DEVELOPMENT OF AN UNSTRUCTURED CARTESIAN FLOW SOLVER

Von der Fakultät für Maschinenwesen der Rheinisch-Wesfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Khalid Mohammad Sultan
aus Benghazi, Libyen

Berichter: Universitätsprofessor Dr.-Ing. Wolfgang Schröder
Universitätsprofessor Dr.-Ing. Dieter Hänel

Tag der münlichen Prüfung: 02.12.2005

Diese Dissertation ist auf der Internetseiten der Hochschulbibliothek online verfügbar

Acknowledgements

This thesis was written during my work as a scholarship holder at the Chair of Fluid Mechanics and Institute of Aerodynamics AIA at the RWTH -AACHEN University, in Germany.

First, I would like to express my appreciation and gratitude to my advisor Professor Dr.-Ing. Wolfgang Schröder for his guidance, proposals, and advice. I would like also to thank Professor Dr.-Ing. Dieter Hänel and Professor Dr.-Ing. Herbert Olivier for accepting to be in the examination committee.

I would like also to thank Dr.-Ing. Matthias Meinke and Dr.-Ing. Andreas Henze for the comments during the development stage of this work. Special sincere gratitude and thanks due to my former colleagues Dr.-Ing. Ehab Fares and Dipl.-Ing. Jens Krömer for the scientific support and the continuous encouragement.

Last but not least, I would like to thank my wife and my children for their continuous support, endless patience, and for providing the comfort, warmness, and the suitable environment at home.

Aachen, December 2005 *Khalid M. Sultan*

Contents

Abstract

An anisotropic unstructured cartesian grid generator and a flow solver were developed for the compressible viscous flow. The grid generator assumes a body configuration given as a set of data points and generates automatically a computational cartesian grid with geometry adaptation. The cartesian flow solver employs an upwind high resolution shock capturing scheme with linear reconstruction to achieve globally second order accuracy. In order to render the flow solver capable of solving supersonic flows, a limiter function was incorporated in the reconstruction algorithm. A solution -based grid adaptation algorithm that employs the discrete wavelet transform was built in the developed flow solver and tested for several standard and non-standard test cases.

Übersicht

Ein anisotroper unstrukturierter kartesischer Gittergenerator und Strömungslöser sind zur Lösung kompressibler reibungsbehafteter Strömungen entwikelt worden. Der Basis für den Gittergenerator wird der Körper mit einer Reihe von Datenpunkten beschrieben, wobei nach dem Einlesen der Datenpunkte automatisch ein kartesisches Gitter mit geometrischen Gitteradaptierung generiert wird. Der kartesische Strömungslöser verwendet ein hochauflösendes Upwind-Schema mit linearer Rekonstruktion der Variablen, um eine allgemeine Genauigkeit von zweiter Ordnung zu erhalten. Ein Limiter ist implementiert worden, so dass der Strömungslöser auch zur Berechenung von überschall Strömungen herausgezogen werden kann. Die Gitteradaption während der Lösung wird mit Hilfe von der diskreten Wavelet-Transformation ausgeführt. Zur Validierung des Gittergenerators und des Strömungslöser werden mehre Testfälle berechnet.

Nomenclature

Latin Letters

$\overline{\overline{H}}$	flux vector .
$\tilde{a}_{L/R}$	split speed of sound.
$\vec{E}$, $\vec{F}$, and $\vec{G}$	flux vectors in x, y, and z directions,respectively.
$\vec{L}(s)$	parametric representation of a line in space.
$\vec{P}(r,t)$	parametric representation of a plane.
$\vec{Q}$	vector of the conservative variables.
$\vec{q}$	heat flux vector.
$\vec{v}$	velocity vector.
$\vec{W}$	vector of the primitive variables.
$\vec{dr}$	position vector.
$\widehat{CFL}$	temporary CFL defined by equation (3.32).
A	surface area of the control volume.
$a_{1/2}$	numerical speed of sound.
$a_{i,i=0,\ldots,3}$	data points that compromise the wavelet window.
$a_{L/R}$	left or right speed of sound at the cell face according to the upstream direction.
$B_{i,n}$	BERNSTEIN Polynomials.
C_o, C_-, and C_+	characteristics.
C_p	pressure coefficient.
C_v	specific heat at constant volume.
CFL	maximum COURANT number allowed by the time advancing scheme.
CFL_{cutB}	cut -back CFL defined by equation (3.31).

d_1	distance between center of cell number one and the face, see figure (4.5).
d_2	distance between center of cell number two and the face, see figure (4.5).
D_i	detail of the wavelet transform.
D_p	pressure diffusion term in $AUSM^+ - au$ scheme.
dA	differential element of the area.
e	specific total energy.
f	pirimitive variable to be reconstructed.
$g_{i,i=0,\ldots,3}$	wavelet coefficients.
$h_{i,i=0,\ldots,3}$	wavelet coefficients.
I	identity tensor.
k	thermal conductivity.
Ld	length of the square cavity.
M_L^+	left split MACH number.
M_R^-	right split MACH number.
$M_{(2)}^{\pm}$	second order polynomial of MACH number used in $AUSM^+ - au$ scheme.
$M_{1/2}$	interface MACH number.
$M_{1/2}^{p/m}$	interface split MACH numbers.
$Mean$	statistical mean.
$Median$	statistical median.
n	degree of the BEZIER curve.
n	number of cells contributing in the slope calculation by the least squares method.
n_x, n_y, and n_z	normal vector in x, y, and z directions, respectively.
p	pressure.
$P(t)$	point coordinates as represented by BEZIER curve.
$p^{\pm}$	split pressures used in $AUSM^+ - au$ scheme.
P_1	outlet plane.
$P_1, P_2, P_3, \ldots, P_n$	set of a given data points that represent the body geometry.
p_L^+	left split pressure.

P_o	inlet plane.
p_R^-	right split pressure.
P_t	total pressure.
$p_{1/2}$	pressure part of convective flux.
Pr_o	PRANDTL number based on the stagnation conditions.
R	gas constant.
Re_o	REYNOLDS number based on the stagnation conditions.
Res	residual.
S_i	scale of the wavelet transform.
T	temperature.
t	parameter in the BEZIER curve.
t	time.
$Threshold$	wavelet threshold.
u, v, and w	velocity components in x, y, and z directions, respectively.
$u_{1/2}$	interface velocity.
u_{x1rc}	reconstructed slope of u_{x1} at the face between cell number one and cell number two.
u_{x1}	slope of the x -component of the velocity vector at the center of cell number one.
u_{x2rc}	reconstructed slope of u_{x2} at the face between cell number one and cell number two.
u_{x2}	slope of the x -component of the velocity vector at the center of cell number two.
Vd	lid velocity of the cavity.

Greek Letters

α	angle of attack.
α_1,α_2, $\ldots$ α_5	coefficients of the RUNGE -KUTTA scheme.
$\chi_{i,i=1,\ldots,5}$	sensors of solution -based grid adaptation.
Δx	length of the cell in x -direction.
Δy	length of the cell in y -direction.

Δz	length of the cell in z -direction.
Δ^+	positively -biased difference in primitive variables of two neighbour cells.
Δ^-	negatively -biased difference in primitive variables of two neighbour cells.
η	composite dimensionless variable introduced by BLASIUS.
γ_o	specific heats ratio.
λ_i	scalar weights of rational BEZIER curves.
ν	kinematic viscosity.
ϕ	limiter function.
ψ	factor that is used in equation (4.22).
ψ_{abcd}	signed volume of tetrahedron *abcd*.
ψ_{abce}	signed volume of tetrahedron *abce*.
ρ	fluid density.
$\rho_{1/2}$	interface density in $AUSM^+ - au$ scheme.
σ	standard deviation.
τ	viscous shear stress.

Subscripts

∞	conditions in the undisturbed flow in the far field.
bc	The state in a boundary cell.
cc	value of the variable at the cell center.
cd	state in the computational domain.
e	neighbour cell in the eastern direction to the face of a cell.
fc	state in a flow cell.
inv	inviscid part of the flux.
L/R	left or right value at the cell face according to the upstream direction.
vis	viscous part of the flux.
ww	western neighbour of the neighbour cell in the western direction to the face of a cell.
w	neighbour cell in the western direction to the face of a cell.

Superscripts

k	number of stages of the RUNGE -KUTTA scheme.
n	previous time level.
$n+1$	next time level.

Acronyms

ADT	Alternating Digital Tree.
AMR	Adaptive Mesh Refinement.
AUSM	Advection Upstream Splitting Method.
CAD	Computer Aided Design.
CFD	Computational Fluid Dynamics.
CFL	COURANT number.
CWT	Continous Wavelet Transform.
DWT	Discrete Wavelet Transform.
FBI	Federal Bureau of Investigation.
LSM	Least Squares Method.
NURBS	Non-Uniform Rational B-Splines.
STL	Standard Template Library.
WSQ	Wavelet Scalar Quantization.

Chapter 1

Introduction

"A scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die and a new generation grows up that is familiar with it."

Maxwell Planck

The growing application fields of the computational fluid dynamics especially in biological fluid dynamics has demanded algorithms that can overcome the challenges associated with these applications; such as the complexity of the geometries[1] encountered. The analysis of flow patterns around heart valves Peskin [53],[55] and Peskin and Printz [56] is a typical example of such applications. In Addition to the capability of dealing with complex geometries, other aspects are considered decisive in evaluating any flow analyzing algorithm; among those are the computing efficiency, ability of automation, and memory efficiency. Although methods based on body-fitted grids have had reasonable sucess in their application to real-world problems, no method has offered a truly automatic method for grid generation of arbitrarily complex geometries. Moreover the computing and memory overhead due to the metric terms and the loss of accuracy associated with them[2] indicate a better performance when using a cartesian grid. Since the first attempt of Peskin [53] to employ a cartesian grid in order to analyze the flow field of the blood through heart valves, no attempt has been carried out to employ this approach in aerodynamics until the late 80's when Clarke et al. [11] had solved the Euler equations for

[1] Which are often deforming in time.
[2] Due to the truncation error.

multi-element airofoils. This was followed by a vast amount of research papers (e.g. Gaffney et al. [27], Berger and Colella [8], Quirk [58]) that were devoted to the cartesian approach.

Recently, some of the authors have shifted the attention to extending the existing inviscid cartesian approaches (especially the **A**daptive **M**esh **R**efinement algorithm of Berger [8]) to viscous flow simulations, see e.g. Coirier [12], Karman [36], and Kalitzin and Iaccarino [35].

Body -fitted grids, either structured or unstructured, have gained their popularity through the ease of implementing the boundary conditions while this is the main challenge when implementing a cartesian grid. The cells that constitute the cartesian grid can extend through the surfaces of the boundary and hence make the introduction of the boundary conditions a relatively difficult task. In contrast to the issue of the boundary conditions, the grid generation process is a time and computational resources consuming process in the case of body-fitted grids. An overview is given by Eiseman [18, 19] of the *Partial Differential Equations* -based techniques and the *Algebraic Method* used in structured -body fitted grid generation. The unstructured -body fitted grid generation is achieved by either destructing an existing structured grid or by the *Advancing Front Method*, for more details see Löhner and Parikh [46]. It is worth noting that the difficulty of the grid generation process increases as the geometry gets more complicated. The cartesian grid does not suffer from this symptom, as the need for a body-fitted boundary is eliminated.

The community of the cartesian grid approach classifies this in general as either structured or unstructured. This classification occurs based on the way in which the data are stored and the mechanism of which the inter-connectivity between each cell and its neighbours is applied. In the structured cartesian grid, which was pioneered by Berger (see the **A**daptive **M**esh **R**efinement work of Berger and Oliger [9], Berger and Colella [8]) and pursued later by other authors (e.g. Quirk [58], and Pember et al. [51]), subgrid -structured patches with local indices are imposed on the base grid in the places requiring high resolution. The connectivity within the base grid and within those subgrids is achieved through a local indexing system. This approach has proven to lead to an excessive number of cells since about 70% of the cells in a subgrid are actually tagged for refinement, this implies an overhead of about 30% Aftosmis [2].

The unstructured cartesian grid is characterized through the tree-based data structures; that is, the grid connectivity information is implied by a tree's logical hierarchical structure (see Finkel and Bentley [22] and Samet [64]). This mechanism for data structure was first employed in the cartesian grid by De zeeuw and Powell [16]. In this mechanism each cell has a pointer to its parent (if this cell is produced

through a refinement process) and to its four children[3] (if it is refined). The Quad-tree and the Oct-tree data structures have suffered from being restricted to isotropic grid refinement, this has led again to a memory and computing time overhead since in viscous fluids - which are at high REYNOLDS numbers characterized by a thin boundary layer - the need for directional refinement is very essential. To overcome this problem WANG et al. [69] and LAHUR and NAKAMURA [37] had implemented a more flexible-tree data structure; namely, the 2^N tree and also called the *Omni*-tree data structure. The saving in the number of cells with the 2^N tree compared to the Oct-tree is over a factor of 6 as investigated by WANG et al. [69]. Both approaches (Oct-tree and Omni-tree) have the problem of computing -time losses in the search process of neighbouring cells. For instance, finding a neighbour cell requires that the tree must be traversed all the way to its root when a refined cell in the farthest level down the hierarchy is neighboured by another one that belongs to the base grid. Although AFTOSMIS [2] has recommended the use of a special search algorithm (e.g. The **A**lternating **D**igital **T**ree), this can only reduce the search time, but does not eliminate it completely. In the author's opinion, this time loss can be eliminated if the whole information concerning connectivity is stored. In the present work all of the connectivity information that is needed is explicitly stored.

The term *cartesian grid method* denotes the special way with which the *irregular*-boundary cells are treated. In general, all cartesian flow solvers employ the *immersed boundary method*[4]; that is, a standard integration scheme is applied on the regular cells in the grid and a special treatment is given to the small -cut cells. In order to distiguish the small -cut cells from other cells, they will be referred to as *silver cells* in the rest of this work.

Glancing at the treatment of the silver cells, one finds three different approaches in the literature, Those are:

- The grid aligned h-box method, that is also known as *the rotated difference scheme*, see JAMESON [32]
- The flux -redistribution procedure of PEMBER et al. [51].
- The cell merging technique introduced by QUIRK [58].

All three approaches were developed with two main targets in mind: Avoiding stability problems due to small boundary cells, and achieving a high order of accuracy at the boundaries. Although the stability problem was overcome, none of the above mentioned schemes had reached a second order accuracy near the boundary, see FORRER [24].

[3] This is in 2-D, and hence the name **quad-tree**.
In 3-D there are eight children, and it is known as **oct-tree**.

[4] Alternatively, the term *embbeded* is used, because a uniform rectangular computational mesh is maintained everywhere in the computational domain and the geometric description is treated as a specialized boundary embbeded in the mesh.

In general, these approaches tried to integrate the silver cell itself (in a special way) as in the first and second approaches or by merging it in a neighbouring silver or normal cell as in the third approach[5]. It should be noted that in this work another technique is used to treat the silver cells. The boundary conditions are imposed on those cells. This has shown a very impressive effect on the execution time for the test cases encountered and yields remarkable and simple implementation details.

The present work was intended to form a nucleus for a general three dimensional unsteady compressible NAVIER-STOKES cartesian flow solver for educational purposes and to provide an additional research tool. Because of time constraints and resource limitations, the work was restricted to two dimensional problems. Nevertheless, the grid generation method will be also presented for the case of three dimensional geometries. Fortunately, the extension of the flow solver - due to the principle of *dimensional splitting* - to three dimensions is straightforward and could be easily built into the existing code.

The rest of this thesis text will proceed as follows: Chapter (2) will explain the data structure and the grid generation process. In chapter (3) several items comprising the mathematical model such as the governing equations, the boundary conditions, and the initial conditions are considered. In chapter (4) the details of the method of solution will be revealed. The standard test cases of the AGARD working group 07 in addition to the flow on the BAC3 aerofoil will be used to validate the developed flow solver for the solution of the EULER equations. To validate the developed flow solver for the solution of the NAVIER-STOKES both the flow on a flat plate at zero incidence and the lid -driven cavity flow will be used. These test cases will be presented in chapter (5). Finally, in chapter (6) a conclusion and some recommendations for future work will be presented.

[5]In the first approach also for the very small cells.

Chapter 2

Data Structure and Grid Generation

"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg."

Bjarne Stroustrup

Normally, the process of grid generation is considered as a background work that is excluded from the presentation of the flow solver formulation. This is true for the structured and unstructured -body fitted grids, since the grid generation process in these types of flow solvers is well established. That is discussed in almost all recent literature concerning these types of grids; for instance, see Fares et al. [20] and Frink et al. [26]. On the other hand, the situation is different in the case of flow solvers that use a cartesian grid, since the major parts that differ are the grid generation and the boundary conditions, the latter depend on the geometrical details of the silver cells[1]. Moreover, the cartesian grid flow solvers do adapt the grid in the course of the solution process in order to obtain high resolution in high gradient zones. Having mentioned this and in order to completely explain the grid generation process, one needs to specify the required information to be stored for each computational cell. The next section explains several items of the data structure employed in this work.

[1]The way the silver cell is cut and the coordinates of the cut points.

2.1 Data Structure

The cartesian grid generator that was developed in this work belongs to the unstructured -cartesian grids family with anisotropic refinement capability. Nevertheless, no tree -like data structure was employed. Instead, all of the connectivity information - that is needed by each cell - is explicitly stored. This has the advantage of eliminating the time loss in the search process. Recalling the *encapsulation* principle from the field of the *C++ programming language*, each cell is treated as an isolated unit (an *object* of the *Class Cell*) that contains the necessary information.

During the development stage of the work, it was very clear that a temporary data structure is needed for reasons of speed and simplicity; that is, some operations that take place during the solution do occur at the cells faces[2]; namely, the computations of the numerical fluxes and the variables' reconstruction process. The process of faces (or surfaces) generation is essential for the solution process and has nothing to do with the grid generation process. Once the initial grid is generated and all the cells that compromise the initial grid are classified to flow, body, and silver cells, the surfaces are generated between all the superfacial cells[3].

Owing to the fact that the *containers* included in the *STL*[4] have a memory overhead that is associated with the excessive pointers employed by the containers themselves in the house -keeping processes, see JOSUTTIS [34], they were excluded from consideration. All of the generated cells are stored in a one dimensional array, another one dimensional array is used to store all of the corresponding generated surfaces. The following subsections enable a closer look at the elements of both the cell class and the surface class.

Cell -Based Data Structure

- An integer is required to store the parent cell identity number (*int parent*), this will enable the children cells of a refined cell to inherit the information from the parent.
- An integer is used to store the number of children *int childrenNum* emerging from the refinement process, see figure (2.1).

[2] In this work sometimes it is also called surfaces
[3] Those cells that have no children.
[4] This refers to the **S**tandard **T**emplate **L**ibrary available in all C++ compilers

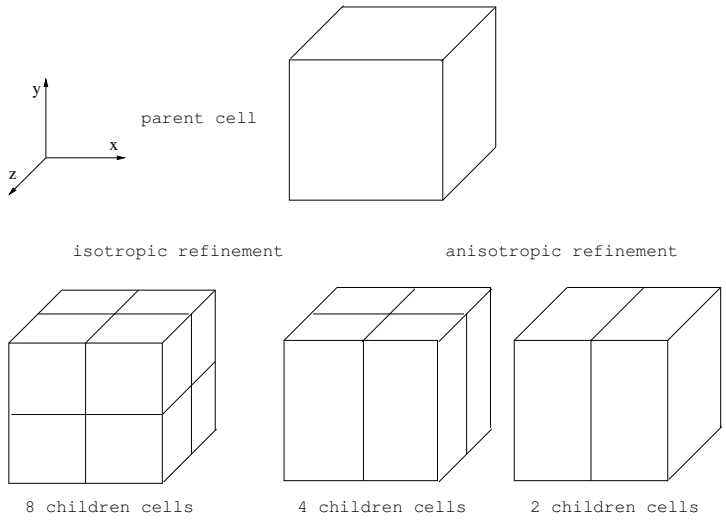

Figure 2.1: Isotropic and anisotropic refinement.

- A pointer to the expected children *int* children* [5] that can be freely resized to hold the identity numbers (integers) of the children cells if the cell is refined. The dynamic memory allocation that is provided by the *C++ programming language* through the use of the pointers is a great advantage here since it enables the sizing of the pointer children according to the refinement type it has undergone and hence it enables the anisotropic refinement without any unnecessary memory usage.
- An array of four pointers *int* neighbourIDs [SpaceDim*2]* to store the identity numbers of the neighbour(s) on each side of the four sides of a cell[6].
- Four integers for the number of neighbours on each side of the cell are stored *int neighbourNum [2*SpaceDim]*.
- Two integers for the cell vertex *int cellVertex [SpaceDim]* [7]. It is stored as an integer to avoid the floating point operations in the search process.
- One more information is needed to complete the connectivity information; namely, the directional level *int level[SpaceDim]*. This is an integer for each direction which tells the cell about its status and history.

Some other information is needed for the flow solver to accomplish its mission. This information consists of:

1. An integer *int cellType* that is needed to classify the cells into either flow, body, or silver cells.
2. A field of integers *int cutposition [SpaceDim+1]* to specify locations at which a silver cell is cut.

[5] This is only a pointer (an integer)

[6] In three dimensional problems it will be six directions, thats why it is written as a function of the number of space dimensions.

[7] This is the left bottom corner of the cell in 2-D and the left bottom back corner in 3-D.

3. A field of doubles *double cutPointCrdnt [SpaceDim+1]* that is needed to specify the coordinates of the cut points in a silver cell.

Figure (2.2) shows the definition of the *Class Cell* which was employed in the code developed by the author. As seen in this figure some other information[8] is included

```
Class Cell {
        int parent;
        int childrenNum;
        int* children;
        int neighbourNum[SpaceDim*2];
        int* neighbourIDs[SpaceDim*2];
        int cellVertex[SpaceDim];
        int level[SpaceDim];
        int cellType;
        int cutPosition[SpaceDim+1];
        double cutPointCrdnt[SpaceDim+1];
        double lengthOfCell[SpaceDim];
        double cellCoordinate[SpaceDim];
        double oldCnsrvtvVrbl[SpaceDim+2];
        double cnsrvtvVrbl[SpaceDim+2];
        double slope[SpaceDim][SpaceDim+2];
        double rghtHndSd[SpaceDim+2];
        void sizeChldrn(int childrenNum);
        void sizeNghbr(int direction, int NghbrNum);
}
```

Figure 2.2: The member variables and member functions of the class cell

such as the length of cell *double lengthOfCell [SpaceDim]*, the cell Coordinate *double cellCoordinate [SpaceDim]*, the conservative variables of the previous time step *double oldCnsrvtvVrbl [SpaceDim+2]*, the conservative variables of the next time step *double cnsrvtvVrbl [SpaceDim+2]*, the directional slopes of the reconstructed variables *double slope [SpaceDim] [SpaceDim+2]*, and the right -hand side *double rghtHndSd [SpaceDim+2]* that will appear in the time -stepping scheme.

The last two items of the class cell are the two member functions that are used

[8]These are needed for two duties: In the process of classifying the cells to flow, body, or silver cells, and during the solution stage.

to assign the suitable array size of the children pointer and the neighbour pointer. Since the anisotropic refinement allows the production of two or four -children cells in two dimensional problems and allows two, four, or eight children cells in three dimensional problems; see figure (2.2), the size of the array children can only be determined if the type of refinement is decided. The same applies for neighour cells on each side of the parent cell; that is, if the parent cell has on a side a number of neighbours k, a new children cell will have on that side the number of neighbour cells that are overlapping it. After searching for those neighbour cells and identifying them, the array neighbour is sized just to hold the exact number of neighbours.

Surface -Based Data Structure

As mentioned in the last subsection, during the early stages of the computer program development process it was clear, that the execution speed could be increased if another surface -based data structure is generated. Computing the numerical fluxes is the main process that occurs virtually at the control volume boundaries (the cell surfaces). In order to compute these by an upwind scheme, section (4.2), the primitive variables at the left and the right sides of the cell surfaces must be interpolated. This process is referred to as the *variables' reconstruction* process, section (4.1). Both processes are performed using the surface -based data structure. In figure (2.3), the member variables of this class are shown.

```
Class Surface {

        int idOfPartners[2];

        double rcnstrctdVrbls[2][SpaceDim+2];

        double flux[SpaceDim+2];

}
```

Figure 2.3: The member variables of the class surface

Each surface object has an array of two integers *int idOfPartners [2]* in which the identity numbers of the two partner cells are stored. Later, this will enable the algorithm that computes the numerical fluxes from accessing the cells that share that surface. Since the RIEMANN solver needs the left and right values of the variables, they are computed using both the values at cell centers and the slopes and then stored in the array *double rcnstrctdVrbls [2] [SpaceDim+2]*. As might be noticed the

first dimension is two referring to the left and right values of the variables to the face. Lastly, an array *double flux [SpaceDim+2]* is included to hold the numerical fluxes that are needed for the time integration scheme.

2.2 Grid Generation

The geometrical details of the solid body can be practically specified via an arbitrary number of BEZIER curves, *CAD*[9] data, *NURBS*[10] curves or surfaces, or a set of data points. Once the geometrical description of the body is given, it is plugged into the grid generation algorithm. It is expected that the grid generator will perform a search on the cells of the initial grid to identify those cells that are located outside, inside, or cut by the body. Moreover, the detailed information of the cut cells such as the cut positions are very essential to the successful implementation of the boundary conditions.

In the present work it was assumed that the geometrical description of the given body is given as a set of data points; that is, $P = \{P_1, P_2, \ldots, P_n\}$. The direction in which this data is given must be specified in order to locate the sides that are outside and inside the body, see figure (2.4).

[9] **C**omputer **A**ided **D**esign software
[10] **N**on **U**niform **R**ational **B-S**plines

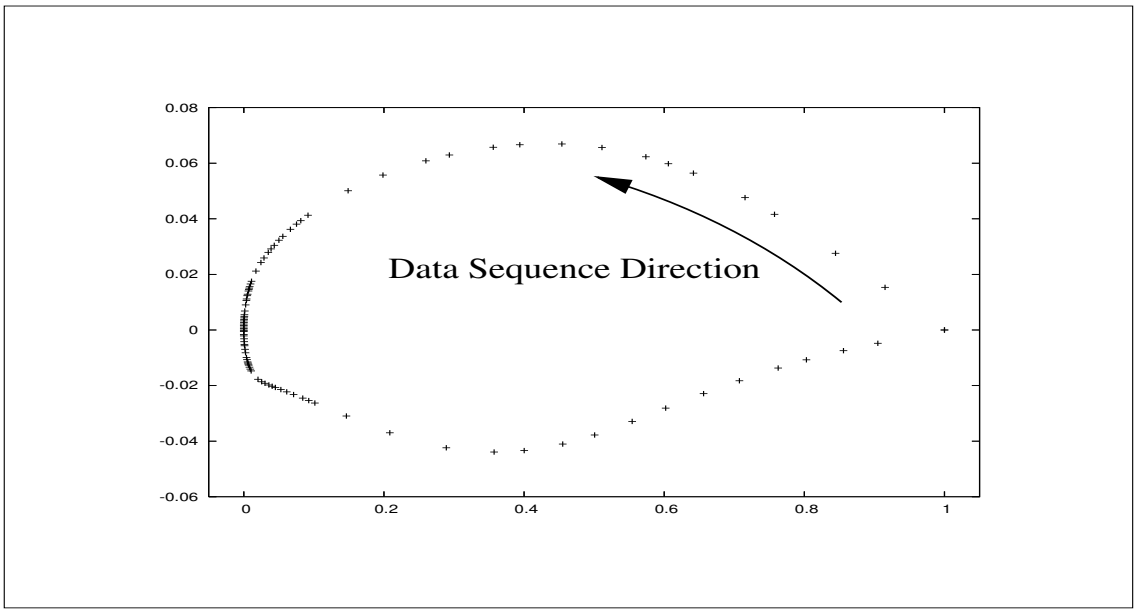

Figure 2.4: Geometrical description of a given body.

