTS	
Outer boundary size factor	18

SOLUTION INPUTS	
Order of scheme	1
Refinement cycle interval	20
log(RMS) for convergence	-10

MULTIGRID INPUTS	
Multigrid type	Saw-Tooth
Multigrid level	7
Fine grid iterations	10
Intermediate step iterations	10
Final grid iterations	10

The code has four different inviscid flux calculation methods. In this problem, these techniques are compared. First, these techniques are compared without applying the solution adaptation since it is easy to compare by looking at the pressure distribution along the chord length. Second, four cycles of solution adaptation are used while obtaining results with different flux methods. The aim is to show that all methods give accurate results compared to the reference data [13], which uses an O type grid with 20480 cells. Below, one can see the cases and the results obtained from these cases.

Table 5.5 Comparison of results for supersonic inviscid flow around NACA 0012 airfoil

Case No	Description	C_D	C_L	Cell Number	Time	CFL
1	No solution refinement with AUSM flux method	0.1688	0.5421	4040	1 minute 35 seconds	0.9
2	No solution refinement with AUSMD flux method	0.1658	0.5253	4040	1 minute 21 seconds	1
3	No solution refinement with AUSMV flux method	0.1684	0.5333	4040	1 minute 8 seconds	1
4	No solution refinement with Roe flux method	0.1648	0.5220	4040	1 minute 18 seconds	1
5	Four solution Refinement with AUSM flux method	0.1603	0.5218	23753	15 minutes 59 seconds	0.9
6	Four solution refinement with AUSMD flux method	0.1599	0.5203	25362	17 minutes 17 seconds	1
7	Four solution refinement with AUSMV flux method	0.1608	0.5209	25331	15 minutes 8 seconds	1
8	Four solution refinement with Roe flux method	0.1595	0.5193	25178	21 minutes 15 seconds	1
-	Reference results [13]	0.1538	0.5138	20480	-	-

In the cases, where solution adaptation is not used, Roe's flux calculation method gives the closest result to the reference data. It overestimates the drag coefficient by 7.1 % and the lift coefficient by 1.6 %. On the other hand, AUSM gives the worst result among the four cases. Overestimating percent is 5.5% at lift coefficient and 9.8% at drag coefficient. All the four cases converge at approximately the same time. When solution adaptation is applied, four methods approach to the reference results. While the minimum overestimating percent is obtained from Roe's method, which is 3.7 % for drag and 1.15 % for lift coefficient, AUSM gives the maximum overestimating percent of 1.6 % for lift and AUSMD gives a maximum percent of

4.6 % for drag coefficient. As a result, the solution adaptation plays an important role to obtain accurate results for all flux methods in this problem. Moreover, one can infer that the difference in results between flux calculation techniques diminishes with the increase in the cycle of solution adaptation. In Figure 5.5, the pressure distribution of cases where solution adaptation is not applied are presented in comparison with the reference data. Then, Figure 5.6 gives the distribution of pressure for cases having four cycles of solution adaptation.

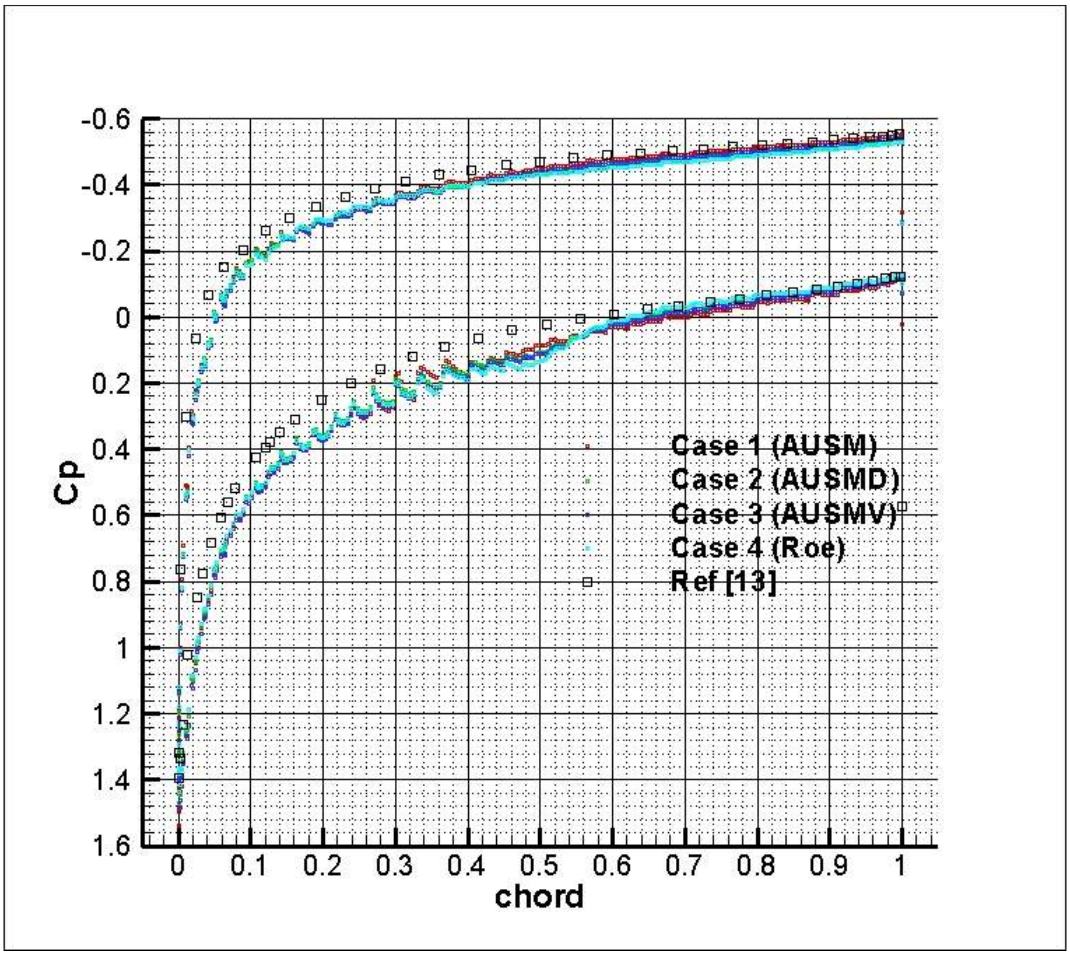


Figure 5.5 Pressure coefficient distribution of the first four cases for supersonic inviscid flow around NACA 0012 airfoil

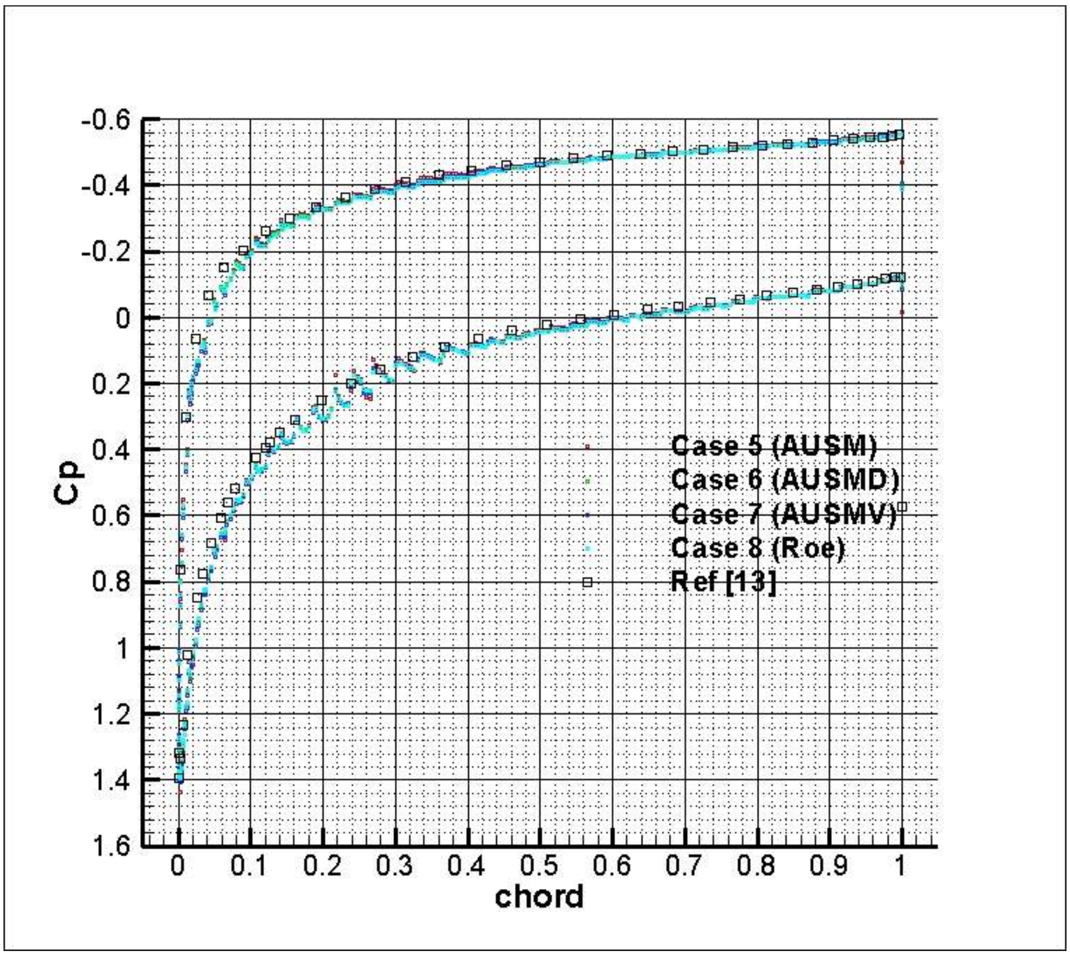


Figure 5.6 Pressure coefficient distribution of the last four cases for supersonic inviscid flow around NACA 0012 airfoil

It can be seen in Figure 5.5 that AUSM method approach to the reference data at the upper surface in the second half of the chord. However, it gives the farthest result at the lower surface. On the other hand, Roe's method captures the lower surface pressure distribution accurately for the second half of the chord, while it gives slightly farther result for the upper surface compared to the other methods. In Figure 5.6, almost all cases capture the accurate results. The slight differences between these methods are difficult to observe. In Figures 5.7 and 5.8, the Mach and pressure contours for the cases, where solution adaptation is used, are presented.

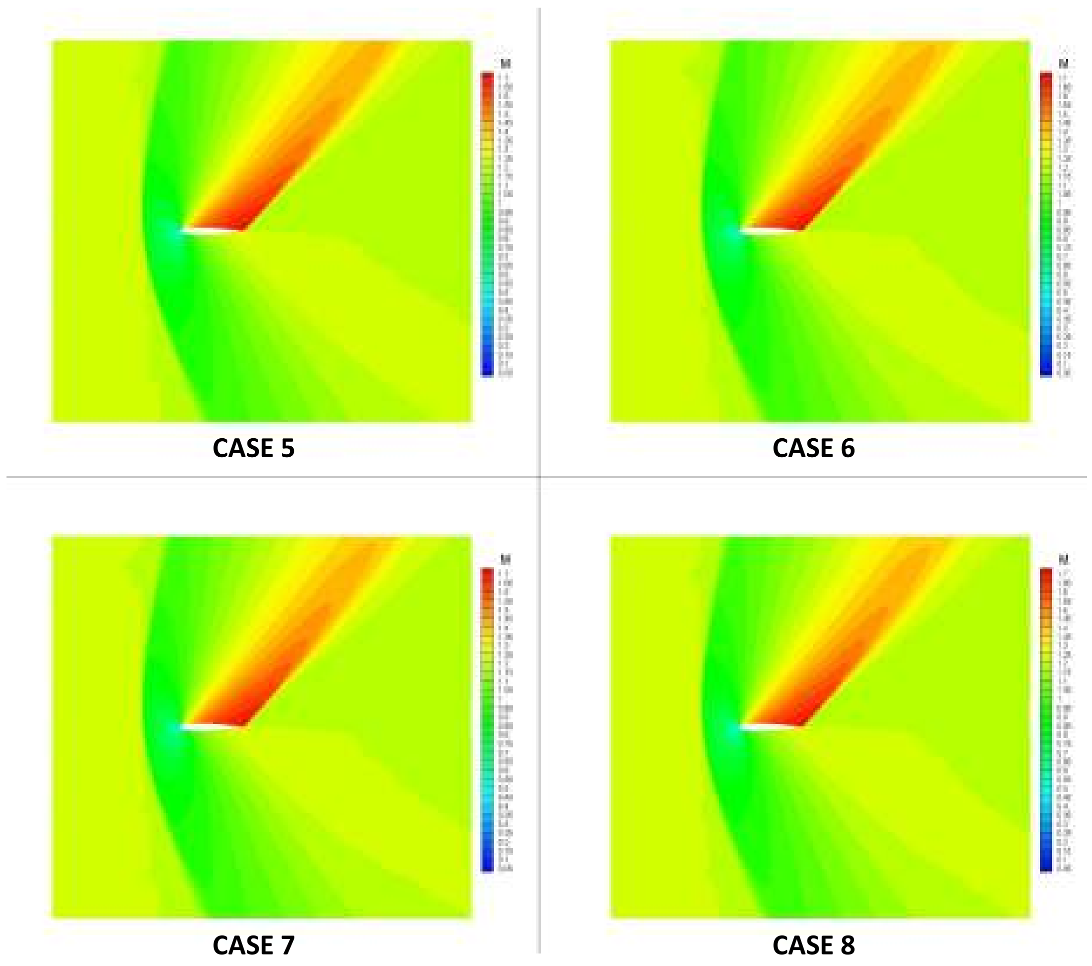


Figure 5.7 Mach contours of cases where solution adaptation is applied around NACA 0012 for supersonic inviscid flow

In Figure 5.7, slight differences can be observed among the flux methods. The bow shock before the leading edge is captured in all cases. The oblique shock at the upper surface of the airfoil is also captured by all of the cases. There are some slight differences in the strength and length of the oblique shock. The length is a little longer for the AUSM method and it shortens in AUSMD. Moreover, Roe gives the shortest length among the four cases while shorter length compared to AUSMD is obtained by AUSMV. In Figure 5.8, one can also see the pressure differences at the oblique shock for these four methods.

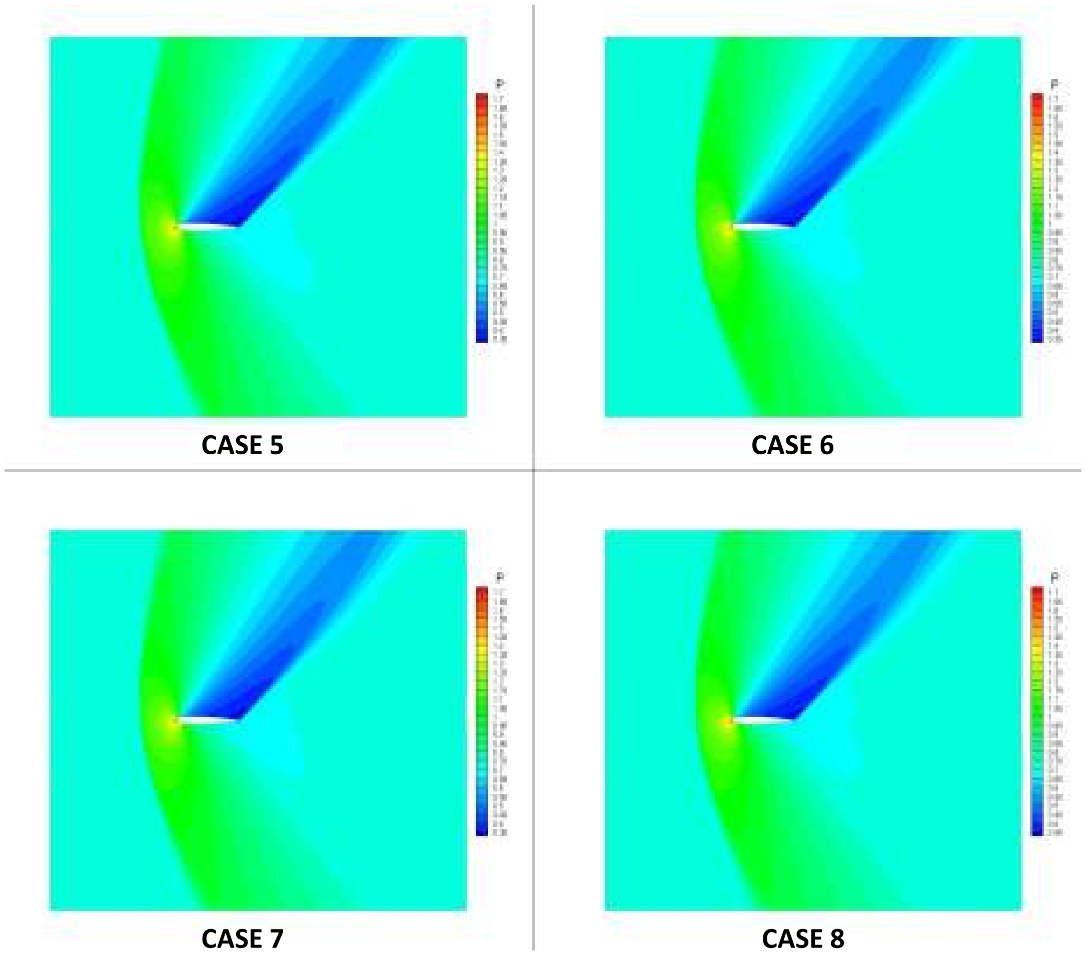


Figure 5.8 Pressure contours of cases where solution adaptation is applied around NACA 0012 for supersonic inviscid flow

5.2 LOW REYNOLDS NUMBER FLOW

In this section, two different problems are examined. While one of them is subsonic, a transonic flow is solved at a relatively low Reynolds number. It is expected to get accurate results for low Reynolds number flows since the flow regime is laminar for those cases. Whereas it is not necessary to use quad grids since Cartesian grids provide sufficiently small sizes in the boundary layer, quad grid is used as an illustration of hybrid mesh effect for the first test problem. The test problems are tabulated below.

Table 5.6 Test problems for low Reynolds number flow

SECTION	AIRFOIL	M_∞	α (degrees)	Re_∞
5.2.1	NACA 0012	0.5	0	5000
5.2.2	NACA 0012	0.8	10	500

5.2.1 Subsonic Flow around NACA 0012

The first problem is the laminar flow around a NACA 0012 airfoil at a Mach number of 0.5, an angle of attack of 0 degrees and a Reynolds number of 5000. The purpose of this test is to show the importance of solution refinement around a symmetrical airfoil. In addition, since the angle of attack is zero, the pressure and friction distribution must be symmetric at the lower and upper surfaces of the airfoil.

Table 5.7 Common properties of the cases without quad cells for subsonic laminar flow

MESH INPUTS	
Outer boundary size factor	18
Quad cell usage	No

SOLUTION INPUTS	
Order of the scheme	1
Flux method	AUSMV
Multistage number	3
CFL safety factor	1
Refinement cycle interval	10
log(RMS) for convergence	-9

MULTIGRID INPUTS	
Multigrid type	Saw-Tooth
Multigrid level	5
Fine grid iterations	15
Intermediate step iterations	10
Final grid iterations	10

Totally seven different cases where quad cells are not used are discussed at first . For all of them, the outer boundary is located 18 chords ahead of the airfoil. For inviscid flux calculations, AUSMV flux method is used. The solver is iterated until the logarithm of root mean square of normalized density residual reaches -9. 5 level multigrid is applied to accelerate the convergence time. In Table 5.7, the common properties for the cases without quad cells regarding the laminar subsonic flow can be shown.

For the test cases, only changing parameter is the number of solution refinement cycle, which is changed from 0 to 6. While applying the solution refinement, the interval between two cycles is set to 10. Drag coefficients of test cases and a numerical reference, ARC2D developed by NASA, are given in Table 5.8. ARC2D is a structured mesh solver which uses a cell-centered method. In addition, time elapsed for solution and cell numbers can be seen in Table 5.8.

Table 5.8 Comparison of results without quad cells for subsonic laminar flow around NACA 0012 airfoil

Case No	Description	C_D	Cell Number	Time
1	No solution refinement	0.0651	4040	0 hour 38 minutes 49 seconds
2	One solution refinement	0.0483	9442	0 hour 43 minutes 56 seconds
3	Two solution refinement	0.0397	21208	1 hour 26 minutes 9 seconds
4	Three solution refinement	0.0356	46488	2 hours 44 minutes 23 seconds
5	Four solution refinement	0.0328	92486	4 hours 37 minutes 41 seconds
6	Five solution refinement	0.0316	172874	8 hours 23 minutes 33 seconds
7	Six solution refinement	0.0311	335606	25 hours 25 minutes 16 seconds
-	Reference (ARC2D) [40]	0.0321	40960	-

It can be shown that while the number of solution refinement cycle is increasing, drag coefficients are approaching to the numerical reference data. However, the convergence time increases greatly. Sixth case gives the closest result for the drag coefficient, which underestimates drag coefficient only by 1.6 %. Whereas one more solution refinement cycle leads to a distant result with respect to the reference data, it gives a slightly close pressure distribution relative to the reference data, as shown in Figure 5.9.