Grid Zooming Process

In order to have a grid suitable for capturing the geometrical details of the given body a relatively fine mesh is required. A fine initial grid will produce a memory overhead and will affect the execution speed of the computer program. This problem was overcome by introducing a *zooming* algorithm; that is, after generating a relatively coarse grid, all cells that are located inside a circle of a diameter that is about 3/2 times greater than the largest dimension of the body are refined recursively until the smallest dimension of any cell in this circle is less than 1/2 of the smallest dimension of the body. This has proven to be an efficient procedure in providing a relatively fine mesh; see figure (2.5).

Determination of the Neighbour Cells

New cells that emerge from the refinement process need to be *informed* about their neighbouring cells[11], their position in the grid (i.e. the cell coordinates), the parent cell, etc. Inserting this information in the new emerging cell is a trivial task, however, if the parent cell has more than one neighbour on a side, the situation is completely different and a search algorithm must be designed and performed. To be more precise, a sample case will be introduced here that is shown in figure (2.6). In this

[11]This is important in performing the surface integral of the numerical flux, see equation (3.3)

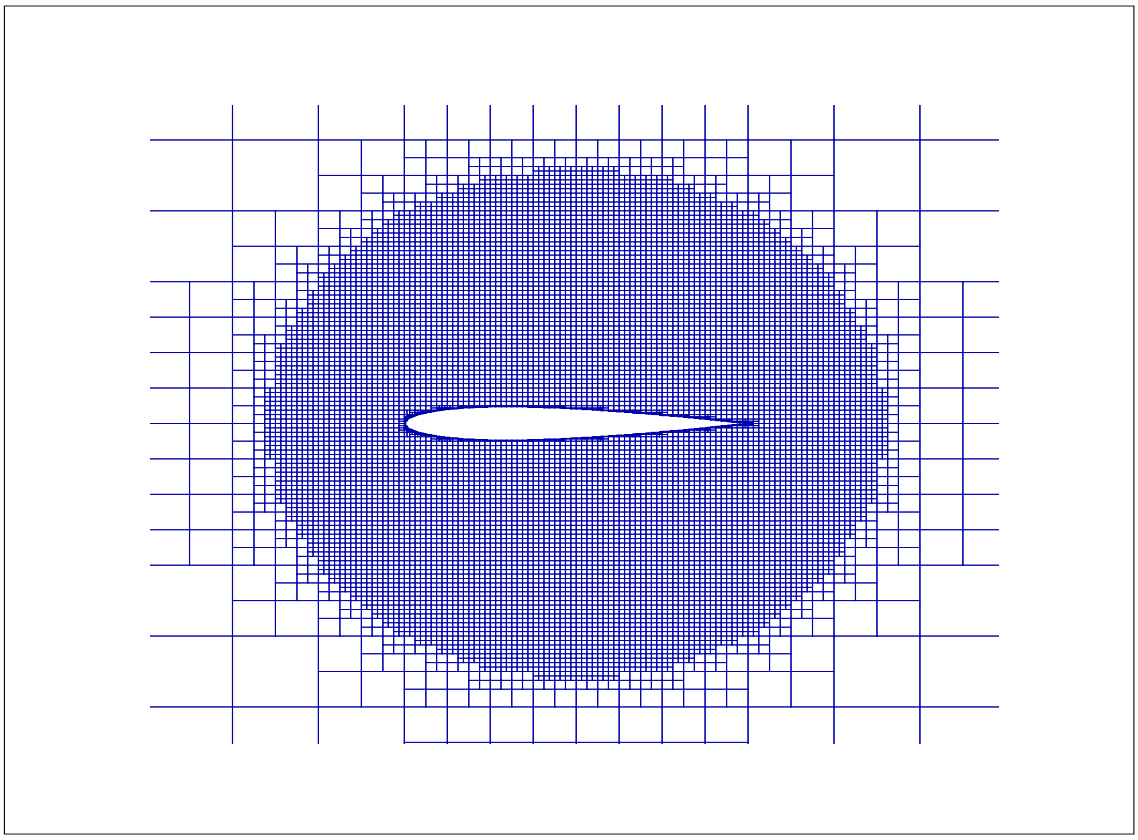

Figure 2.5: Zooming of a specific region to obtain a relatively fine grid.

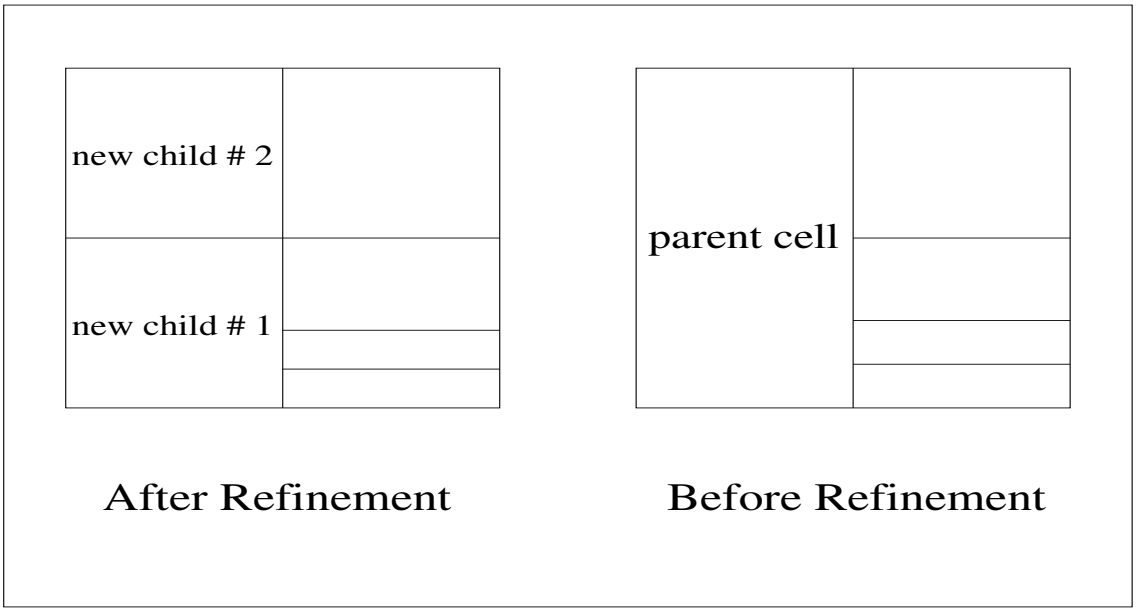

Figure 2.6: Specifying the neighbours for the new emerging cells in the two dimensional case.

figure, it is shown that the parent cell was abutted by four cells on the eastern face. On the left of the figure it is seen that the parent cell has undergone a refinement process that has produced two new children cells. Specifying which neighbour(s) that abutt the new child cell number one is enough to conclude that the rest of neighbours belong to the new child cell number two. This is done by comparing the upper edge of the new child cell number one with the upper edge of each one of the parent's neighbours; if the upper edge of the neighbour is less than that of the new child cell number one, then it is a neighbour. This test is performed on all neighbours until a neighbour is met that has an upper edge that is equal to the upper edge of the new child cell number one. At this neighbour the search is stopped and the rest of the neighbours are assigned to the new child cell number two. The following pseudo-code shows the procedure in detail:

```
for( all neighbours on side E ){
     if( the upper edge of the neighbour <=
         the upper edge of the new child # 1 ){
             this neighbour is a neighbour of the new child # 1;
     }
     else{
         for( all of the remaining neighbours ){
              this neighbour is a neighbour of the new child # 2;
         }
     }
}
```

It should be noted that the distances compared are all stored and computed in an integer form to avoid the problems associated with floating point operations; see [2].

The three dimensional search has a similar algorithm with the only exception that the search is now performed in two space dimensions. For instance - if the face lies on the y-z plane - the upper edge of the neighbour is represented now by its y and z coordinates, see figure (2.7). In order to determine the neighbours for each new child cell, a search procedure is performed that loops over all the neighbours and investigates each one for the overlapping criterion; that is, the new child cell overlaps partially or totally the neighbour. Another pseudo-code is introduced to clarify this procedure:

```
for( all new children )
     for( all neighbours on side W ){
          if( m1 < y2 && n1 < z2 &&
              m2 > y1 && n2 > z1
            ){
                this is a neighbour of the
```

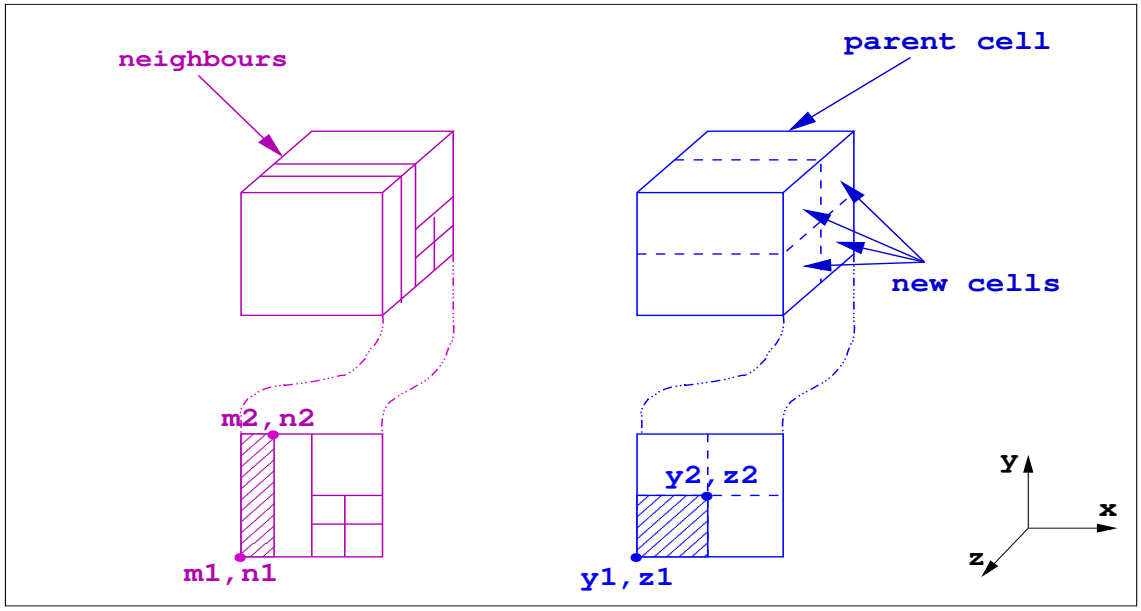

Figure 2.7: Specifying the neighbours for the new emerging cells in the three dimensional case.

```
                instantaneous new child;
            }
        }
}
```

Another similar search algorithm is performed in order to update the neighbours of the parent cell; that is, to assign for each neighbour of the parent cell's neighbours the identity number of the new correct emerging cell.

Cell Classification

After the initial grid has been generated and the data points that are representing the body geometry have been read, the grid cells have to be classified int cells that are flow, body, or cut by the body boundary; namely, the silver cells. There are two methods that are used to perform this classification process [2]. The *winding number* method [23] and the *ray -casting* method [16]. Both of the two methods are applicable to two and three dimensional problems, but owing to the fact that the second method does not involve floating-point computations of many small angles each of which is prone to round -off errors, the second method was chosen to be the tool for classifying the grid cells. As indicated in figure (2.8), a ray that starts at a node in the cartesian grid is assumed to spread in any direction towards the

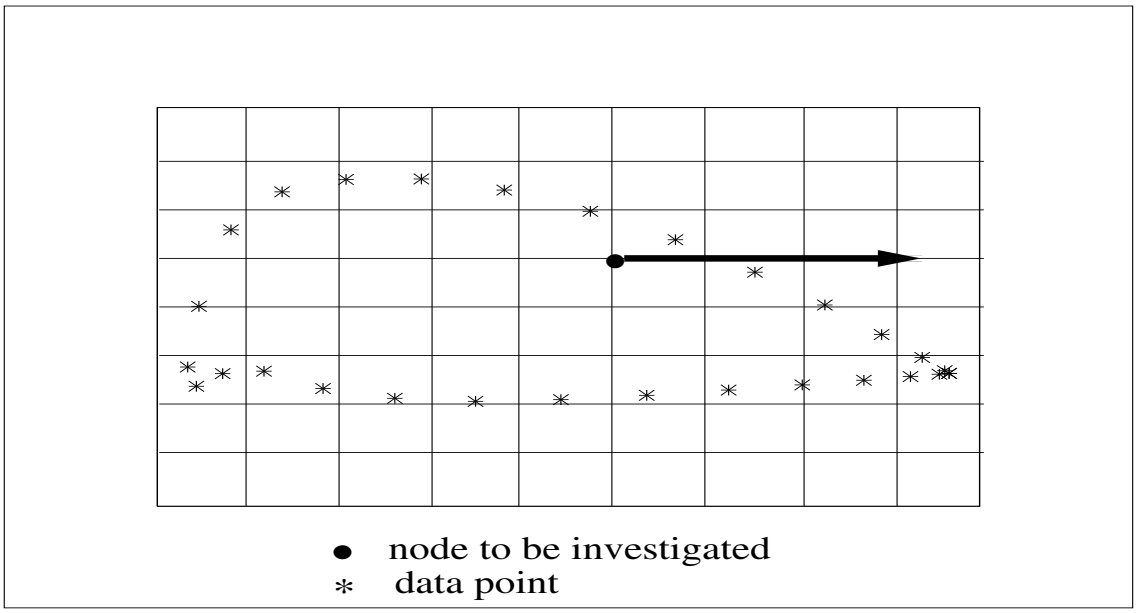

Figure 2.8: The ray -casting method.

outer edge of the computational domain. In figure (2.8) it is assumed to spread horizontally in the positive x direction, the node is classified as inside the body if the ray - when traced from the node until it hits the outer edge of the computational domain - intersects the body boundary an odd number of times, on the other hand, the node is considered outside the body if the number of times is even. It might happen that the ray intersects the body boundary at a location that lies between two successive data points, in this case there must be a way to generate coordinates of the intersection point[12]. The most suitable tool for this task is the BEZIER curves.

BEZIER Curves

BEZIER curves are employed today very widely in the field of computer graphics[13]. They can be easily implemented in computer programs and provide an efficient way of drawing curves with unexpected geometry. A BEZIER curve can be of first degree, of second, or higher degree depending on the number of points that are incorporated. For instance a second degree BEZIER curve is constructed by three points; two end points and one control point. BEZIER curves can be built using the BERNSTEIN polynomials; one can write accordingly,

[12]This problem does not exist in three dimensional geometries since the whole body surface is represented by adjacent triangles, see Appendix(D); however, BEZIER *surfaces* are used during the refinement process in order to have smooth body surfaces.

[13]Among many other softwares; the Adobe's postscript and Adope-Illustrator of the *Adobe Systems Inc.* use the BEZIER curves.

$$P(t) = P_1 B_{0,2} + P_2 B_{1,2} + P_3 B_{2,2} \tag{2.1}$$

Equation (2.1) applies for both the x and y coordinates of points P_1, P_2, and P_3. $B_{i,n}$ are the BERNSTEIN polynomials[14] , they have the following general form for a degree n:

$$B_{i,n}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i} \tag{2.2}$$

where $i = 0, 1, 2, \ldots n$, and $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}$. In equation (2.2) t is a parameter that might have a value between zero and one; that is, $t \in [0, 1]$. According to the definition of the BERNSTEIN polynomials, equation (2.2), the BEZIER curve of second degree is written as:

$$P(t) = P_1 (1-t)^2 + P_2 2t(1-t) + P_3 t^2 \tag{2.3}$$

The first term on the right hand side of equation (2.3) is obtained from equation (2.2). By setting $n = 2$, and $i = 0$, one gets: $B_{0,2}(t) = (1-t)^2$, similarly for $n = 2$ and $i = 1$, the second term is: $B_{1,2}(t) = 2t(1-t)$ and lastly, the third term is obtained for $n = 2$ and $i = 2$ that leads to $B_{2,2}(t) = t^2$.

It should be noted that the degree of the BEZIER curve to be used in representing the body geometry is determined by the number of data points that represent the body geometry. In other words, when the body geometry is given by a great number of data points, it was found that BEZIER curves of first degree are quite good in correctly representing the geometrical details of the given body.

Grid Generation Numerical Algorithm

The grid generation process for a given geometrical configuration can be summarized as follows: A loop over all the cells of the initial grid is executed where for each cell

[14]The computer graphics society sometimes refer to them as *the basis functions*.

another loop is executed that starts a ray at each node (corner) in the cell and traces this ray until the outer edge of the computational domain is hit by the ray. Assuming that the chosen ray is parallel to the x-axis as shown in figure (2.8), the first information known is that the y coordinate of the probable intersection point is equal to that of the node. Now all the generated BEZIER curves that have x-coordinate of any one of the end points that is greater than that of the node must be investigated by inserting the y coordinate of the node in the y version of the equation of the BEZIER curve and computing the value of the parameter t. By feeding this value of the t parameter into the x version of the equation of the BEZIER curve we get the x coordinate of the BEZIER curve at that y value. If the obtained x coordinate is greater than that of the node then the ray intersects the surface of the body. Otherwise the ray does not intersect the surface of the body. In case of intersection, the intersection coodinates are stored in that cell. It is worth noting that some computational time could be saved if the search is limited to those cells that are located in the *zooming* region, see figure (2.5). Finally, a grid smoothing algorithm is applied to all flow cells[15] to ensure an accurate solution. Each cell is allowed to be neighboured by no more than two cells in each direction.

2.3 Geometry -Based Automatic Grid Adaptation

In general, the grid adaptation process occurs either before the solution is triggered or during the solution process. The first is related to the geometrical configuration of the body and hence it is called *geometry -based grid adaptation*. The second is related to the values of the variables after the solution has been advanced in time and the slopes of the variables at some specific features, e.g., stagnation point, shock wave, ...etc. and this is called *solution -based grid adaptation*. In this section, it is focused on the first type of the grid adaptation, see NAKAHASHI [49], the second type of grid adaptation will be presented in section (4.5).

After the initial grid has been generated and all cells have been classified into flow, silver, and body cells, each cut cell is refined and the emerging new cells are also classified. This process is repeated until the largest dimension of the cut cell of the last generation is less than a user defined dimension (or level). For viscous flow computations, an approximate estimation of the boundary layer is performed such that the resulting grid will have at least nine grid nodes inside the boundary layer, see FERZIGER and PERIC [21]; of course the more grid points inside the boundary layer the higher the resolution that would be obtained. DE ZEEUW and POWELL [16] recommended[16] that the slopes of the body faces on two consecutive cut cells are compared and if the difference in slopes is above a threshold value, both cells are refined. They expressed the mathematical form of the test as:

[15] Silver cells are always smaller than flow cells.
[16] There recommendation was for inviscid calculations.

$$\left|\left(\frac{\Delta y}{\Delta x}\right)_{cell1} - \left(\frac{\Delta y}{\Delta x}\right)_{cell2}\right| \leq 0.05 \tag{2.4}$$

Special care should be taken at the faces with small Δx to avoid division by zero.

Chapter 3

Mathematical Model

"Aerodynamically, the bumblebee should not be able to fly, but the bumblebee does not know it so it goes on flying anyway."

Mary Kay Ash

When a compressible, viscous, non-reacting fluid is regarded as a mathematical continuum, its motion is described by the three basic fundamental laws of physics; namely the mass, the linear momentum, and the energy conservations[1]. Accordingly, the NAVIER(1823) -STOKES(1845) equations together with the mass and the energy conservation equations - or one of their several simplified forms - are the starting point for a lot of fluid dynamics simulations.

3.1 Governing Equations

The integral conservative form of the conservations equations, without external and body forces and assuming no heat sources, can be written as:

[1]The flow of chemically reacting flow considers extra relations such as chemical equilibrium constants, reaction rates, and heat of formations.

$$\int_{\Omega} \frac{\partial \vec{Q}}{\partial t} \mathrm{d}\Omega + \oint_{A} \overline{\overline{H}} \cdot \vec{n} \cdot dA = 0 \tag{3.1}$$

The vector $\vec{Q}$ denotes the conservative variables, these are: The ρ, the $\rho\vec{v}$ and the ρe. The flux vector $\overline{\overline{H}}$ denotes the inviscid (subscript $_{inv}$) and the dissipative (subscript $_{vis}$) fluxes through the volume surface A:

$$\vec{Q} = \begin{pmatrix} \rho \\ \rho\vec{v} \\ \rho e \end{pmatrix} \qquad \overline{\overline{H}} = \overline{\overline{H}}_{inv} - \overline{\overline{H}}_{vis} = \begin{pmatrix} \rho\vec{v} \\ \rho\vec{v}\vec{v} + p\overline{\overline{I}} \\ \vec{v}(\rho e + p) \end{pmatrix} - \begin{pmatrix} 0 \\ \overline{\overline{\tau}} \\ \overline{\overline{\tau}} \cdot \vec{v} + \vec{q} \end{pmatrix} \tag{3.2}$$

In the above relation e denotes the specific total energy, $\overline{\overline{I}}$ is the identity tensor, $\overline{\overline{\tau}}$ represents the viscous shear stress tensor, and $\vec{q}$ the heat flux vector.

Taking a cell in the cartesian grid as the control volume, one can rewrite the second term of equation (3.1) as:

$$\overline{\overline{H}} \cdot \vec{n} = \vec{E} \cdot n_x + \vec{F} \cdot n_y + \vec{G} \cdot n_z \tag{3.3}$$

Equation (3.1) can be rewritten in a dimensionless form as:

$$\int_{\Omega} \frac{\partial \vec{Q}}{\partial t} \mathrm{d}\Omega + \oint_{A} (\vec{E} \cdot n_x + \vec{F} \cdot n_y + \vec{G} \cdot n_z) \cdot dA = 0 \tag{3.4}$$

The Flux vectors $\vec{E}$, $\vec{F}$, and $\vec{G}$ can be written in terms of an inviscid and a viscous components, hence:

$$\vec{E}_{inv} - \vec{E}_{vis} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho e + p) \end{pmatrix} - \frac{1}{Re_o} \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \end{pmatrix} \tag{3.5}$$

$$\vec{F}_{inv} - \vec{F}_{vis} = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ \rho v w \\ v(\rho e + p) \end{pmatrix} - \frac{1}{Re_o} \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y \end{pmatrix} \tag{3.6}$$

$$\vec{G}_{inv} - \vec{G}_{vis} = \begin{pmatrix} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ w(\rho e + p) \end{pmatrix} - \frac{1}{Re_o} \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + q_z \end{pmatrix} \tag{3.7}$$

In equations (3.5-3.7), the Re_o denotes the REYNOLDS number based on the stagnation conditions, the reference variables used to normalize the governing equations are given in Appendix (A).

Applying the GAUSS' theorem[2] to equation (3.4) and differentiating with respect to the volume, one gets the differential form of the conservation equations:

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0 \tag{3.8}$$

The components of the symmetric shear stress tensor [3] $\overline{\overline{\tau}}$ for a *Newtonian* fluid are given as function of the velocity gradients as:

$$\tau_{xx} = \frac{2}{3}\mu \left(2\frac{\partial u}{\partial x} - \left(\frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right) \tag{3.9}$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \tag{3.10}$$

[2] Also known as the *divergence theorem*.
[3] STOKESstresses

$$\tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \tag{3.11}$$

$$\tau_{yy} = \frac{2}{3}\mu \left(2\frac{\partial v}{\partial y} - \left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} \right) \right) \tag{3.12}$$

$$\tau_{yz} = \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \tag{3.13}$$

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

Turning now to the heat flux vector $\vec{q}$, which can be written in a dimensionless form as[4]:

$$\vec{q} = \frac{k}{Pr_o(\gamma_o - 1)} \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{pmatrix} \tag{3.15}$$

With k is the thermal conductivity, T is the temperature, Pr_o is the PRANDTL number, and γ_o is the ratio of specific heats.

One more equation is needed in order to have a number of equations that is equal to the number of unknowns (ρ, u, v, w, e, and T). This will be taken as the ideal gas equation of state[5]. This assumption is well verified, see WYLEN and SONNTAG [72]:

$$p = \rho RT \tag{3.16}$$

using the thermodynamic relation $C_v = \frac{R}{\gamma_o - 1}$ and the definition of the specific internal energy $e = C_v T$, the equation of state can be reformulated to relate the pressure p to the conservative variable ρe as:

[4] According to FOURIER 's law of heat conduction.

[5] For the purposes of which this work was intended, i.e. air at moderate temperatures and pressures $\leq 10MPa$.

$$p = (\gamma_o - 1)\left(\rho e - \frac{1}{2}\rho\vec{v}^2\right) \tag{3.17}$$

For air at a wide temperature range (210 to 1800K), the PRANDTL number is considered constant ($Pr_0 = 0.72$). This in addition to the assumption of a constant γ_o implies that:

$$\mu(T) = k(T) \tag{3.18}$$

For air in the above mentioned temperature range, the dynamic viscosity can be considered, as suggested by WHITE [71], as a function of temperature only and it is calculated with an error of less than $\pm 4\%$ via

$$\frac{\mu}{\mu_o} = \left(\frac{T}{T_o}\right)^{0.72} \tag{3.19}$$

3.2 Boundary Conditions at the Fluid -Body interface

The formulation of the boundary conditions of a cartesian flow solver where many of the boundary cells are cut by the body boundaries is not an easy task. Those small irreuglarly -shaped silver cells require a special treatment for two reasons: Firstly, their geometrical shape is not like the other uncut cells, so the control volume taken around them is different and depends on the geometrical details of the cell after it has been cut. Secondly, the time integration step for such a cell is so small that it will render the solution unstable. In figure (3.1), the different types of silver cells are shown. Examining this figure, one concludes that the integration of this cell either by merging it into another neighbouring cell, by following a rotated difference scheme, or by a flux redistribution scheme is not the fast and efficient way for treating such cells. The motivation for the integration might be an increased accuracy, but it turned out that this was not achieved by the different investigators, as was mentioned in chapter (1). Moreover, all of the three previously mentioned approaches are very tedious to incorporate in a computer program. Also, it is worth to mention that the interpolation of the variables on the two sides of the cut plane is a very time -consuming process because of the different possibilities of the position

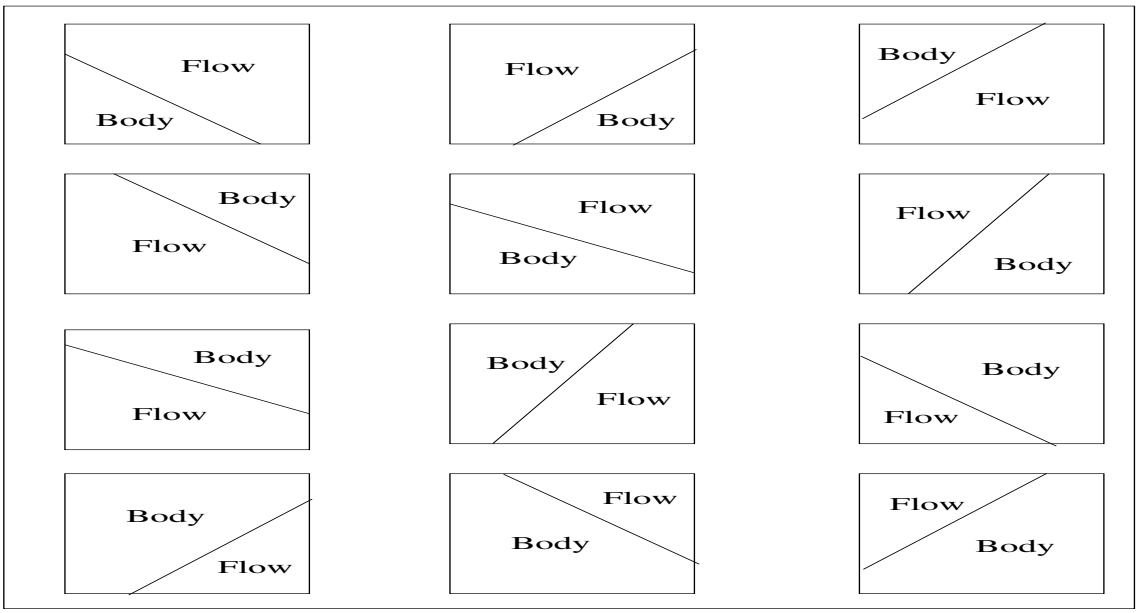

Figure 3.1: The different types of the silver cell in the two dimensional case.

of the cut plane. Therefore, the boundary conditions have directly been imposed on the silver cells. This technique was applied in the inviscid and the viscous flow calculations. In this work, it is assumed that the body walls are impermeable in both inviscid and viscous flow case. The implementation details of the boundary conditions will be explained in the next subsections.