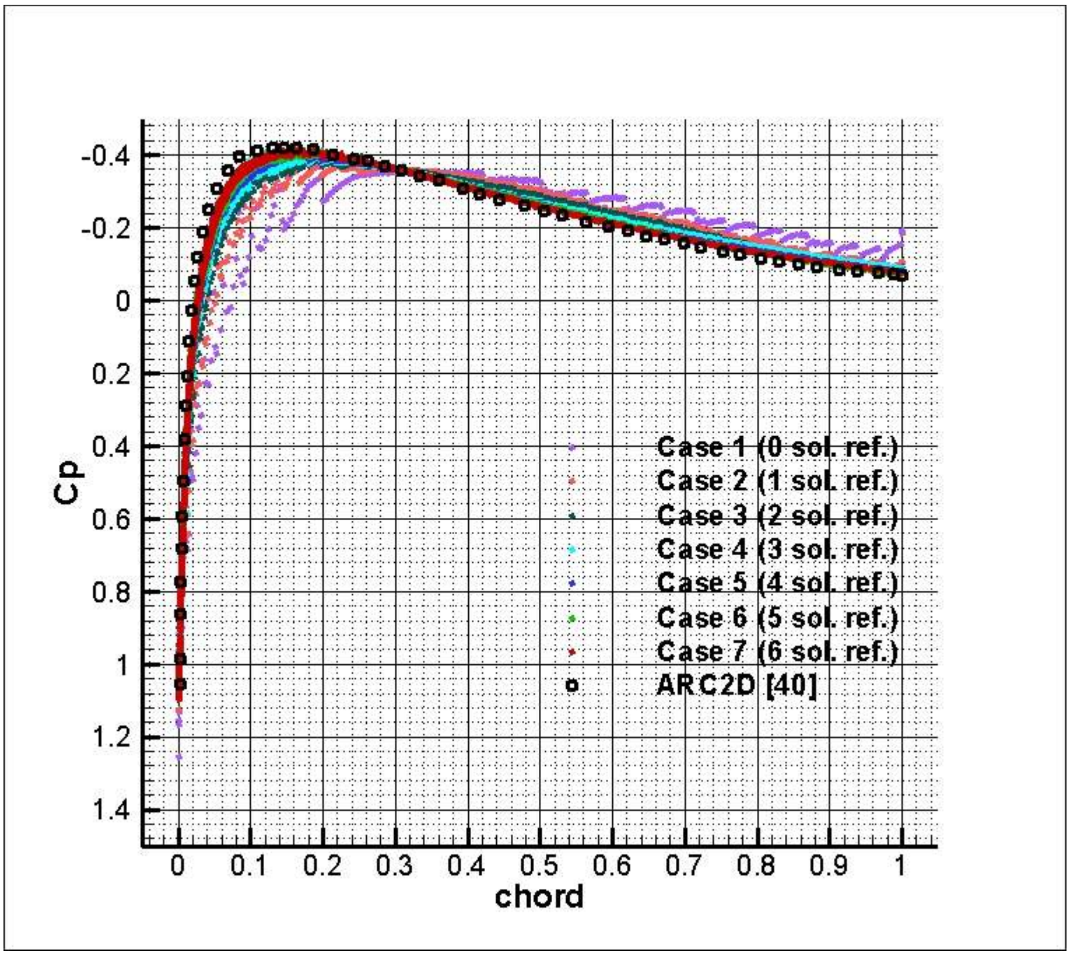


Figure 5.9 Pressure coefficient distribution for subsonic laminar flow around NACA 0012 airfoil

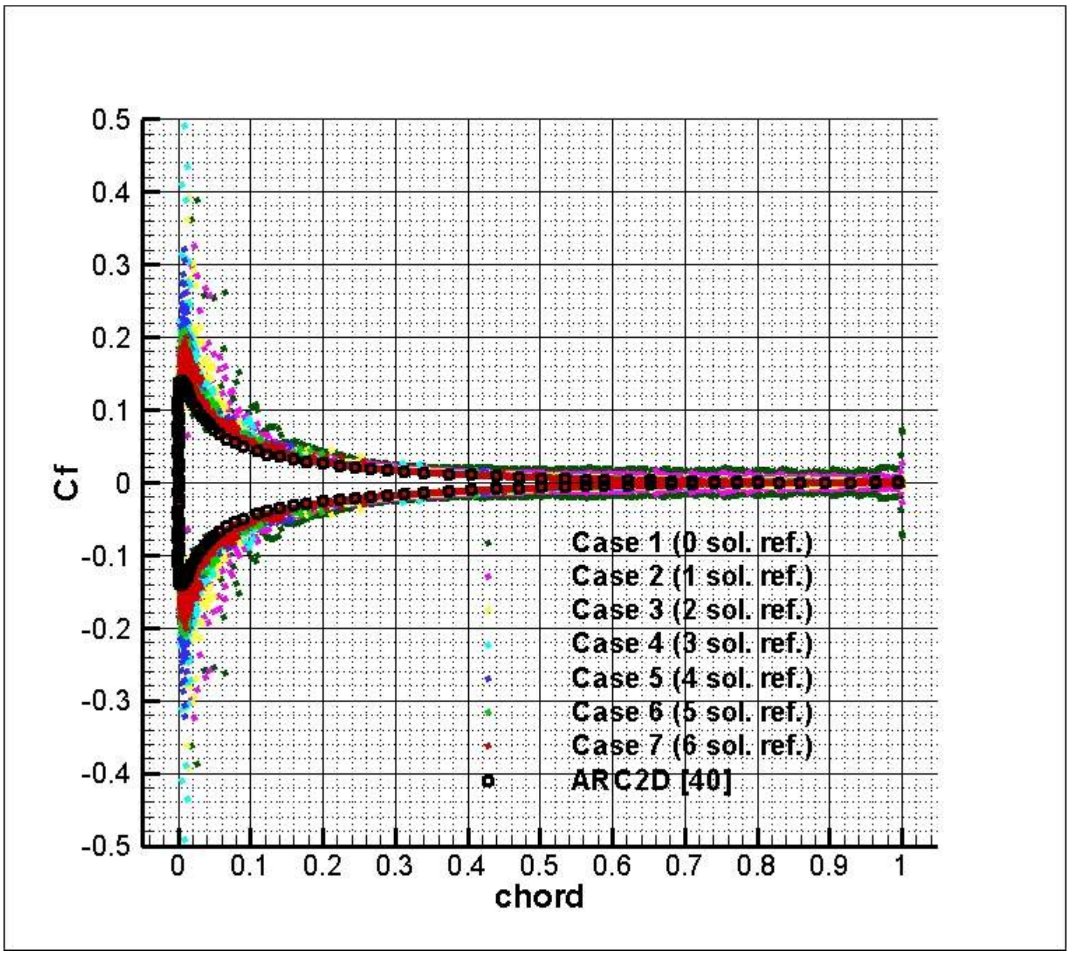


Figure 5.10 Skin friction coefficient distribution for subsonic laminar flow around NACA 0012 airfoil

It can be observed from Figures 5.9 and 5.10 that increasing cycle number results more accurate pressure and skin friction distribution. While the peak of pressure coefficient cannot be captured accurately, Case 7 gives the most closest result. For the initial cases, especially the skin friction coefficients are scattering. With the increasing cycle number, the scattering decreases in a considerable amount since the cell sizes become smaller with the increase in the solution refinement cycle number. As a result, most accurate results are obtained for Case 7. Since it is difficult to observe it from Figures 5.9 and 5.10 due to the excessive number of presented test cases, Case 7 and ARC2D data are compared separately in Figure 5.11.

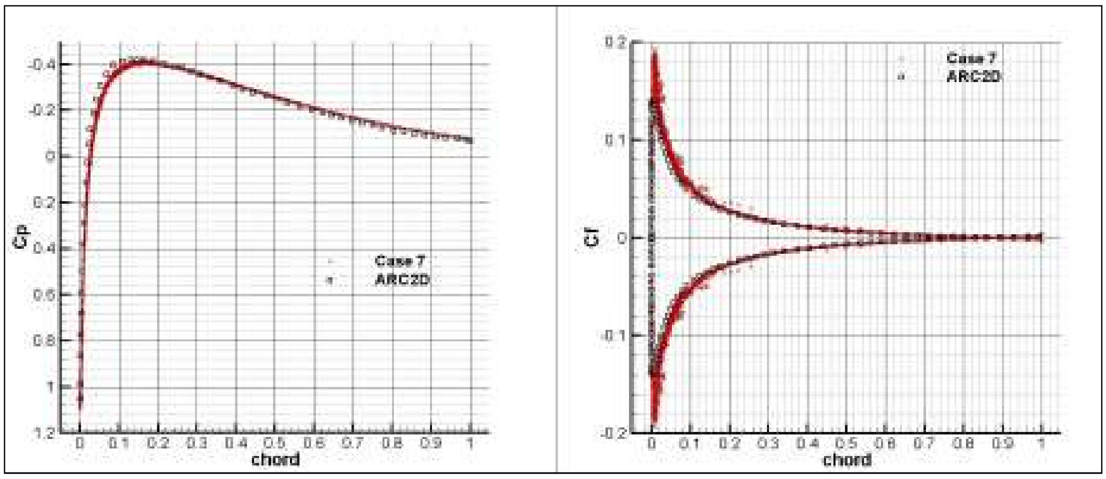


Figure 5.11 Comparison of Case 7 with the reference data for subsonic laminar flow around NACA 0012 airfoil

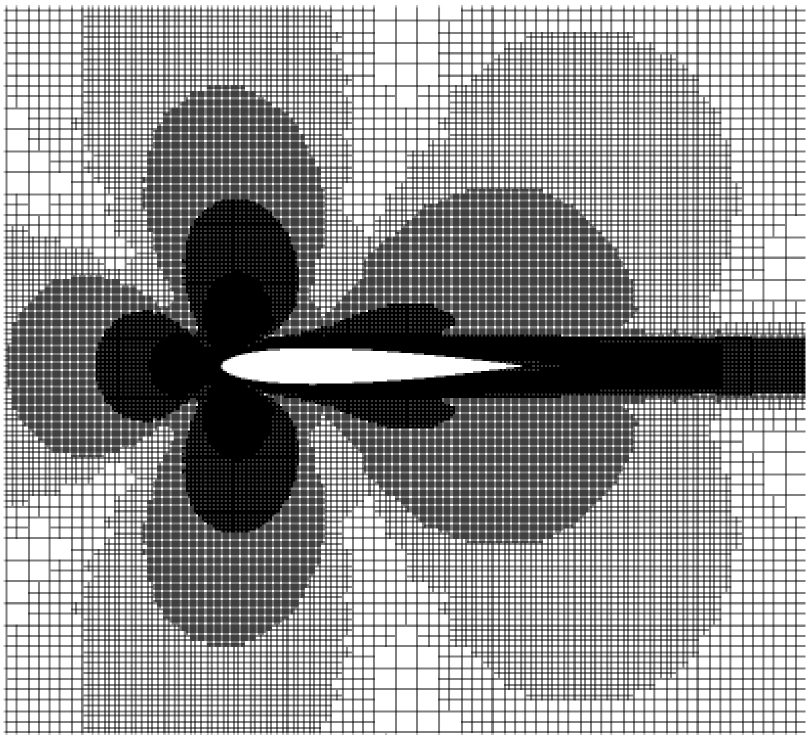


Figure 5.12 The grid of Case 7 around NACA 0012 for subsonic laminar flow

Case 7 gives a very good result for the pressure coefficient. However, the peak of the skin friction coefficient is overestimated by 42 %, while the rest of the friction is in a good agreement with the reference data. In Figure 5.10, one can see that with the increase in the refinement cycle, the overestimating percent decreases. It can be estimated that if the number of refinement cycles is greater than 6, better results can be obtained for the peak. Yet, if one looks at the percent increase in time between sixth and seventh cases, it is not difficult to conclude that time elapsed for convergence increases significantly.

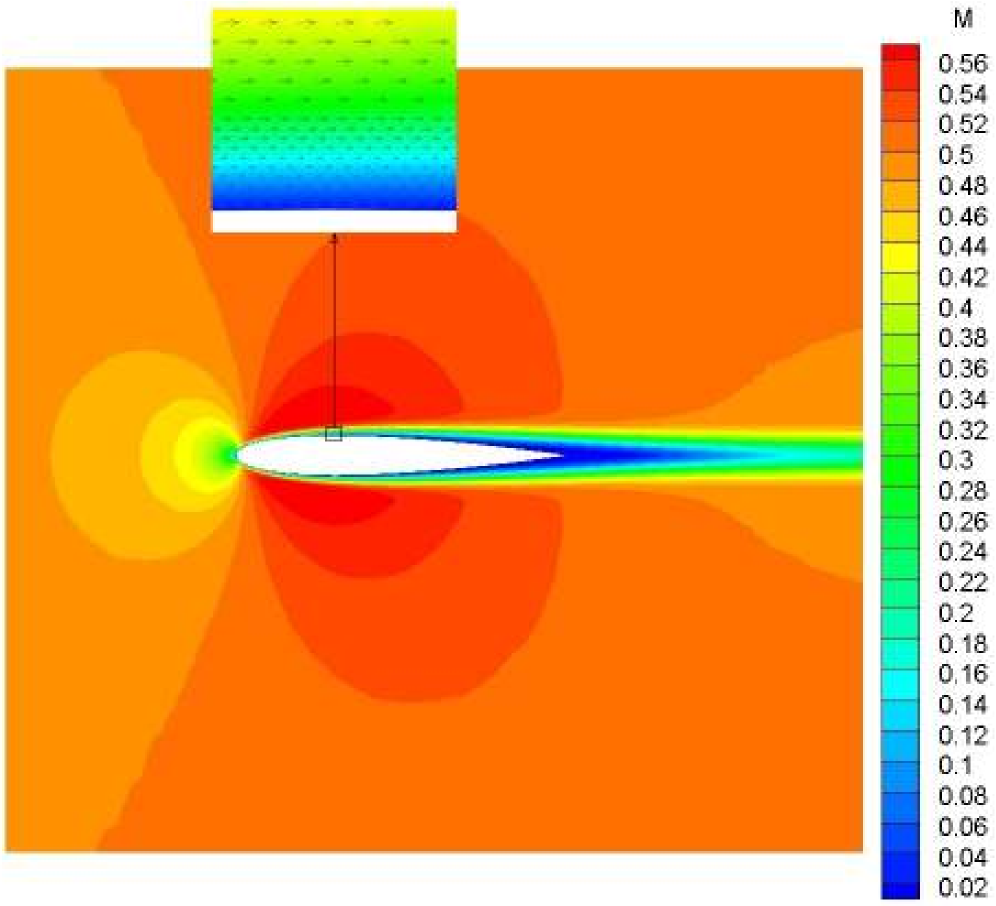


Figure 5.13 Mach contours of Case 7 around NACA 0012 for subsonic laminar flow

The grid used in Case 7 is shown in Figure 5.12. The finer meshes around leading edge, resulted from stagnation points can be seen easily. Moreover, the shear layers become finer with the solution adaptation. The grid around the wake formed after the trailing edge also become smaller by solution adaptation. In addition, Mach and pressure contours for this case are given in Figure 5.13 and 5.14, respectively. The velocity profile at the upper surface is depicted at approximately 30 % of the chord in Figure 5.13.

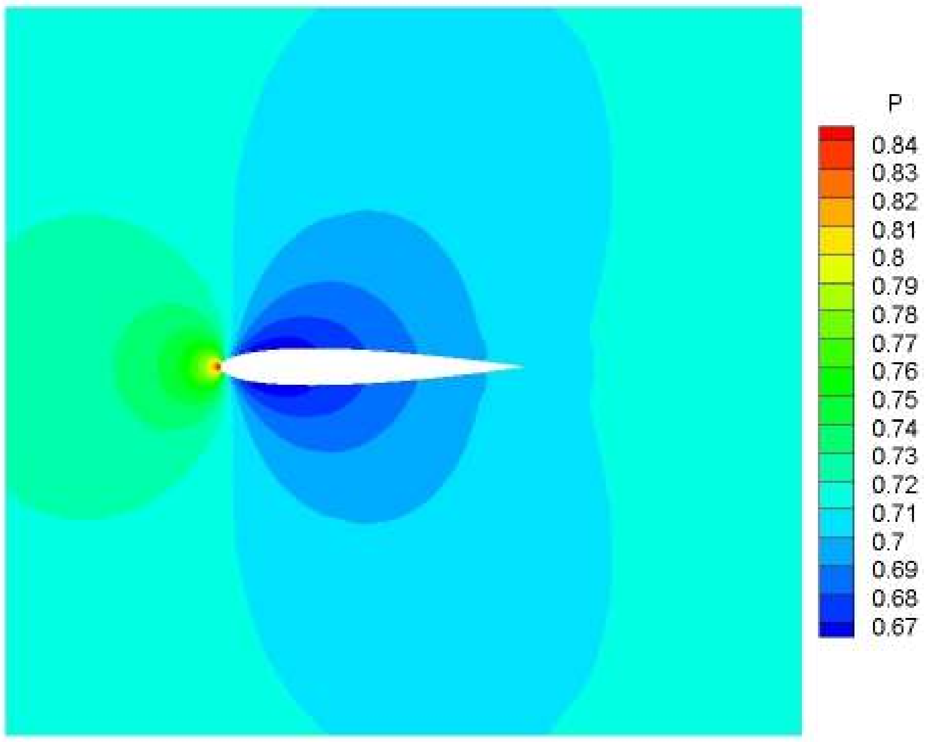


Figure 5.14 Pressure contours of Case 7 around NACA 0012 for subsonic laminar flow

In addition to the cases where only Cartesian grids are used, two more cases, eighth and ninth cases, are considered in order to illustrate the hybrid mesh effect on laminar flows. 16 rows of quad cells are used for both cases. While multigrid technique is not used for the ninth case due to some problems regarding solution adapted grids with multigrid, it is used for the eighth case. The results are tabulated in Table 5.9. As shown in Table 5.8 and 5.9, hybrid grid gives a more accurate result for the grids where solution adaptation is not applied.

Table 5.9 Comparison of results with quad cells for subsonic laminar flow around NACA 0012 airfoil

Case No	Description	C_D	Cell Number	Time
8	No solution refinement	0.0590	14562	15 hours 16 minutes 54 seconds
9	Three solution refinement	0.0371	107250	108 hours 46 minutes 36 seconds
-	Reference (ARC2D) [40]	0.0321	40960	-

In Figure 5.15, one can see the comparison done between grids without solution adaptation for pressure and skin friction coefficient distribution. One can say that the scattering which comes from different cell sizes in the first case diminishes significantly when hybrid mesh is used. Moreover, the peak of the skin friction coefficient is nearly captured in the eighth case unlike the first case where quad cells are not used. Moreover, the general distribution is closer to the reference result compared to the first case. However, the pressure distribution is still far from the reference distribution, as expected since the grid is not sufficiently finer at the critical regions.

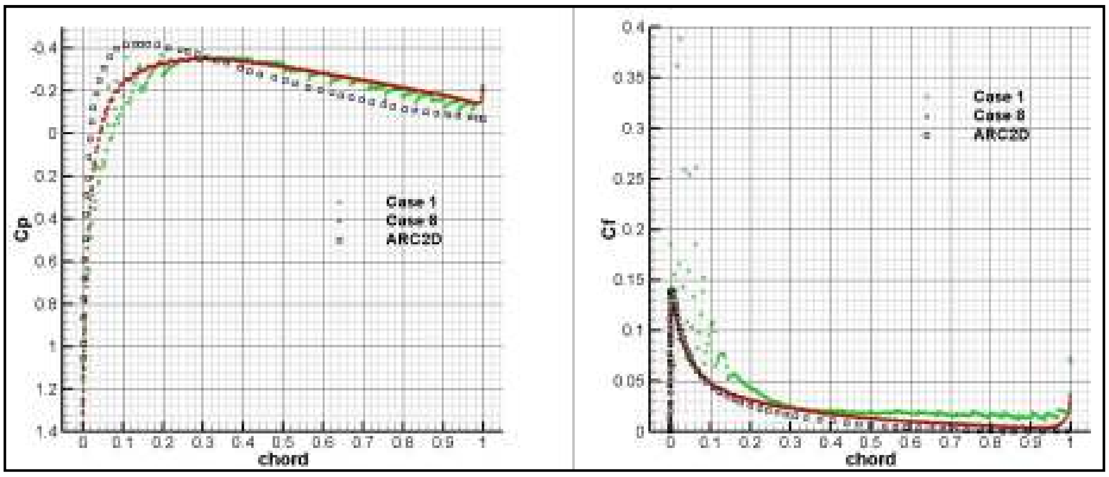


Figure 5.15 Comparison of Case 1, Case 8 and ARC2D for subsonic laminar flow around NACA 0012 airfoil

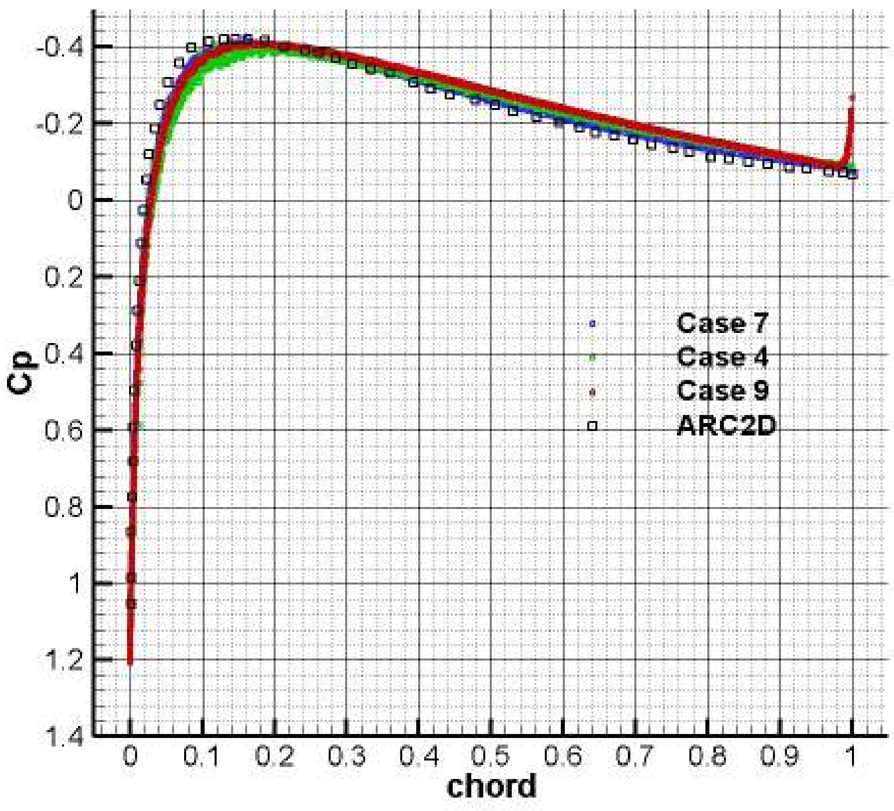


Figure 5.16 Pressure coefficient distribution which hybrid grid is compared with Cartesian grids and reference for subsonic laminar flow around NACA 0012 airfoil

In Figures 5.16 and 5.17, one can see the comparison of the pressure and skin friction distributions of the Case 9 which is a hybrid mesh with three cycles of solution adaptation, with Case 3 which is a Cartesian mesh with three cycles of solution adaptation, Case 7 which is a Cartesian mesh with six cycles of solution adaptation and the reference ARC2D. It can be observed that the best result is obtained by Case 7 for pressure distribution, whereas Case 9 gives the best result for skin friction distribution. The skin friction coefficient distribution along the entire surface is captured accurately including the peak at the trailing edge unlike the others. However, some deviations are observed at the trailing edge both for pressure and skin friction coefficients. As a result, it can be inferred that hybrid mesh gives more accurate and non-scattering results for skin friction coefficients. However, the convergence rate increases significantly. In terms of pressure coefficient distribution, a significant effect is not observed for hybrid mesh on laminar flows.