3.2.1 Inviscid Flow

Considering a boundary cell as shown in figure (3.2), the boundary cell can be a flow or a silver cell; however, the boundary conditions are the same. Since the body surface is a stream surface, the velocity at the boundary cell is tangential to the cut plane. The density and the pressure are both extrapolated from the neighbouring flow cells[6]. The same applies for the temperature if the assumption of an adiabatic wall is employed. In a mathematical form this is written as:

$$\vec{v}_{bc} = \vec{v}_{fc} - (\vec{v}_{fc} \cdot \vec{n}) \cdot \vec{n} \tag{3.20}$$

$$\rho_{bc} = \frac{1}{n} \sum_{i=1}^{n} \rho_{fc} \tag{3.21}$$

[6]All flow cells that share either a face or a vertex with the boundary cell.

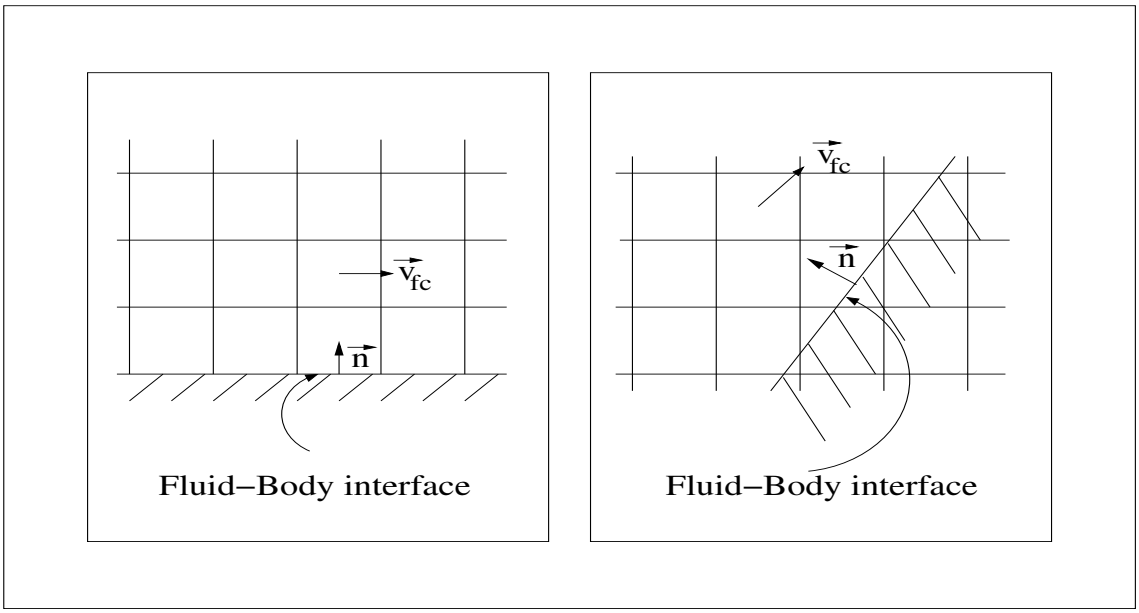

Figure 3.2: A boundary cell can be a flow or a silver cell.

$$p_{bc} = \frac{1}{n}\sum_{i=1}^{n} p_{fc} \tag{3.22}$$

$$T_{bc} = \frac{1}{n}\sum_{i=1}^{n} T_{fc} \tag{3.23}$$

In the above equations the subscript $(_{bc})$ and $(_{fc})$ are used to denote the boundary and flow cells, respectively. The density ρ, the pressure p, and the temperature T were taken as the average value of that of the neighbouring cells. Equation (3.20) can be considered as a mapping tool that transfers the velocity vector in the flow cell, $\vec{v}_{fc}$, to a velocity vector that is parallel to the flow-body plane, $\vec{v}_{bc}$, see [21].

3.2.2 Viscous Flow

In the viscous fluid flow, the velocity at the wall is equal to zero and the treatment of the boundary cells is carried out following the procedure first introduced by Kalitzin and Iaccarino [35]. The idea of this procedure is very simple and can be explained using figure (3.3). The velocity components (u and v) at the flow cell have

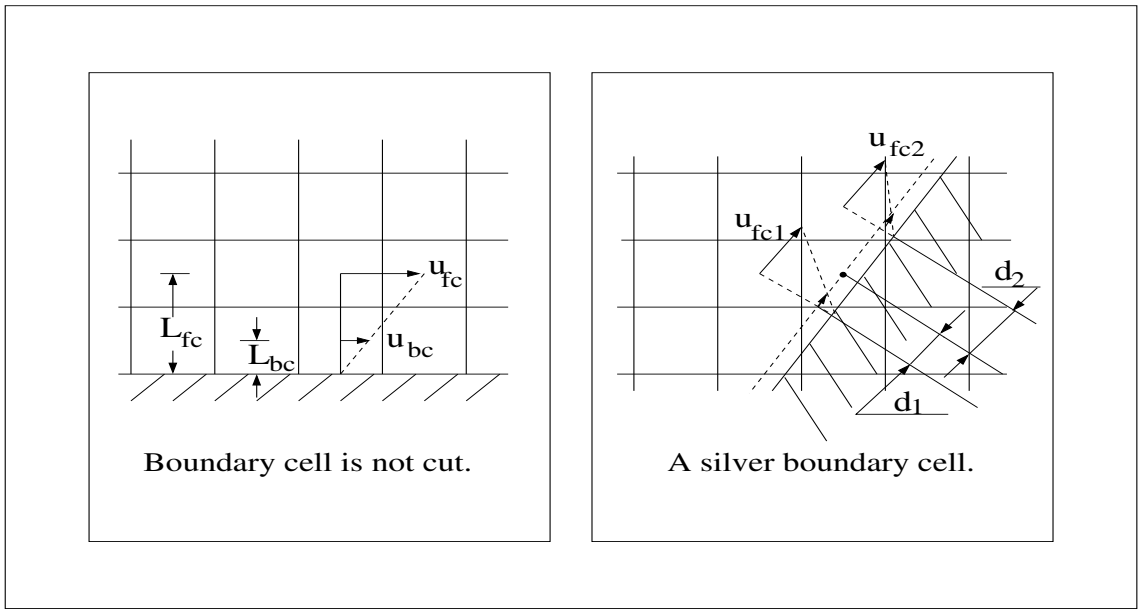

Figure 3.3: Velocity interpolation in the boundary cell in viscous fluid flow.

values of u_{fc} and v_{fc}, while at the wall both are equal to zero. The velocity at the boundary cell is interpolated in two steps: First, interpolating linearly the velocity at a location that has the same distance to the wall as the boundary cell (eqn.3.24). In the second step, the inverese -distance weighted method; see i.e. FRANKE [25] is employed to obtain the final value of the velocity component (eqn.3.25).
In mathematical notation, this can be expressed as[7]:

$$u_{bc} = u_{fc} \cdot \left(\frac{L_{bc}}{L_{fc}} \right) \tag{3.24}$$

Unless the boundary cell is a silver cell, there is no need for further treatment. Otherwise, equation (3.24) is applied to two face -neighbouring flow cells as shown on the right of figure (3.3), this leads to two values of the velocity component u_{bc}; namely, u_{bc1} and u_{bc2}. Now the inverse -distance weighted method which has the property of producing smooth reconstructions is used to compute the final value of the velocity component u_{bc} as:

$$u_{bc} = \frac{(u_{bc1} \cdot d_2 + u_{bc2} \cdot d_1)}{(d_1 + d_2)} \tag{3.25}$$

[7]Only the x-component is shown here; however, the y-component was treated similarly.

3.3 Far Field Boundary Conditions

Suitable boundary conditions must be specified at the outer edge of the computational domain[8] in order to solve the NAVIER-STOKES equations. Since in almost all applications of aerodynamics the systems considered are dominated by the hyperbolic propagation phenomena (i.e. the convective terms) the analysis that follows applies to the one -dimensional EULER equations. The outcome of this analysis is directly applicable to multi -dimensions, see HIRSCH [31]. Consider figure (3.4) in

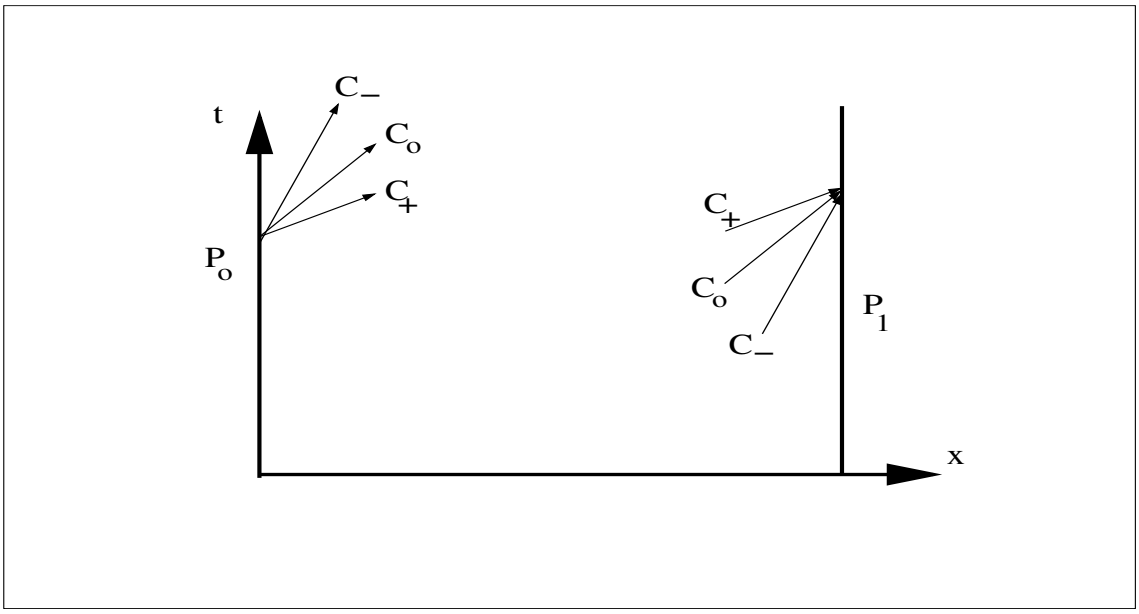

Figure 3.4: Supersonic inlet and outlet flow boundaries.

which a supersonic inlet and a supersonic outlet plane with the three characteristics C_o, C_-, and C_+ are illustrated. At the inlet plane P_o, the physical boundary conditions to be imposed depend on the information transported along the characteristics. The slopes of the three characteristics are u, $u + a$, and $u - a$. These slopes are all positive in this case, and thus three flow variables must be imposed. At the outlet plane P_1, all the slopes are also positive and accordingly all three variables must be extrapolated from the flow field.

Now consider figure (3.5) in which a subsonic inlet and a subsonic outlet plane with the three characteristics C_o, C_-, and C_+ are illustrated. At the inlet flow plane P_o, one of the slopes, namely, C_- is negative. This requires that one of the flow variables be extrapolated from the flow field, while the other two be imposed. The same occurs at the outlet plane P_1; that is, only C_- has a negative slope and reaches

[8]The outer edge of the computational domain might be an inflow or an outflow edge depending on the local velocity vector $\vec{v}$.

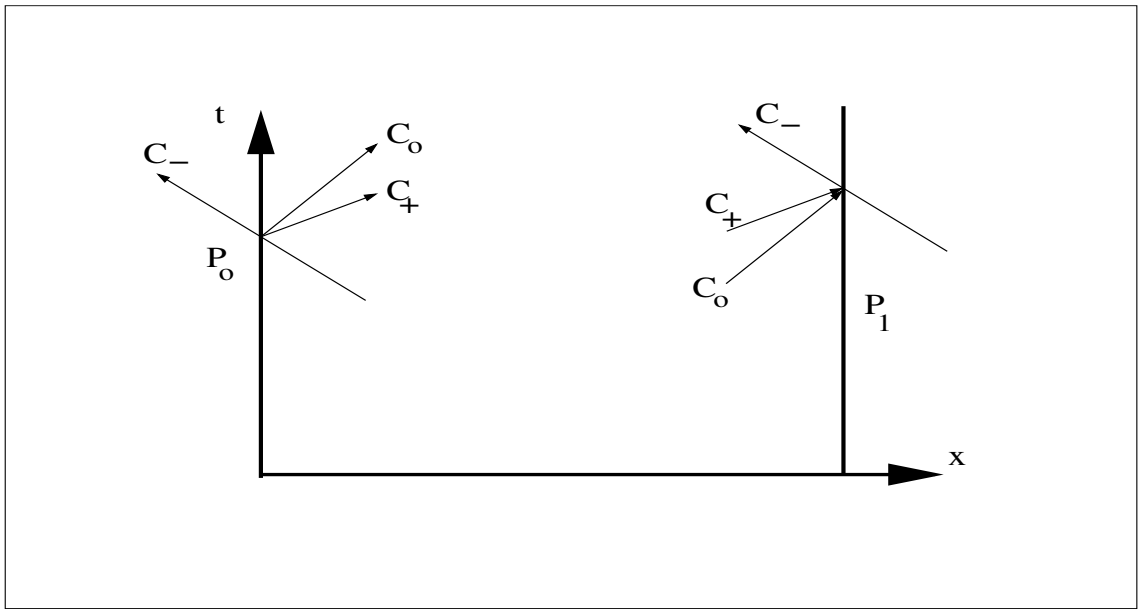

Figure 3.5: Subsonic inlet and outlet flow boundaries.

the outlet from outside the computational domain. Thus only one flow variable is imposed and two are extrapolated from the flow field.

The above analysis was based on the characteristic form of the EULER equations; nevertheless, it has proven to provide an acceptable numerical behaviour in alot of aerodynamic computations. The formulation of boundary conditions on artificial boundary is still a quite active field of research [65, 57] since the better the definition of the boundary values the more accurate and more efficient the solution.

The boundary conditions - used in this work - are summarized as:

Supersonic Inflow and Outflow

$$\vec{Q}_{inflow} = \left(\rho_\infty, \rho_\infty \cdot \vec{v}_\infty, \frac{p_\infty}{\gamma_o - 1} + \frac{1}{2}\rho_\infty \cdot \vec{v}_\infty^2 \right)^T \tag{3.26}$$

$$\vec{Q}_{outflow} = \left(\rho_{cd}, \rho_{cd} \cdot \vec{v}_{cd}, \frac{p_{cd}}{\gamma_o - 1} + \frac{1}{2}\rho_{cd} \cdot \vec{v}_{cd}^2 \right)^T \tag{3.27}$$

Subsonic Inflow and Outflow

$$\vec{Q}_{inflow} = \left(\rho_\infty, \rho_\infty \cdot \vec{v}_\infty, \frac{p_{cd}}{\gamma_o - 1} + \frac{1}{2}\rho_\infty \cdot \vec{v}_\infty^2\right)^T \tag{3.28}$$

$$\vec{Q}_{outflow} = \left(\rho_{cd}, \rho_{cd} \cdot \vec{v}_{cd}, \frac{p_\infty}{\gamma_o - 1} + \frac{1}{2}\rho_{cd} \cdot \vec{v}_{cd}^2\right)^T \tag{3.29}$$

In equations (3.26) to (3.29) the subscript $_\infty$ refers to the undisturbed far field conditions, the subscript $_{cd}$ refers to values at the adjacent flow cell in the computational domain, and the superscript ^{T} denotes the transpose of the matrix of the conservative variables.

3.4 Initial Conditions and a CFL Cut-Back Procedure

Initial values for the primitive variables are required so that the computations can be advanced in time. Generally the distribution of the variables is not known in the computational domain for a specific task, which is why the flow is initially assumed to be uniform. The initial conditions are calculated and stored in each computational cell.

$$\vec{Q}_{init} = \left(\rho_\infty, \rho_\infty \cdot \vec{v}_\infty, \frac{p_\infty}{\gamma_o - 1} + \frac{1}{2}\rho_\infty \cdot \vec{v}_\infty^2\right)^T \tag{3.30}$$

At the beginning of the computations, the flow, which was originally initialized from parallel flow, undergoes drastic changes especially near the walls of the body. To avoid getting unphysical distribution in the initial state, a cut -back procedure for the CFL that was proposed by Coirier [13] is implemented in this work. This can be achieved by requiring the maximum change of the normalized density or pressure to be less than a specified threshold value ε_{ts}; typically, a value of the order of 10^{-2}. The Courant number can be cut back by:

$$CFL_{cutB} = \min\left(CFL, \widehat{CFL}\right) \tag{3.31}$$

and

$$\widehat{CFL} = \frac{\varepsilon_{ts}}{\max\left(\varepsilon_\rho, \varepsilon_p\right)} \tag{3.32}$$

where CFL in equation (3.31) is the maximum COURANT number allowed by the time advancing scheme. Note that this procedure is needed only in the first $10 \sim 20$ time steps.

Chapter 4

Method of Solution

"Science never solves a problem without creating ten more."

George Bernard Shaw

The system of the conservation equations, equation (3.8), comprise a set of non-linear partial differential equations containing both space and time derivatives. The inviscid derivatives possess a hyperbolic nature while the diffusive derivatives have an elliptic nature. According to the nature of the spatial derivatives a *suitable* discretization scheme must be employed to transform these equations into an algebraic form that can be solved numerically by a digital computer. In this work the governing equations are discretized using a *collocated*, cell -centered arrangement of velocity and pressure and advanced in time using a five -stage RUNGE-KUTTA method.

The two - widely used - types of discretization schemes for the convective terms are: The central finite difference schemes and the upwind schemes. The central finite difference schemes were pioneered by LAX and WENDROFF [38, 39]. The upwind schemes were originated by COURANT et al. [14] who had tried to introduce the physical propagation information in the discretization scheme and later they were well established by GODUNOV [29] who introduced a higher level of physical information into what is now known as *Godunov-type* schemes.

Since the central difference schemes do not include the speed and direction of the propagating physical information, they require the inclusion of artificial dissipation terms to suppress the oscillations generated in the vicinity of discontinuities.

JAMESON [33] formulates these terms as functions of the local pressure gradient. In the present work, the $AUSM$ scheme that belongs to the family of upwind schemes was chosen from a variety of avialable upwind achemes[1] because of its simplicity and well performance on the body -fitted structured grids, i.e. see [59, 60].

The $AUSM$-scheme requires as input the values of the variables at the right and left of the cell faces, these values are not directly available since the variables are stored at cell centers. A linear reconstruction is used to interpolate the variables at the cell faces. In the next section the reconstruction method that was employed will be presented.

4.1 The Reconstruction of the Variables

The primitive variables $\vec{W} = (\rho, u, v, p)^T$, where linearly reconstructed as:

$$\vec{W}(x, y) = \vec{W}(x_{cc}, y_{cc}) + \phi \cdot \nabla\vec{W} \cdot \vec{dr} \tag{4.1}$$

In the above equation, the subscript $_{cc}$ refers to the cell center, see figure (4.1), $\nabla\vec{W}_k$ is the slope of the primitive variables at the cells centers, and ϕ is the limiter function. Since these slopes are not known they should be computed. The computation of the slopes is performed by employing the *Least Squares Method*, see BATINA [5] and LIU and SU [44]. The details will be given here for completeness. In order to compute the slopes of a variable at the cell center, a linear distribution of this variable is assumed in the region enclosed by the nearest neighbours. For sake of simplicity[2], the neighbours incorporated in this process are all direct -face neighbours, see figure (4.2). The assumed equation of the linear distribution[3] of the variable can be written as:

$$f(x, y) = f_o + \frac{\partial f}{\partial x}\, x + \frac{\partial f}{\partial y}\, y \tag{4.2}$$

[1] For example: ROE-scheme or TORO-scheme.

[2] Numerical experiment has shown that including the vertex neighbours does not have any effect on the performance.

[3] A local coordinate system was used here that was located at the centroid of cell whose slope is sought.

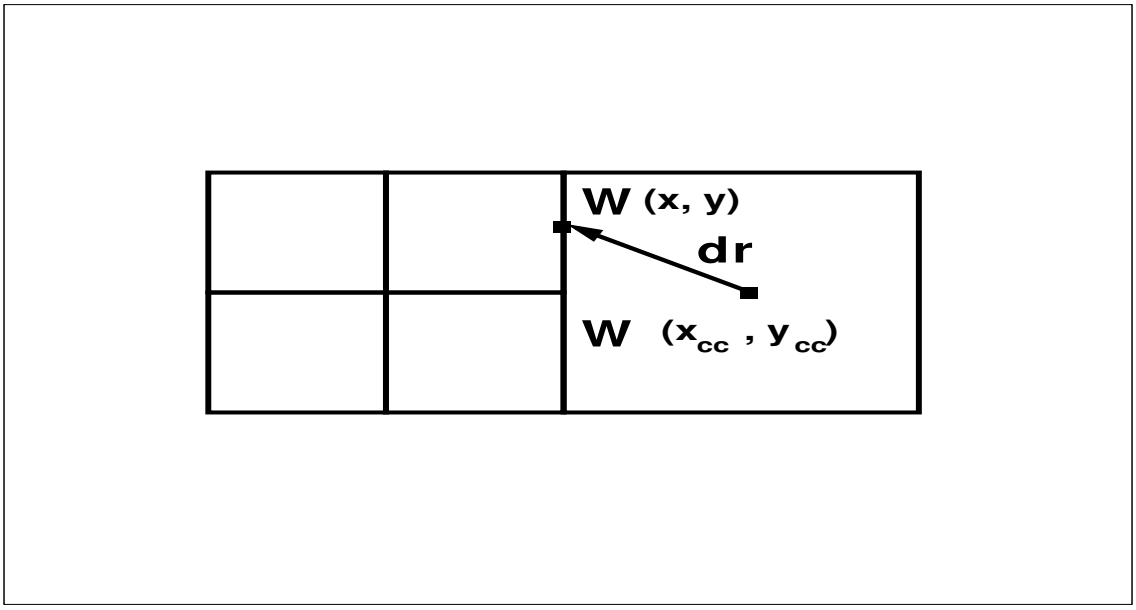

Figure 4.1: Variable reconstruction at the cell faces.

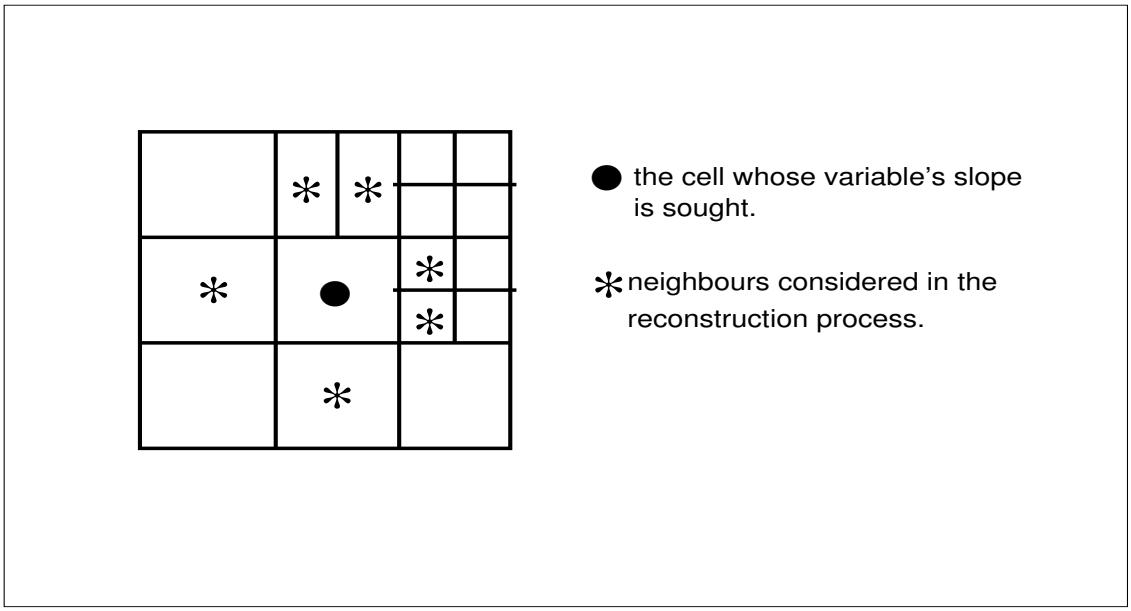

Figure 4.2: Direct -face neighbours are chosen for the slope computation.

Where $f \in \vec{W}$. Writing equation (4.2) for the cell considered and all the direct -face neighbours and using a least squares fit approach, the following system of algebraic equations is obtained:

$$\begin{pmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum {x_i}^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum {y_i}^2 \end{pmatrix} \cdot \begin{pmatrix} f_o \\ \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} \sum f_i \\ \sum f_i x_i \\ \sum f_i y_i \end{pmatrix} \tag{4.3}$$

The system of linear algebraic equations (4.3) can be solved by any standard method [50] for the values of $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. The treatment of the three dimensional case is similar, see Appendix (B).

If the full gradient were used in reconstructing the values at face midpoints, the computed values could fall outside the bounds of the data used in the recontruction process especially in the neibourhood of shock waves. To avoid this situation, the computed gradients are limited via a limiter function ϕ. In the present work the ROE limiter function was used.

$$\phi = \begin{cases} \min\left(|\frac{2\Delta^+}{\Delta^+ + \Delta^-}|, |\frac{2\Delta^-}{\Delta^+ + \Delta^-}|, 1\right) & if \quad sign\left(\Delta^+\right) = sign\left(\Delta^-\right) \\ 0 & if \quad sign\left(\Delta^+\right) \neq sign\left(\Delta^-\right) \end{cases} \tag{4.4}$$

In equation (4.4) the Δ^+ and Δ^- are defined as[4]:

$$\Delta^+ = \vec{W}_e - \vec{W}_w \qquad \Delta^- = \vec{W}_w - \vec{W}_{ww} \tag{4.5}$$

The subscripts $_{e,w,ww}$ refer to the cell that is on the eastern and western side and to the cell that is located to the west of the western cell of the face at which the reconstruction process takes place, respectively. Figure (4.3) shows the three cells participating in the limiter calculation.

[4]Here the x direction stencil is taken as an example, the other two directions are similar.

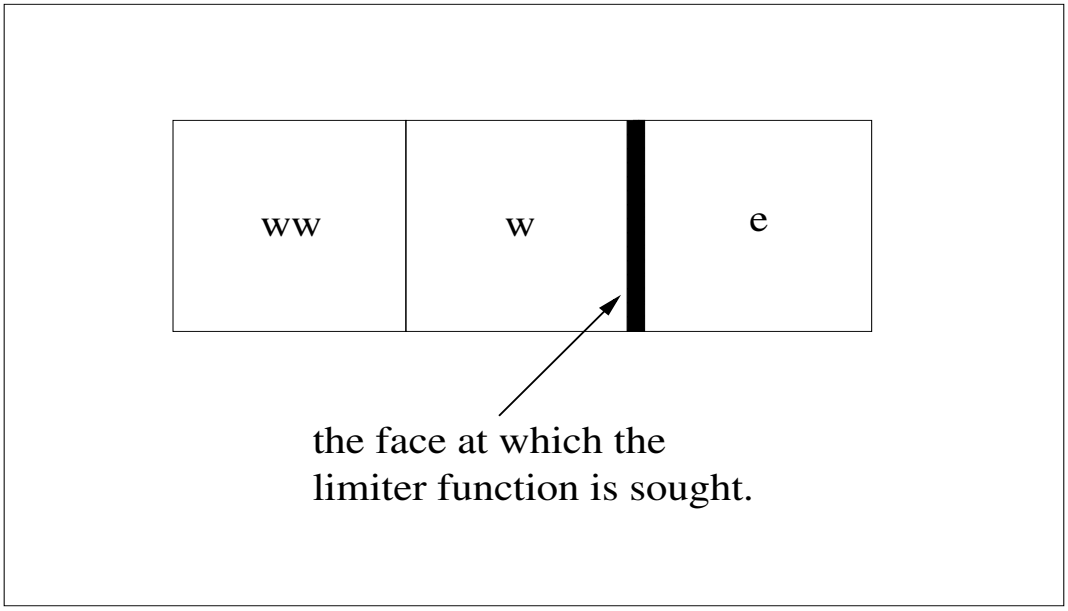

Figure 4.3: Cells that compose the stencil for the limiter function.

4.2 Discretization of the Inviscid Terms: The AUSM and the AUSM$^+$ – au Schemes

The AUSM Scheme :

The **A**dvection **U**pstream **S**plitting **M**ethod or what is known as the **AUSM** scheme was introduced by LIOU [41] in 1993 as a competetive and an attractive alternative for the ROE scheme [63]. Since that time it has been adopted by researchers worldwide because of its simplicity, robustness, and accuracy. In the original version of the scheme, the numerical inviscid flux $\vec{E}_{inv}$ in equation (3.5) - taken as an example[5] - is split into convective and pressure terms. A proper cell interface velocity $u_{1/2}$ is defined and used later to identify the upsteram direction, that is;

$$\vec{E}_{inv} = u_{1/2} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho H \end{pmatrix}_{L/R} + \begin{pmatrix} 0 \\ p_L^+ + p_R^- \\ 0 \\ 0 \end{pmatrix} \tag{4.6}$$

[5]The directions y and z are similar.

This interface velocity $u_{1/2}$ is represented by:

$$u_{1/2} = a_{L/R} M_{1/2} \tag{4.7}$$

In equation (4.6) the subscript $_{L/R}$ refers to either the *left or right* value of the variables at the cell face according to the upstream direction. Mathematically this can be expressed as:

$$(.)_{L/R} = \begin{cases} (.)_L & \text{if } u_{1/2} \geq 0; \\ (.)_R & \text{if } u_{1/2} < 0. \end{cases} \tag{4.8}$$

Substituting equation (4.7) into (4.6), leads to:

$$\vec{E}_{inv} = M_{1/2} \begin{pmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a H \end{pmatrix}_{L/R} + \begin{pmatrix} 0 \\ p_L^+ + p_R^- \\ 0 \\ 0 \end{pmatrix} \tag{4.9}$$

The convective MACH number $M_{1/2}$ is defined by combining the wave speed of a left and right running wave. Rewriting this in a compact form yields:

$$M_{1/2} = M_L^+ + M_R^- \tag{4.10}$$

In defining the split MACH numbers M_L^+ and M_R^- two cases are distiguished based on the *kinematic state* of the flow at the two sides abutting the location where the flux is sought (the cell face). That is, either the flow is supersonic or subsonic. Based on the VAN LEER scheme [67] LIOU chose a second order polynomial for the subsonic state and a first order for the supersonic state.

$$M_{L/R}^{\pm} = \begin{cases} \pm 1/4 \left(M_{L/R} \pm 1 \right)^2 & \text{if } |M_{L/R}| \leq 1; \\ 1/2 \left(M_{L/R} \pm |M_{L/R}| \right) & \text{if } |M_{L/R}| > 1. \end{cases} \tag{4.11}$$

A similar treatment[6] was given to the pressure leading to an equation for the pressure that is similar to equation (4.10):

$$p_{1/2} = p_L^+ + p_R^- \tag{4.12}$$

and similarly two polynomials are used to represent the acoustic waves of the pressure. It should be noted that a third order polynomial is used for the subsonic state and a second order polynomial for the supersonic state.

[6] With the exception of the *upstream biasing*; that is, the pressure wave can travel in all directions and hence does not undergo any upwinding.

AUSM	**AUSM$^+$ – au**
Upwinding is decided according to the local MACH number that is computed from the local flow speed and the local speed of sound.	Upwinding is decided according to the local MACH number that is computed from the local flow speed and the common numerical speed of sound.
In subsonic flow, the split pressure $p^{\pm}$ is computed from a third order polynomial in MACH number.	In subsonic flow, the split pressure $p^{\pm}$ is computed from a fifth order polynomial in MACH number.
In subsonic flow, the split MACH number $M^{\pm}$ is computed from a second order polynomial in MACH number.	In subsonic flow, the split MACH number $M^{\pm}$ is computed from a fourth order polynomial in MACH number.

Table 4.1: Comparison between $AUSM$ and $AUSM^+ - au$ schemes.

$$p^{\pm}_{L/R} = \begin{cases} \pm p/4 \left(M_{L/R} \pm 1\right)^2 \left(2 \mp M_{L/R}\right) & \text{if } |M_{L/R}| \leq 1; \\ p/2 \left(M_{L/R} \pm |M_{L/R}|\right) / M_{L/R} & \text{if } |M_{L/R}| > 1. \end{cases} \tag{4.13}$$

The AUSM$^+$ – au Scheme:

Since the appearance of the first version of the $AUSM$ scheme it has undergone a fast development phase by making use of the hybrid nature of the scheme and by introducing the common numerical speed of sound. The result was the appearance of the two derivatives of the $AUSM$ scheme; namely, the $AUSMDV$ scheme which is an adaptable version of the scheme that can adjust itself to either a flux vector splitting scheme or a flux difference splitting scheme. Thus it enables the scheme to benefit the merits of both splittings, and the so called $AUSM^+$ in which the numerical speed of sound was introduced. In [43] LIOU explained some pitfalls of the existing versions of his invented *masterpiece* and recommended some remedies. Table (4.1) shows a comparison between the original $AUSM$ scheme and the $AUSM^+ - au$[7] scheme [43]. It is worth noting here that in addition to those differences in tabel (4.1) the pressure flux term was expanded by a diffusion term to enhance convergence when calculating flows at low speeds. Appendix (C) explains the details of the $AUSM^+ - au$ scheme.

[7] These two versions were used in this work.

4.3 Discretization of the Viscous Terms

The second difficulty of the anisotropic cartesian flow solver after the issue of the bounday conditions is the discretization of the viscous terms. This was first pointed out by COIRIER [13] who has devoted a complete chapter in his thesis to the analysis of this problem. He showed that the main problem originates from the grid being highly distorted (due to the anisotropic refinement) and thus the choice of the dicretization stencil is not an easy task. In his analysis, he was confronted by two opposing requirements. They are accuracy and positivity. He recommended the use of a *diamond* path scheme; that is, a divergence theorem based reconstruction where the contributing cells form a diamond path around the face in question, see figure (4.4). Once this scheme is implemented on an anisotropically refined grid it

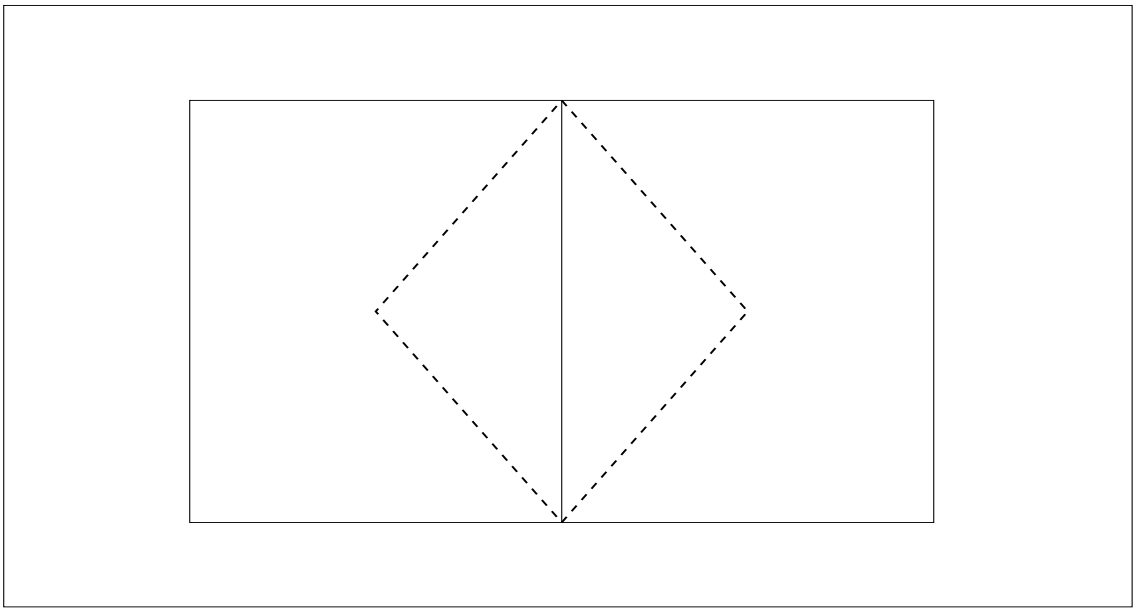

Figure 4.4: Diamond path proposed by COIRIER .

must be modified by including some linearity preserving weighting function.

HAM et al. [30] made use of the auxiliary node concept of FERZIGER and PERIC [21] to solve the incompressible NAVIER-STOKES equations. In their scheme, they split the viscous flux term into *active* and *lagged* parts. Although they claim a better performance than that achieved by the scheme proposed by COIRIER , the author's numerical experiments on the current compressible NAVIER -STOKES solver could not confirm their result.

In this work the velocity slopes that are used in calculating the viscous terms are

taken from the existing slopes that were computed in the variables reconstruction process, see section (4.1). In order to eliminate the *velocity -pressure decoupling* the slopes at the face are computed by considering the slopes of the two cells sharing this face and implicitly including the effect of their neighbours in obtaining the slope at this face. This can be better explained with the help of figure (4.5) where the derivative of the x -component of the velocity u with respect to x is taken as an example. Once the two slopes u_{x1} and u_{x2} are reconstructed at the face to yield u_{x1rc}

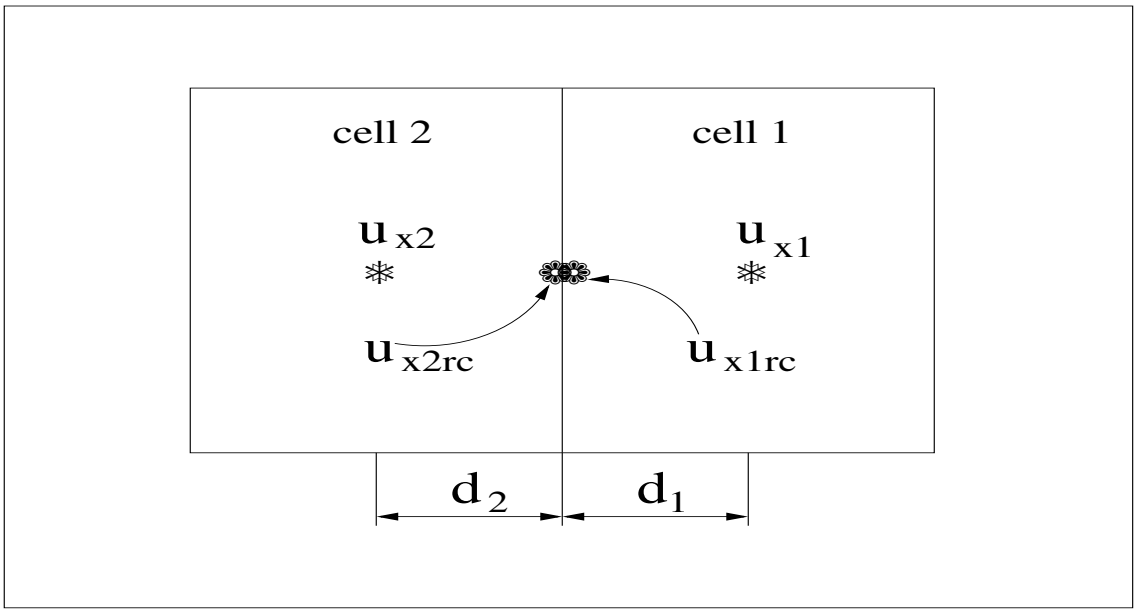

Figure 4.5: Constructing a four points stencil at the face between two cells.

and u_{x2rc}, respectively, the problem is looked at now as an interpolation problem; that is, there is now a stencil of four points that have four values of the slopes, see figure (4.6). A *rational* third degree BEZIER curve is assumed to reperesent the slope -distribution curve that passes through the two end points (the slopes at cell centers), and the two control points (the two reconstructed slopes). The rational BEZIER curves are similar to the ordinary BEZIER curves, equation (2.1), with the addition of scalar weights λ_i that occur in each term of the BEZIER curve equation.

$$P(t) = P_1 B_{0,3} \lambda_1 + P_2 B_{1,3} \lambda_2 + P_3 B_{2,3} \lambda_3 + P_4 B_{3,3} \lambda_4 \tag{4.14}$$

The values for the weights λ_i were obtained through numerical experiments. Typical values are $\lambda_{i,i=1,\ldots,4} = (1.0, 2.5, 2.5, 1.0)$. The parameter t can be assigned a value that depends on the spatial location of the face with respect to the centers of the two cells sharing that face. A relation of the form

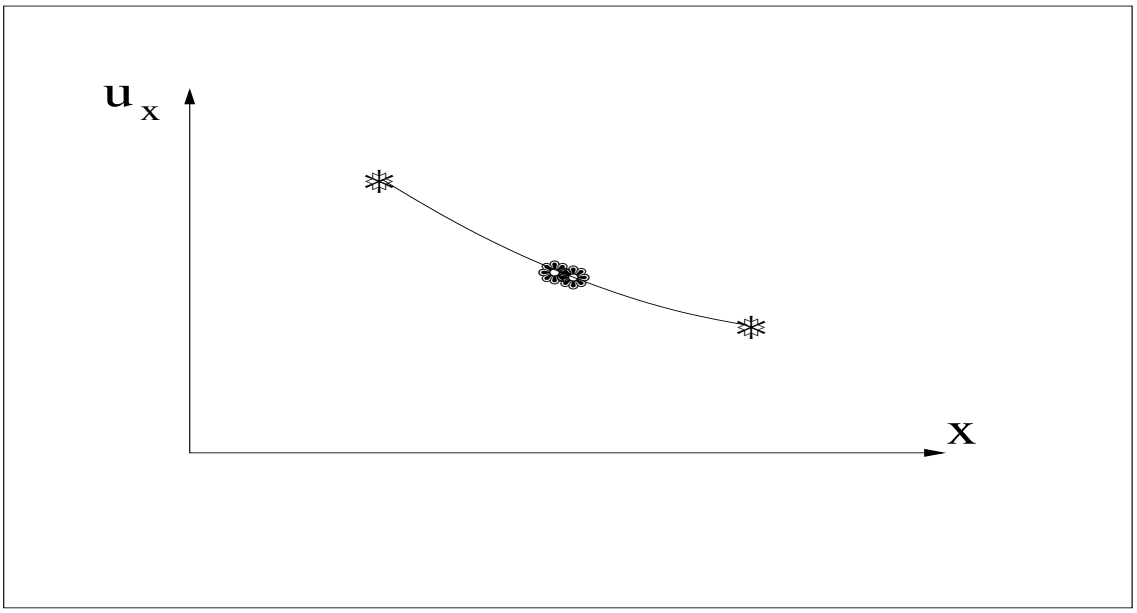

Figure 4.6: A BEZIER curve that represents the slope is assumed.

$$t = \frac{d_1}{d_1 + d_2} \tag{4.15}$$

was found to give satisfactory results. The above value of t is used with a rational BEZIER curve of third degree

$$u_x = (1-t)^3 u_{x1}\lambda_1 + 3t(1-t)^2 u_{x1rc}\lambda_2 + t^2(1-t) u_{x2rc}\lambda_3 + t^3 u_{x2}\lambda_4 \tag{4.16}$$

4.4 Time Stepping Scheme

After the inviscid and the viscous terms are discretized they are transferred to the right hand side of equation (3.8) yielding:

$$\frac{\partial \vec{Q}}{\partial t} = Res\left(\vec{Q}^n\right) \tag{4.17}$$

In equation (4.17); the superscript ^{n} refers to the time level n. Equation (4.17) can be integrated in time using either an implicit or an explicit time integration schemes. In order to avoid the greater memory requirements of the implicit scheme an explicit multistage RUNGE- KUTTA integration scheme of the form:

$$\begin{aligned}
\vec{Q}^{(0)} &= \vec{Q}^n \\
\vec{Q}^{(1)} &= \vec{Q}^{(0)} + \alpha_1 \cdot \Delta t \cdot Res\left(\vec{Q}^{(0)}\right) \\
&\vdots \\
\vec{Q}^{(k)} &= \vec{Q}^{(0)} + \alpha_k \cdot \Delta t \cdot Res\left(\vec{Q}^{(k-1)}\right) \\
&\vdots \\
\vec{Q}^{n+1} &= \vec{Q}^5
\end{aligned}$$

was selected. The values of the coefficients α_k are as given by [61]:
$\alpha_1 = 0.059 \qquad \alpha_2 = 0.14 \qquad \alpha_3 = 0.273 \qquad \alpha_4 = 0.5 \qquad \alpha_5 = 1.0$
The time step is computed following the proposed procedure by GAFFNEY [27] of local time step Δt calculation for non-quadratic cells; that is:

$$\frac{1}{\Delta t} = \frac{1}{\Delta t_x} + \frac{1}{\Delta t_y} + \frac{1}{\Delta t_z} \tag{4.18}$$

where $\Delta t_x = \frac{CFL \quad \Delta x}{|u|+a}$, $\Delta t_y = \frac{CFL \quad \Delta y}{|v|+a}$, and $\Delta t_z = \frac{CFL \quad \Delta z}{|w|+a}$.

In order to accelerate the solution to steady state, the maximum local time step, equation (4.18), for each cell was used. For time accurate calculations, this procedure must be replaced by an algorithm that searches for the minimum time step and uses it globally for the whole computational domain.

4.5 Solution-Based-Automatic Grid Adaptation

The ultimate goal of a numerical solution is to provide an accurate and fast analysis for a given task. This can be achieved through the *optimal* design of the two main components of the numerical solution: the grid and the flow solver. Concentrating on the grid[8], it is assumed that having a fine grid all over the computational domain

[8]The idea of *solver adaptation* is also an attractive one, that is, to use a high resolution (computationally expensive) scheme in regions of higher importance details and a low resolution (computationally cheap) scheme in regions

provides a basis for obtaining high accuracies. On the other hand, the computational time will be increased because of the increased number of cells to be solved. The most likely solution to this conflict is to start the computation on a relatively coarse grid and perform a refinement where ever it is needed. To translate this thought into a computer code the following pseudo code is introduced:

```
for(all cells) {
    //check if any of the sensors is on
    if ( any sensor from a group of sensors is on ){
         tag this cell for refinement;
    }
}
```

This implies that there is a need for a tool that is designed to handle a large amount of data. This is a situation that perfectly matches the ***D**iscrete **W**avelet **T**ransform*, often referred to as **DWT**.

4.5.1 Wavelets as a Data Processing Tool

The origin of wavelets is interdisciplinary. That is, wavelets come from different fields such as engineering, theoretical physics, and mathematics. Lately, a large spectrum of applications has grown and is still developing, ranging from signal or image analysis and processing to numerical analysis. The most benefiting field from the wavelet theory is the image analysis and processing. The **W**avelet **S**calar **Q**uantization method that the **FBI**[9] uses to compress its fingerprint database is an example of such image analysis applications.

Mathematicians like to introduce and work out wavelets in terms of an algorithm applied to an infinite data set. This is known as the ***C**ontinuous **W**avelet **T**ransform*; however, in practical engineering applications the data sets are always finite. Accordingly, the **DWT** is more suited to practical applications although they might introduce the so -called boundary problem if the data does not end up at the edge of the wavelet window, see figure (4.8).

The essential function of wavelets is to allow the compression of a given set of data by

of lower importance details. Although this was not implemented in this work sofar, it is strongly recommended, see chapter (6).

[9] **F**edral **B**ureau of **I**nvestigation.

splitting it up into a low resolution part (scale) and a high resolution part (detail). The first serves as an indicator of the order of magnitude that the data have at a specific range, and the second gives the amplitude of the difference between the data at that range. This process is reversible; that is, with each wavelet transform there is an inverse wavelet transform. Figure (4.7) shows a schematic diagram of

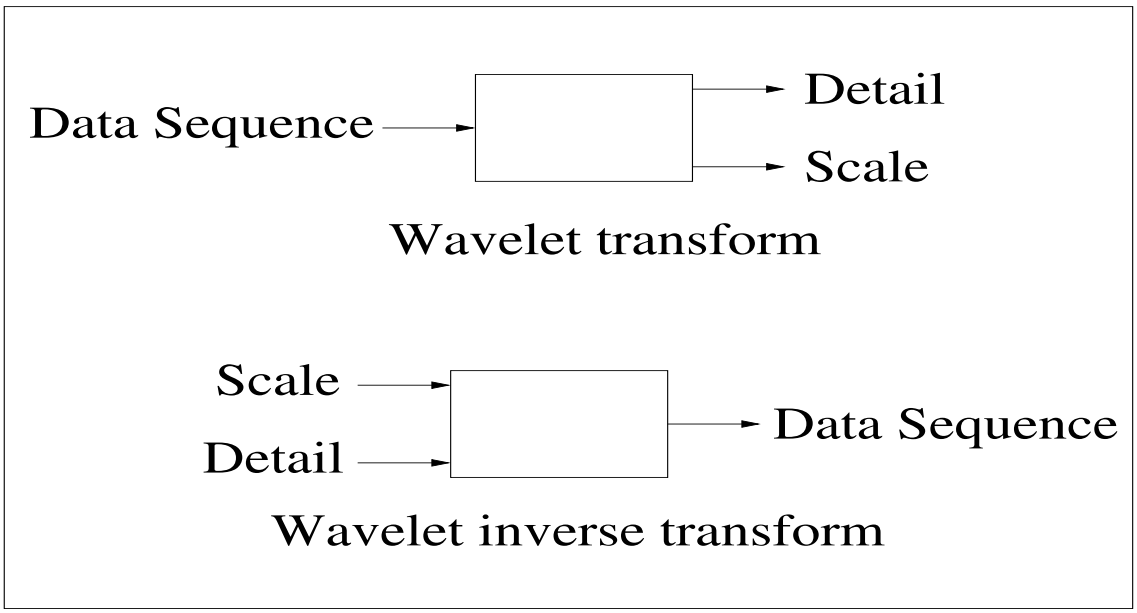

Figure 4.7: Wavelet transform and inverse wavelet transform.

this process.

The selection of the wavelet function depends on the application and the accuracy required. For instance, the HAAR's wavelet is very simple and easy to program, but it suffers from the odd-even or discontinuity problem. The DAUBECHIES' wavelets, on the other hand, are continuous but they do not have a closed analytic form. However, the very favourable advantage of the DEBAUCHIES' wavelets of being better suited to deal with general applications [68] makes it the candidate to implement in the present work. The D4 wavelet[10] has the following form:

$$S_i = a_0 h_0 + a_1 h_1 + a_2 h_2 + a_3 h_3 \tag{4.19}$$

$$D_i = a_0 g_0 + a_1 g_1 + a_2 g_2 + a_3 g_3 \tag{4.20}$$

[10]The DAUBECHIES' wavelet family includes the two, four, six, and eight data points windows, the D4 refers to the version that treat four data points.

where the $a_{i,i=0,\ldots,3}$ are the data points[11], the S_i is the scale[12] of the data at this data range, and D_i is the detail or more commonly *the wavelet function*[13]. The wavelet coefficients $h_{0,\ldots,3}$ and $g_{0,\ldots,3}$ are given as:

$$\begin{aligned} h_0 &= \tfrac{1+\sqrt{3}}{4\sqrt{2}} & g_0 &= h_3 \\ h_1 &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} & g_1 &= -h_2 \\ h_2 &= \tfrac{3-\sqrt{3}}{4\sqrt{2}} & g_2 &= h_1 \\ h_3 &= \tfrac{1-\sqrt{3}}{4\sqrt{2}} & g_3 &= -h_0 \end{aligned} \tag{4.21}$$

Two more items are left to close the presentation of the wavelets; namely, the *thresholding* and the *boundary problem*.

A widely used approach to calculate the refinement threshold is to employ the standard deviation of the data together with the mean value. That is, to set the refinement threshold to some fraction of a standard deviation above the mean value of the data distribution, see Warren et al. [70], Aftosmis [1], and Lahur et al. [37]. Using the mean value of the data together with the standard deviation, the refinement threshold can be written as:

$$Threshold = Mean - \psi \cdot \sigma \tag{4.22}$$

Where $Mean$ is the mean value of the data, σ is the standard deviation of the data, and ψ is a factor whose value is given by Lahur et al. [37] as 0.5. Another simpler yet faster approach to compute the threshold was used in this work that was originally introduced by Percival and Walden [52] who employ the $Median$ of the data. The formula suggested by Percival and Walden [52] is:

$$Threshold = \frac{Median}{0.6745} \tag{4.23}$$

Where the $Median$ is the statistical median of the data, and the factor 0.6745 rescales the numerator so that the threshold is also a suitable estimator for the standard deviation.

[11] It is also known as the wavelet window.
[12] It is also known as the low -pass filter
[13] It is also known as the high -pass filter.

If the number of the data points is not equal to $2n$ where $n = 2, 3, 4, \ldots$, then at the end of the data sequence the wavelet will lack a data point. This is known in the wavelet society as the edge or boundary problem, see figure (4.8). To overcome this

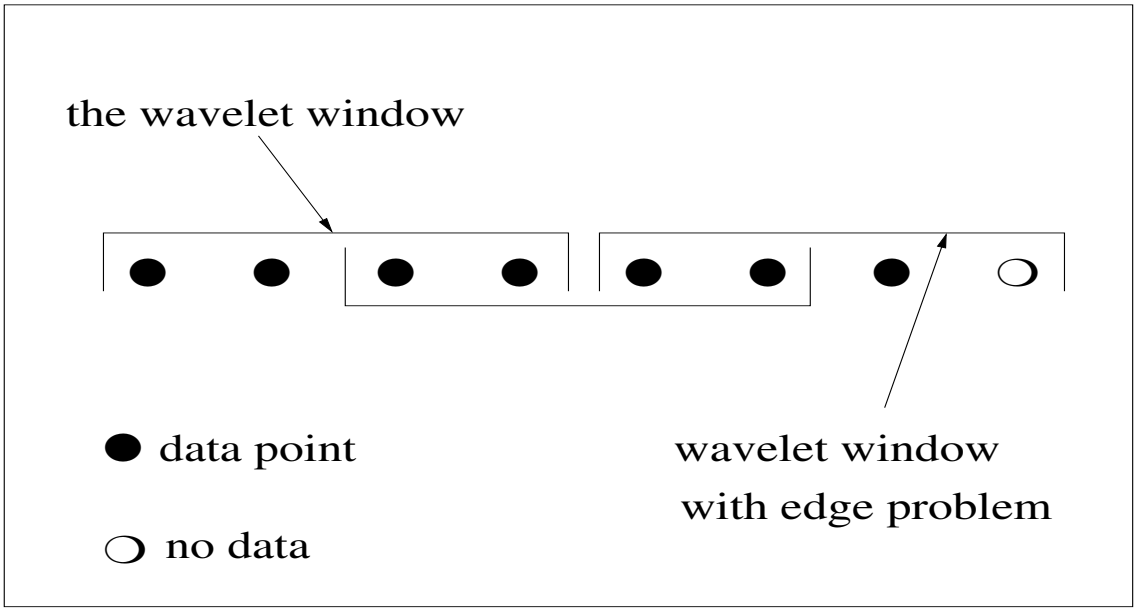

Figure 4.8: Wavelet window and the edge problem.

problem either of the following procedures can be done:

- By using zero padding; that is, filling the places that lack data with zeros.
- Mirroring the data.
- Assuming a periodic data.