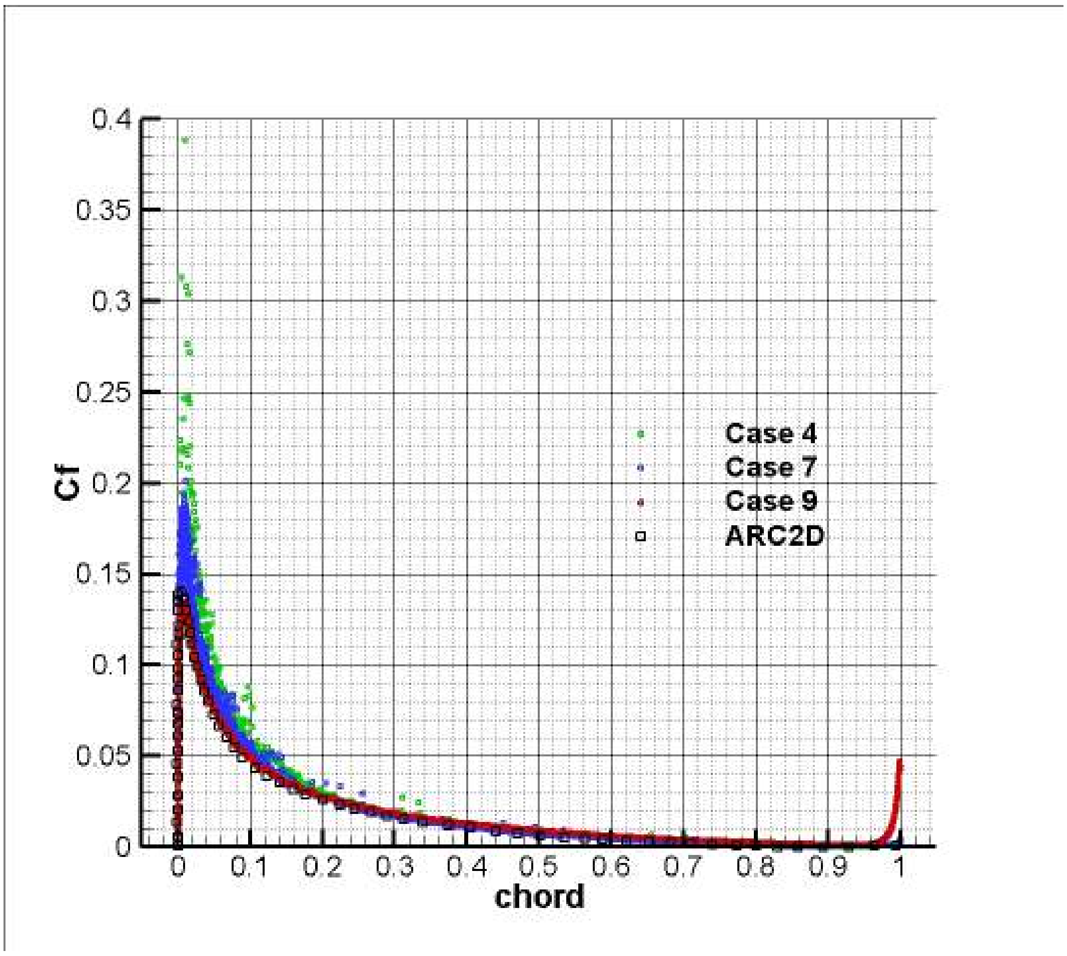


Figure 5.17 Skin friction coefficient distribution which hybrid grid is compared with Cartesian grids and reference for subsonic laminar flow around NACA 0012 airfoil

5.2.2 Transonic Flow Around NACA 0012

The second problem is the laminar flow around a NACA 0012 airfoil at a Mach number of 0.8, an angle of attack of 10 degrees and a Reynolds number of 500. The aim of this problem is to show that the code can be capable of solving very small

Reynolds number flows around an airfoil. Moreover, the importance of solution adaptation can be seen by comparing different solution adaptation cycles.

Similar to the previous problem, seven different test cases are used for the transonic flow. For all of them, AUSMV flux calculation technique is used for inviscid flux calculations. The far-field boundary is placed 18 chords ahead of the airfoil. The interval between two solution refinement cycles is set to 15. Finally, normalized density residual at the last residual is set to -9 as the convergence criteria. Since there are some problems while using the multigrid technique for very low Reynolds numbers, it is not applied for Cases 1 to 6. Therefore, only for the seventh case, the multigrid technique is applied using three cycles after problems are fixed.

Table 5.10 Common properties for transonic laminar flow

MESH INPUTS	
Outer boundary size factor	18
Quad cell usage	No

SOLUTION INPUTS	
Order of scheme	1
Flux method	AUSMV
Multistage number	3
CFL safety factor	1
Refinement cycle interval	15
log(RMS) for convergence	-9

In Table 5.11, one can see the coefficients of drag and lift as well as the convergence time for each case. Since there is no data found in literature for this problem, the comparisons can be done between test cases obtained by the code. However, the pressure coefficients and skin friction coefficients along the wall boundary can be compared with the reference [41], which is a numerical solver named NSC2KE. The mesh used in the reference [41] is a hybrid structured/unstructured mesh with 10924 triangular cells and 5590 meshpoints. These are presented in Figures 5.18 and 5.19.

Table 5.11 Comparison of results for transonic laminar flow around NACA 0012 airfoil

Case No	**Description**	C_D	C_L	**Cell Number**	**Time**
1	No solution refinement	0.2256	0.6752	4040	6 hours 38 minutes 26 seconds
2	One solution refinement	0.2084	0.6156	7378	10 hours 49 minutes 25 seconds
3	Two solution refinement	0.1942	0.5598	12902	17 hours 29 minutes 57 seconds
4	Three solution refinement	0.1841	0.5195	23411	28 hours 36 minutes 12 seconds
5	Four solution refinement	0.1775	0.4931	45024	37 hours 18 minutes 49 seconds
6	Five solution refinement	0.1718	0.4705	87372	77 hours 5 minutes 29 seconds
7	Six solution refinement	0.1686	0.4589	176059	34 hours 33 minutes 37 seconds

It can be seen that the drag and lift coefficients decrease while the solution refinement cycles increase. Furthermore, the solution time increases as the number of cycles increases. For Case 7, since three level multigrid is used as mentioned before, the solution time decreases considerably in comparison to Case 6. Since no numerical or experimental reference data available, the accuracy of these cases cannot be understood by examining the coefficients. However, Figures 5.18 and 5.19 can be used for this comparison.

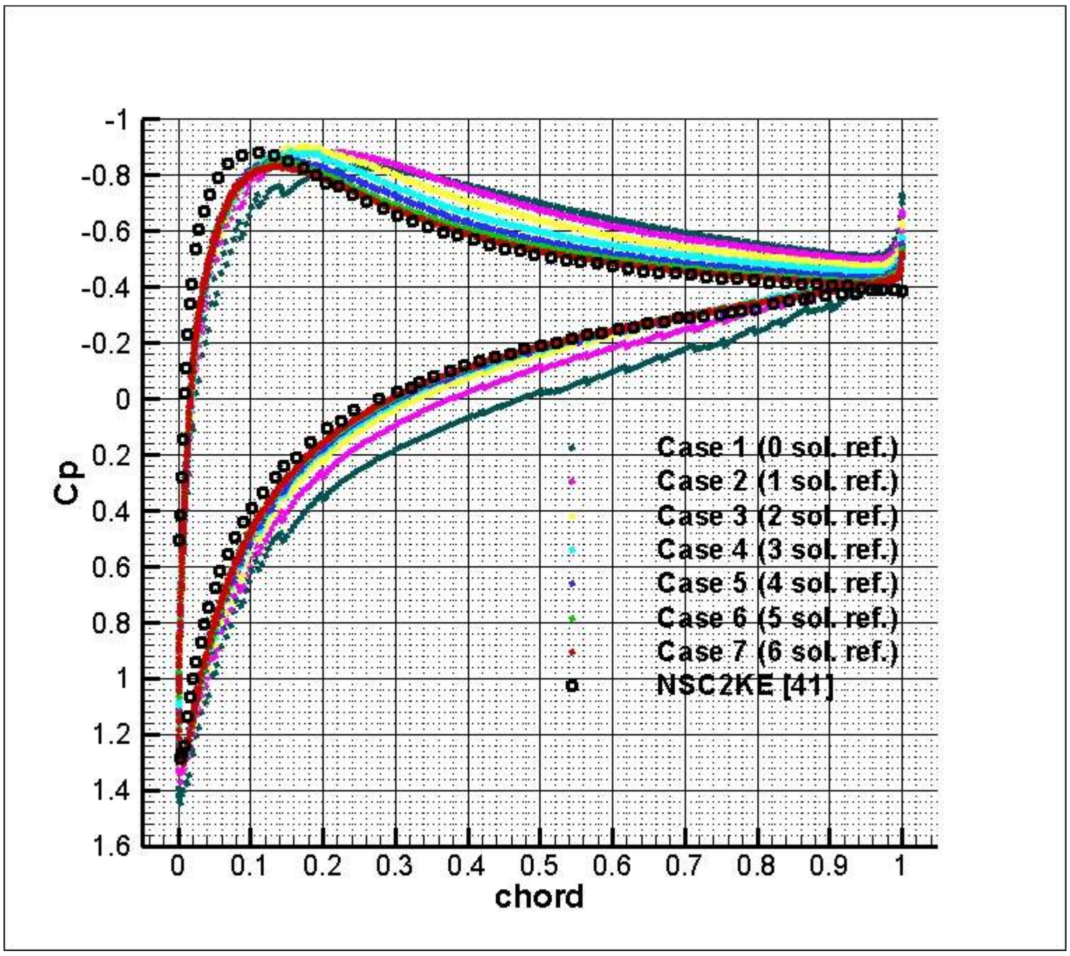


Figure 5.18 Pressure coefficient distribution for transonic laminar flow around NACA 0012 airfoil

As it can be seen in Figures 5.18 and 5.19, the increase in the number of cycles leads to closer results relative to the reference data. For the initial test cases, the results deviated significantly, as the number of cycles is increased by 1. However, the difference becomes smaller for the last cases. For example, if one examines the sixth and seventh cases, there is a slight difference in pressure and skin friction coefficients. Moreover, these two cases give the best results. However, at the regions around the leading edge, the pressure coefficients are underestimated so that the peak cannot be captured accurately. However, the pressure distribution around the other sections of the geometry is in good agreement with the reference data. Moreover, it can be said that the skin friction distribution are captured accurately despite slight differences at the lowest and highest points of the reference data.

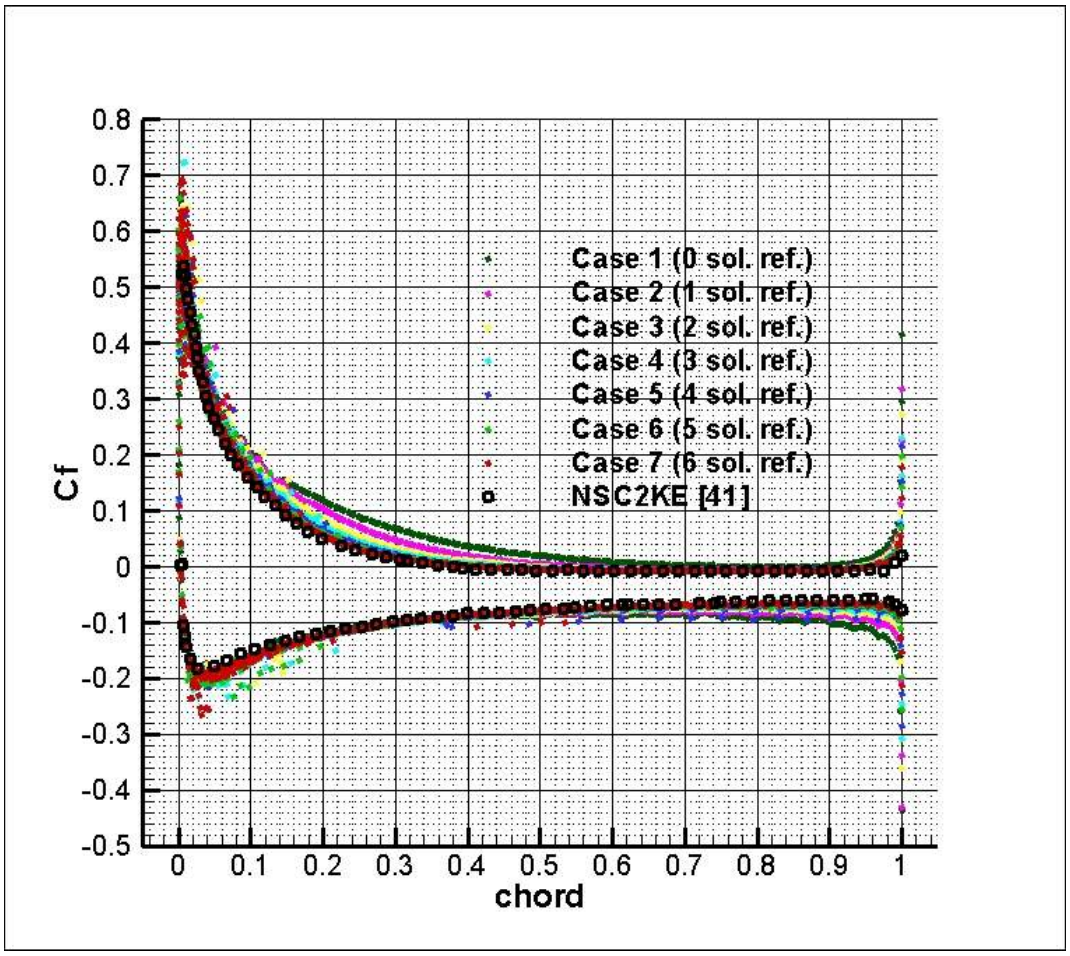


Figure 5.19 Skin friction coefficient distribution for transonic laminar flow around NACA 0012 airfoil

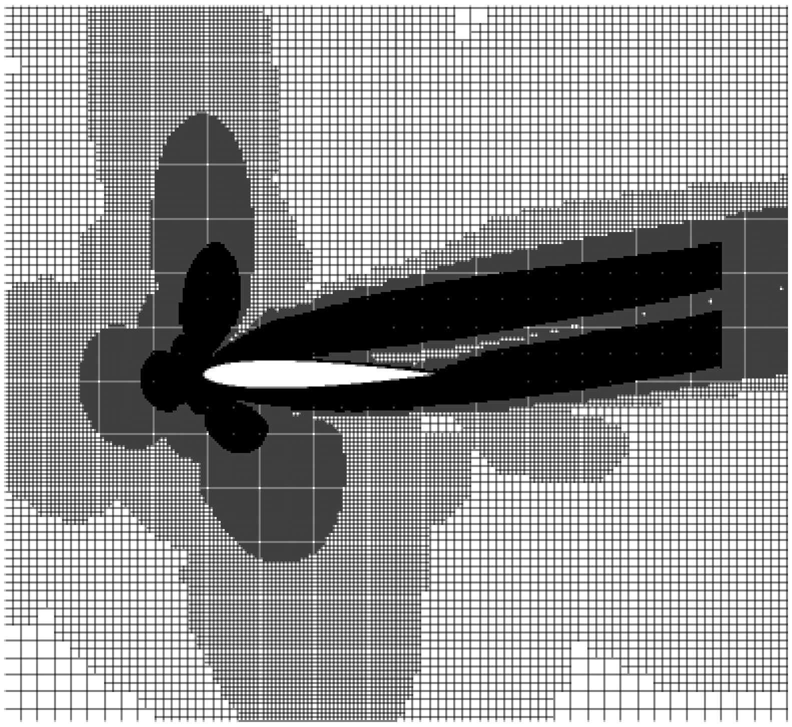


Figure 5.20 The grid of Case 7 around NACA 0012 for transonic laminar flow

The grid used in the seventh case is shown in Figure 5.20. The regions around the leading edge become finer since there are large gradients arising from stagnation points. Moreover, since the flow is coming with an angle of attack, the wake is formed at the upper surface of the airfoil instead of the trailing edge, with an angle different than zero. Since some layers are created around the wake, these grids become smaller with the solution adaptation, as shown above.

In Figures 5.21 and 5.22, Mach contours are presented for the reference data and Case 7, respectively. The similarity between these figures can be seen easily. In addition, the pressure contours and temperature contours for Case 7 are presented in Figures 5.23 and 5.24, respectively.

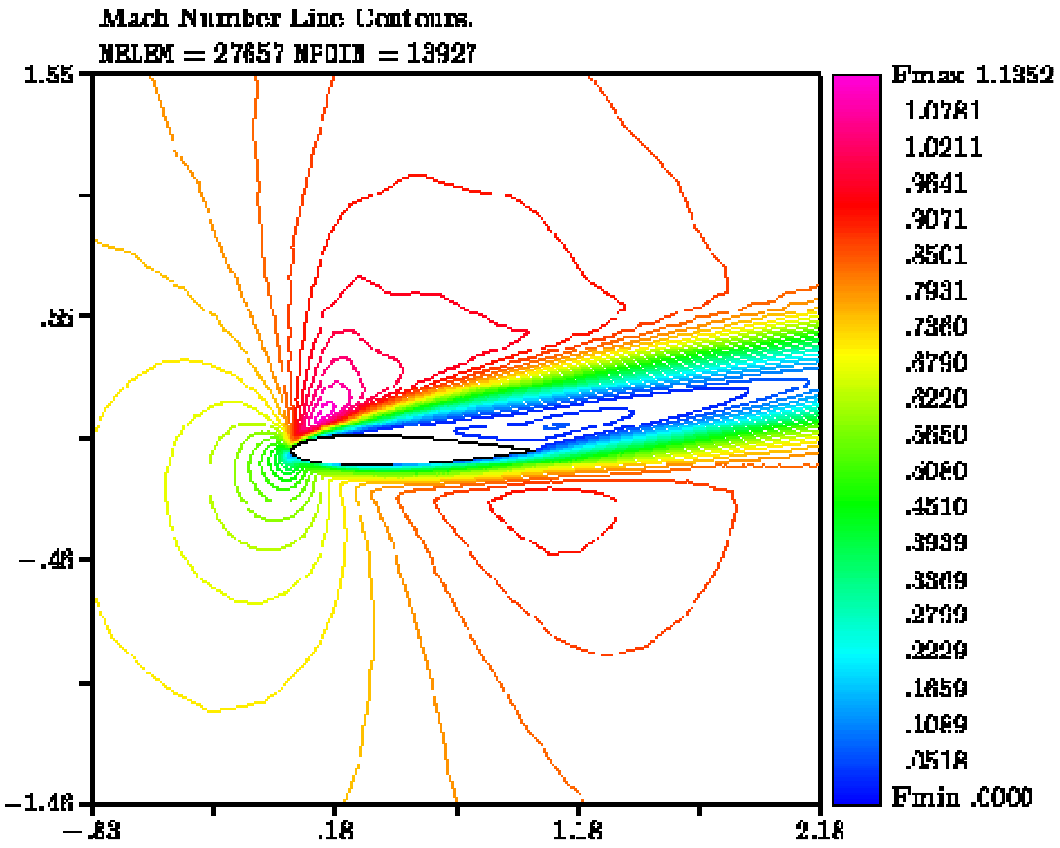


Figure 5.21 Mach contours of reference [41] around NACA 0012 for transonic laminar flow

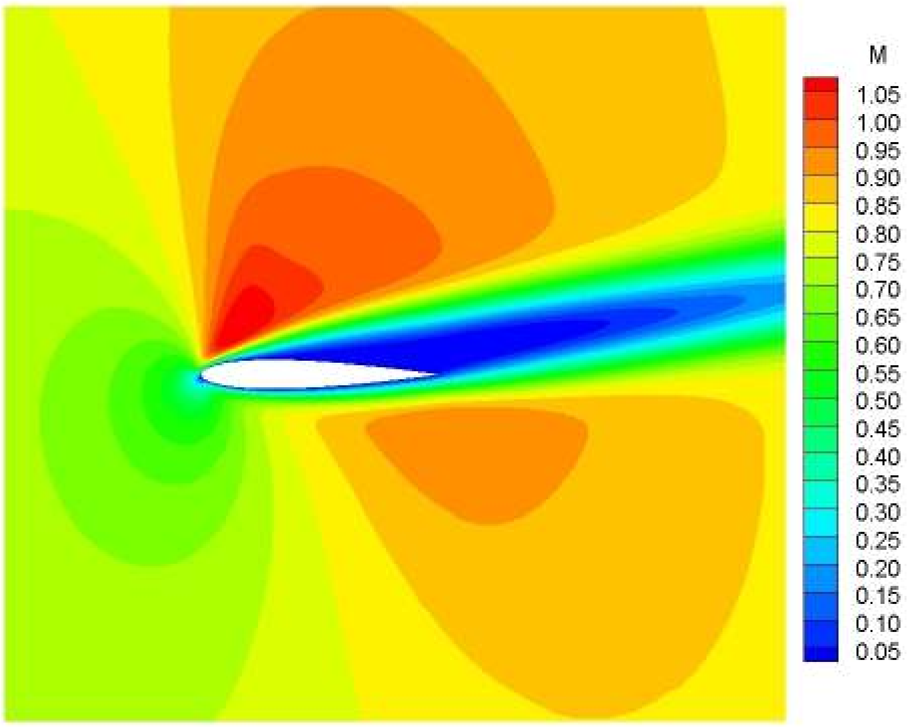


Figure 5.22 Mach contours for Case 7 around NACA 0012 for transonic laminar flow

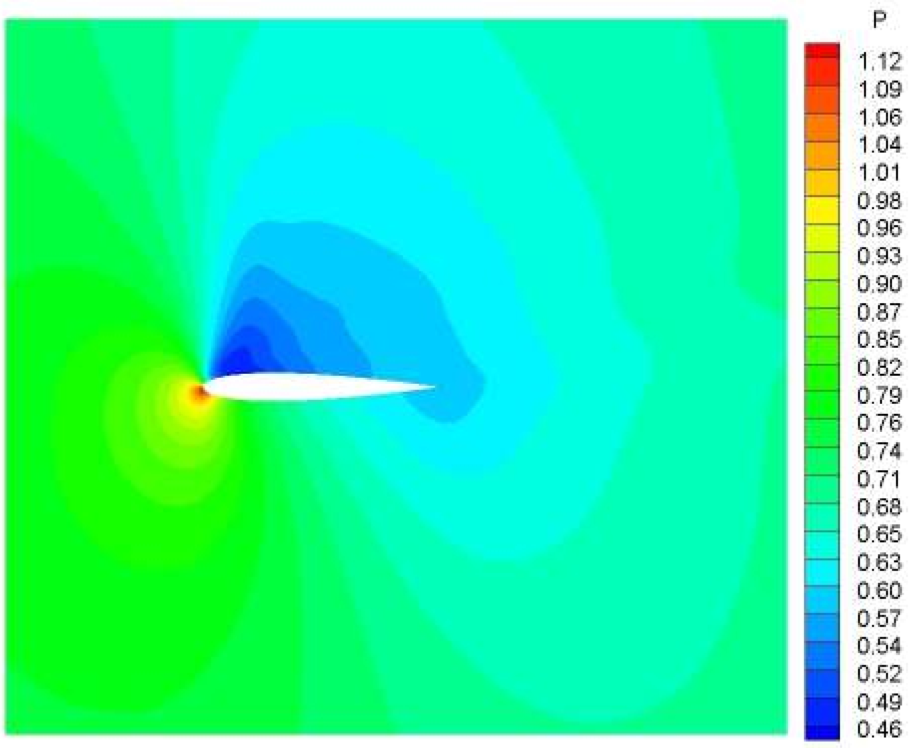


Figure 5.23 Pressure contours for Case 7 around NACA 0012 for transonic laminar flow

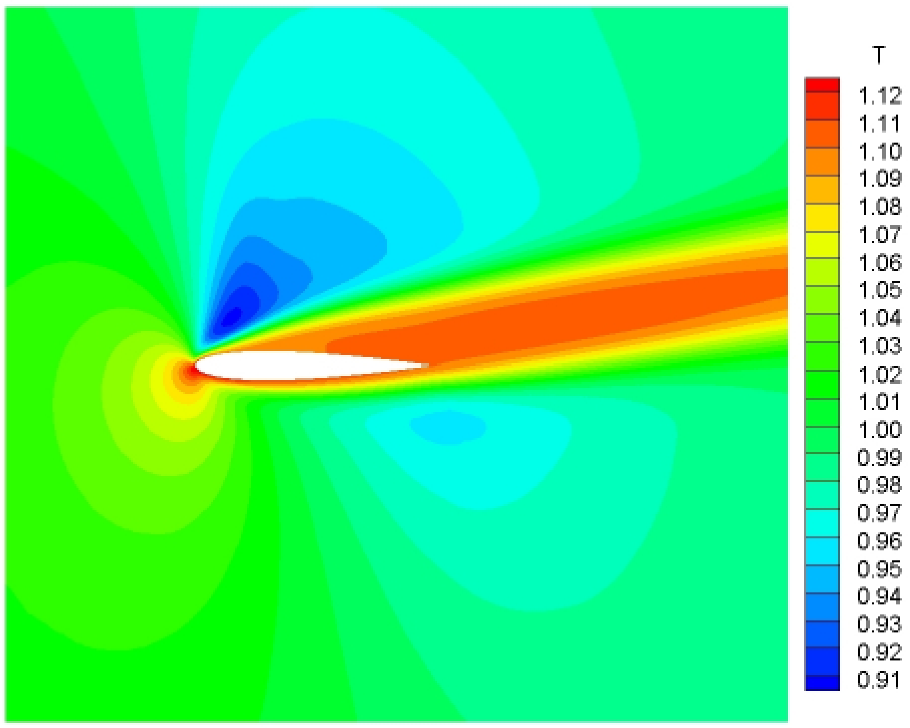


Figure 5.24 Temperature contours for Case 7 around NACA 0012 for transonic laminar flow

5.3 HIGH REYNOLDS NUMBER FLOW

In this section, one problem is tested. In this problem, a multi-element airfoil is used in a subsonic flow. Since the Reynolds number is high, the flow regime is turbulent in this problem. Therefore, it is not expected to get close results compared to the experimental reference data found in reference [42]. The purpose for considering this high Reynolds number flow is to examine the effect of different hybrid meshes which composes of body-fitted and sufficiently smaller boundary layer grids and Cartesian grids outside the boundary layer.

5.3.1 Subsonic Flow Around 30P30N

In this problem, the subsonic flow around a three-element airfoil, i.e. 30P30N, is analyzed at a Mach number of 0.2, an angle of attack of 8 degrees and a Reynolds number of 9 million. While almost all parameters are kept to be the same in all test cases, the only changing parameter is the row number of the Quad cells. Row numbers are changed from 0 to 32 for five test cases and the results obtained from these cases are compared with the experimental results found in reference [42].

Table 5.12 Common properties for subsonic high Reynolds number flow

MESH INPUTS	
Outer boundary size factor	18
Stretch factor	1.1

SOLUTION INPUTS	
Order of scheme	1
Flux method	AUSMV
Multistage number	3
CFL	1
Refinement cycle number	3
Refinement cycle interval	15
log(RMS) for convergence	-8

For all cases, the inviscid flux calculation method is set to AUSMV. Three cycles of solution adaptation is used. The convergence is achieved when the normalized residual reaches -8. The relation between quad cells is determined by using a fixed stretch factor of 1.1 For this problem, the multigrid technique is not applied. The parameters that are kept to be the same for all cases are tabulated in Table 5.12. In this table, the test cases and the results obtained from these are presented.

Table 5.13 Comparison of results for subsonic high Reynolds number flow around the 30P30N airfoil

Case No	Description	C_D	C_L	Cell Number	Time
1	No Quad cells	0.2121	1.1911	38254	6 hours 57 minutes 24 seconds
2	4 rows of Quad cells	0.2010	1.2182	62675	33 hours 43 minutes 47 seconds
3	8 rows of Quad cells	0.2115	1.0278	82702	49 hours 3 minutes 54 seconds
4	16 rows of Quad cells	0.2293	1.0181	111051	97 hours 14 minutes 8 seconds
5	32 rows of Quad cells	0.2191	1.0351	172114	104 hours 47 minutes 12 seconds

As it can be seen, the calculation time increases significantly when quad cells are used. Even though the cycle number of solution adaptation is the same for all cases, the cell number increases considerably as the number of rows increase It is difficult to comment on the drag and lift coefficients due to the fact that the reference results do not exist and the flow regime in these test cases is taken as laminar.

In the Figure 5.25, the pressure coefficient distributions of test cases are presented in comparison to the experimental results found in the literature. For the first two cases, the results at the upper surface are closer to the experimental results even though there is a huge difference. With the increase in the row number, the results are getting further away from the one in the reference. Since the real flow is turbulent, the actual distribution is considerably far away from all of the test cases. It can be observed that the distribution cannot be captured totally around the slat.

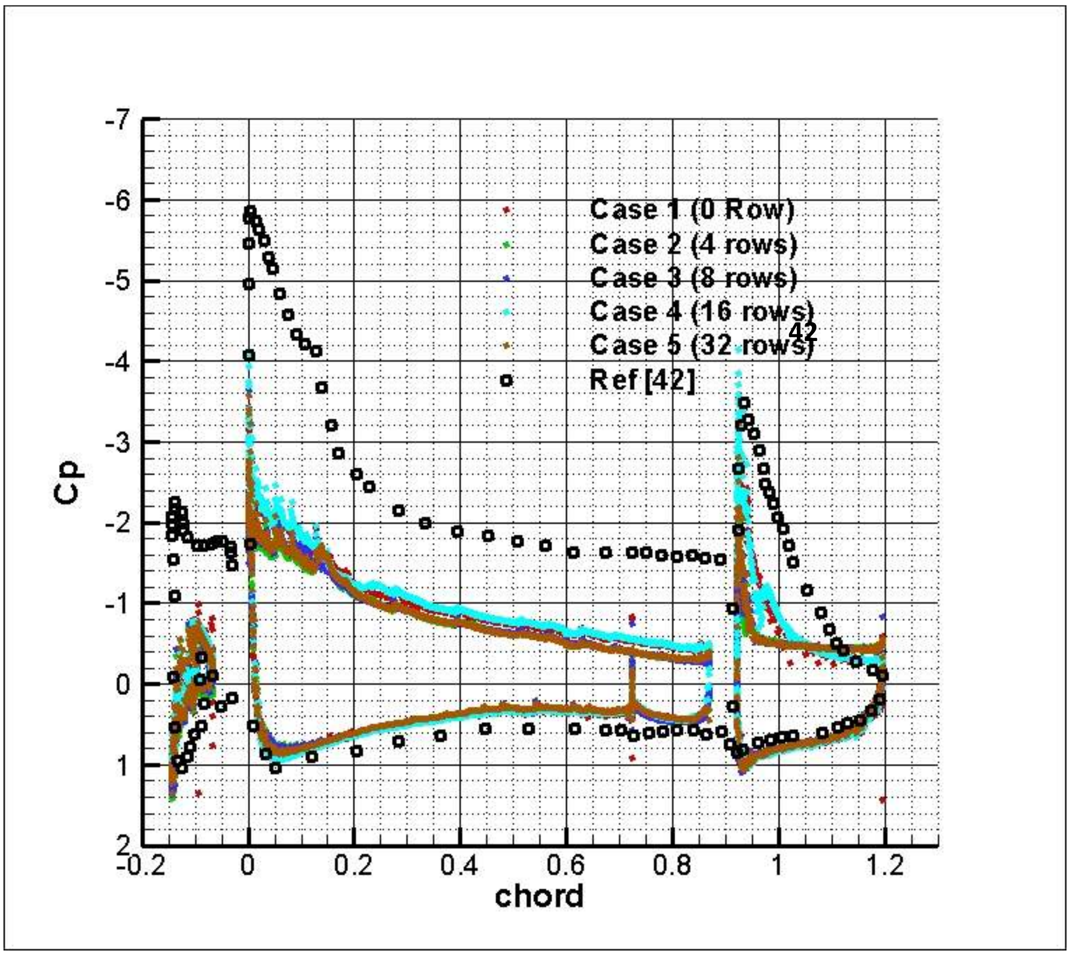


Figure 5.25 Pressure coefficient distribution for subsonic high Reynolds number flow around the 30P30N airfoil

The mesh used for Case 5 is depicted in Figure 5.26. One can also see the grids around the slat and the flap closely. In Figures 5.27 and 5.29, Mach and pressure contours from the fifth case are presented, respectively. The velocity profile on the upper surface of the main element is also shown at approximately 10% of the chord in Figure 5.27. Some streamlines are drawn in Figure 5.28. The vortex near the trailing edge of the main element of the airfoil can be observed.

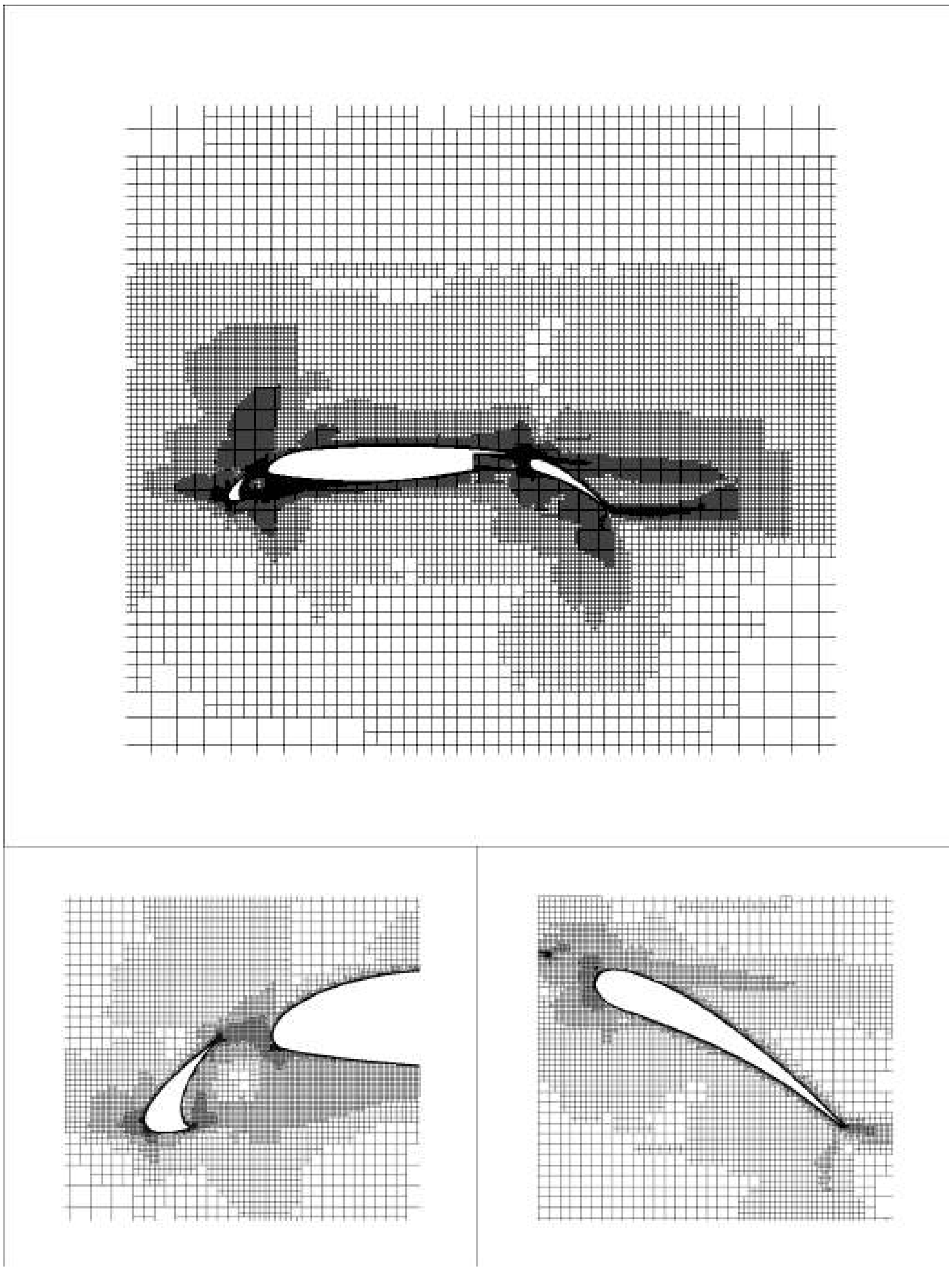


Figure 5.26 The mesh of the whole airfoil, the slat and the flap for Case 5 around the 30P30N airfoil for the subsonic high Reynolds number flow

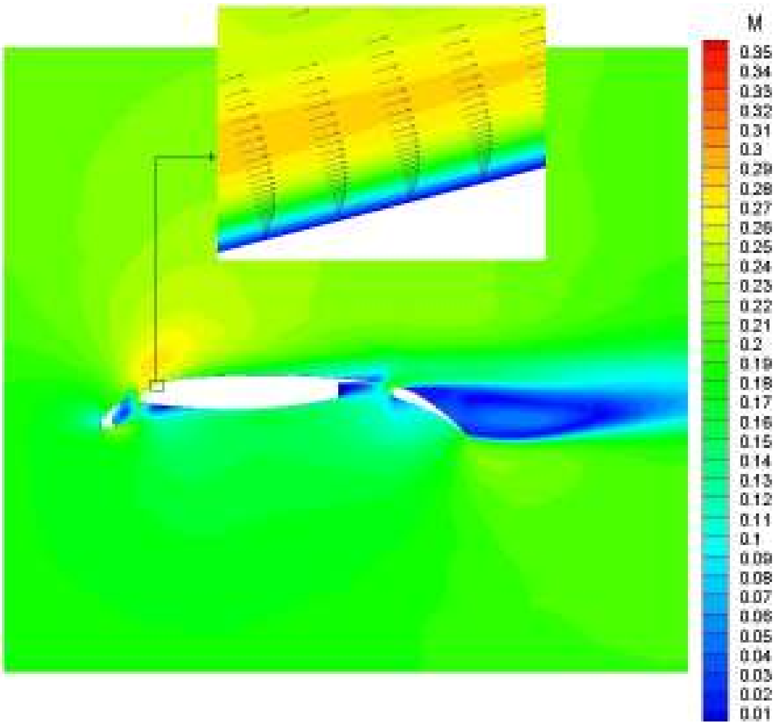


Figure 5.27 Mach contours for Case 5 around the 30P30N airfoil for subsonic high Reynolds number flow

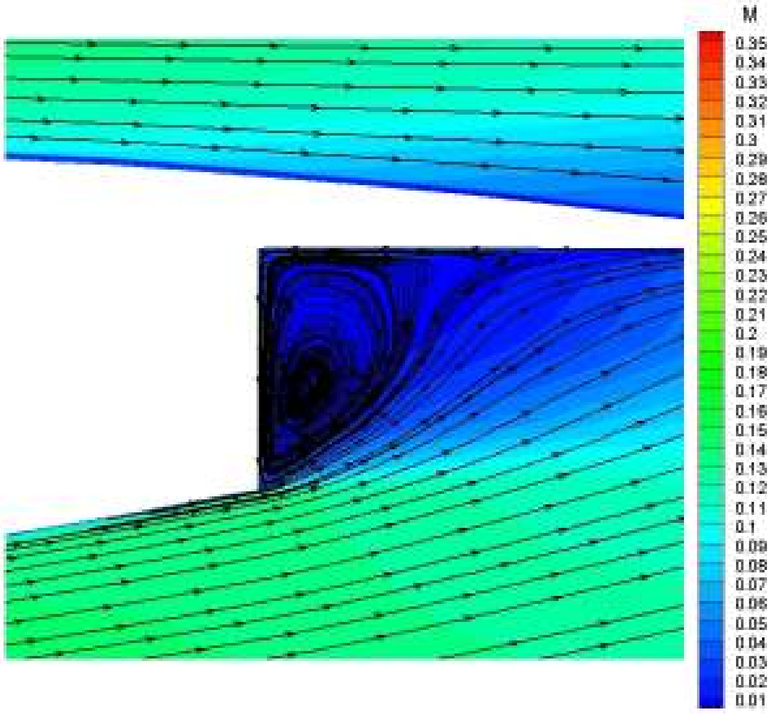


Figure 5.28 Streamlines for Case 5 around the trailing edge of the main element of the 30P30N airfoil for the subsonic high Reynolds number flow

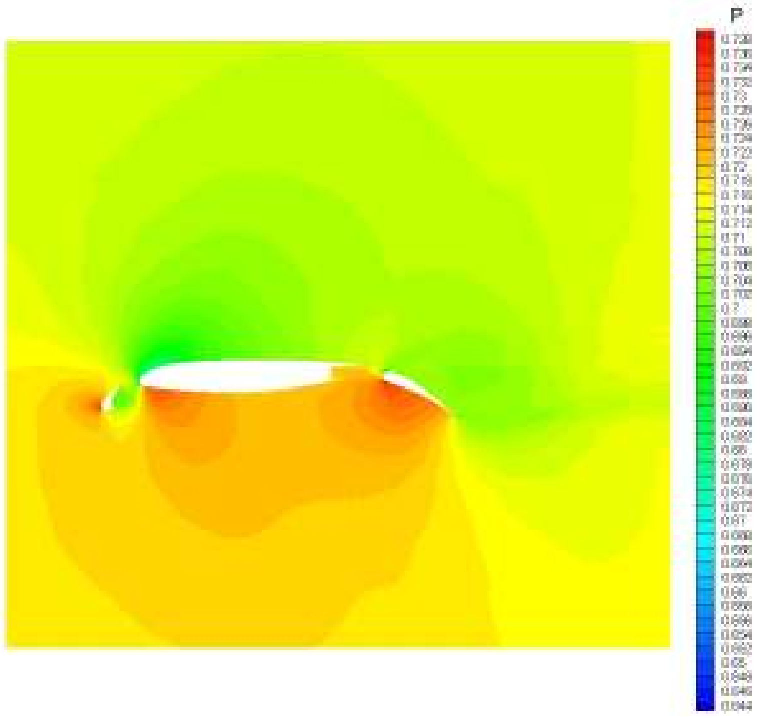


Figure 5.29 Pressure contours for Case 5 around the 30P30N airfoil for the subsonic high Reynolds number flow

CHAPTER 6

CONCLUSION

In this thesis work, the aim is to develop a two-dimensional laminar Navier-Stokes solver which uses finite volume method on Cartesian grids. As viscous flow, only the laminar flow regime is considered. Besides, inviscid flows are also considered by neglecting the viscous terms.