Generally, the zero padding is the most unsuitable one for the job since it can produce significant errors. The other two are used according to the expected behaviour of the data sequence. The data compression algorithm[14] that was employed in the flow solver applies the periodic data whenever this problem emerges. That is, the data at the start of the sequence wraps around to the end.

4.5.2 Formulation of the Adaptation Sensor

As was mentioned in section before, at the beginning of the computations the locations at which there are extreme changes of the flow are not known. Moreover,

[14]Thanks to Ian Kaplan who did a terrific work on wavelets available at www.bearcave.com/misl/misl_tech/wavelets/

some of these features might change their location during the solution process[15]. In order to accurately resolve these features when they appear, there must be a *sensor* that senses the existence of such cases. Several types of adaptation sensors are used in the literature, including some that use the flow variables directly and others that use the slopes of the flow variables. The relative change in the density ρ, the total pressure P_t, and the magnitude of the velocity vector $||\vec{v}||$ were used in the inviscid flow calculations. In the viscous flow calculations the velocity gradients were also taken into consideration. The sensors are summarized as:

$$\chi_1 = \frac{\Delta\rho}{\rho} \quad \chi_2 = \frac{\Delta||\vec{v}||}{||\vec{v}||} \quad \chi_3 = \frac{\Delta P_t}{P_t} \quad \chi_4 = \frac{\partial u}{\partial y} \quad \chi_5 = \frac{\partial v}{\partial x} \tag{4.24}$$

The P_t in equation (4.24) is the total pressure given by:

$$P_t = p\left[\frac{\gamma-1}{2}M^2 + 1\right]^{\frac{\gamma}{\gamma-1}} \tag{4.25}$$

Numerical Algorithm of the Solution -Based Grid Adaptation

The numerical algorithm of the adaptation process proceeds as follows. The variables in equation (4.24) are calculated and plugged into the wavelet algorithm, equations (4.19, 4.20), that in turn compresses the data by splitting it into a sequence of scales and a sequence of details. This step is the main task performed by the wavelet transform algorithm. The sequence of the details is used in equation (4.23) to calculate the threshold for each sensor. Then, the detail of each sensor is compared with the corresponding threshold for every flow cell and the cell is tagged for refinement whenever a detail of one of the sensors is greater than its threshold; that is, if one of the five sensor equations (4.24) is *on*.

[15] For instance: a shock wave.

Chapter 5

Results

"If I have a thousand ideas and only one turns out to be good, I am satisfied."

Alfred Nobel.

The developed cartesian grid generator and flow solver were validated using a number of standard and non standard test cases. The standard test cases of the AGARD working group 07, see DJAFFAR et al. [17] and PULLIAM and BARTON [54] were chosen as validation test cases for the transonic inviscid flows, see table (5.1). The other non -standard inviscid flow test cases are validated through comparison with the MSES flow solver [45] that employs structured -body fitted grids for the EULER equations.

The viscous fluid flow is validated by considering two test cases: The first is the flow on a flat plate at zero incidence that can be compared to the BLASIUS solution. The

Test Case	Ma_∞	AoA
AGARD01	0.8	1.25^o
AGARD02	0.85	1.00^o
AGARD03	0.95	0.00^o
AGARD04	1.2	0.00^o
AGARD05	1.2	7.00^o

Table 5.1: Free stream MACH number and angle of attack for the AGARD01-05 test cases, from DJAFFAR et al. [17].

second test case that is encountered to validate the viscous fluid flow solver is the lid -driven cavity flow for which there are computational results that were obtained and published by GHIA et al. [28].

5.1 Inviscid Flow

The boundary conditions at the fluid -body interface and at the far field boundary that were used for the inviscid calculations are given by equations (3.20-3.23) and equations (3.26-3.29), respectively.

5.1.1 NACA0012 Aerofoil

The first test case to be considered is the transonic flow on the NACA0012 aerofoil at zero angle of attack and MACH number equal to 0.8. Although this case is not a standard one, it is often carried out as a validation test performed by CFD professionals, e.g. see [11]. The steady state solution includes two shock waves that are located at the same position on the upper and lower sides of the aerofoil. In figures (5.1-5.3) the grid, pressure contours, and the Mach number contours are shown, respectively. The grid shown here is the grid obtained by only geometry -based adaptation that was carried out before the flow solution. The distribution of the pressure coefficient C_p on the surface of the aerofoil is given in figure (5.4). The result obtained by the present flow solver is in good agreement with that obtained by the MSES program. The correct prediction of the stationary shock wave is achieved without the need for any further solution -based grid adaptation.

The AGARD01 test case is characterized by two shock waves on the upper and lower sides of the aerofoil that have different strengths, the lower is relatively weak and catching it is a measure of the accuracy of the used flow solver. The pressure contours, MACH number contours, and the distribution of the pressure coefficient C_p are shown in figures (5.5-5.7), where the two shocks are well captured and resolved.

The AGARD02 provides another different oppuritunity to test a flow solver. It is characterized by the presence of two unsymmetrical attached shock waves. The pressure contours, The MACH number contours, and the distribution of pressure coefficient C_p are shown in figures (5.8-5.10).

The transonic flow of the AGARD03 is termed as the *fish tail* shock wave; because two oblique shock waves are emanating form the trailing edge with an angle

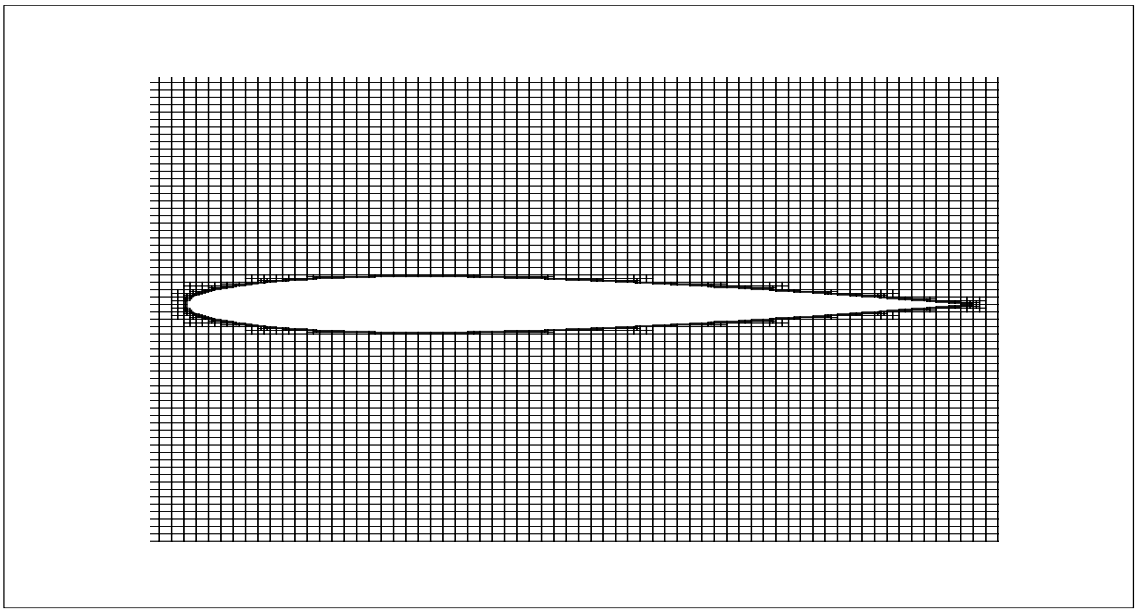

Figure 5.1: Initial grid with geometry -based grid adaptation for NACA0012 aerofoil.

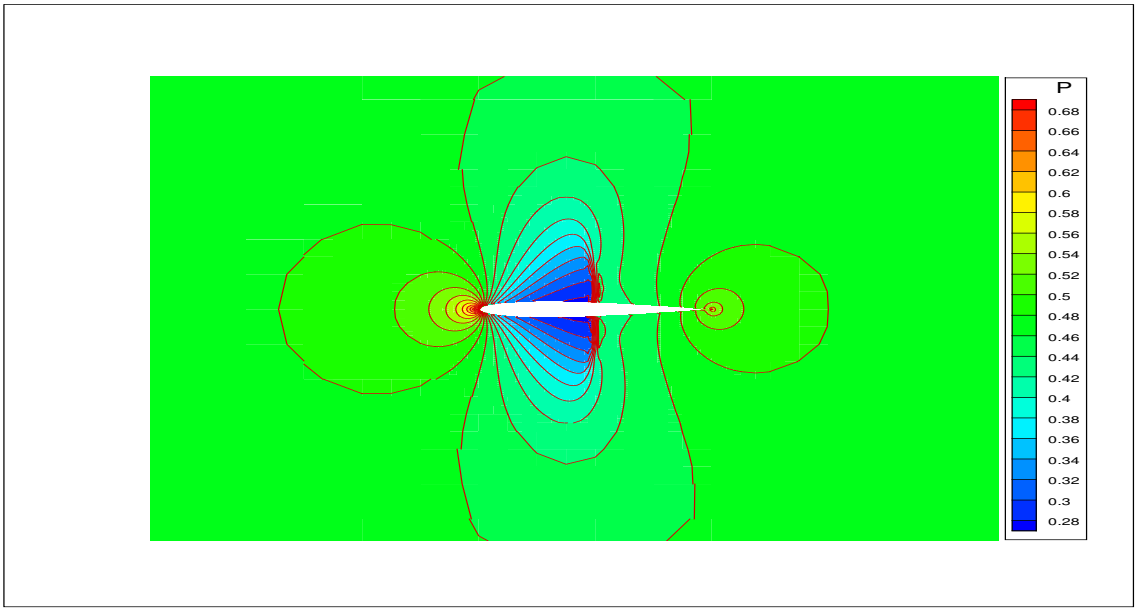

Figure 5.2: Pressure contours, NACA0012 aerofoil at $\alpha = 0°$ and MACH number $= 0.8$
.

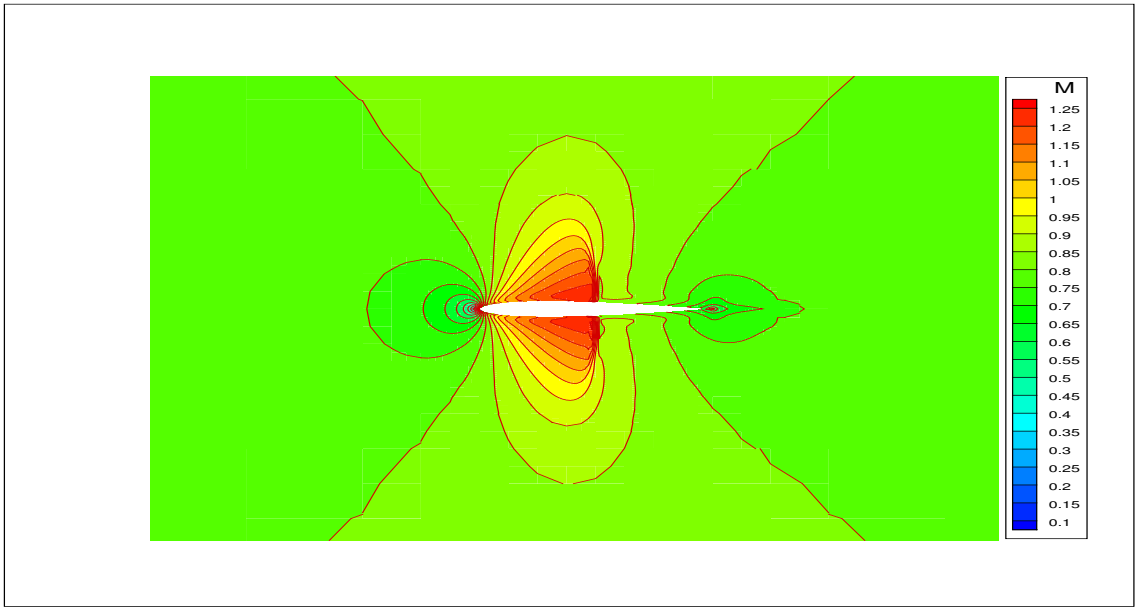

Figure 5.3: MACH number contours on NACA0012 aerofoil at $\alpha = 0°$ and MACH number = 0.8 .

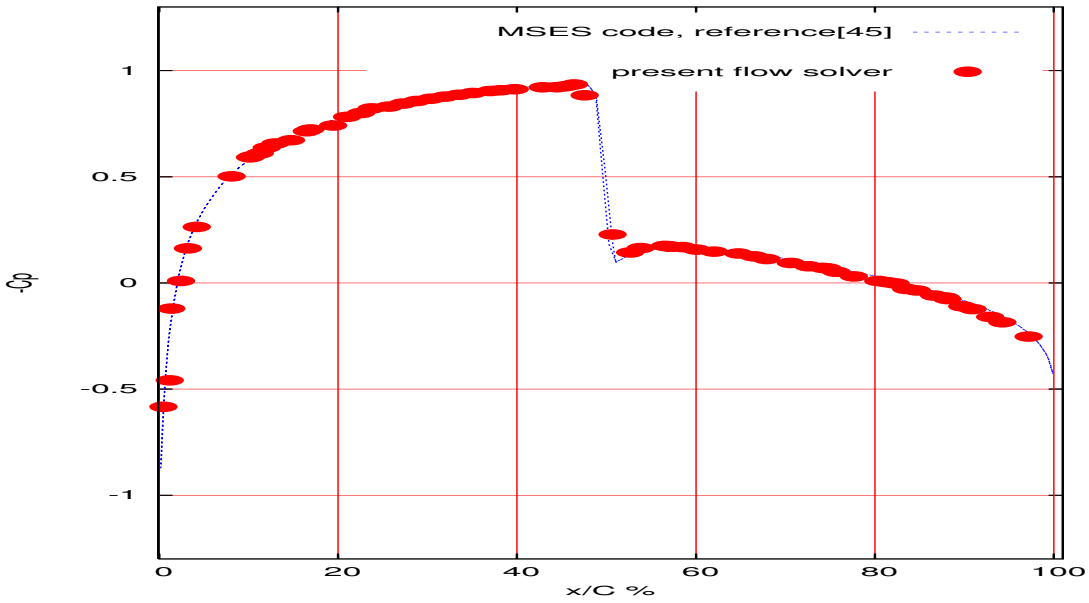

Figure 5.4: C_p distribution, NACA0012 aerofoil at $\alpha = 0°$ and MACH number = 0.8 .

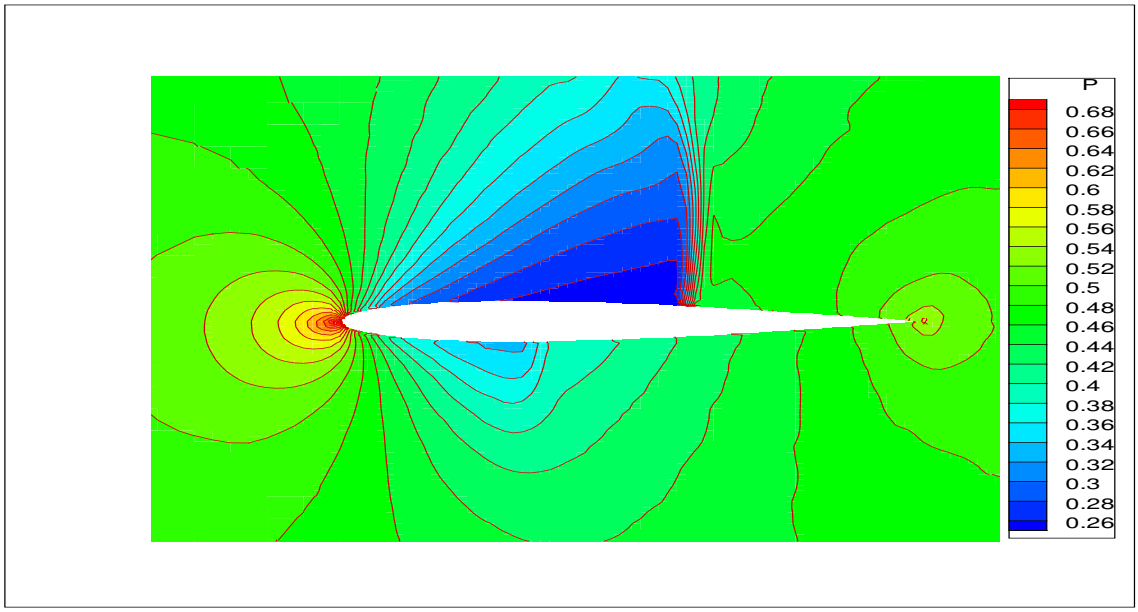

Figure 5.5: Pressure contours, AGARD01.

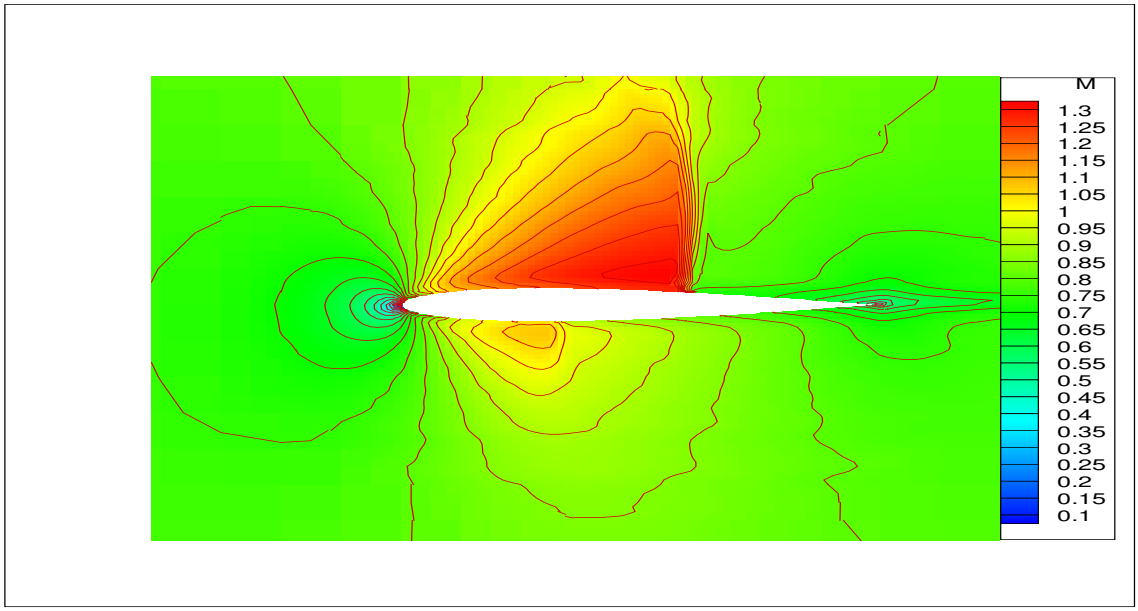

Figure 5.6: MACH number contours, AGARD01.

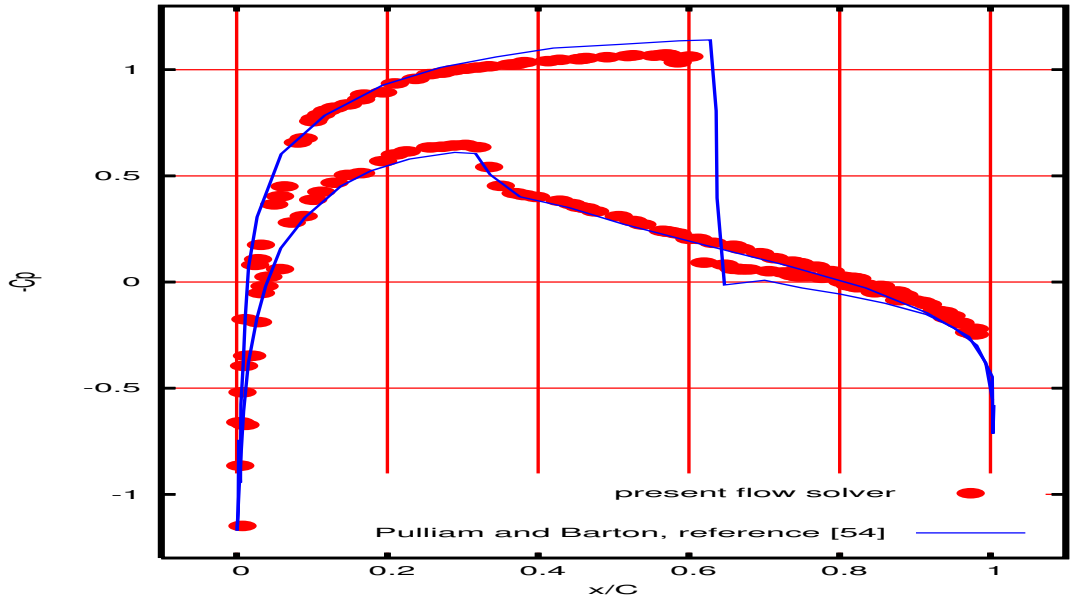

Figure 5.7: C_p distribution, AGARD01.

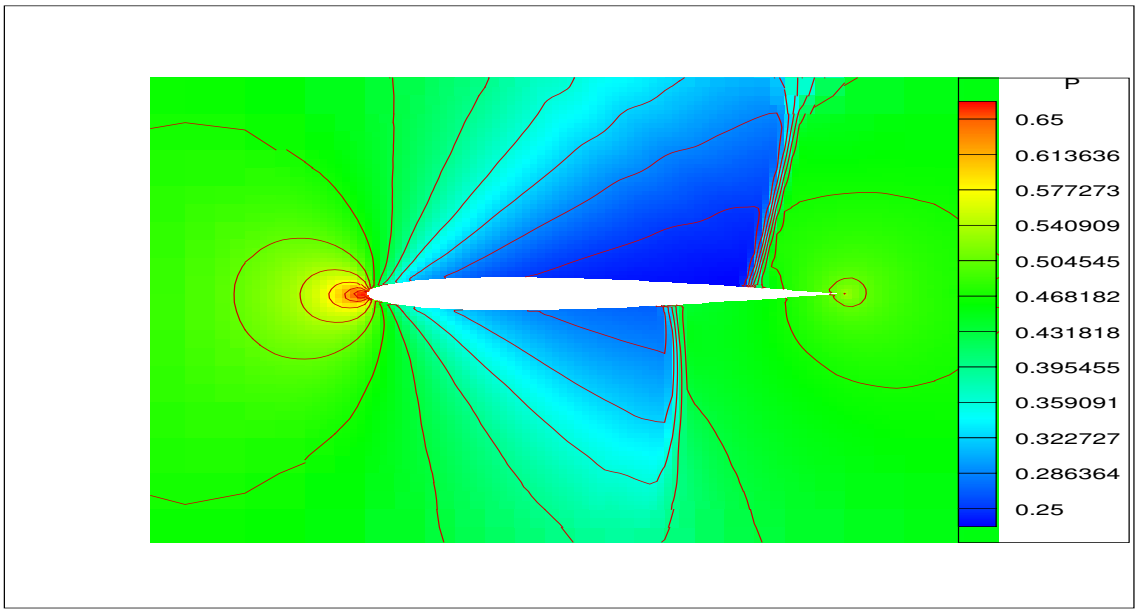

Figure 5.8: Pressure contours, AGARD02.

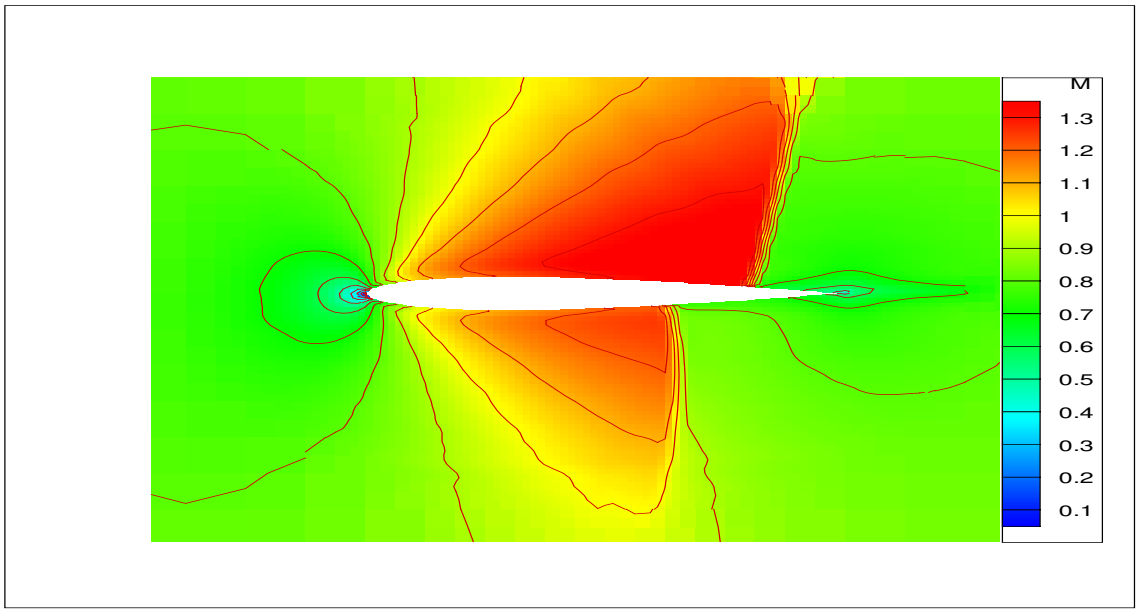

Figure 5.9: MACHnumber contours, AGARD02.

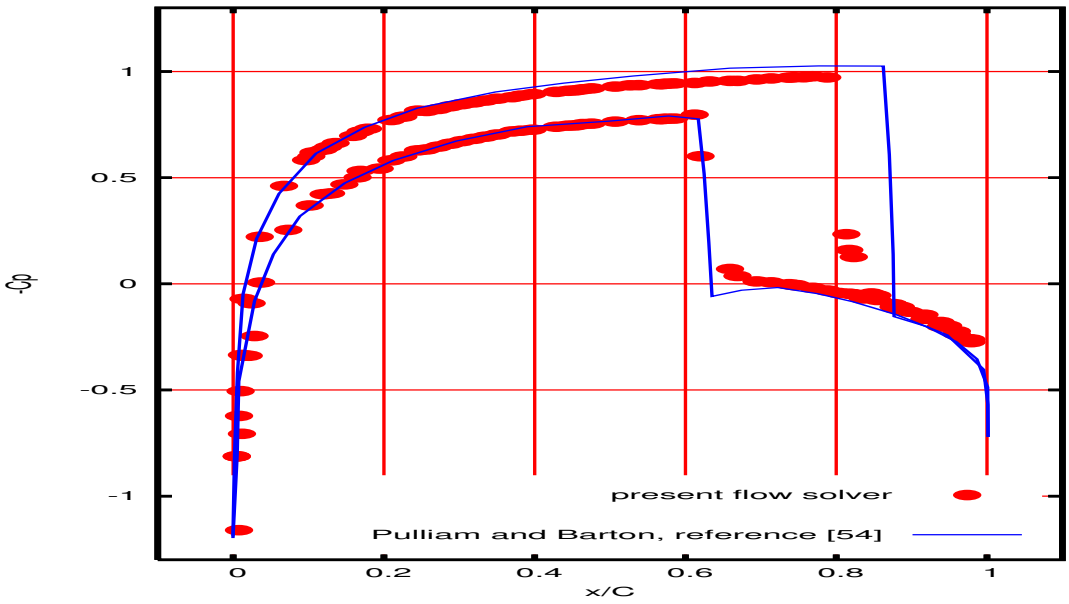

Figure 5.10: C_p distribution, AGARD02.

of about 55° with the chord, see figures (5.11-5.13).