Two cases are analyzed for the validation of the inviscid flow. In the first case, first and second order flux calculation schemes are applied with 0, 3 and 5 cycles of solution adaptation. It is observed that second order gives closer result relative to the results in the reference data when the number of solution adaptation cycle is low. For 5 cycles of solution adaptation, nearly the same results are obtained with the first and second order schemes. In addition to this, the importance of solution refinement is shown for this case. While solution adaptation is not applied, the pressure distribution deviates in a considerable manner from the reference. However, with 5 cycles of solution refinement, the results are getting very close to the reference data especially for the upper surface where the shock wave occurs, although the convergence rate increases excessively.

In the second case, inviscid flux calculation methods are examined with and without the solution adaptation. While using solution refinement, all methods give nearly the same pressure distributions. The differences among them can be observed if solution refinement technique is not used. In the test cases, where the solution adaptation is not applied, Roe's method gives the best result for the lower surface whereas the best result for the upper surface is obtained by AUSM. In the test cases, where solution adaptation is applied, the locations of shocks and peak point of pressure coefficients are captured very well.

Low Reynolds number flow is tested with two problems. In these problems, generally Cartesian grids are used instead of hybrid grids since boundary layer is large enough so that Cartesian grids can produce the sufficient resolution for low Reynolds numbers. In the first problem, a subsonic flow with a Reynolds number of 5000 is analyzed at a Mach number of 0.5. Tests are carried out by changing only one parameter which is the number of cycles of solution adaptation. It is observed that increase in the cycle number leads to more closer results relative to the reference data both for pressure and skin friction coefficient distributions. However, convergence time increases significantly especially for the last case for which 7 cycles of solution adaptation is used. Yet, the best result is obtained from this case among the cases which quad cells are not used, whereas the peak of the skin friction coefficient is slightly overestimated. Moreover, two cases are used in order to observe the effect of hybrid grids on laminar flows. It is inferred that one can obtain more accurate and non-scattering results especially for skin friction coefficients thanks to hybrid grids.

In the second problem for low Reynolds number flows, a transonic non-symmetric flow is analyzed at a relatively low Reynolds number of 500, a Mach number of 0.8 and an angle of attack of 10 degrees around the NACA 0012 airfoil. The aim is to show that non-symmetric flows with relatively lower Reynolds numbers can be captured by the developed code. With the increase in the number of solution adaptation cycles, the results approach to the ones in the reference. In Case 7, where 6 cycles of solution refinement is applied, the closest results with respect to the reference are obtained, whereas the location of the peak of the pressure distribution cannot be captured exactly.

High Reynolds number flow is examined with one test problem. Since the flow regime is changed from laminar to turbulent at high Reynolds numbers, it is not expected to get accurate result with the developed code. The aim is to investigate the hybrid meshes around the 30P30N airfoil. Some differences at the pressure distribution are obtained by using higher amounts of quad cells in the boundary layer. However, comparing them with the experimental reference result is not credible since the developed solver treats the flow laminarly.

To accelerate the convergence rate, multigrid technique is implemented. The affects of it for inviscid and viscous flows are investigated according to the level number with and without solution adaptation, cycle and iteration number at each step. It is observed that level increase causes a larger speed up ratio both for inviscid and viscous flows. Moreover, if solution adaptation is applied to the grid, multigrid effect becomes more dominant so that the amount of acceleration increases significantly. For example, while a maximum acceleration of 6.27 is obtained in a grid, for which the solution adaptation is not applied, the speed up ratio increases to 16.98 for a solution adapted grid in an inviscid flow. Similarly, the speed up ratio is increased from 1.48 to 7.06 for viscous flows when solution adaptation is used.

As a result of cycle tests, slightly slower convergence rates are obtained by the V-Cycle compared to Saw-Tooth cycle since it requires more memory to store the forcing functions which are necessary during the iterations in the prolongation stage. Moreover, some iteration tests are performed to determine the optimum number of iterations for inviscid and viscous flows. 5 and 20 iterations at each step give the best acceleration amounts for inviscid and viscous flow, respectively. Besides, multigrid effect on hybrid grids are also investigated. In these tests, a larger speed up ratio compared to normal grid is obtained. It is inferred that the multigrid technique is more important in hybrid grids, since the cell number is significantly larger even if solution adaptation is not applied.

Some problems are encountered when it is tried to use multigrid on hybrid grids where solution adaptation is applied. Elimination of these problems can be given as a future work. In addition, turbulence models can be added and the code can be converted into three-dimensional form as future works.

REFERENCES

[1] Anderson J.D.Jr., *Computational Fluid Dynamics: The Basics with Applications,* McGraw-Hill, 1995.

[2] Versteeg H.K. & Malalasekera W., *An Introduction to Computational Fluid Dynamics: The Finite Volume Method,* Pearson/Prentice Hall, 2007.

[3] Potter M.C., Wiggert D.C., Hondzo M., & Shih T.I-P. *Mechanics of Fluids,* Brooks/Cole, 2002.

[4] Blazek J., *Computational Fluid Dynamics: Principles and Applications,* Elsevier, 2005.

[5] Carey G., *Computational Grids: Generation, Adaptation and Solution Strategies,* CRC Press, 1997.

[6] Marshall D.D., *Extending the Functionalities of Cartesian Grid Solvers: Viscous Effects Modeling and MPI Parallelization*, PhD Thesis in the Georgia Institute of Technology, 2002.

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

[8] Ye T., Mittal R., Udaykumar H.S. & Shyy W., *An Accurate Cartesian Grid Method For Viscous Incompressible Flows with Complex Immersed Boundaries*, Journal of Computational Physics 156, 209-240, 1999

[9] Wang Z.J., Cphen R.F., Hariharan N., Przekwas A.J. & Grove D., *A 2^n Tree Based Automated Viscous Cartesian Grid Methodology for Feature Capturing*, AIAA-99-3300, 1999

[10] Tucker P.G. & Pan Z., *A Cartesian Cut Cell Method for Incompressible Viscous Flow*, Applied Mathematical Modelling 24, 591-606, 2000

[11] Wang Z.J., *A Fast Nested Multigrid Viscous Flow Solver for Adaptive Cartesian/Quad Grids*, International Journal for Numerical Methods in Fluids 2000; 33: 657-680, 2000

[12] Kirkpatrick M.P., Armfield S.W. & Kent J.H., *A Representation of Curved Boundaries for the Solution of the Navier-Stokes Equations on a Staggered Three-Dimensional Cartesian Grid*, Journal of Computational Physics 184, 1-36, 2003

[13] AGARD Subcommittee C., *Test Cases for Inviscid Flow Field Methods*, AGARD Advisory Report 211, 1986.

[14] Gilmanov A., Sotiropoulos F. & Balaras E., *A General Reconstruction Algorithm for Simulating Flows with Complex 3D Immersed Boundaries on Cartesian Grids*, Journal of Computational Physics 191, 660-669, 2003

[15] Sanmigual-Rojas E., Ortega-Casanova J., del Pino C. & Fernandez-Feria R., *A Cartesian Grid Finite Difference Method for 2D Incompressible Viscous Flows In Irregular Geometries*, Journal of Computational Physics 204, 302-318, 2005

[16] Verstappen R. & Dröge M., *A Symmetry-Preserving Cartesian Grid Method for Computing a Viscous Flow Past a Circular Cylinder*, C.R. Mechanique 333, 51-57, 2005

[17] Singh R. & Shyy W., *Three-Dimensional Adaptive Cartesian Grid Method with Conservative Interface Restructuring and Reconstruction*, Journal of Computational Physics 224, 150-167, 2007

[18] Ito K., Lai M.C. & Li, Z., *A Well-Conditioned Augmented System for Solving Navier-Stokes Equations in Irregular Domains*, Journal of Computational Physics 228, 2616-2628, 2009

[19] Karagiozis K., Kamakoti R. & Pantano C., *A Low Numerical Dissipation Immersed Interface Method for the Compressible Navier-Stokes Equations*, Journal of Computational Physics 229, 701-727, 2010

[20] Hartmann D., Meinke M. & Schröder W., *A Strictly Conservative Cartesian Cut Cell Method for Compressible Viscous Flows on Adaptive Grids*, Computer Methods in Applied Mechanics and Engineering, 2010

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

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

[23] Hunt J., *An Adaptive 3D Cartesian Approach for the Parallel Computation of Inviscid Flow about Static and Dynamic Configurations*, PhD Thesis in the University of Michigan, 2004.

[24] Toro, E F., *Riemann Solvers and Numerical Methods for Fluid Dynamics,* Springer-Verlag, 1999.

[25] Schlichtig, H., *Boundary Layer Theory*, McGraw-Hill, 7th Ed., 1979.

[26] Lassaline, J.V., *A Navier-Stokes Equation Solver Using Agglomerated Multigrid Featuring Directional Coarsening and Line-Implicit Smoothing*, PhD Thesis in the University of Toronto, 2003.

[27] Mavriplis, D.J., *Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes*, AIAA paper 88-3707, First National Fluid Dynamics Congress, Cincinnati, Ohio, July 24-28, 1988.

[28] Mavriplis, D.J., *Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes*, AIAA paper 89-0120, 27th Aerospace Sciences Meeting, Reno, Nevada, January 9-12, 1989.

[29] Coirier W.J., *An Adaptively Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, PhD Thesis in the University of Michigan, 1994.

[30] Laney C.B., *Computational Gas Dynamics*, Cambridge University Press, 1998.

[31] Liou M.S. & Steffen C.J., *A New Flux Splitting Scheme*, Journal of Computational Physics, Vol. 107, pp. 23-39, 1993.

[32] Wada, Y. & Liou M.S., *An Accurate and Robust Flux Splitting Scheme for Shock and Contact Discontinuities.*, M S. 3, s.l. : Siam J. Sci. Comput.,1997, Vol. 18, pp. 633-657.

[33] Aftosmis, M., Gaitonde, D. & Tavares, T.S., AIAA-94-0415, (unpublished), 1994.

[34] Barth T.J., & Jespersen D.C., *The Design and Application of Upwind Schemes on Unstructured Meshes*, AIAA Paper AIAA-89-0366, 1989.

[35] Trottenberg U., Oosterlee C.W., Schüller A., *Multigrid,* Academic Press, 2001.

[36] Brandt A., *Multi-Level Adaptive Solutions to Boundary-Value Problems,* Mathematics for Computation, Vol. 31, pp. 333-390, 1977.

[37] Jameson A., *Solution of the Euler Equations for Two-Dimensional Transonic Flow by a Multigrid Method*, Applied Mathematics and Computation, Vol. 13 Issues 3-4, pp. 327-355, 1983.

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

[39] Russell D. & Wang Z.J., *A Cartesian Grid Method for Modeling Multiple Moving Objects in 2D Incompressible Viscous Flow*, Journal of Computational Physics 191, 177-205, 2003

[40] Gooch C.F., *Solution of the Navier-Stokes Equations on Locally-Refined Cartesian Meshes Using State-Vector Splitting*, PhD Thesis in the Stanford University, 1993.

[41] Bonfiglioli A., *Compressible, Viscous (Laminar) Flow Past a NACA 0012 Profile*, 1998, at: http://www.unibas.it/utenti/bonfiglioli/node6.html, Last Access On February 2011.

[42] Sangho K., Alonso J.J., & Jameson A., *Design Optimization of High-Lift Configurations Using a Viscous Continuous Adjoint Method*, AIAA Paper AIAA-2002-0844, 2002.

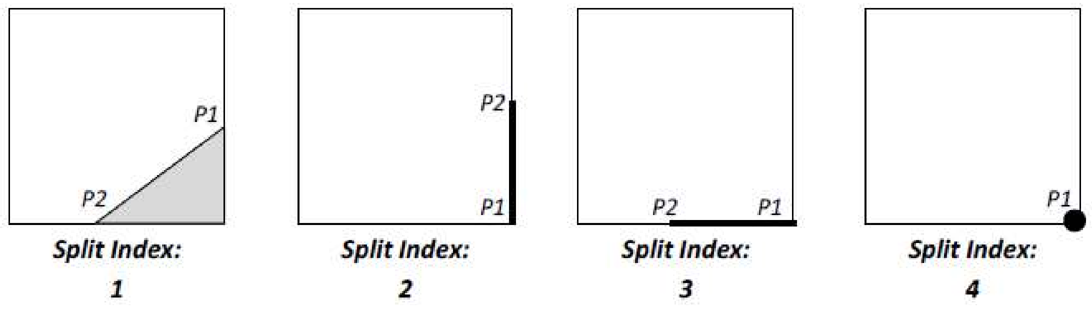
APPENDIX A

CUT AND SPLIT CELLS

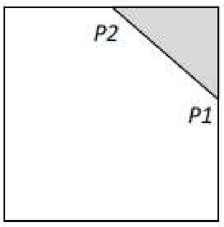
As mentioned, there are a lot of alternatives available for cut and split cells. In this Appendix, the alternatives of cut and split cells are expressed in terms of their square and split indices, seperately. The gray regions represents the part inside the geometry of the cell. Moreover, the sorted intersection points are indicated with P1 to P4.

A.1 CUT CELLS

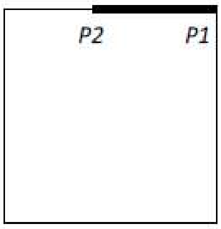
A.1.1 Square Index of 1



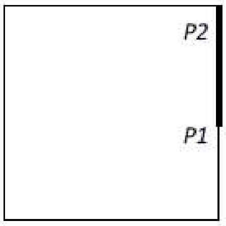
A.1.2 Square Index of 2



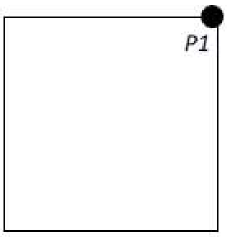
Split Index: 1



Split Index: 2

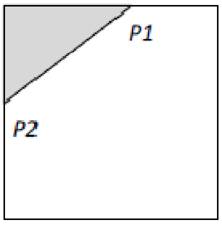


Split Index: 3

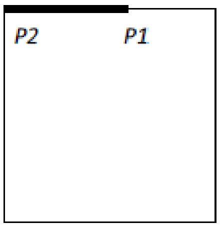


Split Index: 4

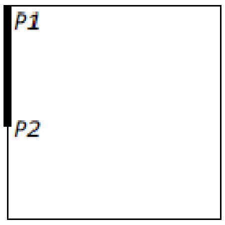
A.1.3 Square Index of 4



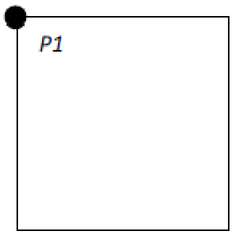
Split Index: 1



Split Index: 2

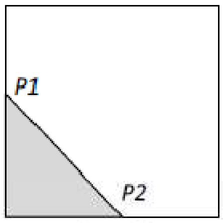


Split Index: 3

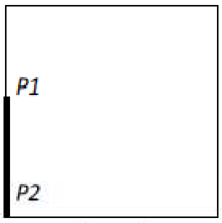


Split Index: 4

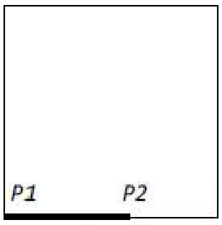
A.1.4 Square Index of 8



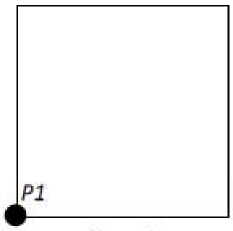
Split Index: 1



Split Index: 2

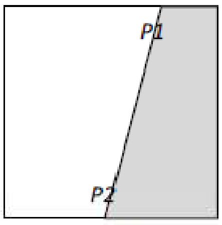


Split Index: 3

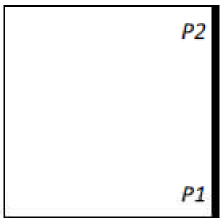


Split Index: 4

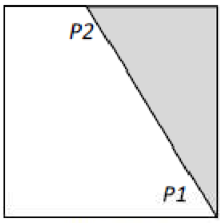
A.1.5 Square Index of 3



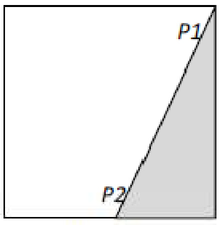
Split Index: 1



Split Index: 2

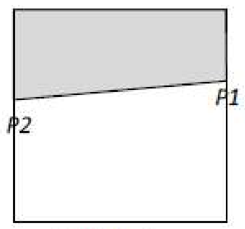


Split Index: 3

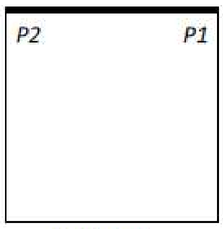


Split Index: 4

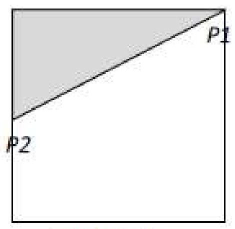
A.1.6 Square Index of 6



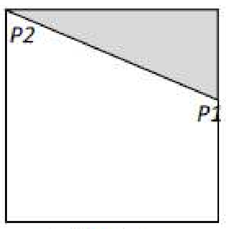
Split Index: 1



Split Index: 2

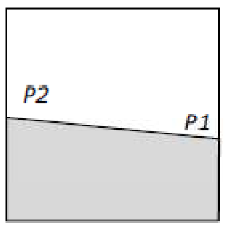


Split Index: 3

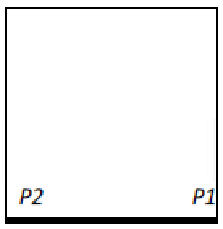


Split Index: 4

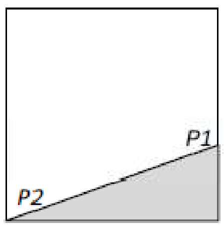
A.1.7 Square Index of 9



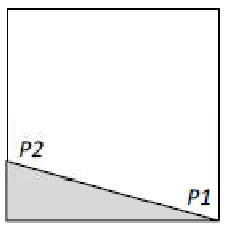
Split Index: 1



Split Index: 2

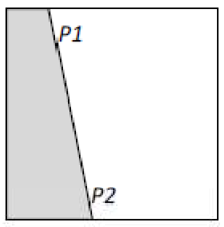


Split Index: 3

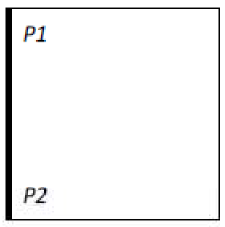


Split Index: 4

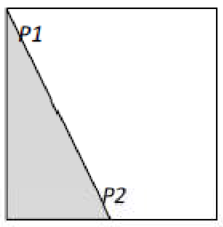
A.1.8 Square Index of 12



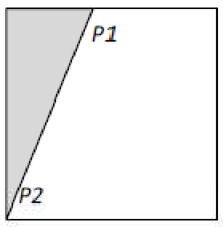
Split Index: 1



Split Index: 2

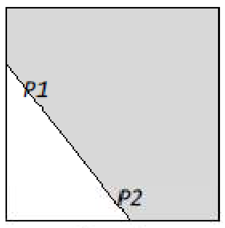


Split Index: 3

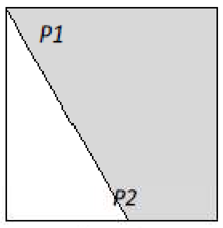


Split Index: 4

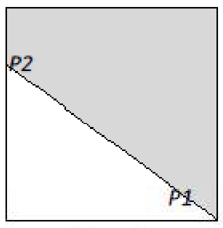
A.1.9 Square Index of 7



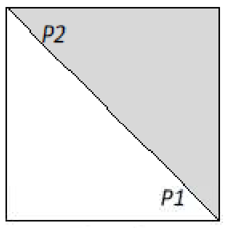
Split Index: 1



Split Index: 2

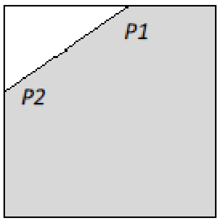


Split Index: 3

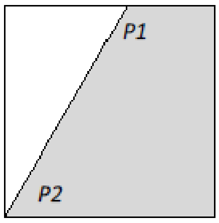


Split Index: 4

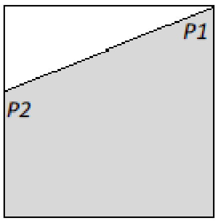
A.1.10 Square Index of 11



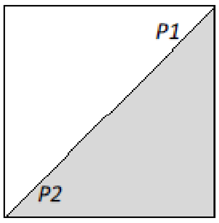
Split Index: 1



Split Index: 2

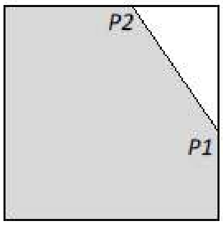


Split Index: 3

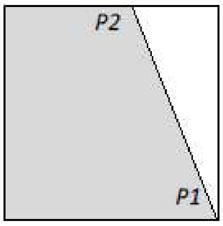


Split Index: 4

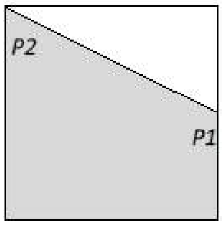
A.1.11 Square Index of 13



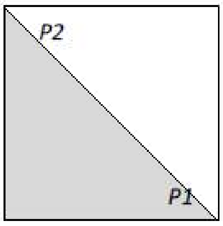
Split Index: 1



Split Index: 2

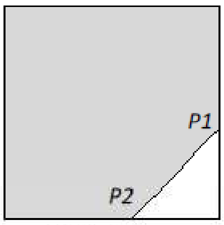


Split Index: 3

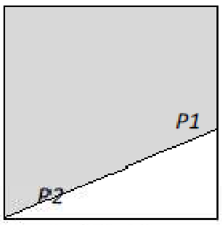


Split Index: 4

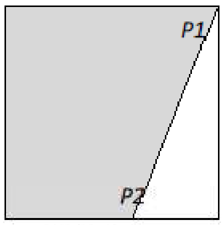
A.1.12 Square Index of 14



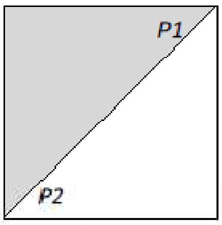
Split Index: 1



Split Index: 2



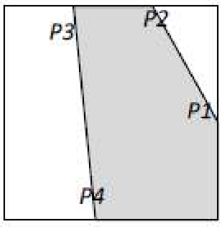
Split Index: 3



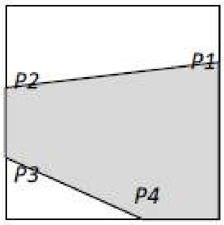
Split Index: 4

A.2 SPLIT CELLS

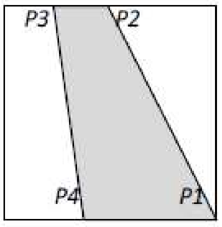
A.2.1 Square Index of 1



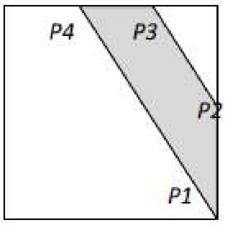
Split Index: 1



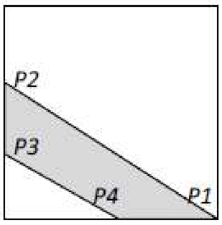
Split Index: 2



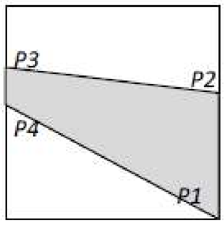
Split Index: 3



Split Index: 4

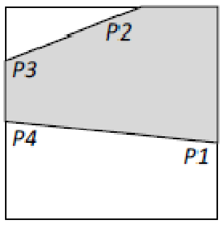


Split Index: 5

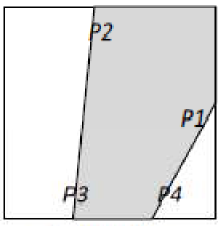


Split Index: 6

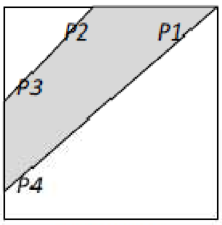
A.2.2 Square Index of 2



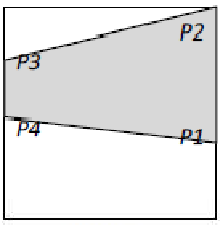
Split Index: 1



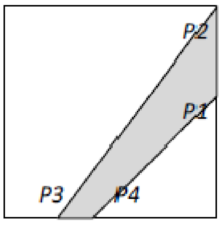
Split Index: 2



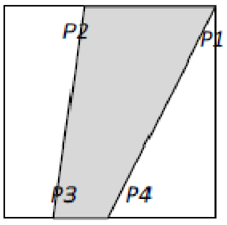
Split Index: 3



Split Index: 4

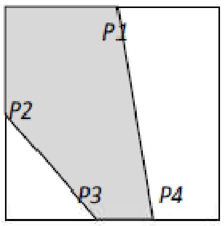


Split Index: 5

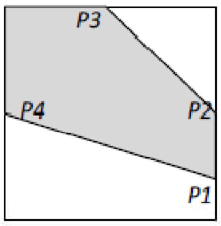


Split Index: 6

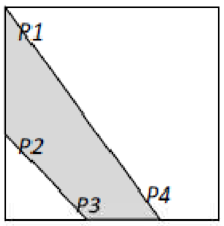
A.2.3 Square Index of 4



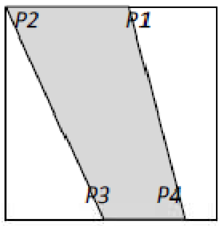
Split Index: 1



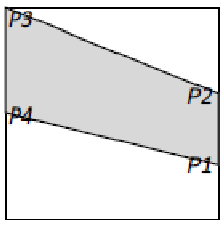
Split Index: 2



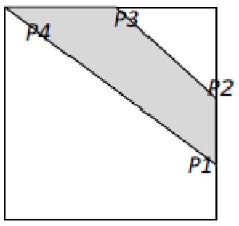
Split Index: 3



Split Index: 4

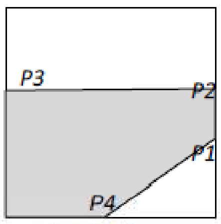


Split Index: 5

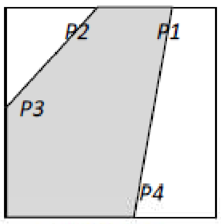


Split Index: 6

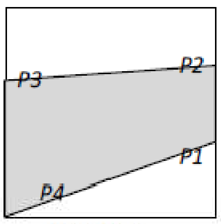
A.2.4 Square Index of 8



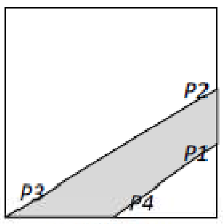
Split Index: 1



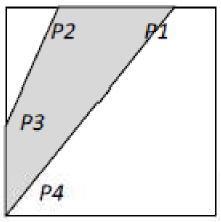
Split Index: 2



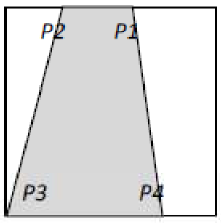
Split Index: 3



Split Index: 4

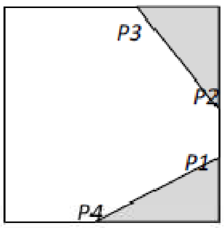


Split Index: 5

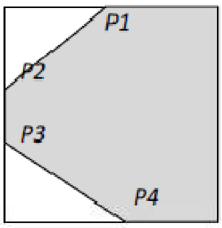


Split Index: 6

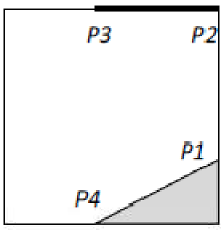
A.2.5 Square Index of 3



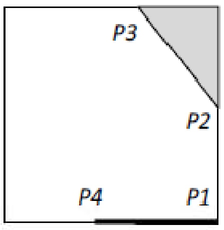
Split Index: 1



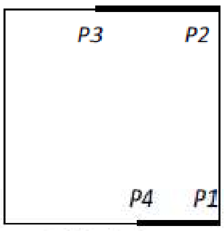
Split Index: 2



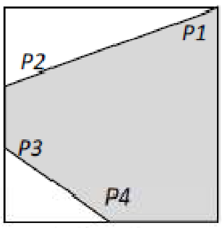
Split Index: 3



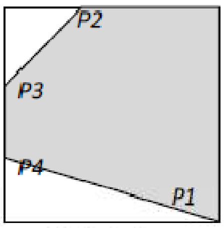
Split Index: 4



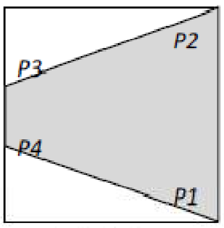
Split Index: 5



Split Index: 6

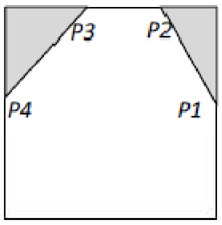


Split Index: 7

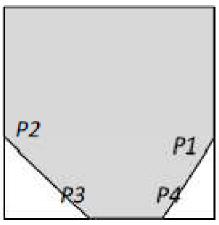


Split Index: 8

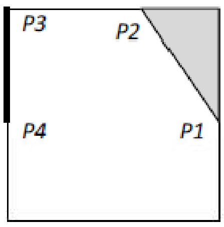
A.2.6 Square Index of 6



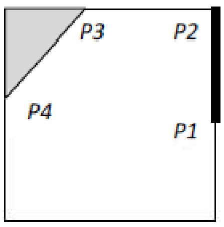
Split Index: 1



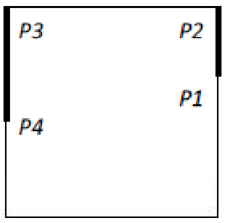
Split Index: 2



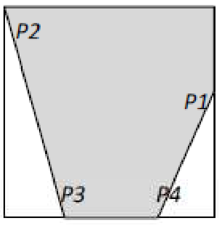
Split Index: 3



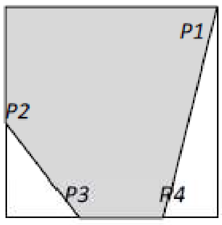
Split Index: 4



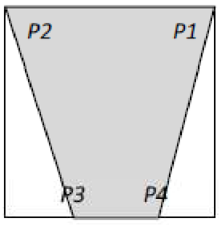
Split Index: 5



Split Index: 6

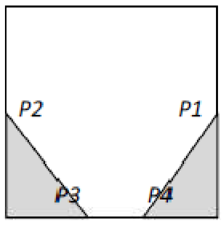


Split Index: 7

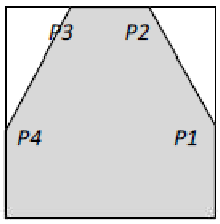


Split Index: 8

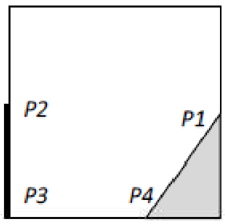
A.2.7 Square Index of 9



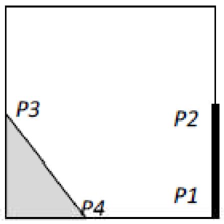
Split Index: 1



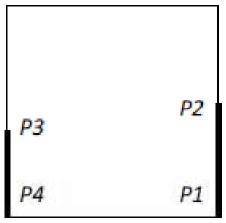
Split Index: 2



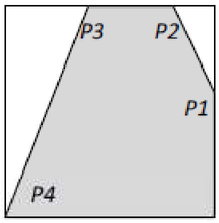
Split Index: 3



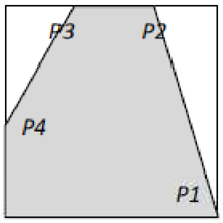
Split Index: 4



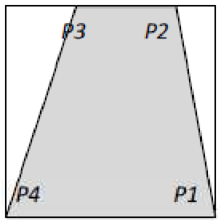
Split Index: 5



Split Index: 6

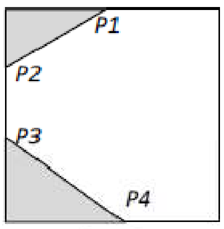


Split Index: 7

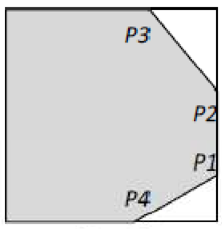


Split Index: 8

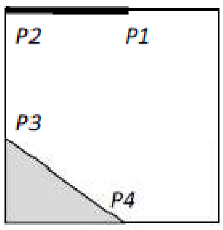
A.2.8 Square Index of 12



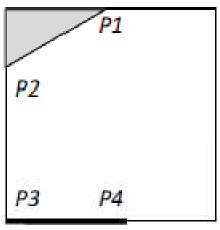
Split Index: 1



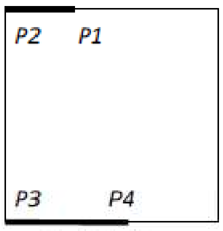
Split Index: 2



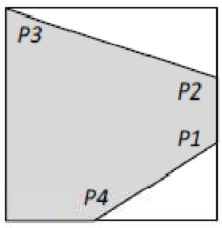
Split Index: 3



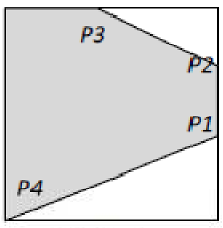
Split Index: 4



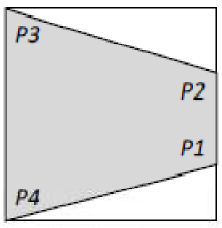
Split Index: 5



Split Index: 6

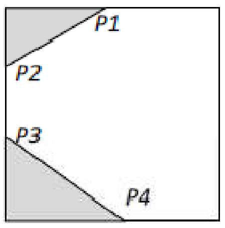


Split Index: 7

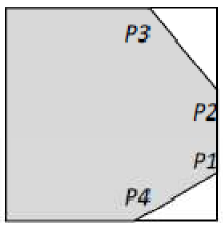


Split Index: 8

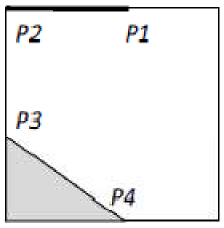
A.2.9 Square Index of 7



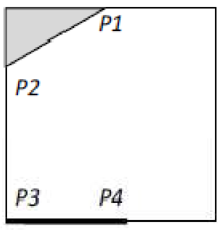
Split Index:
1



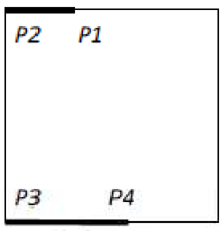
Split Index:
2



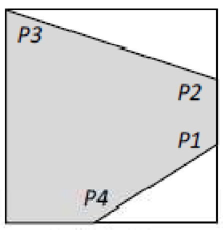
Split Index:
3



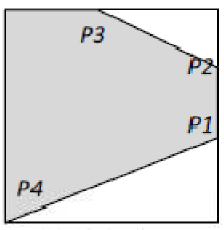
Split Index:
4



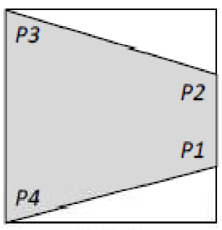
Split Index:
5



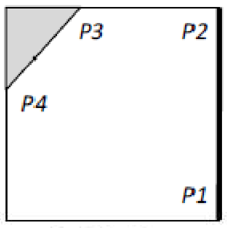
Split Index:
6



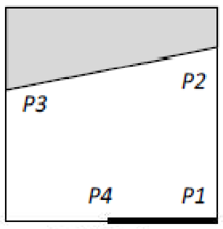
Split Index:
7



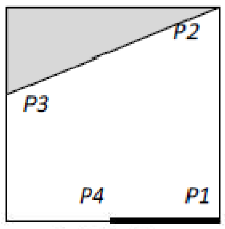
Split Index:
8



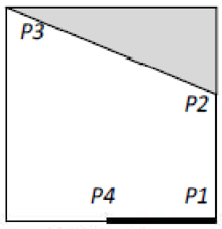
Split Index:
9



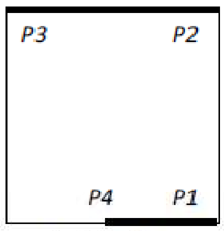
Split Index:
10



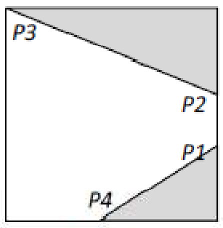
Split Index:
11



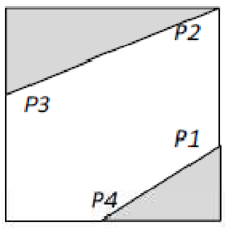
Split Index:
12



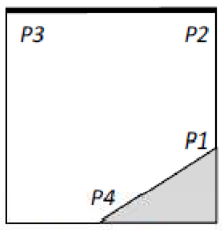
Split Index:
13



Split Index:
14

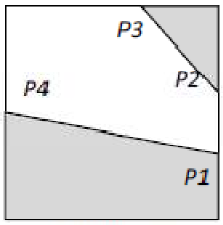


Split Index:
15

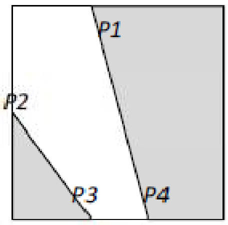


Split Index:
16

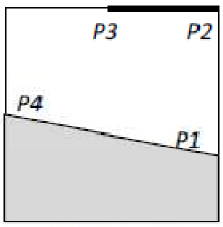
A.2.10 Square Index of 11



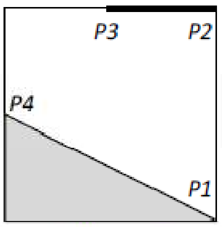
Split Index: 1



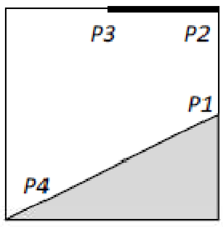
Split Index: 2



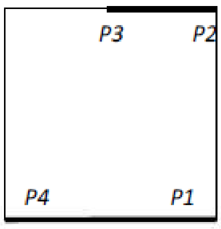
Split Index: 3



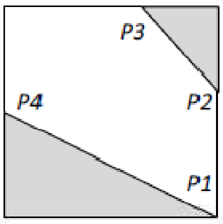
Split Index: 4



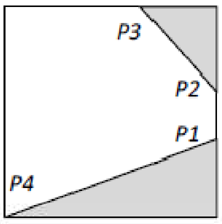
Split Index: 5



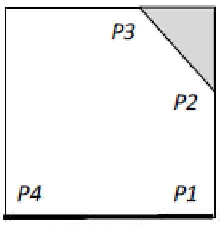
Split Index: 6



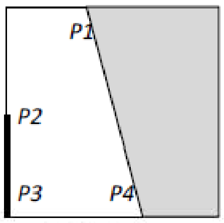
Split Index: 7



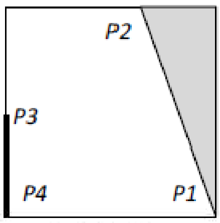
Split Index: 8



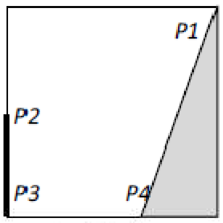
Split Index: 9



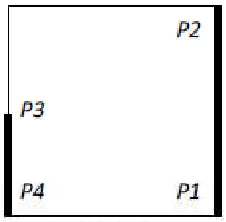
Split Index: 10



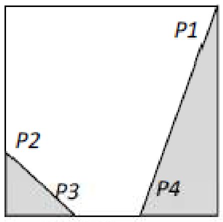
Split Index: 11



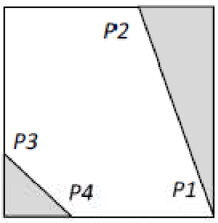
Split Index: 12



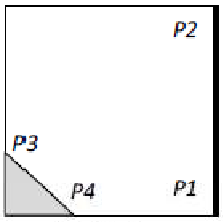
Split Index: 13



Split Index: 14

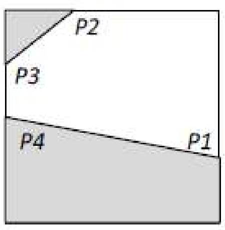


Split Index: 15

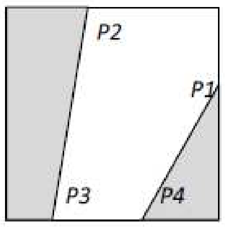


Split Index: 16

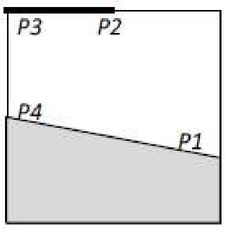
A.2.11 Square Index of 13



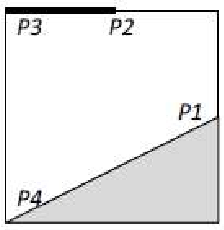
Split Index: 1



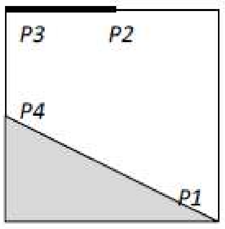
Split Index: 2



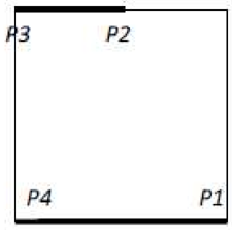
Split Index: 3



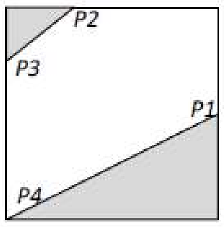
Split Index: 4



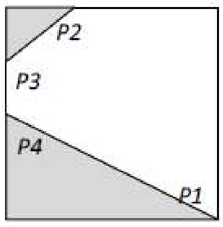
Split Index: 5



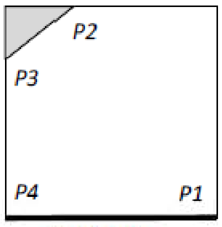
Split Index: 6



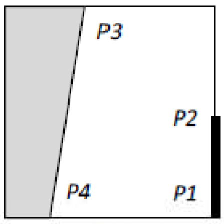
Split Index: 7



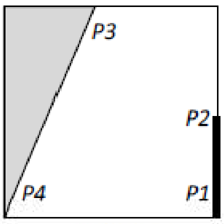
Split Index: 8



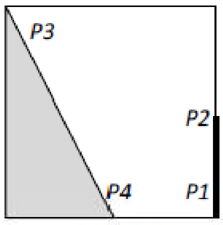
Split Index: 9



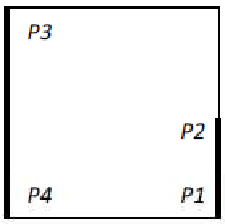
Split Index: 10



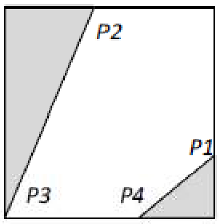
Split Index: 11



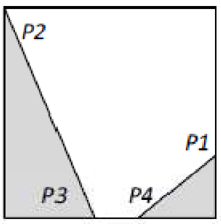
Split Index: 12



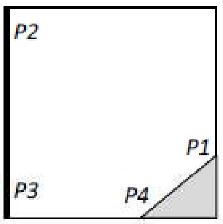
Split Index: 13



Split Index: 14

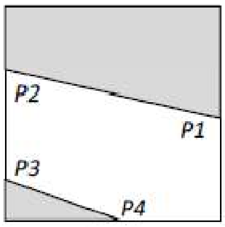


Split Index: 15

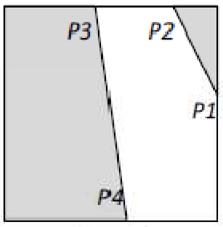


Split Index: 16

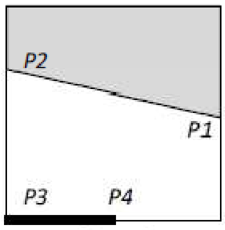
A.2.12 Square Index of 14



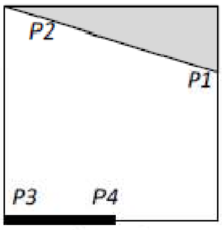
Split Index: 1



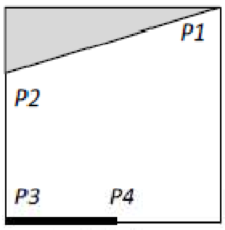
Split Index: 2



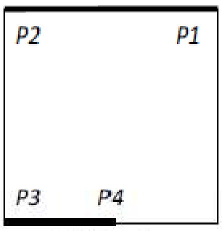
Split Index: 3



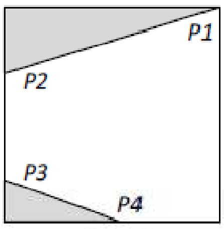
Split Index: 4



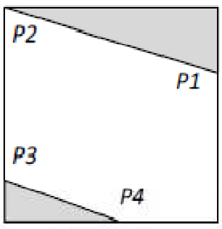
Split Index: 5



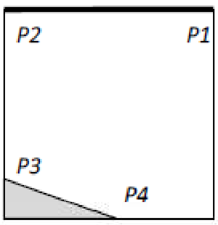
Split Index: 6



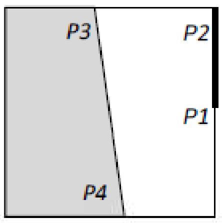
Split Index: 7



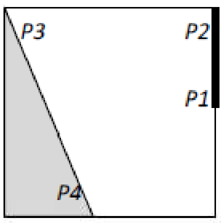
Split Index: 8



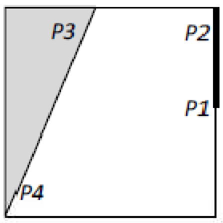
Split Index: 9



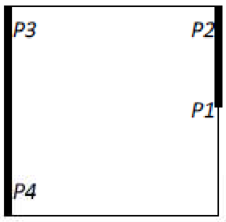
Split Index: 10



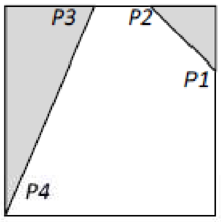
Split Index: 11



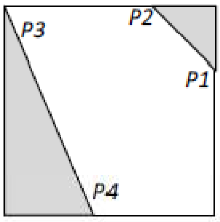
Split Index: 12



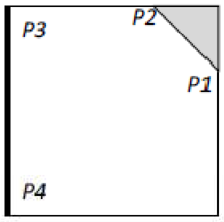
Split Index: 13



Split Index: 14

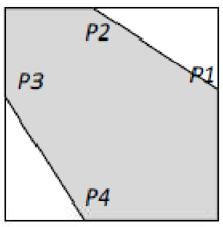


Split Index: 15

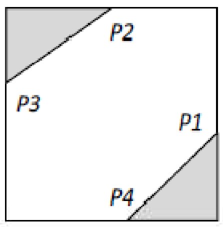


Split Index: 16

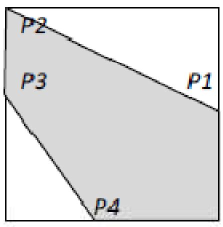
A.2.13 Square Index of 5



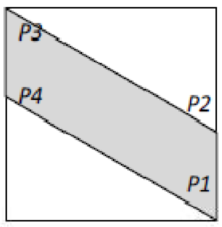
Split Index: 1



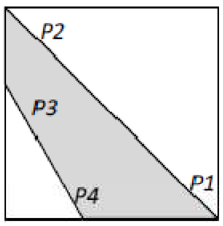
Split Index: 2



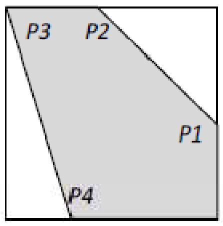
Split Index: 3



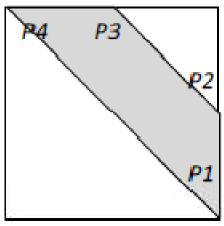
Split Index: 4



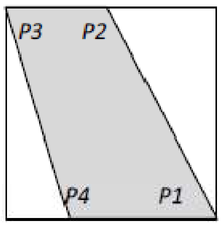
Split Index: 5



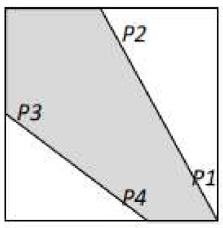
Split Index: 6



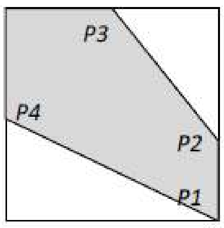
Split Index: 7



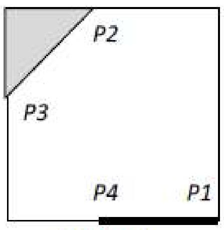