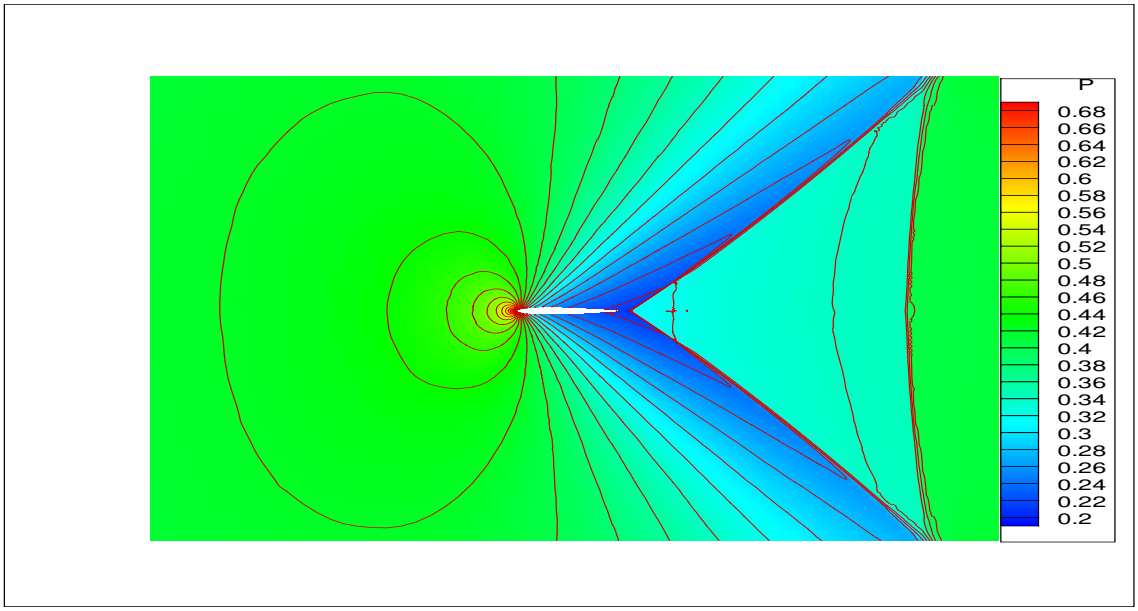

Figure 5.11: Pressure contours, AGARD03.

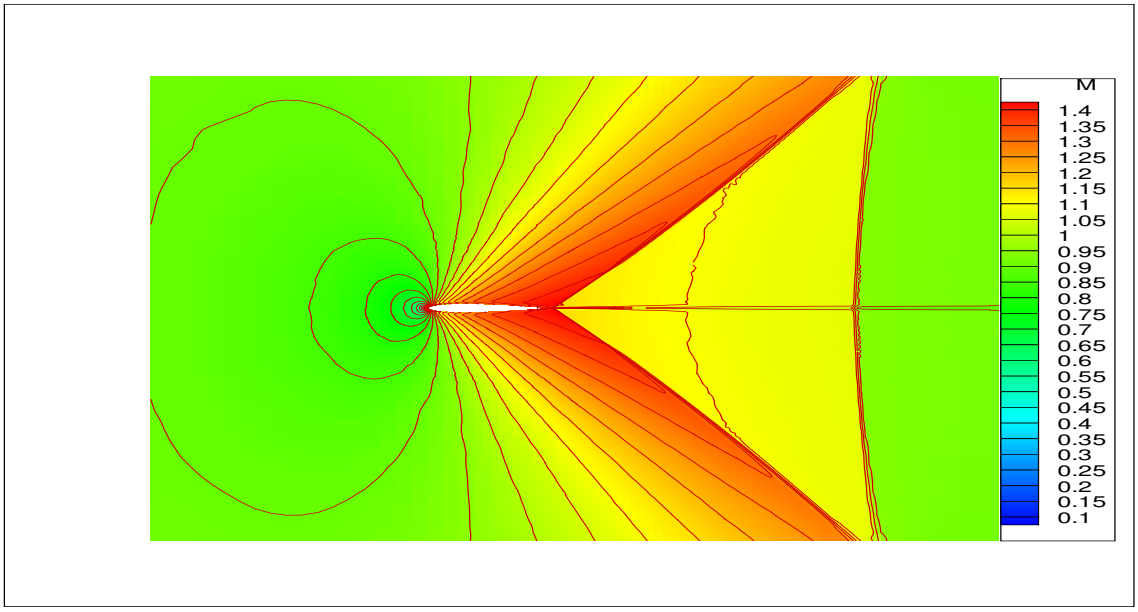

Figure 5.12: MACH number contours, AGARD03.

The pressure contours, MACH number contours, and the C_p distribution for AGARD04 are shown in figures (5.14, 5.17). Here it was observed that the oblique shock waves were better resolved after the grid had been adapted. Moreover, the adaptation sensors were very successful in capturing both the detached and the attached shock waves. The distribution of the pressure coefficient C_p in figure (5.17) has a good

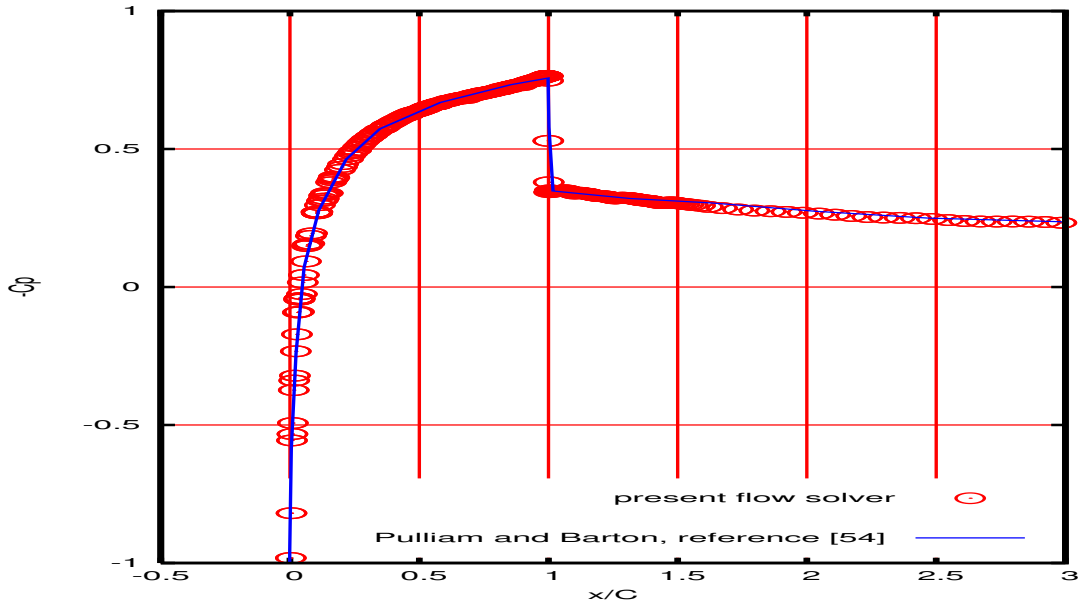

Figure 5.13: C_p distribution, AGARD03.

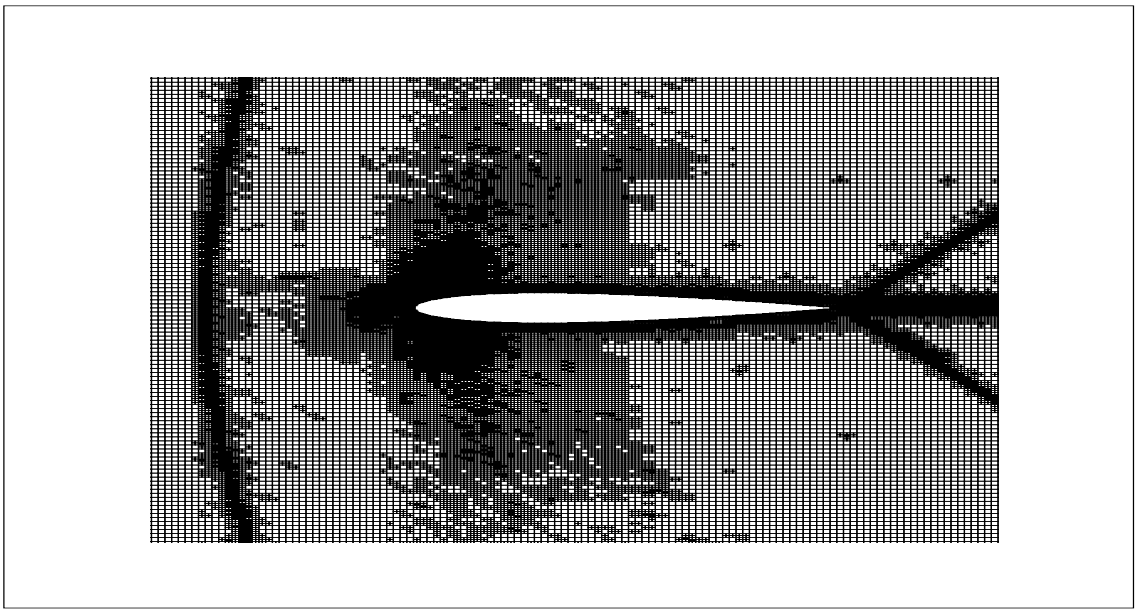

Figure 5.14: Grid after solution -based grid adaptation, AGARD04.

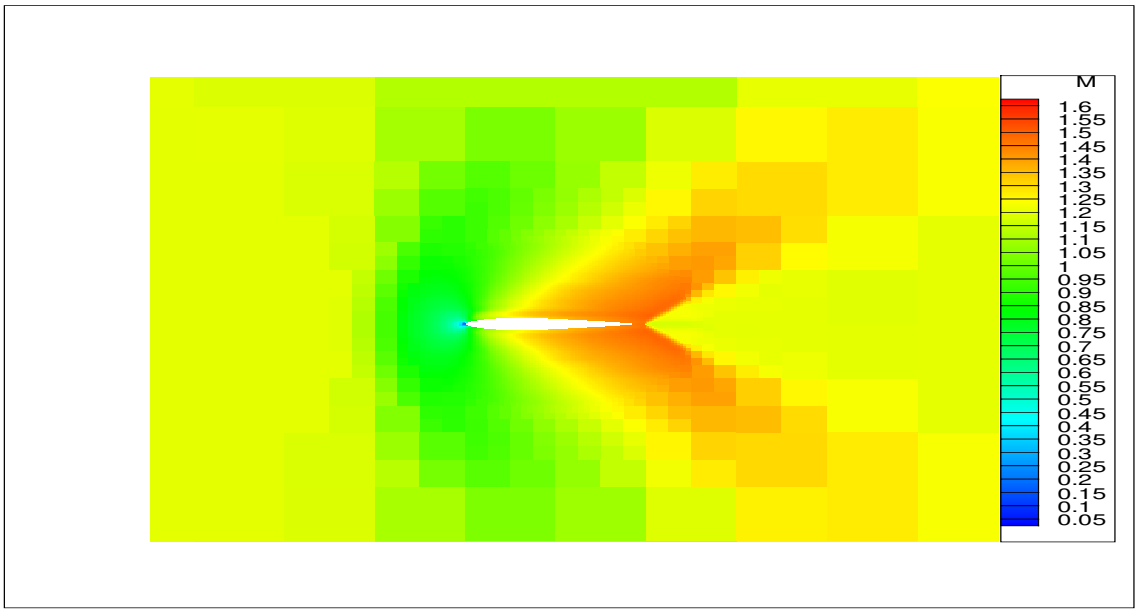

Figure 5.15: MACH number contours without solution -based adaptation, AGARD04.

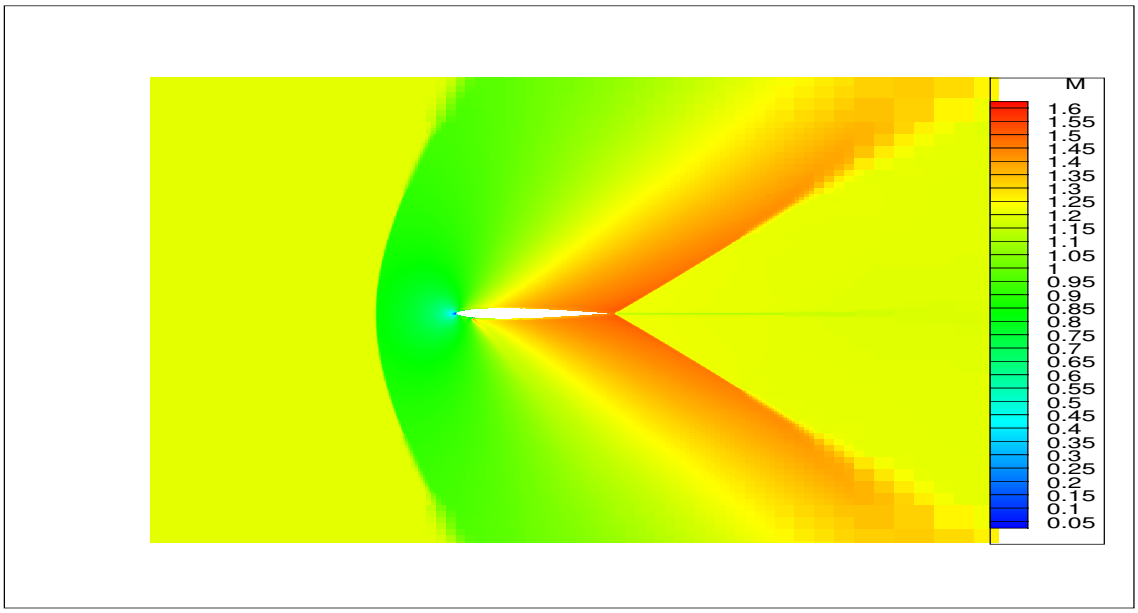

Figure 5.16: MACH number contours with solution -based adaptation, AGARD04.

agreement with that of PULLIAM and BARTON [54].

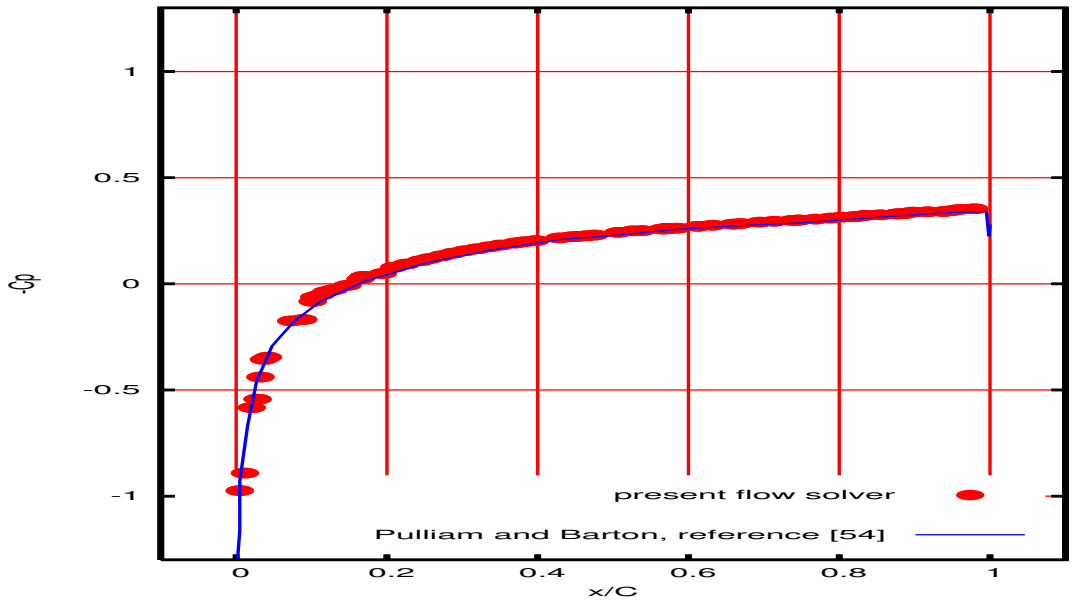

Figure 5.17: C_p distribution, AGARD04.

5.1.2 BAC3 Aerofoil

The BAC 3-11/RES/30/21 (Report AGARD-AR-303) is a research aerofoil that was originally designed to provide a high lift at small angles of attack. The subsonic flow around this aerofoil is presented here to validate the grid generator and flow solver. The unique and very distinguished geometry of this aerofoil serves as an indicator of the capability of the present grid generator since as seen in figure (5.18) the grid generated represents with high resolution the geometrical configuration of this aerofoil. In order to have a comparison reference, the *MSES* program [45] was

run with the same freestream conditions of the present flow solver; that is, with a MACH number equal to 0.3 and an α equal to 0°. This test was carried out to demonstrate the capability of the present flow solver of solving purely -subsonic flows. The pressure contours for this case are shown in figure (5.19). The lower pressure on the upper side of the aerofoil compared to that on the lower side at an angle of attack $\alpha = 0°$ proves the ability of this aerofoil to provide lift at zero angle of attack. The MACH number contours are shown in figure (5.20). The distribution of pressure coefficient C_p on the surface of the aerofoil is given in figure (5.21). As seen in the figure, the result obtained by the present solver is in good agreement with that of the *MSES* program.

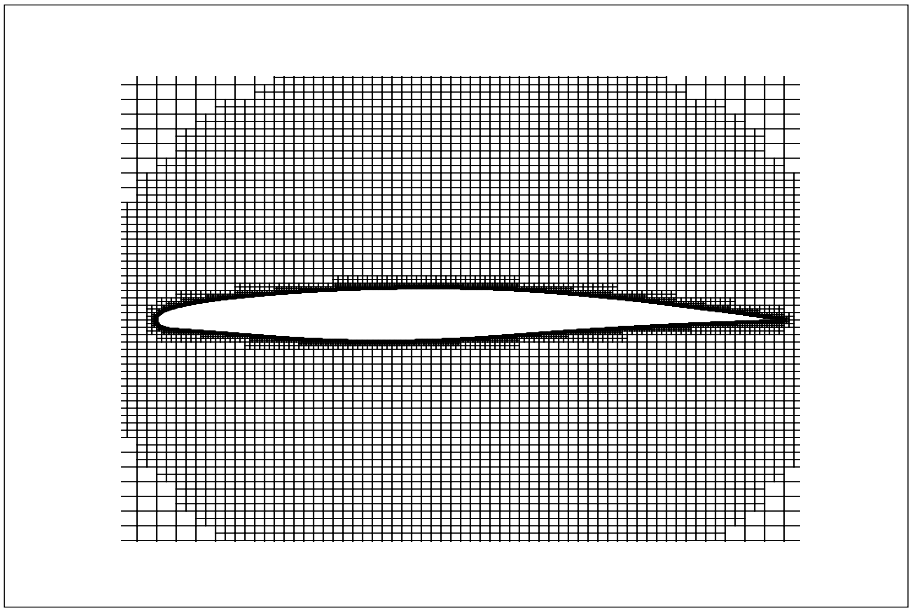

Figure 5.18: Initial grid with geometry -based grid adaptation for BAC3 aerofoil.

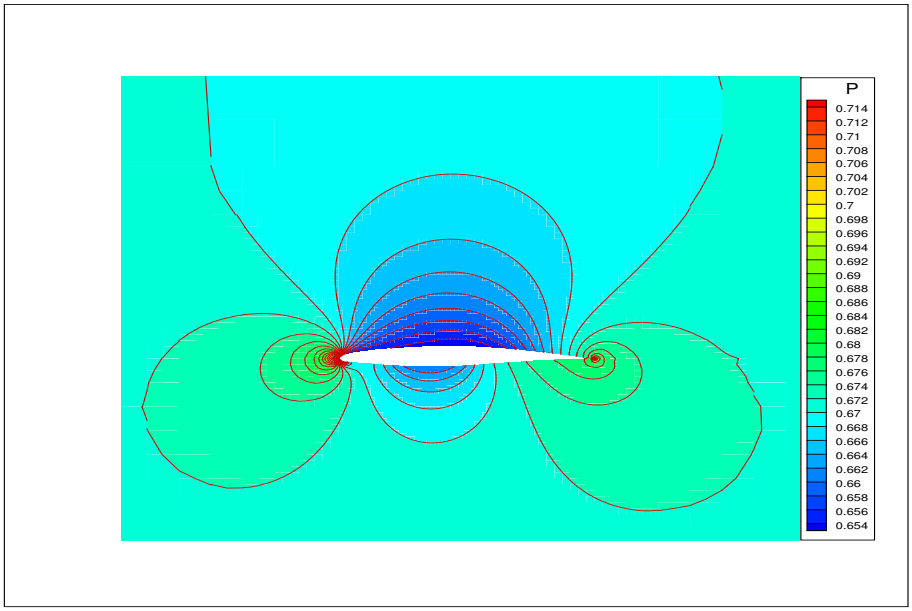

Figure 5.19: Pressure contours, BAC3 aerofoil.

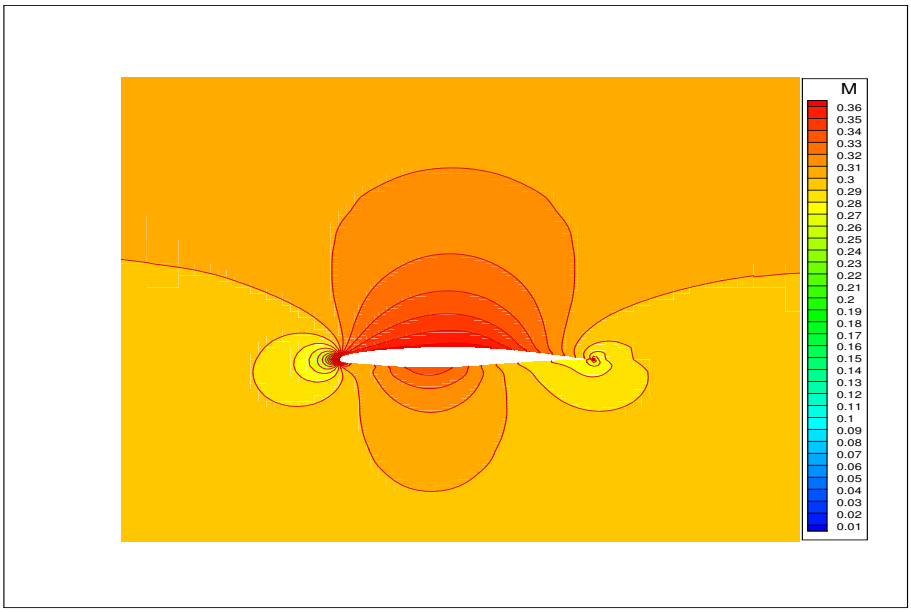

Figure 5.20: MACH number contours, BAC3 aerofoil.

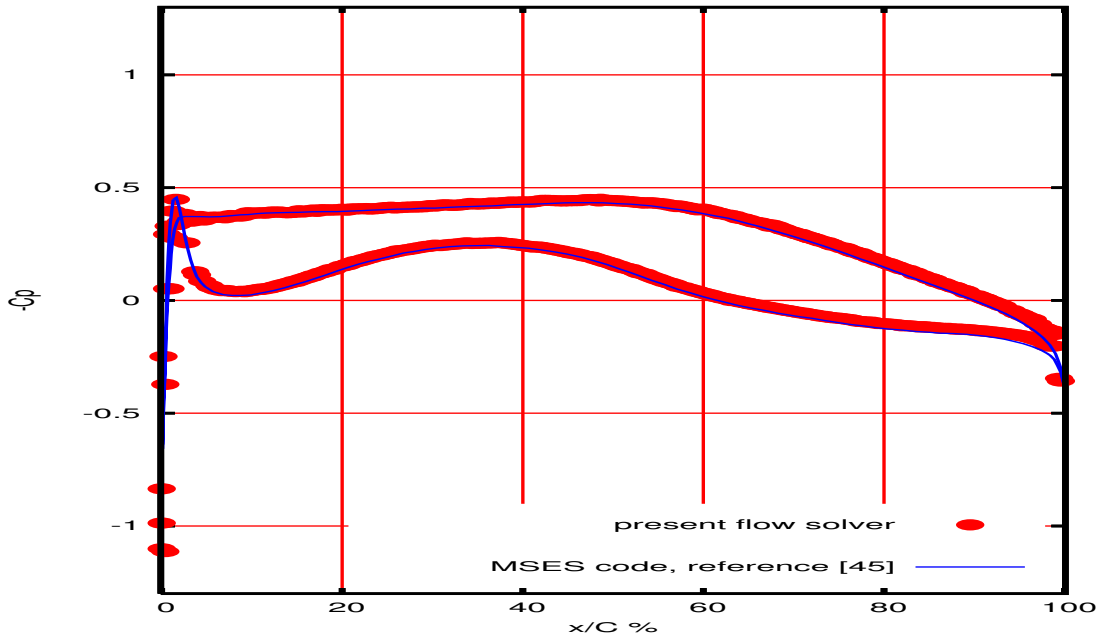

Figure 5.21: C_p distribution, BAC3 Aerofoil.

5.2 Viscous Flow

The boundary conditions at the fluid -body interface and at the far field boundary that were used for the viscous flow calculations are given by equations (3.24,3.25) and equations (3.26-3.29), respectively.

5.2.1 Flat Plate

The laminar flow past a flat plate at zero incidence (with zero pressure gradient) for incompressible flow was solved analytically by BLASIUS in 1908 [10]. It should be noted here that BLASIUS introduced a single composite dimensionless variable $\eta = y\sqrt{\frac{U}{\nu x}}$ so that the partial differential momentum equation[1] could be transformed to an ordinary differential equation of the third order. The two velocity components u and v are shown in figures (5.23, 5.24) respectively, and the grid is shown in figure (5.22). The calculation was performed assuming a MACH number of 0.2 and a REYNOLDS number of $1.5 * 10^4$.

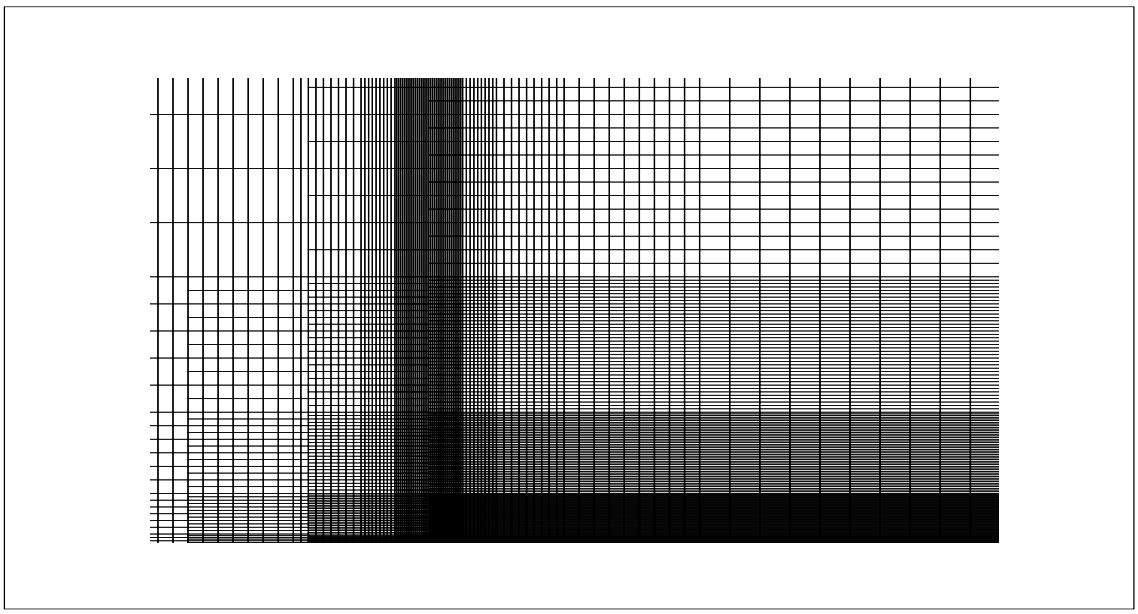

Figure 5.22: The grid at the leading edge of the flat plate.