Split Index: 8



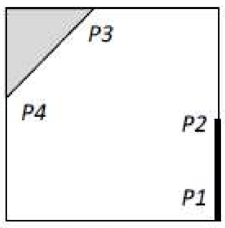
Split Index: 9



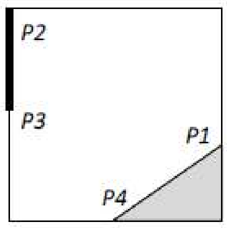
Split Index: 10



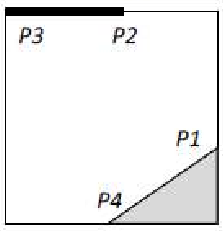
Split Index: 11



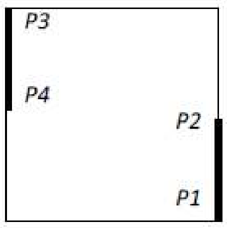
Split Index: 12



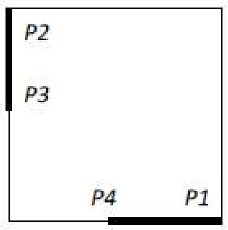
Split Index: 13



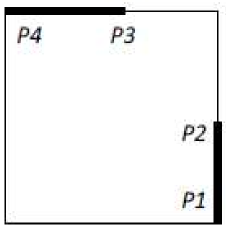
Split Index: 14



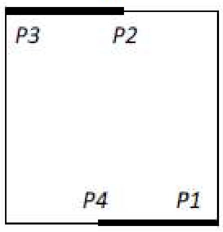
Split Index: 15



Split Index: 16

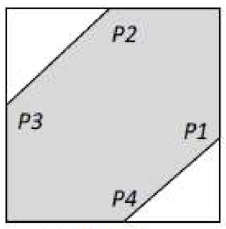


Split Index: 17

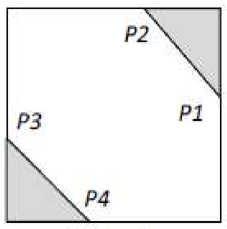


Split Index: 18

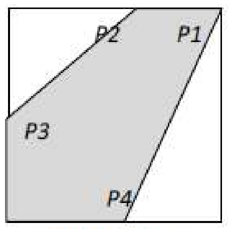
A.2.14 Square Index of 10



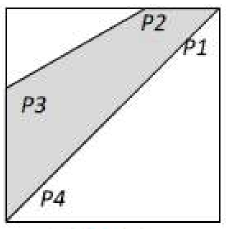
Split Index:
1



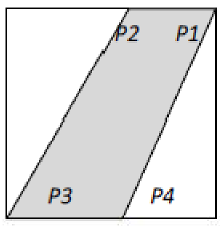
Split Index:
2



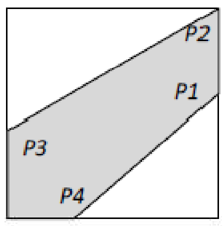
Split Index:
3



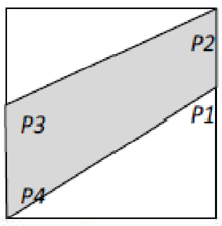
Split Index:
4



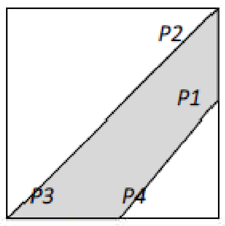
Split Index:
5



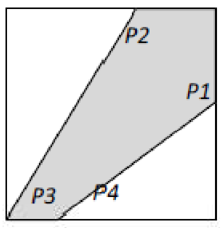
Split Index:
6



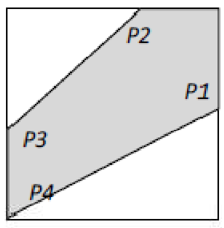
Split Index:
7



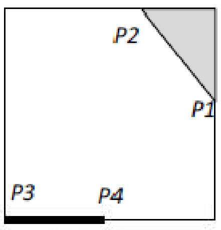
Split Index:
8



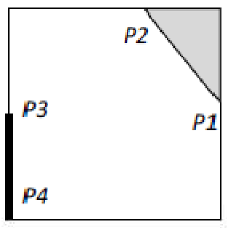
Split Index:
9



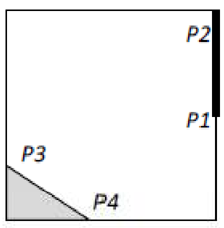
Split Index:
10



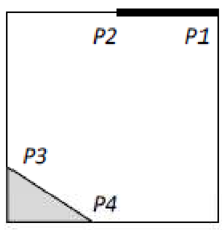
Split Index:
11



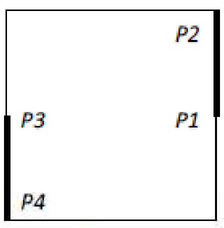
Split Index:
12



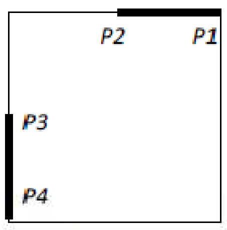
Split Index:
13



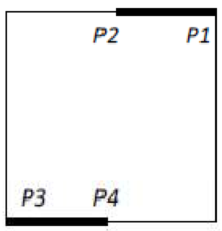
Split Index:
14



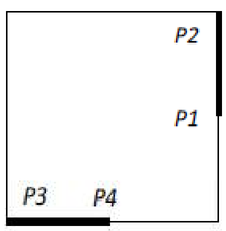
Split Index:
15



Split Index:
16

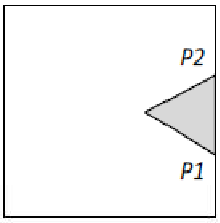


Split Index:
17

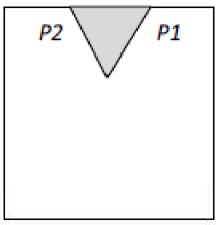


Split Index:
18

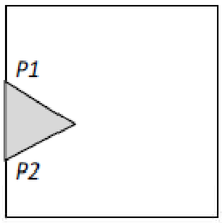
A.2.15 Square Index of -15



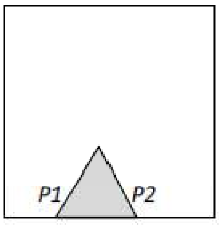
Split Index:
1



Split Index:
2

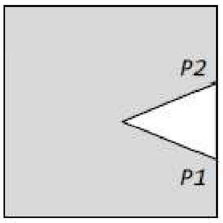


Split Index:
3

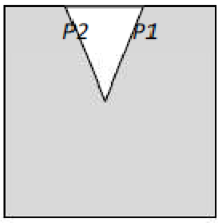


Split Index:
4

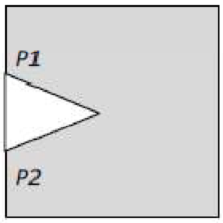
A.2.16 Square Index of -20



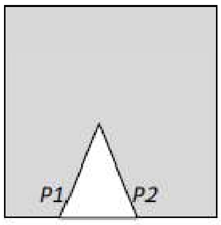
Split Index:
1



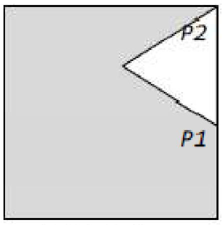
Split Index:
2



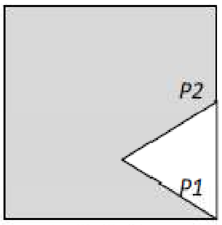
Split Index:
3



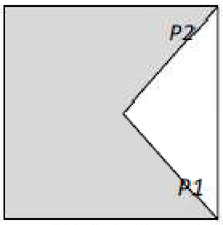
Split Index:
4



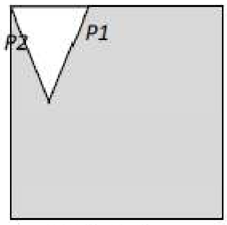
Split Index:
5



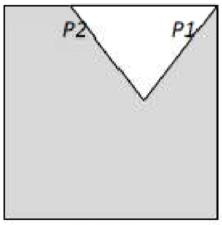
Split Index:
6



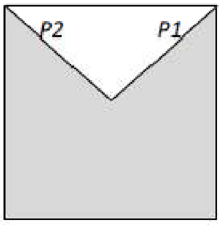
Split Index:
7



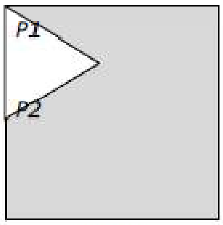
Split Index:
8



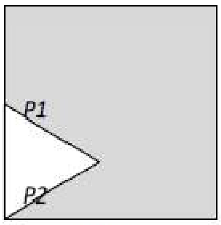
Split Index:
9



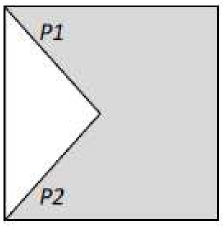
Split Index:
10



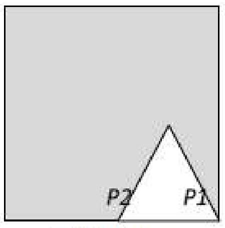
Split Index:
11



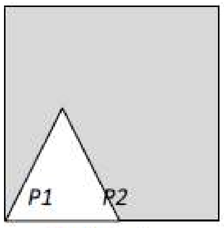
Split Index:
12



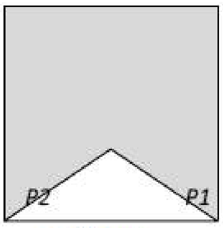
Split Index:
13



Split Index:
14

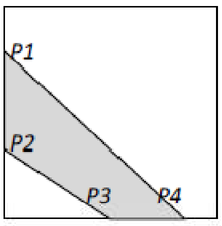


Split Index:
15

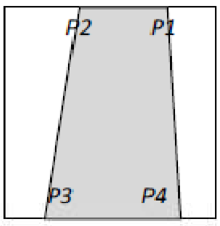


Split Index:
16

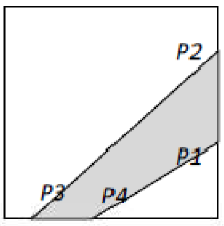
A.2.17 Square Index of -25



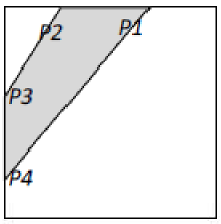
Split Index:
1



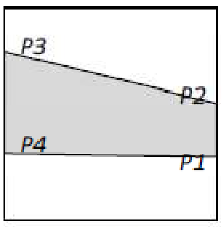
Split Index:
2



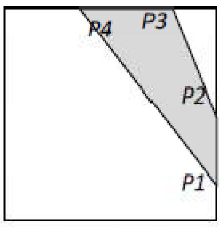
Split Index:
3



Split Index:
4



Split Index:
5



Split Index:
6

APPENDIX B

SAMPLE FILE FORMATS

B.1 SAMPLE MESH INPUT FILE

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----------------------------------------------------------------------------------------------
                    MESH GENERATION INPUTS
-----------------------------------------------------------------------------------------------
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

A) AIRFOIL SELECTION
-----------------------------------------------------------------------------------------------
NLR7301.dat          :  1) Airfoil Name
-----------------------------------------------------------------------------------------------

B) GRID INPUTS
-----------------------------------------------------------------------------------------------
20                   :  2) Outer Boundary Size Factor
8                    :  3) Level of Uniform Mesh
0                    :  4) Shift Amount in X Axis
0                    :  5) Shift Amount in Y Axis
-----------------------------------------------------------------------------------------------

C) BOX ADAPTATION INPUTS
-----------------------------------------------------------------------------------------------
```

```
1.5                  :  6) Boundary Size Factor for X Axis
2.5                  :  7) Boundary Size Factor for Y Axis
0.05                 :  8) Body Division Factor
----------------------------------------------------------------------------------------------------


D) CUT-SPLIT ADAPTATION INPUTS
----------------------------------------------------------------------------------------------------
0                    :  9) Cut-Split Adaptation Cycle
----------------------------------------------------------------------------------------------------

E) CURVATURE ADAPTATION INPUTS
----------------------------------------------------------------------------------------------------
0                    : 10) Curvature Adaptation Cycle
170                  : 11) Threshold Angle
----------------------------------------------------------------------------------------------------

F) BOUNDARY LAYER INPUTS
----------------------------------------------------------------------------------------------------
0                    : 12) Quad Cells Usage (1:Yes, 0:No)
1.1                  : 13) Stretch Factor
16                   : 14) Row Number
----------------------------------------------------------------------------------------------------
```

B.2 SAMPLE INVISCID SOLUTION INPUT FILE

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-------------------------------------------------------------------------------------------------
                    SOLUTION INPUTS FOR INVISCID SOLVER
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

A) FLOW INPUTS
-------------------------------------------------------------------------------------------------
0.85                    :  1) Mach Number
1.0                     :  2) Angle of Attack (in degrees)
-------------------------------------------------------------------------------------------------

B) MEDIUM INPUTS
-------------------------------------------------------------------------------------------------
1.4                     :  3) Specific Heat Ratio
-------------------------------------------------------------------------------------------------

C) SOLVER INPUTS
-------------------------------------------------------------------------------------------------
1                       :  4) Order of Scheme (1: First, 2: Second)
1                       :  5) Flux Method (1: Roe, 2: AUSM, 3: AUSMD, 4: AUSMV)
3                       :  6) Multistage Number (3: Three, 4: Four, 5: Five)
1                       :  7) CFL Safety Factor (between 0 and 1)
0                       :  8) Gradient Limiting (1:Yes, 0:No)
-------------------------------------------------------------------------------------------------

D) SOLUTION ADAPTATION INPUTS
-------------------------------------------------------------------------------------------------
0                       :  9) Refinement Cycle (0 to 6)
```

```
15                    : 10) Coefficient of Refinement Based On Residual
------------------------------------------------------------------------------------------------

E) MULTIGRID INPUTS
------------------------------------------------------------------------------------------------
1                     : 11) Multigrid Type (1: Saw-Tooth, 2: v-Type)
7                     : 12) Multigrid Cycle (0 to 7)
10                    : 13) Fine Grid Iteration Cycle
10                    : 14) Mid Step Iteration Cycle
10                    : 15) Final Grid Iteration Cycle
------------------------------------------------------------------------------------------------

F) ITERATION INPUTS
------------------------------------------------------------------------------------------------
10                    : 16) Iteration Interval of Writing to the Screen
-8.                   : 17) Minimum Log of RMS
------------------------------------------------------------------------------------------------
```

B.3 SAMPLE VISCOUS SOLUTION INPUT FILE

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
------------------------------------------------------------------------------------------------
                    SOLUTION INPUTS FOR VISCOUS SOLVER
------------------------------------------------------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

A) FLOW INPUTS
------------------------------------------------------------------------------------------------
0.5                   :  1) Mach Number
0.0                   :  2) Angle of Attack (in degrees)
5000                  :  3) Reynolds Number
```

```
0.72                    :  4) Prandtl Number
----------------------------------------------------------------------------------------------------


B) MEDIUM INPUTS
----------------------------------------------------------------------------------------------------
1.4                     :  5) Specific Heat Ratio
273.15                  :  6) Free Stream Temperature (in Kelvin)
----------------------------------------------------------------------------------------------------


C) SOLVER INPUTS
----------------------------------------------------------------------------------------------------
1                       :  7) Order of Scheme (1: First, 2: Second)
2                       :  8) Flux Method (1: Roe, 2: AUSM, 3: AUSMD, 4: AUSMV)
3                       :  9) Multistage Number (3: Three, 4: Four, 5: Five)
0.5                     : 10) CFL Safety Factor (between 0 and 1)
0                       : 11) Gradient Limiting (1:Yes, 0:No)
0.25                    : 12) Time Step Coefficient
----------------------------------------------------------------------------------------------------



D) SOLUTION ADAPTATION INPUTS
----------------------------------------------------------------------------------------------------
0                       : 13) Refinement Cycle (0 to 6)
20                      : 14) Coefficient of Refinement Based On Residual
----------------------------------------------------------------------------------------------------


E) MULTIGRID INPUTS
----------------------------------------------------------------------------------------------------
1                       : 15) Multigrid Type (1: Saw-Tooth, 2: v-Type)
0                       : 16) Multigrid Level (0 to 7)
10                      : 17) Fine Grid Iteration Cycle
10                      : 18) Mid Step Iteration Cycle
```

```
10                     : 19) Final Grid Iteration Cycle
--------------------------------------------------------------------------------------------

F) ITERATION INPUTS
--------------------------------------------------------------------------------------------
10                     : 20) Iteration Interval of Writing to the Screen
-6.                    : 21) Minimum Log of RMS
--------------------------------------------------------------------------------------------
```

B.4 SAMPLE MESH OUTPUT FILE

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--------------------------------------------------------------------------------------------
                                   MESH OUTPUT INFO
--------------------------------------------------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

NON-ADAPTED GRID
--------------------------------------------------------------------------------------------

H-GRID
------------------------
Out Cell No            : 1883
Cut Cell No            : 238
Split Cell No          : 3 (2 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 2124
Time                   : 0.25 seconds

2H-GRID
```

```
-------------------------
Out Cell No          : 680
Cut Cell No          : 123
Split Cell No        : 2 (1 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 805

4H-GRID
-------------------------
Out Cell No          : 318
Cut Cell No          : 65
Split Cell No        : 2 (1 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 385

8H-GRID
-------------------------
Out Cell No          : 189
Cut Cell No          : 37
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 226

16H-GRID
-------------------------
Out Cell No          : 159
Cut Cell No          : 19
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 178

32H-GRID
```

```
------------------------
Out Cell No           : 130
Cut Cell No           : 12
Split Cell No         : 0 (0 cells have 2 CV's)
Quad Cell No          : 0
Total Cell No         : 142

64H-GRID
------------------------
Out Cell No           : 98
Cut Cell No           : 11
Split Cell No         : 0 (0 cells have 2 CV's)
Quad Cell No          : 0
Total Cell No         : 109

128H-GRID
------------------------
Out Cell No           : 54
Cut Cell No           : 7
Split Cell No         : 0 (0 cells have 2 CV's)
Quad Cell No          : 0
Total Cell No         : 61
-----------------------------------------------------------------------------------------------


1. ADAPTED GRID
-----------------------------------------------------------------------------------------------


H-GRID
------------------------
Out Cell No           : 3271
Cut Cell No           : 249
```

```
Split Cell No        : 3 (2 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 3523

2H-GRID
-------------------------
Out Cell No          : 932
Cut Cell No          : 129
Split Cell No        : 2 (1 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 1063

4H-GRID
-------------------------
Out Cell No          : 440
Cut Cell No          : 66
Split Cell No        : 2 (1 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 508

8H-GRID
-------------------------
Out Cell No          : 209
Cut Cell No          : 41
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 250

16H-GRID
-------------------------
Out Cell No          : 164
Cut Cell No          : 26
```

```
Split Cell No          : 0 (0 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 190

32H-GRID
------------------------
Out Cell No            : 144
Cut Cell No            : 16
Split Cell No          : 0 (0 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 160

64H-GRID
------------------------
Out Cell No            : 110
Cut Cell No            : 11
Split Cell No          : 0 (0 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 121

128H-GRID
------------------------
Out Cell No            : 82
Cut Cell No            : 9
Split Cell No          : 0 (0 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 91
------------------------------------------------------------------------------------------------


2. ADAPTED GRID
------------------------------------------------------------------------------------------------
```

```
H-GRID
------------------------
Out Cell No            : 5979
Cut Cell No            : 266
Split Cell No          : 3 (2 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 6248

2H-GRID
------------------------
Out Cell No            : 1735
Cut Cell No            : 144
Split Cell No          : 2 (1 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 1881

4H-GRID
------------------------
Out Cell No            : 544
Cut Cell No            : 78
Split Cell No          : 2 (1 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 624

8H-GRID
------------------------
Out Cell No            : 274
Cut Cell No            : 43
Split Cell No          : 0 (0 cells have 2 CV's)
Quad Cell No           : 0
Total Cell No          : 317
```

```
16H-GRID
------------------------
Out Cell No          : 186
Cut Cell No          : 31
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 217

32H-GRID
------------------------
Out Cell No          : 151
Cut Cell No          : 24
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 175

64H-GRID
------------------------
Out Cell No          : 124
Cut Cell No          : 15
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 139

128H-GRID
------------------------
Out Cell No          : 94
Cut Cell No          : 9
Split Cell No        : 0 (0 cells have 2 CV's)
Quad Cell No         : 0
Total Cell No        : 103
------------------------------------------------------------------------------------------------
```

B.5 SAMPLE SOLUTION OUTPUT FILE

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
------------------------------------------------------------------------------------------------
                         SOLUTION OUTPUT INFO
------------------------------------------------------------------------------------------------
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

DATA
------------------------------------------------------------------------------------------------
Time                    : 0 hours 5 minutes 56 seconds
Iteration               : 4410
log(RMS)                : -10.02105
------------------------------------------------------------------------------------------------

COEFFICIENTS
------------------------------------------------------------------------------------------------
Drag Coefficient        : 0.04717
Lift Coefficient        : 0.96187
------------------------------------------------------------------------------------------------
```

APPENDIX C

AIRFOIL COORDINATES

In this appendix, coordinates of three airfoil are given as nodes. In a row, while first value is the node number, the second and third one represents the node's *x* and *y* coordinates, respectively.

C.1 NACA 0012

Table C.1 Coordinates of NACA 0012

NODE	X	Y
1	1.00000	0.00000
2	0.98530	-0.00216
3	0.96662	-0.00480
4	0.94288	-0.00806
5	0.91268	-0.01208
6	0.87428	-0.01697
7	0.84541	-0.02045
8	0.81783	-0.02372
9	0.79431	-0.02644
10	0.76315	-0.02988
11	0.73347	-0.03301
12	0.70578	-0.03582
13	0.68691	-0.03764
14	0.66688	-0.03957
15	0.64397	-0.04167
16	0.62271	-0.04355
17	0.59235	-0.04611
18	0.56483	-0.04830

Table C.1 Coordinates of NACA 0012 (continued)

19	0.54025	-0.05013
20	0.51179	-0.05210
21	0.49094	-0.05344
22	0.47494	-0.05440
23	0.45118	-0.05570
24	0.42719	-0.05687
25	0.40283	-0.05789
26	0.36530	-0.05911
27	0.33031	-0.05980
28	0.29878	-0.06001
29	0.26716	-0.05976
30	0.23892	-0.05909
31	0.21375	-0.05809
32	0.18516	-0.05642
33	0.16156	-0.05454
34	0.13887	-0.05221
35	0.12371	-0.05032
36	0.10329	-0.04727
37	0.09080	-0.04506
38	0.07599	-0.04201
39	0.06628	-0.03971
40	0.05510	-0.03669
41	0.04669	-0.03408
42	0.03916	-0.03145
43	0.03187	-0.02858
44	0.02673	-0.02631
45	0.02310	-0.02456
46	0.02117	-0.02356
47	0.01894	-0.02234
48	0.01681	-0.02109
49	0.01563	-0.02036
50	0.01440	-0.01956
51	0.01340	-0.01887
52	0.01240	-0.01816
53	0.01142	-0.01743
54	0.01030	-0.01654
55	0.00891	-0.01534
56	0.00792	-0.01442
57	0.00728	-0.01379
58	0.00662	-0.01310
59	0.00594	-0.01235

Table C.1 Coordinates of NACA 0012 (continued)

60	0.00537	-0.01169
61	0.00476	-0.01092
62	0.00424	-0.01022
63	0.00386	-0.00969
64	0.00323	-0.00870
65	0.00261	-0.00763
66	0.00221	-0.00684
67	0.00178	-0.00593
68	0.00137	-0.00492
69	0.00102	-0.00396
70	0.00073	-0.00306
71	0.00051	-0.00227
72	0.00029	-0.00139
73	0.00014	-0.00072
74	0.00000	0.00000
75	0.00014	0.00072
76	0.00029	0.00139
77	0.00051	0.00227
78	0.00073	0.00306
79	0.00102	0.00396
80	0.00137	0.00492
81	0.00178	0.00593
82	0.00221	0.00684
83	0.00261	0.00763
84	0.00323	0.00870
85	0.00386	0.00969
86	0.00424	0.01022
87	0.00476	0.01092
88	0.00537	0.01169
89	0.00594	0.01235
90	0.00662	0.01310
91	0.00728	0.01379
92	0.00792	0.01442
93	0.00891	0.01534
94	0.01030	0.01654
95	0.01142	0.01743
96	0.01240	0.01816
97	0.01340	0.01887
98	0.01440	0.01956
99	0.01563	0.02036
100	0.01681	0.02109

Table C.1 Coordinates of NACA 0012 (continued)

101	0.01894	0.02234
102	0.02117	0.02356
103	0.02310	0.02456
104	0.02673	0.02631
105	0.03187	0.02858
106	0.03916	0.03145
107	0.04669	0.03408
108	0.05510	0.03669
109	0.06628	0.03971
110	0.07599	0.04201
111	0.09080	0.04506
112	0.10329	0.04727
113	0.12371	0.05032
114	0.13887	0.05221
115	0.16156	0.05454
116	0.18516	0.05642
117	0.21375	0.05809
118	0.23892	0.05909
119	0.26716	0.05976
120	0.29878	0.06001
121	0.33031	0.05980
122	0.36530	0.05911
123	0.40283	0.05789
124	0.42719	0.05687
125	0.45118	0.05570
126	0.47494	0.05440
127	0.49094	0.05344
128	0.51179	0.05210
129	0.54025	0.05013
130	0.56483	0.04830
131	0.59235	0.04611
132	0.62271	0.04355
133	0.64397	0.04167
134	0.66688	0.03957
135	0.68691	0.03764
136	0.70578	0.03582
137	0.73347	0.03301
138	0.76315	0.02988
139	0.79431	0.02644
140	0.81783	0.02372
141	0.84541	0.02045

Table C.1 Coordinates of NACA 0012 (continued)

142	0.87428	0.01697
143	0.91268	0.01208
144	0.94288	0.00806
145	0.96662	0.00480
146	0.98530	0.00216
147	1.00000	0.00000

C.2 RAE 2822

Table C.2 Coordinates of RAE 2822

NODE	X	Y
1	0.00000	0.00000
2	0.00060	0.00323
3	0.00241	0.00642
4	0.00541	0.00945
5	0.00961	0.01269
6	0.01498	0.01579
7	0.02153	0.01875
8	0.02923	0.02163
9	0.03806	0.02445
10	0.04801	0.02726
11	0.05904	0.03004
12	0.07114	0.03280
13	0.08427	0.03552
14	0.09840	0.03817
15	0.11349	0.04073
16	0.12952	0.04321
17	0.14645	0.04558
18	0.16422	0.04778
19	0.18280	0.04987
20	0.20215	0.05187
21	0.22221	0.05377
22	0.24295	0.05556
23	0.26430	0.05713
24	0.28622	0.05848
25	0.30866	0.05967
26	0.33156	0.06070
27	0.35486	0.06155

Table C.2 Coordinates of RAE 2822 (continued)

28	0.37851	0.06220
29	0.40245	0.06263
30	0.42663	0.06285
31	0.45099	0.06286
32	0.47547	0.06261
33	0.50000	0.06212
34	0.52453	0.06135
35	0.54901	0.06030
36	0.57336	0.05895
37	0.59754	0.05733
38	0.62149	0.05547
39	0.64514	0.05339
40	0.66845	0.05112
41	0.69134	0.04857
42	0.71378	0.04612
43	0.73570	0.04338
44	0.75705	0.04075
45	0.77778	0.03795
46	0.79785	0.03514
47	0.81720	0.03231
48	0.83578	0.02948
49	0.85355	0.02670
50	0.87048	0.02397
51	0.88651	0.02131
52	0.90160	0.01874
53	0.91574	0.01627
54	0.92886	0.01393
55	0.94096	0.01170
56	0.95200	0.00964
57	0.96194	0.00775
58	0.97077	0.00606
59	0.97847	0.00455
60	0.98502	0.00326
61	0.99039	0.00218
62	0.99459	0.00132
63	0.99759	0.00069
64	0.99940	0.00030
65	1.00000	0.00000
66	0.99940	-0.00001
67	0.99759	0.00009
68	0.99459	0.00026

Table C.2 Coordinates of RAE 2822 (continued)

69	0.99039	0.00048
70	0.98502	0.00071
71	0.97847	0.00094
72	0.97077	0.00113
73	0.96194	0.00125
74	0.95200	0.00125
75	0.94096	0.00113
76	0.92886	0.00081
77	0.91574	0.00027
78	0.90160	-0.00049
79	0.88651	-0.00149
80	0.87048	-0.00273
81	0.85355	-0.00422
82	0.83578	-0.00594
83	0.81720	-0.00792
84	0.79785	-0.01013
85	0.77778	-0.01256
86	0.75705	-0.01524
87	0.73570	-0.01812
88	0.71378	-0.02118
89	0.69134	-0.02438
90	0.66845	-0.02770
91	0.64514	-0.03110
92	0.62149	-0.03463
93	0.59754	-0.03791
94	0.57336	-0.04127
95	0.54901	-0.04452
96	0.52453	-0.04761
97	0.50000	-0.05044
98	0.47547	-0.05297
99	0.45099	-0.05515
100	0.42663	-0.05689
101	0.40245	-0.05817
102	0.37851	-0.05893
103	0.35486	-0.05919
104	0.33156	-0.05900
105	0.30866	-0.05843
106	0.28622	-0.05753
107	0.26430	-0.05638
108	0.24295	-0.05498
109	0.22221	-0.05340

Table C.2 Coordinates of RAE 2822 (continued)

110	0.20215	-0.05167
111	0.18280	-0.04977
112	0.16422	-0.04775
113	0.14645	-0.04561
114	0.12952	-0.04333
115	0.11349	-0.04094
116	0.09840	-0.03844
117	0.08427	-0.03584
118	0.07114	-0.03315
119	0.05904	-0.03042
120	0.04801	-0.02761
121	0.03806	-0.02472
122	0.02923	-0.02180
123	0.02153	-0.01880
124	0.01498	-0.01580
125	0.00961	-0.01273
126	0.00541	-0.00957
127	0.00241	-0.00658
128	0.00060	-0.00317
129	0.00000	0.00000

C.3 30P30N

C.3.1 Coordinates of Main Body

Table C.3 Coordinates of main element of 30P30N

NODE	X	Y
1	0.72270	0.05620
2	0.72270	0.04620
3	0.72270	0.03620
4	0.72270	0.02620
5	0.72270	0.01620
6	0.72270	0.00620
7	0.72270	0.00060
8	0.71370	-0.00080
9	0.69440	-0.00460

Table C.3 Coordinates of main element of 30P30N (continued)

10	0.68050	-0.00700
11	0.66700	-0.00930
12	0.65390	-0.01160
13	0.64020	-0.01420
14	0.62580	-0.01650
15	0.61130	-0.01910
16	0.59560	-0.02140
17	0.57940	-0.02370
18	0.56080	-0.02630
19	0.54090	-0.02890
20	0.51930	-0.03090
21	0.47890	-0.03510
22	0.44230	-0.03760
23	0.41550	-0.03870
24	0.37110	-0.03960
25	0.30900	-0.03970
26	0.25970	-0.03840
27	0.23060	-0.03740
28	0.20740	-0.03610
29	0.18750	-0.03480
30	0.17080	-0.03400
31	0.15500	-0.03250
32	0.12800	-0.02990
33	0.10330	-0.02760
34	0.09220	-0.02650
35	0.08290	-0.02540
36	0.07280	-0.02420
37	0.06330	-0.02280
38	0.05450	-0.02150
39	0.04650	-0.02040
40	0.03980	-0.01940
41	0.03220	-0.01830
42	0.02550	-0.01690
43	0.01960	-0.01590
44	0.01520	-0.01500
45	0.01060	-0.01380
46	0.00670	-0.01230
47	0.00390	-0.01010
48	0.00190	-0.00720
49	0.00040	-0.00320
50	0.00000	0.00000

Table C.3 Coordinates of main element of 30P30N (continued)

51	0.00070	0.00510
52	0.00230	0.00980
53	0.00550	0.01520
54	0.00900	0.01930
55	0.01320	0.02360
56	0.01740	0.02710
57	0.02120	0.02980
58	0.02440	0.03200
59	0.02710	0.03370
60	0.03060	0.03580
61	0.03360	0.03760
62	0.04170	0.04170
63	0.05100	0.04650
64	0.06020	0.05020
65	0.07080	0.05380
66	0.08110	0.05730
67	0.09490	0.06090
68	0.10820	0.06400
69	0.12630	0.06760
70	0.14670	0.07010
71	0.18040	0.07290
72	0.21540	0.07520
73	0.25190	0.07750
74	0.27480	0.07900
75	0.30300	0.08000
76	0.39250	0.08250
77	0.41250	0.08250
78	0.43250	0.08250
79	0.45250	0.08250
80	0.47250	0.08250
81	0.49250	0.08250
82	0.51060	0.08250
83	0.56240	0.08120
84	0.59670	0.07940
85	0.62530	0.07880
86	0.64810	0.07740
87	0.66850	0.07620
88	0.68810	0.07530
89	0.70610	0.07390
90	0.72130	0.07270
91	0.73660	0.07140

Table C.3 Coordinates of main element of 30P30N (continued)

92	0.75060	0.07020
93	0.76290	0.06910
94	0.77630	0.06780
95	0.78740	0.06650
96	0.79940	0.06510
97	0.81250	0.06360
98	0.82530	0.06230
99	0.84010	0.06030
100	0.84960	0.05930
101	0.85870	0.05770
102	0.86820	0.05620
103	0.83820	0.05620
104	0.80820	0.05620
105	0.75820	0.05620
106	0.72270	0.05620

C.3.2 Coordinates of Slat

Table C.4 Coordinates of slat of 30P30N

NODE	X	Y
1	-0.10130	-0.06720
2	-0.10250	-0.07260
3	-0.10340	-0.07780
4	-0.10350	-0.08460
5	-0.10310	-0.08670
6	-0.10170	-0.09190
7	-0.10030	-0.09540
8	-0.09940	-0.09800
9	-0.09810	-0.09920
10	-0.09370	-0.10360
11	-0.09840	-0.10450
12	-0.10430	-0.10590
13	-0.11080	-0.10720
14	-0.11280	-0.10750
15	-0.12270	-0.10780
16	-0.12940	-0.10750
17	-0.13290	-0.10660

Table C.4 Coordinates of slat of 30P30N (continued)

18	-0.13650	-0.10540
19	-0.13950	-0.10300
20	-0.14120	-0.10010
21	-0.14260	-0.09810
22	-0.14300	-0.09630
23	-0.14320	-0.09330
24	-0.14260	-0.08810
25	-0.14010	-0.08200
26	-0.13650	-0.07630
27	-0.13150	-0.06920
28	-0.12880	-0.06550
29	-0.12500	-0.06130
30	-0.12180	-0.05770
31	-0.11920	-0.05460
32	-0.11340	-0.04850
33	-0.10600	-0.04190
34	-0.10180	-0.03770
35	-0.09700	-0.03370
36	-0.09240	-0.02980
37	-0.08720	-0.02550
38	-0.08210	-0.02120
39	-0.07770	-0.01760
40	-0.07280	-0.01360
41	-0.06740	-0.00900
42	-0.07040	-0.01280
43	-0.07410	-0.01720
44	-0.07770	-0.02190
45	-0.08130	-0.02670
46	-0.08380	-0.03020
47	-0.08640	-0.03450
48	-0.08880	-0.03880
49	-0.09110	-0.04290
50	-0.09240	-0.04530
51	-0.09370	-0.04770
52	-0.09490	-0.05000
53	-0.09610	-0.05260
54	-0.09730	-0.05530
55	-0.09870	-0.05940
56	-0.10000	-0.06290
57	-0.10130	-0.06720

C.3.3 Coordinates of Flap

Table C.5 Coordinates of flap of 30P30N

NODE	X	Y
1	1.01170	0.01320
2	1.01520	0.01130
3	1.01870	0.00910
4	1.02140	0.00780
5	1.02490	0.00560
6	1.03080	0.00220
7	1.03680	-0.00150
8	1.04180	-0.00470
9	1.04660	-0.00800
10	1.05090	-0.01080
11	1.05680	-0.01500
12	1.06280	-0.01920
13	1.06820	-0.02330
14	1.07500	-0.02790
15	1.07950	-0.03180
16	1.08490	-0.03590
17	1.09050	-0.04030
18	1.09660	-0.04510
19	1.10280	-0.05000
20	1.11030	-0.05590
21	1.11710	-0.06170
22	1.12400	-0.06720
23	1.12970	-0.07190
24	1.13750	-0.07870
25	1.14260	-0.08360
26	1.14810	-0.08850
27	1.15540	-0.09500
28	1.16050	-0.09980
29	1.16650	-0.10590
30	1.17080	-0.11000
31	1.17610	-0.11530
32	1.18160	-0.12060
33	1.18630	-0.12540
34	1.19090	-0.12990
35	1.19590	-0.13530
36	1.19000	-0.13080

Table C.5 Coordinates of flap of 30P30N (continued)

37	1.18500	-0.12690
38	1.17980	-0.12300
39	1.16040	-0.10880
40	1.15270	-0.10280
41	1.14360	-0.09620
42	1.13400	-0.08980
43	1.12520	-0.08410
44	1.10930	-0.07440
45	1.09680	-0.06730
46	1.08400	-0.06050
47	1.07290	-0.05480
48	1.05910	-0.04790
49	1.04360	-0.04090
50	1.02530	-0.03310
51	1.00690	-0.02560
52	0.98020	-0.01560
53	0.96800	-0.01110
54	0.95330	-0.00630
55	0.94460	-0.00300
56	0.93890	-0.00050
57	0.93240	0.00330
58	0.92810	0.00740
59	0.92490	0.01140
60	0.92170	0.01590
61	0.92010	0.02050
62	0.92010	0.02510
63	0.92140	0.02910
64	0.92370	0.03270
65	0.92670	0.03500
66	0.93040	0.03690
67	0.93430	0.03770
68	0.93940	0.03840
69	0.94530	0.03790
70	0.94980	0.03730
71	0.95540	0.03620
72	0.96110	0.03500
73	0.96750	0.03270
74	0.97440	0.03030
75	0.98010	0.02810
76	0.98480	0.02620
77	0.99340	0.02250

Table C.5 Coordinates of flap of 30P30N (continued)

78	0.99800	0.02020
79	1.00320	0.01760
80	1.00810	0.01510
81	1.01170	0.01320