Examining the two figures, it is noticed that the velocity component in the main

[1]This is the equation of the momentum in the direction parallel to the flat plate. The momentum equation in the direction perpendicular to the flat plate was dropped out during the derivation of the boundary layer equations.

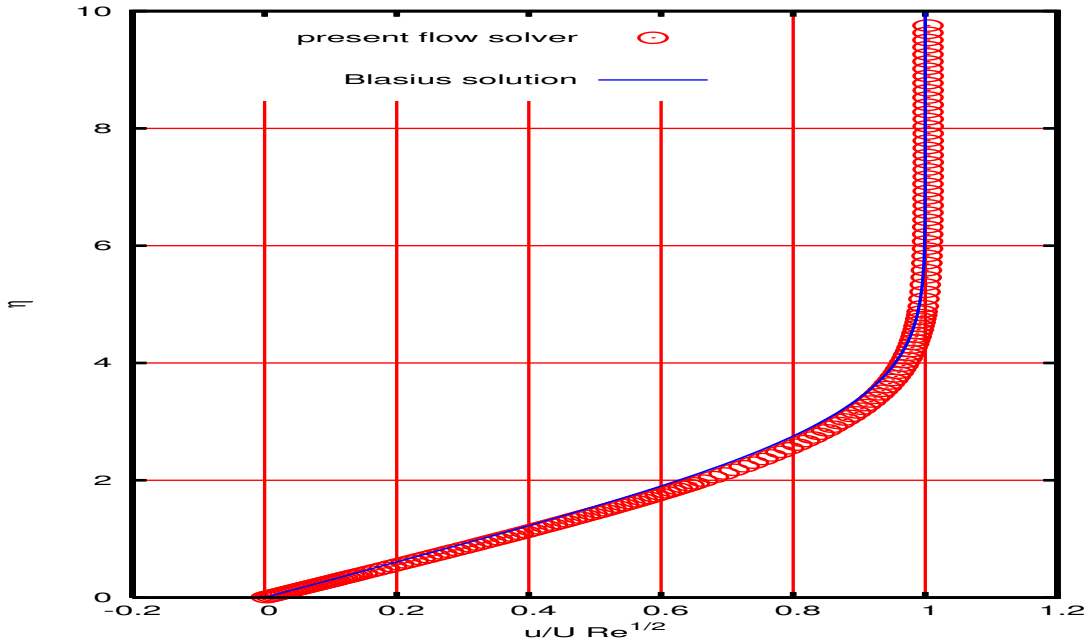

Figure 5.23: Distribution of the velocity component u.

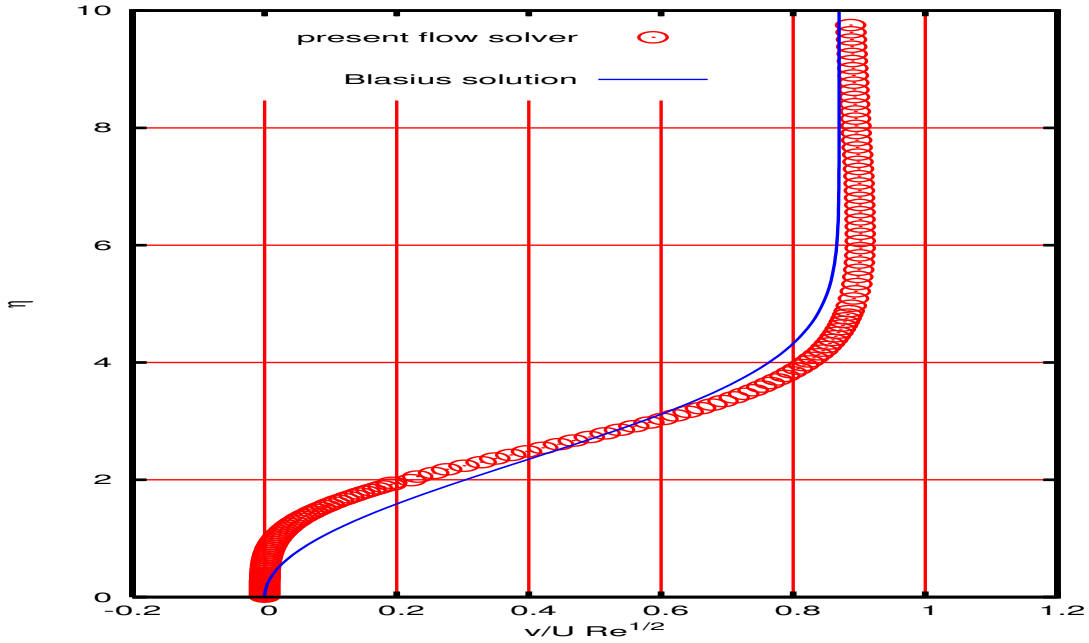

Figure 5.24: Distribution of the velocity component v.

flow direction u is in good agreement with that of the BLASIUS solution. However, the vertical velocity component v which is in the order of O ($Re^{-1/2}$) has some deviations from the solution of BLASIUS . This can be explained by the simplifications of the boundary layer equations compared to the NAVIER-STOKES equations. This leads to the necessity of only one boundary condition on v at one y position in the boundary layer equations; that is, the only condition imposed on the v component is the no slip boundary condition. Furthermore, it has to be emphasized that the fluid -surface boundary formulation is only first order accurate. That is, it possesses quite a diffusive impact on the solution right by the wall.

5.2.2 Lid -Driven Cavity Flow

The computation of the laminar lid -driven cavity flow was carried out on a uniform cartesian grid of 3844 cells ($62X62$). Since the results published by GHIA et al. [28] were obtained by solving the incompressible NAVIER-STOKES equations using an implicit multi -grid method, the MACH number was set equal to 0.1 in order to get a similar condition. The results will be shown and compared for the case of Re = 100.

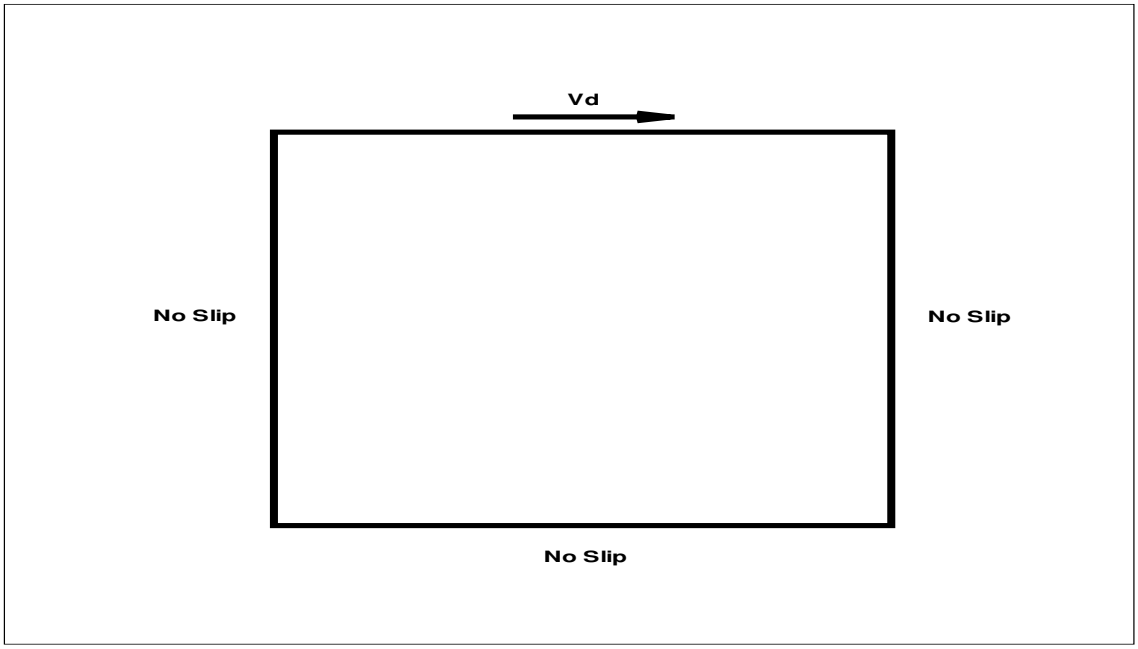

Figure 5.25: Schematic diagram of the lid-driven Cavity flow.

Due to the moving cavity lid a strong vortex is induced in the cavity that in turn induces smaller secondary vortices in the corner regions. A schematic diagram of the lid -driven cavity flow and the final grid after solution -based grid adaptation

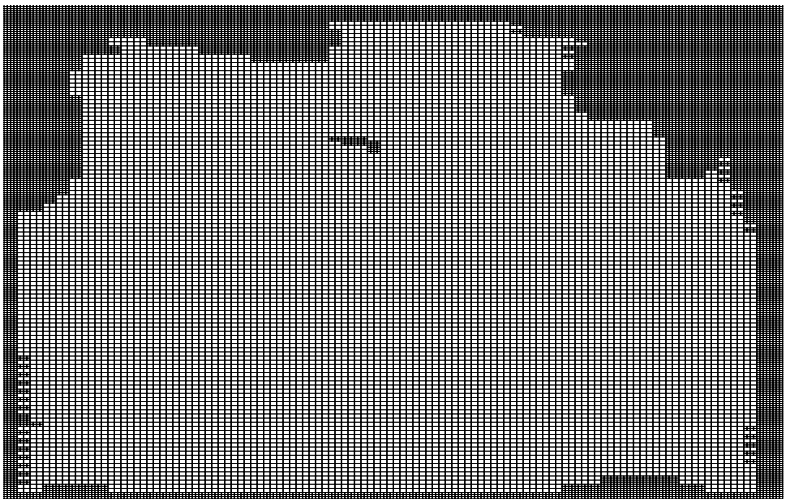

Figure 5.26: Final adapted grid, Re = 100, lid-driven Cavity.

are shown in figures (5.25, 5.26), respectively.

As seen in figure (5.26), the sensors resolve the high gradient locations at both edges of the cavity as well as at the walls of the cavity. However, the locations at the lower cavity corners where the secondary vortices do appear are not well resolved. As mentioned before, the present implementation of the fluid -surface boundary formulation is only first order accurate resulting in a higher dissipation near the walls. The effect of this high dissipation is doubled near the corners since two perpendicular walls do meet at this place. This explains the incapability of the flow solver to resolve the secondary -weak- vortices.

The distribution of the u component of the velocity vector along a vertical line at the cavity center is shown in figure (5.27) and the distribution of the v component of the velocity vector along a horizontal line at the cavity center is shown in figure (5.28). Examining the two figures (5.27, 5.28), it is noticed that the effect of the boundary conditions at the fluid -surface interface (that are of first order) on the solution accuracy in this case is stronger than the case of the flat plate; especially, on the distribution of the y -component of the velocity vector v. This is expected because in figure (5.28) v is drawn along a horizontal line at the cavity center. The fluid -surface boundary is applied two times in this direction, once at the right wall and the next one at the left wall, see figure (5.25). Thus the greater influence on the v distribution compared to the influence on the distribution of u, where the fluid -surface boundary is applied only once at the lower wall of the cavity.

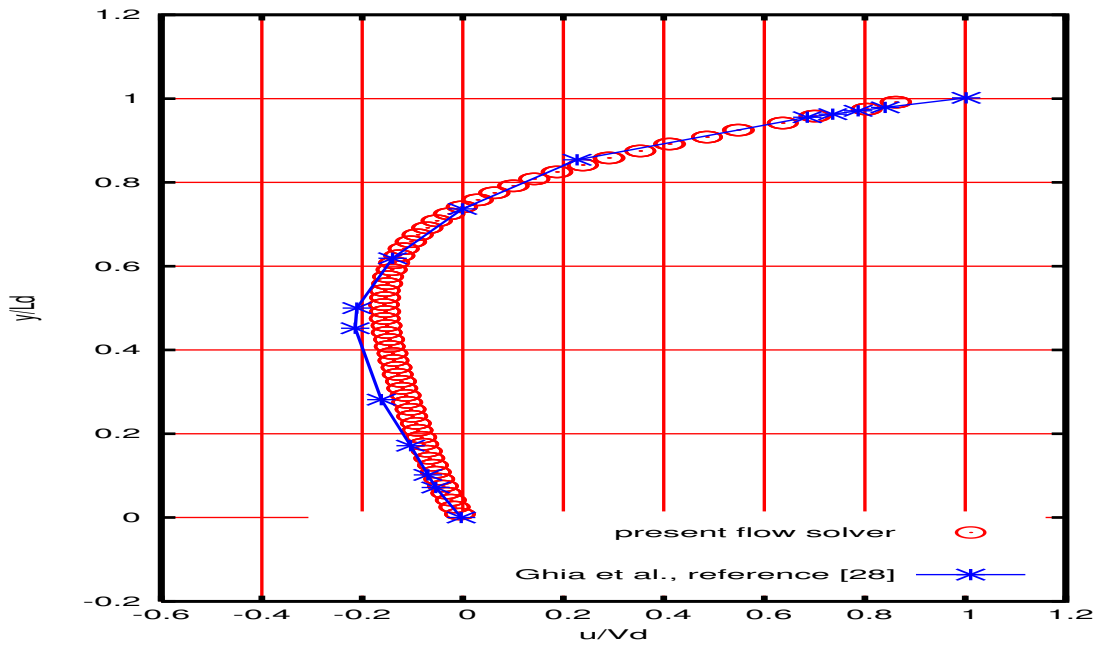

Figure 5.27: X -component of the velocity vector (u) along a vertical line at the cavitycenter.

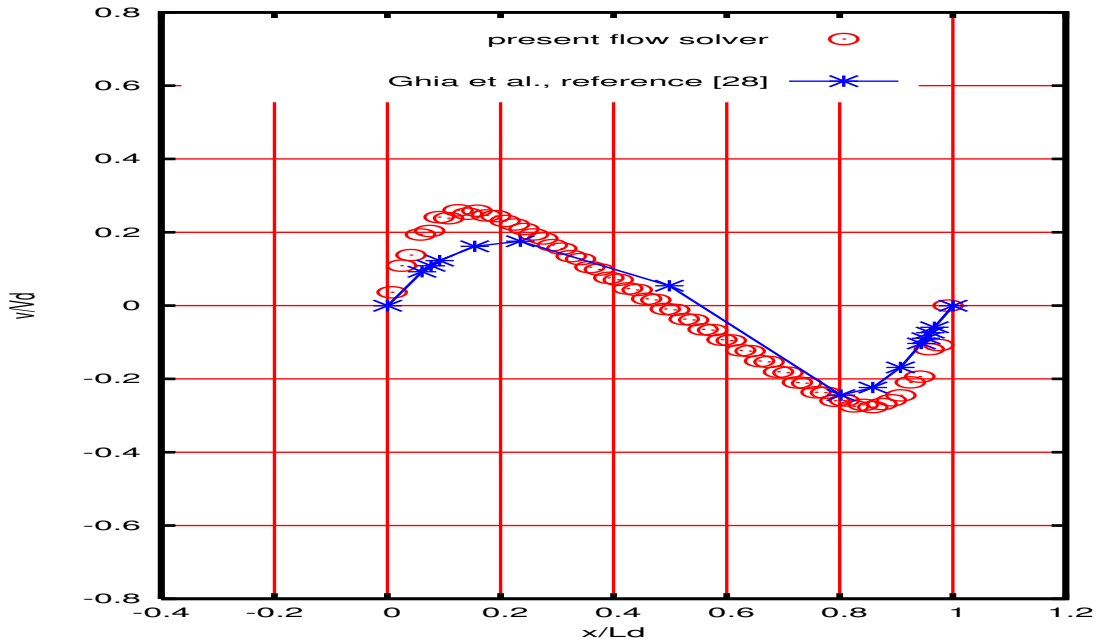

Figure 5.28: Y -component of the velocity vector (v) along a horizontal line at the cavity center.

Chapter 6

Conclusions

"We must accept finite disappointment ...
but we must never lose infinite hope."

Martin Luther King Jr.

A cartesian grid generator and a flow solver has been developed and tested for a range of aerodynamic test cases. The grid generator was designed such that the body description is given in terms of a set of data points. By using the ray casting method the grid was geometrically adapted to obtain a high resolution representation of the body geometry.

The flow solver included a high resolution shock capturing upwind scheme; namely the $AUSM^+ - au$ scheme. A linear reconstruction of the primitive variables was performed using a linear expansion and the slopes of the variables at the centers of the cells were computed by the least squares method. A solution -based grid adaptation algorithm was incorporated in the flow solver by designing a set of adaptation sensors. Each of these sensors was especially intended to capture a specific flow feature. The DAUBECHIE's wavelet function was used to compress the data (sensors) such that the grid adaptation time could be reduced by a factor of approximately 1/2.

By comparing the results of the present flow solver for some standard and non standard test cases with previously published results, it has turned out that the results of the present solver for inviscid flow are in good agreement with those of the literature. For the case of viscous flow, the results have a small deviation that is caused by the implementation of the boundary conditions at the fluid -body interface. The present cartesian grid method might produce better results for viscous

flow calculations if a higher number of boundary cells is produced[1].

The present grid generator has to be extended to be capable to read and treat other forms of data that describe a given geometry such as *CAD* data. The flow solver must be generalized for turbulent flows and furthermore, a transition -position prediction algorithm that can predict the transition position for high REYNOLDS number flows. Another possibility to extend and upgrade the present work is to apply an adaptive -RIEMANN -problem solver algorithm that employs a simple and efficient RIEMANN -problem solver in flow regions with low gradients; as a primary suggestion, the TORO scheme [66] can be employed as a fast and computaionally efficient scheme.

In addition, the grid generator and the flow solver has to be written for three -dimensional problems. However, modifying the code for parallel computing machines might be a prerequisite for the three dimensional problems since the memory requirements will be extreme by the currently available computers and the corresponding computing time will be considerably high without the use of parallel computing machines.

[1] After personal cummunications with Professor Kazuhiro Nakahashi of the Tohoku University-Japan, who has employed a cartesian solver to solve the viscous flow around the NACA0012, he mentioned that in order to solve the viscous flow even in 2D, a very high number of cells is required and the solution was carried out by using parallel computing machines.

Bibliography

[1] Aftosmis, M. J. *"Upwind Method for Simulation of Viscous Flow on Adaptively Refined Meshes."* AIAA J., 32(2)268-277,(1994).

[2] Aftosmis, M. J. *"Solution Adaptive Cartesian Grid Method for Aerodynamic Flows with Complex Geometries."* Von Karman Institute for Fluid Dynamics, Lecture Series 1997-02.

[3] Aftosmis, M. J., Berger, M., and Melton, J. *"Robust and Efficient Cartesian Mesh Generation for Component -Based Geometry."* AIAA paper 97-0196.

[4] Bartels, R., Beaty, J., Barsky, B. *"An Introduction to Splines for Use in Computer Graphics and Geometric Modeling."* Elsevier Publishing Company, ISBN: 1-55860-400-6, (1987).

[5] Batina, J. T. *"A Gridless Euler/ Navier-Stokes Solution Algorithm for Complex-Aircraft Applications."* AIAA paper 93-0333, (1993).

[6] Beam, R. M. and Warming, R.F. *"An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form."* J. Comp. Phys. Vol. 22, 87-109, (1976).

[7] Beam, R. M. and Warming, R.F. *"An Implicit Factored Scheme for the Compressible Navier-Stokes Equations."* AIAA Journal 16, 393-402, (1978).

[8] Berger, M. J. and Colella, P. *"Local Adaptive Mesh Refinement for Shock Hydrodynamics."* J. Comp. Phys. 82, 64-84 (1989).

[9] Berger, M. J. and Oliger, J. *"Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations"* J. Comp. Phys. 53, 484-512 (1984).

[10] Blasius, H. *Grenzschichten in Flüssigkeiten mit kleiner Reibung"* Z. Math. Physik Vol. 56 pp 1-37 (1908).

[11] Clarke, D., Salas, M., and Hassan. H. *"Euler Calculations for Multi-Element Airofoils Using Cartesian Grids."* AIAA Jol., **24**, (1986).

[12] Coirier, W. J. *"An Adaptively -Refined, Cartesian, Cell -Based Scheme for the Euler Equations."* NASA TM-106754, (1994).

[13] Coirier, W. J. *"An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations"* Ph. D. Thesis, the university of Michigan, (1994).

[14] Courant, R., Issacson, E. and Reeves, M. *"On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences."*, Comm. Pure and Applied Mathematics,5, 243-255.

[15] Daubechies, I. *"Orthogonal Bases of Compactly Supported Wavelets."* Commun. Pure Appl. Math. 41(7), 909-996 (1988).

[16] De Zeeuw, D. and Powell, K. G. *"An Adaptively Refined Cartesian Mesh Solver for the Euler Equations."* J. Comp. Phys., 104, 56-68 (1993).

[17] Djaffar A., Guido, B., Habashi, W., Fortin, M., Dompierre, J., and Vallet, M. *"Anisotropic Mesh Adaptation: Towards User-Independent, Mesh-Independent and Solver-Independent CFD. Part II Structured Grids."* Int. J, Numer. Meth. Fluids 2002, 39:657-673, (2002).

[18] Eiseman, P. R. *"Coordinate Generation with Precise Controls over Mesh Properties."* J. Comp. Phys. Vol. 47, (1982).

[19] Eiseman, P. R. *"Adaptive Grid Generation."* Computer Methods in Applied Mechanics and Engineering, volume 64, (1987).

[20] Fares, E., Meinke, M., Schröder, W. *"Numerical Simulation of the Interaction of Wingtip Vortices and Engine Jets in the Near Field."* AIAA paper 2000-2222, (2000).

[21] Ferziger, J. H. and Peric, M."Computational Methods for Fluid Mechanics." Third Edition, Springer Verlag Berlin Heidelberg, (2002).

[22] Finkel, R. A. and Bentley, J. L. *"Quadtrees: A Data Structure for Retrieval on Composite Keys."* Acta Informatica 4(1):1-9, (1974).

[23] Foley, J., van Dam, A., Feiner, S., Hoghes, J. *"Computer Graphics: Principles and Practice."* Addison -Wesley, Reading, MA, (1995).

[24] Forrer, H., *"Second Order Accurate Boundary Treatment for Cartesian Grid."* Research Report No. 96-13 Eidgenössische Technische Hochschule (1996).

[25] Franke, R. *"Scattered Data Interpolation: Tests of Some Methods."* Math. Comput. Vol. 38, 181-200 (1982).

[26] Frink, N. T., Parikh, P., and Pirzaadeh, S. *"A Fast Upwind Solver for the EULER Equations on Three-Dimensional Unstructured Meshes."* AIAA paper 91-0102 (1991).

[27] Gaffney, R., Hassan, H., and Salas, M. *"EULER Calculations for Wings Using Cartesian Grids."* AIAA paper 87-0356, (1987).

[28] Ghia, U., Ghia, K.N. and Shin, C.T. *"High-Re Solutions for Incompressible Flow Using the NAVIER-STOKES Equations and a Multi-grid Method."*, J. Comp. Phys., Vol.48, 1982, p.p. 387-441.

[29] Godunov, S. K.*"A Difference Scheme for Numerical Computation of Discontinuous Solution of Hydrodynamic Equations."* Math. Sbornik, 47, 271-306(in Russian), Traslated US Joint Pub.Res. Service, JPRS 7226(1969).

[30] Ham, F. E., Lien, F. S., Strong, A.B. *"A Cartesian Grid Method with Transient Anisotropic Adaptation."* J. Comp. Phys. 179, 469-494, (2002).

[31] Hirsch, C. *"Numerical Computation of Internal and External Flows."*, Vol. 2, John Wiley and Sons, (1992)

[32] Jameson, A.*"Iterative Solution of Transonic Flow over Airofoils and Wings, Including Flows at Mach 1."* Comm. Pure App. Math, Vol. 27, p.p. 283-309, (1974).

[33] Jameson, A., Schmidt, W., Turkel, E. *"Numerical Solution of the EULER Equations by Finite Volume Methods Using RUNGE-KUTTA Time-Stepping Schemes."* AIAA 1981-1259, (1981).

[34] Josuttis, N. M. *"The C++ Standard Library A Tutorial and Reference."* Addison Wesley Longman Inc. 4^{th} printing (2000).

[35] Kalitzin, G. and Iaccarino, G. *"Toward Immersed Boundary Simulation of High REYNOLDS Number Flows."* Annual Res. Briefs, Center for Turbulence Research, (2003).

[36] Karman, S. L. Jr. *"Splitflow: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries."* AIAA 95-0343, (1995).

[37] Lahur, P. R. and Nakamura, Y. *"Anisotropic Cartesian Grid Adaptation."* AIAA paper 2000-2243 (2000).

[38] Lax, P.D. and Wendroff, B. *"Systems of Conservation Laws."* Comm. Pure and Applied Mathematics, Vol. 13, pp 217-237, (1960).

[39] Lax, P.D. and Wendroff, B. *"Difference Schemes for Hyperbolic Equations with High Order of Accuracy."* Comm. Pure and Applied Mathematics, Vol. 17, pp 381-398, (1964).

[40] LeVeque, R. J. and Berger, M. J. *"A Rotated Difference Scheme for Cartesian Grids in Complex Geometries."* AIAA paper 91-1602 (1991).

[41] Liou, M. S. and Steffen, C. J. *"A New Flux Splitting Scheme."* J. Comp. Phys. 107, 23-39, (1993).

[42] Liou, M. S. *"A Sequel to AUSM: AUSM$^+$."* J. Comp. Phys. 129 364-382 (1996).

[43] Liou, M. S. *"Ten Years in the Making-AUSM-Family."* AIAA paper 2001-2521, (2001).

[44] Liu, J. L. and Su S. J. *"A Potential Gridless Solution Method for the Compressible EULER/NAVIER-STOKES Equations."* AIAA paper 96-0526, (1996).

[45] Drela, M. *"MSES: A multielement airfoil design analysis system."* A Research Program by: MIT Computational Aerospace Sciences Laboratory, (1996).

[46] Löhner, R. and Parikh, P. *"Generation of Three -Dimensional Unstructured Grids by the Advancing -Front Method."* AIAA paper 88-0515, (1988).

[47] Michel, R. *Etude de la Transition sur les Profils d'aile-Establissment d'un Point de Transition et calcul de la Trainee de Profil en Incompressible* ONERA Report No. 1/ 1578 A, (1952).

[48] Moretti, G. *The λ-scheme.* Computers and Fluids, 7, pp 191-205, (1979).

[49] Nakahashi, K. *"An Automatic Grid Generator for the Unstructured Upwind Method."* AIAA 9th Computational Fluid Dynamics Conference, (1989).

[50] Press, W.H., Teukolosky, S. A., Vetterling, W. T., and Flannery, B.P. *"Numerical Recipes in C++, The Art of Scientific Computing."* Second Edition, Cambridge University Press, (2002).

[51] Pember R. B., Colella, P., Crutchfield, W. Y., and Welcome, M.L. *"An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions."* J. Comp. Phys. 120, 278-304 (1995).

[52] Percival, D. B. and Walden, A. T. *"Wavelet Methods for Time Series Analysis."* Cambridge, England Cambridge Univ. Press. (2000).

[53] Peskin, C.S. *"Flow Patterns Around Heart Valves: A Digital Computer Method for Solving the Equations of Motions."* Ph.D.Thesis, Albert Einstein College of Medicine (1972).

[54] Pulliam, T., and Barton, J. *"EULER Computation of AGARD Working group 07 Test Cases."* AIAA-85-0018 (1985).

[55] Peskin, C.S. *"Numerical Analysis of Blood Flow in the Heart."* J. Comp. Phys. Vol. 25, pp 220-252 (1977).

[56] Peskin, C.S. and Printz, B. *"Improved Volume Conservation in the Computation of Flows with Immeresed Elastic Boundaries."* J. Comp. Phys. Vol. 105, pp 33-46 (1993).

[57] Poinsot, T. J. and Lele, S. K. *"Boundary Conditions for Direct Simulations of Compressible Viscous Flows."* J. Comp. Phys., Vol. 101, pp104-129, (1992).

[58] Quirk, J. *"An Alternative to Unstructured Grids for Computing Gas Dynamics Flows around Arbitrarily Complex Two Dimensional Bodies."* Computers Fluids Vol. 23 No. 1, pp 125-142 (1994).

[59] Fares, Ehab *"Numerical Simulation of the Interaction of Wingtip Vortices and Engine Jets in the Near Field"* Ph.D. Disseratation, Aerodynamisches Institut Aachen -RWTH AACHEN, 2002.

[60] Opela, Mario *"Grobstruktur-Simulation der Interaktion des Nachlaufs eines bewegten Kreiszylinders mit einer Turbinenschaufel"* Ph.D. Disseratation, Aerodynamisches Institut Aachen -RWTH AACHEN, 2003.

[61] Meinke, Matthias *"Numerische Lösung der Navier-Stokes-Gleichungen für instatonäre Strömungen mit Hilfe der Mehrgittermethode"* Ph.D. Dissertation, Aerodynamisches Institut Aachen -RWTH AACHEN, 1993.

[62] Raithby, G. D. *"Skew Upstream Differencing Schemes for Problems Involving Fluid Flow."* Comp. Methods Appl. Mech. Eng. Vol. 9, pp153-164, (1976).

[63] Roe, P. L. *"Approximate Riemann solvers, Parameter Vectors and Difference Schemes."* J. Comp. Phys. 43, 357 (1981).

[64] Samet, H. *"The Design and Analysis of Spatial Data Structures."* Addison-Wesley Series on Computer Sience and Information Processing. Addison-Wesley Publishing Company, (1990).

[65] Thompson, K. W. *"Time Dependent Boundary Conditions for Hyperbolic Systems."* J. Comp. Phys. Vol 89, pp 439-461, (1990).

[66] Toro, E.F. *"A linearised Riemann solver for the time -dependent Euler equations of gas dynamics."* Proc. R. Soc. Lond. A 434, 683 (1991).

[67] Van Leer, B. *Flux Vector Splitting for the EULER equations.* Lecture Notes in Physics Vol. 170 pp507-512, (1982).

[68] Wang, J. Z., Wiederhold, G., Firschein, O., and Wei, S. X. *"Content-Based Image Indexing and Searching Using Daubechies' Wavelets."* Int. J. of Digit. Lib. 1, 311-328 (1997).

[69] 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 paper 99-3300 (1999).

[70] Warren, G.P., Anderson, W.K., Thomas, J.L., and Krist, S.L. *"Grid Convergence for Adaptive Methods."* AIAA paper 91-1592-cp, (1991).

[71] White, F. M. *"Viscous Fluid Flow."*, Second Edition, McGrawHill Inc. (1991).

[72] Wylen, G.V. and Sonntag, R.E.*"Fundamentals of Classical Thermodynamics."* Second Edition, John Wiley and Sons, Inc.(1978).

Appendix A

Reference variables

In order to normalize the governing equations, reference quantities are chosen as:

$$x = \frac{x^*}{L^*} \quad y = \frac{y^*}{L^*} \quad t = \frac{t^*}{L^*/a_o^*} \quad T = \frac{T^*}{T_o^*} \quad u = \frac{u^*}{a_o^*} \quad v = \frac{v^*}{a_o^*} \quad w = \frac{w^*}{a_o^*}$$

$$\rho = \frac{\rho^*}{\rho_o^*} \quad p = \frac{p^*}{\rho_o^* a_o^{*2}} \quad e = \frac{e^*}{a_o^*} \quad \mu = \frac{\mu^*}{\mu_o^*} \quad k = \frac{k^*}{k_o^*} \quad c_p = \frac{c_p^*}{c_{po}^*} \quad c_v = \frac{c_v^*}{c_{vo}^*}$$

Substituting in the governing equations, the following dimensionless numbers are obtained:

- REYNOLDS number: $Re_o = \frac{\rho_o^* a_o^* L^*}{\mu_o^*}$
- PRANDTL number: $Pr_o = \frac{\mu_o^* c_{po}^*}{k_o^*}$
- Ratio of specific heats: $\gamma_o = \frac{c_{po}^*}{c_{vo}^*}$

A constant specific heat ratio is assumed and taken for air $\gamma_o = 1.4$. The REYNOLDS number appearing in equation (3.5 -3.7) is the REYNOLDS number based on the stagnation conditions. In order to calculate this from the REYNOLDS number that is based on the free stream conditions; the following relation is derived:

$$Re_o = \frac{Re * T_\infty^{0.72}}{\rho_\infty M_\infty \sqrt{T_\infty}} \tag{A.1}$$

Appendix B

LSM applied to three dimensions

The assumed equation of the linear distribution of a variable can be written as:

$$f(x,y,z) = f_o + \frac{\partial f}{\partial x}\, x + \frac{\partial f}{\partial y}\, y + \frac{\partial f}{\partial z}\, z \tag{B.1}$$

Writing equation (B.1) for the cell considered and all the direct -face neighbours and using a least squares fit approach, the following system of algebriac equations is obtained:

$$\begin{pmatrix} n & \sum x_i & \sum y_i & \sum z_i \\ \sum x_i & \sum {x_i}^2 & \sum x_i y_i & \sum x_i z_i \\ \sum y_i & \sum x_i y_i & \sum {y_i}^2 & \sum y_i z_i \\ \sum z_i & \sum x_i z_i & \sum y_i z_i & \sum {z_i}^2 \end{pmatrix} \cdot \begin{pmatrix} f_o \\ \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix} = \begin{pmatrix} \sum f_i \\ \sum f_i x_i \\ \sum f_i y_i \\ \sum f_i z_i \end{pmatrix} \tag{B.2}$$

The above system of equations can be solved by any standard method for the values of the variable gradients $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, and $\frac{\partial f}{\partial z}$.

Appendix C

AUSM$^+$ – au scheme

The $AUSM^+ - au$ scheme uses the numerical speed of sound $a_{1/2}$ that can be written as:

$$a_{1/2} = \min\left(\tilde{a}_L, \tilde{a}_R\right) \tag{C.1}$$

where the subscript $_L$ and $_R$ refers to the values at the left and right sides of the face, respectively. $\tilde{a}$ is taken as $\tilde{a} = \frac{a^{*2}}{\max(a^{*2},|u|)}$ and the critical speed of sound is calculated from $a^{*2} = \frac{2}{\gamma+1}a_t^2$ and a_t is the speed of sound based on the total enthalpy. Liou [42] has given other formulae to calculate the numerical speed of sound; namely,

$$a_{1/2} = \frac{1}{2}\left(a_L + a_R\right) \tag{C.2}$$

and

$$a_{1/2} = \sqrt{a_L a_R} \tag{C.3}$$

Like in the original $AUSM$ scheme, the inviscid flux is split into convective and pressure terms; that is,

$$\vec{F}_{inv} = a_{1/2} \begin{pmatrix} M_{1/2}^{p/m}\rho \\ M_{1/2}^{p/m}\rho u \\ M_{1/2}^{p/m}\rho v \\ M_{1/2}^{p/m}\rho H \end{pmatrix}_{L/R} + \begin{pmatrix} 0 \\ 0 \\ p_L^+ + p_R^- - D_p \\ 0 \end{pmatrix} \tag{C.4}$$

In equation (C.4) the interface split MACH numbers $M_{1/2}^{p/m}$ are given by:

$$\begin{aligned} M_{1/2}^{p} &= \tfrac{1}{2}\left(M_{1/2} + |M_{1/2}|\right) \\ M_{1/2}^{m} &= \tfrac{1}{2}\left(M_{1/2} - |M_{1/2}|\right) \end{aligned} \tag{C.5}$$

The upwinding is achieved through the interface MACH number; that is,

$$(.)_{L/R} = \begin{cases} (.)_L & \text{if } M_{1/2} \geq 0; \\ (.)_R & \text{if } M_{1/2} < 0. \end{cases} \tag{C.6}$$

where the interface MACH number $M_{1/2}$ is computed from

$$M_{1/2} = M^+(M_L) + M^-(M_R) \tag{C.7}$$

In defining the split MACH numbers $M^+(M_L)$ and $M^-(M_R)$, LIOU has chosen this time a fourth order polynomial for the subsonic state and retained the first order polynomial without change for supersonic state, thus only the subsonic polynomial will be written:

$$M_{L/R}^{\pm} = M_{(2)}^{\pm}\left(1 \mp 2M_{(2)}^{\mp}\right) \tag{C.8}$$

and the second order polynomial, $M_{(2)}^{\pm}$, has the form:

$$M_{(2)}^{\pm} = \pm\frac{1}{4}\left(M \pm 1\right)^2 \tag{C.9}$$

The pressure diffusion term D_p is given by:

$$D_p = p^+ p^- \rho_{1/2} a_{1/2}^2 \left(M_R - M_L\right) \tag{C.10}$$

In computing the split pressure $p^\pm$ for subsonic state, LIOU has switched to a fifth order polynomial of the form:

$$p^\pm = M_{(2)}^\pm \left[(\pm 2 - M) \mp 3 M M_{(2)}^\mp\right] \tag{C.11}$$

In equation (C.10) the interface density is computed from:

$$\rho_{1/2} = \frac{(\rho_L + \rho_R)}{2\sqrt{\rho_L \rho_R}} \tag{C.12}$$

Appendix D

Cell classification: Three dimensional bodies

Considering a three dimensional body as shown in figure (D.1), where the geometrical description of the body is provided by a set of data points given at several x -planes. The first step in the grid generation process is to generate a set of trian-

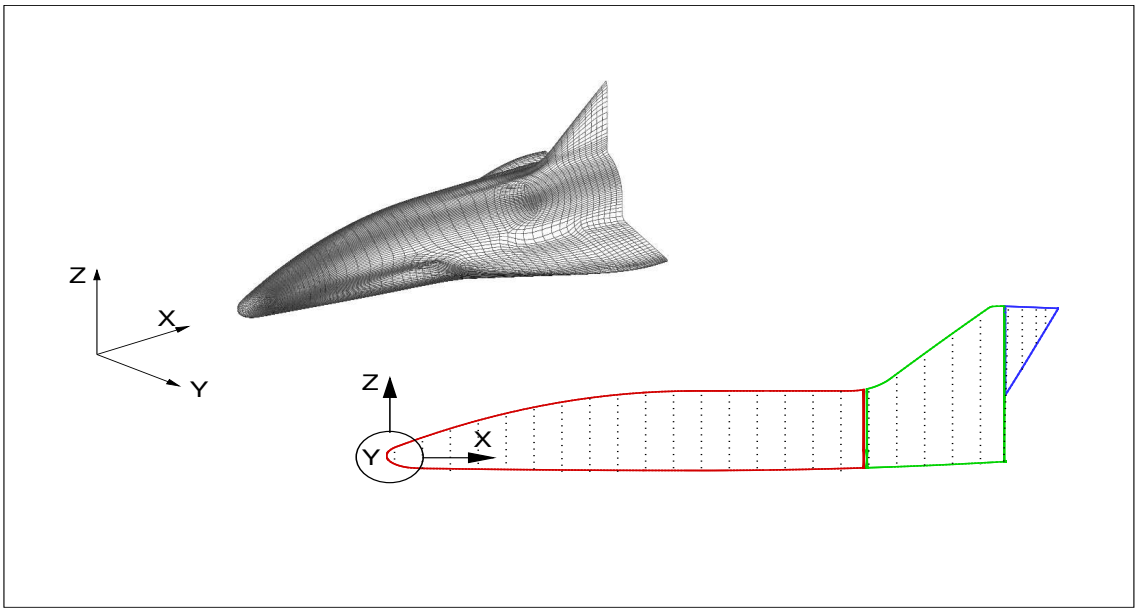

Figure D.1: A three dimensional body represented by a set of data points given at several x -planes.

gles that cover the whole geometry given. This will be followed by performing a loop on all the cells of the computational domain where at each cell another loop is

performed on the twelve edge lines compromising the edges of the hexahedral cell, see figure (D.2). The body surface cuts a computational cell if one or more of the

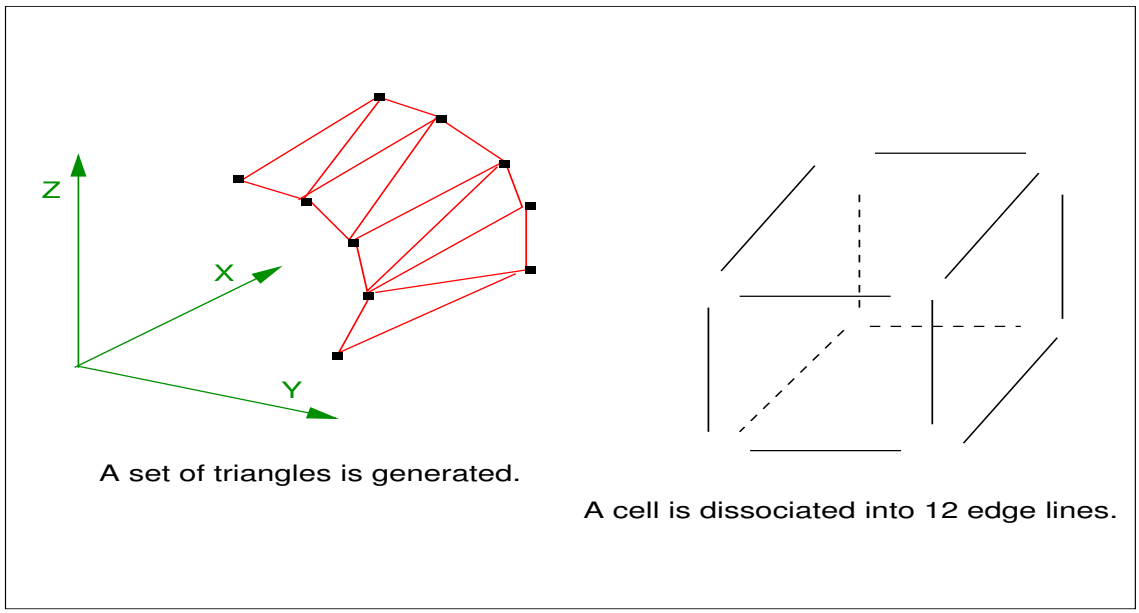

Figure D.2: A set of triangles are generated that cover the whole surface of the body, and the twelve edges of each cell are represented by twelve lines.

twelve edge lines cuts any triangle. AFTOSMIS et al. [3] have given a three -step procedure that when performed can decide wether the line pierces the triangle or not. Moreover, the procedure can specify the coordinates of the pierce point. The most practical aspect of this procedure is that if the result of the first step were negative, the rest of the steps would not be necessary to perform. The three steps are:

Examining the Possibility of Intersection:

This is the first test to be performed that requires the computation of the signed volume of the two tetrahedrons that are formed by joining the three vertices of the triangle with the two ends of the line; see figure (D.3). The signed volumes of the two tetrahedrons are given by:

$$\psi_{abce} = \begin{pmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ e_x & e_y & e_z & 1 \end{pmatrix} \tag{D.1}$$

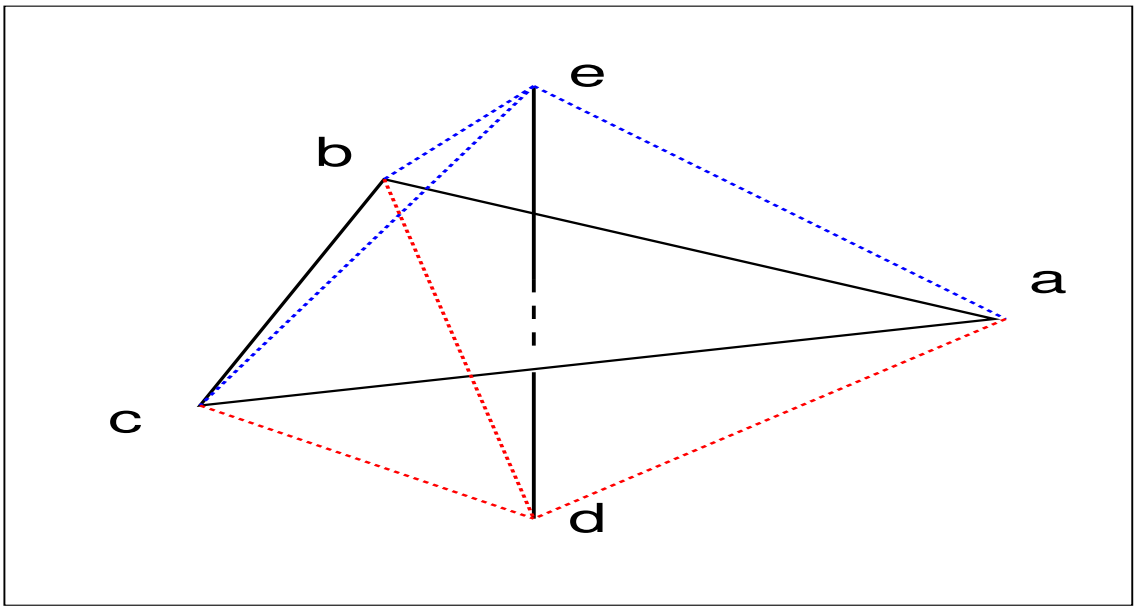

Figure D.3: The two tetrahedrons formed by joining the triangle's vertices with the two ends of the line.

$$\psi_{abcd} = \begin{pmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{pmatrix} \tag{D.2}$$

The line could intersect the triangle if and only if the two signed volumes do have *opposite signs*.

Examining the Possibility of Piercing:

The second test examines if the possibility of piercing is there. The first test might succeed for some cases that do not have the possibility of piercing; for example, consider the line $\overline{ab}$ and the triangle shown in figure (D.4), in this case the result of the first test will be positive. The signed volumes of the two tetrahedrons will have opposite signs, although the possibility of piercing does not exist. This is clear from the drawing on the left of figure (D.4) when the view is taken from another angle. In order to perform this test, three tetrahedrons are constructed this time, see figure (D.5). The line do pierces the triangle if and only if the three signed volumes of the three tetrahedrons have the same sign.

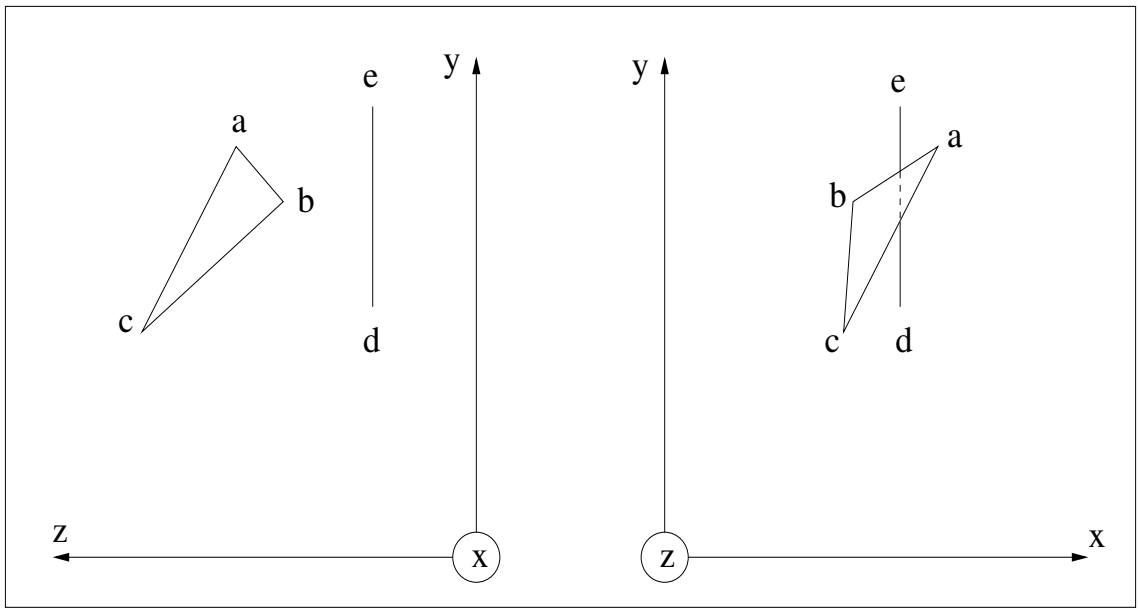

Figure D.4: A typical case where the first test successes but the second fails (not to scale).

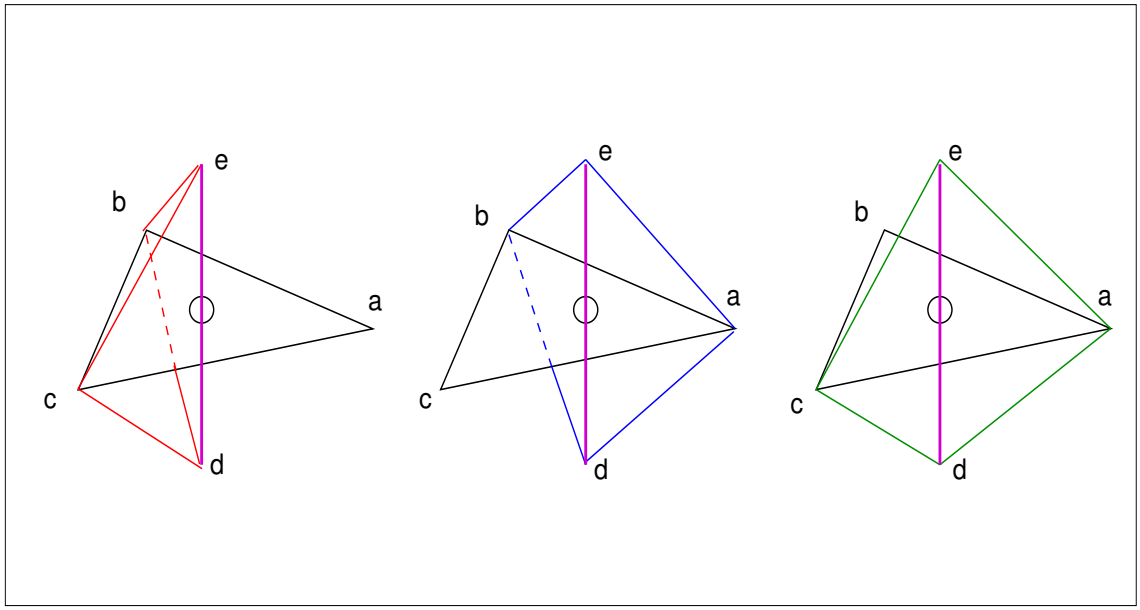

Figure D.5: Three tetrahedrons are constructed for the second test.

Finding the Coordinates of the Pierce Point:

Now the only job left is to find the coordinates of the pierce point, this can be achieved by using the parametric representation of both the line and the triangle and using the fact that at the pierce point the two pareametric equations are equal, see figure (D.6).

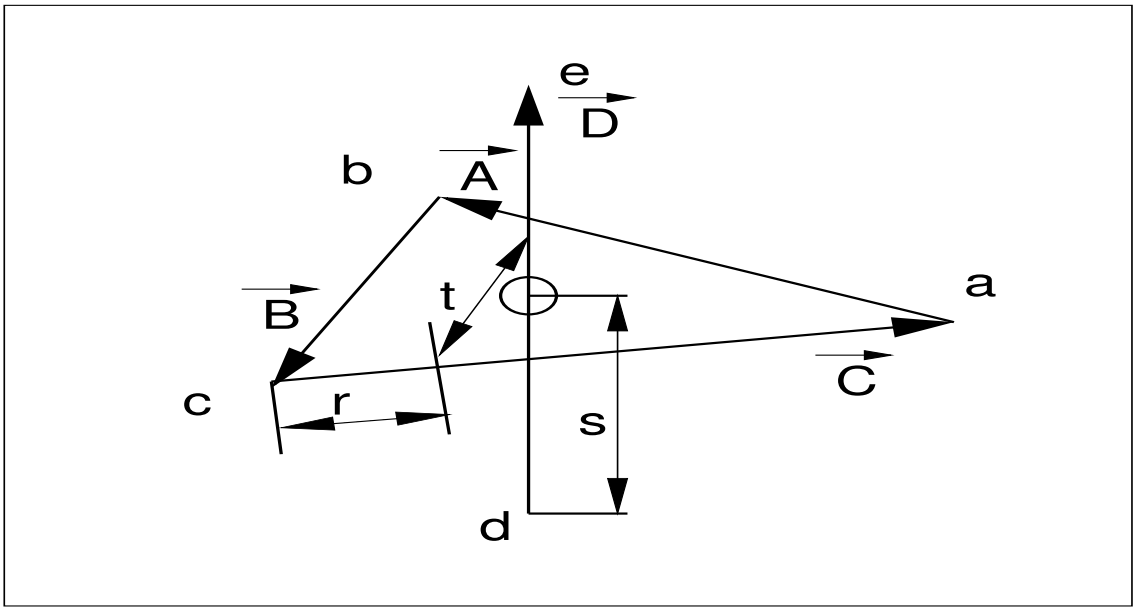

Figure D.6: Parametric representation of both a line and a triangle.

The parametric representation of the plane (triangle) is:

$$\vec{P}(r,t) = \vec{c} + r \cdot \vec{C} - t\vec{B} \tag{D.3}$$

and the parametric representation of the line is:

$$\vec{L}(s) = \vec{d} + s \cdot \vec{D} \tag{D.4}$$

At the pierce point, the following eqaution can be written and solved for the values of r,t,and$s$.

$$\vec{P}(r,t) = \vec{L}(s) \tag{D.5}$$

List of Figures

List of Tables

Lebenslauf

Name, Vorname:	*Sultan, Khalid Mohammad*
Geburtstag:	23. *August* 1965
Geburtsort:	*Benghazi – Libyen*
Familienstand:	*Verheiratet und habe drei Kinder*; *Yousef*, *Yakeen*, *und Yasid.*
1971 - 1977:	*Besuch der Grundschule in Shahid Saleh Buisier.*
1977 - 1980:	*Besuch der Vorbereitungsschule in Ubaida Iben Elgarah.*
1980 - 1983:	*Besuch der Sekundarschule in Shuhada Janayer mit Abschluss Abitur.*
1983 - 1988:	*Studium des Maschinenbaus an der Garyounis Univaersitaet in Benghazi – Libyen, mit Abschluss Bakkalaureus.*
1988 - 1990:	*Grundwehrdienst.*
1990 - 1995:	*Wissenschaftler Assistent an der Maschinenbau–abteilung der Fakultaet der Ingenieurwissenschaf–ten an der Garyounis Univeraersitaet mit Abs–chluss Magister.*
1995 - 1999:	*Wissenschaftler Angestellter an der Fakultaet der Ingenieurwissenschaften der Garyounis Univers–itaet.*
1999 - 2000:	*Goethe Institut* (*Intinsives Sprachkurs*).
2000 - 2005:	*Wissenschaftler Assistent* (*Stepindiat*) *am Aero–dynamischen Institut an der RWTH – Aachen.*